Portfolio Optimisation with Sequential Monte Carlo

Yow Tzu Lim

Submitted in partial fulfilment of the requirements for the MSc in Statistics

Imperial College, London

Department of Mathematics

August 27, 2014

Declaration

The work contained in this thesis is my own work unless otherwise stated.

I am the primary author of all work reported in this thesis. Advice on specific aspects of the work was provided by my supervisor, Dr. Nikolas Kantas.

Abstract

Constructing an optimal portfolio is the ultimate goal of every investment manager; but the optimality criteria may be very different to each of them. For index tracker fund manager, the main objective of portfolio management is to track and replicate the exposure of a benchmark index.

In this study, we view this as a stochastic control problem. We adopt the Bayesian view and treat the investment decision controls as random variables. The objective is to search for the sequence of control parameters that results in a portfolio that tracks a benchmark index in an optimal way. We investigate here the potential of using Sequential Monte Carlo (SMC) techniques as the means of determining such strategy. We examine the feasibility of this approach using two examples. The first example is a toy example with the target reference set to be oscillating oscillating wave. In the second example, a real world index — the German's DAX index level is used as the target reference. In both cases, the approach looks very promising. Lastly, we demonstrate how the Model Predictive Control (MPC) concept can be incorporated to improve tracking performance.

The thesis concludes with an evaluation on the work done, the extent to which the work justify the thesis hypothesis and some possible directions on how SMC techniques can be applied to address a wider range of relevant problems on the domain of concern.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my gratitude to all who have provided support in accomplishing this thesis. I would like to thank my supervisor, Dr. Nikolas Kantas, for the opportunity given to carry out this research and also for his guidance, expertise and encouragement during the course of this research.

I would also like to express my appreciation to my wife Ka Ki Lai for her endless support and care in my personal life during the course of this master degree study. I would like to thank God Almighty, who has given me the strength and peace to work hard on this project. I would like to express my gratitude to all who have provided support in accomplishing this thesis.

Chapter 1

Introduction

Capital allocation is a common challenge for every investor. In the investment decision making process, investors decide how much capital is allocated to different investable assets to form a portfolio that performs better than any other possible portfolios according to some criteria. These criterion can be different among investors. Some investors may also have additional considerations such as tax considerations and legal restriction on investment assets or holding periods.

Two common objectives, which often contradicting, are the financial return and the investment risk. The Markowitz's modern portfolio theory [1] proposes a portfolio selection framework. In the Markowitz's framework, it is assumed that investor attempt to maximize a portfolio's return and minimize the risk (measured by the variance of the portfolio return). Based on these criteria, the set of non-dominated portfolios, commonly known as the *efficient portfolios* of a given investment period can be found. However, using variance as a risk measure has its limitation. Variance is a symmetric measure; an out-performing asset (than the expected return) is deemed to as risky as an under-performing one. Many alternative risk measurements have been proposed, e.g. Sortino ratio, Conditional Value at Risk (CVaR), etc. Refer [2] for details.

Surprisingly, there are some investment managers who have no interest in maximising their portfolio's return. Instead, the main objective of portfolio management for such a fund is to simply track and replicate the exposure of a benchmark index as close as possible. These funds are attractive as it provides the investors the exposure to the market at low fees and taxes because only minimal active management is required. Passively tracking a benchmark index also makes the fund less vulnerable to the change of fund managers. The performance of these funds are often assessed in term of how well the fund tracks the benchmark index using some pre-defined metrics. These index tracking funds are the focus of the thesis.

1.1 Technical Approach

Traditionally, portfolio optimization have been explored in an analytical fashion, adopting necessary assumption as necessary. This seems rather restrictive; there are many instances where numerical method has been used to derive an approximate or even more effective solution to the problem in question. For example, Monte Carlo technique is used to approximate integral, heuristic search technique such as simulated annealing applied in engineering domains, etc.

Our approach to the problem in this thesis is a radical one. We view a portfolio optimisation as a *stochastic* control problem. We adopt the Bayesian view and treat these parameters as random variables. The objective is to find the sequence of control parameters that optimise the reward function defined in terms tracking error between the fund and the benchmark index. We investigate the potential of using Sequential Monte Carlo (SMC) techniques as the means of determining the optimal, or at least excellent, strategies for the portfolio optimisation problem in question. The main reason of choosing SMC techniques is its ability to carry out *sequential* update on the posterior distribution over time fit well with parameter inference in stochastic process. Moreover, some techniques have achieved significant success in their applications on many domains. Of course, other heuristic search techniques are also potentially applicable.

To investigate this approach, we first applied the technique on to track the output of a simple deterministic reference model. This model is doubly useful. It demonstrates the concept nicely and serves as a basic model to allow us to gain further understanding on the tunable parameters. We then considered the problem of tracking a real-world index with its constituent prices modelled as Brownian motion with drift. Using SMC techniques, we search for the optimal strategy (the set of control parameters at each time point) that optimise against the reward function defined in terms of minimizing the tracking error and the transaction costs involved in maintaining the positions. Lastly, we introduce the concept of Model Predictive Control (MPC) and this concept can be used here to improve the tracking performance.

1.2 Thesis hypothesis

Formally, the hypothesis of the thesis is stated as follows:

> Sequential Monte Carlo (SMC) techniques have the potential to be an effective means of searching the optimal strategy for index tracking funds.

We attempt to examine this hypothesis from three different perspectives:

1. Exploring the potential of SMC techniques in searching the optimal strategy from index tracking error with constraints and minimizing the transaction costs.
2. Exploring the sensitivity of the techniques in terms of the parameter settings, trade-off between the estimation accuracy and computational efforts numerically and providing suggestions for real-world problem.
3. Introducing the concept of model predictive control (MPC) and how it can be used together with SMC in improving tracking performance.

Given the time frame of the project, we fully understand it is impossible to evaluate our approach on full scale strategy. The aim is to establish the plausibility or, at the very least, a greater understanding of the strengths and weaknesses of the above approach.

1.3 Thesis organisation

The subsequent chapters of this thesis are organised as follows:

- Chapter 2 reviews some fundamental concept in Monte Carlo method that are related to this thesis. It begins with a brief introduction to basic methods such as perfect Monte Carlo sampling, rejection sampling, importance sampling. It then introduces the Sequential Monte Carlo (SMC) techniques, along with various enhancements proposed, e.g., MCMC move, Marginalisation, used in this thesis.
- Chapter 3 briefly review the state of the art of portfolio optimization problem. It then discusses how a portfolio problem can be transformed naturally into standard parameter estimation in Sequential Monte Carlo framework. It then presents a toy experiment in which we attempt to use SMC to track a reference signal generated by a known synthetic model. Next, it presents experiment in using SMC to infer the optimal control for portfolio that tracks real-world indices. In particular, we focus on the major stock indices, how we can track the index using SMC techniques. Lastly, it introduces the MPC techniques to further improve the tracking performance.
- Chapter 4 concludes the thesis by evaluating the degree to which the hypothesis has been justified and outlines potential work for the future.

Chapter 2

Monte Carlo Methods

Monte Carlo Methods have achieved significant success in its application to various domains in the last few decades. This chapter reviews some fundamental concept in Monte Carlo method that are related to this thesis. It first begins with a summary on the main concept of Bayesian inference. It then discusses some basic Monte Carlo methods such as perfect Monte Carlo sampling, rejection sampling, importance sampling. Lastly, it details the Sequential Monte Carlo (SMC) techniques, along with various enhancement made to the framework, used in this thesis.

2.1 Bayesian Inference

In Bayesian inference framework, each unknown parameter in the model is assumed to be random variable and is associated with a prior distribution that characterises the initial belief. The inference process is merely about updating the belief with new observable evidence in a systematic fashion using Bayes theorem.

Formally, let $\mathcal{M}$ be the Bayes model of interest, θ be the set of parameters of the model, $p\left(\theta \mid \mathcal{M}\right)$ be the prior distribution (initial belief) and $p(x \mid \theta, \mathcal{M})$ be the likelihood (probability of observing an observation x given the model) then posterior distribution (updated belief) is given as follows:

$$\begin{aligned} p(\theta \mid x, \mathcal{M}) &= \frac{p(x \mid \theta, \mathcal{M})\ p(\theta \mid \mathcal{M})}{p(x \mid \mathcal{M})} \\ &\propto p(x \mid \theta, \mathcal{M})\ p(\theta \mid \mathcal{M}) \end{aligned} \tag{2.1}$$

$$\text{posterior} \propto \text{likelihood} \times \text{prior} \tag{2.2}$$

This problem formulation is elegant, but there remains some subtle issues in practice. One particular issue is the calculation of the normalisation constant $p(x \mid \mathcal{M})$ in

(2.1), which demands us to be able to carry out the following integral analytically:

$$p(x \mid \mathcal{M}) = \int p(x \mid \theta, \mathcal{M}) p(\theta \mid \mathcal{M}) \, d\theta \tag{2.3}$$

This is often infeasible. A often way to circumvent this requirement is by making use of conjugate prior that yields posterior distributions from the same family in an analytical fashion. Moreover, the need of calculating integral that does not possess analytic solution also arises in the marginalisation process of nuisance parameters, calculating expectation of a function, etc.

2.2 Perfect Monte Carlo

Instead of a closed form solution, the Method Carlo methods offer a numerical solution in estimating the integral using simple sampling techniques. Consider the calculation the expectation of a function, I of the following form:

$$I = \mathbb{E}_p[f(x)] = \int f(x) p(x) \, dx \tag{2.4}$$

Assuming we are able to sample N independent and identically distributed (i.i.d.) samples of x from $p(\cdot)$, denote these as $\{x^{(i)}\}$ where $i \in \{1 \dots N\}$, a Monte Carlo estimate of I using the the point masses of the samples is:

$$\hat{I} = \frac{1}{N} \sum_{i=1}^{N} f(x^{(i)}) \tag{2.5}$$

This approximation can be viewed as discretization of the continuous distribution with *random* support. This estimate is unbiased and converge almost surely to I as $N \to \infty$ by the Law of Large number [3].

Moreover, if the variance of $f(\cdot)$ is bounded ($\sigma_f^2 < \infty$), as $N \to \infty$, then the following central limit theorem holds:

$$\sqrt{N}(\hat{I} - I) \Longrightarrow N(0, \sigma_f^2) \tag{2.6}$$

where $\Longrightarrow$ denotes convergence in distribution [4]. The key point to note here is this convergence rate of $\frac{1}{\sqrt{N}}$ is independent of the dimensions of x. This is in contrast with any deterministic method that has a rate that decreases as the integral dimension increases [3]. This is the main advantage of Monte Carlo integration.

2.3 Rejection sampling

However, it is not always possible to sample directly from the distribution $p(\cdot)$. Suppose we can find an instrumental distribution (a.k.a. proposal distribution), $q(\cdot)$, that is easy to sample from and has the property such that $cq(x)$ dominates $p(x)$ for all x, i.e., $cq(x) \geq p(x) \geq 0$ for all x, then to get a random sample from $p(\cdot)$, we can first sample from $q(\cdot)$ instead and accept the sample with acceptance probability $\alpha(x) = \frac{p(x)}{cq(x)}$. If the sample is rejected, the process is repeated until success. This rejection algorithm is summarised in Algorithm 1[1].

Algorithm 1 Rejection Sampling

```
1:  function RejectionSampling(N)
2:      $\mathcal{X} = \{\ \}$.
3:      repeat
4:          sample $x \sim q(\cdot)$.
5:          sample $u \sim \mathcal{U}(0,1)$.
6:          if $u \leq \frac{p(x)}{cq(x)}$ then.
7:              $\mathcal{X} \leftarrow \mathcal{X} \cup \{x\}$.
8:          end if
9:      until len($\mathcal{X}$)=N.
10:     return $\mathcal{X}$.
11: end function
```

Looking at the acceptance ratio formula, it is not difficult to see that the optimal instrumental distribution, q^*, is the one that minimizes the space bounded by $cq(x)$ subject to the constraint that it still dominates the target density $p(x)$. As the dimension of x increases, this algorithm becomes very inefficient because the acceptance ratio which is essentially defined as the ratio of two spaces tends towards zero. Therefore, many generated examples would be rejected.

2.4 Importance sampling

Instead of making a binary accept-reject decision on each sample, the key concept in important sampling is assign weighting to each sample (obtained from the instrumental distribution, $q(\cdot)$) based on how well the sample resembles the target distribution, $p(\cdot)$. More formally, assume we have an instrumental distribution, $q(\cdot)$ that has support that

[1]The set notation is used here, despite it is actually a collection that allows duplicates. Therefore, the $\cup$ operation should be viewed as a simple join of two collections.

includes $p(\cdot)$, we can re-write (2.4) as:

$$\begin{aligned} I &= \int f(x)\frac{p(x)}{q(x)}q(x)\ dx \\ &= \int f(x)w(x)q(x)\ dx \\ &= \mathbb{E}_q[f(x)w(x)] \end{aligned} \tag{2.7}$$

where $w(x)$ is commonly referred as the importance weight. This reformulation leads to the following Monte Carlo estimate of I:

$$\begin{aligned} \hat{I} &= \frac{\frac{1}{N}\sum_{i=1}^{N}\tilde{w}(x^{(i)})f(x^{(i)})}{\frac{1}{N}\sum_{j=1}^{N}\tilde{w}(x^{(j)})} \\ &= \sum_{i=1}^{N}\frac{\tilde{w}(x^{(i)})}{\sum_{j=1}^{N}\tilde{w}(x^{(j)})}f(x^{(i)}) \\ &= \sum_{i=1}^{N}\hat{w}(x^{(i)})f(x^{(i)}) \end{aligned} \tag{2.8}$$

where $\tilde{w}(x^{(i)}) = \dfrac{p(x^{(i)})}{q(x^{(i)})}$ and $\hat{w}(x^{(i)}) = \dfrac{\tilde{w}(x^{(i)})}{\sum_{j=1}^{N}\tilde{w}(x^{(j)})}$ are referred to as unnormalized and normalised importance weight respectively [5]. This estimate is biased as it consists of the ratio of two estimates, yet it is still asymptotically consistent.

To obtain samples from the target distribution, $p(\cdot)$, an additional resampling step can be introduced. In the first step, we draw a set of samples $\{\tilde{x}^{(i)}\}$ from the instrumental distribution and compute their associated normalised importance weights, $\hat{w}(\tilde{x}^{(i)})$. In the resampling step, we draw the final sample set, $\{x^{(i)}\}$ from this intermediate set of samples by taking into account the importance weights. This algorithm is summarised in Algorithm 2

There are many ways of implementing the resampling stage. A simple direct implementation is to select the sample from the intermediate stage according to a Multinomial distribution with the success probability parameter set to the vector of normalised weights, $\hat{w}(x^{(i)})$, i.e., the chance of a sample point being replicated is proportional to its weight. This resampling step however introduces extra variance to the estimators, yet this can be a crucial step in the sequential scheme that we shall look in the following section to avoid sampling degeneracy over time.

2.5 Sequential Monte Carlo

To motivate why Sequential Monte Carlo is useful, consider a target distribution of interest $p(x_{0:n})$, and for simplicity, assuming we can sample directly from the distribution $p(x_{0:n})$, the minimal computational complexity of the sampling scheme would be

Algorithm 2 Importance Sampling

```
1:  function ImportanceSampling(N)
2:      $\tilde{\mathcal{X}} = \{\ \}$
3:      repeat
4:          sample $\tilde{x} \sim q(\cdot)$.
5:          $\tilde{\mathcal{X}} \leftarrow \tilde{X} \cup \{\tilde{x}\}$.
6:      until len($\tilde{X}$)=N.
7:      calculate importance weights, $\hat{w}(\tilde{x}^{(i)}) = \frac{\tilde{w}(\tilde{x}^{(i)})}{\sum_{j=1}^{N} \tilde{w}(\tilde{x}^{(j)})}$.
8:      $\mathcal{X} = \{\ \}$.
9:      repeat
10:         sample $x$ from $\tilde{\mathcal{X}}$ according to the importance weights, $\hat{w}$.
11:         $\mathcal{X} \leftarrow \mathcal{X} \cup \{x\}$.
12:     until len($\mathcal{X}$)=N.
13:     return $\mathcal{X}$.
14: end function
```

at least linear in n. Sequential Monte Carlo (SMC) provides a way to obtain samples of x for each sequential time step follows time$t$ in a *fixed* amount of computational time in Hidden Markov Models (HMMs). We shall begin with a brief introduction on HMMs that is crucial to understand SMC in the next section. Refer [5] for further details of inference techniques for HMMs in general.

2.5.1 Hidden Markov Models

HMMs can be seen as a class of models that consist of two related processes: an underlying Markov process, X_t, which is the target process of interest, and a observable process, Y_t, which its state can be measured and therefore provides some information about X_t. Moreover, it is assumed that these two processes have conditional independence properties as shown using the graphical model representation in Figure 2.1. These properties can be summarised as follows:

$$\begin{aligned} p(x_t \mid x_{0:t-1}) &= f(x_t \mid x_{t-1}) \\ p(y_t \mid x_{0:t}, y_{0:t-1}) &= g(y_t \mid x_t) \end{aligned} \tag{2.9}$$

where $f(x_t \mid x_{t-1})$ is the Markov transition density and $g(y_t \mid x_t)$ is the conditionally independent likelihood.

This class of models are designed to model evolving systems that output some observable events over time, which depends on the internal state of the system in a online manner. This imposes an implicit requirement on computation perspective: the

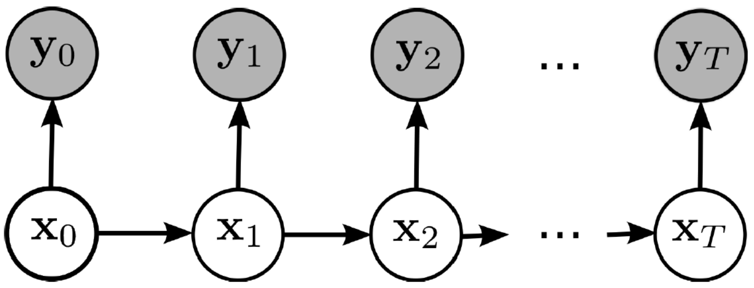

Figure 2.1: Hidden Markov Models

estimate calculation cost should remain constant over time, i.e., the calculation cost does not increase with the increasing number of states.

Arguably, the most common inference problem in HMMs is the smoothing distribution, $p(x_{0:t} \mid y_{0:t})$, that is estimating the states $x_{0:t}$ based on the sequence of observations up to time t, $y_{0:t}$. Using Bayes rules, we can write the density of the distribution of interest in the recursion form as follows:

$$\begin{aligned} p(x_{0:t} \mid y_{0:t}) &= \frac{p(x_{0:t}, y_{0:t})}{p(y_{0:t})} \\ &= \frac{p(x_{0:t-1}, y_{0:t-1}) f(x_t \mid x_{t-1}) g(y_t \mid x_t)}{p(y_{0:t})} \\ &= p(x_{0:t-1} \mid y_{0:t-1}) \frac{f(x_t \mid x_{t-1}) g(y_t \mid x_t)}{p(y_t \mid y_{0:t-1})} \end{aligned} \tag{2.10}$$

This recursion is often re-written into two separate steps: the prediction step (the estimation of distribution of all states up to time t given only states up to time $t-1$) and the update step (the correction of the predicted distribution taking into account the new observation) as follows:

$$\begin{aligned} p(x_{0:t} \mid y_{0:t-1}) &= p(x_{0:t-1} \mid y_{0:t-1}) f(x_t \mid x_{t-1}) \\ p(x_{0:t} \mid y_{0:t}) &= \frac{p(x_{0:t} \mid y_{0:t-1}) g(y_t \mid x_t)}{p(y_t \mid y_{0:t-1})} \end{aligned} \tag{2.11}$$

Moreover, the estimate of any other smoothing distribution $p(x_{j:k} \mid y_{1:t})$ where $(0 \leq j \leq k \leq t)$ can be obtained by integrating out x_i that are not interested in as follows:

$$p(x_{j:k} \mid y_{0:t}) = \int p(x_{0:t} \mid y_{0:t}) dx_{0:j-1,k+1:t} \tag{2.12}$$

A common smoothing distribution of interest is the marginal distribution at time t, $p(x_t \mid y_{0:t})$, which is also referred to as the filtering distribution.

Another distribution of interest is the prediction distribution, that is the estimation of the distribution of any unseen *future* states using only all observations up to current time. If we let $j = 0$ and $k \geq t$ in (2.12), we obtain the following equation:

$$p(x_{j:k} \mid y_{0:t}) = p(x_{j:t} \mid y_{0:t}) \prod_{i=t+1}^{k} f(x_i \mid x_{i-1}) \tag{2.13}$$

Therefore, any prediction density can be obtained by simply integrating out the variables of not interest from the above equation.

While the distribution estimation problem may appear to be simple, it is in fact far from being resolved in practice. The integrals involved in the above equations are often intractable and can only be estimated except in a very specific setting as discussed below.

2.5.2 Kalman Filter

In a conditional linear Gaussian state space as follows:

$$\begin{aligned} X_t &= A_t(U_t)X_{t-1} + B_t(U_t)W_t + F_t(U_t) \\ Y_t &= C_t(U_t)X_t + D_t(U_t)V_t + G_t(U_t) \end{aligned} \tag{2.14}$$

where $\{U_t\}_{t\geq 0}$ is a deterministic control input sequence that is used regulate the hidden states, A_t, B_t, C_t, D_t, F_t, G_t are appropriate matrix/vector functions of U_n and $\{W_t\}_{t\geq 0}$ and $\{V_t\}_{t\geq 0}$ are independent sequences of standard Gaussian random variable, i.e., $W_t, V_t \sim \mathcal{N}(0, I)$, the transition density and likelihood of this model are Gaussian distributions with center lied at a point of a linear combination of the known conditional control parameters, u_t of the following form:

$$\begin{aligned} f_t(x_t \mid x_{t-1}, u_t) &= \mathcal{N}(A_t(u_t)x_{t-1} + F_t(u_t), B_t(u_t)B_t(u_t)^T) \\ g_t(y_t \mid x_t, u_t) &= \mathcal{N}(C_t(u_t)x_t + G_t(u_t), D_t(u_t)D_t(u_t)^T) \end{aligned} \tag{2.15}$$

Using the properties of Gaussian distribution, the integral involved can be resolved analytically. This leads to the widely used *Kalman Filter* [6] that has the following recursive solution as follows:

$$\mu_{t|t-1} = A_t(u_t)(\mu_{t-1|t-1})X_{t-1} + F_t(u_t) \tag{2.16}$$

$$\Sigma_{t|t-1} = A_t(u_t)\Sigma_{t-1|t-1}A_t(u_t)^T + B_t(u_t)B_t(u_t)^T \tag{2.17}$$

$$S_t = C_t(u_t)\Sigma_{t|t-1}C_t(u_t)^T + D_t(u_t)D_t(u_t)^T \tag{2.18}$$

$$y_{t|t-1} = C_t(u_t)\mu_{t|t-1} + G_t(u_t) \tag{2.19}$$

$$\mu_{t|t} = \mu_{t|t-1} + \Sigma_{t|t-1}C_t(u_t)S_t^{-1}(y_t - y_{t|t-1}) \tag{2.20}$$

$$\Sigma_{t|t} = \Sigma_{t|t-1} - \Sigma_{t|t-1}C_t(u_t)S_t^{-1}C_t(u_t)\Sigma_{t|t-1} \tag{2.21}$$

where $\mu_{t|t-1}$ and $\Sigma_{t|t-1}$ are the predicted mean and co-variance matrix of the state x_t, $y_{t|t-1}$ and S_t are the mean and co-variance matrix of the measurement at time t and $\mu_{t|t}$ and $\Sigma_{t|t}$ are the estimated mean and co-variance matrix of the state x_t after seeing the observation y_t.

There are various extensions have been developed on top of this approach. For example, the Extended Kalman Filter (EKF) which uses Taylor Series expansion to linearise at the conditional variables locally [7], Unscented Kalman Filter which further extend the idea in EKF by only using a minimal set of well chosen samples [8], etc.

2.5.3 Sequential Important Sampling (SIS)

In general, there is however no analytical solution for this estimation problem. Sequential Monte Carlo provides a systematic way to approximate the solution to this estimation problem. Let assume that it is possible to decompose the selected proposal distribution into recursion form as follows:

$$\begin{aligned} q_{0:t}(x_{0:t} \mid y_{0:t}) &= q_{0:t-1}(x_{0:t-1} \mid y_{0:t-1}) q_t(x_t \mid x_{t-1}, y_t) \\ &= q_0(x_0 \mid y_0) \prod_{i=1}^{t} q_i(x_i \mid x_{i-1}, y_i) \end{aligned} \tag{2.22}$$

then it is possible to obtain sample $x_{0:t}$ by first sampling $x_0 \sim q_0(\cdot)$ at time 0 and then $x_i \sim q_i(x_i \mid x_{i-1}, y_i)$ for all time i from 1 to t. The corresponding weight associated to each sample $x_{0:t}$ can also be decomposed into recursion form as follows:

$$\begin{aligned} \tilde{w}_t &= \frac{p_{0:t}(x_{0:t} \mid y_{0:t})}{q_{0:t}(x_{0:t} \mid y_{0:t})} \\ &= \frac{p_{0:t-1}(x_{0:t-1} \mid y_{0:t-1})}{q_{0:t-1}(x_{0:t-1} \mid y_{0:t-1})} \frac{p_{0:t}(x_{0:t} \mid y_{0:t})}{p_{0:t-1}(x_{0:t-1} \mid y_{0:t-1}) q_t(x_t \mid x_{0:t-1}, y_t)} \\ &= \tilde{w}_{t-1} \frac{p_{0:t}(x_{0:t} \mid y_{0:t})}{p_{0:t-1}(x_{0:t-1} \mid y_{0:t-1}) q_t(x_t \mid x_{0:t-1}, y_t)} \\ &\propto \tilde{w}_{t-1} \frac{f_t(x_t \mid x_{t-1}) g_t(y_y \mid x_t)}{q_t(x_t \mid x_{t-1}, y_t)} && (2.23) \\ &\propto \tilde{w}_0 \prod_{i=1}^{t} \alpha_i(x_i) && (2.24) \end{aligned}$$

where $\alpha_t(x_t)$ is often referred to as incremental importance weight function. This is the key concept in SIS.

Using these weighted samples $\{(x_{0:t}^{(i)}, \tilde{w}_t^{(i)})\}_{1 \le i \le N}$, it is possible to estimate any function f defined on the space using the self normalised importance sampling estimator in the same way as (2.8) as follows:

$$\begin{aligned}\hat{f}(x_{0:t}) &= \sum_{i=1}^{N} \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^{N} \tilde{w}_t^{(j)}} f(x_{0:t}^{(i)}) \\ &= \sum_{i=1}^{N} \hat{w}^{(i)} f(x_{0:t}^{(i)}) \end{aligned} \tag{2.25}$$

The SIS algorithm is summarised in Algorithm 3.

Algorithm 3 Sequential Importance Sampling

```
1:  function SequentialImportanceSampling(N, T)
2:      Set $t \leftarrow 0$.
3:      For $i \in 1, \dots, N$, sample $x_0^{(i)} \sim q(x_0^{(i)} \mid y_0^{(i)})$.
4:      For $i \in 1, \dots, N$, calculate the unnormalized importance weights:
            $$\tilde{w}_0^{(i)} = \frac{f(x_0^{(i)}) g_0(y_0 \mid x_0^{(i)})}{q_0(x_0^{(i)})}$$
5:      For $i \in 1, \dots, N$, normalize the importance weights:
            $$\hat{w}_0^{(i)} = \frac{\tilde{w}_0^{(i)}}{\sum_{i=1}^{N} \tilde{w}_0^{(i)}}$$
6:      Set $t \leftarrow t + 1$.
7:      while $t \leq T$ do
8:          For $i \in 1, \dots, N$, sample $x_t^{(i)} \sim q(x_t^{(i)} \mid y_{t-1}^{(i)}, x_{t-1}^{(i)})$.
9:          For $i \in 1, \dots, N$, set $x_{0:t}^{(i)} \leftarrow (x_{0:t-1}^{(i)}, x_t^{(i)})$.
10:         For $i \in 1, \dots, N$, calculate the unnormalized importance weights:
                $$\tilde{w}_t^{(i)} = w_{t-1}^{(i)} \frac{f_t(x_t^{(i)} \mid x_{t-1}^{(i)}) g_t(y_t \mid x_t^{(i)})}{q_t(x_t^{(i)} \mid x_{t-1}^{(i)})}$$
11:         For $i \in 1, \dots, N$, normalize the importance weights:
                $$\hat{w}_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{i=1}^{N} \tilde{w}_t^{(i)}}$$
12:     end while
13: end function
```

2.5.4 Optimal Proposal Distribution

While SIS is attractive, it is nothing but a specialised version of importance sampling introduced earlier in Section 2.4. As the state space increases with the number of time step t, direct importance sampling on a state space that is increasing in size is not very efficient. The weights of the samples will degenerate over time, in the sense that the weights start to concentrate only on a small number of samples. Consequently, many samples will have negligible weights and do not contribute much in the estimating the expectation. See [9] for a step-by-step illustration. The weight degeneracy issue cause the quality of the estimate degrades over time.

To alleviate this issue, looking at (2.23), it is obvious that the variance of importance weights can be minimised by using a proposal distribution of the following form:

$$q_t(x_t \mid x_{t-1}, y_t) \propto f_t(x_t \mid x_{t-1}) g_t(y_t \mid x_t) \tag{2.26}$$

This is often referred to as the optimal proposal distribution.

In general, it is not always possible to sample from this optimal proposal distribution. Yet, the knowledge of its form can be helpful in designing a reasonable good proposal distribution, which one can sample from. Better proposal *reduces* the amount of variance introduced, but it *does not eliminate* the weight degeneracy problem.

2.5.5 Sequential Importance Resampling (SIR)

The variance in importance weights accumulates over iterations. This suggests a possible solution is to "reset" the weights associated to the samples somehow during the iterations [9]. Sequential Importance Resampling (SIR) introduces an additional resampling step to SIS step in a similar fashion as discussed in Section 2.4. After resampling, the weight of each sample is reset to be equal, i.e., $\frac{1}{N}$.

Besides the simplest multinomial resampling scheme, many different resampling schemes have been proposed in the literature. For example, stratified resampling [10] as the name suggested splitting the samples into strata to ensure the good coverage on the resulting sample set, residual resampling [11] which is able to decrease the variance of the weights due to resampling, etc. See [12] for further details on the comparison of these sampling schemes. The SIR algorithm is now summarised in Algorithm 4.

2.5.6 Effective sample size (ESS)

Resampling step induces additional Monte Carlo variance to the weights. Yet, this step is necessary to avoid accumulation of estimation variance onto the weights over time and therefore result in a more estimate.

To trade-off these two competing requirements, one possible way is to monitor the effective sample size (ESS) which provides a measure on the quality of the weighted samples. The ESS value can be estimated as follows:

$$ESS \approx \frac{1}{E[w^2]} \approx \frac{\left(\sum_{i=0}^{N} w_i\right)^2}{\sum_{i=0}^{N} w_i^2} \tag{2.27}$$

A common way to integrate this idea is to trigger the resampling step only if the ESS_t fall below certain threshold at time t, say $\frac{N}{2}$. See [9] for further details on ESS.

Algorithm 4 Sequential Importance Resampling

```
 1: function SequentialImportanceResampling(N, T)
 2:     Set $t \leftarrow 0$.
 3:     For $i \in 1, \dots, N$, sample $x_0^{(i)} \sim q(x_0^{(i)} \mid y_0^{(i)})$.
 4:     For $i \in 1, \dots, N$, calculate the unnormalized importance weights:
            $\tilde{w}_0^{(i)} = \frac{f(x_0^{(i)}) g_0(y_0 \mid x_0^{(i)})}{q_0(x_0^{(i)})}$
 5:     For $i \in 1, \dots, N$, normalize the importance weights:
            $\hat{w}_0^{(i)} = \frac{\tilde{w}_0^{(i)}}{\sum_{i=1}^{N} \tilde{w}_0^{(i)}}$
 6:     Set $t \leftarrow t + 1$.
 7:     while $t \leq T$ do
 8:         For $i \in 1, \dots, N$, sample $x_t^{(i)} \sim q(x_t^{(i)} \mid y_{t-1}^{(i)}, x_{t-1}^{(i)})$.
 9:         For $i \in 1, \dots, N$, set $x_{0:t}^{(i)} \leftarrow (x_{0:t-1}^{(i)}, x_t^{(i)})$.
10:         For $i \in 1, \dots, N$, calculate the unnormalized importance weights:
                $\tilde{w}_t^{(i)} = w_{t-1}^{(i)} \frac{f_t(x_t^{(i)} \mid x_{t-1}^{(i)}) g_t(y_t \mid x_t^{(i)})}{q_t(x_t^{(i)} \mid x_{t-1}^{(i)})}$
11:         For $i \in 1, \dots, N$, normalize the importance weights:
                $\hat{w}_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{i=1}^{N} \tilde{w}_t^{(i)}}$
12:         Resample: For $i \in 1, \dots, N$, resample $x_{0:t}^{(i)} \sim \frac{\sum_{i=1}^{N} \hat{w}_t^{(i)} \delta_{x_{0:t}^{(i)}}}{\sum_{j=1}^{N} \hat{w}_t^{(j)}}$
13:     end while
14: end function
```

2.5.7 Resample-Move Algorithm

However, resampling is not a silver bullet for sampling impoverishment. Essentially, resampling provides a mechanism to eliminate low weight samples to give way to replicate *copies* of high weight samples. This allows all samples to participate and contribute to the distribution estimation. This is obvious beneficial for the case of estimating filtering distribution and predictive distribution. Over time, this replication result in decrease in the number of distinct samples for previous time steps. Eventually, many samples will have the share the same sample trajectory. This is a fundamental weakness of SMC, in which the sample path history is never re-written. This lose of diversity in the sample set will have a negative impact when it comes to any smoothing distribution estimation.

To counteract this sample impoverishment, Resample-Move Algorithm [13] is proposed to introduce some perturbation to the samples (so to diversify them) without changing the distribution they represent. This is accomplished by using MCMC steps with a Markov Kernel, K that is invariant to the target distribution, $p(\cdot)$. In the original paper, this is achieved by simply introducing an additional MCMC "move" step to each sample after the resampling step using a kernel K that is invariant to the target distribution. This Resample-Move algorithm is summarised in Algorithm 5.

This does not entirely solve the smoothing distribution estimation problem. To apply Markov Kernels with invariant distribution corresponding to the smoothing distribution, the space that Markov kernel is defined has to increase at each iteration. This implies the computation time increases linearly with time. Moreover, fast mixing high dimension Markov kernel in itself is not easy to design [9].

To trade-off between the two requirements, one could adopt a sliding windows approach, in which MCMC Kernels which diversify the samples of the previous n time step at each iteration. Adding this sliding window approach into the standard SIR algorithm will make each iteration has an additional *fixed* computational cost.

2.5.8 Marginalised SMC

In practice, many models may not be entirely Gaussian. Some states may be linear and Gaussian, conditional upon other states. The naive way to model this is to use SMC techniques to model all the states.

A better approach would be making good use of the Gaussian properties. The idea is to split the states into the linear Gaussian states and the non-linear states. Then, the Kalman Filter which is optimal and possesses a closed form solution can be used to

Algorithm 5 Resample-Move Algorithm

```
1: function ResampleMoveAlgorithm(N, T)
2:     Set $t \leftarrow 0$.
3:     For $i \in 1, \dots, N$, sample $x_0^{(i)} \sim q(x_0^{(i)} \mid y_0^{(i)})$.
4:     For $i \in 1, \dots, N$, calculate the unnormalized importance weights:
           $$\tilde{w}_0^{(i)} = \frac{f(x_0^{(i)}) g_0(y_0 \mid x_0^{(i)})}{q_0(x_0^{(i)})}$$
5:     For $i \in 1, \dots, N$, normalize the importance weights:
           $$\hat{w}_0^{(i)} = \frac{\tilde{w}_0^{(i)}}{\sum_{i=1}^N \tilde{w}_0^{(i)}}$$
6:     Set $t \leftarrow t + 1$.
7:     while $t \leq T$ do
8:         For $i \in 1, \dots, N$, sample $x_t^{(i)} \sim q(x_t^{(i)} \mid y_{t-1}^{(i)}, x_{t-1}^{(i)})$.
9:         For $i \in 1, \dots, N$, set $x_{0:t}^{(i)} \leftarrow (x_{0:t-1}^{(i)}, x_t^{(i)})$.
10:        For $i \in 1, \dots, N$, calculate the unnormalized importance weights:
               $$\tilde{w}_t^{(i)} = w_{t-1}^{(i)} \frac{f_t(x_t^{(i)} \mid x_{t-1}^{(i)}) g_t(y_t \mid x_t^{(i)})}{q_t(x_t^{(i)} \mid x_{t-1}^{(i)})}$$
11:        For $i \in 1, \dots, N$, normalize the importance weights:
               $$\hat{w}_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{i=1}^N \tilde{w}_t^{(i)}}$$
12:        Resample: For $i \in 1, \dots, N$, resample $x_{0:t}^{(i)} \sim \frac{\sum_{i=1}^N \hat{w}_t^{(i)} \delta_{x_{0:t}^{(i)}}}{\sum_{j=1}^N \hat{w}_t^{(j)}}$
13:        Move: For $i \in 1, \dots, N$, sample $x_{0:t}^{(i)} \sim K_t(\cdot)$, where $K_t$ is $p_t$-invariant.
14:    end while
15: end function
```

model the linear Gaussian states and the SMC techniques can be used to model only the non-linear states. This marginalisation setting will yield estimates with smaller variance.

To illustrate this, let consider the a simple conditional linear Gaussian Model (2.14) introduced earlier, restated as follows to ease referencing:

$$\begin{aligned} X_t &= A_t(U_t)X_{t-1} + B_t(U_t)W_t + F_t(U_t) \\ Y_t &= C_t(U_t)X_t + D_t(U_t)V_t + G_t(U_t) \end{aligned}$$

with the model dynamic specified as follows:

$$\begin{aligned} p_t(u_t \mid u_{t-1}) &= (\text{any given form}) \\ f_t(x_t \mid x_{t-1}, u_t) &= \mathcal{N}(A_t(u_t)x_{t-1} + F_t(u_t), B_t(u_t)B_t(u_t)^T) \\ g_t(y_t \mid x_t, u_t) &= \mathcal{N}(C_t(u_t)x_t + G_t(u_t), D_t(u_t)D_t(u_t)^T) \end{aligned} \tag{2.28}$$

At any time t, the full posterior distribution of this model can be factorised as follows:

$$p(u_{0:t}, x_{0:t} \mid y_{0:t}) = p(x_{0:t} \mid u_{0:t}, y_{0:t})p(u_{0:t} \mid y_{0:t}) \tag{2.29}$$

Looking at the right hand side of the equation, the first term is Gaussian and can be estimated optimally using Kalman Filter. For the second term, we can rewrite it into the following recursion form:

$$\begin{aligned} p(u_{0:t} \mid y_{0:t}) &\propto p(y_t \mid u_{0:t}, y_{0:t-1})p(u_{0:t} \mid y_{0:t-1}) \\ &= p(y_t \mid u_{0:t}, y_{0:t-1})p(u_t \mid u_{0:t-1}, y_{0:t-1})p(u_{0:t-1} \mid y_{0:t-1}) \\ &= p(u_{0:t-1} \mid y_{0:t-1})p(y_t \mid u_{0:t}, y_{0:t-1})p(u_t \mid u_{0:t-1}, y_{0:t-1}) \end{aligned} \tag{2.30}$$

Assuming that it is also possible to decompose the selected importance density into recursion form as follows:

$$q_{0:t}(u_{0:t} \mid y_{0:t}) = q_{0:t-1}(u_{0:t-1} \mid y_{0:t-1})q_t(u_t \mid u_{0:t-1}, y_{0:t}) \tag{2.31}$$

then the associated unnormalized weight of each sample can be derived in a similar fashion as given in Section 2.5.3 into the recursion form as follows:

$$\tilde{w}_t = \tilde{w}_{t-1}\frac{p_t(u_t \mid u_{t-1})p_t(y_t \mid u_{0:t}, y_{0:t-1})}{q_t(u_t \mid u_{0:t-1}, y_{0:t})} \tag{2.32}$$

Compare to (2.22), the main difference here is that the weights now depends on the whole path space from time 0 up to t. The Marginalised SMC is summarised in Algorithm 6.

The discussion here has been focused on conditional linear Gaussian model. The discrete state space HMMs is another important class of HMMs in which HMM forward algorithm [14] can be used to marginalise the internal states. See [5] for further details.

Algorithm 6 Marginalised Sequential Monte Carlo

```
1: function MarginalisedSequentialMonteCarlo(N, T)
2:     Set $t \leftarrow 0$.
3:     For $i \in 1, \dots, N$, sample $u_0^{(i)} \sim q(u_0^{(i)} \mid y_0^{(i)})$.
4:     For $i \in 1, \dots, N$, calculate the unnormalized importance weight:
           $$\tilde{w}_0^{(i)} = \frac{p(u_0^{(i)}) g_0(y_0 \mid u_0^{(i)})}{q_0(u_0^{(i)})}$$
5:     For $i \in 1, \dots, N$, normalize the importance weight:
           $$\hat{w}_0^{(i)} = \frac{\tilde{w}_0^{(i)}}{\sum_{i=1}^N \tilde{w}_0^{(i)}}$$
6:     Set $t \leftarrow t + 1$.
7:     while $t \leq T$ do
8:         For $i \in 1, \dots, N$, sample $u_t^{(i)} \sim q(u_t^{(i)} \mid y_{t-1}^{(i)}, u_{t-1}^{(i)})$.
9:         For $i \in 1, \dots, N$, calculate the unnormalized importance weight:
               $$\tilde{w}_t = \tilde{w}_{t-1} \frac{p_t(u_t \mid u_{t-1}) p_t(y_t \mid u_{0:t}, y_{0:t-1})}{q_t(u_t \mid u_{0:t-1}, y_{0:t})}$$
           where
               $$p(y_t \mid u_{0:t}^{(i)}, y_{1:t-1}) \sim N(y_{t|t-1}, S_t) \tag{2.33}$$
           with $y_{t|t-1}$ and $S_t$ are updated with Kalman Filter:
               $$\begin{aligned}
               \mu_{t|t-1} &= A_t(u_t)(\mu_{t-1|t-1})X_{t-1} + F_t(u_t) \\
               \Sigma_{t|t-1} &= A_t(u_t)\Sigma_{t-1|t-1}A_t(u_t)^T + B_t(u_t)B_t(u_t)^T \\
               S_t &= C_t(u_t)\Sigma_{t|t-1}C_t(u_t)^T + D_t(u_t)D_t(u_t)^T \\
               y_{t|t-1} &= C_t(u_t)\mu_{t|t-1} + G_t(u_t) \\
               \mu_{t|t} &= \mu_{t|t-1} + \Sigma_{t|t-1}C_t(u_t)S_t^{-1}(y_t - y_{t|t-1}) \\
               \Sigma_{t|t} &= \Sigma_{t|t-1} - \Sigma_{t|t-1}C_t(u_t)S_t^{-1}C_t(u_t)\Sigma_{t|t-1}
               \end{aligned}$$
10:        For $i \in 1, \dots, N$, normalize the importance weight:
               $$\hat{w}_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{i=1}^N \tilde{w}_t^{(i)}}$$
11:        Resample: For $i \in 1, \dots, N$, resample $u_{0:t}^{(i)} \sim \frac{\sum_{i=1}^N \hat{w}_t^{(i)} \delta_{u_{0:t}^{(i)}}}{\sum_{j=1}^N \hat{w}_t^{(j)}}$
12:        Move: For $i \in 1, \dots, N$, sample ${}_{0:t}^{(i)} \sim K_t(\cdot)$, where $K_t$ is $p_t$-invariant.
13:    end while
14: end function
```

2.6 Conclusion

This chapter presents a review of Monte Carlo methods, with a particular focus on Sequential Monte Carlo (SMC) techniques used extensively in this thesis for portfolio optimisation. This chapter begins with a brief introduction on the traditional Monte Carlo sampling techniques. Then, it details the SMC techniques, along with various extensions proposed to improve the performance of the algorithm.

It is worth to have a remark here that the SMC techniques are not only applicable to sequential filtering problem. For example, it has been established that it is possible to use SMC techniques within MCMC framework (pMCMC, where p stands for particle) [15] to solve other problems. In the next chapter, we will show how SMC can be used as a maximiser to search for a optimal strategy for the optimal portfolio, given a multiplicative reward function.

Chapter 3

Portfolio optimisation

Optimal portfolio is the ultimate goal of every investment manager; but the optimality criteria can be very different to each of them. In the infamous Markowitz's modern portfolio theory [1]. it is assumed that investor attempt to maximize a portfolio's return and minimize the risk (measured by the variance of the portfolio return). For index tracker fund manager, the main objective of portfolio management is to track and replicate the exposure of a benchmark index. The lack of active management generally makes the fund less vulnerable to change of management and has the advantages of lower fees and taxes. It is the latter the focus of the thesis lies.

This chapter begins with a brief introduction on index tracking fund and a short summary on some causing factors of the tracking error between a index fund and its benchmark index. It then describes how the portfolio optimisation problem in terms of minimising the tracking error can be formulated as a stochastic control problem, which is then turned into a parameter estimation in the SMC framework. Next, it details some of the experiments carried out to verify this approach, starting with a simple proof of concept experiment in Section 3.3, continue on with a more complicated example with real-world data in Section 3.4 and concludes the chapter with an experiment in demonstrating how the Model Predictive Control (MPC) can be incorporating into the basic SMC framework proposed.

3.1 Index Tracking Fund

An financial index can be thought of a summary statistics of a market, typically formed as weighted average of the prices of some financial instruments. For example, FTSE 100 is an index that attempts to represents the performance of the 100 biggest companies in UK. This index is however not an investable financial product one can invest directly.

To allows investors to take exposure to the markets (as represented by the indices), index tracker funds are introduced. These funds generally follow very strict and transparent rules with the main objective to track their benchmark indices as close as possible. Investors of these funds therefore are only exposed to the market risks of the chosen indices, with minimal exposure to the risk associated with the investment process of the traditional active fund management.

3.1.1 Index Replication

To track an index, the simplest way is to replicate the index by investing all the components of the benchmark index. However, this can be costly and difficult to manage, considering some of the indices may have huge number of components, e.g., MSCI World index that consists of ≈ 1600 components from many different countries. Instead, one could partial replicate the index by sampling some of the components that are most representative. This can mean those components with larger weights, more volatiles or less correlated in returns (assuming these returns average out the gain/loss among them). This partial replication could save transaction cost, at the same time, introducing some tracking errors to the funds.

Instead of physically replicating the index by investing in the components, the portfolio manager also have the option to enter a swap agreement with a counter-party (typically investment bank) that provides the exact return of the stock market or commodity it's tracking. Essentially, this transfers the market risk to the counter-party, at the same introducing the counter-party default risk. This technique is known synthetic replication.

3.1.2 Causing factors of tracking error

To evaluate the performance of index tracking fund, different metrics have been introduced to quantify the mismatch between the performance of a fund and its benchmark index. For example, tracking difference is the sum of absolute difference in returns between of a fund and its benchmark. Here, we adopt the tracking error as our metric, which is defined to be the standard deviation of the absolute difference of the returns of the fund and the benchmark defined in [16] as follows:

$$\epsilon = \sqrt{\mathbb{Var}(r_p - r_b)} \tag{3.1}$$

where r_p is the return of the portfolio and r_b is the return of the benchmark index.

There are many factors that can cause tracking errors, some of which are summarised as follows:

1. benchmark index rebalance — the benchmark index is re-weighting its constituents periodically to reflex the changes on the market based on its methodology. To track the index, the fund has to adjust its portfolio accordingly. This will incur some transaction costs. During the index rebalance period, cash drag may also happen between the liquidation of the constituents that have weights reduced/dropped and the addition of the constituents that have weights increased/added. This cash is essentially not participating in the market and therefore do not reflex the changes on the benchmark index.

2. replication and sampling techniques — funds may choose to replicate the benchmark index by selecting a subset of the constituents (often the ones with larger weights and more liquid) in an effort to minimize the transaction costs. This exclusion of the smaller, less liquid constituents may introduce another source of tracking error, especially under stressed market.

3. different assumption on dividend reinvestment and taxation — This is best illustrate with examples. For example, the benchmark index calculation may assume immediate reinvestment of dividends on ex-dividend dates but the fund may only able to reinvest the dividend after receiving it. The tax treatment that is applied on the dividends may also be different.

4. total expense ratio — there is an additional expense charged to the fund on daily basis to cover the management cost.

This list is by no mean exclusive. See [16] for further details.

3.2 Technical Approach

Instead of using a traditional deterministic state space models, we choose to use a *stochastic* modelling approach to carry out the portfolio optimisation. In particular, we focus here the the problem in minimizing the tracking error between a portfolio and its benchmark index. Our aim is to determine what investment actions (buy or sell) on the necessary index components a portfolio manager has to do on a daily basis across the investment horizon to track the benchmark index well.

In the next section, we proceed by presenting the stochastic state space model that we assume throughout this thesis. This model is by no mean to compete the state of the art model in realistic portfolio optimisation, but rather to motivate further work in this direction.

3.2.1 Model specification and optimisation objective

We adopt the simple conditional linear Gaussian Model (2.14) introduced earlier, restated as follows to ease referencing:

$$\begin{aligned} X_t &= A_t(U_t)X_{t-1} + B_t(U_t)W_t + F_t(U_t) \\ Y_t &= C_t(U_t)X_t + D_t(U_t)V_t + G_t(U_t) \end{aligned} \tag{3.2}$$

where $\{U_t\}_{t\geq 0}$ is a deterministic control input sequence that is used regulate the hidden states, A_t, B_t, C_t, D_t, F_t, G_t are appropriate matrix/vector functions of U_n and $\{W_t\}_{t\geq 0}$ and $\{V_t\}_{t\geq 0}$ are independent sequences of standard Gaussian random variable, i.e., $W_t, V_t \sim \mathcal{N}(0, I)$. The transition density and likelihood of this model are Gaussian distributions with center lied at a point of a linear combination of the known conditional control parameters, u_t of the following form:

$$\begin{aligned} p_t(u_t \mid u_{t-1}) &= \text{(any given form)} \\ f_t(x_t \mid x_{t-1}, u_t) &= \mathcal{N}(A_t(u_t)x_{t-1} + F_t(u_t), B_t(u_t)B_t(u_t)^T) \\ g_t(y_t \mid x_t, u_t) &= \mathcal{N}(C_t(u_t)x_t + G_t(u_t), D_t(u_t)D_t(u_t)^T) \end{aligned} \tag{3.3}$$

With this model, the optimisation objective is to search for a sequence of controls $u_{1:t}$ that would result in a sequence of observations $y_{1:t}$ that tracks the reference signal $y^{ref}_{1:T}$ as close as possible. This problem is often known as stochastic regulation problem. We adopt the finite horizon multiplicative reward function proposed in [17] here:

$$J(u_{1:T}, y^{ref}_{1:T}, x_0) = \mathbb{E}_{x_0}\left[-\frac{1}{2}\sum_{t=1}^{T}\left(||y^{ref}_t - C_t(u_t)x_t - G_t(u_t)||^2_{(D_t(u_t)D_t(u_t)^T)^{-1}} + ||u_t - u_{t-1}||^2_{L_t}\right)\right] \tag{3.4}$$

where the expectation is taken with respect to the whole path of the Markov Chain $\{X_t\}_{t\geq 0}$, starting with $X_0 = x_0$, i.e., $E_{x_0}[\phi(X_{1:t})] = \displaystyle\int \phi(X_{1:T}) \prod f_t(x_t \mid x_{t-1})\ dx_{1:t}$, with D_t and L_t are assumed to be known. The corresponding optimal open loop policy is:

$$u^*_{1:T} = \arg\max_{u_{1:T}} J(u_{1:T}; y^{ref}_{1:T}; x_0) \tag{3.5}$$

with the assumption that the maximum is attainable. This reward function is closely related to the risk sensitive control discussed in [18], with slightly difference. This reward function is absence of an explicitly risk sensitive constant and assume the presence of $y^{ref}_{1:T}$ as the reference target instead of the observable states. Nevertheless, a risk sensitivity constant could still be introduced through D_n and L_n if necessary.

3.2.2 Problem formulation

Under the realm of Bayesian inference framework, we treat the control inputs u as random variables that admit a prior distribution. We also assume here that the sequence of control u_n is a Markov process with transition distribution $p(u_t \mid u_{t-1})$. The objective is to compute the marginal posterior distribution density:

$$\pi_t(u_{0:t}) = p(u_{0:t} \mid y^{ref}_{0:t}) \tag{3.6}$$

by standard marginalisation the distribution $p(x_{0:t}, u_{0:t} \mid y_{0:t})$ which in itself satisfies the following recursive equation:

$$p(x_{0:t}, u_{0:t} \mid y_{0:t}) = p(x_{0:t-1}, u_{0:t-1} \mid y_{0:t-1}) \frac{p(y_t \mid x_t, u_t) p(x_t, u_t \mid x_{t-1}, u_{t-1})}{p(y_t \mid y_{0:t-1})} \tag{3.7}$$

One straight-forward way to approximate the recursion is directly apply SMC algorithm by consider the hidden states is a sequence of paired states $\{(X_t, Y_t)\}_{t\geq 0}$. A better approach be using the Marginalised SMC approach (see Section 2.5.8 for detail), which splits the states into the linear Gaussian states and the non-linear states. Then, the Kalman Filter can be used to model the linear Gaussian states and the SMC can be to model only the non-linear states. This marginalisation setting often yields estimates with smaller variance.

To use Marginalised SMC for this problem, we consider the following factorisation:

$$p(x_{0:t}, u_{0:t} \mid y_{1:t}) = p(x_{0:t} \mid u_{0:t}, y_{1:t}) p(u_{0:t} \mid y_{1:t}) \tag{3.8}$$

we can see that given $p(u_{0:t} \mid y_{1:t})$, $p(x_{0:t} \mid u_{0:t}, y_{1:t})$ is simply a Gaussian mixture model, which can be computed optimally with Kalman Filter. The density $p(u_{0:t} \mid y_{1:t})$ that is not linear Gaussian has the following recursion form as shown in (2.30):

$$p(u_{0:t} \mid y_{1:t}) = p(u_{0:t-1} \mid y_{0:t-1}) p(y_t \mid u_{0:t}, y_{0:t-1}) p(u_t \mid u_{0:t-1}, y_{0:t-1}) \tag{3.9}$$

which can be approximated with standard SMC algorithm. The Marginalised SMC algorithm is summarised in Algorithm 6.

3.2.3 MAP estimation for the control sequence

Based on the model defined in (3.2), the conditional likelihood density of the model is:

$$g_t(y^{ref}_t \mid x_t, u_t) \propto \exp\left(-\frac{1}{2} ||y^{ref}_t - C_t(u_t)x_t - G_t(u_t)||^2_{(D_t(u_t)D_t(u_t)^T)^{-1}}\right) \tag{3.10}$$

and using the standard Kalman recursion discussed in Section 2.5.2, the posterior distribution $p(y_t^{ref} \mid y_{t-1}^{ref}, u_{0:t})$ is given as follows:

$$p(y_t^{ref} \mid y_{t-1}^{ref}, u_{0:t}) \propto \exp\left(-\frac{1}{2}||y_t^{ref} - y_{t|t-1}(u_{0:t})||^2_{S_t(u_{0:t})^{-1}}\right) \tag{3.11}$$

where $y_{t|t-1}$ and S_t are the mean and covariance matrix of the measurement at time t obtained in the Kalman Filter recursion.

By assuming the Markov transition density for u_t to be:

$$p(u_t \mid u_{t-1}) \propto \exp\left(-\frac{1}{2}||u_t - u_{t-1}||^2_{L_t}\right) \tag{3.12}$$

the Maximum a Posteriori (MAP) estimate of $U_{0:T}$ is given to be:

$$\tilde{u}^*_{0:t} = \arg\max_{u_{0:t}} \pi_n(u_{0:t}) \tag{3.13}$$

which essentially specifies the model that best explains the reference $y_{0:t}^{ref}$ as the output sequence from a general set of models defined in (3.2).

In [17], it had been shown that assuming a Markov transition density for u_t in (3.12), the optimal control defined in (3.5) and the MAP estimate defined in (3.13) are the same. Moreover, following the monotonicity of the transformation $p(\cdot)^\gamma$, the mode of $p(\cdot)^\gamma$ is the same for all $\gamma > 0$. The main difference is that it is easier to sample closer to the mode when $\gamma > 1$ because the density has sharper peak and vice versa. Putting all these together, we have the follows:

$$u^*_{0:t} = \tilde{u}^*_{0:t} = \arg\max_{u_{0:t}} \pi_n(u_{0:t})^\gamma, \ \gamma > 0 \tag{3.14}$$

In SMC algorithm, this can be easily estimated as follows:

$$\hat{u}^*_{0:t} = arg\max_{u_{0:t}} \pi_n(u_{0:t}^{(i)})^\gamma = \arg\max_{u_{0:t}} p(u_{0:t}^{(i)} \mid y_{0:t}^{ref})^\gamma \tag{3.15}$$

As long as the support of the proposal density include the support of $(u_{0:t}^{(i)} \mid y_{0:t}^{ref})$, this estimate converge asymptotically to the $\tilde{u}^*_{0:t}$ as $N \to \infty$. A better estimate can be obtained by embedding the Viterbi strategy as discussed in [19], but this comes with a computational cost that is quadratic in terms of the number of samples.

To add diversity to the population samples, the Resample-Move step (see Section 2.5.7) can be added. This step perturbs the samples yet leave the distribution the samples represent unchanged by using a MCMC kernel that is invariant in density.

3.3 Example 1: Tracking stationary oscillating wave

We first consider here a simple linear Gaussian state space model as follows:

$$\begin{aligned} X_t &= X_{t-1} + W_t + U_t \\ Y_t &= X_t + V_t \end{aligned} \tag{3.16}$$

with $W_t, V_t \sim \mathcal{N}(0, I)$. This model is essential an instance of the class of model presented earlier in (3.2), with $A_t = B_t = C_t = D_t = I$, $F_t(u_t) = u_t$, $G_t(u_t) = 0$ and $u_0 = 0$. The target reference is set to be an oscillating wave: $y_t^{ref} = \cos(0.2\pi t + 0.3)$ and L_1 is set to 0.1 similar to the setting used in [17].

This model serves two purposes here. Firstly, it is sufficient simple to verify the implementation[1]. Secondly, it serves as good benchmark problem in which we can examine the sensitivity of the algorithm to different parameter settings.

We proceed by examining the proposed algorithm with proposal density $q_t(\cdot)$ set to be the same as the Markov transition density and *all* different combination of settings as follows:

1. Various time period length, T: 5, 10 and 20.
2. Various number of samples, N: 100, 500, 1000, 5000 and 10000.
3. Resample step: Always vs. Selectively (trigger threshold set to $\frac{ESS}{2}$).
4. Resample-Move step with MCMC (random walk proposal): Always vs. disabled.
5. Various γ settings: Constant function of 1, 50, 100, 1000 and increasing function of time $t + 1$, $10(t + 1)$, $50(t + 1)$ and $100(t + 1)$[2].

3.3.1 Results and discussion

We first evaluate the effect of selectively enabling resample step with ESS mechanism. In *all* the settings, it is found that simple runs with resample step enabled always at each iteration have relatively better performance. Figure 3.1 shows the performance in terms of $\log \pi_T(\hat{u}^*_{0:t})$ of 30 independent runs ($T = 20$ and $N = 10000$) of the algorithms. All other runs with different settings of T and N concur with this finding. From here onwards, we assumes ESS mechanism is disabled unless stated otherwise.

[1]Strictly speaking, testing only increases confidence of the implementation correctness but does not prove no bug.

[2]$t + 1$ instead of t because time-step t starts with 0

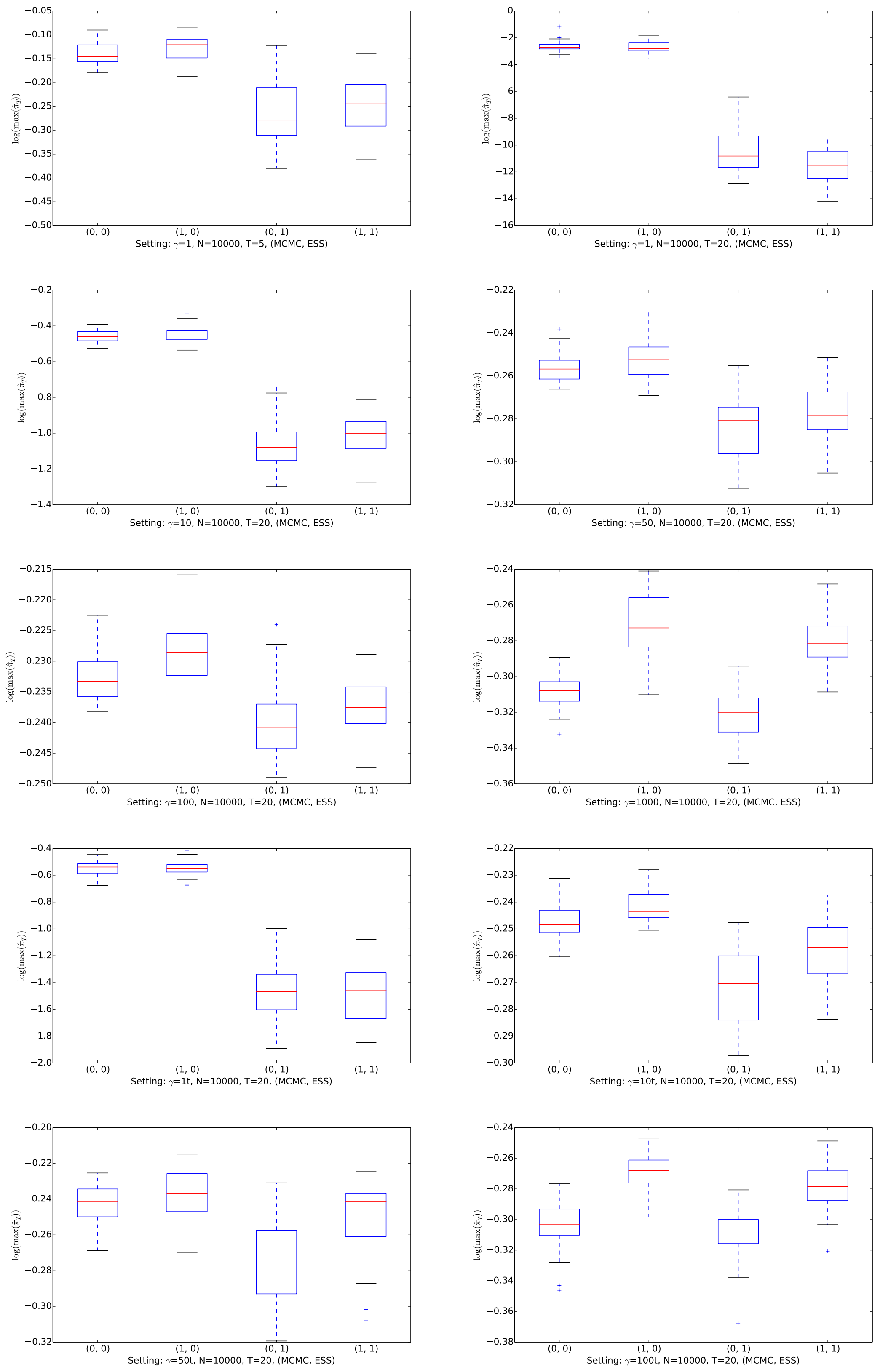

Figure 3.1: The log $\pi_T(\hat{u}^*_{0:t})$ box-plots of 30 independent runs ($T = 20$ and $N = 10000$) of the algorithms.

Moreover, in the same figure, we can see that the Resample-Move step does improve performance, albeit in most cases the improvement is marginal. The improvement is more significant when γ is set to be high. This is depicted clearly in Figures 3.2, 3.3 and 3.4, in which it presents similar box-plots of $\log \pi_T(\hat{u}^*_{0:t})$ of 30 independent runs of the algorithms for different T, N and γ settings. One possible explanation to this is that high γ settings help in optimisation by encouraging particle converge to high density area. Excessively high γ settings will lead to pre-mature convergence particles to local optimal (as one can see the $\log \pi_T(\hat{u}^*_{0:t})$ values found in high γ settings are not very good). The perturbation Resample-Move step introduces may help particles to escape from local optimal.

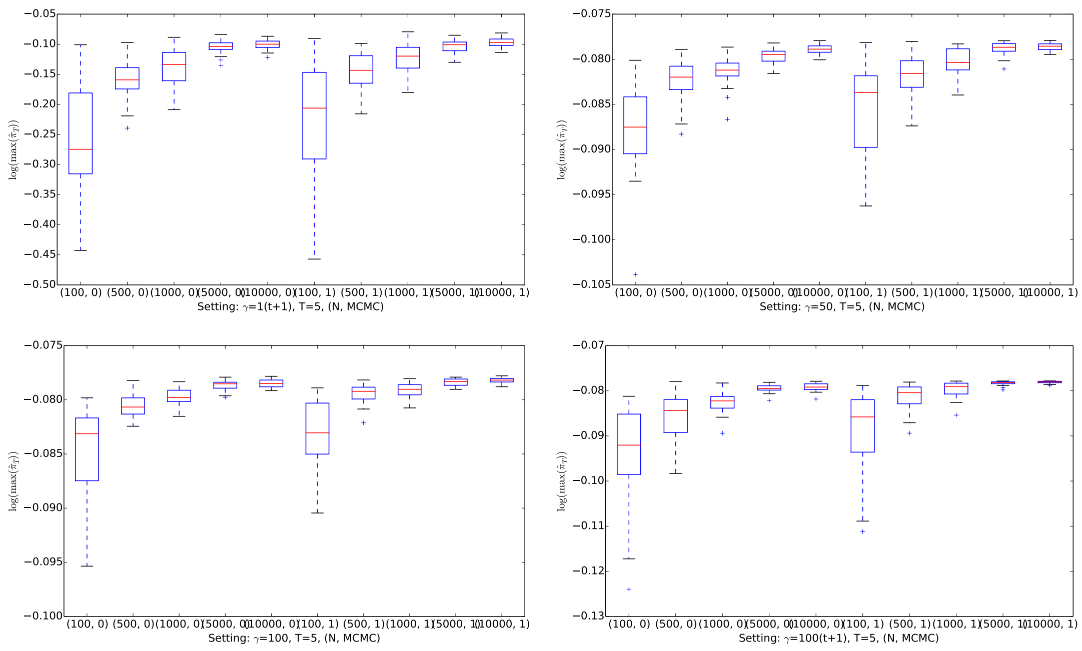

Figure 3.2: The $\log \pi_T(\hat{u}^*_{0:t})$ box-plots of 30 independent runs with various settings for $T = 5$. The 5 left/rightmost box-plots in each figure are for the runs without/with resample-move step.

In terms of the number of samples used, we can the performance improves with the number of samples used for all time periods considered as shown in Figures 3.2, 3.3 and 3.4. This is in-line with our expectation. For shorter time-period considered, the performance improvement sometimes is "saturated" even with lower number of samples, says 1000 and 5000. This suggests the optimal control parameters may have been found.

A more interesting result is to compare the performance with different γ settings. Let first consider fixed γ settings. Excessive high γ settings lead to pre-mature con-

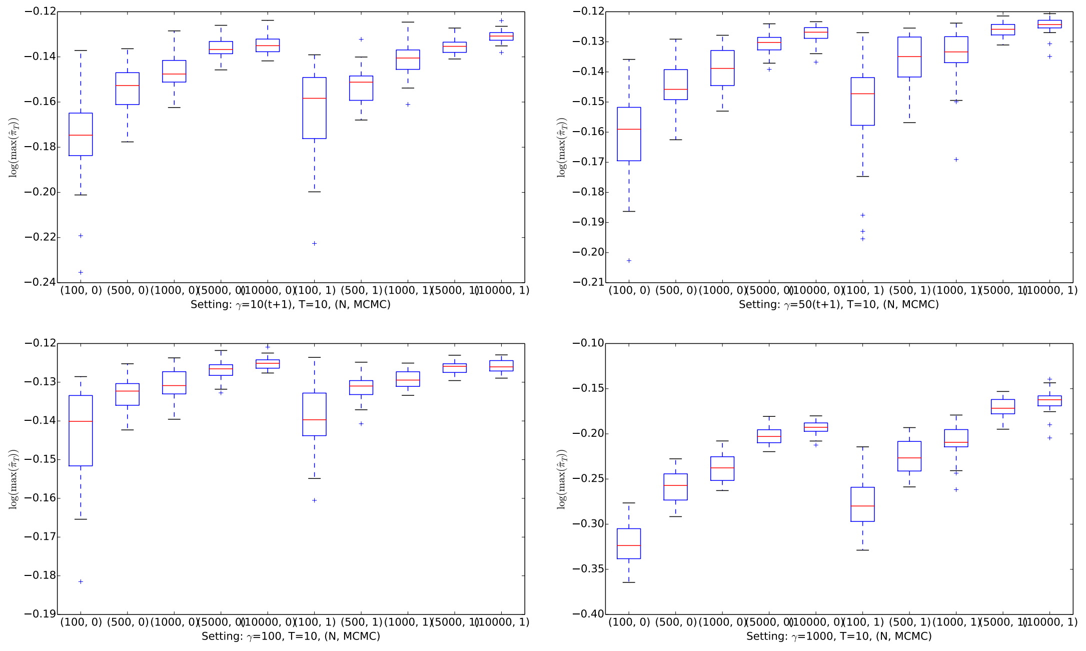

Figure 3.3: The $\log \pi_T(\hat{u}^*_{0:t})$ box-plots of 30 independent runs with various settings for $T = 10$. The 5 left/rightmost box-plots in each figure are for the runs without/with resample-move step.

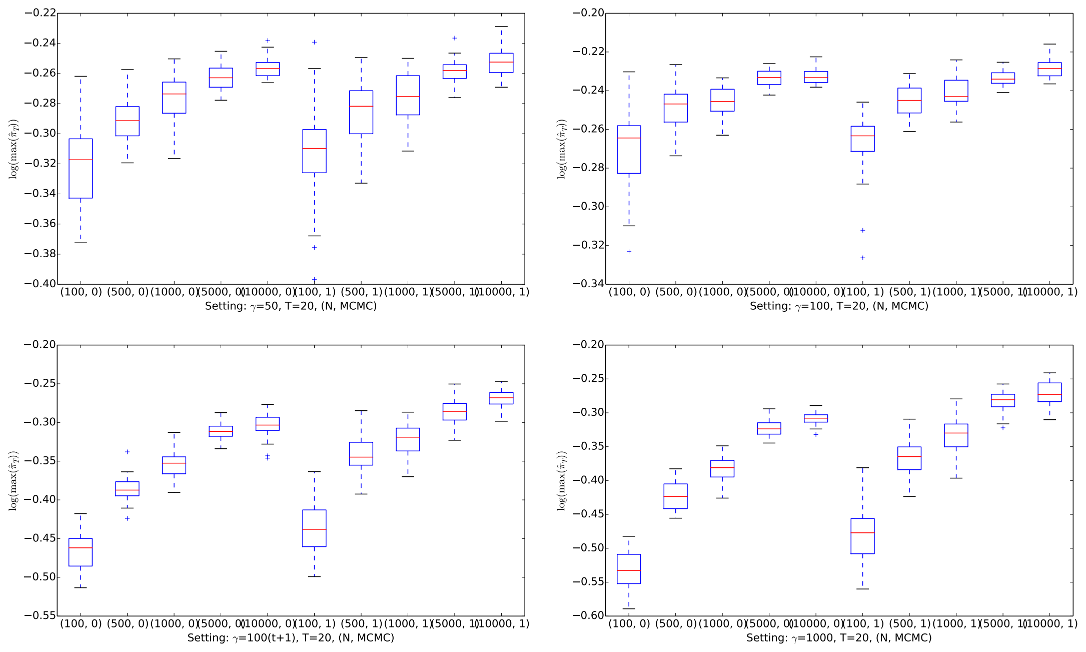

Figure 3.4: The $\log \pi_T(\hat{u}^*_{0:t})$ box-plots of 30 independent runs with various settings for $T = 20$. The 5 left/rightmost box-plots in each figure are for the runs without/with resample-move step.

vergence issue discussed earlier and stuck at local optimal. On the other hand, low γ settings also does not perform too well either (consider the process of searching for the maximum point by randomly sampling a very flat distribution). The optimal γ setting seems to lie sometimes in between. In this case, the $\gamma = 100$ setting results in the best performance.

In [17], it was suggested that using an increasing γ setting may be a good pragmatic compromise between accuracy and good mixing. We investigate further on this by setting γ to be increasing function of the form αt, where α is constant and t is the time step. The results show that the performance seems improve when theα value is low, but the result is worse when the α value is high. This suggests a capping on the maximum value on α or a more moderate in t may be more suitable. We therefore carry out two extra γ settings: $50\log(t+2)$ and $100\log(t+2)$ to investigate this and the results are shown in Figure 3.5. The results obtained are not statistically different from a simple fixed $\gamma = 100$ setting.

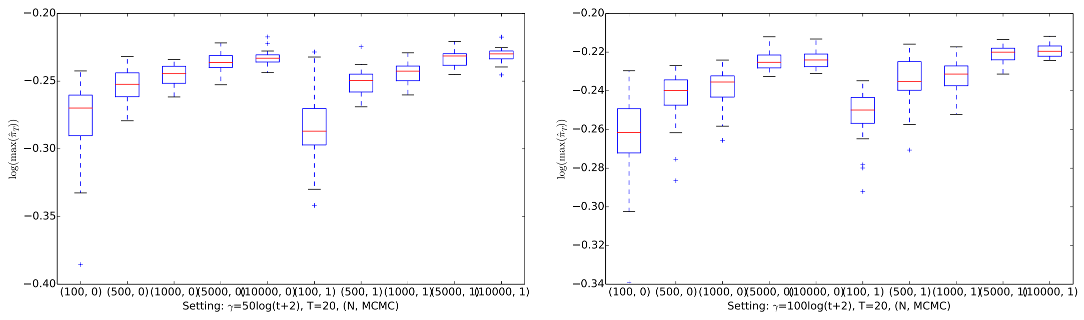

Figure 3.5: The $\log \pi_T(\hat{u}^*_{0:t})$ box-plots of 30 independent runs with four different γs settings. The leftmost/rightmost 5 box-plots in each figure are for the runs without/with resample-move step.

Lastly, we conclude this section by showing the output of the model using our estimated $u^*_{0:T}$ obtained in various runs in Figure 3.6 and their corresponding tracking errors in Figure 3.7. The results looks very promising. Except in the runs with the $\gamma = 1$ setting in which there is a lot of noise in tracking, all other runs are able to track the reference signal very close. For further details about the experiments, the implementation code and the plots for all other runs can be obtained at the on-line repository of this project at GitHub.

In the next section, we shall investigate the use of the same algorithm to track a major financial index using real-world data.

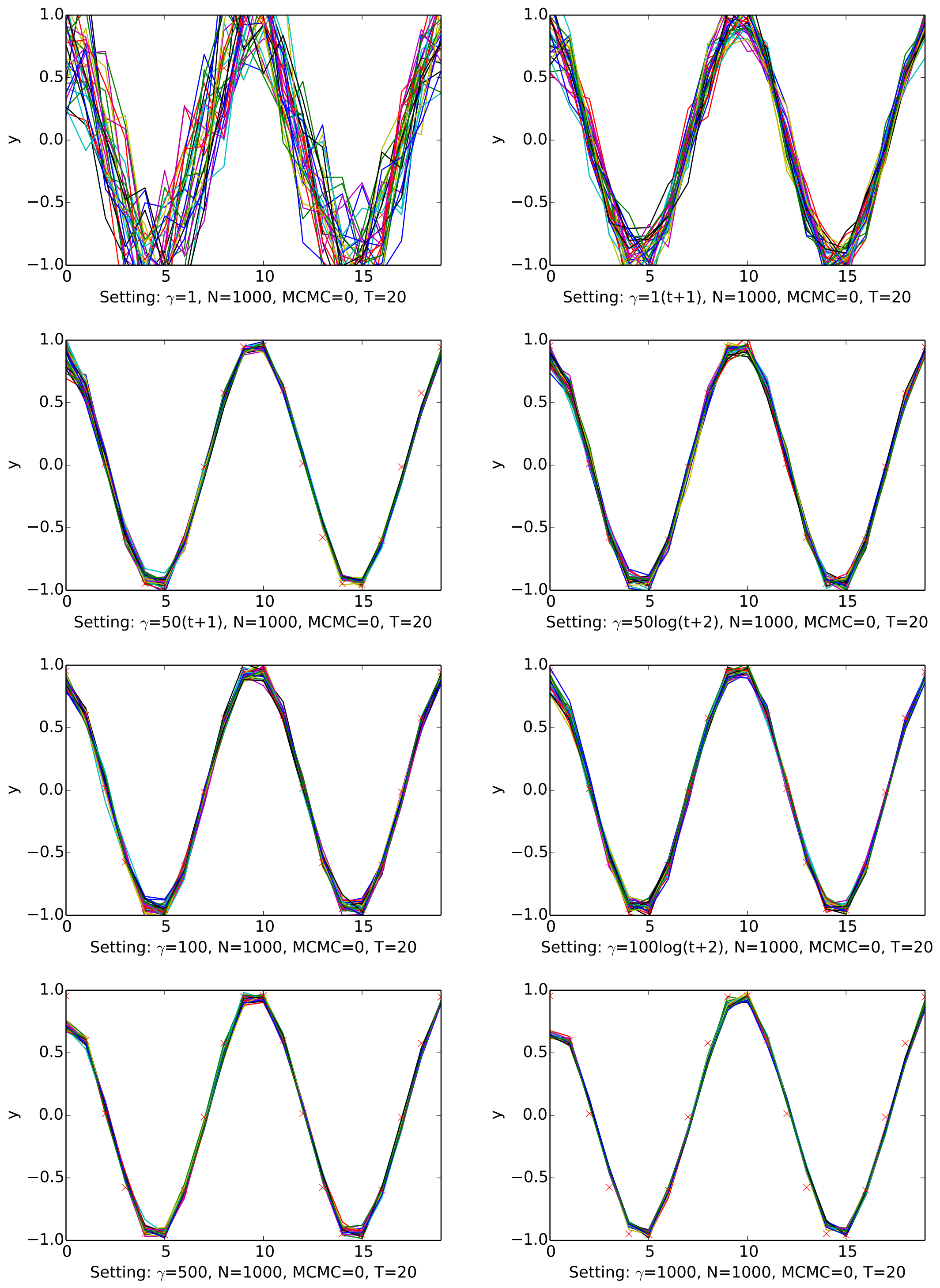

Figure 3.6: The estimated y output of the model using $u^*_{0:T}$ in 30 independent runs with various γ settings, $T = 20$, $N = 1000$ and resample-move disabled.

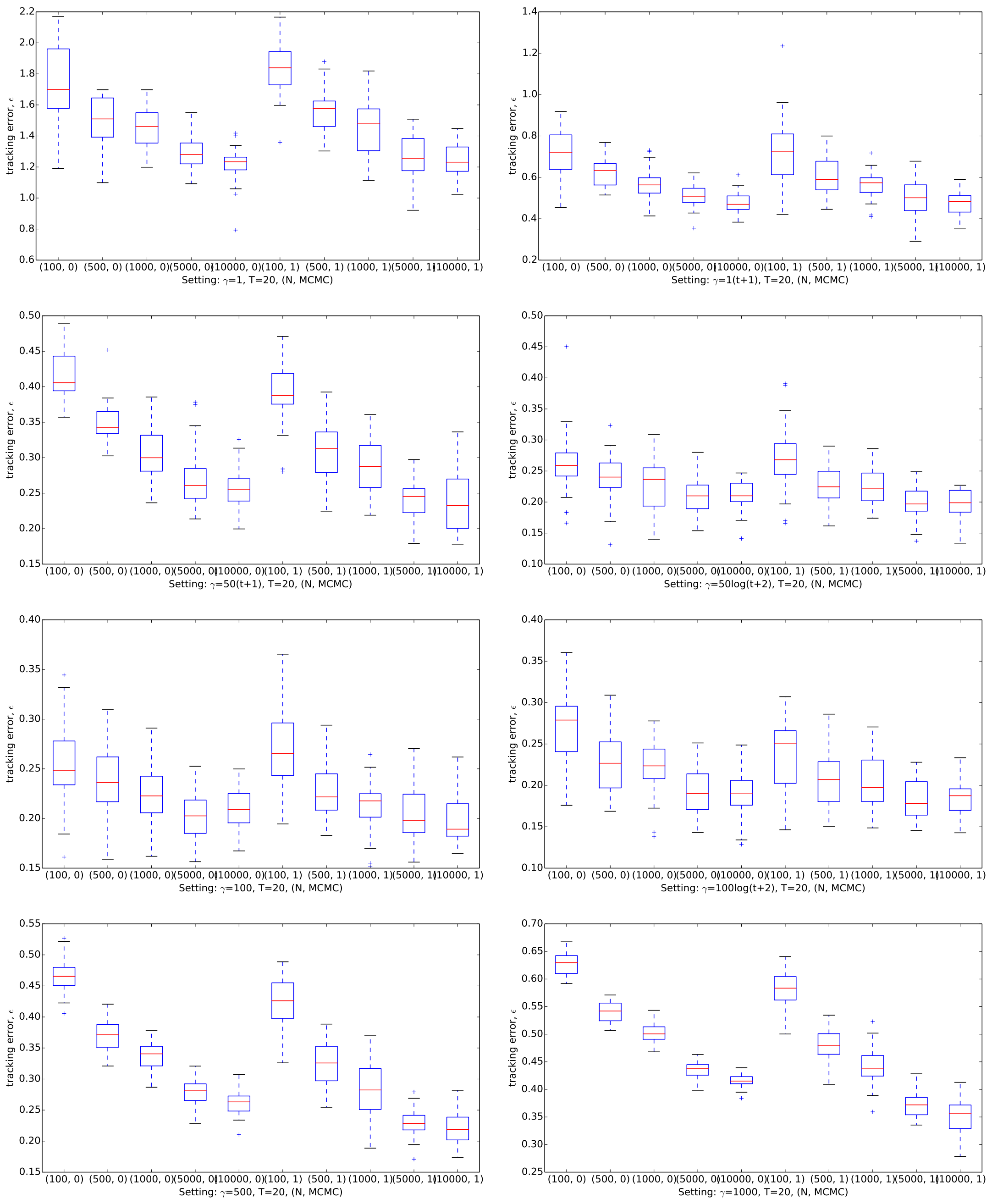

Figure 3.7: The tracking error of the model using $u^*_{0:T}$ in 30 independent runs with various γ settings, $T = 20$, $N = 1000$ and resample-move disabled.

3.4 Example 2: Tracking the DAX Index

In this example, we look at a real-world example. We look at the German's DAX (Deutscher Aktienindex) Index, which consists of 30 German's blue chip stocks listed on Frankfurt Stock Exchange as the constituents from 1st January 2014 up to 30th June 2014. It represents 80% of the aggregated prime standard's market capitalization. The DAX Index is chosen for various pragmatic reasons. It is one of the major world indices that is tracked by various index funds, it has small number of components and the data is freely accessible[1]. For further detail on the DAX Index methodology, refer [20].

3.4.1 Tracking the DAX4 sub-index

In the first experiment, we define a simple hypothetical sub-index, DAX4 consists of four stocks from the DAX Index that have the highest weights as of 2nd January 2014, namely Bayer (BAYN), Siemens (SIE), BASF (BAS) and Daimler (DAI). Then, the DAX4 index level is calculated as the simple weighted average of the adjusted close prices obtained from Yahoo Finance as follows:

$$y = \sum_{s \in \mathcal{S}} w_s x_s \tag{3.17}$$

where y is the index level, $\mathcal{S}$ consists of the four stocks and w_s and x_s are the weight and price of stock s respectively. Here the weights of these four stocks are assigned to be 0.4, 0.3, 0.2 and 0.1 respectively. The adjusted close price of each stock along with the calculated index level are shown in Figure 3.8.

The portfolio optimisation problem is formulated as such y is viewed as the target reference level that a portfolio manager attempt to replicate as close as possible by changing the holding position on each component stocks in $\mathcal{S}$, at the same time, minimize the transaction cost incurred in position changes.

To solve this problem using the SMC, the following state space modelling is used:

$$X_t = X_{t-1} + F_t(U_t) + \mu_{t_0} + \Sigma_{t_0} \tag{3.18}$$

$$Y_t = U_t^T X_t + 0.1V_t, \ V_t \sim \mathcal{N}(0, I) \tag{3.19}$$

where $\{X_t\}_{t\geq 0}$ is a vector of stock price processes modelled as Arithmetic Brownian Motion with drift, $\{U_t\}_{t\geq 0}$ is a vector of control input processes, each represents the

[1]Actually, we first looked at Dow Jones Industrial Average (DJIA) Index obtained. It was later found some of the data is not publicly available. Having said that, the preliminary findings concur with the findings we have with the DAX Index reported here.

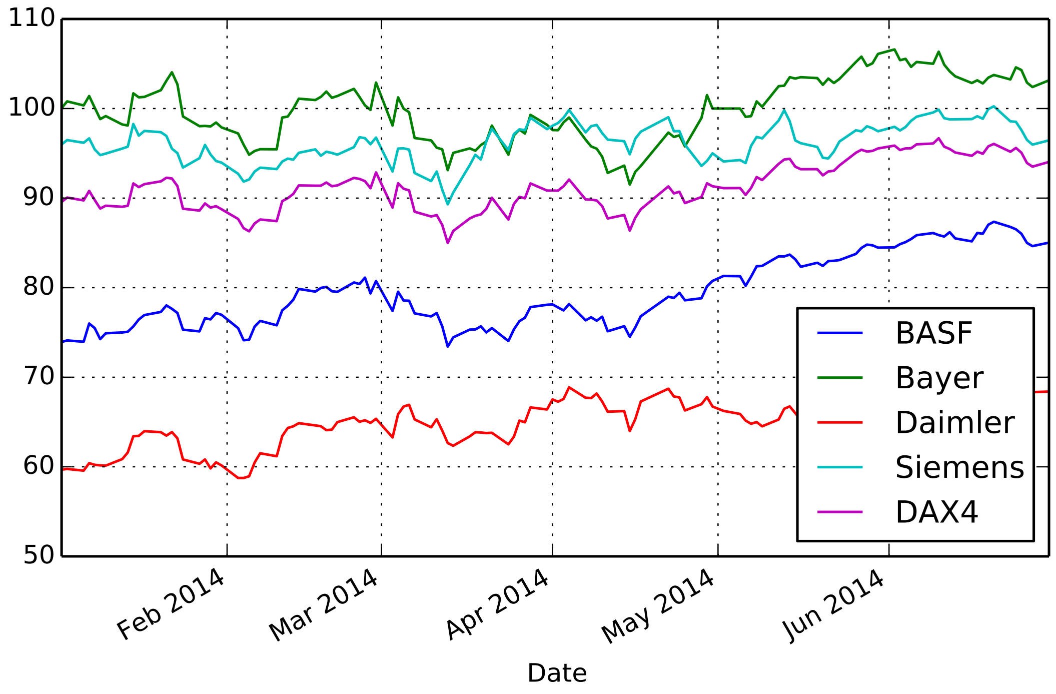

Figure 3.8: The adjusted close price of the 4 stocks and the calculated DAX4 index level.

position we have for each stock, $F_t(u_t)$ can be viewed as the market impact on price due to position changes and is set to be $0.1u_t$ here, μ_{t_0} and Σ_{t_0} are vector of the estimated mean and the estimated covariance matrix of the stock returns and $\{Y_t\}_{t\geq 0}$ is the process represents the index level. The values of μ_t and Σ_t here are estimated using Exponential Weighted Moving Average (EWMA) approach with the decay rate, λ, set to 0.94, which can be easily calculated in a recursion form as follows:

$$P_t = \lambda P_{t-1} + (1 - \lambda) Q_t \tag{3.20}$$

where P_t is the EWMA estimate and Q_t is the new observation at time step t. It is obvious from these equations that the EWMA estimate at time t depends on all the preceding estimates at time s, where $s < t$. To ensure the quality of EWMA estimate, a 6 months warm up period is used, i.e., the EWMA estimate is calculated from 1st

July 2013 onwards. The estimates of μ_{t_0} and Σ_{t_0} obtained are as follows:

$$\mu_{t_0} = \begin{pmatrix} 0.00049655 \\ 0.00215094 \\ 0.00156870 \\ 0.00203186 \end{pmatrix} \Sigma_{t_0} = \begin{array}{cc} \begin{array}{cccc} BAYN & SIE & BAS & DAI \end{array} & \\ \begin{pmatrix} 0.00011122 & 0.00009616 & 0.00010233 & 0.00009828 \\ 0.00009616 & 0.00013975 & 0.00010037 & 0.00008377 \\ 0.00010233 & 0.00010037 & 0.00014317 & 0.00010481 \\ 0.00009828 & 0.00008377 & 0.00010481 & 0.00013398 \end{pmatrix} & \begin{array}{c} BAYN \\ SIE \\ BAS \\ DAI \end{array} \end{array} \tag{3.21}$$

It is worth here to re-iterate here that this model is just a means to an end to demonstrate the idea. Other sophisticated models, e.g., Geometric Brownian Motion with drift, Jump diffusion model, etc., are possible, perhaps better. This model is again an instance of the model class defined in (3.2).

Using this model, we can write the reward function as follows:

$$J(u_{1:T}, y^{ref}_{1:T}, x_0) = \mathbb{E}_{x_0}\left[-\frac{1}{2}\sum_{t=1}^{T}\left(||y^{ref}_t - C_t(u_t)x_t||^2_{(D_t(u_t)D_t(u_t)^T)^{-1}} + ||u_t - u_{t-1}||^2_{L_t}\right)\right] \tag{3.22}$$

Here, we will define $L = 0.1\,\text{diag}(4,3,2,1)$ and also impose additional constraints on each component of u is in the range $[0, 0.5]$. The idea behind these choices is to construct a scenario in which there is different amount of transaction cost involved in changing positions on each component and a constraint on the maximum amount on each component one portfolio can have. This time, we will use the SMC algorithm to target π directly, i.e, fixed $\gamma = 1$ setting. For the importance sampling step, $U_{1,n}$ is sampled uniformly from the range bounded the constraints mentioned.

3.4.1.1 Results and discussion

The simulation is carried out for $N = 10000$ with no resample-move step implemented and the results are shown in Figure 3.9. The output y produced by the model using the estimated $\hat{u}^*_{0:T}$ is tracking the reference index level very close with the average annualised[1] tracking error of 0.014824 over the 30 runs tracking error. However, the estimated set of control parameter $\hat{u}^*_{0:T}$ which represents the holding position of each stock is not very stable.

To reduce the number of changes in holding positions, the value of L matrix can be scaled up. To demonstrate this, the same experiment is re-run with $L = 500\,\text{diag}(4,3,2,1)$. The results are shown in Figure 3.10. It is obvious the estimated set of control parameters $\hat{u}^*_{0:T}$ produced in this case is much more stable. However, this comes with a price on the tracking error. The average annualised tracking error with this setting is increased to 0.038650 over the 30 runs.

[1]the annualisation factor is set to be $\sqrt{252}$ as the daily return is measured here.

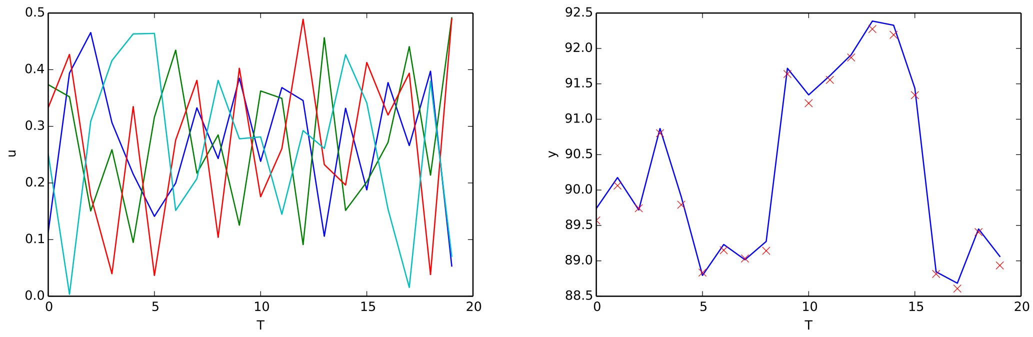

Figure 3.9: The estimated $\hat{u}^*_{0:T}$ and the corresponding y output of the model in one of the 30 independent runs for target reference, y^{ref} set to be the calculated DAX4 index level and $L = 0.1\,\text{diag}(4, 3, 2, 1)$.

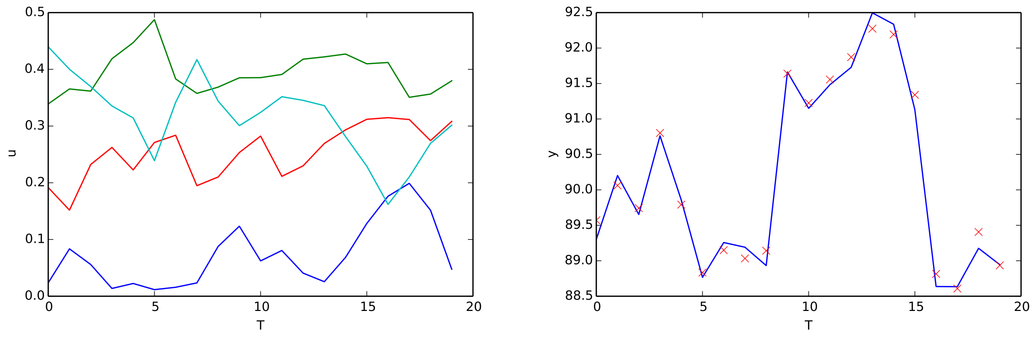

Figure 3.10: The estimated $\hat{u}^*_{0:T}$ and the corresponding y output of the model in one of the 30 independent runs for target reference, y^{ref} set to be the calculated DAX4 index level and $L = 500\,\text{diag}(4, 3, 2, 1)$.

3.4.2 Tracking the DAX index

Given the results look promising, we attempt to track the original DAX index which consists of 30 components using the similar model with minimal changes as follows:

$$X_t = X_{t-1} + F_t(U_t) + \mu_{t_0} + \Sigma_{t_0} \tag{3.23}$$

$$Y_t = 30U_t^T X_t + 0.1V_t,\ V_t \sim \mathcal{N}(0, I) \tag{3.24}$$

with the same parameter settings as before except the following:

1. The target reference y^{ref} is the daily close level of the DAX index.
2. Dimensionality of U and X are increased from 4 to 30.
3. The L matrix is set to be $50I_{30\times 30}$, i.e., the transaction cost of each component is equal.

As mentioned, sometimes it can be more efficient (not only in terms of computational), but also in terms of transaction cost and management perspective to the index with only a subset of its constituents.

This can be achieved with minimal change on the model proposed. We choose the top 6 highest weighted constituents of DAX index by including Allianz SE (ALV) and SAP SE (SAP), in which they all together constitute more than 50% of the weights for the DAX index level as of 2nd January 2014. Using only these 6 component stocks, $L = 50I_{6\times 6}$ and the DAX index level as the target reference y^{ref}, the experiment is re-run for 30 times.

3.4.2.1 Results and discussion

As before, the estimated $\hat{u}^*_{0:T}$ and the corresponding y output of the model produced by the estimated $\hat{u}^*_{0:T}$ in one of the 30 independent runs for full and partial replication are shown in Figure 3.11 and 3.12 respectively.

In the full replication case, the average annuualised tracking error over 30 runs is $0.026493XX$. However, the computational cost increases significantly in this experiment due to the dimensionality increases in U and X. Using the computation cost of the previous $DAX4$ experiment as the baseline model, the computational cost for this experiment is ≈ 7.2 times more for each iteration.

In the partial replication case, the average annualised tracking error over 30 runs is 0.026493. In terms of computational cost, the relative cost factor in comparison to the baseline is ≈ 1.1. In other words, this setting is $6.5\times$ speed-up in comparison to the full replication setting.

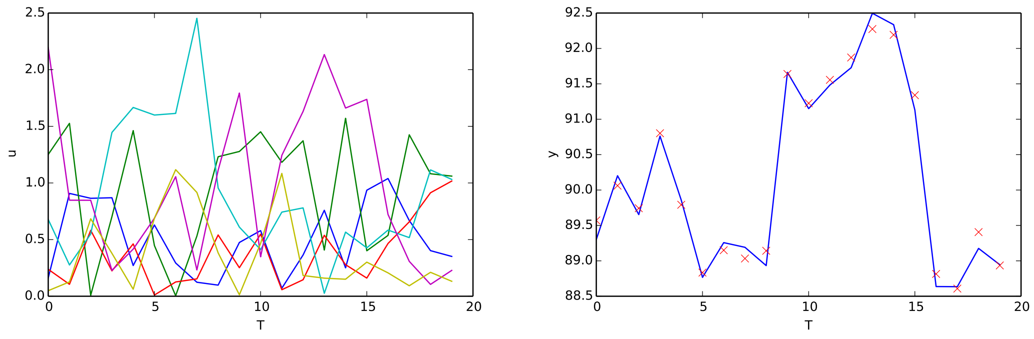

Figure 3.11: The estimated $\hat{u}^*_{0:T}$ and the corresponding y output of the model in one of the 30 independent runs for target reference, y^{ref} set to be the calculated DAX index level with full replication.

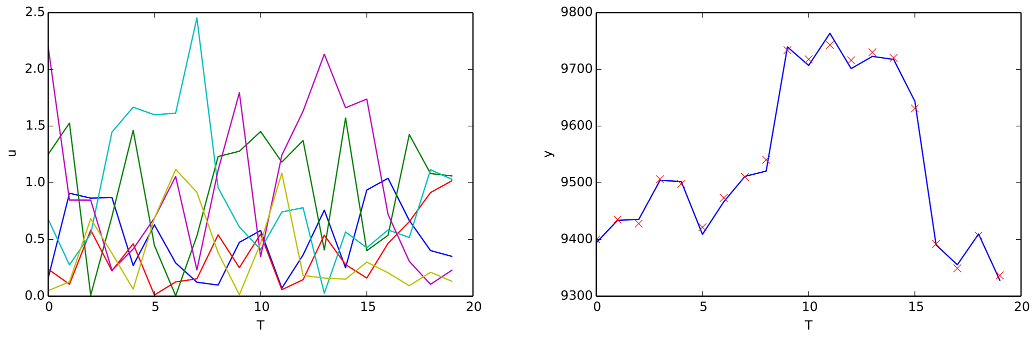

Figure 3.12: The estimated $\hat{u}^*_{0:T}$ and the corresponding y output of the model in one of the 30 independent runs for target reference, y^{ref} set to be the calculated DAX index level with partial replication using 6 component stocks.

3.5 Model Predictive Control

Up to this point, the discussion has been focussed on predicting the optimal (as outlined in the reward function) strategy for the time period $0 : T$. However, as time progress, there are more realisation of the reference signal $y_{t'}^{ref}, t' < T$ obtained. It makes sense to take this information into account in the optimisation process. In this section, we shall demonstrate how the Model Predictive Control (MPC) algorithm can be incorporated easily for this problem.

MPC is an established technique in control theory which has achieved significant success in many different domains. It is a kind of receding horizon control algorithm, in which at time step t, the model is used to optimise the controls for the process, $u_{t:t+W}$ for a finite horizon for $t : t + W$. However, only the first set of controls, i.e., u_t is applied to the system. At time $t + 1$, the whole optimisation process is repeated for the same horizon, but shifted by 1 time step until $t = T$. Refer [21] for further details on MPC.

In this problem, the SMC algorithm can be executed on a daily basis after the market close on day t to predict the optimal $u^*_{t:t+T}$, i.e., $W = T$, but only the first set of controls, u_t is applied. The same SMC algorithm is repeated on the next day with a shifted time horizon and so on. To demonstrate this idea, we repeat the same experiment on the DAX index with partial replication using 6 components with the highest weights as before.

The results are shown in Figure 3.13. We can see a significant improvement of performance with only tracking error of 1. over the 30 runs. Having said that, the computational requirement here is *much* higher. Whilst the cost of each iteration remains the same, for a finite horizon, there is WT iterations as opposed to T iterations.

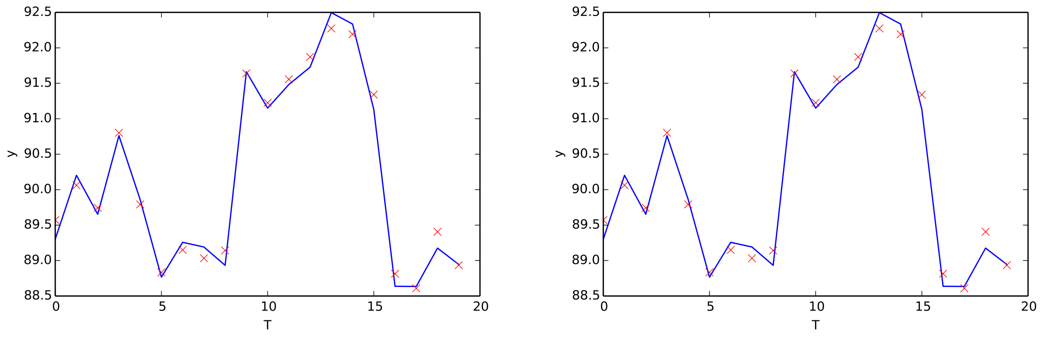

Figure 3.13: The y output of the model produced by the estiamted $\hat{u}^*_{0:T}$ in one of the 30 independent runs with the target reference, y^{ref} set to be the calculated DAX index level using full replication and partial replication methods.

3.6 Conclusions

This chapter presents some proof-of-concept experiments that have been carried out to validate our proposal: searching for the optimal control for a portfolio to track a target reference signal using SMC algorithms. It first presents an experiment with a oscillating wave as target reference signal and continues with an experiment with a major real-world index as target reference signal. In all cases, the results show that SMC techniques are able to search for optimal control that results an output that is able to track the reference signal.

Chapter 4

Evaluation and conclusions

The work reported in the previous chapter provides evidence to support the thesis hypothesis:

> Sequential Monte Carlo (SMC) techniques have the potential to be an effective means of searching the optimal strategy for index tracking funds.

This chapter reviews the work that has been done, evaluates the extent to which they justify the thesis hypothesis and concludes the thesis by addressing the directions for future work.

4.0.1 Thesis contributions

In the first experiment, we demonstrated the potential of SMC techniques in tracking a deterministic simple oscillating wave. The sensitivity of the techniques in terms of parameter settings and trade-off between the estimation accuracy and computational efforts are explored numerically. Several results are found. Firstly, it is found that a simple resampling step at each iteration performs better the more advanced method in which the resampling step is selectively triggered depending on the ESS value. Secondly, it is found that the improvement of resample-move step is marginal. Considering the amount of computational cost involved and the extra amount of design cost involved, this may not be worthwhile. Thirdly, it is found that setting γ as an increasing function of t does lead to better performance, albeit extra care must be paid on the amount of increment at each step to avoid pre-mature convergence to local optimal.

In the second experiment, we explore the potential of SMC techniques in searching the optimal strategy for index tracking error with constraints and minimizing the transaction costs. We adopt the same framework with slightly different model. Although this framework has been applied in various engineering domains, there is no previous work to my knowledge in the application of such techniques in portfolio tracking error optimisation. In particular, we show how index can be fully replicated or partially replicated with transaction cost taken into account. The experiment results show the SMC techniques are rather promising.

In the third experiment, we introduce the concept of model predictive control (MPC) and how it can be used together with SMC techniques in improving tracking performance. Whist this technique is nothing new in control theory, its application to this portfolio optimisation problem is novel. The experimental results do suggest that MPC indeed improve the tracking performance.

4.1 Envisaged future work

Having discussed the contributions of the thesis, we now outline numerous possible directions for future work that have been identified during the course of this research.

4.1.1 More realistic models

The Arithmetic Brownian Model with drift used in this thesis is rather simple. A possible extension work is to consider a more advanced model, e.g., Geometric Brownian Model, Jump Diffusion model, etc. Moving away from conditional Gaussian model also introduces another complication. Without the conditional Gaussian assumption, the inner Kalman Filter recursion is no longer optimal. A possible solution to this is substituting the Kalman Filter with a nested SMC algorithm. This setup is known as the SMC2 algorithm [22].

4.1.2 Parallel computation

The nested SMC setup inevitably add consideration computation requirements. A possible speed up is to parallelise the steps in SMC algorithm. This is very straightforward for all the steps, except the resampling step, which remains an interesting research topic on its own.

4.1.3 More complex financial indices

The benchmark index used in this thesis is rather simple. This can be potentially an issue. Having said that, this index is a simple, but by no means a “toy” index. It is a major financial index in the world. There are however indices with much large number of components, e.g., ≈ 1600 for MSCI World Index. This translates to a high dimensional problem in which may be difficult for the proposed model here.

A possible way to cope with this is to just track the index with partial replication using preselected subset of components, perhaps using Principle Component Analysis (PCA). Alternatively, we could use a divide and conquer approach, tracking multiple sub-indices separately. Yet, there is still much to answer here, for example:

- How to split an index into smaller components in a systematic manner?
- How to deal with index components changes during announcement and implementation?

4.2 Closing remarks

The work reported in this thesis demonstrates a considerable degree of originality supported by extensive experimentation. The case studies are necessarily limited given the limited amount of time frame. However, the results demonstrate that portfolio optimisation using Sequential Monte Carlo techniques have very considerable promise. We recommend these approaches to the research community for further investigation.

References

[1] Harry Markowitz. Portfolio Selection. *The Journal of Finance*, 7(1):77–91, 1952. 1, 20

[2] R. Tyrrell Rockafellar and Stanislav Uryasev. Optimization of Conditional Value-at-Risk. *Journal of Risk*, 2:21–41, 2000. 1

[3] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. 5

[4] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12:656–704, 2009. 5

[5] Olivier Cappé, Eric Moulines, and Tobias Ryden. *Inference in Hidden Markov Models (Springer Series in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. 7, 8, 17

[6] Rudolph Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960. 10

[7] Greg Welch and Gary Bishop. An Introduction to the Kalman Filter. Technical report, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1995. 11

[8] Eric A. Wan and Rudolph Van Der Merwe. The Unscented Kalman Filter. In *Kalman Filtering and Neural Networks*, pages 221–280. Wiley, 2001. 11

[9] Adam M Johansen, Ludger Evers, and N Whiteley. Monte Carlo Methods. *Lecture Notes*, 200, 2010. 13, 14, 15

[10] Genshiro Kitagawa. Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996. 13

[11] Jun S. Liu and Rong Chen. Sequential Monte Carlo Methods for Dynamic Systems. *Journal of the American Statistical Association*, 93:1032–1044, 1998. 13

[12] R. Douc and O. Cappe. Comparison of resampling schemes for particle filtering. In *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on.* 13

[13] Carlo Berzuini and Walter Gilks. RESAMPLE-MOVE Filtering with Cross-Model Jumps. In Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors, *Sequential Monte Carlo Methods in Practice*, Statistics for Engineering and Information Science, pages 117–138. Springer New York, 2001. 15

[14] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of IEEE*, pages 257–286, 1989. 17

[15] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society Series B*, 72(3):269–342, 2010. 19

[16] Ben Johnson, Hortense Bioy, Alastair Kellett, and Lee Davidson. On the Right Track: Measuring Tracking Efficiency in ETFs. February 2013. 21, 22

[17] P. Del Moral N. Kantas and R. Vinter. Particle methods for Stochastic Regulation: towards an application for power scheduling. Technical report, Department of Electrical and Electronic Engineering, Imperial Colege, London, March 2011. 23, 25, 26, 30

[18] P. Whittle. *Risk-sensitive Optimal Control.* Wiley-Interscience series in systems and optimization. Wiley, 1990. 23

[19] Simon Godsill, Arnaud Doucet, and Mike West. Maximum a Posteriori Sequence Estimation Using Monte Carlo Particle Filters. *Annals of the Institute of Statistical Mathematics*, 53(1):82–96, 2001. 25

[20] Factsheet DAX. Websites, June 2014. 33

[21] C. E. Garcia, D. M. Prett, and M. Morari. Model Predictive Control: Theory and Practice&Mdash;a Survey. *Automatica*, 25(3):335–348, May 1989. 39

[22] N. Chopin, P. E. Jacob, and O. Papaspiliopoulos. SMC2: An efficient algorithm for sequential analysis of state space models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(3):397–426, 2013. 42