University of New South Wales

MSc (Statistics) Project

Nonlinear Filtering for Non-Stationary Multivariate Cointegration Models

Author:
Shamin Kinathil

Supervisor:
Dr. Gareth Peters

Session 1, 2011

Abstract

The study of non-linear filtering for non-stationary multivariate cointegration models combines the disciplines of engineering and time series analysis. Within this thesis we consider different methods of filtering for multivariate cointegration models and also techniques to estimate the parameters of the models. The effectiveness of the filtering and parameter estimation techniques are investigated within a series of case studies utilising novel implementations of the filtering and estimation algorithms in Object-oriented MATLAB. Finally, multivariate cointegration model estimation techniques are applied to a set of empirical data, consisting of futures contract pairs of indices, interest rates and bonds.

Contents

List of Figures

List of Tables

Notation

The following notation is used throughout the exposition: bold type represents vector quantities, with upper case denoting random variables, and lower case their realisation. Normal type represents non-stochastic (or deterministic) quantities (vectors unless explicitly stated otherwise). The notation for scalar quantities (including scalar random variables) will be obvious from context, and are rarely used.

Notation	Definition
$t \in \mathbb{N}$	The time $t = \{1, 2 \ldots\}$
$\mathbb{R}^{tn_y}$	$\mathbb{R}^{n_y} \overset{t+1 \ times}{\times \ldots \times} \mathbb{R}^{n_y}$
$\mathbf{X}_t \in \mathbb{R}^{n_x}$	A random (state) vector at time t, with n_x components
$\mathbf{x}_t \in \mathbb{R}^{n_x}$	A realisation of the random vector $\mathbf{X}_t$
$\mathbf{Y}_{1:t} \in \mathbb{R}^{tn_y}$	The set of random vectors (observations) $\{\mathbf{Y}_{1:t}, \ldots, \mathbf{Y}_t\}$, each with n_y components.
$\mathbf{y}_{1:t} \in \mathbb{R}^{tn_y}$	The set of realisations of $\mathbf{Y}_{1:t}$, namely $\{y_1, \ldots, y_t\}$
$\hat{\mathbf{X}}_{t\|t-1} : \mathbb{R}^{(t-1)n_y} \to \mathbb{R}^{n_x}$	An estimator of $\mathbf{X}_t$ as a function of random observations $\mathbf{Y}_{1:t-1}$
$\hat{\mathbf{x}}_{t\|t-1} : \mathbb{R}^{(t-1)n_y} \to \mathbb{R}^{n_x}$	An estimator of $\mathbf{X}_t$ as a function of random observations $\mathbf{y}_{1:t-1}$
$\tilde{\mathbf{X}}_{t\|t-1} : \mathbb{R}^{(t-1)n_y} \to \mathbb{R}^{n_x}$	An estimation error associated with the estimator, defined as $\tilde{\mathbf{X}}_{t\|t-1} = \mathbf{X}_t - \hat{\mathbf{X}}_{t\|t-1}$
$\tilde{\mathbf{x}}_{t\|t-1} : \mathbb{R}^{(t-1)n_y} \to \mathbb{R}^{n_x}$	An estimation error associated with the estimator, defined as $\tilde{\mathbf{x}}_{t\|t-1} = \mathbf{X}_t - \hat{\mathbf{x}}_{t\|t-1}$
x′	x transpose
$\mathbf{V}_t \in \mathbb{R}^{n_v}$	The observation noise vector, with n_v components. Assumed to be zero mean and white i.e. $\mathbb{E}(\mathbf{V}_i \mathbf{V}_j') = \delta_{ij} R_i \quad \forall i, j \in \mathbb{N}$
$\mathbf{W}_t \in \mathbb{R}^{n_w}$	The observation noise vector, with n_w components. Assumed to be zero mean and white i.e. $\mathbb{E}(\mathbf{W}_i \mathbf{W}_j') = \delta_{ij} Q_i \quad \forall i, j \in \mathbb{N}$
$\mathbf{u}_t \in \mathbb{R}^{n_u}$	The *deterministic* control vector at time t with n_u components
$\cup$	Union
$\sim$	Distributed as
$\sum$	Summation sign
$\prod$	Product sign
$\to$	Converges to, approaches
$\xrightarrow{p}$	Converges in probability to
$\xrightarrow{d}$	Converges in distribution to
i.i.d.	Independent, identically distributed
lim	Limit
plim	Probability limit
L	Lag operator
Δ	Differencing operator
$\mathbb{E}$	Expectation
Var	Variance
Cov	Covariance, covariance matrix
$MVN(\mu, \Sigma)$	(Multivariate) normal distribution with mean (vector) μ and variance (covariance matrix) Σ
MSE	Mean squared error
MMSE	Minimum Mean Square Error
ARMA	Autoregressive moving average (process)
MA	Moving average process
VAR(p)	Vector autoregressive process of order p
VECM	Vector error correction model
$\otimes$	Kronecker product
$\circ$	Composition function
$\|M\|$	Determinant of M
M^{-1}	Inverse of M
vec	Column stacking operator
vech	Column stacking operator for symmetric matrices
$O(\cdot)$	Big Oh
$o(\cdot)$	Little Oh
$\mathbb{I}_m$	$(m \times m)$ Identity matrix
T	is the sample size
$\lfloor \cdot \rfloor$	Floor function

Chapter 1

Introduction

The study of multivariate cointegration models is an important topic within a spectrum of seemingly disparate fields, such as econometrics, political science and biology. In this thesis we focus on the application of multivariate cointegration models and their estimation to the field of econometrics. We also develop novel implementations of relevant algorithms in Object-oriented MATLAB to produce a toolbox that can be used for the estimation and filtering of multivariate cointegration models.

Cointegration is an important part of econometric studies, due to the almost inherent nature of non-stationarity amongst empirical time series models. Cointegrating relationships have been found within many areas of finance, including spot and future prices for a given commodity or asset, the ratio of relative prices and an exchange rate and between equity prices and dividends [2]. By utilising cointegration models, the components of a time series model are allowed to deviate from their inherent relationships in the short term, but retain their long term associations. The long term associations between certain financial assets have in turn, been used to design of trading strategies and optimise portfolios.

The representing multivariate cointegration models in state space form, the study of cointegration can be linked to filtering, a traditional topic in control engineering. Filtering is concerned with elucidating the state of a system given noisy measurements. From an econometric viewpoint, filtering can be used to determine the state of the various components of the multivariate cointegration model in an on-line setting. Filters come in many forms and each places a different set of restrictions on the dynamics of the system they estimate.

This thesis is divided into many chapters. Chapter 2 reviews multivariate cointegration models, chapter 3 investigates filtering for multivariate cointegration models. Chapter 4, applies the information presented in the preeding chapters to a set of empirical data. The final chapter, chapter 5, concludes the thesis and presents avenues for further research.

Chapter 2

Multivariate Cointegration Models

The Wold Decomposition theorem [3] states that a single stationary time series with no deterministic components has an infinite Moving Average (MA) representation. This representation can in turn, be generally approximated by a finite Autoregressive Moving Average (ARMA) process [4]. Financial time series are rarely stationary in practice and must be transformed before they can be considered stationary processes. Two common transformations are differencing and cointegration.

A series with no deterministic component which has a stationary, invertible, ARMA representation after differencing d times, is said to be integrated of order d, denoted $\mathbf{X}_i \sim I(d)$ [4]. In practice, many financial variables contain a unit root, and are thus integrated of order one.

The phenomenon of spurious regressions was first observed by Yule in 1926 [5] and in econometrics by Granger and Newbold [6]. Spurious regressions can arise when apparently meaningful regression relationships are found between variables that are in fact independent. The results of such a regression can lead to inefficient estimates, sub-optimal forecasts and invalidate significance tests [6]. An often used technique to circumvent problems with spurious regression is to utilise a cointegrating regression.

This chapter explores the idea of multivariate financial time series through the veil of cointegration. The phenomenon of cointegration is defined in Section (2.1) and its use within different financial applications are introduced. Different representations of cointegration models are covered within Section (2.2), including Phillip's triangular representation, Stock and Watson's common trends representation [7] and the Vector Error Correction Model (VECM) of Engle and Granger [4]. Methods for estimating the parameters of multivariate cointegration models are presented in Section (2.3), including Unrestricted Least Squares (ULS), the Engle Granger Method,

Dynamic Ordinary Least Squares (DOLS), the Canonical Correlation Regression method and the Johansen Maximum Likelihood method. A thorough derivation of the Johansen Maximum Likelihood Estimator is also presented.

In the final two sections, we conduct two case studies, the first of which investigates the efficacy of the Johansen Maximum Likelihood method when applied to multivariate cointegration models in the VECM form, given in Section (2.4). In the second case study, given in Section (2.5), the ability of the Johansen Maximum Likelihood method to estimate the cointegration rank of a VECM is deduced.

2.1 Cointegration

The concept of cointegration was introduced by Engle and Granger in 1987 [4]. To motivate the discussion of cointegration, we begin by assuming that there exist n time series, $\mathbf{X}_{i,t}$ where $i = 1, \ldots, n$, each integrated of the same order d. If a linear combination of the series,

$$Z_t = \sum_{i=1}^{n} \beta_i \mathbf{X}_{i,t}, \tag{2.1}$$

has an order of integration $r < d$, then the series are said to be cointegrated. The linear combination shown in Equation (2.1) is called a *cointegrating relationship* [8]. Given n time series, there can be from 0 to at most n - 1 cointegrating relationships. A cointegrating relationship may be seen as a long-term equilibrium phenomenon which allows the cointegrating variables, the $\mathbf{X}_{i,t}$, to deviate from their relationships in the short term, but retain their long term associations.

Cointegrating relationships are found within many areas of finance, including spot and future prices for a given commodity or asset, the ratio of relative prices and an exchange rate and between equity prices and dividends [2]. The relationship between spot and futures prices is to be expected, since they represent prices for the same underlying asset at different points in time. As a result, both prices will be affected by new information in similar ways. The long-run relationship between spot and futures prices can be expressed by the *cost of carry* model, see [9]. Other applications of cointegration in finance is within the design of trading strategies and portfolio optimisation, see [10].

2.2 Representation

Three different specifications of a cointegrated system have been used extensively in the literature on the representation, estimation and inference of cointegrated systems: Phillip's triangular representation, Stock and Watson's common trends representation [7] and the Vector Error

Correction Model (VECM) [11]. The three representations are investigated in the following sections.

2.2.1 Triangular Representation

The triangular representation was first introduced by Phillips in 1991 [12]. The innovation behind the triangular representation is to partition an n-dimensional, $I(1)$ time series vector, $\mathbf{X}_t$ and ϵ_t, an n_x-dimensional stationary time series with non-singular covariance matrix Σ, into subvectors, n_{x1} and n_{x2} where $n_x = n_{x1} + n_{x2}$. The representation assumes that the cointegrated system is given by,

$$\begin{aligned} \mathbf{X}_{1,t} &= \beta' \mathbf{X}_{2,t} + \epsilon_{1,t}, \\ \Delta \mathbf{X}_{2,t} &= \epsilon_{2,t}, \end{aligned}$$

where,

β: $\mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ is a matrix of coefficients,
$\mathbf{X}_{1,t} \in \mathbb{R}^{n_{x1}}$,
$\Delta \mathbf{X}_{2,t} \in \mathbb{R}^{n_{x2}}$,
$\epsilon_{1,t} \in \mathbb{R}^{n_{x1}}$,
$\epsilon_{2,t} \in \mathbb{R}^{n_{x2}}$.

2.2.2 Common Trends

The relationship between cointegration and common trends was first elucidated by Stock and Watson in 1988 [7]. In the Common Trends representation, an n_x-dimensional cointegrated process $\mathbf{X}_t$, with $r < n_x$ cointegrating relationships, can be represented formally in terms of $n_x - r$ integrated series, called *common trends*, plus stationary components [7]. The Common Trends representation is given by,

$$\mathbf{X}_{i,t} = \sum_{j=1}^{n_x - r} \gamma_j U_{j,t} + \epsilon_{i,t},$$

where,

$\mathbf{X}_{i,t} \in \mathbb{R}^{n_x}$ is the i^{th} cointegrated process at time t,
γ_j is the j^{th} loading matrix,
$U_{j,t}$ is the j^{th} common trend,
$\epsilon_{i,t} \in \mathbb{R}^{n_x}$ is the stationary disturbance.

2.2.3 Vector Error Correction Model

The Error Correction Model (ECM) formulation created by Engle and Granger in 1987 is directly derived from the Granger Representation Theorem [4]. The theorem states that a multivariate integrated process is cointegrated if and only if it can be represented in the ECM form with appropriate restrictions. The converse is also held to be true. Precursors to the ECM formulation utilised by Engle and Granger include those of Phillips [13], who introduced the concept of error correction in econometrics, Sargan [14], who introduced different methods to estimating structural equations with autocorrelated errors and finally Hendry [15].

An n_x-dimensional Vector Autoregressive process of order p, VAR(p), can be represented in the Error Correction Model form as follows,

$$\Delta \mathbf{X}_t = \Pi \mathbf{X}_{t-1} + \sum_{i=1}^{p-1} \Gamma_i \Delta \mathbf{X}_{t-i} + \phi D_t + \epsilon_t,$$

where,

$\Delta \mathbf{X}_t \in \mathbb{R}^{n_x}$	$= \mathbf{X}_t - \mathbf{X}_{t-1}$,
$\Pi : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the long-run multiplier matrix,
$\Gamma_i : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is i^{th} the lag matrix,
$\Phi : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	,
$D_t \in \mathbb{R}^{n_x}$	is a vector of deterministic terms, defined as a polynomial in time t,
$\epsilon_t \in \mathbb{R}^{n_x}$	is independent identically distributed multivariate, correlated errors .

The term in levels, Π, can be placed at any $\mathbf{X}_{t-i}$. The appeal of the VECM formulation is that it combines flexibility in dynamic specification with desirable long-run properties [16], which are mentioned below.

The cointegration properties of the VECM model depend on the rank r of the long-run multiplier matrix Π. If $r = 0$, then the VECM model does not exhibit any cointegration relationship and it can be estimated as a stable process in first differences. If $r = n_x$, that is, if the matrix Π is of full rank, then the VAR(p) model itself is stable and can be estimated as a stationary process. If, however, the rank r is intermediate, $0 < r < n_x$, then the VECM process exhibits cointegration. In this case, we can write the matrix Π as the product $\Pi = \alpha\beta'$ where both α and β are $n_x \times r$ matrices of rank r. The r columns of the matrix β are the cointegrating vectors of the process. The $\beta' \mathbf{X}_t$ reflect common trends while α contains the loading factors of the common trends. Therefore, a cointegrated VECM can be represented by,

$$\Delta \mathbf{X}_t = \alpha\beta' \mathbf{X}_{t-1} + \sum_{i=1}^{p-1} \Gamma_i \Delta \mathbf{X}_{t-i} + \phi D_t + \epsilon_t, \tag{2.2}$$

where,

$\alpha : \mathbb{R}^{n_x} \to \mathbb{R}^r$ contains the loading factors for the common trends,
$\beta : \mathbb{R}^{n_x} \to \mathbb{R}^r$ contains the cointegrating vectors.

It is important to note that the decomposition of the long-run multiplier matrix Π into $\alpha\beta'$ is not unique. In fact, for every non-singular matrix $Q : \mathbb{R}^r \to \mathbb{R}^r$, we can define $\alpha^\star = \alpha Q'$ and $\beta^\star = \beta Q^{-1}$ and get $\Pi = \alpha^\star \beta^{\star'}$. This shows that the cointegration relations are not unique [17].

If cointegration exists, the VECM representation will generate better forecasts than the corresponding representation in first-differenced form, particularly over medium and long-run horizons. This is because under cointegration, Z_t, Equation (2.1), will have finite forecast error variance, whereas any other linear combination of the forecasts of the individual series in $\mathbf{X}_t$ will have infinite variance [18]. An example application of the VECM representation is [2] and [19], where the VECM representation was used to model changes in the log of a stock index.

2.3 Parameter Estimation

The methods for estimating the parameters in a multivariate cointegration model can be split into two brances: Least Squares estimators and Maximum Likelihood estimators. Examples of Least Squares estimation methods include, Unrestricted Least Squares (ULS), the Engle Granger Method, Dynamic Ordinary Least Squares (DOLS) and the Canonical Correlation Estimator, each of which will be explored in the proceeding sections. The Maximum Likelihood estimation method that will be explored further is the Johansen Maximum Likelihood method.

2.3.1 Unrestricted Least Squares Method

To motivate the discussion of the Unrestricted Least Squares (ULS) method for parameter estimation of Vector Error Correction Models (VECM), we begin by considering a VECM of order p without deterministic terms, given by,

$$\Delta \mathbf{X}_t = \Pi \mathbf{X}_{t-1} + \sum_{i=1}^{p-1} \Gamma_i \Delta \mathbf{X}_{t-i} + \epsilon_t, \quad \epsilon_t \sim N(0, \Sigma_u) \tag{2.3}$$

where the notation has been previously defined in Section (2.2.3). It is convenient to transform the VECM shown in Equation (2.3) into matrix notation, as follows,

$$\Delta \mathbf{Y} = \Pi \mathbf{X}_{-1} + \Gamma \Delta \mathbf{X} + E, \tag{2.4}$$

where,

$$\begin{aligned}
\Delta \mathbf{Y} &= [\Delta \mathbf{X}_1, \ldots, \Delta \mathbf{X}_t], \\
\mathbf{X}_{-1} &= [\mathbf{X}_0, \ldots, \mathbf{X}_{t-1}], \\
\Gamma &= [\Gamma_1, \ldots, \Gamma_{p-1}], \\
\Delta \mathbf{X} &= [\Delta \mathbf{X}_{t-1}, \ldots, \Delta \mathbf{X}_{t-p+1}]', \\
E &= [\epsilon_1, \ldots, \epsilon_T].
\end{aligned}$$

The Least Squares estimation of the matrix parameters under the formulation, shown in Equation (2.4), can be shown to be [17],

$$\left[\hat{\Pi} : \hat{\Gamma}\right] = \left[\Delta \mathbf{Y} \mathbf{X}'_{-1} : \Delta \mathbf{Y} \Delta \mathbf{X}'\right] \begin{bmatrix} \mathbf{X}_{-1}\mathbf{X}'_{-1} & \mathbf{X}_{-1}\Delta \mathbf{X}' \\ \Delta \mathbf{X}\mathbf{X}'_{-1} & \Delta \mathbf{X}\Delta \mathbf{X} \end{bmatrix}, \tag{2.5}$$

$$\hat{\Sigma}_u = (T - n_x p)^{-1} \left(\Delta \mathbf{Y} - \hat{\Pi}\mathbf{X}_{-1} - \hat{\Gamma}\Delta \mathbf{X}\right) \left(\Delta \mathbf{Y} - \hat{\Pi}\mathbf{X}_{-1} - \hat{\Gamma}\Delta \mathbf{X}\right)'. \tag{2.6}$$

Asymptotic Properties

The asymptotic properties of the Least Squares estimator, shown in Equations (2.5) and (2.6), respectively, are consistent and have a central limit theorem given by [17],

$$\sqrt{T}\text{vec}\left(\left[\hat{\Pi} : \hat{\Gamma}\right] - [\Pi : \Gamma]\right) \xrightarrow{d} N(0, \Sigma_{co}),$$

where,

$$\begin{aligned}
0\ \Sigma_{co} &= \left(\begin{bmatrix} \beta & 0 \\ 0 & I_{n_x p - n_x} \end{bmatrix} \Omega^{-1} \begin{bmatrix} \beta' & 0 \\ 0 & I_{n_x p - n_x} \end{bmatrix}\right) \otimes \Sigma_u, \\
\Omega &= p\lim \tfrac{1}{T} \begin{bmatrix} \beta' \mathbf{X}_{-1}\mathbf{X}_{-1}\beta & \beta' \mathbf{X}_{-1}\Delta \mathbf{X}' \\ \Delta \mathbf{X}\mathbf{X}'_{-1}\beta & \Delta \mathbf{X}\Delta \mathbf{X} \end{bmatrix}.
\end{aligned}$$

The matrix $\left(\begin{bmatrix} \beta & 0 \\ 0 & I_{n_x p - n_x} \end{bmatrix} \Omega^{-1} \begin{bmatrix} \beta' & 0 \\ 0 & I_{n_x p - n_x} \end{bmatrix}\right)$ is consistently estimated by,

$$T \begin{bmatrix} \mathbf{Y}_{-1}\mathbf{Y}'_{-1} & \mathbf{Y}_{-1}\Delta \mathbf{X}' \\ \Delta \mathbf{X}\mathbf{Y}'_{-1} & \Delta \mathbf{X}\Delta \mathbf{X} \end{bmatrix}^{-1},$$

and $\hat{\Sigma}$ is a consistent estimator of Σ_u.

An interesting point to note is that if the cointegrating rank $r = 0$ then the LS estimate converges faster than the usual rate of $\sqrt{T}$. This is summarised by the result,

$$\sqrt{T}\left(\hat{\Pi} - \Pi\right) = o_p(1).$$

2.3.2 Engle Granger Method

The Engle and Granger estimation method [4] is based upon the Vector Error Correction Model (VECM) representation and proceeds in two steps. Assuming that all variables in the cointegrating regression are $I(1)$, we first estimate the cointegration regression equation, Equation (2.7), by Ordinary Least Squares (OLS),

$$\mathbf{X}_{1,t} = \mu + \beta \mathbf{X}_{2,t} + \epsilon_t. \tag{2.7}$$

The residuals of the cointegration regression, $\hat{\epsilon}_t$, are then tested to see if they are $I(1)$. This test can be carried out via number of unit root tests, an example of which is the Dickey-Fuller (DF) test for unit roots [20]. The DF test on the residual sequence, $\hat{\epsilon}_t$, considers the autoregression of the residuals,

$$\Delta\hat{\epsilon}_t = \rho\hat{\epsilon}_{t-1} + \nu_t, \tag{2.8}$$

and examines the following hypotheses,

$$\begin{aligned} H_0: &\quad \rho = 0 \\ H_1: &\quad \rho < 1. \end{aligned}$$

If we reject the null hypothesis of the Dickey Fuller test, then the OLS estimate of β, $\hat{\beta}$, is super-consistent [21], that is, the estimator $\hat{\beta}$ has a convergence rate equal to the sample size. Other unit root tests that could be considered include the Phillips-Peron test [22] and the Augmented Dickey-Fuller test [23].

The $\hat{\epsilon}_{t-1}$ term in Equation (2.8) is then included in the VECM formulation and the remaining parameters can be estimated by OLS. Given the super-consistency of $\hat{\beta}$, the asymptotic distributions of the estimates will be identical to using the true value of β [4].

2.3.3 Dynamic Ordinary Least Squares Estimation

The Dynamic Ordinary Least Squares (DOLS) estimation procedure proposed by Stock and Watson [24], estimates the parameter β via the following regression,

$$\mathbf{X}_{1,t} = \beta' \mathbf{X}_{2,t} + d(L)\delta \mathbf{X}_{2,t} + \epsilon_t. \tag{2.9}$$

The regression equation used by Stock and Watson, Equation (2.9), can be derived by first considering the triangular representation for $\mathbf{X}_t$, shown in Equations (2.10) and (2.11), respectively,

$$\mathbf{X}_{1,t} = \beta' \mathbf{X}_{2,t} + \epsilon_{1,t}, \tag{2.10}$$

$$\Delta \mathbf{X}_{2,t} = \epsilon_{2,t}, \tag{2.11}$$

where $\epsilon_t \sim$ i.i.d. $N(0, \Sigma)$ and the covariance matrix Σ is block diagonal, the Ordinary Least Squares (OLS) estimator of β is the Maximum Likelihood (ML) estimator.

Making $\epsilon_{2,t}$ independent of $\epsilon_{1,t}$, yields,

$$\begin{aligned} \mathbb{E}\left[\epsilon_{1,t} \middle| \{\Delta \mathbf{X}_{2,t}\}\right] &= \mathbb{E}\left[\epsilon_{1,t} \middle| \{\epsilon_{2,t}\}\right] \\ &= d(L)\Delta \mathbf{X}_{2,t}, \end{aligned}$$

where,

$d(L) \quad \sum_{j=-q}^{+q} d_j L^j.$

Thus, Equation (2.10), can be written as,

$$\mathbf{X}_{1,t} = \beta' \mathbf{X}_{2,t} + d(L)\Delta \mathbf{X}_{2,t} + \nu_t^2, \tag{2.12}$$

where,

$\nu_t^2 \quad = \epsilon_{1,t} - \mathbb{E}\left[\epsilon_{1,t} \middle| \{\epsilon_{2,t}\}\right].$

By assuming that ϵ_t is Gaussian and the data is generated by the Triangular representation, Equations (2.10) and (2.11), respectively, can be reformulated into its two sided analogue, shown in Equation (2.9).

The Maximum Likelihood estimator of β can then be calculated by estimating β in Equation (2.9) by Generalised Least Squares (GLS) methods. Since $\mathbf{X}_{1,t}$ in Equation (2.11) is $I(1)$, an asymptotically equivalent estimator of β can be obtained by estimating β by Ordinary Least Squares (OLS). This estimator is known as the Dynamic OLS (DOLS) estimator.

2.3.4 Johansen Maximum Likelihood Method

Maximum Likelihood (ML) estimation for Vector Autoregressive (VAR) models was first developed by Søren Johansen in 1991 [25]. The ML method is often known as the Johansen method. A novel implementation of the Johansen ML method in Object-oriented MATLAB is given in Appendix (B).

Consider an Vector Error Correction Model of order p,

$$\Delta \mathbf{X}_t = \alpha\beta' \mathbf{X}_{t-1} + \sum_{i=1}^{p-1} \Gamma_i \Delta \mathbf{X}_{t-i} + \phi D_t + \epsilon_t, \qquad \epsilon_t \sim MVN(0, \Sigma), \tag{2.13}$$

where, the parameters of the VECM are as defined in Section (2.2.3).

The aim of the Johansen method is to determine a linear combination $Z_t = \beta' \mathbf{X}_t$ which is $I(0)$. Equation (2.13) can also be represented by,

$$Z_{0t} = \alpha\beta' Z_{1t} + \Psi Z_{2t} + \epsilon_t, \tag{2.14}$$

$$\epsilon_t = Z_{0t} - \alpha\beta' Z_{1t} + \Psi Z_{2t}, \tag{2.15}$$

where,

Z_{0t} $= \Delta \mathbf{X}_t$,
Z_{1t} $= \mathbf{X}_{t-1}$,
Ψ $= (\Gamma_1, \cdots, \Gamma_{k-1}, \Phi)$,
Z_{2t} $= (\Delta \mathbf{X}_t, \cdots, \Delta \mathbf{X}_{t-p+1}, D_t)'$,
ϵ_t is independent identically distributed multivariate, correlated, Gaussian noise with covariance Ω.

By assuming that ϵ_t are independently sampled from an identical multivariate Gaussian distribution with covariance Ω and zero mean, we can solve for the model parameters by maximising the log likelihood function for a sample size T of Equation (2.14) as follows,

$$
\begin{aligned}
logL(\alpha,\beta,\Psi,\Omega) &= \frac{1}{\sqrt{2\pi\Omega}}\exp\left(-\frac{1}{2}\sum_{t=1}^{T}(\epsilon_t)'\Omega^{-1}(\epsilon_t)\right) \\
&= constants - \frac{T}{2}log(|\Omega|) - \frac{1}{2}\sum_{t=1}^{T}(\epsilon_t)'\Omega^{-1}(\epsilon_t) \\
&= constants - \frac{T}{2}log(|\Omega|) \\
&- \frac{1}{2}\sum_{t=1}^{T}\left(Z_{0t} - \alpha\beta' Z_{1t} + \Psi Z_{2t}\right)'\Omega^{-1}\left(Z_{0t} - \alpha\beta' Z_{1t} + \Psi Z_{2t}\right). \qquad (2.16)
\end{aligned}
$$

We proceed by minimising the log-likelihood with respect to the Ψ matrix, keeping all other parameters fixed. The matrix derivative of Equation (2.16), ignoring the irrelevant constant term, is defined by,

$$
\begin{aligned}
\frac{\partial logL(\alpha,\beta,\Psi,\Omega)}{\partial\Psi_{kl}} &= \frac{\partial}{\partial\Psi_{kl}}\left\{-\frac{T}{2}log(|\Omega|) - \frac{1}{2}\sum_{t=1}^{T}(Z_{0t,\alpha} - (\alpha\beta' Z_{1t})_\alpha + \Psi_{\alpha\gamma}Z_{2t,\gamma})\right. \\
&\quad \left.\cdot \quad \Omega^{-1}_{\alpha\beta}(Z_{0t,\beta} - (\alpha\beta' Z_{1t})_\beta + \Psi_{\beta\delta}Z_{2t,\delta})\right\} \\
&= \frac{1}{2}\sum_{t=1}^{T}\left\{(Z_{0t,\alpha} - (\alpha\beta' Z_{1t})_\alpha + \Psi_{\alpha\gamma}Z_{2t,\gamma})\,\Omega^{-1}_{\alpha k}Z_{2t,\gamma} + \right. \\
&\quad \left. Z_{2t,l}\Omega^{-1}_{k\beta}\left(Z_{0t,\beta} - (\alpha\beta' Z_{1t})_\beta + \Psi_{\beta\delta}Z_{2t,\delta}\right)\right\}. \qquad (2.17)
\end{aligned}
$$

Upon assuming that the log-likelihood function is convex, the minima of Equation (2.17) is given by,

$$
\sum_{t=1}^{T}(Z_{0t} - \alpha\beta' Z_{1t} + \Psi Z_{2t})Z'_{2t} = 0. \qquad (2.18)
$$

Solving for Ψ, we substitute the results back into Equation (2.16) and proceed to minimise the log-likelihood with respect to α with β and Ω held fixed. By defining the product moment matrices,

$$
M_{ij} = \frac{1}{T}\sum_{t=1}^{T}Z_{it}Z'_{jt} \quad \text{i, j = 0, 1, 2,}
$$

the solution of Equation (2.18) with respect to Ψ is given by,

$$
\hat{\Psi} = M_{02}M_{12}^{-1} - \alpha\beta' M_{12}M_{22}^{-1}. \qquad (2.19)
$$

Next we remove the short run transitory effects, $\hat{\Psi} Z_{2t}$, in Equation (2.14). By using the Frisch-Waugh-Lovell theorem [26], we define the following auxiliary regressions,

$$
\begin{aligned}
Z_{0t} &= M_{02} M_{22}^{-1} Z_{2t} + R_{0t}, \\
Z_{1t} &= M_{12} M_{22}^{-1} Z_{2t} + R_{1t}.
\end{aligned}
$$

The residuals of the auxiliary regressions can then be used to define a "concentrated" model,

$$
\begin{aligned}
R_{0t} &= \alpha \beta' R_{1t} + \epsilon_t, \qquad & \epsilon_t \sim MVN(0, \Sigma), \\
\epsilon_t &= R_{0t} - \alpha \beta' R_{1t}. & (2.20)
\end{aligned}
$$

The concentrated model is important for understanding both the statistical and economic properties of the VECM in Equation (2.13). The VECM model contains both short-run adjustment and intervention effects, whereas the concentrated model contains long-run adjustments [27].

The log-likelihood of the concentrated model, Equation (2.20), can be expressed as,

$$
logL(\alpha, \beta, \Omega) = constants - \frac{T}{2} log(|\Omega|) - \frac{1}{2} \sum_{t=1}^{T} (R_{0t} - \alpha \beta' R_{1t})' \Omega^{-1} (R_{0t} - \alpha \beta' R_{1t}). \tag{2.21}
$$

We proceed by minimising Equation (2.21), with respect to α with Ω and β held fixed,

$$
\begin{aligned}
\frac{\partial logL(\alpha, \beta, \Omega)}{\partial \alpha_{kl}} &= \frac{\partial}{\partial \alpha_{kl}} \left\{ -\frac{T}{2} log(|\Omega|) - \frac{1}{2} \sum_{t=1}^{T} (R_{0t,\alpha} - \alpha_{\alpha\gamma} (\beta' R_{1t})_\gamma) \Omega^{-1}_{\alpha\beta} \right. \\
&\quad \cdot \quad \left. (R_{0t,\beta} - \alpha_{\beta\delta} (\beta' R_{1t})_\delta) \right\} \\
&= \frac{1}{2} \sum_{t=1}^{T} \Big\{ (R_{0t,\alpha} - \alpha_{\alpha\gamma} (\beta' R_{1t})_\gamma) \Omega^{-1}_{\alpha k} (\beta' R_{1t})_l + (\beta' R_{1t})_l \Omega^{-1}_{k\beta} \\
&\quad \cdot \quad (R_{0t,\beta} - \alpha_{\beta\delta} (\beta' R_{1t})_\delta \Big\}. \qquad (2.22)
\end{aligned}
$$

Upon assuming that the log-likelihood function is convex, the minima of Equation (2.22) corresponds to the solution of,

$$
\sum_{t=1}^{T} (R_{0t} - \alpha(\beta' R_{1t}))(\beta' R_{1t})' = 0. \tag{2.23}
$$

By defining the residual sum of squares as matrices,

$$S_{ij} = \frac{1}{T}\sum_{t=1}^{T} R_{it}R_{jt}^{'} \quad \text{i, j} = 0,\, 1$$

the solution of Equation (2.23) with respect to α is given by,

$$\hat{\alpha} = S_{01}\beta(\beta' S_{11}\beta)^{-1}. \tag{2.24}$$

Substituting Equation (2.24) into Equation (2.21), we minimise the result with respect to Ω with β held fixed. For convenience we will use the notation $M(\beta) = S_{01}\beta(\beta' S_{11}\beta)^{-1}\beta'$. Solving for Ω,

$$\begin{aligned}
\frac{\partial logL(\beta,\Omega)}{\partial \Omega_{kl}} &= -\frac{T}{2}\Omega_{kl}^{-1} + \frac{1}{2}\sum_{t=1}^{T}(R_{0t,\alpha} - M(\beta)_{\alpha\gamma}R_{1t,\gamma})\Omega_{\alpha k}^{-1}\Omega_{l\beta}^{-1}(R_{0t,\beta} - M(\beta)_{\beta\delta}R_{1t,\delta}) \\
&= -\frac{T}{2}\Omega_{kl}^{-1} + \sum_{t=1}^{T}\Big\{\Omega_{\alpha k}^{-1}R_{0t,\alpha}R_{0t,\beta}\Omega_{l\beta}^{-1} - \Omega_{\alpha k}^{-1}R_{0t,\alpha}M(\beta)_{\beta\delta}R_{1t,\delta}\Omega_{l\beta}^{-1} \\
&- \quad \Omega_{\alpha k}^{-1}M(\beta)_{\alpha\gamma}R_{1t,\gamma}R_{0t,\beta}\Omega_{l\beta}^{-1} \\
&+ \quad \Omega_{\alpha k}^{-1}M(\beta)_{\alpha\gamma}R_{1t,\gamma}R_{1t,\gamma}M(\beta)_{\beta\delta}R_{1t,\delta}\Omega_{l\beta}^{-1}\Big\}.
\end{aligned}$$

In matrix form this reduces to,

$$\begin{aligned}
\frac{\partial logL(\beta,\Omega)}{\partial \Omega} &= -\frac{T}{2}\Omega^{-1} + \frac{1}{2}\sum_{t=1}^{T}(\Omega^{-1})'\{R_{0t}R_{0t}' - R_{0t}R_{1t}'M(\beta)' - M(\beta)R_{1t}R_{0t}' \\
&\quad + \quad M(\beta)R_{1t}R_{1t}'M(\beta)'\}(\Omega^{-1})' \\
&= -\frac{T}{2}\Omega^{-1} + -\frac{T}{2}(\Omega^{-1})' + \{S_{00} - S_{01}M(\beta)' - M(\beta)S_{10} \\
&\quad + \quad M(\beta)S_{11}M(\beta)'\}(\Omega^{-1})'.
\end{aligned} \tag{2.25}$$

Multiplying both sides of Equation (2.25) by Ω yields,

$$\Omega = S_{00} - S_{01}M(\beta)' - M(\beta)S_{10} + M(\beta)S_{11}M(\beta)'. \tag{2.26}$$

By noting that,

$$\begin{aligned} S_{01}M(\beta)' &= S_{01}S_{01}\beta(\beta' S_{11}\beta)^{-1}\beta'' \\ &= S_{01}\beta(\beta' S_{11}'\beta)^{-1'}\beta' S_{01}' \\ &= S_{01}\beta(\beta' S_{11}'\beta)^{-1}\beta' S_{01} \\ &= M(\beta)S_{10}, \end{aligned}$$

and

$$\begin{aligned} M(\beta)S_{11}M(\beta)' &= S_{01}\beta(\beta' S_{11}'\beta)^{-1}\beta' S_{11}\beta(\beta' S_{11}'\beta)^{-1}\ \beta' S_{10} \\ &= S_{01}\beta(\beta' S_{11}'\beta)^{-1}\beta' S_{01}, \end{aligned}$$

Equation (2.26), can be written as,

$$\hat{\Omega} = S_{00} - S_{01}\beta(\beta' S_{11}'\beta)^{-1}\beta' S_{01}. \tag{2.27}$$

For a multivariate normal distribution, up to an overall constant, the maximum possible value of the likelihood function is $|\Omega|^{-\frac{T}{2}}$, which is also denoted by,

$$\begin{aligned} L_{max}^{-\frac{2}{T}}(\beta) &= |\Omega(\beta)| \\ &= |S_{00} - S_{01}\beta(\beta' S_{11}'\beta)^{-1}\beta' S_{01}|. \end{aligned} \tag{2.28}$$

Therefore, the problem of maximising the likelihood function reduces to finding the matrix β which maximises the determinant of Equation (2.28). To find the determinant we make use of the expression,

$$M = \begin{pmatrix} S_{00} & S_{01}\beta \\ \beta' S_{01} & \beta' S_{01}\beta \end{pmatrix},$$

and note that the determinant of M is,

$$\begin{aligned} |M| &= |S_{00}||\beta' S_{01}\beta - \beta' S_{10}S_{00}^{-1}S_{01}\beta| \\ &= |\beta' S_{01}\beta||S_{00} - S_{01}\beta(\beta' S_{11}\beta)^{-1}\beta' S_{10}|. \end{aligned} \tag{2.29}$$

We see that the determinant of Ω is contained in Equation (2.29). Solving for $|\Omega|$ yields,

$$\begin{aligned}|\Omega| &= |S_{00}|\frac{|\beta^{'}S_{11}\beta - \beta^{'}S_{10}S_{00}^{-1}S_{01}\beta|}{|\beta^{'}S_{11}\beta|} \\ &= |S_{00}|\frac{|\beta^{'}(S_{11} - S_{10}S_{00}^{-1}S_{01})\beta|}{|\beta^{'}S_{11}\beta|}.\end{aligned}$$

The values of β which maximise the likelihood function are the solutions to,

$$(S_{11} - S_{10}S_{00}^{-1}S_{01})\vec{v}^{i} = p^{i}S_{11}\vec{v}^{i},$$

or equivalently for $\lambda^{i} = 1 - p^{i}$,

$$S_{10}S_{00}^{-1}S_{01}\vec{v}^{i} = \lambda^{i}S_{11}\vec{v}^{i}. \tag{2.30}$$

The vectors $\vec{v}^{i}$ represent cointegration relations. By choosing the normalisation $\vec{v}^{i}S_{11}\vec{v}^{j} = \mathbb{I}_{n_x}$ if $i = j$ and 0 otherwise, then $\beta' S_{10}S_{00}^{-1}S_{01}\beta = \text{diag}(\lambda_1, \lambda_2, \ldots)$. With this choice of β, the maximum likelihood function is then,

$$L_{max}^{-\frac{2}{T}} = \left(|S_{00}|\prod_{i=1}^{r}(1 - \hat{\lambda}_i)\right). \tag{2.31}$$

In summary, the Johansen Maximum Likelihood method produces estimates of the parameters of a Vector Error Correction Model (VECM) of the form given in Equation (2.13). The parameter estimates and the resulting likelihood function are summarised in Table (2.1), noting that here r represents the cointegration rank.

Table 2.1: Summary of Estimates from the Johansen Maximum Likelihood Method

Estimator	Form
$\hat{\alpha}$	$S_{01}\beta(\beta' S_{11}\beta)^{-1}$
$\hat{\beta}$	$\left[\vec{v}^{1}, \ldots, \vec{v}^{r}\right]' S_{11}^{-1/2}$
$\hat{\Omega}$	$S_{00} - S_{01}\beta(\beta' S_{11}'\beta)^{-1}\beta' S_{01}$
$\hat{\Psi}$	$M_{02}M_{12}^{-1} - \alpha\beta' M_{12}M_{22}^{-1}$
Maximum Likelihood	$\left(\lvert S_{00}\rvert \prod_{i=1}^{r}(1 - \hat{\lambda}_i)\right)$

Asymptotic Properties

The Maximum Likelihood (ML) estimators of the parameters of the Vector Error Correction Model in Equation (2.13) are consistent and have the following central limit theorem [17],

$$\sqrt{T}\text{vec}\left([\hat{\Pi}:\hat{\Gamma}]-[\Pi:\Gamma]\right)\xrightarrow{d}N(0,\Sigma_{co}),$$

where,

$$\Sigma_{co} = \left(\begin{bmatrix}\beta & 0\\ 0 & I_{n_x p-n_x}\end{bmatrix}\Omega^{-1}\begin{bmatrix}\beta' & 0\\ 0 & I_{n_x p-n_x}\end{bmatrix}\right)$$

and

$$\sqrt{T}\text{vech}\left(\tilde{\Sigma}_u-\Sigma_u\right)\xrightarrow{d}N(0,2D_{n_x}^{+}\left(\Sigma_u\otimes\Sigma_u\right)D_{n_x}^{+'})$$

where,

D_{n_x} is the duplication matrix

$D_{n_x}^{+}$ $= (D_{n_x}^{'}D_{n_x})^{-1}D_{n_x}^{'}$, the Moore-Pemrose generalised inverse of D_{n_x}

Furthermore, $\tilde{\Sigma}$ is asymptotically independent of $\hat{\Pi}$ and $\hat{\Gamma}$.

The ML estimator of $[\Pi:\Gamma]$ has the same asymptotic distribution as the LS estimator presented in Section (2.3.1).

2.3.5 Canonical Correlation Regression Estimator

The use of canonical correlation analysis (CCA) was first proposed by Bossaerts in 1988 [8]. A more rigorous methodology was introduced by Bewley and Yang in 1995, allowing for deterministic trends and other variables explaining short-run dynamics, termed Level Canonical Correlation Analysis (LCCA) [28]. In the LCCA methodology the canonical correlations are computed in levels, that is, the first differences of $\mathbf{X}_t$.

The Level Canonical Correlation Analysis (LCCA) estimation method is computationally similar to the Johansen method, covered in Section (2.3.4). In the LCCA method, exogenous variables are removed by regressing $\mathbf{X}_t$ and $\mathbf{X}_{t-1}$ onto those variables. The residuals from the regression, R_{0t} and R_{1t}, are then used in the following regression,

$$R_{0t}=BR_{1t}+\epsilon_t. \tag{2.32}$$

The canonical correlations in Equation (2.32) are determined by solving the following eigenvalue problem,

$$\left|S_{10}S_{00}^{-1}S_{01}-\lambda S_{11}\right|=0, \tag{2.33}$$

where,

$$S_{ij} = \frac{1}{T}\sum_{t=1}^{T} R_{it}R_{jt}^{'} \quad \text{i, j = 1, 2.}$$

.

The eigenvalue problem solved in the LCCA method, Equation (2.33), is the same as that solved in the Johansen method, Equation (2.30). However, different interpretations are given to the results, since in the LCCA method we are seeking canonical correlations between variables in levels while in the Johansen methods we correlate both levels and differences [8]. The LCCA method incorporates four types of cointegration rank tests, two Dickey-Fuller type tests, a trace test, and a maximum eigenvalue test.

2.3.6 Cointegration Rank

The Johansen maximum likelihood estimation method and its extensions critically depend on correctly estimating the number of cointegrating relationships, r. Two Likelihood Ratio (LR) tests for the determination of cointegration rank r have been suggested in relationship with the Johansen method: the trace test and the maximum eigenvalue test. It is important to note that the distribution of the LR test statistics for both the trace and maximum eigenvalue tests are non-standard and as such have been tabulated in [29] and [30].

Trace Test

The trace test is immediately suggested by the Johansen method and examines the following hypotheses,

$$\begin{aligned} H_0: &\quad \text{rank}(\Pi) \leq r && (\text{or} \quad \Pi = \alpha\beta') \\ H_1: &\quad \text{rank}(\Pi) = n_x && (\text{or} \quad \Pi \text{ is full rank}) \end{aligned}$$

The LR test and resulting test statistic is given by,

$$Q\left(H_0/H_1\right) = \frac{|S_{00}| \prod_{i=1}^{r}\left(1-\hat{\lambda}_i\right)}{|S_{00}| \prod_{i=1}^{n_x}\left(1-\hat{\lambda}_i\right)}$$

$$logQ\left(H_0/H_1\right) = -\sum_{i=r+1}^{n_x} log\left(1-\hat{\lambda}_i\right).$$

The trace statistic has a limit distribution which can be expressed in terms of a $(n_x - r)$-dimensional Brownian motion B with i.i.d. components [25] as,

$$tr\left\{\int (dB)F^{'}\left[\int FF^{'}du\right]^{-1}\int F(dB)^{'}\right\} \tag{2.34}$$

where,

$F^{'}$ $= (F_1^{'}, F_2^{'})$,
$F_{1i}(t)$ $= B_1(t) - \int B_i(u)du$,
$F_2(t)$ $= t - \frac{1}{2}$.

The trace test does not provide the exact number of unit roots. Therefore, to estimate the value of r a sequence of trace tests must be performed.

Maximum Eigenvalue Test

The Maximum Eigenvalue test examines the following hypotheses,

$$\begin{aligned} H_0: &\quad \text{rank}(\Pi) = r \\ H_1: &\quad \text{rank}(\Pi) = r+1 \end{aligned}$$

The LR test and resulting test statistic is given by,

$$\begin{aligned} Q\left(H_0/H_1\right) &= \frac{|S_{00}|\prod_{i=1}^{r}\left(1-\hat{\lambda}_i\right)}{|S_{00}|\prod_{i=1}^{r+1}\left(1-\hat{\lambda}_i\right)} \\ logQ\left(H_0/H_1\right) &= -log\left(1-\hat{\lambda}_{r+1}\right). \end{aligned} \tag{2.35}$$

The LR test statistic, Equation (2.35), is asymptotically distributed as the maximum eigenvalue of the matrix given in Equation (2.34).

2.4 Case Study: Johansen Maximum Likelihood Estimation

The Johansen Maximum Likelihood (ML) estimation method provides estimates of the parameters Vector Autoregressive Model (VECM) of order p. In this study we evaluate the efficacy of

the Johansen ML method at estimating the parameters of a cointegrated VECM with a deterministic term. The case study utilises the novel implementation of the Johansen ML method in Object-oriented MATLAB is given in Appendix (B).

2.4.1 Model

Consider a data generating system (DGS) defined by the following two equations,

$$M_t = AM_{t-1} + \eta_t, \qquad \eta_t \sim MVN(0, \Sigma) \tag{2.36}$$

$$Y_t = M_t + \alpha\beta' X_{t-1} + \Gamma(X_t - X_{t-1}) + \epsilon_t, \qquad \epsilon_t \sim MVN(0, \Omega) \tag{2.37}$$

where,

$M_t \in \mathbb{R}^{n_x}$	is the state vector,
$Y_t \in \mathbb{R}^{n_x}$	is the measurement vector,
$A : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the system matrix,
$\eta_t \in \mathbb{R}^{n_x}$	is the white, matrix variate Gaussian, zero mean process noise vector with covariance Σ,
$\alpha : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the loading factor matrix,
$\beta : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the cointegration vector matrix,
$\Gamma : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the lag matrix,
$\epsilon_t \in \mathbb{R}^{n_x}$	is the white, matrix variate Gaussian, zero mean process noise vector with covariance Ω.

Equation (2.36) is known as the *system* equation and Equation (2.37) is known as the *measurement* equation. We note that the system equation is Vector Autoregressive (VAR) process of order 1. The measurement equation is in the form of a Vector Error Correction Model (VECM) of order 2 with a deterministic term M_t, which is driven by a latent process, given by the system equation.

A total of 50 data sets were generated from the DGS, with each data set containing 10,000 observations. The set of parameters used to generate the data from the DGS are shown in Table (2.2) and constitute the "true" set of parameters, against which the estimates of the Johansen ML method will be compared.

Using the current settings for the α and β matrices, we can see that the measurement equation of the DGS, Equation (2.37), is actually a cointegrated VECM; since, $\text{rank}(\alpha\beta') = 1$ which is less than n_x.

A typical realisation of the measurement equation is shown in Figure (2.1).

Table 2.2: Case Study Data Generating System Parameter Values

Parameter	Value
n_x	2
A	$\begin{pmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{pmatrix}$
Σ	$\begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}$
α	$\begin{pmatrix} 0.2 & 0.2 \\ 0.2 & 0.2 \end{pmatrix}$
β	$\begin{pmatrix} 1 & 0 \\ -1 & -1 \end{pmatrix}$
Γ	$\begin{pmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{pmatrix}$
Ω	$\begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}$
r	1

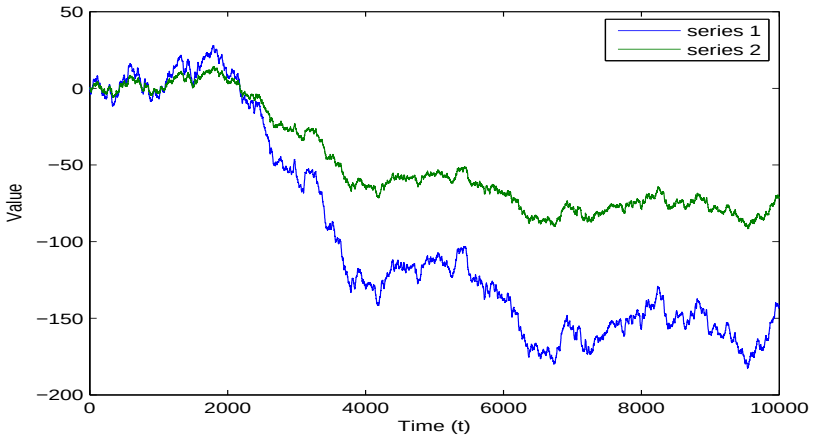


Figure 2.1: A typical realisation of the measurement equation of the Case Study Model. Note that there are two series, which corresponds with the choice of n_x, in the DGS, shown in Table (2.2).

2.4.2 Method

The observations generated from the measurement equation, Equation (2.37), Data Generating System (DGS) were presented to the Johansen Maximum Likelihood (ML) method in two ways: batch and sliding window. The order of the Vector Error Correction Model (VECM) estimated by the Johansen ML method was set to 2.

In the batch procedure, the data sets were partitioned into subsets, each with an incrementally increasing number of observations, where the size of the increments was a predefined constant, Batch_size. At each time step, t, a subset of data $\mathbb{Y}_{1:t} = \{y_1, \ldots, y_{t+\text{Batch\_size}}\}$ was used within

the Johansen ML method. By presenting the data in batch subsets, it is possible to evaluate the effect of increasing the number of observations on the performance of the Johansen ML method.

In the sliding window procedure, each of the data sets were partitioned into subsets, each with a constant number of observations. At each time step, t, a subset of data $\mathbb{Y}_{t:(t+\text{Window\_size})} = \{y_t, \ldots, y_{t+\text{Window\_size}}\}$ was used within the Johansen ML method. By presenting the data in sliding window subsets, it is possible to evaluate the dynamics of the parameter estimates throughout different sections of the observation series.

2.4.3 Results

A summary of the results of the Johansen ML method for both batch and sliding window estimation is provided in the proceeding sections.

Batch Estimation

The results of the batch estimation method are presented in Figure (2.2). The figure shows the Mean Square Error (MSE) associated with the average estimated parameter over all 50 data sets for a given batch data size.

From Figure (2.2) it is evident that the estimates produced by the Johansen ML method vary with the size of the observations. The estimates generated with smaller batch data sizes are, in most cases, largely different from those produced with larger batch subset sizes. For example, the Mean Square Error (MSE) of $\hat{\alpha}$ in batch number 1 is larger than 1×10^{-4}, whereas for batch number 20, which uses all available data, the MSE is below 1×10^{-4}. The most marked example of this phenomenon is the MSE of the trace of the estimated Lag matrix, $\hat{\Gamma}$. In general, the gradual increase of the number of observations used within the Johansen ML method shows a stabilisation in the MSE of the estimated parameters. The exception to this rule is the MSE of the estimated cointegration rank, $\hat{r}$, which continues to fluctuate irrespective of the number of observations.

The results of the batch estimation procedure shows that the Johansen ML method produces estimates of the parameters of a given Vector Error Correction Model (VECM) with increasing efficacy as the number of observations are increased. The small MSE of the estimates over all batch sizes demonstrates the ability of the Johansen ML method to correctly estimate the parameters of a VECM with a deterministic term.

Sliding Window Estimation

The results of the sliding window estimation method are presented in Figure (2.3). The figure shows the Mean Square Error (MSE) associated with the average estimated parameter over all 50 data sets for a given sliding window.

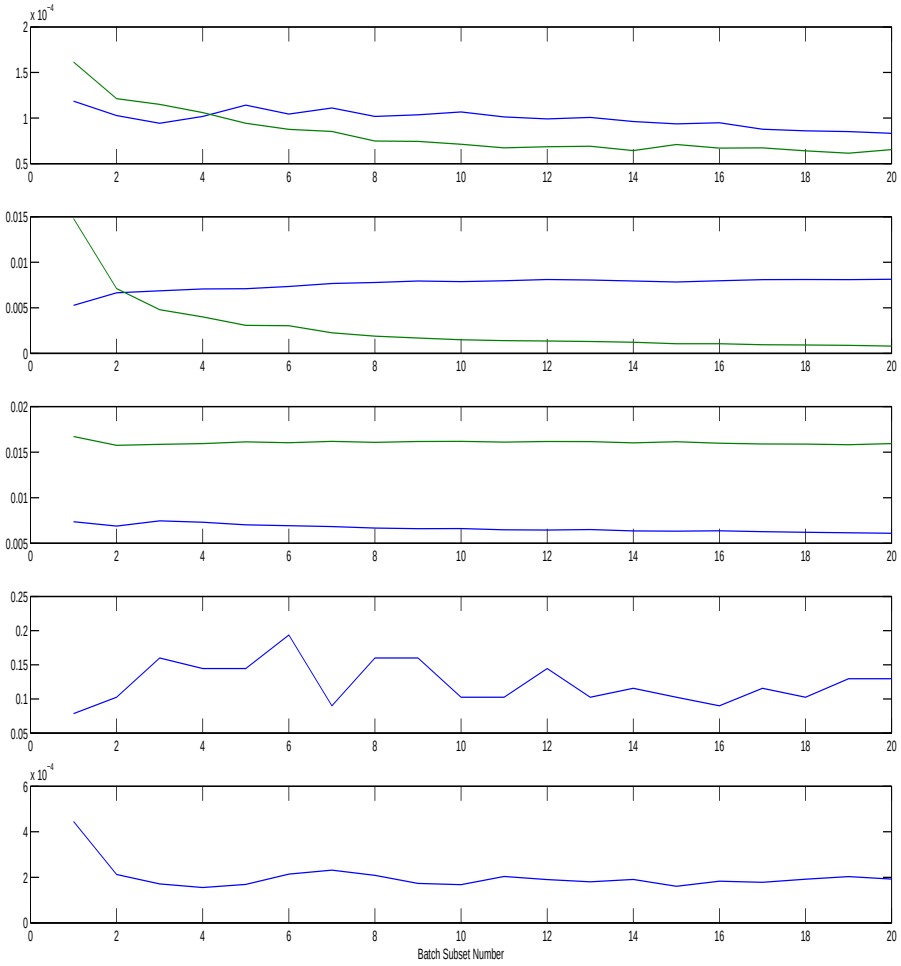


Figure 2.2: Johansen Case Study: Batch Data Results. The figures, from top to bottom, represent the Mean Square Error (MSE) of the $\hat{\alpha}$, $\hat{\beta}$, $\hat{M}_t$, $\hat{r}$ and trace($\hat{\Gamma}$), estimates for each batch subset over all data sets, respectively. In the first three Figures, the blue and green plots relate to the first and second components of the $\hat{\alpha}$, $\hat{\beta}$ and $\hat{M}_t$ vectors.

From Figure (2.3) it is evident that the estimates produced by the Johansen ML method vary over the history of the observations series. The estimates generated with the initial sliding window data are, in most cases, largely different from those produced with latter sliding window data. For example, the Mean Square Error (MSE) of $\hat{\alpha}$ in sliding window number 1 is larger than 9×10^{-5}, whereas for the final sliding window, the MSE is below 6×10^{-5}. The most marked example of this phenomenon is the MSE of the trace of the estimated Lag matrix, $\hat{\Gamma}$,

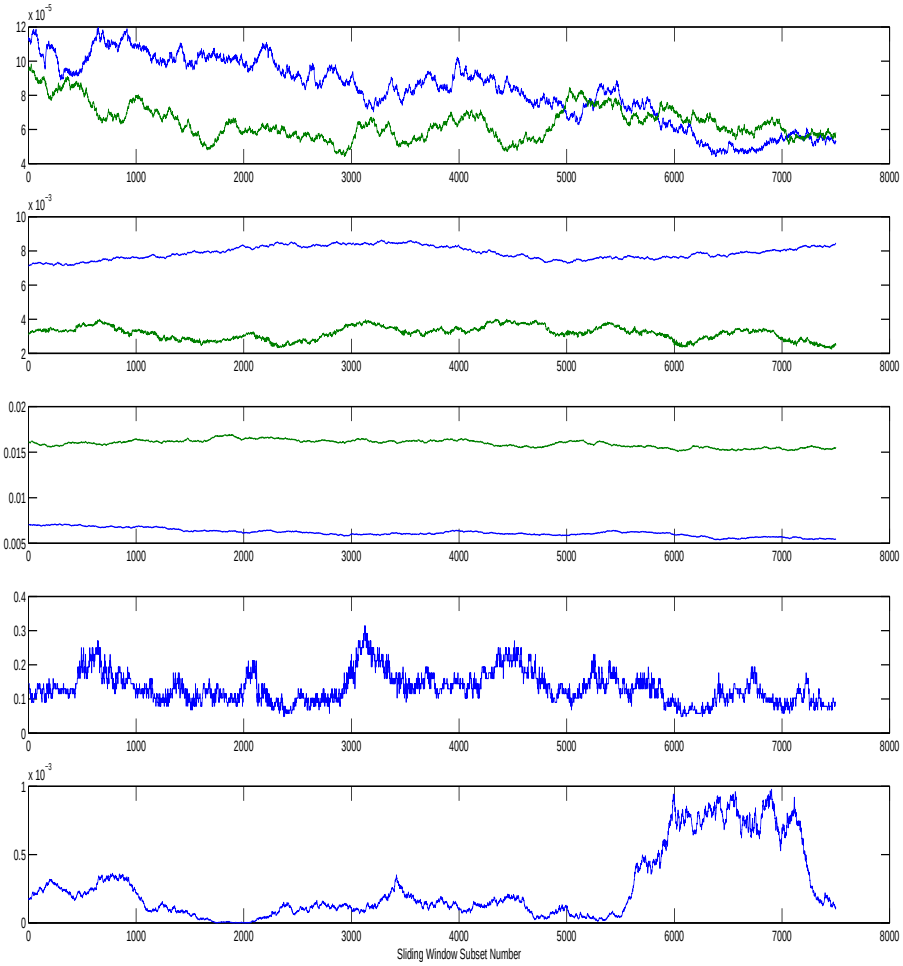


Figure 2.3: Johansen Case Study: Sliding Window Results. The figures, from top to bottom, represent the Mean Square Error (MSE) of the $\hat{\alpha}$, $\hat{\beta}$, $\hat{M}_t$, $\hat{r}$ and trace($\hat{\Gamma}$), estimates for each sliding window over all data sets, respectively. In the first three Figures, the blue and green plots relate to the first and second components of the $\hat{\alpha}$, $\hat{\beta}$ and $\hat{M}_t$ vectors.

which exhibits large fluctuations in the MSE of the estimate between sliding window numbers 5000 and 6000 and 7000 and 7500. The use of sliding window estimation technique shows that, in general, the MSE of the estimates produced by the Johansen ML method do not stabilise, that is, they are different for each sliding window. A notable exception to this is the estimate of the deterministic term, $\hat{M}_t$, which appears to have a relatively state MSE over the course of the observation history.

By comparing the Mean Square Error (MSE) of the estimates produced by the batch and sliding window estimation procedure, it is clear that they are lower in the batch estimation procedure. By using batch data, the wild fluctuations in parameter estimates within each sliding window, are mitigated. However, the sliding window estimation procedure is particularly useful when we wish to examine the stochasticity of the parameters in the VECM specification over the course of the observation series.

2.5 Case Study: Cointegration Rank Estimation

The purpose of this case study is to evaluate the efficacy of the Johansen Maximum Likelihood (ML) method at predicting the cointegration rank, r, of Vector Error Correction Models (VECM) with different rank specifications. The case study utilises the novel implementation of the Johansen ML method in Object-oriented MATLAB is given in Appendix (B).

2.5.1 Model

The Vector Error Correction Models (VECM)'s investigated were derived from a study by Sugita in 2002 [1]. In the study, Sugita specified five VECM's of order 1, each with an incrementally increasing number of cointegrating relationships, summarised by the rank, r.

The specifications of the five VECM's are shown in Table (2.3).

Table 2.3: Sugita Vector Error Correction Model (VECM) Specification, [1]. In the model specification, $\Delta Y_t = Y_t - Y_{t-1}$, $\mu = \begin{bmatrix}0.1 & 0.1 & 0.1 & 0.1\end{bmatrix}'$ and $\epsilon_t \sim N(0, \mathbb{I}_4)$

Model Number	Model Specification	Rank
1	$\Delta Y_t = \mu + \epsilon_t$	0
2	$\Delta Y_t = \mu + \begin{bmatrix}-0.2\\-0.2\\-0.2\\-0.2\end{bmatrix} \begin{bmatrix}1 & 0 & 0 & -1\end{bmatrix} Y_{t-1} + \epsilon_t$	1
3	$\Delta Y_t = \mu + \begin{bmatrix}-0.2 & -0.2\\0.2 & -0.2\\0.2 & 0.2\\-0.2 & 0.2\end{bmatrix} \begin{bmatrix}1 & 0 & 0 & -1\\0 & 1 & 0 & -1\end{bmatrix} Y_{t-1} + \epsilon_t$	2
4	$\Delta Y_t = \mu + \begin{bmatrix}-0.2 & -0.2 & -0.2\\0.2 & -0.2 & -0.2\\0.2 & 0.2 & -0.2\\0.2 & 0.2 & 0.2\end{bmatrix} \begin{bmatrix}1 & 0 & 0 & -1\\0 & 1 & 0 & -1\\0 & 0 & 1 & -1\end{bmatrix} Y_{t-1} + \epsilon_t$	3
5	$\Delta Y_t = \mu + \begin{bmatrix}-0.2 & -0.2 & -0.2 & -0.2\\0.2 & -0.2 & -0.2 & -0.2\\0.2 & 0.2 & -0.2 & -0.2\\0.2 & 0.2 & 0.2 & -0.2\end{bmatrix} \begin{bmatrix}1 & 0 & 0 & -1\\0 & 1 & 0 & -1\\0 & 0 & 1 & -1\\0 & 0 & 0 & -1\end{bmatrix} Y_{t-1} + \epsilon_t$	4

A total of 2000 data sets were generated for each of the five models; 1000 containing 50 observations and 1000 containing 100 observations.

2.5.2 Method

The efficacy of the Johansen ML method at estimating the cointegration rank, r, of a Vector Error Correction Model (VECM) was evaluated by using all five models in Table (2.3).

An estimate of the cointegrating rank, r, was calculated using the Maximum Eigenvalue test, shown in Section (2.3.6), by a p-value with a 5 percent significance level. The frequency of each estimated rank was then counted over all data sets for both 50 observations and 100 observations, to produce a rate at which the estimated rank was chosen.

2.5.3 Results

The frequency with which the Johansen ML estimation method estimated each possible cointegrating rank, r, for the five models are shown in Table (2.4) and Table (2.5). Table (2.4) shows the estimation results for the 1000 data sets with 50 observations, whereas Table (2.5) shows the results for the 1000 data sets with 100 observations.

Table 2.4: Cointegration Rank Estimates for Datasets with 50 Observations. The most frequently estimated rank is shown in bold.

Model Number (Rank)	Estimated Rank	Frequency of Estimation
1 (0)	0	**0.8870**
	1	0.0590
	2	0.0010
	3	0.0010
	4	0.0520
2 (1)	0	0.0220
	1	**0.7460**
	2	0.1150
	3	0.0210
	4	0.0960
3 (2)	0	0.0040
	1	0.0750
	2	**0.7480**
	3	0.0720
	4	0.1010
4 (3)	0	0.0020
	1	0.1960
	2	**0.4450**
	3	0.1330
	4	0.2240
5 (4)	0	0.0000
	1	0.0030
	2	0.1280
	3	0.0400
	4	**0.8290**

Table 2.5: Cointegration Rank Estimates for Datasets with 100 Observations. The most frequently estimated rank is shown in bold.

Model Number (Rank)	Estimated Rank	Frequency of Estimation
1 (0)	0	**0.9120**
	1	0.0600
	2	0.0000
	3	0.0020
	4	0.0260
2 (1)	0	0.0000
	1	**0.7710**
	2	0.0900
	3	0.0200
	4	0.1190
3 (2)	0	0.0000
	1	0.0000
	2	**0.8770**
	3	0.0610
	4	0.0620
4 (3)	0	0.0000
	1	0.0000
	2	0.2550
	3	**0.5190**
	4	0.2260
5 (4)	0	0.0000
	1	0.0000
	2	0.0000
	3	0.0010
	4	**0.9990**

The results in Tables (2.4) and (2.5), respectively, show that, in general, the most frequently estimated rank is the true rank of each model. The sole exception to this rule is Model Number 4 in Table (2.4). In this particular case, the rank was most frequently estimated as $r = 2$, when in fact the true rank was $r = 3$.

The results also show that the Johansen ML estimation method suffers from a shortage of observations. Upon comparing Table (2.5) to Table (2.4) it is evident that the estimated rank is more frequently correct with 100 observations than with 50 observations. An increase in the number of observations also improves the frequency with which estimated rank of Model Number 4 is correct; increasing the number of observations to 100 increases the frequency of estimating the true rank, $r = 3$, from 0.1330 to 0.5190. This degeneracy is especially present in the models with higher true rank.

2.6 Summary

Within this chapter we have presented the definition, specification and estimation of multivariate cointegration models within finance. The most popular estimation technique, Johansen's Maximum Likelihood method, was derived and its efficacy evaluated in two separate case studies. The Johansen Maximum Likelihood method was demonstrated to be an effective estimator of the parameters of a Vector Error Correction Model (VECM) and also of the cointegrating rank of the VECM. However, it was shown that the efficacy of the Johansen method is reduced when data is scarce. A common limitation to the estimation procedures introduced in this chapter is that they operate on batches of data and are not generally amenable to situations where estimation is required to be on-line. We now proceed to investigate techniques for the on-line estimation of a set of latent parameters given potentially noisy data.

Chapter 3

Filtering for Multivariate Cointegration Models

Filtering and estimation are two of the most pervasive tools of engineering. Whenever the state of a system must be estimated from noisy measurements, some kind of state estimator is employed to combine measurement from many sources together to produce an accurate estimate of the true system state.

This chapter provides an exposition on filtering techniques for multivariate cointegration models. We begin the chapter with an explanation of the general filtering problem for stochastic, discrete state space model and motivate its optimal Bayesian solution, in Section (3.1). The Kalman filter is then introduced as a tractable Bayesian optimal estimator, under a restricted state space model, in Section (3.2). An extension to the Kalman filter for non-linear state space models, the Extended

Kalman filter, is then presented in Section (3.3).

A succinct introduction to Monte Carlo methods, in conjunction with two fundamental Monte Carlo sampling techniques, rejection sampling and importance

sampling is introduced in Section (3.4). The Sequential Monte Carlo (SMC) paradigm, based on the importance sampling technique, is then presented in the following section, Section (3.5). Three SMC algorithms, Sequential Importance Sampling, Sequential Importance Sampling Resampling (SISR) and the Auxiliary Particle Filter are also introduced. Resampling schemes used within SMC methods are reviewed in Section (3.5.4), including Multinomial, Residual, Stratified and Systematic resampling. The properties of the different schemes are also stated.

In the final sections we conduct three case studies, the first of which examines the effectiveness of the Kalman filter at deducing the state of a latent process within a Vector Error Correction

Model (VECM), (3.6). In Section (3.7), the second study, we investigate the efficacy of the Extended Kalman filter at estimating the state of a non-linear

latent process within a VECM. The effect of each resampling scheme on the ability of the SISR filter to elucidate the state of a two non-linear latent processes within a VECM was researched in Section (3.8).

3.1 The Filtering Problem

To motivate the discussion of the filtering problem in a general context, first consider two stochastic discrete-time state space models of the form,

$$\mathbf{X}_t = f_t\left(\mathbf{X}_{t-1}, \mathbf{W}_{t-1}\right) \tag{3.1}$$

$$\mathbf{Y}_t = h_t\left(\mathbf{X}_t, \mathbf{V}_t\right), \tag{3.2}$$

where,

$\mathbf{X}_t \in \mathbb{R}^{n_x}$	is the state vector,
$\mathbf{Y}_t \in \mathbb{R}^{n_y}$	is the measurement vector,
$f_t(\cdot,\cdot) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_v} \to \mathbb{R}^{n_x}$	is a non-linear, time variant, deterministic function,
$h_t(\cdot,\cdot) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_v} \to \mathbb{R}^{n_x}$	is a non-linear, time variant, deterministic function,
$\{\mathbf{W}_{t-1}, t \in \mathbb{N}\}$	i.i.d. system noise sequences,
$\{\mathbf{V}_t, t \in \mathbb{N}\}$	i.i.d. measurement noise sequences.

Equation (3.1) is known as the *system equation* and Equation (3.2) is known as the *measurement equation*. It is of interest to note that the system equation is a Markov process of order one.

The general objective of filtering is to provide a sequence of estimates from the system equation, $\{\mathbf{x}_t, t \in \mathbb{N}\}$, based on the set of all observations from the measurement equation, $\mathbb{Y}_{1:t} = \{\mathbf{y}_i, i = 1, \cdots, t\}$ up to time t.

From the Bayesian viewpoint, the aim of the filtering problem is to calculate a posterior distribution $p\left(\mathbf{X}_t | \mathbf{Y}_{1:t}\right)$ which encompasses the degree of belief in the state $\mathbf{x}_t$ taking different values given the observed data $\mathbf{y}_{1:t}$. Given then the initial probability distribution function of the state vector, $p(\mathbf{X}_t)$, also known as the prior, the posterior distribution may be obtained recursively in two stages: prediction and update.

Prediction Stage

The prediction stage calculates an *a priori* estimate of the posterior distribution $p\left(\mathbf{X}_t|\mathbf{Y}_{1:t}\right)$ at time t. The *a priori* estimate is calculated by using the *a posteriori* estimate at the previous time step and the identity, $p\left(\mathbf{X}_t|\mathbf{X}_{t-1}, \mathbf{Y}_{1:t-1}\right) = p\left(\mathbf{X}_t|\mathbf{X}_{t-1}\right)$, in the Chapman-Kolmogorov equation, as follows,

$$\begin{aligned} p(\mathbf{X}_t|\mathbf{Y}_{1:t-1}) &= \int p(\mathbf{X}_t, \mathbf{X}_{t-1}|\mathbf{Y}_{1:t-1}) d\mathbf{X}_{t-1} \\ &= \int p(\mathbf{X}_t|\mathbf{X}_{t-1}, \mathbf{Y}_{1:t-1}) p(\mathbf{X}_{t-1}|\mathbf{Y}_{1:t-1}) d\mathbf{X}_{t-1} \\ &= \int p(\mathbf{X}_t|\mathbf{X}_{t-1}) p(\mathbf{X}_{t-1}|\mathbf{Y}_{1:t-1}) d\ \mathbf{X}_{t-1}. \end{aligned} \tag{3.3}$$

Update Stage

The update stage produces an *a posteriori* estimate of the posterior distribution at time t, given a measurement, $\mathbf{y}_t$. The *a posteriori* estimate is calculated through the following equations,

$$\begin{aligned} p(\mathbf{X}_t|\mathbf{Y}_{1:t}) &= \frac{p(\mathbf{Y}_{1:t}|\mathbf{X}_t) p(\mathbf{X}_t)}{p(\mathbf{Y}_{1:t})} \\ &= \frac{p(\mathbf{Y}_t, \mathbf{Y}_{1:t-1}|\mathbf{X}_t) p(\mathbf{X}_t)}{p(\mathbf{Y}_t, \mathbf{Y}_{1:t-1})} \\ &= \frac{p(\mathbf{Y}_t|\mathbf{Y}_{1:t-1}, \mathbf{X}_t) p(\mathbf{Y}_{1:t-1}|\mathbf{X}_t) p(\mathbf{X}_t)}{p(\mathbf{Y}_t|\mathbf{Y}_{1:t-1}) p(\mathbf{Y}_{1:t-1})} \\ &= \frac{p(\mathbf{Y}_t|\mathbf{Y}_{1:t-1}, \mathbf{X}_t) p(\mathbf{X}_t|\mathbf{Y}_{1:t-1}) p(\mathbf{Y}_{1:t-1}) p(\mathbf{X}\ _t)}{p(\mathbf{Y}_t|\mathbf{Y}_{1:t-1}) p(\mathbf{Y}_{1:t-1}) p(\mathbf{X}_t)} \\ &= \frac{p(\mathbf{Y}_t|\mathbf{X}_t) p(\mathbf{X}_t|\mathbf{Y}_{1:t-1})}{p(\mathbf{Y}_t|\mathbf{Y}_{1:t-1})}, \end{aligned} \tag{3.4}$$

where the normalizing constant in Equation (3.4) is given by,

$$p(\mathbf{Y}_t|\mathbf{Y}_{1:t-1}) = \int p\left(\mathbf{Y}_t|\mathbf{X}_t\right) p\left(\mathbf{X}_t|\mathbf{Y}_{1:t-1}\right).$$

In the update stage, the measurement $\mathbf{y}_t$ is used to modify the prior density to obtain the required posterior density of the current state.

The recursive structure of the optimal Bayesian solution to the filtering problem is clearly evident from Equations (3.3) and (3.4), respectively. However, the optimal solution is conceptual, due to the fact that, in general, the integrals in the prediction and update stages are high dimensional and cannot be determined analytically.

By restricting the nature of the system and measurement equations, an optimal solution may be tractable, an example of an algorithm that utilises this approach is the Kalman filter [31]. In situations where imposing restrictions is undesirable, algorithms that approximate the optimal Bayesian solution can be used, two examples of which are Extended Kalman filter and Sequential Monte Carlo methods [32].

3.2 The Kalman Filter

The Kalman filter was introduced in 1960 by R.E. Kalman and is a recursive Bayesian optimal estimator for linear systems with Gaussian noise characteristics [31]. Since its introduction, the Kalman filter has been widely used for state, parameter and system identification for both linear and non-linear systems.

The earliest precursor to the Kalman filter was the invention and usage of the Least Squares estimation technique by Gauss in the study of planetary orbits [33]. The application of the Least Squares estimation technique to the estimation of stochastic processes was pioneered by Kolmogorov in 1941 and Weiner in 1942 and resulted in the development of the Minimum Mean Square Error (MMSE) technique [34]. The use of recursive estimation in systems where there is dynamic evolution of the quantity being estimated was established by Follins in 1956 [34]. Kalman combined and extended the aforementioned techniques to derive Bayesian optimal estimator for linear systems with Gaussian noise characteristics.

The Kalman filter consists of a set of recursive equations used to propagate states of a system with a given approximate dynamic model and a set of noisy measurements. In the presence of Gaussian posterior densities, the Kalman filter is considered to be an optimal filter that can be completely characterised by the mean and covariance at each time step t, see [35].

3.2.1 State Space Representation

The Kalman filter applies to state space models as defined below,

$$\mathbf{X}_t = F_t\mathbf{X}_{t-1} + \mathbf{W}_{t-1}, \qquad \mathbf{W}_{t-1} \sim N(0, Q_t) \tag{3.5}$$

$$\mathbf{Y}_t = H_t\mathbf{X}_t + \mathbf{V}_t, \qquad \mathbf{V}_t \sim N(0, R_t) \tag{3.6}$$

where,

$\mathbf{X}_t \in \mathbb{R}^{n_x}$	is the state vector,
$\mathbf{Y}_t \in \mathbb{R}^{n_y}$	is the measurement vector,
F_t: $\mathbb{R}^{n_v} \rightarrow \mathbb{R}^{n_x}$	is the system matrix,
$\mathbf{W}_t \in \mathbb{R}^{n_x}$	is the white, Gaussian, zero mean noise vector with covariance Q_t,
H_t: $\mathbb{R}^{n_v} \rightarrow \mathbb{R}^{n_x}$	is the measurement matrix,
$\mathbf{V}_t \in \mathbb{R}^{n_x}$	is the white, Gaussian, zero mean measurement noise vector with covariance R_t.

Equation (3.5) is known as the *system equation* and Equation (3.6) is known as the *measurement equation*. The system and measurement matrices, F_t and H_t, respectively, as well as the noise parameters $\mathbf{W}_t$ and $\mathbf{V}_t$, respectively, are time variant.

3.2.2 Derivation of the Kalman Filter Predict-Update Recursions

An overview of the derivation of the prediction and update stages of the Kalman filter are shown in the proceeding sections. Here, we assume that $\mathbf{X}_0$, $\mathbf{W}_t$ and $\mathbf{V}_t$ are Gaussian for all $t \geq 0$ and $\mathbb{Y}_{1:t} = \{y_i, i = 1, \cdots, t\}$ is the set of all measurements from the measurement equation up to time t.

Prediction Equations

To begin, we apply the projection operator to the system equation, Equation (3.5), giving an *a priori* prediction of the state of the system at time $t+1$, given measurements $\mathbb{Y}_t$.

$$\begin{aligned} \tilde{\mathbf{X}}_{t+1|t} &\stackrel{\triangle}{=} \mathbb{E}(F_t \mathbf{X}_{t|t} + \mathbf{W}_t | \mathbb{Y}_t) \\ &= F_t \hat{\mathbf{X}}_{t|t}. \end{aligned} \tag{3.7}$$

The *a priori* error covariance matrix associated with $\tilde{\mathbf{X}}_{t+1|t}$ is given by,

$$\begin{aligned} \tilde{P}_{t+1|t} &\stackrel{\triangle}{=} \mathbb{E}(\tilde{\mathbf{X}}_{t+1|t} \tilde{\mathbf{X}}_{t+1|t}^{'} | \mathbb{Y}_t) \\ &= F_t P_{t|t} F_t^{'} + Q_t. \end{aligned} \tag{3.8}$$

The innovation, the difference between the real value and its estimate, is given by,

$$
\begin{aligned}
e_{t+1} &= \mathbf{Y}_{t+1} - \tilde{\mathbf{Y}}_{t+1|t} \\
&= (H_{t+1}\mathbf{X}_{t+1} + \mathbf{V}_{t+1}) - H_{t+1}\tilde{\mathbf{X}}_{t+1|t} \\
&= H_{t+1}\mathbf{X}_{t+1} + \mathbf{V}_{t+1}. \qquad (3.9)
\end{aligned}
$$

The covariance matrix associated with the innovation, denoted by S_{t+1}, can then be calculated as follows,

$$
\begin{aligned}
S_{t+1} &= \mathbb{E}(e_{t+1}e_{t+1}^{'}|\mathbb{Y}_t) \\
&= H_{t+1}\tilde{P}_{t+1|t}H_{t+1}^{'} + R_{t+1}. \qquad (3.10)
\end{aligned}
$$

The covariance between the *a priori* state estimate, Equation (3.7), and the innovation, Equation (3.9), is given by,

$$
\mathbb{E}(\tilde{\mathbf{X}}_{t+1|t}e_{t+1}^{'}|\mathbb{Y}_t) = \tilde{P}_{t+1|t}H_{t+1}^{'}. \qquad (3.11)
$$

The Kalman gain can then be calculated using the innovation covariance, Equation (3.10), and the covariance between the state and the measurement, Equation (3.11). The Kalman gain is the component of the Kalman Filter algorithm that ensures that the filter yields the MMSE estimate,

$$
K_{t+1} = \tilde{P}_{t+1|t}H_{t+1}^{'}S_{t+1}^{-1}. \qquad (3.12)
$$

From Equation (3.12), we can see that the Kalman gain is proportional to the *a priori* error covariance matrix and inversely proportional to the innovation covariance matrix.

Update Equations

An *a posteriori* estimate of the state at time t utilises the prediction equations as follows,

$$
\hat{\mathbf{X}}_{t+1|t+1} = \tilde{\mathbf{X}}_{t+1|t} + K_{t+1}e_{t+1}. \qquad (3.13)
$$

From Equation (3.13), we note that the Kalman gain serves to regulate the influence of the innovation, e_{t+1}, on the *a posteriori* estimate.

The *a posteriori* estimate of the covariance structure of $\hat{\mathbf{X}}_{t+1|t+1}$, is given by,

$$\hat{P}_{t+1|t+1} = \tilde{P}_{t+1|t} - K_{t+1}S_{t+1}K_{t+1}^{'}. \tag{3.14}$$

At each time step in the Kalman filter, the entire history of the state is summarised by a set of sufficient statistics, namely the *a posteriori* state and covariance, $\hat{\mathbf{X}}_{t+1|t+1}$ and $\hat{P}_{t+1|t+1}$, respectively.

The Kalman filter is the Minimum Mean Square Estimator (MMSE) of the state at each time t. When the matrix K_{t+1} in the *a posteriori* update equation, Equation (3.13), is chosen to be the Kalman gain, Equation (3.12), the trace of the *a posteriori* estimate covariance matrix, $\hat{P}_{t+1|t+1}$, is minimised. This is equivalent to minimising the mean square error of the estimates.

The use of the Kalman filter is limited by the ubiquitous non-linearity and non-Gaussian nature of the physical world. Hence, since the publication of the Kalman filter, numerous efforts have been devoted to the generic filtering problem, most often in the Kalman filtering framework.

3.2.3 The Algorithm

With $P_{t|t}$ and $\hat{\mathbf{x}}_{t|t}$ known, the Kalman filter algorithm proceeds from iteration $t \rightarrow t+1$ as shown in Table (3.1). A novel implementation of the Kalman filter algorithm in Object-oriented MATLAB is given in Appendix (C).

Table 3.1: Kalman Filter Algorithm

1.	$\tilde{\mathbf{x}}_{t+1\|t} = F_t\hat{\mathbf{x}}_{t\|t}$	State prediction
2.	$\tilde{P}_{t+1\|t} = F_tP_{t\|t}F_t^{'} + Q_t$	Covariance prediction
3.	$S_{t+1} = H_{t+1}\tilde{P}_{t+1\|t}H_{t+1}^{'} + R_{t+1}$	Innovation prediction
4.	$K_{t+1} = \tilde{P}_{t+1\|t}H_{t+1}^{'}S_{t+1}^{-1}$	Kalman Gain
5.	$\hat{P}_{t+1\|t+1} = \tilde{P}_{t+1\|t} - K_{t+1}S_{t+1}K_{t+1}^{'}$	Covariance Update
6.	$\hat{\mathbf{y}}_{t+1\|t} = H_{t+1}\tilde{\mathbf{x}}_{t+1\|t}$	Measurement Prediction
7.	Receive new measurement $\mathbf{y}_{t+1}$	
8.	$e_{t+1} = H_{t+1}\tilde{\mathbf{x}}_{t+1} + \mathbf{V}_{t+1}$	Calculate the innovation
9.	$\hat{\mathbf{x}}_{t+1\|t+1} = \tilde{\mathbf{x}}_{t+1\|t} + K_{t+1}e_{t+1}$	Update the State Estimate

Alternate Forms of Covariance Update

The *a priori* error covariance matrix, $\tilde{P}_{t+1|t}$, Equation (3.8), is, by definition a positive semi-definite matrix. However, computed values of Equation (3.8) can lose this property due to numerical errors [36]. When this occurs the Kalman gain, Equation (3.12), may have the wrong sign and the estimates of the Kalman filter may diverge. An approach that is used to alleviate these effects, is to use an alternate form of covariance update. Two forms of covariance update are shown in Table (3.2).

Table 3.2: Alternate Forms for Covariance Update in the Kalman Filter

Form	Equation
Standard	$\hat{P}_{t+1\|t+1} = \tilde{P}_{t+1\|t} - K_{t+1}S_{t+1}K_{t+1}^{'}$
Joseph	$\hat{P}_{t+1\|t+1} = [\mathbb{I}_{n_x} - K_{t+1}H_{t+1}]\tilde{P}_{t\|t+1}[\mathbb{I}_{n_x} - K_{t+1}H_{t+1}]' + \tilde{P}_{t+1\|t} + K_{t+1}R_{t+1}K_{t+1}^{'}$

The standard version for the update of the covariance preserves the symmetric structure of $\hat{P}_{t+1|t+1}$, but risks losing positive definiteness due to the subtracted term. In contrast, the Joseph form [37] is guaranteed to preserve positive definiteness, provided that $\tilde{P}_{t|t+1} \geq 0$, at the expense of computational efficiency [38]. The Joseph form can be shown to have an $\approx n_x^3$ computational complexity [39], where n_x refers to the dimension of the state vector $\mathbf{X}_t$. This is because the Joseph requires $1.5n_x^3 + 2n_x^2 + n_x$ floating point operations in its evaluation [40].

3.2.4 Consistency and Performance

The consistency of the state estimates calculated by the Kalman filter can be evaluated via the following statistic,

$$\epsilon_t = \tilde{\mathbf{x}}_t^{'} P_{t|t}^{-1} \tilde{\mathbf{x}}_t.$$

If the filter is consistent, then the $\mathbb{E}[\epsilon_t] = n_x$ [38]. The innovation sequence, Equation (3.9), of a consistent Kalman filter will be a white noise process.

An inconsistent Kalman filter gives rise to a sequence of divergent state estimates [41]. Two symptoms of divergence are: the error covariance matrix P_t increases without bound while the theoretical P_t converges to a steady state, and the Kalman gain approaches 0 [42]. Several solutions have been offered to this problem, see [42]:

1. Increase Q_t, the system noise covariance;

2. Increase Q_t adaptively based on the innovations sequence e_t;

3. Use a subset of the measurement history to form all estimates;

4. Use an alternative form of covariance update, refer to Section (3.2.3);

5. Apply exponential weighting to the measurements; and/or

6. Apply an ad-hoc lower bound to the Kalman gain.

3.3 The Extended Kalman Filter

The Extended Kalman Filter (EKF) was created by S.F. Schmidt [43] as an extension to the Kalman Filter. The earliest application of the EKF was within satellite navigation [43], with more recent applications appearing in the fields of time series, where the EKF has been used to estimate missing observations of economic time series [44] and biochemical networks [45]. The key innovation of the Extended Kalman filter was to approximate or linearise the non-linear process and measurement models prior to applying the recursion equations of the standard Kalman filter.

3.3.1 State Space Representation

Consider the following representations of the process and measurement equations

$$\mathbf{X}_t = f_t\left(\mathbf{X}_{t-1}\right) + \mathbf{W}_{t-1}, \qquad \mathbf{W}_{t-1} \sim N(0, Q_t) \tag{3.15}$$

$$\mathbf{Y}_t = h_t\left(\mathbf{X}_t\right) + \mathbf{V}_t, \qquad \mathbf{V}_t \sim N(0, R_t) \tag{3.16}$$

where,

$\mathbf{X}_t \in \mathbb{R}^{n_x}$	is the state vector,
$\mathbf{Y}_t \in \mathbb{R}^{n_y}$	is the measurement vector,
$f_t(\cdot) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_v} \to \mathbb{R}^{n_x}$	is a non-linear, time variant, deterministic function,
$\mathbf{W}_t \in \mathbb{R}^{n_x}$	is the white, Gaussian, zero mean noise vector with covariance Q_t,
$h_t(\cdot) : \mathbb{R}^{n_x} \times \mathbb{R}^{n_v} \to \mathbb{R}^{n_x}$	is a non-linear, time variant, deterministic function,
$\mathbf{V}_t \in \mathbb{R}^{n_x}$	is the white, Gaussian, zero mean measurement noise vector with covariance R_t.

Equation (3.15) is known as the *system equation* and Equation (3.16) is known as the *measurement equation*. The notation and assumptions are identical to those that were used for the Kalman filter in Section (3.2.1).

If both of the functions $f_t(\cdot)$ and $h_t(\cdot)$ are linear, the problem reduces to that covered by the Kalman filter. If however, the functions are non-linear and sufficiently smooth they can be approximated by linear functions. The linear functions are constructed as locally linearised Jacobian functions about a nominal trajectory as follows,

$$\hat{F} = \left.\frac{\partial f_t(x)}{\partial x}\right|_{x=f_t\left(\hat{\mathbf{X}}_{t|t}\right)},$$

$$\hat{H} = \left.\frac{\partial h_t(x)}{\partial x}\right|_{x=f_t\left(\hat{\mathbf{X}}_{t|t-1}\right)}.$$

The EKF utilises the first term in a Taylor series expansion of the non-linear function about the nominal trajectory. If the actual trajectory, $\mathbf{X}_t$, is not sufficiently close to $\hat{\mathbf{X}}_t$ then higher order terms of the Taylor series expansion must be included.

3.3.2 Derivation of the Extended Kalman Filter Predict-Update Recursions

Consider the non-linear terms in the system and measurement equations given by Equations (3.15) and (3.16), respectively. Linearising the terms $f_t(\mathbf{X}_{t-1})$ and $h_t(\mathbf{X}_t)$ about $\hat{\mathbf{X}}_{t|t}$ and $\hat{\mathbf{X}}_{t|t-1}$, respectively, yields the following,

$$f_t(\mathbf{X}_{t-1}) = f_t(\hat{\mathbf{X}}_{t|t}) + J_{f_t}(\mathbf{X}_t - \hat{\mathbf{X}}_{t|t}), \tag{3.17}$$

$$h_t(\mathbf{X}_t) = h_t(\hat{\mathbf{X}}_{t|t-1}) + J_{h_t}(\mathbf{X}_t - \hat{\mathbf{X}}_{t|t}), \tag{3.18}$$

where,

J_{f_t} is the Jacobian of f_t evaluated at $\hat{\mathbf{X}}_{t|t}$,
J_{h_t} is the Jacobian of h_t evaluated at $\hat{\mathbf{X}}_{t|t-1}$.

Substituting the linearised equations, Equations (3.17) and (3.18), respectively, into Equations (3.15) and (3.16) yields an approximate state space model, which can then be used within the Kalman filter. The approximate state space model is given by,

$$\mathbf{X}_{t+1} = J_{f_t}\mathbf{X}_t + \mathbf{M}_t + \mathbf{W}_t,$$
$$\mathbf{Y}_{t+1} = J_{h_t}\mathbf{X}_t + \mathbf{N}_t + \mathbf{V}_t,$$

where,

$\mathbf{M}_t = f_t(\hat{\mathbf{X}}_{t|t}) - J_{f_t}\hat{\mathbf{X}}_{t|t}$,
$\mathbf{N}_t = \hat{\mathbf{X}}_{t|t-1} - J_{h_t}\hat{\mathbf{X}}_{t|t-1}$.

3.3.3 The Algorithm

With $P_{t|t}$ and $\hat{\mathbf{x}}_{t|t}$ known, the Extended Kalman filter algorithm proceeds from iteration $t \rightarrow t+1$ as shown in Table (3.3). A novel implementation of the Extended Kalman filter algorithm in Object-oriented MATLAB is given in Appendix (C).

Table 3.3: Extended Kalman Filter Algorithm

1.	Using $\hat{\mathbf{x}}_{t\|t}$, calculate J_{f_t}	Jacobian Calculation
2.	$\tilde{\mathbf{x}}_{t+1\|t} = f_t(\hat{\mathbf{x}}_{t\|t})$	State prediction
3.	Using $\tilde{\mathbf{x}}_{t+1\|t}$, calculate $J_{h_{t+1}}$	Jacobian Calculation
4.	$\tilde{P}_{t+1\|t} = J_{f_t} P_{t\|t} J_{f_t}^{'} + Q_t$	Covariance prediction
5.	$S_{t+1} = J_{h_{t+1}} \tilde{P}_{t+1\|t} J_{h_{t+1}}^{'} + R_{t+1}$	Innovation prediction
6.	$K_{t+1} = \tilde{P}_{t+1\|t} J_{h_{t+1}}^{'} S_{t+1}^{-1}$	Kalman Gain
7.	$\hat{P}_{t+1\|t+1} = \tilde{P}_{t+1\|t} - K_{t+1} S_{t+1} K_{t+1}^{'}$	Covariance Update
8.	Receive new measurement $\mathbf{y}_{t+1}$	
9.	$e_{t+1} = \mathbf{y}_{t+1} - h_{t+1}(\hat{\mathbf{x}}_{t\|t})$	Calculate the innovation
10.	$\hat{\mathbf{x}}_{t+1\|t+1} = \tilde{\mathbf{x}}_{t+1\|t} + K_{t+1} e_{t+1}$	Update the State Estimate

3.3.4 Consistency and Performance

The performance of the Extended Kalman Filter (EKF) depends on the linear or non-linear nature of the system and measurement equations. In the case where both are linear, the performance of the EKF is the same as the Kalman filter, which is discussed in Section (3.2.4).

In non-linear situations, the performance of the Extended Kalman Filter can be unstable, due to two well known reasons:

1. Violations of the local linearity assumption; and

2. The derivation of the Jacobian matrices.

The linearisation will fail if R_t or Q_t are too large, since this results in large perturbations from the nominal trajectory. The derivation of the Jacobian matrices are non-trivial in most applications and often lead to significant implementation difficulties. Furthermore, in some situations, the Jacobian matrices may not exist at all.

An alternative to the Extended Kalman Filter which provides performance equivalent to that of the Kalman filter for linear systems, is known as the Unscented Kalman Filter (UKF) [46]. The UKF, which approximates a Guassian distribution rather than an arbitrary non-linear function or transformation, [46], addresses the shortfalls of the EKF, without increasing computational burden [42]. For further information, refer to [46], [42].

3.4 Monte Carlo Methods

Monte Carlo methods use statistical sampling and estimation techniques to evaluate the solutions to mathematical problems.

Consider a probability distribution, p, from which we want to determine a point-estimate, such as the maximum likelihood or the method of moments. Such an estimate can be determined, for any suitable function f, through the following expression,

$$p(f) = \mathbb{E}[f(\mathbf{X})] = \int f(\mathbf{X})p(dX). \tag{3.19}$$

Analytic calculations of *p(f)*, Equation (3.19), are rarely possible and in order to determine such estimates an empirical approximation must be used. Such an approximation is given by,

$$\hat{p}(f) = \frac{1}{N_s}\sum_{i=1}^{N_s} f(\mathbf{X}^{(i)}),$$

where,

N_s	is the number of independent samples from $f(\mathbf{X})$,
$f(\mathbf{X}^{(i)})$	is the i^{th} sample from $f(\mathbf{X})$.

In most practical situations it is infeasible to draw samples directly from the target distribution p. An oft used solution is to introduce a proposal distribution from which it is easy to sample and whose density can be evaluated point wise. A weighing and/or selection mechanism is used to correct for bias from the proposal distribution. A good proposal distribution should take into account key features of the target, such as multi-modality, and ideally should be as close as possible to the target distribution. As the dimension increases, finding a good proposal distribution becomes more difficult.

Two classes of Monte Carlo algorithms that are often used for sampling from high-dimensional distributions and Markov Chain Monte Carlo (MCMC) and Sequential Monte Carlo (SMC). MCMC relies on sampling a realization of a Markov chain with invariant distribution $p(f)$. SMC, on the other hand is a sequential implementation of importance sampling, employing a population of samples and a sequence of probability distributions, with the final distribution being $p(f)$. In the proceeding sections we introduce two fundamental techniques that can be used to generate random samples from a proposal distribution: rejection sampling and importance sampling.

3.4.1 Rejection Sampling

Rejection sampling is a technique introduced by von Neumann [47] which samples from a target distribution, $p(\mathbf{X})$, known up to a proportionality constant, by sampling from another easy to sample proposal distribution $q(\mathbf{X})$. The method assumes that there exists a known constant $C < \infty$ such that $p(\mathbf{X}) \leq Cq(\mathbf{X})$ for every $\mathbf{x}$. The Rejection sampling procedure is presented in Table (3.4).

Table 3.4: Rejection Sampling Algorithm

1.	Draw $\mathbf{X} \sim q$
2.	Accept $\mathbf{X}$ as a sample from p with probability $\frac{p(\mathbf{X})}{Cq(\mathbf{X})}$

The samples from rejection sampling are exact and the acceptance probability for a random variable is inversely proportional to the constant C [48]. Rejection sampling works best if the proposal distribution is a good approximation of the target distribution. The choice of constant C is also critical; if C is too small, the samples are not reliable because of low rejection rate; if C is too large, the algorithm will be inefficient since the acceptance rate will be low.

From a Bayesian perspective, if the prior, $p(\mathbf{X})$, is used as the proposal distribution, $q(\mathbf{X}_{1:t}|\mathbf{Y}_{1:t})$ and the likelihood $p(\mathbf{Y}_{1:t}|\mathbf{X}_{1:t}) \leq C$ where C is assumed to be known, then the bound on the posterior is given by,

$$\begin{aligned} p(\mathbf{X}_{1:t}|\mathbf{Y}_{1:t}) &= \frac{p(\mathbf{Y}_{1:t}|\mathbf{X}_{1:t})p(\mathbf{X}_t)}{p(\mathbf{Y}_{1:t})} \\ &< \frac{Cq(\mathbf{X}_{1:t}|\mathbf{Y}_{1:t})}{p(\mathbf{Y}_{1:t})} \\ &\equiv C^* q(\mathbf{X}_{1:t}|\mathbf{Y}_{1:t}), \end{aligned}$$

where,

$C^* \quad = \frac{C}{p(\mathbf{Y}_{1:t})}$.

The acceptance rate for a sample is given by,

$$\frac{p(\mathbf{X}_{1:t}|\mathbf{Y}_{1:t})}{C^* q(\mathbf{X})}.$$

3.4.2 Importance Sampling

Importance sampling (IS) was first introduced by Marshall in 1954 [49] and aims to sample a probability distribution in a region of "importance". The idea of importance sampling is to choose a proposal distribution $q(\mathbf{X})$ in place of the true probability distribution $p(\mathbf{X})$, which is difficult to sample. The support of $q(\mathbf{X})$ is assumed to cover that of $p(\mathbf{X})$.

Under these assumptions, the Monte Carlo integration problem, shown in Equation (3.19), can be rewritten as,

$$\int f(\mathbf{X})p(\mathbf{X})d\mathbf{X} = \int f(\mathbf{X})\frac{p(\mathbf{X})}{q(\mathbf{X})}q(\mathbf{X})d\mathbf{X}. \tag{3.20}$$

Importance sampling can be used to draw a number of independent samples from $q(\mathbf{X})$ to obtain an estimate of Equation (3.20). Each sample, $f(\mathbf{X}^{(i)})$, is assigned an importance weight, $W(\mathbf{X}^{(i)})$, as follows,

$$\hat{f} = \frac{1}{N_s} \sum_{i=1}^{N_s} W(\mathbf{X}^{(i)}) f(\mathbf{X}^{(i)}), \tag{3.21}$$

where,

N_s	is the number of independent samples,
$W(\mathbf{X}^{(i)}) \propto \frac{p(\mathbf{X}^{(i)})}{q(\mathbf{X}^{(i)})}$	are the un-normalised importance weights.

The un-normalised importance weights can be normalised using,

$$\tilde{W}(\mathbf{X}^{(i)}) = \frac{W(\mathbf{X}^{(i)})}{\sum_{j=1}^{N_s} W(\mathbf{X}^{(j)})}.$$

The IS approximation of Equation (3.20) utilising normalised importance weights is given then by,

$$\hat{f} = \frac{1}{N_s} \sum_{i=1}^{N_s} \tilde{W}(\mathbf{X}^{(i)}) f(\mathbf{X}^{(i)}). \tag{3.22}$$

The IS estimate given by Equation (3.22) is biased but consistent with the bias vanishing at a rate $O(N_s)$ [50], that is, the bias vanishes at a rate that is bounded from above by N_s.

For importance sampling to perform well in practice, it is important that the variance of the importance weights are finite. If this condition is not satisfied then the variance of the corresponding estimate given by Equation (3.22) will be infinite. A finite importance weight can be achieved through the careful selection of the proposal distribution $q(\mathbf{X})$.

Choice of Importance Distribution

An intuitive choice for an proposal distribution would be to choose $q(\mathbf{X})$ to be as close to possible to $p(\mathbf{X})$ such that the variance of the estimate, Equation (3.21), is minimised. The variance of Equation (3.22) can be written as [50],

$$
\begin{aligned}
\mathrm{Var}\left[\hat{f}\right] &= \frac{1}{N_s}\mathrm{Var}\left[f(\mathbf{X})W(\mathbf{X})\right] \\
&= \frac{1}{N_s}\mathrm{Var}\left[f(\mathbf{X})\frac{p(\mathbf{X})}{q(\mathbf{X})}\right] \\
&= \frac{1}{N_s}\int\left[f(\mathbf{X})\frac{p(\mathbf{X})}{q(\mathbf{X})} - \mathbb{E}\left[f(\mathbf{X})\right]\right]^2 q(\mathbf{X})d\mathbf{X} \\
&= \frac{1}{N_s}\int\left[\left(\frac{(f(\mathbf{X})p(\mathbf{X}))^2}{q(\mathbf{X})}\right) - 2p(\mathbf{X})f(\mathbf{X})\mathbb{E}\left[f(\mathbf{X})\right]\right]d\mathbf{X} + \frac{\mathbb{E}\left[f(\mathbf{X})\right]^2}{N_s} \\
&= \frac{1}{N_s}\int\left[\left(\frac{(f(\mathbf{X})p(\mathbf{X}))^2}{q(\mathbf{X})}\right)\right]d\mathbf{X} + \frac{\mathbb{E}\left[f(\mathbf{X})\right]^2}{N_s} \qquad (3.23)
\end{aligned}
$$

It is clear that Equation (3.23) is minimised when $q(\mathbf{X}) = |f(\mathbf{X})|p(\mathbf{X})$. However, this choice cannot be made as it precludes the use of the Importance Sampling algorithm. This simple result indicates that it is sensible to choose an proposal distribution that is as close as possible to the target distribution $p(\mathbf{X})$. Also, although it is possible to construct samplers for which the variance is finite without satisfying this condition, it is advisable to select $q(\mathbf{X})$ so that $W(\mathbf{X}^{(i)}) < C < \infty$ [51].

In some cases one knows the function of interest f prior to selection of an proposal distribution. If this is true then the optimal proposal distribution, $q^{optimal}(\mathbf{X})$ [52], can be determined as follows,

$$
q^{optimal}(\mathbf{X}) = \frac{|f(\mathbf{X})|p(\mathbf{X})}{\int |f(\mathbf{Z})|f(\mathbf{Z})d\mathbf{Z}}. \qquad (3.24)
$$

3.5 Sequential Monte Carlo Methods

Sequential Monte Carlo (SMC) methods are a technique for implementing recursive Bayesian filter via Monte Carlo (MC) simulations.

Under this framework, SMC methods sample from a sequence of distributions distributions, p_t, defined on the support E^t, via importance sampling. The key idea is to represent the required posterior density function at time t by a set of random samples with associated weights [35]. Estimates are then computed on the samples and their associated weights.

In the Sequential Monte Carlo framework the posterior distribution is empirically represented by a weighted sum of N_s samples as follows,

$$p_t(\mathbf{X}_{1:t}|\mathbf{Y}_{1:t}) \approx \frac{1}{N_s}\sum_{t=1}^{N_s} w_t^{(i)}\delta(\mathbf{X}_t - \mathbf{X}_t^{(i)})$$
$$\equiv \hat{p}_t(\mathbf{X}_{1:t}|\mathbf{Y}_{1:t}), \tag{3.25}$$

where,

N_s	is the number of independent samples,
$w_t^{(i)}$	is the weight of sample i at time t,
$\mathbf{X}_t^{(i)}$	are i.i.d. samples drawn from $p_t(\mathbf{X}_{1:t}\|\mathbf{Y}_{1:t})$ at time t,
$\delta(\cdot)$	is the delta dirac function.

When N_s is sufficiently large, $\hat{p}_t(\mathbf{X}_{1:t}|\mathbf{Y}_{1:t})$, Equation (3.25), approximates the true posterior $p_t(\mathbf{X}_{1:t}|\mathbf{Y}_{1:t})$.

The structure of a typical Sequential Monte Carlo algorithm consists of three iterative operations: mutation, correction and selection [53]. The mutation step consists of moving samples $\mathbf{X}_{t-1}$ to $\mathbf{X}_t$ via a mutation kernel, $M_t(\mathbf{X}_{t-1}, \mathbf{X}_t)$ [54]. In the correction stage, the samples are assigned importance weights via a weight function $V_t(\mathbf{X}_t)$ [53]. The final stage, selection, involves replacing the current set of samples, $\mathbf{X}_t$, with a new, uniformly weighted set of samples, $\tilde{\mathbf{X}}_t$, often chosen by a resampling scheme. The selection stage acts to retain samples with large importance weights to serve as the starting point for the next mutation step. A pictorial representation of the SMC algorithm is given in Figure (3.1).

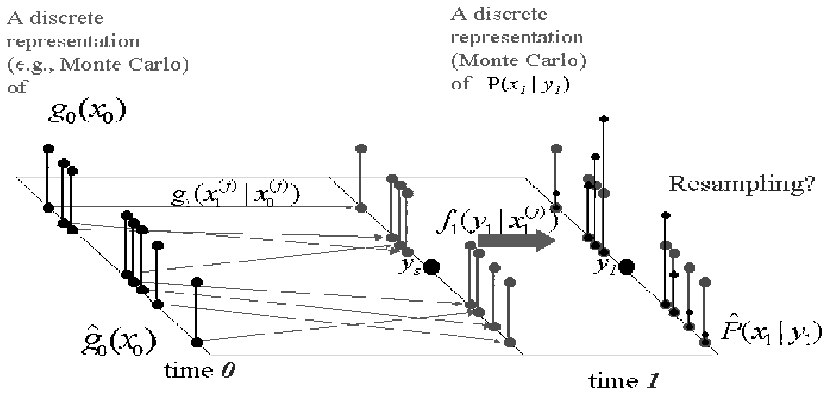


Figure 3.1: A pictorial representation of a Sequential Monte Carlo algorithm. At time 0, one has a discrete representation of the current posterior distribution. One propagates to time 1 by first sampling those $x_1^{(m)}$ from its prior distribution and then correcting this sampling by an importance reweighting and resampling, [55]. Note the change in notation.

This general structure encompasses the vast majority of SMC algorithms, two examples of such algorithms include the Sequential Importance Sampling (SIS) and Sequential Importance Re-

sampling (SISR).

3.5.1 Sequential Importance Sampling

The Sequential Importance Sampling (SIS) algorithm is a Monte Carlo method that forms the basis for most sequential Monte Carlo filters [35]. The SIS algorithm alternates the mutation and correction steps of the typical Sequential Monte Carlo (SMC) algorithm, but does not perform a selection stage. As such, the importance weights are not initialised to one at each iteration and are updated recursively.

To motivate the discussion of the recursive importance weight updating mechanism utilised by the Sequential Importance Sampling (SIS) algorithm, we begin by considering the sequential construction of the proposal distribution $q(\mathbf{X})$. The proposal distribution constructed by the SIS algorithm has the form $q(\mathbf{X}_{1:t}|\mathbf{Y}_{1:t})$. If the proposal distribution, $q(\mathbf{X}_{1:t}|\mathbf{Y}_{1:t})$, is chosen to factorise as,

$$q(\mathbf{X}_{1:t}|\mathbf{Y}_{1:t}) = q(\mathbf{X}_1)\prod_{t=2}^{n} q(\mathbf{X}_t|\mathbf{X}_{1:t-1}, \mathbf{Y}_{1:t}), \tag{3.26}$$

then importance sampling can be performed recursively. The posterior density can be expressed in a recursive form as,

$$\begin{aligned} p(\mathbf{X}_{1:t}|\mathbf{Y}_{1:t}) &= \frac{p(\mathbf{Y}_t|\mathbf{X}_{1:t}|\mathbf{Y}_{1:t-1})p(\mathbf{X}_{1:t}|\mathbf{Y}_{1:t-1})}{p(\mathbf{Y}_t|\mathbf{Y}_{1:t-1})} \\ &= \frac{p(\mathbf{Y}_t|\mathbf{X}_{1:t}|\mathbf{Y}_{1:t-1})p(\mathbf{X}_t, \mathbf{X}_{1:t-1}, \mathbf{Y}_{1:t-1})p(\mathbf{X}_{1:t-1}|\mathbf{Y}_{1:t-1})}{p(\mathbf{Y}_t|\mathbf{Y}_{1:t-1})} \\ &= p(\mathbf{X}_{1:t-1}|\mathbf{Y}_{1:t-1})\frac{p(\mathbf{Y}_t|\mathbf{X}_t)p(\mathbf{X}_t|\mathbf{X}_{t-1})}{p(\mathbf{Y}_t|\mathbf{Y}_{1:t-1})} \\ &\propto p(\mathbf{X}_{1:t-1}|\mathbf{Y}_{1:t-1})p(\mathbf{Y}_t|\mathbf{X}_t)p(\mathbf{X}_{\ t}|\mathbf{X}_{t-1}). \end{aligned} \tag{3.27}$$

The recursive factorisations of the importance and posterior densities, Equations (3.26) and (3.27), respectively, can then be used to derive the recursive calculation of importance weights, according to,

$$\begin{aligned} W_t(\mathbf{X}_{1:t}) &= \frac{p(\mathbf{X}_{1:t}|\mathbf{Y}_{1:t})}{q(\mathbf{X}_{1:t}|\mathbf{Y}_{1:t})} \\ &\propto \frac{p(\mathbf{Y}_t|\mathbf{X}_t)p(\mathbf{X}_t|\mathbf{X}_{t-1})p(\mathbf{X}_{1:t-1}|\mathbf{Y}_{1:t-1})}{q(\mathbf{X}_t|\mathbf{X}_{1:t-1}, \mathbf{Y}_{1:t-1})q(\mathbf{X}_{1:t-1}|\mathbf{Y}_{1:t-1})} \\ &= W_{t-1}(\mathbf{X}_{1:t-1})\frac{p(\mathbf{Y}_t|\mathbf{X}_t)p(\mathbf{X}_t|\mathbf{X}_{t-1})}{q(\mathbf{X}_{\ t}|\mathbf{X}_{1:t-1}, \mathbf{Y}_{1:t})}. \end{aligned} \tag{3.28}$$

Furthermore, if $q(\mathbf{X}_t|\mathbf{X}_{1:t-1}, \mathbf{Y}_{1:t})$ can be expressed as $q(\mathbf{X}_t|\mathbf{X}_{1:t-1}, \mathbf{Y}_t)$, the recursive weight update, Equation (3.28), can be further simplified to,

$$W_t(\mathbf{X}_{1:t}) = W_{t-1}(\mathbf{X}_{1:t-1}) \frac{p(\mathbf{Y}_t|\mathbf{X}_t)p(\mathbf{X}_t|\mathbf{X}_{t-1})}{q(\mathbf{X}_t|\mathbf{X}_{1:t-1}, \mathbf{Y}_t)}. \tag{3.29}$$

The SIS algorithm consists of the recursive propagation of samples and weights as each measurement is received.

The Sequential Importance Sampling Algorithm proceeds from iteration $t \to t+1$ as shown in Table (3.5).

Table 3.5: Sequential Importance Sampling Algorithm

1.	Draw N_s i.i.d particles from the importance density $\mathbf{X}_{t+1}^{(i)} \sim p(\mathbf{X}_{t+1}|\mathbf{X}_t^{(i)})$
2.	Update the weights according to Equation (3.29), incrementing the time subscript by 1

Despite the convenient recursive structure of the SIS algorithm, it also suffers from significant drawbacks. The *degeneracy* problem occurs when after a few iterations of the SIS algorithm, only a few of the sample weights will be non-zero. Furthermore, the importance weights used in the algorithm may have large variances, resulting in inaccurate estimates [56]. The practical implication of this is the loss of computational efficiency, since samples with trivial importance weights will be updated.

Effective Sample Size

The effective sample size is a convenient heuristic that can be used to quantify the extent of degeneracy in a Sequential Monte Carlo algorithm [57] and can be used to design resampling strategies. The effective sample size, N_{eff}, is given by,

$$\mathrm{N}_{eff} = \frac{N_s}{1 + \mathrm{Var}\left(\tilde{W}_t(\mathbf{X}_{1:t})\right)}, \tag{3.30}$$

where,

N_s is the number of samples,
$\tilde{W}_t(\mathbf{X}_{1:t}) = \frac{W_t(\mathbf{X}_t)}{\sum_{j=1}^{N_s} W_t(\mathbf{X}_t)}$.

The use of the effective sample size, Equation (3.30), as a measure of the efficiency of a Sequential Importance Sampling (SIS) algorithm seen by using the delta method [58]. We begin by considering two estimators of f in Equation (3.19), the Importance Sampling estimate, $\hat{f}_{IS}$, whose form is given in Equation (3.21) and a crude Monte Carlo estimate of the form,

$$\hat{f}_{MC} = \frac{1}{N_s}\sum_{i=1}^{N_s} f(\mathbf{Y}_i).$$

The two estimates are related to each other in the following form,

$$\begin{aligned}\hat{\mu} &= \frac{\hat{f}_{IS}}{\hat{f}_{MC}} \\ &= \frac{Z}{W}.\end{aligned} \tag{3.31}$$

The variance of $\hat{\mu}$, Equation(3.31), can be determined using the standard delta method for ratio statistics, which may be viewed as a combination of a suitable central limit theorem and a Taylor series expansion. We begin by noting that,

$$\mathrm{Var}(\hat{\mu}) \approx \frac{1}{N_s}\left[\mu^2\mathrm{Var}(W) + \mathrm{Var}(Z) - 2\mu\mathrm{Cov}(W,Z)\right]. \tag{3.32}$$

Denoting $H = f(\mathbf{X}_{1:t})$, we observe that,

$$\begin{aligned}\mathrm{Cov}(W,Z) &= \mathbb{E}(HW) - \mu \\ &= \mathrm{Cov}(W,H) + \mu\mathbb{E}(W) - \mu.\end{aligned} \tag{3.33}$$

Similarly,

$$\begin{aligned}\mathrm{Var}(Z) &= \mathbb{E}(WH^2) - \mu^2 \\ &\approx \mathbb{E}(W)\mathbb{E}^2(H) + \mathrm{Var}(H)\mathbb{E}(W) + 2\mu\mathrm{Cov}(W,H) - \mu^2.\end{aligned} \tag{3.34}$$

The remainder term in Equation (3.34) is given by,

$$\mathbb{E}[\{W - \mathbb{E}(W)\}(H-\mu)^2].$$

Substituting Equations (3.33) and (3.34), respectively, into Equation (3.32), we find that,

$$\mathrm{Var}(\hat{\mu}) \approx \frac{1}{N_s}\left(\frac{\mathrm{Var}(H)}{1+\mathrm{Var}(W)}\right).$$

The relative efficiency of estimating f by $\hat{f}_{MC}$ instead of $\hat{f}_{IS}$ is given by,

$$\frac{\text{Var}\left[\hat{f}_{MC}\right]}{\text{Var}\left[\hat{f}_{IS}\right]} \approx \frac{1}{1+\text{Var}\left[\tilde{W}_t(\mathbf{X}_{1:t})\right]}, \tag{3.35}$$

where,

$$\tilde{W}_t(\mathbf{X}_{1:t}) \quad = \frac{W_t(\mathbf{X}_t)}{\sum_{j=1}^{N_s} W_t(\mathbf{X}_t)}.$$

Equation (3.35) can be interpreted as meaning that the N_s weighted samples are worth $1/\left(1+\text{Var}\left[\tilde{W}_t(\mathbf{X}_{1:t})\right]\right)$ i.i.d. samples drawn from the target distribution [55]. In practice, the true N_{eff} is not available, thus its estimate, $\hat{N}_{eff}$, is used instead. $\hat{N}_{eff}$ is given by,

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N_s}(\tilde{\mathbf{W}}_t^{(i)})^2}. \tag{3.36}$$

3.5.2 Sequential Importance Sampling Resampling

The Sequential Importance Sampling Resampling (SISR) algorithm was first used by Rubin in 1987 in the field of Monte Carlo inference [59]. The innovation behind the SIR algorithm is to insert a resampling step between two importance sampling steps in the Sequential Importance Sampling (SIS) algorithm. The resampling step works to rectify the degeneracy problem of the SIS algorithm by eliminating samples with trivial importance weights and propagating samples with larger weights.

The Sequential Importance Sampling Resampling algorithm proceeds from iteration $t \rightarrow t+1$ as shown in Table (3.6). A novel implementation of the Sequential Importance Sampling Resampling algorithm in Object-oriented MATLAB is given in Appendix (D).

Table 3.6: Sequential Importance Sampling Resampling Algorithm

1.	Draw N_s i.i.d particles from the importance density $\mathbf{X}_{t+1}^{(i)} \sim p(\mathbf{X}_{t+1}\|\mathbf{X}_t^{(i)})$
2.	Update the weights according to Equation (3.29), incrementing the time subscript by 1
3.	Normalise the weights, $w_{t+1}^{(i)} = \frac{w_{t+1}^{(i)}}{\sum_{j=1}^{N_s} w_{t+1}^{(i)}}$
4.	Perform Multinomial Sampling, Section (3.5.4), if $\hat{N}_{eff}$, Equation (3.36), $\leq$ Threshold

The resampling step, Step 4, of the Sequential Importance Sampling Resampling algorithm alleviates the degeneracy problem but can, in turn, introduce *sample impoverishment*. A plethora of resampling schemes have been proposed, with each differing in their properties and the amount of Monte Carlo variation introduced, refer to Section 3.5.4. Sample impoverishment arises through the repeated selection of samples with high importance weights, which leads to a loss of diversity

amongst the samples. A commonly used technique to circumvent sample impoverishment is to use Markov Chain Monte Carlo resampling schemes.

3.5.3 Auxiliary Particle Filter

The Auxiliary Particle Filter (APF) was first introduced by Pitt and Shephard in 1999 [60] and is a popular alternative to Sequential Importance Sampling Resampling (SISR) algorithms. The key innovation behind the APF is to modify the original sequence of target distributions to guide samples in promising regions [61].

In the Auxiliary Particle Filter, an auxiliary variable, corresponding to a sample index, is generated according to a distribution which weights each sample in terms of its compatibility with the current measurement [62]. Then the new state value is sampled as the offspring of the sample indicated by this auxiliary variable.

The Auxiliary Particle filter algorithm proceeds from iteration $t \rightarrow t+1$ as shown in Table (3.7). A novel implementation of the Auxiliary Particle filter algorithm in Object-oriented MATLAB is given in Appendix (E).

Table 3.7: Auxiliary Particle Algorithm

1.	Draw N_s $u_{t+1}^{(i)} \sim p(\mathbf{X}_{t+1}\|\mathbf{X}_t^{(i)})$
2.	Update the weights according to $w_{t+1}^{(i)} = p(\mathbf{Y}_{t+1}\|u_{t+1}^{(i)})w_t^{(i)}$
3.	Normalise the weights, $w_{t+1}^{(i)} = \frac{w_{t+1}^{(i)}}{\sum_{j=1}^{N_s} w_{t+1}^{(i)}}$
4.	Perform Multinomial Sampling, Section (3.5.4), from the discrete measure given by the inputs $\mathbf{X}_t^{(i)}$ and the normalised weights, tracking the parent of the j^{th} sample, i^j
5.	Draw N_s i.i.d particles from the importance density $\mathbf{X}_{t+1}^{(i)}$
6.	Update the weights according to $w_{t+1}^{(i)} = \frac{p(\mathbf{Y}_{t+1}\|\mathbf{X}_{t+1}^{(j)})}{p(\mathbf{Y}_{t+1}\|u_{t+1}^{(ij)})}$
7.	Normalise the weights, $w_{t+1}^{(i)} = \frac{w_{t+1}^{(i)}}{\sum_{j=1}^{N_s} w_{t+1}^{(i)}}$

It is immediately apparent that, as SISR and the APF are essentially the same algorithm with a different choice of importance weights [61], shown in Step 6, of Table (3.7).

3.5.4 Resampling Schemes

In the context of Sequential Monte Carlo (SMC) algorithms, resampling consists of selecting a set of new samples and weights, such that the disparity between the weights is reduced. As such, resampling constitutes a crucial and computationally expensive part of SMC algorithms [63]. A plethora of resampling methods have been proposed, the most basic of which are the multinomial

[32], residual [64], stratified [65] and systematic [35] resampling. The resampling methods are presented in the proceeding sections. It is of interest to note that all of the methods are unbiased and can be implemented in computational complexity of $O(N_s)$ [63]. A novel implementation of the Multinomial, Residual, Stratified and Systematic resampling schemes in Object-oriented MATLAB is given in Appendix (F).

In the proceeding sections we assume that at time t we have a collection of samples and their associated weights, $\left\{\mathbf{X}_t^{(i)}, w_t^{(i)}\right\}_{i=1}^{N_s}$, and allow $\left\{\tilde{\mathbf{X}}_t^{(i)}\right\}_{i=1}^{N_s}$ to denote the collection of samples after resampling.

Multinomial Resampling

Multinomial resampling is the simplest resampling scheme and is based on an idea that is at the core of the bootstrap method [66]. In practice, the method utilises repeated applications of the "inversion" method, which uses the inverse of the cumulative distribution associated with the normalised weights, $F^{-1}(\cdot)$. An outline of the algorithm is provided in Table (3.8).

Table 3.8: Multinomial Resampling Algorithm

1.	Draw $\left\{U^i\right\}_{1\leq i\leq N_s}$ uniform random numbers on the interval $(0,1]$
2.	Select $\tilde{\mathbf{X}}^{(i)} = \mathbf{X}^{(i)}\left(F^{-1}(U^i)\right)$

This is equivalent to drawing a vector of replicate counts, $\mathbf{M}_t$, from a multinomial distribution with N_s trials. The Multinomial distribution in this case is given in Equation (3.37).

$$\text{Multinomial}(\mathbf{M}_t|N_s,\mathbf{W}_t) = \begin{cases} \frac{N_s!}{\prod_{i=1}^{N_s} M_t^{(i)}!}\prod_{i=1}^{N_s} w_t^{M_t^{(i)}} & ,\text{if } \sum_{i=1}^{N_s} M_t^{(i)} = N_s \\ 0 & ,\text{otherwise.} \end{cases} \tag{3.37}$$

The variance of the multinomial resampling scheme is given by [66],

$$\text{Var}\left[\frac{1}{N_s}\sum_{i=1}^{N_s} f(\tilde{\mathbf{X}}_t^{(i)})\right] = \frac{1}{N_s}\left\{\sum_{i=1}^{N_s} w_t^{(i)} f^2(\mathbf{X}_t^{(i)}) - \left[\sum_{i=1}^{N_s} w_t^{(i)} f(\mathbf{X}_t^{(i)})\right]^2\right\}. \tag{3.38}$$

Residual Resampling

Residual sampling is a technique that can be used to decrease the variance due to resampling [57]. An outline of the residual resampling algorithm is shown in Table (3.9).

By deterministically replicating those particles that we expect at least one of in the replicated set and randomly replicating the non-integer (residual) components of $\left\lfloor N_s w_t^{(i)} \right\rfloor$, we retain the unbi-

Table 3.9: Residual Resampling Algorithm

1.	Allocate $n^{(i)} = \left\lfloor N_s w_t^{(i)} \right\rfloor$ copies of sample $\mathbf{X}_t^{(i)}$ to $\tilde{\mathbf{X}}_t^{(i)}$
2.	Set $R = \sum_{i=1}^{N_s} n^{(i)}$
3.	Resample $N_s - R$ samples from $\mathbf{X}_t$ according to the Multinomial distribution, Equation (3.37)

ased behaviour of the Multinomial resampling scheme whilst reducing the variance it introduces [67].

To calculate the variance of the residual resampling we begin by decomposing the estimator into,

$$\frac{1}{N_s}\sum_{i=1}^{N_s} f(\tilde{\mathbf{X}}_t^{(i)}) = \sum_{i=1}^{m} \frac{\left\lfloor N_s w_t^{(i)} \right\rfloor}{N_s} f(\mathbf{X}_t^{(i)}) + \frac{1}{N_s}\sum_{i=1}^{N_s-R} f(\mathbf{X}_t^{M_t^{(i)}}). \tag{3.39}$$

From Equation (3.39), it is evident that the estimator involves both deterministic and stochastic terms. The variance of residual resampling is thus given by [66],

$$\begin{aligned}
\frac{1}{(N_s)^2}\mathrm{Var}\left[\sum_{i=1}^{N_s-R} f(\mathbf{X}_t^{M_t^{(i)}})\right] &= \frac{N_s - R}{(N_s)^2}\mathrm{Var}\left[f(\mathbf{X}_t^{M_t^{(i)}})\right] \\
&= \frac{1}{N_s}\sum_{i=1}^{N_s} w_t^{(i)} f^2(\mathbf{X}_t^{(i)}) - \sum_{i=1}^{N_s} \frac{\left\lfloor N_s w_t^{(i)} \right\rfloor}{(N_s)^2} f^2(\mathbf{X}_t^{(i)}) \\
&\quad - \frac{N_s - R}{N_s}\left\{\sum_{i=1}^{N_s} \bar{w}^{(i)} f(\mathbf{X}_t^{(i)})\right\}^2,
\end{aligned} \tag{3.40}$$

where,

$$\bar{w}^{(j)} \quad = \mathbb{P}\left(M_t^{(i)} = j\right), \quad i = 1, \ldots, N_s - R, \quad j = 1, \ldots, N_s.$$

To compare the variance of the residual resampling scheme to the multinomial resampling scheme, we first write,

$$\sum_{i=1}^{N_s} w_t^{(i)} f(\mathbf{X}_t^{(i)}) = \sum_{i=1}^{N_s} \frac{\left\lfloor N_s w_t^{(i)} \right\rfloor}{N_s} f(\mathbf{X}_t^{(i)}) - \frac{N_s - R}{N_s}\sum_{i=1}^{N_s} \bar{w}^{(i)} f(\mathbf{X}_t^{(i)}). \tag{3.41}$$

Applying Jensen's inequality to the square of the right hand side of Equation (3.41), yields,

$$\left\{\sum_{i=1}^{N_s} w_t^{(i)} f(\mathbf{X}_t^{(i)})\right\}^2 \leq \sum_{i=1}^{N_s} \frac{\left\lfloor N_s w_t^{(i)} \right\rfloor}{N_s} f^2(\mathbf{X}_t^{(i)}) - \frac{N_s - R}{N_s}\left\{\sum_{i=1}^{N_s} \bar{w}^{(i)} f(\mathbf{X}_t^{(i)})\right\}^2. \tag{3.42}$$

Upon combining Equation (3.42) with Equation (3.40), we can see that the variance of residual resampling is always smaller than that of multinomial resampling, Equation (3.38).

Stratified Resampling

Stratification is a method that originated from survey sampling [63]. In the stratified resampling scheme, the domain of the random variable is partitioned into N_s disjoint sets, such that $(0,1] = (0,1/N_s] \cup, \ldots, \cup (N_s - 1/N_s, 1]$. Sampling is then conducted from the restricted density in each strata using multinomial resampling [63]. An outline of the residual resampling algorithm is shown in Table (3.10).

Table 3.10: Stratified Resampling Algorithm

1.	Draw $\left\{I^i\right\}_{1\leq i\leq N_s}$ random numbers according to $I^i = \frac{(i-1)+U^i}{N_s}$, where $U^i \sim Uniform((0,1])$
2.	Select $\tilde{\mathbf{X}}^{(i)} = \mathbf{X}^{(i)}\left(F^{-1}(I^i)\right)$

To calculate the variance of the stratified resampling is given by [66],

$$\begin{aligned}\text{Var}\left[\frac{1}{N_s}\sum_{i=1}^{N_s} f(\tilde{\mathbf{X}}_t^{(i)})\right] &= \frac{1}{(N_s)^2}\sum_{i=1}^{N_s}\text{Var}\left[f\circ\tilde{\mathbf{X}}_t\circ F^{-1}(U^i)\right] \\ &= \frac{1}{N_s}\sum_{i=1}^{N_s} w_t^{(i)} f^2(\mathbf{X}_t^{(i)}) - \frac{1}{N_s}\sum_{i=1}^{N_s}\left[N_s \int_{(i-1)/Ns}^{i/N_s} f\circ\tilde{\mathbf{X}}_t\circ F^{-1}(u)du\right]^2.\end{aligned}$$

Using Jensen's inequality,

$$\begin{aligned}\frac{1}{N_s}\sum_{i=1}^{N_s}\left[N_s \int_{(i-1)/Ns}^{i/N_s} f\circ\tilde{\mathbf{X}}_t\circ F^{-1}(u)du\right]^2 &\geq \sum_{i=1}^{N_s}\left[\int_{(i-1)/Ns}^{i/N_s} f\circ\tilde{\mathbf{X}}_t\circ F^{-1}(u)du\right] \\ &= \left[\sum_{i=1}^{N_s} w_t^{(i)} f(\mathbf{X}_t^{(i)})\right]^2. \qquad (3.43)\end{aligned}$$

From Equation (3.43) we can see that the variance of stratified resampling is always smaller than that of multinomial resampling, Equation (3.38).

Systematic Resampling

The systematic resampling procedure extends the stratified resampling procedure. In the systematic resampling scheme, the random variables drawn in each strata are deterministically linked [66]. This is achieved by setting,

$$I^i = \frac{(i-1)}{N_s} + U, \tag{3.44}$$

where,

U is a single Uniform $((0, 1/N_s])$.

An outline of the Systematic resampling algorithm is shown in Table (3.11).

Table 3.11: Systematic Resampling Algorithm

1.	Draw $\left\{I^i\right\}_{1\leq i\leq N_s}$ random numbers according to $I^i = \frac{(i-1)+U}{N_s}$, where $U \sim \bar{U}niform((0, 1])$
3.	Select $\tilde{\mathbf{X}}^{(i)} = \mathbf{X}^{(i)}\left(F^{-1}(I^i)\right)$

The new set of samples produced by the systematic resampling scheme are not independent [68]. This makes studying the theoretical variance of the scheme difficult. Empirical studies of the systematic resampling scheme have shown that systematic resampling has the lowest variance of all previously mentioned resampling schemes [66].

3.6 Case Study: Kalman Filter

In this case study we investigate the efficacy of the Kalman filter at estimating the state of two Data Generating Systems (DGS)'s. Both DGSs consist of a latent linear system model and a cointegrated Vector Error Correction Model (VECM) with a deterministic term as the measurement model. The case study utilises the novel implementation of the Kalman filter algorithm in Object-oriented MATLAB is given in Appendix (C).

3.6.1 Model

The Data Generating Systems (DGS)'s used in the investigation are shown in Table (3.12).

where,

Table 3.12: Case Study Data Generating System (DGS) Equations

Data Generating System	System Equation	Measurement Equation
1	$M_t = A + \eta_t$	$Y_t = M_t + \alpha\beta' X_{t-1} + \Gamma(X_{t-1} - X_{t-2}) + \epsilon_t$
2	$M_t = AM_{t-1} + C + \eta_t$	$Y_t = M_t + \alpha\beta' X_{t-1} + \Gamma(X_{t-1} - X_{t-2}) + \epsilon_t$

n_x	is the dimension of the state vector,
$M_t \in \mathbb{R}^{n_x}$	is the state vector,
$Y_t \in \mathbb{R}^{n_x}$	is the measurement vector,
$A : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the system matrix,
$C : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the system matrix,
$\eta_t \in \mathbb{R}^{n_x}$	is the white, matrix variate Gaussian, zero mean process noise vector with covariance Σ,
$\alpha : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the parameter matrix,
$\beta : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the cointegration vector matrix,
$\Gamma : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the lag matrix,
$\epsilon_t \in \mathbb{R}^{n_x}$	is the white, matrix variate Gaussian, zero mean process noise vector with covariance Ω.

From Table (3.12), it is evident that the system equation of DGS 1, is a matrix variate Gaussian process with mean A and covariance Σ, whereas the system equation of DGS 2 is a Vector Autoregressive (VAR) process of order 1. The measurement equations of both DGS 1 and 2 is given by Vector Error Correction Model (VECM) of order 2 with a deterministic term whose latent process is given by the system equations.

The Kalman filter can be used to recursively estimate the state of the system model given a set of noisy measurements. However, before the filter can be used, the DGS must have Gaussian noise characteristics and be expressed in the state space form covered in Section (3.2.1). It is clear from Table (3.12), that whilst both DGSs have Gaussian noise characteristics, the system equation of DGS 2 and the measurement equations of both DGSs are not in state space form. The system equation of DGS 1 can be transformed as follows,

$$\begin{aligned} M_t &= AM_{t-1} + C + \eta_t \\ M_t^{aug} &= AM_{t-1} + \eta_t, \end{aligned}$$

where,

$$M_t^{aug} = M_t - C.$$

The measurement equations of both Data Generating Systems can be augmented into the required state space form by noting that $Y_t = \Delta X_t = X_t - X_{t-1}$ and proceeding as follows,

$$X_t - X_{t-1} = M_t + \alpha\beta' X_{t-1} + \Gamma(X_{t-1} - X_{t-2}) + \epsilon_t$$
$$Y_t^{aug} = M_t + \epsilon_t,$$

where,

$$Y_t^{aug} = X_t - (\alpha\beta' + I + \Gamma)X_{t-1} + \Gamma X_{t-2}.$$

We can now write the equations of the DGS in the state space form shown in Table (3.13), noting that $\mathbb{I}_{n_x}$ is an identity matrix of dimension n_x.

Table 3.13: Case Study Augmented Data Generating System (DGS) Equations

Augmented DGS	System Equation	Measurement Equation
1	$M_t = A + \eta_t$	$Y_t^{aug} = \mathbb{I}_{n_x} M_t + \epsilon_t$
2	$M_t^{aug} = AM_{t-1} + \eta_t$	$Y_t^{aug} = \mathbb{I}_{n_x} M_t + \epsilon_t$

For each augmented Data Generating System, a total of 300 data sets were generated. 50 data sets containing 50, 100, 200, 400, 800 and 1000 observations respectively. The parameters that were used with each augmented DGS are presented in Table (3.14).

Table 3.14: Case Study Augmented Data Generating System (DGS) Parameter Values

Parameter	Value
n_x	2
A	$\begin{pmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{pmatrix}$
C	$\begin{pmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{pmatrix}$
Σ	$\begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}$
α	$\begin{pmatrix} 0.2 & 0.2 \\ 0.2 & 0.2 \end{pmatrix}$
β	$\begin{pmatrix} 1 & 0 \\ -1 & -1 \end{pmatrix}$
Γ	$\begin{pmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{pmatrix}$
Ω	$\begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}$

We note that the α and β matrices of the augmented Data Generating System, given in Table (3.14), introduce a cointegrating relationship into the measurement equations of both DGSs, since, $\text{rank}(\alpha\beta') = 1$ which is less than n_x.

3.6.2 Method

The efficacy of the Kalman filter at estimating the state of the augmented Data Generating System (DGS) is evaluated using a spectrum of Signal to Noise Ratios (SNR)'s, provided in Table (3.16).

The Kalman filter for the system equation of each DGS had the following form,

$$\mathbf{X}_t = F\mathbf{X}_{t-1} + \mathbf{W}_{t-1}, \qquad \mathbf{W}_{t-1} \sim MVN(0, Q_t) \tag{3.45}$$

$$\mathbf{Y}_t = H\mathbf{X}_t + \mathbf{V}_t, \qquad \mathbf{V}_t \sim MVN(0, R_t) \tag{3.46}$$

where,

$\mathbf{X}_t \in \mathbb{R}^{n_x}$	is the state vector,
$\mathbf{Y}_t \in \mathbb{R}^{n_x}$	is the measurement vector,
F_t: $\mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the system matrix,
$\mathbf{W}_{t-1} \in \mathbb{R}^{n_x}$	is the white, Gaussian, zero mean process noise vector with covariance Q_t,
H_t: $\mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the measurement matrix,
$\mathbf{V}_t \in \mathbb{R}^{n_x}$	is the white, Gaussian, zero mean measurement noise vector with covariance R_t.

The parameter values used in the aforementioned Kalman filter are shown in Table (3.24).

Table 3.15: Kalman Filter Parameter Values

Parameter	Value
F	$\begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}$
Q_t	$\begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}$
H	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
R_t	$\begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}$

The spectrum of Signal to Noise Ratio (SNR) settings used for each simulation is shown in Table (3.16). The SNR was varied by increasing or decreasing the covariance of the measurement noise vector, R_t, relative to the covariance of the process noise vector, Q_t.

3.6.3 Results

A summary of the results of each simulation for both Data Generating Systems are shown in the proceeding sections.

Table 3.16: Signal to Noise Ratio (SNR) settings

Simulation Number	Signal to Noise Ratio (dB)
1	10
2	0
3	-10

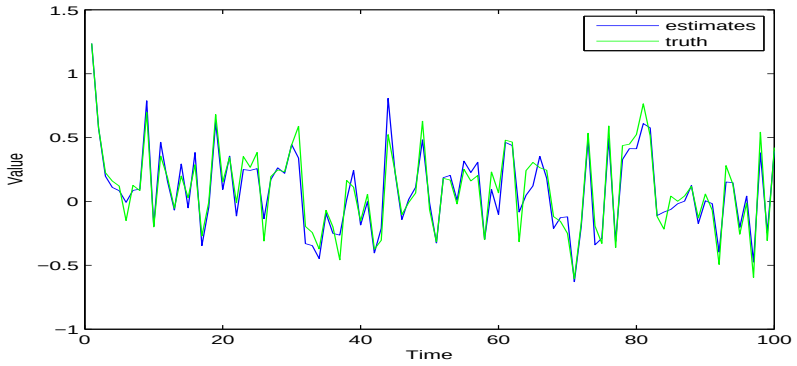
Data Generating System 1

The results of the case study for augmented Data Generating System (DGS) 1 are shown in the proceeding Figures and Tables. Figure (3.2) presents a typical set of results from the execution of the Kalman filter on DGS 1 with varying Signal to noise Ratios (SNR)'s. Table (3.17) presents a summary of the Mean Square Error (MSE) of the Kalman filter state estimates with varying SNRs. Figure (3.3) shows a typical set of innovation process, Equation (3.9), observations from the Kalman filter with varying SNRs. Table (3.17) presents a summary of the average innovation processes of the Kalman filter over all SNRs.

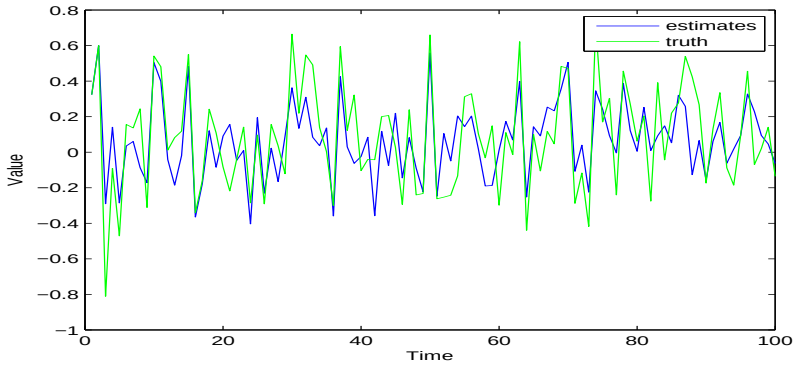
From Figure (3.2), it is clear that the *a posteriori* state estimate, $\hat{\mathbf{X}}_{1,t+1|t+1}$, approximates the true state vector, $\mathbf{X}_{1,t}$, more accurately at higher Signal to Noise Ratio (SNR) values. In the lowest SNR setting the *a posteriori* state estimate is shown to approximate the mean of the latent process. An explanation for this phenomenon is the effect that the SNR setting has on the Kalman gain, Equation (3.12), in the Kalman filter algorithm. In high SNR settings, the covariance of the measurement noise vector R_t is low relative to the covariance of the system noise vector Q_t, this results in a high Kalman gain value, which in turn means that the *a posteriori* state estimate will incorporate new information. Hence, in high SNR settings, the *a posteriori* state estimate is able to track the true state of the latent process through its peaks and troughs. This is in stark contrast to low SNR settings, where the covariance of the measurement noise vector R_t is high relative to the covariance of the system noise vector Q_t. In this situation, the Kalman gain value is low and the *a posteriori* state estimate will be less influenced by new information and will give more importance to the *a posteriori* state estimate at the previous time step.

From Table (3.17) it is clear that the MSE of the estimate of $\mathbf{X}_t$ decreases with increasing SNR. As a result, the covariance of the error of the estimate also decreases with increasing SNR. We also see an almost halving in the average MSE over all number of observations as the SNR is increased from -10dB to 0dB and an order of magnitude decrease as the SNR is increased to 10dB. A similar pattern in the Standard Error of the MSE is also observed.

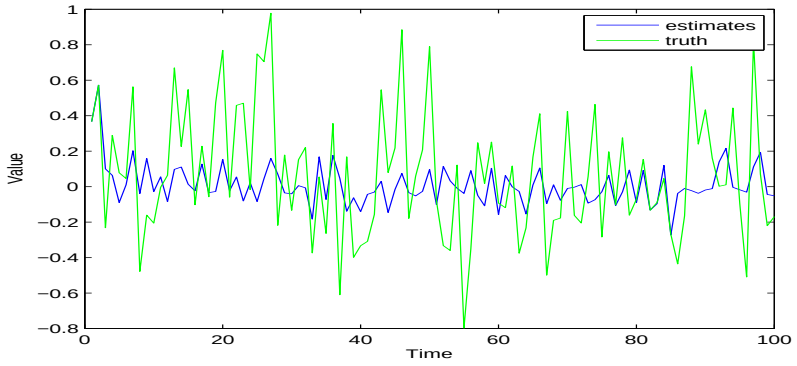
From Figure (3.3), it is clear innovation process of the Kalman filter algorithm becomes more normal with higher SNR values.



(a) Simulation 1 (SNR 10 dB)

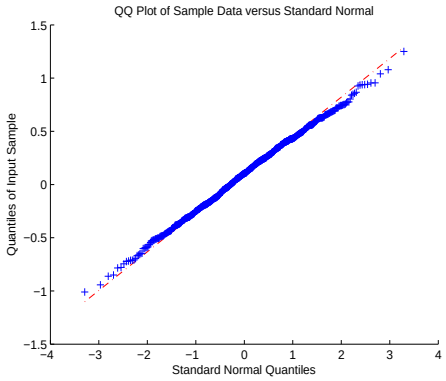


(b) Simulation 2 (SNR 0 dB)

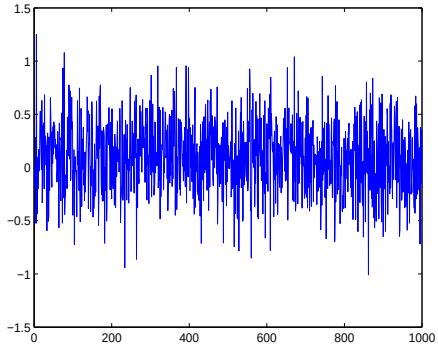


(c) Simuilation 3 (SNR -10 dB)

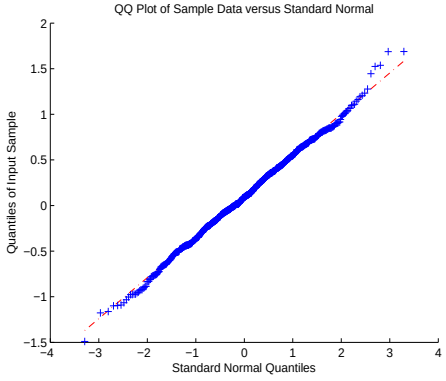
Figure 3.2: A set of typical Kalman filter results for $\mathbf{X}_{1,t}$ from Data Generating System 1. In Figures (a), (b) and (c), the blue line refers to the *a posteriori* state estimate, $\hat{\mathbf{X}}_{1,t+1|t+1}$ and green line refers to the true state vector, $\mathbf{X}_{1,t}$, at time t.



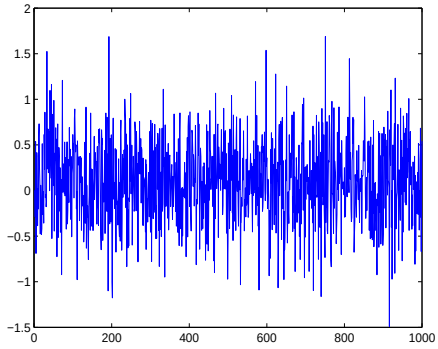
(a) Simulation 1 (SNR 10 dB)



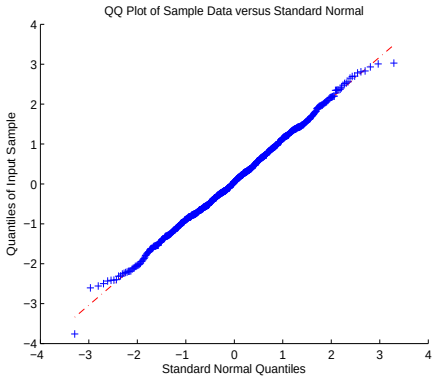
(b) Simulation 1 (SNR 10 dB)



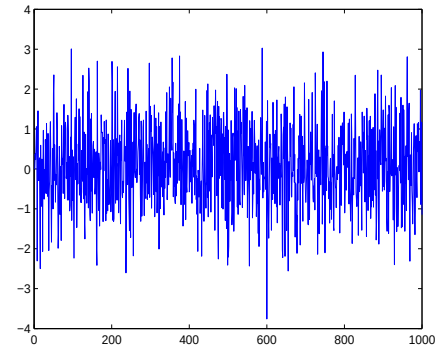
(c) Simulation 2 (SNR 0 dB)



(d) Simulation 2 (SNR 0 dB)



(e) Simulation 3 (SNR -10 dB)



(f) Simulation 3 (SNR -10 dB)

Figure 3.3: A set of typical Kalman filter innovations, $e_{1,t}$, from Data Generating System 1. The first column contains quantile-quantile plots for the $e_{1,t}$, whereas the second contains the $e_{1,t}$.

Data Generating System 2

The results of the case study for Data Generating System (DGS) 2 are shown in the proceeding Figures and Tables. Figure (3.4) presents a typical set of results from the execution of the

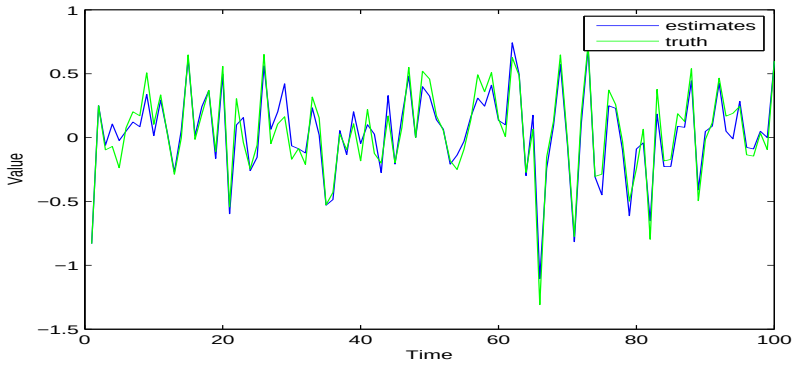
Kalman filter on DGS 2 with varying Signal to noise Ratios (SNR)'s. Table (3.19) presents a summary of the Mean Square Error (MSE) of the Kalman filter state estimates with varying SNRs. Figure (3.5) shows a typical set of innovation process observations from the Kalman filter with varying SNRs. Table (3.19) presents a summary of the average innovation processes of the Kalman filter over all SNRs.

From Figure (3.4), it is clear that the *a posteriori* state estimate, $\hat{\mathbf{X}}_{1,t+1|t+1}$, approximates the true state vector, $\mathbf{X}_{1,t}$, more accurately at higher Signal to Noise Ratio (SNR) values. In the lowest SNR setting the *a posteriori* state estimate is shown to approximate the mean of the latent process. As with DGS 1, an explanation for this phenomenon is the affect that the SNR setting has on the Kalman gain, Equation (3.12), in the Kalman filter algorithm.

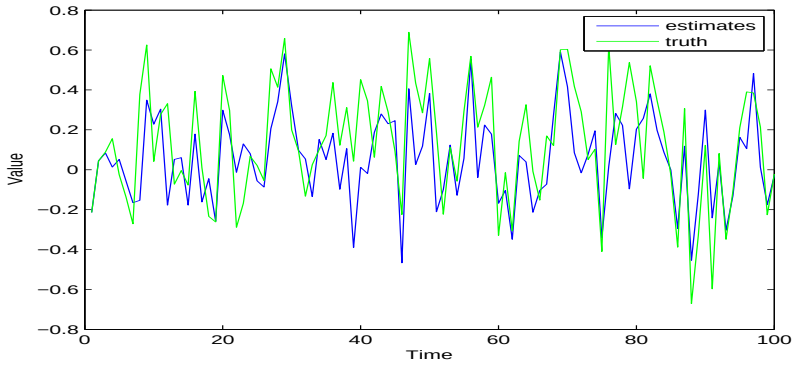
From Table (3.19) it is clear that the MSE of the estimate of $\mathbf{X}_t$ decreases with increasing SNR. As a result, the covariance of the error of the estimate also decreases with increasing SNR. We also see an almost halving in the average MSE over all number of observations as the SNR is increased from -10dB to 0dB and an order of magnitude decrease as the SNR is increased to 10dB. A similar pattern in the Standard Error of the MSE is also observed. We also note that the MSE for DGS 2 slightly higher than that of DGS 1, shown in Table (3.17). This corresponds to a decrease in the efficiency of estimation with increasing complexity of the latent linear system model.

From Figure (3.5), it is clear innovation process of the Kalman filter algorithm becomes more normal with higher SNR values.

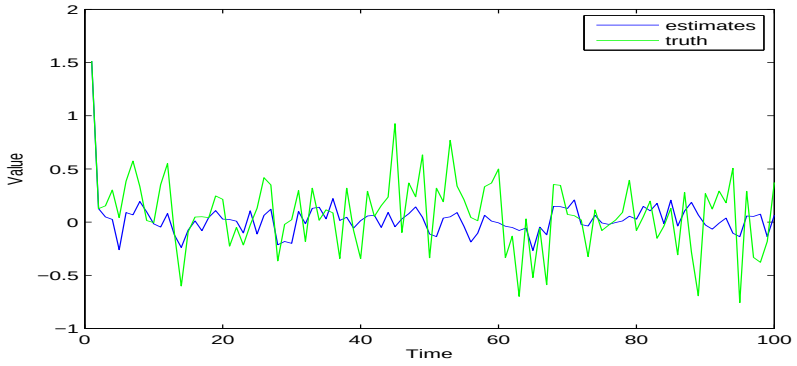
From Table (3.20), we can see that the mean innovation for all simulation from DGS 2 is slightly larger than those from DGS 1. This corresponds to a decrease in the efficiency of estimation with increasing complexity of the latent linear system model.



(a) Simulation 1 (SNR 10 dB)

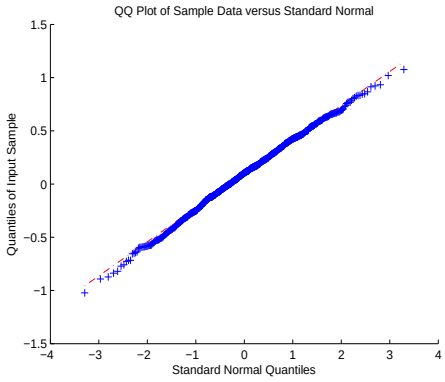


(b) Simulation 2 (SNR 0 dB)

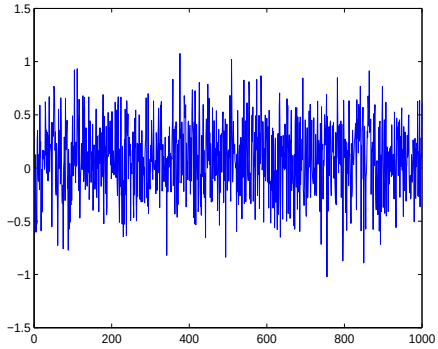


(c) Simulation 3 (SNR -10 dB)

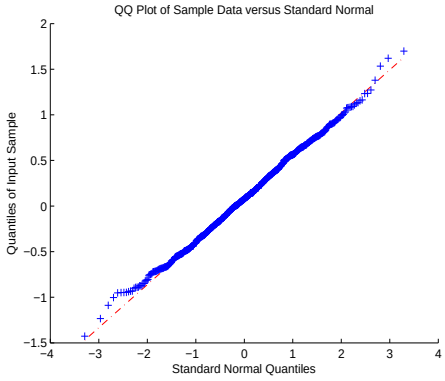
Figure 3.4: A set of typical Kalman filter results for $\mathbf{X}_{1,t}$ from Data Generating System 2. In Figures (a), (b) and (c), the blue line refers to the *a posteriori* state estimate, $\hat{\mathbf{X}}_{1,t+1|t+1}$ and the green line refers to the true state vector, $\mathbf{X}_{1,t}$, at time t.



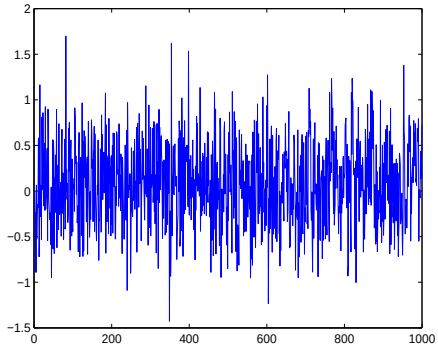
(a) Simulation 1 (SNR 10 dB)



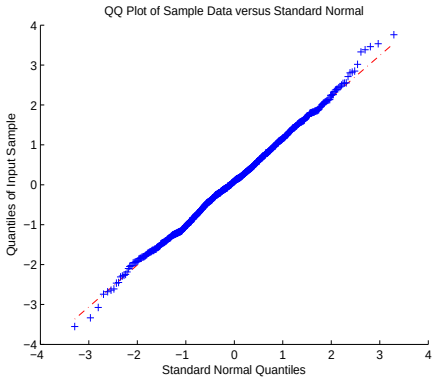
(b) Simulation 1 (SNR 10 dB)



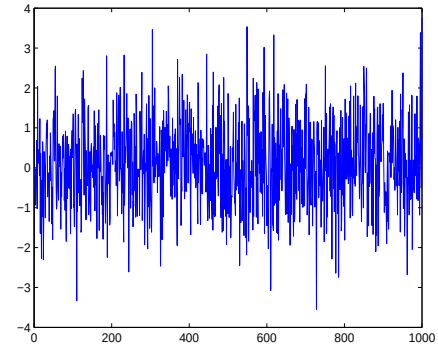
(c) Simulation 2 (SNR 0 dB)



(d) Simulation 2 (SNR 0 dB)



(e) Simulation 3 (SNR -10 dB)



(f) Simulation 3 (SNR -10 dB)

Figure 3.5: A set of typical Kalman filter innovations, $e_{1,t}$, from Data Generating System 1. The first column contains quantile-quantile plots for the $e_{1,t}$, whereas the second contains the $e_{1,t}$.

Table 3.17: Mean Squared Error (MSE) of the Kalman filter estimates from Data Generating System 1

Simulation Number (SNR)	Number of Observations	MSE $\mathbf{X}_{1,t}$ (Standard Error)	MSE $\mathbf{X}_{2,t}$ (Standard Error)
1 (10 dB)	50	0.0087 (0.0089)	0.0083 (0.0089)
	100	0.0088 (0.0090)	0.0088 (0.0090)
	200	0.0090 (0.0091)	0.0091 (0.0091)
	400	0.0091 (0.0091)	0.0091 (0.0091)
	800	0.0091 (0.0091)	0.0091 (0.0091)
	1000	0.0091 (0.0091)	0.0092 (0.0091)
2 (0 dB)	50	0.0523 (0.0491)	0.0492 (0.0491)
	100	0.0495 (0.0496)	0.0535 (0.0496)
	200	0.0526 (0.0499)	0.0519 (0.0499)
	400	0.0526 (0.0500)	0.0519 (0.0500)
	800	0.0521 (0.0501)	0.0520 (0.0501)
	1000	0.0520 (0.0501)	0.0529 (0.0501)
3 (-10 dB)	50	0.1017 (0.0898)	0.0932 (0.0898)
	100	0.0934 (0.0908)	0.0966 (0.0908)
	200	0.0980 (0.0912)	0.0958 (0.0912)
	400	0.0996 (0.0914)	0.0985 (0.0914)
	800	0.0985 (0.0916)	0.0989 (0.0916)
	1000	0.0993 (0.0916)	0.0987 (0.0916)

Table 3.18: Mean of the innovation process from Data Generating System 1

Simulation Number (SNR)	Number of Observations	$e_{1,t}$	$e_{2,t}$
1 (10 dB)	50	0.0995	0.0876
	100	0.0843	0.0844
	200	0.0926	0.0898
	400	0.0914	0.0914
	800	0.0899	0.0912
	1000	0.0884	0.0909
2 (0 dB)	50	0.0976	0.0892
	100	0.0940	0.1003
	200	0.0957	0.0993
	400	0.0944	0.0935
	800	0.0916	0.0971
	1000	0.0946	0.0961
3 (-10 dB)	50	0.0790	0.1419
	100	0.0890	0.0817
	200	0.0950	0.1009
	400	0.0801	0.1114
	800	0.1083	0.0982
	1000	0.0988	0.0928

Table 3.19: Mean Squared Error (MSE) of the Kalman filter estimates from Data Generating System 2

Simulation Number (SNR)	Number of Observations	MSE $\mathbf{X}_{1,t}$ (Standard Error)	MSE $\mathbf{X}_{2,t}$ (Standard Error)
1 (10 dB)	50	0.0090 (0.0089)	0.0089 (0.0089)
	100	0.0090 (0.0090)	0.0091 (0.0090)
	200	0.0091 (0.0091)	0.0089 (0.0091)
	400	0.0091 (0.0091)	0.0090 (0.0091)
	800	0.0092 (0.0091)	0.0092 (0.0091)
	1000	0.0092 (0.0091)	0.0092 (0.0091)
2 (0 dB)	50	0.0507 (0.0491)	0.0513 (0.0491)
	100	0.0526 (0.0496)	0.0509 (0.0496)
	200	0.0526 (0.0499)	0.0529 (0.0499)
	400	0.0531 (0.0500)	0.0525 (0.0500)
	800	0.0526 (0.0501)	0.0528 (0.0501)
	1000	0.0530 (0.0501)	0.0526 (0.0501)
3 (-10 dB)	50	0.0963 (0.0898)	0.0932 (0.0898)
	100	0.0993 (0.0908)	0.1030 (0.0908)
	200	0.1010 (0.0912)	0.1017 (0.0912)
	400	0.1003 (0.0914)	0.1025 (0.0914)
	800	0.1019 (0.0916)	0.1026 (0.0916)
	1000	0.1012 (0.0916)	0.1019 (0.0916)

Table 3.20: Mean of the innovation process from Data Generating System 2

Simulation Number (SNR)	Number of Observations	$X_{1,t}$	$X_{2,t}$
1 (10 dB)	50	0.0973	0.0997
	100	0.1006	0.1060
	200	0.0981	0.0995
	400	0.1012	0.0993
	800	0.1027	0.1013
	1000	0.1011	0.1027
2 (0 dB)	50	0.1157	0.0984
	100	0.0979	0.1012
	200	0.1104	0.0951
	400	0.1081	0.1070
	800	0.1034	0.1052
	1000	0.1036	0.1027
3 (-10 dB)	50	0.1302	0.0757
	100	0.1187	0.1021
	200	0.0892	0.1147
	400	0.1041	0.1158
	800	0.1183	0.1072
	1000	0.1175	0.1061

3.7 Case Study: Extended Kalman Filter

In this case study we investigate the efficacy of the Extended Kalman filter at estimating the state of two Data Generating Systems (DGS)'s. Each DGS consists of a latent linear and non-linear system model and a cointegrated Vector Error Correction Model (VECM) with a deterministic term as the measurement model.

3.7.1 Model

The equations of the Data Generating Systems (DGSs) used in this case study are presented in Table (3.21).

Table 3.21: Case Study Data Generating System (DGS) Equations

Data Generating System	System Equation	Measurement Equation
1	$M_t = AM_{t-1} + C + \eta_t$	$Y_t = M_t + \alpha\beta' X_{t-1} + \Gamma(X_{t-1} - X_{t-2}) + \epsilon_t$
2	$M_t = AM_{t-1}^2 + BM_{t-1} + C + \eta_t$	$Y_t = M_t + \alpha\beta' X_{t-1} + \Gamma(X_{t-1} - X_{t-2}) + \epsilon_t$

where

n_x	is the dimension
$M_t \in \mathbb{R}^{n_x}$	is the state vector
$Y_t \in \mathbb{R}^{n_x}$	is the measurement vector
$A : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the system matrix
$B : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the system matrix
$C : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the system matrix
$\eta_t \in \mathbb{R}^{n_x}$	is the white, matrix variate Gaussian, zero mean process noise vector with covariance Σ
$\alpha : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the parameter matrix
$\beta : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the cointegration vector matrix
$\Gamma : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the lag matrix
$\epsilon_t \in \mathbb{R}^{n_x}$	is the white, matrix variate Gaussian, zero mean process noise vector with covariance Ω

From Table (3.21), it is evident that the system equation of DGS 1, is a Vector Autoregressive (VAR) process of order 1, whereas the system equation of DGS 2 is a non-linear VAR process of order 1, with the non-linearity introduced via a quadratic term. The measurement equations of both DGS 1 and 2 is given by Vector Error Correction Model (VECM) of order 2 with a deterministic term whose latent process is given by the system equations.

The Extended Kalman filter can be used to recursively estimate the states of the system given a set of noisy measurements. However, before the filter can be applied, the DGS must have Gaussian noise characteristics and be expressed in the state space form covered in Section (3.2.1).

It is clear from Table (3.21), that whilst both DGSs have Gaussian noise characteristics, they the system and measurement equations of both DGSs are not in state space form. The system equation of DGS 1 can be transformed as follows,

$$\begin{aligned} M_t &= AM_{t-1} + C + \eta_t \\ M_t^{aug} &= AM_{t-1} + \eta_t, \end{aligned}$$

where,

$$M_t^{aug} = M_t - C.$$

The system equation of DGS 2 can be transformed as follows,

$$\begin{aligned} M_t &= AM_{t-1}^2 + BM_{t-1} + C + \eta_t \\ M_t^{aug} &= BM_{t-1} + \eta_t, \end{aligned}$$

where,

$$M_t^{aug} = M_t - AM_{t-1}^2 - C.$$

The measurement equations of both Data Generating Systems can be augmented into the required state space form by noting that $Y_t = \Delta X_t = X_t - X_{t-1}$ and proceeding as follows,

$$\begin{aligned} X_t - X_{t-1} &= M_t + \alpha\beta' X_{t-1} + \Gamma(X_{t-1} - X_{t-2}) + \epsilon_t \\ Y_t^{aug} &= M_t + \epsilon_t, \end{aligned}$$

where,

$$Y_t^{aug} = X_t - (\alpha\beta' + I + \Gamma)X_{t-1} + \Gamma X_{t-2} \ .$$

We can now write the equations of the DGS in the following state space form, noting that $\mathbb{I}_{n_x}$ is an identity matrix of dimension n_x.

Table 3.22: Case Study Augmented Data Generating System (DGS) Equations

Augmented DGS	System Equation	Measurement Equation
1	$M_t^{aug} = AM_{t-1} + \eta_t$	$Y_t^{aug} = \mathbb{I}_{n_x} M_t + \epsilon_t$
2	$M_t^{aug} = BM_{t-1} + \eta_t$	$Y_t^{aug} = \mathbb{I}_{n_x} M_t + \epsilon_t$

For each augmented Data Generating System, a total of 300 data sets were generated. 50 data sets containing 50, 100, 200, 400, 800 and 1000 observations respectively. The parameters that were used with each DGS are presented in Table (3.23).

Table 3.23: Case Study Augmented Data Generating System (DGS) Parameter Values

Parameter	Value
n_x	2
A	$\begin{pmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{pmatrix}$
B	$\begin{pmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{pmatrix}$
C	$\begin{pmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{pmatrix}$
Σ	$\begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}$
α	$\begin{pmatrix} 0.2 & 0.2 \\ 0.2 & 0.2 \end{pmatrix}$
β	$\begin{pmatrix} 1 & 0 \\ -1 & -1 \end{pmatrix}$
Γ	$\begin{pmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{pmatrix}$
Ω	$\begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}$

We note that the α and β matrices of the augmented Data Generating System, given in Table (3.23), introduce a cointegrating relationship into the measurement equations of both DGSs, since, $\text{rank}(\alpha\beta') = 1$ which is less than n_x.

3.7.2 Method

The efficacy of the Extended Kalman filter at estimating the state of the Data Generating System (DGS) using a spectrum of Signal to Noise Ratios (SNR)'s, provided in Table (3.25).

The Extended Kalman filter for the system equation of each DGS had the following form,

$$X_t = f_t(X_{t-1}) + W_{t-1}, \qquad W_{t-1} \sim MVN(0, Q_t) \tag{3.47}$$

$$Y_t = h_t(X_t) + V_t, \qquad V_t \sim MVN(0, R_t) \tag{3.48}$$

where,

$X_t \in \mathbb{R}^{n_x}$	is the state vector,
$Y_t \in \mathbb{R}^{n_x}$	is the measurement vector,
f_t: $\mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is a non-linear function,
$W_t \in \mathbb{R}^{n_x}$	is the white, Gaussian, zero mean process noise vector with covariance Q_t,
h_t: $\mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is a non-linear function,
$V_t \in \mathbb{R}^{n_x}$	is the white, Gaussian, zero mean measurement noise vector with covariance R_t.

At this point we note the similarity between the augmented DGS equations and the Extended Kalman filter equations, namely $M_t^{aug} \equiv X_t$, A and$B \approx f_t$, $Y_t^{aug} \equiv Y_t$ and $\mathbb{I}_{n_x} \approx h_t$. The non-linear functions of the EKF were approximated using the following Jacobian matrices,

$$
\begin{aligned}
f_t &= \begin{bmatrix} \mu_{1,t} \\ \mu_{2,t} \end{bmatrix} &&= \left(\begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix} \begin{bmatrix} \mu_{1,t-1} \\ \mu_{2,t-1} \end{bmatrix} + \begin{bmatrix} w_{1,t-1} \\ w_{2,t-1} \end{bmatrix} \right) \\
\hat{F} &= \frac{\partial f_t(\mu)}{\partial X}|_{\mu = f_t(\hat{\mu}_{t|t})} \\
J_{f_t}(\mu_1, \mu_2) &= \begin{bmatrix} \frac{\partial \mu_{1,t}}{\partial \mu_1} & \frac{\partial \mu_{2,t}}{\partial \mu_2} \\ \frac{\partial \mu_{1,t}}{\partial \mu_1} & \frac{\partial \mu_{2,t}}{\partial \mu_2} \end{bmatrix} \\
\hat{F} &= \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix}
\end{aligned}
$$

$$
\begin{aligned}
h &= \begin{bmatrix} y_{1,t} \\ y_{2,t} \end{bmatrix} &&= \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_{1,t} \\ \mu_{2,t} \end{bmatrix} + \begin{bmatrix} v_{1,t-1} \\ v_{2,t-1} \end{bmatrix} \right) \\
\hat{H} &= \frac{\partial h_t(\mu)}{\partial X}|_{\mu = f_t(\hat{\mu}_{t|t-1})} \\
J_{h_t}(\mu_1, \mu_2) &= \begin{bmatrix} \frac{\partial y_{1,t}}{\partial \mu_1} & \frac{\partial y_{2,t}}{\partial \mu_2} \\ \frac{\partial y_{1,t}}{\partial \mu_1} & \frac{\partial y_{2,t}}{\partial \mu_2} \end{bmatrix} \\
\hat{H} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}
\end{aligned}
$$

The parameter values used in the aforementioned Kalman filter are shown in Table (3.24).

The spectrum of Signal to Noise Ratio (SNR) settings used for each simulation is shown in table 3.25. The SNR was varied by increasing or decreasing the covariance of the measurement noise vector, R_t, relative to the covariance of the process noise vector, Q_t.

Table 3.24: Extended Kalman Filter Parameter Values

Parameter	Value
$\hat{F}$	$\begin{pmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{pmatrix}$
Q_t	$\begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}$
$\hat{H}$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
R_t	$\begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}$

Table 3.25: Signal to Noise Ratio (SNR) settings

Simulation Number	Signal to Noise Ratio (dB)
1	10
2	0
3	-10

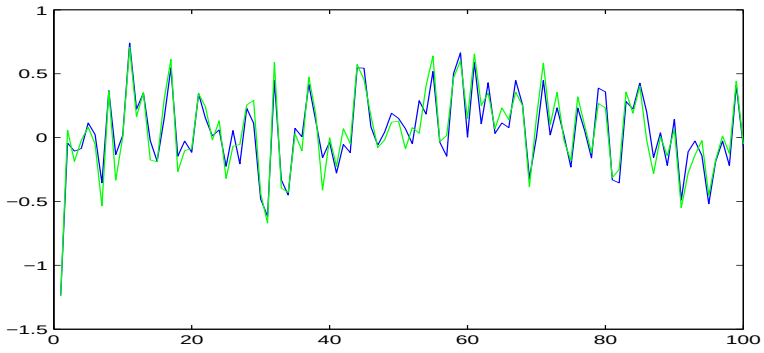
3.7.3 Results

A summary of the results of each simulation for both Data Generating Systems are shown in the proceeding sections.

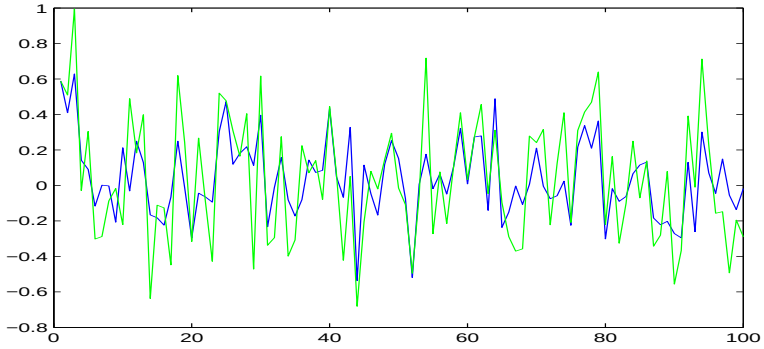
Data Generating System 1

The results of the case study for Data Generating System (DGS) 1 are shown in the proceeding Figures and Tables. Figure (3.6) presents a typical set of results from the execution of the Extended Kalman filter on DGS 1 with varying Signal to noise Ratios (SNR)'s. Table (3.26) presents a summary of the Mean Square Error (MSE) of the Extended Kalman filter state estimates with varying SNRs. Figure (3.7) shows a typical set of innovation process observations from the Extended Kalman filter with varying SNRs. Table (3.26) presents a summary of the average innovation processes of the Extended Kalman filter over all SNRs.

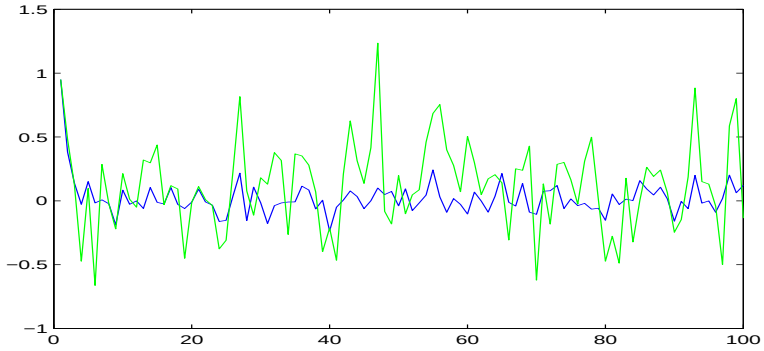
From Figure (3.6), it is clear that the *a posteriori* state estimate, $\hat{\mathbf{X}}_{1,t+1|t+1}$, approximates the true state vector, $X_{1,t}$, more accurately at higher Signal to Noise Ratio (SNR) values. In the lowest SNR setting the *a posteriori* state estimate is shown to approximate the mean of the latent process. An explanation for this phenomenon is the effect that the SNR setting has on the Kalman gain, Equation (3.12), in the Kalman filter algorithm. In high SNR settings, the covariance of the measurement noise vector R_t is low relative to the covariance of the system noise vector Q_t, this results in a high Kalman gain value, which in turn means that the *a posteriori* state estimate will incorporate new information. Hence, in high SNR settings, the *a posteriori* state estimate is able to track the true state of the latent process through its peaks and troughs.



(a) Simulation 1 (SNR 10 dB)



(b) Simulation 2 (SNR 0 dB)



(c) Simulation 3 (SNR -10 dB)

Figure 3.6: A set of typical Extended Kalman filter results for $\mathbf{X}_{1,t}$ from Data Generating System 1. In Figures (a), (b) and (c), the blue line refers to the *a posteriori* state estimate, $\hat{\mathbf{X}}_{1,t+1|t+1}$ and the green line refers to the true state vector, $\mathbf{X}_{1,t}$, at time t.

This is in stark contrast to low SNR settings, where the covariance of the measurement noise vector R_t is high relative to the covariance of the system noise vector Q_t. In this situation, the Kalman gain value is low and the *a posteriori* state estimate will be less influenced by new information and will give more importance to the *a posteriori* state estimate at the previous time step.

Table 3.26: Mean Squared Error (MSE) of the Extended Kalman filter estimates from Data Generating System 1

Simulation Number (SNR)	Number of Observations	MSE $\mathbf{X}_{1,t}$ (Standard Error)	MSE $\mathbf{X}_{2,t}$ (Standard Error)
1 (10 dB)	50	0.0094 (0.0089)	0.0090 (0.0089)
	100	0.0093 (0.0090)	0.0091 (0.0090)
	200	0.0093 (0.0091)	0.0091 (0.0091)
	400	0.0092 (0.0091)	0.0090 (0.0091)
	800	0.0091 (0.0091)	0.0092 (0.0091)
	1000	0.0091 (0.0091)	0.0091 (0.0091)
2 (0 dB)	50	0.0512 (0.0491)	0.0521 (0.0491)
	100	0.0508 (0.0496)	0.0519 (0.0496)
	200	0.0525 (0.0499)	0.0517 (0.0499)
	400	0.0533 (0.0500)	0.0534 (0.0500)
	800	0.0529 (0.0501)	0.0531 (0.0501)
	1000	0.0532 (0.0501)	0.0531 (0.0501)
3 (-10 dB)	50	0.0951 (0.0898)	0.0978 (0.0898)
	100	0.1025 (0.0908)	0.0983 (0.0908)
	200	0.0988 (0.0912)	0.1007 (0.0912)
	400	0.1030 (0.0914)	0.1029 (0.0914)
	800	0.1028 (0.0916)	0.1014 (0.0916)
	1000	0.1007 (0.0916)	0.1010 (0.0916)

From Table (3.26) it is clear that the MSE of the estimate of X_t decreases with increasing SNR. As a result, the covariance of the error of the estimate also decreases with increasing SNR. We also see an almost halving in the average MSE over all number of observations as the SNR is increased from -10dB to 0dB and an order of magnitude decrease as the SNR is increased to 10dB. A similar pattern in the Standard Error of the MSE is also observed.

From Figure (3.7), it is clear innovation process of the Extended Kalman filter algorithm becomes more normal with higher SNR values.

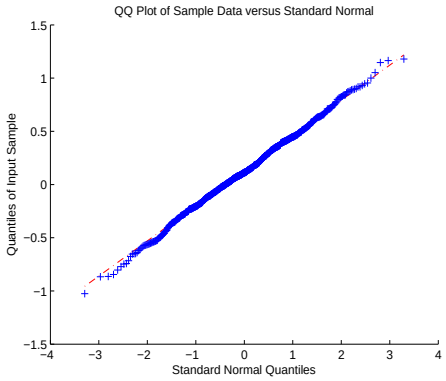
Table 3.27: Mean of the innovation process from Data Generating System 1

Simulation Number	Number of Observations	$\mathbf{X}_{1,t}$	$\mathbf{X}_{2,t}$
1 (10 dB)	50	0.1004	0.1092
	100	0.1059	0.1121
	200	0.1106	0.1121
	400	0.1106	0.1078
	800	0.1101	0.1088
	1000	0.1101	0.1125
2 (0 dB)	50	0.1094	0.1024
	100	0.1118	0.1079
	300	0.1100	0.1076
	400	0.1106	0.1107
	800	0.1096	0.1090
	1000	0.1088	0.1098
3 (-10 dB)	50	0.1100	0.1015
	100	0.1066	0.1062
	200	0.1145	0.1102
	400	0.1072	0.1048
	800	0.1049	0.1100
	1000	0.1081	0.1045

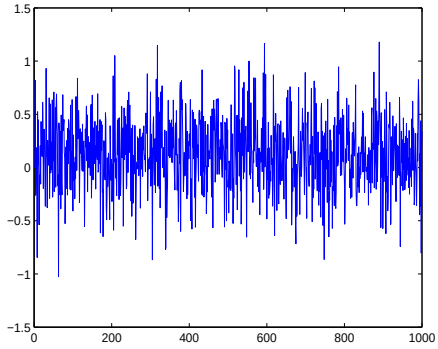
Data Generating System 2

The results of the case study for Data Generating System (DGS) 2 are shown in the proceeding Figures and Tables. Figure (3.8) presents a typical set of results from the execution of the Extended Kalman filter on DGS 1 with varying Signal to noise Ratios (SNR)'s. Table (3.28) presents a summary of the Mean Square Error (MSE) of the Extended Kalman filter state estimates with varying SNRs. Figure (3.9) shows a typical set of innovation process observations from the Extended Kalman filter with varying SNRs. Table (3.28) presents a summary of the average innovation processes of the Extended Kalman filter over all SNRs.

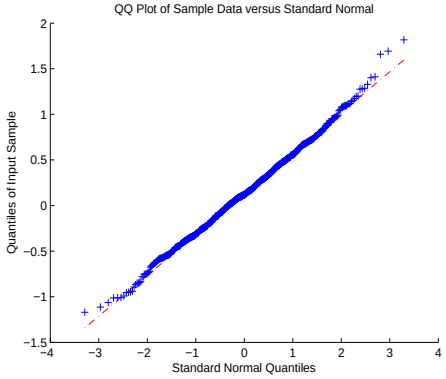
From Figure (3.8), it is clear that the *a posteriori* state estimate, $\hat{\mathbf{X}}_{1,t+1|t+1}$, approximates the true state vector, $X_{1,t}$, more accurately at higher Signal to Noise Ratio (SNR) values. In the lowest SNR setting the *a posteriori* state estimate is shown to approximate the mean of the



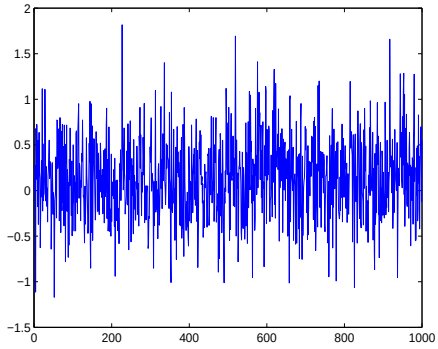
(a) Simulation 1 (SNR 10 dB)



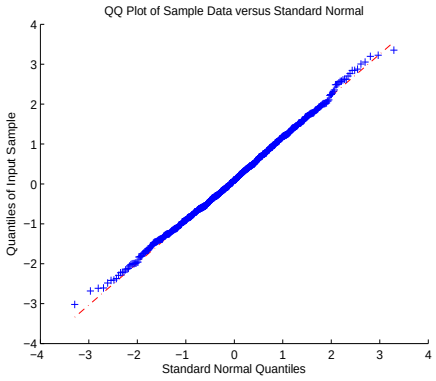
(b) Simulation 1 (SNR 10 dB)



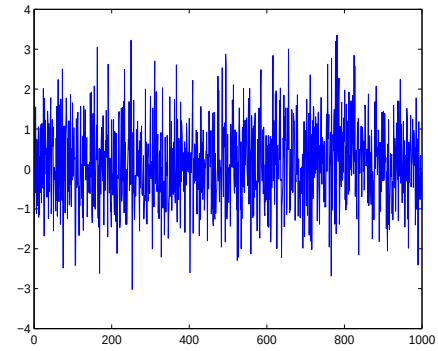
(c) Simulation 2 (SNR 0 dB)



(d) Simulation 2 (SNR 0 dB)

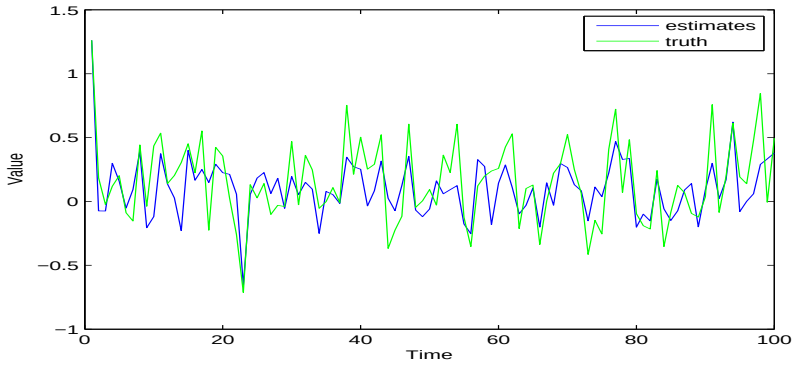


(e) Simulation 3 (SNR -10 dB)

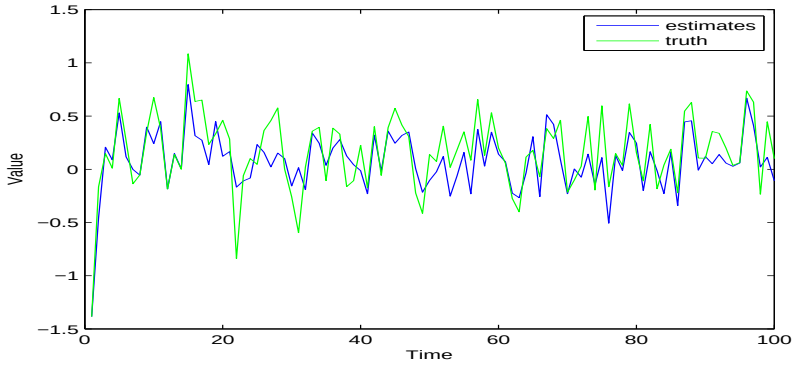


(f) Simulation 3 (SNR -10 dB)

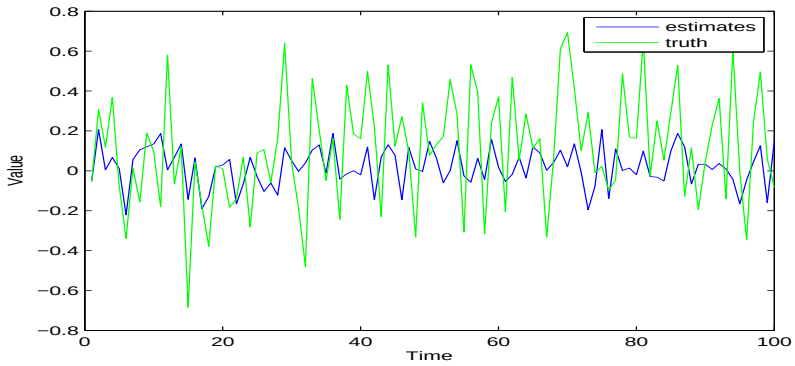
Figure 3.7: A set of typical Extended Kalman filter innovations, $e_{1,t}$, from Data Generating System 1. The first column contains quantile-quantile plots for the $e_{1,t}$, whereas the second contains the $e_{1,t}$.



(a) Simulation 3 (SNR 10 dB)



(b) Simulation 3 (SNR 0 dB)



(c) Simulation 3 (SNR -10 dB)

Figure 3.8: A set of typical Extended Kalman filter results for $\mathbf{X}_{1,t}$ from Data Generating System 2. In Figures (a), (b) and (c), the blue line refers to the *a posteriori* state estimate, $\hat{\mathbf{X}}_{1,t+1|t+1}$ and the green line refers to the true state vector, $\mathbf{X}_{1,t}$, at time t.

latent process. As with DGS 1, an explanation for this phenomenon is the affect that the SNR setting has on the Kalman gain, Equation (3.12), in the Kalman filter algorithm.

Table 3.28: Mean Squared Error (MSE) of the Extended Kalman filter estimates from Data Generating System 2

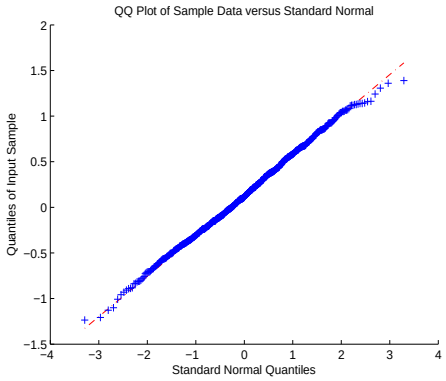
Simulation Number (SNR)	Number of Observations	MSE $\mathbf{X}_{1,t}$ (Standard Error)	MSE $\mathbf{X}_{2,t}$ (Standard Error)
1 (10 dB)	50	0.0090 (0.0491)	0.0089 (0.0491)
	100	0.0090 (0.0496)	0.0091 (0.0496)
	200	0.0091 (0.0499)	0.0089 (0.0499)
	400	0.0091 (0.0500)	0.0090 (0.0500)
	800	0.0092 (0.0501)	0.0092 (0.0501)
	1000	0.0092 (0.0501)	0.0092 (0.0501)
2 (0 dB)	50	0.0507 (0.0491)	0.0513 (0.0491)
	100	0.0526 (0.0496)	0.0509 (0.0496)
	200	0.0526 (0.0499)	0.0529 (0.0499)
	400	0.0531 (0.0500)	0.0525 (0.0500)
	800	0.0526 (0.0501)	0.0528 (0.0501)
	1000	0.0530 (0.0501)	0.0526 (0.0501)
3 (-10 dB)	50	0.0963 (0.0898)	0.0932 (0.0898)
	100	0.0993 (0.0908)	0.1030 (0.0908)
	200	0.1010 (0.0912)	0.1017 (0.0912)
	400	0.1003 (0.0914)	0.1025 (0.0914)
	800	0.1019 (0.0916)	0.1026 (0.0916)
	1000	0.1012 (0.0916)	0.1019 (0.0916)

From Table (3.28) it is clear that the MSE of the estimate of X_t decreases with increasing SNR. As a result, the covariance of the error of the estimate also decreases with increasing SNR. We also see an almost halving in the average MSE over all number of observations as the SNR is increased from -10dB to 0dB and an order of magnitude decrease as the SNR is increased to 10dB. A similar pattern in the Standard Error of the MSE is also observed. We also note that the MSE for DGS 2 slightly higher than that of DGS 1, shown in Table (3.26). This corresponds to a decrease in the efficiency of estimation with increasing complexity of the latent system equation.

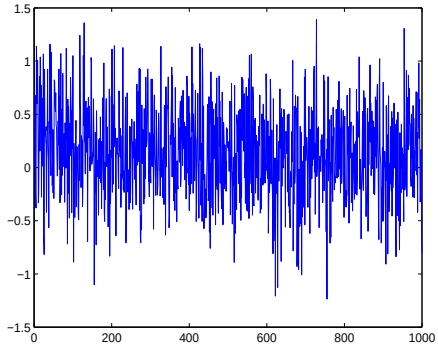
Table 3.29: Mean of the innovation process from Data Generating System 2

Simulation Number (SNR)	Number of Observations	$\mathbf{X}_{1,t}$	$\mathbf{X}_{2,t}$
1 (10 dB)	50	0.0973	0.0997
	100	0.1006	0.1060
	200	0.0981	0.0995
	400	0.1012	0.0993
	800	0.1027	0.1013
	1000	0.1011	0.1027
2 (0 dB)	50	0.1157	0.0984
	100	0.0979	0.1012
	200	0.1104	0.0951
	400	0.1081	0.1070
	800	0.1034	0.1052
	1000	0.1036	0.1027
3 (-10 dB)	50	0.1302	0.0757
	100	0.1187	0.1021
	200	0.0892	0.1147
	400	0.1041	0.1158
	800	0.1183	0.1072
	1000	0.1175	0.1061

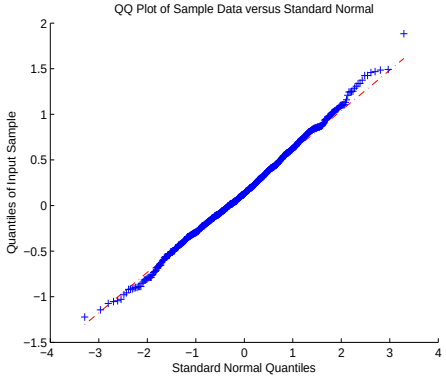
From Table (3.29), we can see that the mean innovation for all simulation from DGS 2 is slightly larger than those from DGS 1. This corresponds to a decrease in the efficiency of estimation with increasing complexity of the latent linear system model.



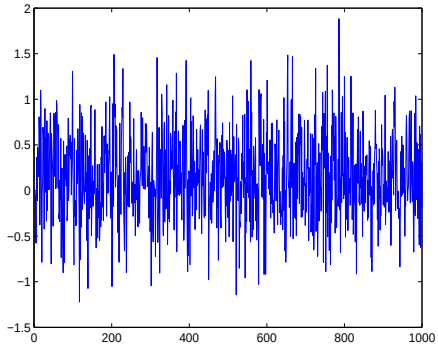
(a) Simulation 1 (SNR 10 dB)



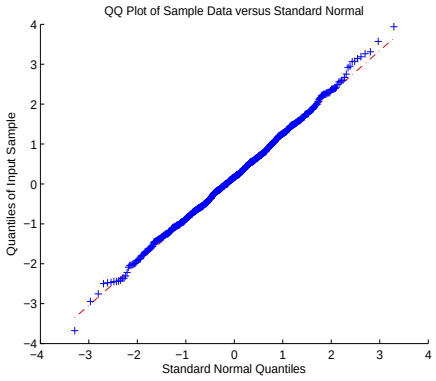
(b) Simulation 1 (SNR 10 dB)



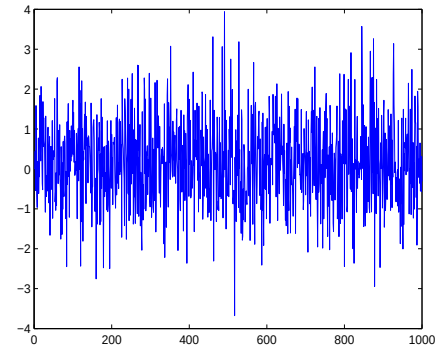
(c) Simulation 2 (SNR 0 dB)



(d) Simulation 2 (SNR 0 dB)



(e) Simulation 3 (SNR -10 dB)



(f) Simulation 3 (SNR -10 dB)

Figure 3.9: A set of typical Extended Kalman filter innovations, $e_{1,t}$, from Data Generating System 2. The first column contains quantile-quantile plots for the $e_{1,t}$, whereas the second contains the
$e_{1,t}$.

3.8 Case Study: Sequential Importance Resampling Filter

In this case study we investigate the efficacy of the Sequential Importance Resampling (SISR) filter at estimating the state of two Data Generating Systems (DGS)'s. Both DGSs consist of a latent non-linear system model and a cointegrated Vector Error Correction Model (VECM) with a deterministic term as the measurement model. The case study utilises the novel implementation of the SISR filter algorithm in Object-oriented MATLAB is given in Appendix (D).

3.8.1 Model

The Data Generating Systems (DGS)'s used in the investigation are shown in Table (3.30).

Table 3.30: Case Study Data Generating System (DGS) Equations

Data Generating System	System Equation	Measurement Equation
1	$M_t = AM_{t-1} + D + \eta_t$	$Y_t = M_t + \alpha\beta' X_{t-1} + \Gamma(X_{t-1} - X_{t-2}) + \epsilon_t$
2	$M_t = AM_{t-1}^3 + BM_{t-1}^2 + CM_{t-1} + D + \eta_t$	$Y_t = M_t + \alpha\beta' X_{t-1} + \Gamma(X_{t-1} - X_{t-2}) + \epsilon_t$

where,

n_x	is the dimension,
$M_t \in \mathbb{R}^{n_x}$	is the state vector,
$Y_t \in \mathbb{R}^{n_x}$	is the measurement vector,
$A : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the system matrix,
$B : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the system matrix,
$C : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the system matrix,
$D : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the system matrix,
$\eta_t \in \mathbb{R}^{n_x}$	is the white, matrix variate Gaussian, zero mean process noise vector with covariance Σ,
$\alpha : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the parameter matrix,
$\beta : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the cointegration vector matrix,
$\Gamma : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is the lag matrix,
$\epsilon_t \in \mathbb{R}^{n_x}$	is the white, matrix variate Gaussian, zero mean process noise vector with covariance Ω.

From Table (3.30), it is evident that the system equation of DGS 1 is a Vector Autoregressive (VAR) process of order 1, whereas the system equation of DGS 2 is a non-linear VAR process of order1, with the non-linearity introduced via cubic and quadratic terms. The measurement equations of both DGS 1 and 2 is given by Vector Error Correction Model (VECM) of order 2 with a deterministic term whose latent process is given by the system equations.

The SISR filter can be used to recursively estimate the state of the system equation given a set of noisy measurements. However, before the filter can be applied, the system and measurement

equations of both DGSs must be expressed in state space form. The system equation of DGS 1 can be transformed as follows,

$$\begin{aligned} M_t &= AM_{t-1} + C + \eta_t \\ M_t^{aug} &= AM_{t-1} + \eta_t, \end{aligned}$$

where,

$M_t^{aug} \quad = M_t - C.$

The system equation of DGS 2 can be transformed as follows,

$$\begin{aligned} M_t &= AM_{t-1}^3 + BM_{t-1}^2 + CM_{t-1} + D + \eta_t \\ M_t^{aug} &= CM_{t-1} + \eta_t, \end{aligned}$$

where,

$M_t^{aug} \quad = M_t - AM_{t-1}^3 - BM_{t-1}^2 - D.$

The measurement equations of both Data Generating Systems can be augmented into the required state space form by noting that $Y_t = \Delta X_t = X_t - X_{t-1}$ and proceeding as follows,

$$\begin{aligned} X_t - X_{t-1} &= M_t + \alpha\beta' X_{t-1} + \Gamma(X_{t-1} - X_{t-2}) + \epsilon_t \\ Y_t^{aug} &= M_t + \epsilon_t, \end{aligned}$$

where,

$Y_t^{aug} \quad = X_t - (\alpha\beta' + I + \Gamma)X_{t-1} + \Gamma X_{t-2}\ .$

We can now write the equations of the DGS in the following state space form, noting that $\mathbb{I}_{n_x}$ is an identity matrix of dimension n_x.

Table 3.31: Case Study Augmented Data Generating System (DGS) Equations

Augmented DGS	System Equation	Measurement Equation
1	$M_t^{aug} = AM_{t-1} + \eta_t$	$Y_t^{aug} = \mathbb{I}_{n_x} M_t + \epsilon_t$
2	$M_t^{aug} = CM_{t-1} + \eta_t$	$Y_t^{aug} = \mathbb{I}_{n_x} M_t + \epsilon_t$

For each augmented data generating system, a total of 300 data sets were generated. 50 data sets containing 50, 100, 200, 400, 800 and 1000 observations respectively. The parameters that were used with each DGS is presented in Table (3.32).

Table 3.32: Case Study Augmented Data Generating System (DGS) Parameter Values

Parameter	Value
n_x	2
A	$\begin{pmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{pmatrix}$
B	$\begin{pmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{pmatrix}$
C	$\begin{pmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{pmatrix}$
D	$\begin{pmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{pmatrix}$
Σ	$\begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}$
α	$\begin{pmatrix} 0.2 & 0.2 \\ 0.2 & 0.2 \end{pmatrix}$
β	$\begin{pmatrix} 1 & 0 \\ -1 & -1 \end{pmatrix}$
Γ	$\begin{pmatrix} 0.1 & 0.1 \\ 0.1 & 0.1 \end{pmatrix}$
Ω	$\begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}$

We note that the α and β matrices of the augmented Data Generating System, given in Table (3.32), introduce a cointegrating relationship into the measurement equations of both DGSs, since, $\text{rank}(\alpha\beta') = 1$ which is less than n_x.

3.8.2 Method

The efficacy of the Sequential Importance Resampling (SISR) filter consisting of 100 samples (particles) at estimating the state of the Data Generating System (DGS) using a spectrum of Signal to Noise Ratios (SNR)'s, provided in Table (3.35), and resampling schemes, provided in Table (3.34).

The Sequential Importance Resampling filter for the system equation of each DGS had the following form,

$$X_t = f_t\left(X_{t-1}\right) + W_{t-1}, \qquad W_{t-1} \sim MVN(0, Q_t) \tag{3.49}$$

$$Y_t = h_t\left(X_t\right) + V_t, \qquad V_t \sim MVN(0, R_t) \tag{3.50}$$

where,

$X_t \in \mathbb{R}^{n_x}$	is the state vector,
$Y_t \in \mathbb{R}^{n_x}$	is the measurement vector,
f_t: $\mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is a non-linear function,
$W_t \in \mathbb{R}^{n_x}$	is the white, Gaussian, zero mean process noise vector with covariance Q_t,
h_t: $\mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$	is a non-linear function,
$V_t \in \mathbb{R}^{n_x}$	is the white, Gaussian, zero mean measurement noise vector with covariance R_t.

The prior distribution, proposal distribution and likelihood used in the SISR filter are shown below in Table (3.33).

Table 3.33: Components of the Sequential Importance Resampling Particle Filter

Component	Form
$p(X_0)$ (Prior)	$\delta_{X_0}(X) = X_0$
$p(X_{t+1}\|X_t^{(i)})$ (Proposal)	$MVN(X_{t+1}\|f_t(X_{t-1}), \Sigma)$
$p(Y_t\|X^{(i)})_t$ (Likelihood)	$MVN(Y_t\|h_t(X_t), \Omega)$

The resampling schemes used within each simulation are shown below in Table (3.34). A thorough exposition of the resampling schemes is provided in Section (3.5.4).

Table 3.34: Case Study Resampling Schemes

Resampling Scheme Number	Resampling Scheme
1	Multinomial
2	Residual
3	Stratified
4	Systematic

The spectrum of Signal to Noise Ratio (SNR) settings used for each simulation is shown in table 3.35. The SNR was varied by increasing or decreasing the covariance of the measurement noise vector, R_t, relative to the covariance of the process noise vector, Q_t.

Table 3.35: Simulation Signal to Noise Ratio (SNR) settings

Simulation Number	Signal to Noise Ratio (dB)
1	10
2	0
3	-10

3.8.3 Results

A summary of the results of each simulation for both Data Generating Systems are shown in the proceeding sections.

Data Generating System 1

From Figure (3.10), it is clear that the *a posteriori* state estimate, $\hat{\mathbf{X}}_{1,t+1|t+1}$, approximates the true state vector, $\mathbf{X}_{1,t}$, more accurately at intermediate Signal to Noise Ratio (SNR) values. In the lowest SNR setting the *a posteriori* state estimate is shown to approximate the mean of the latent process, whereas in the highest SNR setting, the *a posteriori* state estimate is subject to wild fluctuations. An explanation for this phenomenon is the effect of sample impoverishment, that is, the loss of diversity in the sample (particle) population. This in turn leads to divergence of the estimates. Filter divergence may be caused by the use of a sub-optimal proposal density. If the prior density has a broader distribution compared to the likelihood, then only a few samples (particles) will have a high importance weight. Another explanation is a lack of samples (particles) used.

Table 3.36: Mean Squared Error (MSE) and Effective Sample Size of the Sequential Importance Resampling filter estimates from Data Generating System 1 with Signal to Noise Ratio of 10dB

Resampling Scheme	Number of Observations	MSE $\mathbf{X}_{1,t}$	MSE $\mathbf{X}_{2,t}$	Effective Sample Size
1	50	0.1160	0.1088	1.0277
	100	0.1361	0.1345	1.0289
	500	0.1663	0.1699	1.0004
	1000	0.1789	0.1777	1.0003
2	50	0.1139	0.1141	1.0264
	100	0.1330	0.1325	1.0278
	500	0.1671	0.1669	1.0007
	1000	0.1767	0.1787	1.0003
3	50	0.1140	0.1126	1.0274
	100	0.1341	0.1337	1.0285
	500	0.1668	0.1709	1.0005
	1000	0.1756	0.1780	1.0004
4	50	0.1203	0.1134	1.0262
	100	0.1371	0.1351	1.0293
	500	0.1654	0.1691	1.0006
	1000	0.1791	0.1784	1.0004

From Table (3.36) it is evident that the use of the residual resampling scheme produces the state estimates with the lowest Mean Square Error (MSE). It is of interest to note that the residual resampling scheme also has the lowest average Effective Sample Size (ESS). An increase in the number of observations resulted in an increased MSE and lower average ESS. This suggests that the filter diverged.

From Table (3.37) it is evident that the use of the residual resampling scheme produces the state estimates with the lowest Mean Square Error (MSE).

From Table (3.38) it is evident that the use of the residual resampling scheme produces the state estimates with the lowest Mean Square Error (MSE). It is of interest to note that the residual

Table 3.37: Mean Squared Error (MSE) and Effective Sample Size of the Sequential Importance Resampling filter estimates from Data Generating System 1 with Signal to Noise Ratio of 0dB

Resampling Scheme	Number of Observations	MSE $\mathbf{X}_{1,t}$	MSE $\mathbf{X}_{2,t}$	Effective Sample Size
1	50	0.1289	0.1237	1.6036
	100	0.1468	0.1456	1.4907
	500	0.1715	0.1741	1.0428
	1000	0.1826	0.1823	1.0239
2	50	0.1258	0.1354	1.5918
	100	0.1460	0.1456	1.4936
	500	0.1738	0.1726	1.0396
	1000	0.1814	0.1825	1.0263
3	50	0.1317	0.1245	1.6035
	100	0.1518	0.1441	1.4962
	500	0.1712	0.1752	1.0409
	1000	0.1806	0.1824	1.0265
4	50	0.1332	0.1286	1.6054
	100	0.1496	0.1499	1.5012
	500	0.1706	0.1782	1.0372
	1000	0.1825	0.1815	1.0235

Table 3.38: Mean Squared Error (MSE) and Effective Sample Size of the Sequential Importance Resampling filter estimates from Data Generating System 1 with Signal to Noise Ratio of -10dB

Resampling Scheme	Number of Observations	MSE $\mathbf{X}_{1,t}$	MSE $\mathbf{X}_{2,t}$	Effective Sample Size
1	50	0.0998	0.0987	0.0998
	100	0.1016	0.1016	0.1016
	500	0.1267	0.1287	0.1267
	1000	0.1466	0.1470	0.1466
2	50	0.0983	0.0991	0.0983
	100	0.1005	0.1022	0.1005
	500	0.1282	0.1300	0.1282
	1000	0.1452	0.1456	0.1452
3	50	0.0989	0.0979	0.0989
	100	0.1018	0.1013	0.1018
	500	0.1275	0.1289	0.1275
	1000	0.1459	0.1467	0.1459
4	50	0.0995	0.0997	0.0995
	100	0.1016	0.1041	0.1016
	500	0.1271	0.1305	0.1271
	1000	0.1453	0.1459	0.1453

resampling scheme also has the lowest Effective Sample Size (ESS).

The use of the Residual resampling scheme achieved the lowest Mean Square Error (MSE) over all Signal to Noise Ratio (SNR) settings. This despite having the lowest average Effective Sample Size (ESS) over all SNR settings. On the other hand, the resampling scheme that achieved the highest MSE was the Systematic resampling scheme, which also has the highest average ESS.

Data Generating System 2

From Figure (3.11), it is clear that the *a posteriori* state estimate, $\hat{\mathbf{X}}_{1,t+1|t+1}$, approximates the true state vector, $\mathbf{X}_{1,t}$, more accurately at higher Signal to Noise Ratio (SNR) values. In the lowest SNR setting the *a posteriori* state estimate is shown to approximate the mean of the latent process. The Figure shows that the state estimates fluctuate wildly near the end of the observations. An explanation for this phenomenon is the effect of sample impoverishment, that is, the loss of diversity in the sample (particle) population. This in turn leads to divergence of the estimates. As before, an possible explanation is the use of a sub-optimal proposal distribution for the new set of samples (particles).

Table 3.39: Mean Squared Error (MSE) and Effective Sample Size of the Sequential Importance Resampling filter estimates from Data Generating System 2 with Signal to Noise Ratio of 10dB

Resampling Scheme	Number of Observations	MSE $\mathbf{X}_{1,t}$	MSE $\mathbf{X}_{2,t}$	Effective Sample Size
1	50	0.1191	0.1100	1.0637
	100	0.1387	0.1344	1.0247
	500	0.1680	0.1732	1.0004
	1000	0.1828	0.1825	1.0004
2	50	0.1198	0.1150	1.0636
	100	0.1343	0.1342	1.0249
	500	0.1698	0.1696	1.0005
	1000	0.1804	0.1842	1.0003
3	50	0.1208	0.1138	1.0632
	100	0.1361	0.1357	1.0251
	500	0.1710	0.1720	1.0005
	1000	0.1799	0.1828	1.0004
4	50	0.1229	0.1143	1.0637
	100	0.1380	0.1386	1.0247
	500	0.1695	0.1729	1.0006
	1000	0.1830	0.1832	1.0004

From Table (3.39) it is evident that the use of the residual resampling scheme produces the state estimates with the lowest Mean Square Error (MSE). It is of interest to note that the multinomial resampling scheme also has the lowest Effective Sample Size (ESS). An increase in the number of observations resulted in an increased MSE and lower average ESS. This suggests that the filter diverged.

From Table (3.40) it is evident that the use of the stratified resampling scheme produces the state estimates with the lowest Mean Square Error (MSE). It is of interest to note that the multinomial resampling scheme also has the lowest Effective Sample Size (ESS). An increase in the number of observations resulted in an increased MSE and lower average ESS. This suggests that the filter diverged.

From Table (3.41) it is evident that the use of the stratified resampling scheme produces the

Table 3.40: Mean Squared Error (MSE) and Effective Sample Size of the Sequential Importance Resampling filter estimates from Data Generating System 2 with Signal to Noise Ratio of 0dB

Resampling Scheme	Number of Observations	MSE $\mathbf{X}_{1,t}$	MSE $\mathbf{X}_{2,t}$	Effective Sample Size
1	50	0.1333	0.1272	1.8965
	100	0.1501	0.1485	1.4751
	500	0.1753	0.1778	1.0402
	1000	0.1866	0.1869	1.0340
2	50	0.1294	0.1359	1.8764
	100	0.1451	0.1479	1.4893
	500	0.1773	0.1751	1.0404
	1000	0.1843	0.1869	1.0377
3	50	0.1357	0.1262	1.8781
	100	0.1522	0.1452	1.4823
	500	0.1742	0.1780	1.0397
	1000	0.1846	0.1864	1.0362
4	50	0.1342	0.1317	1.8877
	100	0.1499	0.1529	1.4860
	500	0.1732	0.1805	1.0388
	1000	0.1862	0.1849	1.0358

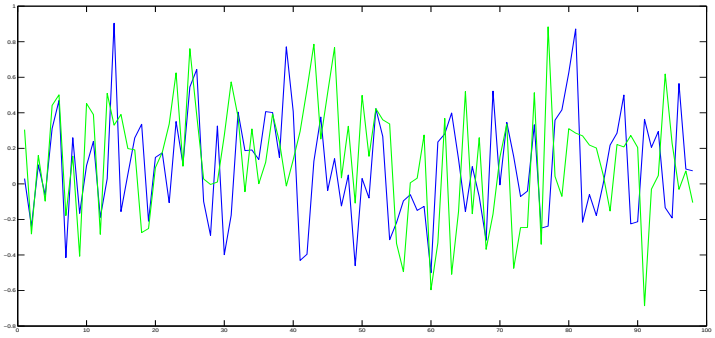
Table 3.41: Mean Squared Error (MSE) and Effective Sample Size of the Sequential Importance Resampling filter estimates from Data Generating System 2 with Signal to Noise Ratio of -10dB

Resampling Scheme	Number of Observations	MSE $\mathbf{X}_{1,t}$	MSE $\mathbf{X}_{2,t}$	Effective Sample Size
1	50	0.1028	0.0998	43.2630
	100	0.1042	0.1043	27.3130
	500	0.1292	0.1325	7.8044
	1000	0.1495	0.1501	4.9767
2	50	0.1008	0.1009	43.5530
	100	0.1025	0.1040	27.1720
	500	0.1315	0.1331	7.6089
	1000	0.1494	0.1498	4.9049
3	50	0.1011	0.0999	43.2460
	100	0.1035	0.1030	26.6220
	500	0.1310	0.1322	7.7754
	1000	0.1491	0.1498	4.8871
4	50	0.1008	0.1011	42.3860
	100	0.1038	0.1064	25.7300
	500	0.1308	0.1347	8.0253
	1000	0.1477	0.1485	5.1462

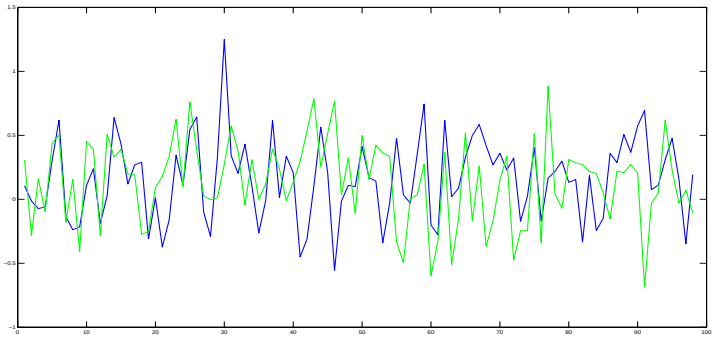
state estimates with the lowest Mean Square Error (MSE). It is of interest to note that the stratified resampling scheme also has the lowest Effective Sample Size (ESS). The average ESS of the SISR filter with a low SNR setting over all resampling schemes is notably larger than the other settings. A low SNR setting entails the system noise being larger than the measurement noise, which in turn allows corrections to be made by the random component of the SISR filter

update. That is, sub-optimal information is corrected.

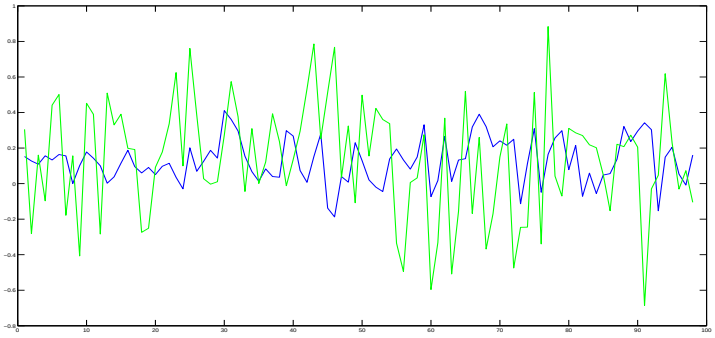
The use of the Stratified resampling scheme achieved the lowest Mean Square Error (MSE) over all Signal to Noise Ratio (SNR) settings. This despite having amongst the lowest Effective Sample Size (ESS) over all SNR settings. On the other hand, the resampling scheme that achieved the highest MSE was the Systematic resampling scheme, which also has the highest ESS.



(a) Simulation 1 (SNR 10 dB)

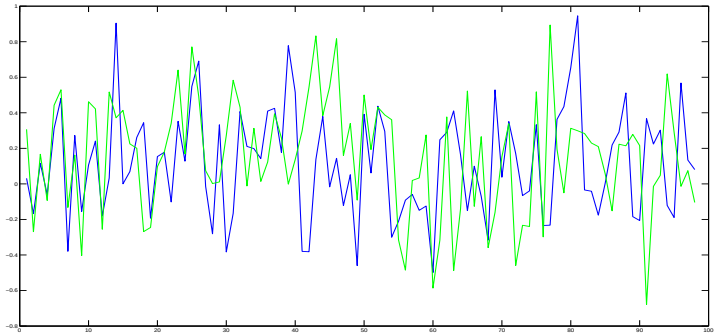


(b) Simulation 2 (SNR 0 dB)

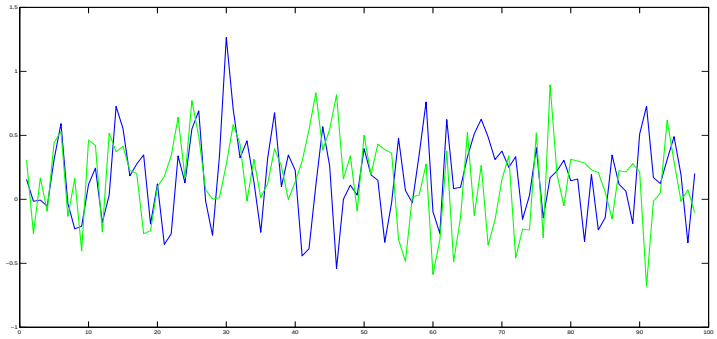


(c) Simulation 3 (SNR -10 dB)

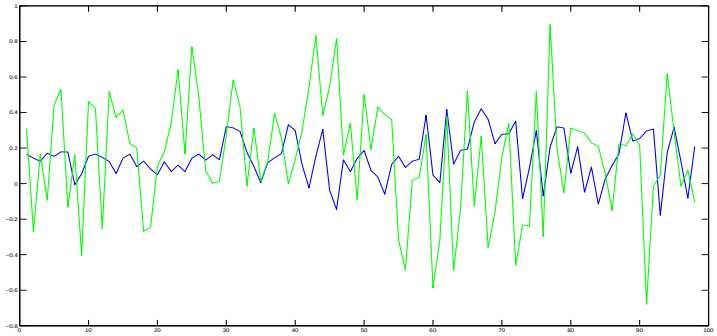
Figure 3.10: A set of typical Sequential Importance Sampling Resampling filter results for $\mathbf{X}_{1,t}$ from Data Generating System 1 using the Multinomial Resampling Scheme. In Figures (a), (b) and (c), the blue line refers to the *a posteriori* state estimate, $\hat{\mathbf{X}}_{1,t+1|t+1}$ and the green line refers to the true state vector, $\mathbf{X}_{1,t}$, at time t.



(a) Simulation 1 (SNR 10 dB)



(b) Simulation 2 (SNR 0 dB)



(c) Simulation 3 (SNR -10 dB)

Figure 3.11: A set of typical Sequential Importance Sampling Resampling filter results for $\mathbf{X}_{1,t}$ from Data Generating System 2 using the Multinomial Resampling Scheme. In Figures (a), (b) and (c), the blue line refers to the *a posteriori* state estimate, $\hat{\mathbf{X}}_{1,t+1|t+1}$ and the green line refers to the true state vector, $\mathbf{X}_{1,t}$, at time t.

3.9 Summary

Within this chapter we have presented the definition of the general filtering problem in conjunction with optimal and approximate solutions. The Kalman filter was presented as the optimal Baysian estimator under a high restricted state space model. The Extended Kalman filter was presented as an approximation of the optimal Bayesian solution that can be used in non-linear state space models. Three Sequential Monte Carlo approximations to the Bayesian interpretation of the general filtering problem, Sequential Importance Sampling, Sequential Importance Sampling Resampling and the Auxiliary Particle Filter were also discussed. It was noted that Sequential Monte Carlo approximations operate under much less rigorous bounds on the state space model. The Kalman and Extended Kalman filters were shown to be effective at estimating the state of a latent linear process, with the Kalman filter producing estimates with a lower Mean Square Error for linear latent processes. The Sequential Importance Sampling Resampling (SISR) filter was shown to depend heavily on the specification of its proposal distribution, number of particles and resampling scheme. We now move proceed to combine the ideas of the previous two chapters to elucidate the dynamics of a set of empirical data.

Chapter 4

Empirical Data Analysis

Pairs trading is a popular trading strategy used by statistical arbitrage hedge funds and private investors alike. The mechanisms and implementations of a pairs trading scheme are simple to implement. The aim of pairs trading is to finds two stocks whose prices move together over an indicated historical time period. If the pair prices deviate wide enough, the strategy calls for shorting the increasing-price asset, while simultaneously buying the declining-price asset. The idea behind the pair trade is to profit from convergence forces that eliminate short-term price deviations in favour of long-term historical pricing relationships. In a pricing world that is relatively efficient, simple strategies based on mean-reversion concepts should not generate consistent profits [69]. However, in 2006, Gatev et al. interpreted pairs as cointegrated prices and found that pairs trading generates consistent arbitrage profits in the U.S. equity markets [70]. The investigation by Gatev et al. is in stark contrast to more traditional methods of pairs trading, which have sought to identify trading pairs based on correlation, which seeks to identify short-term pricing relationships.

This chapter provides an investigation on the application of filtering techniques for multivariate cointegration models on empirical data. The empirical data used in the investigation, futures contracts on indices, interest rates and bonds, are presented in Section (4.1). In the subsequent Section, Section (4.2), we provide an exposition on commodities, an area of significant interest for speculative investors and arbitrageurs, significant market places for commodities trading and also introduce a variety of financial instruments that are linked to commodities. In the final section, Section (4.6), we conduct a case study which attempts to elucidate possible cointegrating relationships within the empirical data using the Johansen Maximum Likelihood method.

4.1 The Data

The data used in the empirical study consists of the following futures contract pairs: Five year (FV) and Ten year (TU) notes on US Treasury Bills, Australian Dollar (AUD) and Canadian Dollar (CAD) interest rates and Australian Dollar interest rate and E-mini NASDAQ-100 index (NQ). Prices from each pair were extracted in segments, with each segment containing price data taken at ten minute intervals over a varying number of trading days. The first and last trading day over all segments is shown in Table (4.1).

Table 4.1: Pair Futures Contract Price History Start and End Date

Pair	Start Date	End Date
AUD - CAD	2000-02-14	2006-03-09
FV - TU	1999-12-13	2003-11-25
NQ - AD	1999-12-13	2004-03-10

The number of trading days within each segment varies based on the roll date of the futures contract, which is not strictly equal to the maturity date. The roll date of each contract was set by the broker based on observed volume. The number of prices within each trading day for all segments is shown in Table ().

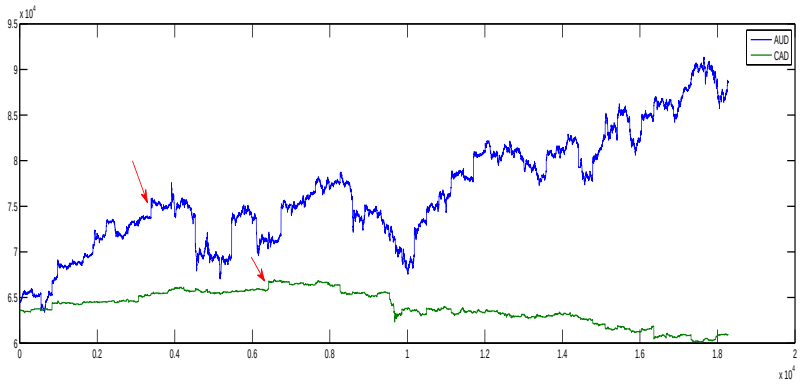


Figure 4.1: Futures Contract prices between Australian Dollar and Canadian Dollar Interest Rates between 2000-02-14 and 2000-03-07. Two examples of level shifts, also known as structural breaks, are shown by red arrows.

Upon inspection of the price data, it is evident that it contains *level shifts*, also known as *structural breaks*, this is illustrated in Figure (4.1). With the current data the level shifts occur at the boundary between the close of a financial market and its subsequent open and also within a trading day. It is hypothesised that the day boundary level shifts are due to the flow of information relating to the pairs on other financial markets whilst the market that the pairs are actively traded on are closed. For the purpose of the following investigation, the level shifts occurring at day boundaries were removed by discarding prices ten minutes before and after the

day boundary, which in affect removes the last price and first price of consecutive days. The intra-day level shifts were not removed.

4.2 Commodities

A commodity is defined as being a good for which there is demand and has either full or partial fungibility. Commodities are classified as being either *hard* commodities which are generally mined and *soft* commodities which are generally grown. Financial instruments that are linked to commodities are used by commodity producers, traders and consumers to hedge, speculate or arbitrage against uncertain price movements in either the underlying commodity or financial instrument.

Financial instruments are available as either standardised or tailor-made contracts. Standardized contracts are usually traded on commodity exchanges, whereas tailor-made contracts are traded Over the Counter (OTC) directly between two market participants. The types of commodity linked financial instruments are numerous and varied, they include forward contracts, futures contracts, options and swaps.

4.3 Commodity Market Places

Commodities contracts are traded on commodity exchanges and over the counter.

4.3.1 Commodity Exchanges

Commodity exchanges are market places for the trading of standardised commodity contracts that have been defined by the exchange [9]. The first organised commodity exchange market with a standardised futures clearing system was established in 1730 in the Dojima section of Osaka, Japan [71]. A recent survey lists over seventy commodity exchanges in operation world-wide, with the majority of them located in Asia [72]. The primary commodity exchanges are CME Group (CME), New York Mercantile Exchange (NYMEX), Eurex, Korean Exchange (KRX) and Shanghai Futures Exchange (SHFE).

Commodity exchanges are involved in the trading of standardised commodity contracts. A commodity contract specifies the type of underlying asset, its quality, volume and specific delivery times and procedures. All exchanges have a clearinghouse, which automatically acts as a counterpart to all transactions on the exchange; and guarantees the performance of the parties to the transaction. Regulation of commodity exchanges are commonplace and are designed to maintain the solvency of the market and approve the contracts specified by the exchange [9].

Korea Futures Exchange

The Korea Exchange (KRX) was founded in 2005 from the merger of four domestic Korean exchanges, the Korea Stock Exchange (KSE), the Korea Futures Exchange (KOFEX), the Kosdaq Market and the Kosdaq Committee. In 2010 the KRX was the largest derivatives exchange in the world with over 3.7 billion futures and options traded and/or cleared [73]. The KRX clears its exchange traded business through its own clearing house. The KRX offers contracts in futures, options, warrants, interest rates, index products, equity products as well as commodity products.

CME Group Inc.

The CME Group, Inc. (CME) was formed in 2007 by the merger of the Chicago Mercantile Exchange and the Chicago Board of Trade (CBOT). In 2010 the CME Group had the second highest number of futures and options traded and/or cleared of all derivatives exchanges worldwide [73]. CME Group offers contracts in all major asset classes including: commodity, interest rate, foreign currency, stock index, metals and alternative investment instruments such as weather and real estate derivatives [74]. CME Clearing is the CME's central futures clearing mechanism, which settles all trades and acts as the counterparty between buyers and sellers. The actions of the Group are regulated by the Commodity Futures Trading Commission (CFTC).

New York Mercantile Exchange

The New York Mercantile Exchange (NYMEX) was established in 1972 and is the world's largest energy and metals commodity exchange. In 2008 the NYMEX became a unit of the CME Group Inc. The NYMEX offers trading in crude oil, petroleum products, natural gas, coal, electricity, gold, silver, copper, aluminium, platinum group metals, emissions and soft commodities contracts [75]. The NYMEX WTI Crude Oil Futures was the most traded energy futures contract worldwide in 2010 [73]. Nine other futures and options contracts traded on the NYMEX were ranked in the top twenty most traded energy futures and options worldwide in 2010. The NYMEX has its own clearing operation where all contracts are cleared.

Eurex

The Eurex was formed in 1998 and is jointly operated by Deutsche Borse, and the SIX Swiss Exchange [76]. In 2010 the Eurex exchange was the third largest derivatives exchange in the world with over 2.6 billions futures and options traded and/or cleared [73]. The actions of the Eurex are regulated by Bundesanstalt für Finanzdienstleistungsaufsichtthe (BaFin) in Germany and its U.S. operations are subject to the regulation of both the Commodity Futures Trading Commission (CFTC) and the Securities and Exchange Commission (SEC). The Eurex exchange

clears its exchange traded derivatives business through its own clearing house, Eurex Clearing AG.

The Eurex is one of the world's most diverse derivatives exchanges, offering product ranges in interest rates, equity, equity index, exchange traded funds, credit, inflation, commodities, weather and property derivatives. The Eurex also matches, clears and settles over the counter transactions via its Eurex Bonds and Eurex Repo subsidiaries [76].

Shanghai Futures Exchange

The Shanghai Futures Exchange (SHFE) was formed in 1998 from the merger of the Shanghai Metal Exchange, the Shanghai Cereals and Oil Exchange and the Shanghai Commodity Exchange [77]. In 2010 the SHFE was the eleventh largest derivatives exchange in the world with over 621 million futures and options traded and/or cleared [73]. The SHFE is a self-regulated, non-profit organization, overseen by the China Securities Regulatory Commission.

The Shanghai Futures Exchange offers futures contracts in copper cathodes, aluminium, natural rubber, fuel oil, zinc, gold, steel wire and steel rebar. In 2010, the SHFE steel rebar, zinc, copper cathodes, aluminium futures contracts were the first, second, fourth and thirteenth, respectively, most traded metals futures and options worldwide. The SHFE natural rubber futures contract was the second most traded agricultural futures contract worldwide in 2010 [73].

4.3.2 Over the Counter Markets

The Over the Counter (OTC) market is an important alternative to exchanges and when measured in terms of total volume traded, has become the larger of the two [9]. At the end of June 2010, the gross market value of the global OTC derivatives market was estimated to be 24.673 billion USD [78].

Over the Counter markets are involved in the trading of non-standardised contracts. A non-standardised contract is the outcome of direct negotiation between two market participants and are tailored to suit the specific requirements of these participants. Since OTC contracts are traded directly between two parties, they cannot be easily traded or resold.

In contrast to commodity exchanges, price information in OTC markets are not transparent. Moreover, the absence of clearing houses for OTC markets means that all participants run counterpart risks. In an effort to reduce counterpart risk, collateralization agreements are often used between participants.

4.4 Market Participants

The liquidity found in commodity markets, both exchange and over the counter, make it attractive for traders. Three broad categories of traders can be identified, each with differing investment goals, they are hedgers, traders and arbitrageurs.

Hedgers use derivatives to reduce the risks faced from potential adverse future movements in a market variable. On the commodity exchanges, hedgers can easily find a counterpart for the proposed sale or purchase of a contract. This is due to the presence of both other hedgers who wish to take an opposite position and speculators.

Speculators use them to bet on the future directions of a market variable.

Arbitrageurs take offsetting positions in two or more instruments to lock in a profit [9].

4.5 Financial Instruments

Financial instruments are any contracts that give rise to a financial asset of one party and a financial liability or equity instrument of another party. Examples of financial instruments linked to commodities include futures contracts, forward contracts, options and swaps.

4.5.1 Futures Contracts

A commodity futures contract is a contract between two parties to buy or sell a specified quantity of a commodity at a future date at a price agreed upon when entering into the contract. The price paid at future settlement of the contract is known as the *futures* price. The *spot* price of a commodity future contract represents the cost of immediate settlement of the contract. A commodity futures contract is exchange traded and defined on standardised assets [9]. Upon entering a futures contract, no cash changes hands between buyers and sellers. Hence, the value of the contract is zero at its inception [79].

Because the future spot price is unknown today, a futures contract is a way to lock in the terms of trade for future transactions. In determining the fair futures price, market participants will compare the current futures price to the spot price that can be expected to prevail at the maturity of the futures contract. In other words, futures markets are forward looking and the futures price will embed expectations about the future spot price. If spot prices are expected to be much higher at the maturity of the futures contract than they are today, the current futures price will be set at a high level relative to the current spot price. In the converse situation, where spot prices are expected to be much lower at the maturity of the futures contract than they are today, the current futures price will be set at a low level relative to the current spot price [79].

Foreseeable trends in spot markets are taken into account when the futures prices are set. As a result expected movements in the spot price are not a source of return to an investor in futures.

Futures investors will benefit when the spot price at maturity turns out to be higher than expected when they entered into the contract, this situation is known as *contango*. The converse situation, in which the spot price is lower than anticipated, is known as *normal backwardation*. A futures contract is therefore a bet on the future spot price, and by entering into a futures contract an investor assumes the risk of unexpected movements in the future spot price. Furthermore, as the delivery period for a futures contract is approached, the futures price converges to the spot price of the underlying asset [9]. Unexpected deviations from the expected future spot price are by definition unpredictable, and should average out to zero over time.

Commodity futures have become widespread investment vehicles among traditional and alternative asset managers. They are now commonly used for strategic and tactical asset allocations. The strategic appeal of commodity indices comes from their equity-like return, their inflation-hedging properties and their role for risk diversification [80]. Recent research has also established that commodity futures can be used to generate abnormal returns [81].

4.5.2 Forward Contracts

Forward contracts are non-standardised agreements to purchase or sell a specified amount of a commodity on a fixed future date at a predetermined price. A forward contract is traded in the over the counter market between two parties, one of which assumes the *long position* and the other the *short position*. The party with the long position agrees to buy the underlying asset on a certain specified future date for a certain specified price. The other party agrees to sell the asset on the same date for the same price.

The use of a predetermined price in the forward contract eliminates risk associated with adverse price movements affecting both parties. If the spot price of the contract at maturity is higher than at inception, then the party assuming the long position benefits. If, on the other hand, the spot price is lower, then the short position profits. There is an inherent credit or default risk associated with forward contracts, which can be overcome by using collateralized agreements.

Forward contracts are mostly used to hedge the risk of holding a certain commodity or of having the obligation to deliver or acquire it at a future date. This is called *forward cover* and involves the execution of a set of offsetting transactions simultaneously in the spot and the forward markets.

4.5.3 Options

An option contract is the right, but not the obligation, to purchase or sell a certain commodity at a predetermined price on or before a specified date. There are two types of options: a *call option* and a *put option*. A call option gives the holder a right to buy the underlying asset for a certain price, known as the *strike price*, by a certain date, known as the *maturity date*. A

put option gives the holder the right to sell the underlying asset by a certain date for the strike price. American options can be exercised at any time up to the maturity date whereas European options can only be exercised on the maturity date itself [9]. Options contracts are traded on both exchange and over the counter markets.

There are two sides to every options contract, a participant that has taken the long position and a participant that has taken the short position. The participant with the short position receives cash up front but has potential liabilities later [9].

Options markets have four types of participants: buyers and sellers of call options and buyers and sellers of put options. This right to buy at a pre-set price is attractive for those who think that the market price will increase; it will enable them to buy at the lower price. It gives price protection to consumers and to processors and traders for the cost of the commodities they purchase. The profit or loss of the party assuming the short position is the inverse of the party assuming the long position.

Options can perform almost the same hedging functions as futures or forward contracts. Options contracts differ from the aforementioned contracts in two important aspects; namely the setting of floor and ceiling prices and the counterparty risk faced by a buyer of an option contract.

4.5.4 Swaps

A swap is a contract between two parties to exchange cash flows in the future. The agreement defines the dates when the cash flows are to be paid and the way in which they are to be calculated. Usually the calculation of the cash flows involves the future value of a market variable such as an interest rate [9].

Swaps were developed in the over the counter market as a long term price risk management instrument, some swaps are also traded on exchanges. As a commodity swap is a purely financial transaction, it has the advantage of allowing the producer and consumer to hedge their price exposure without directly affecting their commodity production, distribution or procurement activities.

4.6 Case Study: Johansen Maximum Likelihood Estimation

The Johansen Maximum Likelihood (ML) estimation method provides estimates of the parameters Vector Autoregressive Model (VECM) of order p. In this study we evaluate the efficacy of the Johansen ML method at estimating the parameters of three empirical data sets, each containing a pair of futures contracts. The case study utilises the novel implementation of the Johansen ML method in Object-oriented MATLAB is given in Appendix (B).

4.6.1 Model

The empirical data used within the case study consists of ten minute prices of the following pairs of futures contracts: Five year (FV) and Ten year (TU) notes on US Treasury Bills, Australian Dollar (AUD) and Canadian Dollar (CAD) interest rates and Australian Dollar interest rate and E-mini NASDAQ-100 index (NQ). A thorough explanation of the empirical data is presented in Section (4.6).

The number of trading days within for each pair and the average number of prices available within each trading day is shown in Table (4.2). From Table (4.2) it is clear that the AUD-CAD pair contains the longest price history, with the largest number of prices per trading day.

Table 4.2: Empirical Data Statistics

Pair	Number of Trading Days	Average Number of Prices per Day
AUD-CAD	789	520
FV-TU	502	372
NQ-AUD	538	291

4.6.2 Method

The observations from the prices of each pair were presented to the Johansen Maximum Likelihood (ML) method in batch. The order of the Vector Error Correction Model (VECM) estimated by the Johansen ML method was varied from 1 to 4.

In the batch procedure, the data sets were partitioned into subsets, each with an incrementally increasing number of observations, where the size of the increments was a predefined constant, Batch_size. At each time step, t, a subset of data $\mathbb{Y}_{1:t} = \{y_1, \ldots, y_{t+\text{Batch\_size}}\}$ was used within the Johansen ML method. By presenting the data in batch subsets, it is possible to evaluate the effect of increasing the number of observations on the performance of the Johansen ML method.

4.6.3 Results

A complete set of results for estimation of the parameters of a Vector Error Correction Model based on the futures contract pairs are presented in Appendices (H), (I) and (G), respectively.

E-mini NASDAQ-100 - Australian Dollar Interest Rate

From Appendix (I), it is evident that the most Vector Error Correction Model (VECM) specification that permits greatest amount of cointegrating relationships, over all batch data sets, is of order 4. However, the most frequently estimated cointegrating within this VECM specification

was 0. That is, within the VECM(4) model, the vast majority of batch data sets were found not to contain any cointegrating relationships.

The lack of cointegrating relationships found within the futures contracts may be attributable to model misspecification, or a true lack of cointegrating relationships. Model misspecification may arise through either under- or over-specification of the order of the VECM model or through the use of a static representation for the VECM model as opposed to a more dynamic representation, in which the components that make up the VECM are allowed to vary over time. It has been found that both the under- or over-specification of the lag length, which is related to the VECM order, has a negative impact on the hypothesis tests for cointegration rank [82]. While the effect on size is small for hypothesis tests for rank equal to zero, the effect on power is more substantial for certain parameter combinations. This effect becomes even more pronounced for tests of the null hypotheses of a cointegration rank of one [83]. A dynamic VECM representation can be achieved by allowing all parameters to varying over time. Popular choices for incorporating dynamic behaviour into a VECM include: stochastic volatility, deterministic trends, and markov switching regimes. Dynamic representations may be more effective at modelling the structural breaks that are present within the futures contracts price series. Structural breaks may invalidate cointegrating relationships that exist within the prices series.

Australian Dollar Interest Rate - Canadian Dollar Interest Rate

From Appendix (H), it is evident that the most Vector Error Correction Model (VECM) specifications that permit the greatest amount of cointegrating relationships, over all batch data sets, are of orders 1 and 2. However, the most frequently estimated cointegrating within these VECM specifications was 0. That is, within the VECM(1) and VECM(2) model, the vast majority of batch data sets were found not to contain any cointegrating relationships.

As mentioned previously in Section (4.6.3), lack of cointegrating relationships found within the futures contracts may be attributable to model misspecification, or a true lack of cointegrating relationships.

Five Year U.S. Treasury Bill - Ten Year U.S. Treasury Bill

From Appendix (H), it is evident that the most Vector Error Correction Model (VECM) specifications that permit the greatest amount of cointegrating relationships, over all batch data sets, are of orders 1 and 2. The most frequently estimated cointegrating within these VECM specifications was 1. That is, within the VECM(1) and VECM(2) model specifications, all of batch data sets were found to contain a cointegrating relationship. The Akaike [84], Hannan-Quinn [85], and Schwarz [86] information criterion can be used to select the most parsimonious specification.

When considered from an economic viewpoint, the discovery of a stable cointegrating relationship within the two futures contracts is expected. This is because the futures contracts relate to

different maturity dates, five and ten years, of the same underlying asset, a U.S. Treasury bill. As such, both prices will be affected by events, financial and otherwise, in similar ways.

The estimates of the parameters of the two Vector Error Correction Model specifications utilising the cointegration rank estimates are shown in Figures (4.2) and (4.3) respectively.

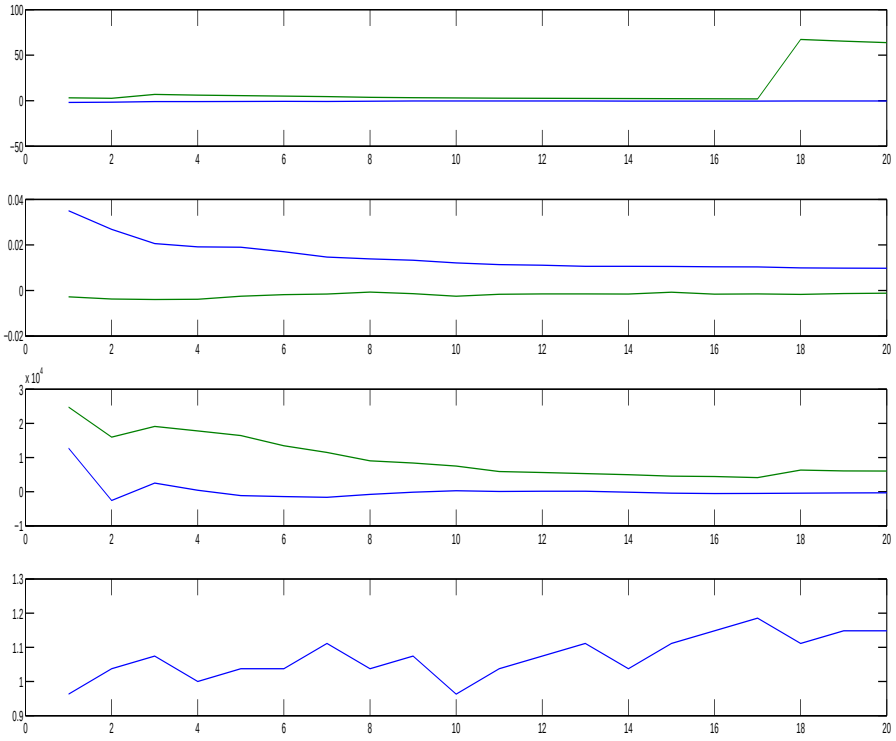


Figure 4.2: Five Year U.S. Treasury Bill - Ten Year U.S. Treasury Bill Futures Contract Pair. The figures, from top to bottom, represent the estimates of a Vector Error Correction Model of over 1, $\hat{\alpha}$, $\hat{\beta}$, $\hat{D}_t$ and $\hat{r}$ and for each batch subset over all data sets. In the first three Figures, the blue and green plots relate to the first and second components of the $\hat{\alpha}$, $\hat{\beta}$ and $\hat{D}_t$ vectors.

From Figure (4.2) it is evident that the estimates produced by the Johansen ML method stabilise as the number of observations is increased. In the first panel of Figure (4.2), the estimate of the

α vector, we can see that it is largely close to zero until the 17th batch data set, whereupon the estimates of the first component explodes by two orders of magnitude. It is also worth noting that the estimate of the cointegrating rank increases with a larger number of observations.

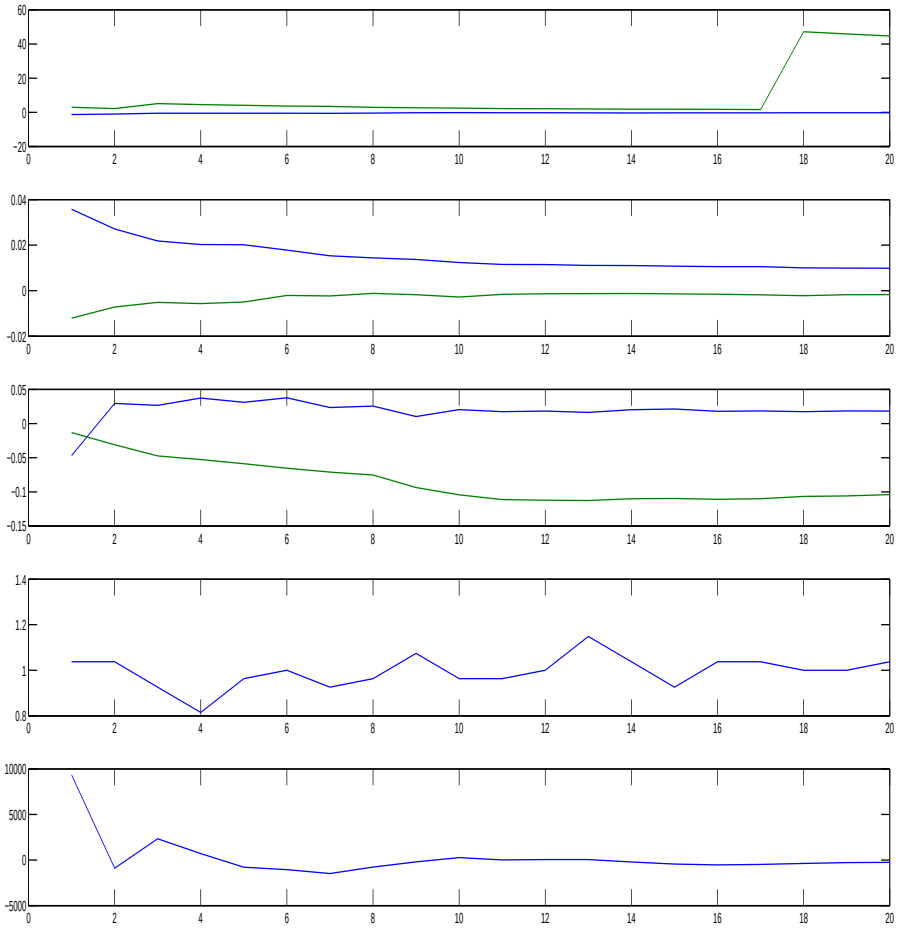


Figure 4.3: Five Year U.S. Treasury Bill - Ten Year U.S. Treasury Bill Futures Contract Pair. The figures, from top to bottom, represent the estimates of a Vector Error Correction Model of over 2, $\hat{\alpha}$, $\hat{\beta}$, $\hat{D}_t$, $\hat{r}$ and trace($\hat{\Gamma}$), or each batch subset over all data sets, In the first three Figures, the blue and green plots relate to the first and second components of the $\hat{\alpha}$, $\hat{\beta}$ and $\hat{D}_t$ vectors.

From Figure (4.3) it is evident that the estimates produced by the Johansen ML method stabilise as the number of observations is increased. In the first panel of Figure (4.3), the estimate of the α vector, we can see that it is largely close to zero until the 17th batch data set, whereupon the

estimates of the first component explodes by two orders of magnitude. It is also worth noting that the estimate of the trace of the lag matrix Γ decreases with a larger number of observations.

4.7 Summary

Within this chapter we have presented a set of empirical data which may be used within a pairs trading strategy. The dynamics of the three pairs which constitute the empirical data were presented and mention was made of the existence of structural breaks within their price series. In the case study we found that one of the pairs, futures contracts on Five year and Ten year U.S. Treasury Bills, exhibited a cointegrating relationship throughout its price history. The presence of this relationship meant that the pairs could be exploited within a pairs trading strategy to generate consistent profits. No such cointegrating relationship was found amongst the entire history of the other pairs that were considered.

Chapter 5

Conclusion and Extensions

Within this report we have explored the field of multivariate cointegrated time series, examining alternate representations and techniques for estimation of their parameters. The most popular estimation method, Johansen's Maximum Likelihood (ML) method, was found to be an effective estimator of the parameters of a multivariate cointegration model specified as a Vector Error Correction Model (VECM). A novel implementation of the Johansen ML method in Object-oriented MATLAB was also developed. A final application of the Johansen ML method to empirical futures contract data revealed its sensitivity to model misspecification, particularly in regards to the lag term in the VECM specification.

By representing multivariate cointegrated time series in state space form, we explored filtering techniques, with the aim of estimating a deterministic term in a VECM. Five types of filters were introduced; Kalman filter, Extended Kalman filter, Sequential Importance Sampling filter, Sequential Importance Sampling Resampling filter and the Auxiliary particle filter. The advantages and disadvantages of each filter were explained, with particular attention given to the Kalman, Extended Kalman and Sequential Importance Sampling Resampling filters; novel Object-oriented MATLAB implementations of the filters were also developed. The efficacy of each of the filters at discerning the deterministic intercept term of a VECM was shown to depend heavily upon the specification of the filtering model, particularly on the Signal to Noise ratio, and in the case of the Sequential Monte Carlo (SMC) methods, the resampling scheme.

Future extensions to the topics covered in the report can be categorised into three fields: multivariate cointegration models, filtering and parameter estimation. Theoretical extensions to the multivariate cointegration models include increasing the dimension of the system from 2 dimensions and including a greater number of dynamic terms within the VECM representation of the model. Examples of such terms include: stochastic volatility, stochastic trend, time varying lag structure and cointegration projection space. This would in effect lead to a time varying specification of a VECM [87]. A time varying VECM may result in a more robust represen-

tation of empirical time series, which often include such degeneracies as structural breaks and non-stationarity.

Extensions to the filtering component of this report include considering Sequential Monte Carlo methods, particularly those that utilise a different proposal mechanism, such as Markov Chain Monte Carlo [50], [88]. Another avenue for exploration is the implementation of SMC methods using parallel computing. The parallel computing paradigm can be used to alleviate the computation burden associated with iterating the SMC algorithm by performing calculations simultaneously over many computational nodes; often thousands of samples have to be evaluated at each filter iteration in order to achieve a satisfactory approximation of the posterior density. The effective speed gain of any parallelised algorithm is limited by the parts that must be performed sequentially. An exampling of a parallel implementation of an SMC algorithm is given in [89]. Parallel implementations of the resampling methods used within SMC algorithms are given in [90].

Bibliography

[1] K. Sugita, "Testing for cointegration rank using bayes factors," Royal Economic Society Annual Conference 2002 171, Royal Economic Society, Aug. 2002.

[2] C. Brooks, A. G. Rew, and S. Ritson, "A trading strategy based on the lead-lag relationship between the spot index and futures contract for the ftse 100," *International Journal of Forecasting*, vol. 17, no. 1, pp. 31 – 44, 2001.

[3] H. O. A. Wold and P. Whittle, "A study in the analysis of stationary time-series.," *Biometrika*, vol. 42, pp. viii + 236, 1955.

[4] R. F. Engle and C. W. J. Granger, "Co-integration and error correction: Representation, estimation, and testing," *Econometrica*, vol. 55, no. 2, pp. pp. 251–276, 1987.

[5] G. U. Yule, "Why do we sometimes get nonsense-correlations between time-series?–a study in sampling and the nature of time-series," *Journal of the Royal Statistical Society*, vol. 89, no. 1, pp. pp. 1–63, 1926.

[6] C. W. J. Granger and P. Newbold, "Spurious regressions in econometrics," *Journal of Econometrics*, vol. 2, pp. 111–120, July 1974.

[7] J. H. Stock and M. W. Watson, "Testing for common trends," *Journal of the American Statistical Association*, vol. 83, no. 404, pp. pp. 1097–1107, 1988.

[8] S. Rachev, S. Mittnik, F. Fabozzi, S. Focardi, and T. Jasic, *Financial Econometrics: From Basics to Advanced Modeling Techniques*. John Wiley & Sons, Ltd., 2007.

[9] J. C. Hull, *Options, Futures, and Other Derivatives with Derivagem CD (7th Edition)*. Prentice Hall, 7 ed., May 2008.

[10] C. Alexandra and A. Dimitriu, "The cointegration alpha: Enchanced index tracking and long-short equity market neutral stragies," ICMA Centre Discussion Papers in Finance icma-dp2002-08, Henley Business School, Reading University, 2002.

[11] N. Cappuccio and D. Lubian, "Practitioners' corner:triangular representation and error correction mechanism in cointegrated systems," *Oxford Bulletin of Economics and Statistics*, vol. 58, no. 2, pp. 409–415, 1996.

[12] P. C. B. Phillips, "Optimal inference in cointegrated systems," *Econometrica*, vol. 59, no. 2, pp. pp. 283–306, 1991.

[13] A. W. Phillips, "Stabilisation policy in a closed economy," *The Economic Journal*, vol. 64, no. 254, pp. pp. 290–323, 1954.

[14] J. Sargan, "Wages and prices in the united kingdom: A study of econometric methodology," in *Econometric Analysis for National Economic Planning* (P. Hart, G. Mills, and J. Whitaker, eds.), pp. 25–63, London, UK: Butterworth Co., 1964.

[15] D. F. Hendry and J.-F. Richard, "The econometric analysis of economic time series," *International Statistical Review / Revue Internationale de Statistique*, vol. 51, no. 2, pp. pp. 111–148, 1983.

[16] J. J. Dolado, J. Gonzalo, and F. Marmol, *Cointegration*, pp. 634–654. Blackwell Publishing Ltd, 2007.

[17] H. Lütkepohl, *New Introduction to Multiple Time Series Analysis*. Springer-Verlag Berlin Heidelberg, 2006.

[18] R. F. Engle and B. S. Yoo, "Forecasting and testing in co-integrated systems," *Journal of Econometrics*, vol. 35, no. 1, pp. 143 – 159, 1987.

[19] Y. K. Tse, "Lead-lag relationship between spot index and futures price of the nikkei stock average," *Journal of Forecasting*, vol. 14, no. 7, pp. 553–563, 1995.

[20] D. A. Dickey and W. A. Fuller, "Distribution of the estimators for autoregressive time series with a unit root," *Journal of the American Statistical Association*, vol. 74, no. 366, pp. pp. 427–431, 1979.

[21] J. H. Stock, "Asymptotic properties of least squares estimators of cointegrating vectors," *Econometrica*, vol. 55, no. 5, pp. pp. 1035–1056, 1987.

[22] P. C. B. Phillips and P. Perron, "Testing for a unit root in time series regression," *Biometrika*, vol. 75, pp. 335–346, 1988.

[23] S. E. Said and D. A. Dickey, "Testing for unit roots in autoregressive-moving average models of unknown order," *Biometrika*, vol. 71, no. 3, pp. pp. 599–607, 1984.

[24] J. H. Stock and M. W. Watson, "A simple estimator of cointegrating vectors in higher order integrated systems," *Econometrica*, vol. 61, no. 4, pp. pp. 783–820, 1993.

[25] S. Johansen, "Estimation and hypothesis testing of cointegration vectors in gaussian vector autoregressive models," *Econometrica: Journal of the Econometric Society*, vol. 59, no. 6, pp. 1551–1580, 1991.

[26] M. C. Lovell, "A simple proof of the fwl (frisch-waugh-lovell) theorem," Wesleyan Economics Working Papers 2005-012, Wesleyan University, Department of Economics, Dec. 2005.

[27] K. Juselius, *The Cointegrated VAR Model: Methodology and Applications*. No. 9780199285679 in OUP Catalogue, Oxford University Press, October 2006.

[28] R. Bewley and M. Yang, "Tests for cointegration based on canonical correlation analysis," *Journal of the American Statistical Association*, vol. 90, no. 431, pp. pp. 990–996, 1995.

[29] S. Johansen, *Likelihood-Based Inference in Cointegrated Vector Autoregressive Models*. No. 9780198774501 in OUP Catalogue, Oxford University Press, October 1995.

[30] J. G. MacKinnon, A. A. Haug, and L. Michelis, "Numerical distribution functions of likelihood ratio tests for cointegration," *Journal of Applied Econometrics*, vol. 14, pp. 563–77, Sept.-Oct 1999.

[31] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[32] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," *Radar and Signal Processing, IEE Proceedings F*, vol. 140, pp. 107 –113, apr. 1993.

[33] C. F. Gauss and C. H. Davis, *Theory of the motion of the heavenly bodies moving about the sun in conic sections [microform] : a translation of Gauss's "Theoria motus" / with an appendix by Charles Henry Davis*. Little, Brown, Boston :, 1857.

[34] H. W. Sorenson, "Least-squares estimation: from gauss to kalman," *IEEE Spectrum*, vol. 7, pp. 63–68, July 1970.

[35] M. S. Arulampalam, S. Maskell, and N. Gordon, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, pp. 174–188, 2002.

[36] J. Farrell and M. Barth, *The Global Positioning System & Inertial Navigation*. No. 4, McGraw-Hill, 1999.

[37] P. D. J. Richard S. Bucy, *Filtering For Stochastic Processes With Applications To Guidance*. American Mathematical Society, 2005.

[38] Y. Bar-Shalom, T. Kirubarajan, and X.-R. Li, *Estimation with Applications to Tracking and Navigation*. New York, NY, USA: John Wiley & Sons, Inc., 2002.

[39] M. S. Grewal and A. P. Andrews, *Kalman Filtering: Theory and Practice Using MATLAB*, vol. 5. John Wiley & Sons, Inc., 2001.

[40] C. L. Thornton, *Triangular Covariance Factorizaions for Kalman Filtering*. PhD thesis, University of California, Los Angeles, 1976.

[41] R. Fitzgerald, "Divergence of the kalman filter," *Automatic Control, IEEE Transactions on*, vol. 16, pp. 736 – 747, dec 1971.

[42] M. Piggott, "Filtering for linear/nonlinear state space models," tech. rep., University of New South Wales, 2010.

[43] S. Schmidt, *Advances in Control Systems. Theory and Applications,*, vol. 3, ch. Application of State-Space Methods to Navigation Problems., pp. 293–340. New York, San Francisco, London: Academic Press, 1966.

[44] A. C. Harvey and R. G. Pierse, "Estimating missing observations in economic time series," *Journal of the American Statistical Association*, vol. 79, no. 385, pp. pp. 125–131, 1984.

[45] X. Sun, L. Jin, and M. Xiong, "Extended kalman filter for estimation of parameters in nonlinear state-space models of biochemical networks," *PLoS ONE*, vol. 3, p. e3758, 11 2008.

[46] S. J. Julier and J. K. Uhlmann, "A new extension of the kalman filter to nonlinear systems," pp. 182–193, 1997.

[47] J. von Neuman, "Various techniques used in connection with random digits," *National Bureau of Standards, Applied Math Series*, vol. 11, pp. 36–38, 1951.

[48] I. Murray, "Note on rejection sampling and exact sampling with the metropolised independence sampler," 2004.

[49] A. W. Marshal, *The use of multi-stage sampling schemes in Monte Carlo computations. In Symposium on Monte Carlo methods.* Wiley, 1954.

[50] Z. Chen, "Bayesian filtering: From kalman filters to particle filters, and beyond," *Adaptive Syst Lab McMaster University, Hamilton, Ontario Canada*, pp. 1–69, 2003.

[51] A. Doucet and A. M. Johansen, *A Tutorial on Particle filtering and smoothing: Fiteen years later. In The Oxford Handbook of Nonlinear Filtering.* Oxford University Press, 2011.

[52] J. Geweke, "Bayesian inference in econometric models using monte carlo integration," *Econometrica*, vol. 57, no. 6, pp. pp. 1317–1339, 1989.

[53] N. Chopin, "Central limit theorem for sequential monte carlo methods and its application to bayesian inference," *The Annals of Statistics*, vol. 32, no. 6, pp. pp. 2385–2411, 2004.

[54] G. W. Peters, Y. Fan, and S. A. Sisson, "On sequential monte carlo, partial rejection control and approximate bayesian computation," 2008.

[55] J. Liu, *Monte Carlo strategies in scientific computing.* New York, Berlin, Heidelberg: Springer Verlag, 2008.

[56] S. N. Maceachern, M. Clyde, and J. S. Liu, "Sequential importance sampling for nonparametric bayes models: The next generation," *Canadian Journal of Statistics*, vol. 27, no. 2, pp. 251–267, 1999.

[57] J. S. Liu and R. Chen, "Sequential monte carlo methods for dynamic systems," *Journal of the American Statistical Association*, vol. 93, pp. 1032–1044, 1998.

[58] A. Kong, J. S. Liu, and W. H. Wong, "Sequential imputations and bayesian missing data problems," *Journal of the American Statistical Association*, vol. 89, no. 425, pp. pp. 278–288, 1994.

[59] D. B. Rubin, *Multiple Imputation for Nonresponse in Surveys*, pp. i–xxix. John Wiley & Sons, Inc., 1987.

[60] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *Journal of the American Statistical Association*, vol. 94, no. 446, pp. pp. 590–599, 1999.

[61] A. M. Johansen and A. Doucet, "A note on auxiliary particle filters," *Statistics & Probability Letters*, vol. 78, no. 12, pp. 1498 – 1504, 2008.

[62] N. Whiteley and A. M. Johansen, *Bayesian Time Series Models*, ch. Auxiliary particle filtering: recent developments. Cambridge University Press, 2011.

[63] J. D. Hol, T. B. Schön, and F. Gustafsson, "On resampling algorithms for particle filters," in *Nonlinear Statistical Signal Processing Workshop*, 2006.

[64] J. S. Liu, "Metropolized independent sampling with comparisons to rejection sampling and importance sampling," *Statistics and Computing*, vol. 6, pp. 113–119, 1996. 10.1007/BF00162521.

[65] G. Kitagawa, "Monte carlo filter and smoother for non-gaussian nonlinear state space models," *Journal of Computational and Graphical Statistics*, vol. 5, no. 1, pp. pp. 1–25, 1996.

[66] 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*, pp. 64 – 69, 2005.

[67] A. M. Johansen and L. Evans, "Monte carlo methods: Lecture notes." University of Bristol, 2007.

[68] H. R. Künsch, "Recursive monte carlo filters: Algorithms and theoretical analysis," *The Annals of Statistics*, vol. 33, no. 5, pp. pp. 1983–2021, 2005.

[69] J. P. Broussard and M. Vaihekoski, "Profitability of pairs trading strategy in finland." Available at SSRN: http://ssrn.com/abstract=1729182, December 2010.

[70] E. Gatev, W. N. Goetzmann, and K. G. Rouwenhorst, "Pairs trading: Performance of a relative-value arbitrage rule," *The Review of Financial Studies*, vol. 19, no. 3, pp. pp. 797–827, 2006.

[71] U. Schaede, "Forwards and futures in tokugawa-period japan:a new perspective on the dojima rice market," *Journal of Banking & Finance*, vol. 13, no. 4-5, pp. 487 – 513, 1989.

[72] U. Secretariat, "Overview of the world's commodity exchanges," January 2006.

[73] "Annual volume survey 2010," March 2011.

[74] MarketsWiki, "Cme group, inc.," 2011.

[75] MarketsWiki, "New york mercantile exchange, inc.," 2011.

[76] MarketsWiki, "Eurex," 2011.

[77] MarketsWiki, "Shanghai futures exchange," 2011.

[78] "Triennial and regular otc derivatives market statistics: Positions in global over-the-counter (otc) derivatives markets at end-june 2010," 2010.

[79] F. Black, "The pricing of commodity contracts," *Journal of Financial Economics*, vol. 3, no. 1-2, pp. 167–179, 1976.

[80] G. B. Gorton and K. G. Rouwenhorst, "Facts and fantasies about commodity futures," *Financial Analysts Journal*, vol. 62, pp. 47–68, 2006.

[81] G. R. Jensen, R. R. Johnson, and J. M. Mercer, "Tactical asset allocation and commodity futures.," *Journal of Portfolio Management*, vol. 28, pp. 100–111, 2002.

[82] D. Maringer and P. Winker, "Optimal lag structure selection in vec-models," Computing in Economics and Finance 2004 155, Society for Computational Economics, Aug. 2004.

[83] R. Bewley and M. Yang, "On the size and power of system tests for cointegration," *The Review of Economics and Statistics*, vol. 80, pp. 675–679, November 1998.

[84] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, pp. 716–723, Dec. 1974.

[85] E. J. Hannan and B. G. Quinn, "The Determination of the Order of an Autoregression," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 41, no. 2, pp. 190–195, 1979.

[86] G. Schwarz, "Estimating the dimension of a model," *Annals of Statistics*, vol. 6, pp. 461–464, 1978.

[87] G. Koop, R. Leon-Gonzalez, and R. W. Strachan, "Bayesian inference in the time varying cointegration model," Working Paper Series 23-08, Rimini Centre for Economic Analysis, Jan. 2008.

[88] C. Andrieu, A. Doucet, and R. Holenstein, "Particle markov chain monte carlo methods," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 72, no. 3, pp. 269–342, 2010.

[89] A. C. Sankaranarayanan, A. Srivastava, and R. Chellappa, "Algorithmic and Architectural Optimizations for Computationally Efficient Particle Filtering," *Image Processing, IEEE Transactions on*, vol. 17, no. 5, pp. 737–748, 2008.

[90] J. Míguez, "Analysis of parallelizable resampling algorithms for particle filtering," *Signal Process.*, vol. 87, pp. 3155–3174, December 2007.

Appendix A

MATLAB Phillips' Triangular Representation Estimation Implementation

```
classdef Triangular < handle
%TRIANGULAR Phillip's Triangular Representation Estimation

  % METHODS (Static)
  methods(Static = true)

    % Function: Estimate
    % Input:    Data (DataGeneratingSystem)
    % Output:   None
    % NOTES:
    % 1. This code only works with Data of dimension 2
    % 2. The restriction in 1. can be easily removed
    function results = Estimate(Data)

      % Initialise the results data structure
      results = NaN(Data.numberDataSets, 1);

      % Iterate through the datasets in the Data
      for dataSetNum = 1:Data.numberDataSets
        % The current dataset
        d1 = Data.getDataSet(dataSetNum);

        % Split the dataset into two vectors, one for each series
        y1 = d1(:, 1);
        y2 = d1(:, 2);

        % Construct the design matrix
        X = horzcat(y1);

        % Calculate the OLS estimate of Beta
        beta = (X' * X)\(X' * y2);
```

```
        results(dataSetNum) = beta;
      end
    end
  end
end
```

Appendix B

MATLAB Johansen Maximum Likelihood Method Implementation

```
classdef Johansen < handle
  %JOHANSEN Johansen Class
  %   The Johansen Class contains the code to run the Johansen ML method

  % PROPERTIES (Constant)
  properties(Constant)

    % Critical Values for the Johansen Trace and Maximum EigenValue Statistics
    % References: MacKinnon, Haug, Michelis (1996) 'Numerical distribution
    % functions of likelihood ratio tests for cointegration',
    % Queen's University Institute for Economic Research Discussion paper.
    % NOTES:
    %   1. Only lags <= 12 are supported
    %   2. The criticalValues for the appropriate NumberLags will be a (3x1)
    %   vector of percentiles for the trace statistic: [90 95 99]

    % Trace test
    CriticalValuesTable = [
    2.7055   3.8415   6.6349
    13.4294  15.4943  19.9349
    27.0669  29.7961  35.4628
    44.4929  47.8545  54.6815
    65.8202  69.8189  77.8202
    91.1090  95.7542 104.9637
    120.3673 125.6185 135.9825
    153.6341 159.5290 171.0905
    190.8714 197.3772 210.0366
    232.1030 239.2468 253.2526
    277.3740 285.1402 300.2821
    326.5354 334.9795 351.2150
    ];
```

```
  % Max eigenvalue test
  CriticalValuesTable2 = [
  2.7055   3.8415   6.6349
  12.2971  14.2639  18.5200
  18.8928  21.1314  25.8650
  25.1236  27.5858  32.7172
  31.2379  33.8777  39.3693
  37.2786  40.0763  45.8662
  43.2947  46.2299  52.3069
  49.2855  52.3622  58.6634
  55.2412  58.4332  64.9960
  61.2041  64.5040  71.2525
  67.1307  70.5392  77.4877
  73.0563  76.5734  83.7105];

  % The default significance level for the trace statistic hypothesis test
  % NOTE: The value should correspond to a column of the CriticalValuesTable
  DefaultSignificanceLevel = 2;
end

% METHODS (Static)
methods(Static = true)

  % Function: Estimate
  % Input:    Data (Matrix), Number Lags (Integer)
  % Output:   Results (Cell Array)
  % NOTES:
  function results = Estimate(Data, NumberLags)

    % Construct \Delta X_t (This is Z_0t)
    % NOTE: X_0t will have one less row because of the differencing operation
    z_0t = diff(Data);
    tz_0t = z_0t;

    % Construct X_{t-1} (Ths is Z_1t)
    z_1t = Data;
    z_1t(end, :) = [];

    % Construct (\Delta X_t, ..., \Delta X_{t-p+1}, D_t).(This is z_2t)
    z_2t = ones(size(z_0t, 1), 1);
    % Iterate through the NumberLags and build up the z_2t matrix iteratively
    % NOTE: This can be vectorised
    for lagNum = 1:NumberLags - 1
      tz_0t = tz_0t(1:end - 1, :);
      z_2t = [z_2t(2:end, :) tz_0t];
    end

    if(NumberLags > 0)
      % The lagged versions of z_0t and z_1t
      z_0t = z_0t((NumberLags):end, :);
      z_1t = z_1t((NumberLags):end, :);
    end

    % The number of observations remaining
    [numObservations, numSeries] = size(z_0t);

    % The moment matrices
    m_00 = (z_0t' * z_0t)/numObservations;
    m_01 = (z_0t' * z_1t)/numObservations;
```

```
    m_02 = (z_0t' * z_2t)/numObservations;
    m_11 = (z_1t' * z_1t)/numObservations;
    m_12 = (z_1t' * z_2t)/numObservations;
    m_22 = (z_2t' * z_2t)/numObservations;

    m_22Invm_12 = m_22\m_12';

    % Squared residual matrices
    s_00 = m_00 - (m_02 * (m_22\m_02'));
    s_01 = m_01 - (m_02 * m_22Invm_12);
    s_11 = m_11 - (m_12 * m_22Invm_12);

    % At this point we have all of the information required to estimate the
    % parameters
    [alpha, beta, omega, eigenValues] = ...
    Johansen.SolveEigenValueProblem(s_01, s_00, s_11);

    % The Psi matrix
    % NOTE: This matrix contains estimates of the lag parameters
    psi = (m_02 - alpha * beta' * m_12)/(m_22);

    % Calculate the test statistics
    [traceStatistics, maxEigenStatistics] = ...
    Johansen.CalculateTestStatistics(eigenValues, numObservations);

    % Determine the rank
    [rank, testVector] =
Johansen.DetermineCointegrationRank(maxEigenStatistics);
    %       [rank, testVector] =
    %Johansen.DetermineCointegrationRank(traceStatistics);

    % Calculate the maximum of the lof likelihood
    if(rank > 0)
      test = (1 - eigenValues);
      maxLikelihood = det(s_00) * prod(test(rank, :), 1);
    else
      maxLikelihood = 0;
    end

    % Construct the results Cell Array
    C = cell(7,1);
    C(1,1) = {alpha};
    C(2,1) = {beta};
    C(3,1) = {omega};
    C(4,1) = {psi};
    C(5,1) = {rank};
    C(6,1) = {testVector};
    C(7,1) = {maxLikelihood};
    results = C;
  end

  % Function: Determine Cointegration Rank
  % Input:    Test Statistics (Vector), Significance Level (Integer)
  % Output:   Rank (Integer), Test Vector (Vector)
  % NOTES:
  function [rank, testVector] = DetermineCointegrationRank(TestStatistics, .
    SignificanceLevel)
    % DetermineCointegrationRank.
```

```
      if(nargin == 2)
        significanceLevel = SignificanceLevel;
      else
        % The default SignificanceLevel
        significanceLevel = Johansen.DefaultSignificanceLevel;
      end

      numTestStatistics = length(TestStatistics);

      % The appropriate critical values
      % NOTE:
      % 1. This is a column vector
      % 2. Bonferroni adjustment needed to maintain a p-value of 0.05 for all
      % tests together and not individually
      criticalValues =
flipud(Johansen.CriticalValuesTable2(1:numTestStatistics,  ...
      significanceLevel));
      % The Bonferroni correction
      %       criticalValues = criticalValues / numTestStatistics;

      % The null and alternative hypotheses
      nullHypotheses = 0:(numTestStatistics - 1);
      altHypotheses = nullHypotheses + 1;

      % The indices where the null hypothesis would be rejected in a sequential
      % testing framework
      nullRejectedIndices = (TestStatistics > criticalValues)';

      % Find the where the null hypothesis was rejected
      % NOTE: nullRejectedMask = ~nullNotRejectedMask
      nullRejectedMask = find(nullRejectedIndices);
      %       nullNotRejectedMask = ~nullRejectedMask;

      % Determine the cointegration rank
      % NOTE:
      if(isempty(nullRejectedMask))
        % The null hypotheses has not been rejected in any of the tests.
        % Therefore, return the last alternate hypothesis
        rank = nullHypotheses(1);
      else
        % The smallest value of r for which the null hypothesis was not
%rejected
        maxNullReject = max(nullRejectedMask);
        % Return the alternate hypothesis
        rank = altHypotheses(maxNullReject);
      end

    %       if(isempty(nullRejectedMask))
    %         % The null hypotheses has not been rejected in any of the tests.
    %         % Therefore, return the first null hypothesis
    %         rank = nullHypotheses(1);
    %       else
    %         % The smallest value of r for which the null hypothesis was not
    %rejected
    %         minNullNotReject = min(nullNotRejectedMask);
    %
    %         if(minNullNotReject == 0)
    %           rank = altHypotheses(end);
    %         else
```

```
%           % Return the null hypothesis at the minNullNotReject index
%           rank = nullHypotheses(minNullNotReject);
%         end
%       end

  testVector = nullRejectedIndices;
end

% Function: Calculate Test Statistics
% Input:    EigenValues (Vector), Number of Observations (Integer)
% Output:   Trace Statistics (Vector), Maximum EigenValue Statistics
%           (Vector)
% NOTES:
function [trace, maxEigen] = CalculateTestStatistics(EigenValues, ...
NumberObservations)

  numEigenValues = length(EigenValues);

  % Common expression for both the maximum eigenvalues and trace statistic
  % calculations
  inner = log(1 - EigenValues);

  % The LR test statistic for the maximum eigenvalue test
  maxEigen = -NumberObservations * inner;

  % The LR test statistic for the trace test
  % Iterate through the possible ranks (1..full rank) for the Pi matrix
  trace = zeros(numEigenValues, 1);
  for r = 1:numEigenValues
    % Calculate the LR test statistic for the trace test
    trace(r,1) = -NumberObservations * sum(inner(r:end, 1));
  end
end

% Function: Solve Eigen Value Problem
% Input:    S_01 (Matrix), S_00 (Matrix), S_11 (Matrix)
% Output:   Alpha (Matrix), Beta (Matrix), Omega (Matrix), Normalised
%           EigenValues (Vector)
% NOTES:
function [alpha, beta, omega, eigenValues] = ...
  SolveEigenValueProblem(S_01, S_00, S_11)

  S_10 = S_01';

  % Solve the eigenvalue problem
  % NOTE: The calculation of may lead to Exceptions/Warnings
  try
    eigMatrix = S_11\(S_10/S_00) * S_01;

    % The eigenvecors and eigenvalues
    % NOTE:
    % 1. [V,D] = EIG(X,'nobalance') performs the computation with balancing
    % disabled, which sometimes gives more accurate results for certain
    % problems with unusual scaling.
    % 2. Should test for symmetric eigMatrix ?
    [vectorMatrix, valueMatrix] = eig(eigMatrix);
  catch Exception
    % The exception is most probably (empirical observations) caused by
    % ill-conditioned or singular s_00 or s_11 matrices
```

```
      %         fprintf('\tJohansen:Estimate:Cond(S_00) = %s\tCond(s_11) =
  %%s\n',
    %
      %...
        %                                                       cond(S_00),
        %cond(s_11));

        % Check the Exception.identifier
        if(strcmp(Exception.identifier, 'MATLAB:eig:matrixWithNaNInf'))
          % Attempt to solve the eigenvalue problem using the pseudo inverse
          % method
          eigMatrix = pinv(S_11) * (S_10 * pinv(S_00)) * S_01;
          [vectorMatrix, valueMatrix] = eig(eigMatrix);
        else
          rethrow(Exception)
        end
      end

      % Normalise the eigenvectors
      try
        innerMatrix = vectorMatrix' * S_11 * vectorMatrix;

        % The cholesky decomposition of the innerMatrix
        % NOTE: The cholesky decomposition function requires that a matrix be
        % positive definite i.e. symmetric and all of its eigenvalues are
        % positive
        cholDecomposition = chol(innerMatrix);

        % Normalise the eigen vectors
        % NOTE: This can lead to the following warning:
        % 'Warning: Matrix is close to singular or badly scaled.'
        normVectorMatrix = vectorMatrix/(cholDecomposition);
        [~, msgID] = lastwarn;

        % Check the lastwarn message ID
        if(strcmp(msgID, 'MATLAB:nearlySingularMatrix'))
          % Use the pseudo-inverse method
          normVectorMatrix = vectorMatrix * pinv(cholDecomposition);
          lastwarn('');
        end

        % Transpose the normalised eigenvector matrix
        normVectorMatrix = transpose(normVectorMatrix);
      catch Exception
        % The Exception is most probably caused by the requirements of the
        % cholesky decomposition function being violated i.e. the innerMatrix
%is
        % not positive definite

        % Report which requirement was violated
        %         if(~(min(eig(innerMatrix)) > 0))
        %           disp('Johansen:Estimate:Negative/Zero eigenvalues')
        %         end
        %
        %         if(~(all(all(innerMatrix == innerMatrix'))))
        %           disp('Johansen:Estimate:Non-symmetric')
        %         end

        rethrow(Exception)
```

```
      end

      % The eigenvalues and their indices in the normalised eigenvector matrix
      [sortedValues sortIndices] = sort(diag(valueMatrix));
      eigenValues = flipud(sortedValues);
      eigenValueIndices = flipud(sortIndices);

      % The estimates of alpha, beta and omega
      beta = transpose(normVectorMatrix(eigenValueIndices, :));
      alpha = S_01 * beta;
      %       alpha = S_01 * beta/(beta' * S_11 * beta)
      omega = S_00 - alpha * (beta' * S_11 * beta) * alpha';
    end
  end
end
```

Appendix C

MATLAB Kalman Filter and Extended Kalman Filter Implementation

```
classdef Filter < handle
  %FILTER Filter Class
  % The Filter Class contains the implementations of the Kalman and Extended
  % Kalman filters

  % METHODS (Static)
  methods(Static = true)

    % Function: Kalman
    % Input:    State Vector (), System Matrix (), Process Noise
    %           Vector (), State Cov Matirx (), Measurement Vector (),
    %           Measurement Matrix (), Measurement Noise Vector ()
    % Output:   State Estimate (), Covariance Estimate (), Kalman Gain (),
    %           Innovations ()
    % NOTES:
    % 1. State Equation:
    %   x_{t} = A_{t-1}*x_{t-1} + B_{t-1}*u_{t-1} + w_{t-1}, w_{t} ~ N(0,
%Q_{t})
    % 2. Observation Equation:
    %   z_{t} = H_{t}*x_{t} + v_{t}, v_{t} ~ N(0, R_{t})
    function [X_Post, P_Post, K, e] = Kalman(X, P, Obs, StateParams, ObsParams)
      % Kalman. Kalman Filter.

      A = StateParams.TransitionMatrix;
      Q = StateParams.NoiseMatrix;
      H = ObsParams.TransitionMatrix;
      R = ObsParams.NoiseMatrix;

      % Prediction for state vector and covariance
      X_priori = A * X;
      P_priori = (A * P * A') + Q;
```

```
      % Compute Kalman gain factor
      S = (H * P_priori * H') + R;
      K = (P_priori * H') / S;

      % Correction based on observation
      e = Obs - (H * X_priori);
      X_Post = X_priori + (K * (e));

      % Joseph form of covariance update
      M = eye(size(X, 1)) - (K * H);
      P_Post = (M * P_priori * M') + (K * R * K');
    end

    % Function: Extended Kalman Filter
    % Input:    A Priori State Estimate (Matrix), A Priori Covariance
    %           Estimate (Matrix), Observation (Vector),
    %           System Equation Parameters (Struct), Observation Equation
    %           Parameters (Struct)
    % Output:   A Posteriori State Estimate (Matrix), A Posteriori
    %           Covariance Estimate (Matrix)
    % NOTES:
    % 1. The SystemFunc, SystemDerivFunc, MeasureDerivFunc are function handles
    function [X_Post, P_Post, K, e] = ExtendedKalman(X_Priori, P_Priori, Obs,
  ...
      SystemParams, ObsParams)
      % ExtendedKalmanFilter.

      % System Equation Function Handle, System Equation Taylor Series
      % Handle and Measurement Equation Taylor Series Function Handle
      SystemFunc = SystemParams.SystemEquationFunctionHandle;
      SystemDerivFunc = SystemParams.SystemEquationDerivativeFunctionHandle;
      MeasureDerivFunc = ObsParams.MeasurementEquationDerivativeFunctionHandle;

      % State Prediction
      % \hat{x_{k|k-1}} = f(\hat{x_{k-1|k-1}})
      xPredicted = SystemFunc(X_Priori);

      % Jacobian calculation
      jacobianSystemMatrix = SystemDerivFunc(X_Priori);

      % Jacobian calculation
      jacobianMeasurementMatrix = MeasureDerivFunc(xPredicted);

      % Replace the Transition matrices for the System and Observation
      % Struct parameters by their jacobian counterparts
      SystemParams.TransitionMatrix = jacobianSystemMatrix;
      ObsParams.TransitionMatrix = jacobianMeasurementMatrix;

      % Run the Kalman Filter with the modified transition marices
      [X_Post, P_Post, K, e] = Filter.Kalman(xPredicted, P_Priori, Obs, ...
      SystemParams, ObsParams);
    end
  end
end
```

Appendix D

MATLAB Sequential Importance Resampling Filter Implementation

```
classdef GenericParticleFilter < handle
  %GenericParticleFilter Particle Filter Class
  %   This file contains code for the SIS-R Filter

  % PROPERTIES (Public)
  properties(GetAccess = public, SetAccess = protected)

    % The name of the filter
    name

    % The number of particles
    numberParticles

    % The threshold at which resampling will be executed
    resampleThreshold

    % The resampling scheme
    resampler

    % The final set of particles and their associated importance weights
    finalParticles
    finalImportanceWeights
    finalEffectiveSampleSize
    mmse
  end

  % METHODS (Static)
  methods(Static = true)

    % Function: Effective Sample Size
    % Input:    Weights (Vector)
```

```
    % Output:   Effective Sample Size (Float)
    % NOTES:
    function results = EffectiveSampleSize(Weights)
      % EffectiveSampleSize.

      % The effective sample size
      % NOTE: This calculation is from Marcus Piggott's paper
      results = 1/(sum(Weights.^2));
    end
  end

  % METHODS (Public)
  methods(Access = public)

    % Function: Particle Filter
    % Input:
    % Output:   None
    % NOTES:
      function this = GenericParticleFilter(NumberParticles, ResamplingScheme,
...ResamplingThreshold)
      % GenericParticleFilter. The default Constructor.

      this.name = 'SIS-R Filter';

      this.numberParticles = NumberParticles;

      this.resampleThreshold = ResamplingThreshold;
      this.resampler = ResamplingScheme;
    end

    % NOTES:
    %   1. This uses logged weights
    function [particles, weights, ess] = run(this, NumberTimeSteps, Data, ...
    SystemModelFunctionHandle, MeasurementModelFunctionHandle, ...
    InitialParticles, InitialWeights, Covariance)
      % Run.

      [dataLength, dataDimension] = size(Data);

      % Ensure that there is sufficient data for the specified
      % NumberTimeSteps
      if(NumberTimeSteps > dataLength)
        NumberTimeSteps = dataLength;
      end

      % The initial particles and their importance weights
      weights = InitialWeights;
      particles = InitialParticles;

      ess = NaN(NumberTimeSteps, 1);

      % Iterate through the each time step
      for t = 1:NumberTimeSteps

        % The current data point
        currentObservation = Data(t, :);

        % Generate the new states from the SystemModelFunctionHandle
        % NOTE: The SystemModelFunctionHandle is the proposal distribution
```

```
        particlesNew = SystemModelFunctionHandle(particles);
        particles = particlesNew;

        if(t == 1)
          % Log the initial set of weights
          logWeightsNew = log(weights);
        else
          % Generate a new set of importance weights
          logWeightsNew = logWeightsNew - ( ...
          (-(dataDimension/2) * log(2 * pi) -0.5 * log(det(Covariance))).*
ones(1,
          this.numberParticles) ...
          -0.5 * (((currentObservation(1) * ones(this.numberParticles, 1) -
          particlesNew(:, 1)).^2)./(2 * Covariance(1)^2))' ...
          -0.5 * (((currentObservation(2) * ones(this.numberParticles, 1) -
          particlesNew(:, 2)).^2)./(2 * Covariance(end)^2))' ...
          );
        end

        % The largest weight
        lMax = max(logWeightsNew);

        % Correct the weights up to a multiplicative factor
        correctedWeights = exp(logWeightsNew - lMax);

        % Normalise the weights
        weights = correctedWeights./sum(correctedWeights);
        stem(weights);

        % Calculate the effective sample size
        effectiveSampleSize =
GenericParticleFilter.EffectiveSampleSize(weights);
        ess(t) = effectiveSampleSize;

        % Resample if the effectiveSampleSize is less than the
        % this.resampleThreshold
        if (round(effectiveSampleSize) < this.resampleThreshold)
          %           fprintf('\tResampling (ESS: %d, Threshold: %d)\n',
          %round(effectiveSampleSize), ...
          %
          %this.resampleThreshold);

          % The indices of the particles that are to be resampled
          resampledIndices = this.resampler(weights);

          % The particles after resampling
          particles = particles(resampledIndices,:);

          % Assign the importance weights to be uniform for the resampled
          % particles
          weights = ones(size(weights, 1), 1)/this.numberParticles;
        end
      end
    end
  end
end
```

Appendix E

MATLAB Auxiliary Filter Implementation

```
classdef AuxiliaryParticleFilter < GenericParticleFilter
  %AUXILIARYPARTICLEFILTER Auxiliary Particle Filter
  %   Auxiliary Particle Filter

  % METHODS (Public)
  methods(Access = public)

    % Function: Auxiliary Particle Filter
    % Input:    Number of Particles (Integer),
    %           Resampling Scheme Function Handle (Function Handle), Resampling
    %           Threshold (Integer)
    % Output:   None
    % NOTES:
    function this = AuxiliaryParticleFilter(NumberParticles, ResamplingScheme,
      ResamplingThreshold)
      % AuxiliaryParticleFilter. The default Constructor.

      this = this@GenericParticleFilter(NumberParticles, ResamplingScheme, ...
      ResamplingThreshold);

      this.name = 'Auxiliary Particle Filter';
    end

    % Function: Run
    % Input:    NumberTimeSteps (Integer)
    % Output:   Particles (Vector), Weights (Vector)
    % NOTES:
    function [mmse, effectiveSampleSizes] = runAlteredWeightUpdate(this, ...
              NumberTimeSteps, Data, ProposalFunctionHandle, ...
              MeasurementModelFunctionHandle, InitialParticles, InitialWeights,
          Covariance)
      % Run.

      % Ensure that there is sufficient data for the specified
      % NumberTimeSteps
```

```
      if(NumberTimeSteps > length(Data))
        NumberTimeSteps = length(Data);
      end

      % The initial particles and their importance weights
      weights = InitialWeights;
      particles = InitialParticles;

      logWeights = zeros(NumberTimeSteps, this.numberParticles);
      w2 = zeros(NumberTimeSteps, this.numberParticles);
      normalisedWeights = zeros(NumberTimeSteps, this.numberParticles);
      mmse = NaN(NumberTimeSteps, 2);
      effectiveSampleSize = NaN(1, NumberTimeSteps);

      % The constant term used in the weight update equation
      constantTerm = (-1 * log(2 * pi) -0.5 * ...
      log(det(Covariance))).*ones(1, this.numberParticles);

      % Iterate through the each time step
      for t = 1:NumberTimeSteps

        % The current data point
        currentDataPoint = Data(t, :);
        currentObs = repmat(currentDataPoint, this.numberParticles, 1);

        % Generate the new states from the systemModelFunctionHandle
        particlesPrior = ProposalFunctionHandle(particles);
        %         particles = particlesNew;

        % Generate a new set of importance weights
        if(t == 1)
          logWeights(t, :) = log(weights);
        else
          %           currentObs = repmat(currentDataPoint,
%this.numberParticles, 1);
          % NOTE: Dimension/2
          % This only for diagonal covariance
          logWeights(t,:) =  logWeights(t-1,:) - constantTerm ...
          - 0.5 * (((currentObs(:, 1) - particlesPrior(:, 1)).^2)./(2 *
      (Covariance(1)^2)))' ...
          - 0.5 * (((currentObs(:, 2) - particlesPrior(:, 2)).^2)./(2 *
      (Covariance(end)^2)))';
        end

        maxLogWeight = max(logWeights(t, :));

        % NOTE:
        % 1. correct only up to a multiplicative factor for unnormalized
        % weights
        w2(t, :) = exp(logWeights(t, :) - maxLogWeight);

        normalisedWeights(t, :) = w2(t, :)./sum(w2(t, :));

        % The effective sample size
        effectiveSampleSize(t) = 1/sum(normalisedWeights(t, :).^2);

        % Resample if the effectiveSampleSize is less than the
        % this.resampleThreshold
        if (round(effectiveSampleSize(1,t)) < this.resampleThreshold)
```

```
          % The indices of the particles that are to be resampled
          resampledIndices = this.resampler(normalisedWeights(t,:));
        else
          resampledIndices = 1:length(particles);
        end

        % New particles based on the resampledIndices
        particles = ProposalFunctionHandle(particles(resampledIndices, :));

        % Assign the importance weights
        numerator = constantTerm ...
        - 0.5 * (((currentObs(:, 1) - particles(:, 1)).^2)./(2 *
    (Covariance(1)^2)))' ...
        - 0.5 * (((currentObs(:, 2) - particles(:, 2)).^2)./(2 *
    (Covariance(end)^2)))';
        denominator = constantTerm ...
        - 0.5 * (((currentObs(:, 1) - particlesPrior(resampledIndices,
1)).^2)./(2
    * (Covariance(1)^2)))' ...
        - 0.5 * (((currentObs(:, 2) - particlesPrior(resampledIndices,
2)).^2)./(2
    * (Covariance(end)^2)))';

        w2(t, :) = numerator./denominator;

        % Normalise the importance weights
        normalisedWeights(t, :) = w2(t, :)./sum(w2(t, :));

        % Store the MMSE
        mmse(t,:) = particles' * normalisedWeights(t, :)';
      end

      effectiveSampleSizes = effectiveSampleSize;
    end

    % Function: Run
    % Input:    NumberTimeSteps (Integer)
    % Output:   Particles (Vector), Weights (Vector)
    % NOTES:
    function [particles, weights] = run(this, NumberTimeSteps)
      % Run.

      % Ensure that there is sufficient data for the specified
      % NumberTimeSteps
      if(NumberTimeSteps > length(this.data))
      NumberTimeSteps = this.data;
      end

      % The initial particles and their importance weights
      weights = this.initialImportanceWeights;
      particles = this.initialParticles;

      % Iterate through the each time step
      for i = 1:NumberTimeSteps
        fprintf('Time = %i\n', i);

        % The current data point
        currentDataPoint = this.data(i);
```

```
        % Generate the new states from the systemModelFunctionHandle
        particlesPrior = this.systemModelFunctionHandle(particles);

        % Generate a new set of importance weights from the
        % measurementModelFunctionHandle (the likelihood)
        weightsNew = this.measurementModelFunctionHandle(particlesPrior,
    currentDataPoint);

        % Update the current importance weights by multiplying it by the
        % likelihood (weightsNew)
        weights = weights .* weightsNew;

        % Normalise the importance weights
        weights = weights / sum(weights);

        % Calculate the effective sample size
        effectiveSampleSize =
GenericParticleFilter.EffectiveSampleSize(weights);

        % Resample if the effectiveSampleSize is less than the
        % this.resampleThreshold
        if (effectiveSampleSize < this.resampleThreshold)
          fprintf('Resampling (ESS: %d, Threshold: %d)\n', effectiveSampleSize,
%
  %...
          this.resampleThreshold);

          % The indices of the particles that are to be resampled
          resampledIndices = this.resampler(weights);
        end

        % New particles based on the resampledIndices
        particles =
this.systemModelFunctionHandle(particles(resampledIndices));

        % Assign the importance weights
        numerator = this.measurementModelFunctionHandle(...
        particles, currentDataPoint);
        denominator = this.measurementModelFunctionHandle(...
        particlesPrior(resampledIndices, :), currentDataPoint);
        weights = numerator./denominator;

        % Normalise the importance weights
        weights = weights / sum(weights);
      end

      % Store the final set of particles and their associated importance
%weights
      this.finalParticles = particles;
      this.finalImportanceWeights = weights;
    end
  end
end
```

Appendix F

MATLAB Resampling Schemes Implementation

```
classdef ResamplingScheme < handle
  % RESAMPLINGSCHEME
  % Resampling Schemes

  % METHODS (Static)
  % TODO:
  % 1.  All of the Resampling Schemes should check the input Weights to
  %     ensure that they are already normalised!
  % 2.  Utilise the overload the Multinomial function, since the CDF component
  %     is used by all other schemes
  methods(Static)

    % Function: Multinomial
    % Input:    Weights (Vector)
    % Output:   Selected Indices (Vector)
    % NOTES:
    function results = Multinomial(Weights, StartIndex)
      % Multinomial.
      %       fprintf('ResamplingScheme.Multinomial\n');

      if(nargin == 2 && StartIndex > 1)
        startIndex = StartIndex;
      else
        startIndex = 1;
      end

      numberWeights = length(Weights);

      % The CDF of the Weights
      empiricalCDF = cumsum(Weights);

      % Initialise the results data structure
      results = NaN(numberWeights, 1);

      % Iterate through the Weights
```

```
      for i = startIndex:numberWeights
        % Find the first weight that is greater than the randomly generated
        % number
        randNum = rand;
        index = find(empiricalCDF > randNum, 1);

        results(i) = index;
      end
    end

    % Function: Residual
    % Input:    Weights (Vector)
    % Output:   Selected Indices (Vector)
    % NOTES:
    function results = Residual(Weights)
      % Residual.
      %       fprintf('ResamplingScheme.Residual\n');

      numberWeights = length(Weights);

      % Initialise the results data structure
      results = NaN(numberWeights, 1);

      % The counts of each index that will be in the results
      tmp = numberWeights .* Weights;
      indexReplicationCount = floor(tmp);

      % The number of indices that will be replicated in the results
      residualCount = sum(indexReplicationCount);

      % The number of indices that will be drawn using the
      % ResamplingScheme.Multinomial method
      numberMultinomialResampledWeights = numberWeights - residualCount;

      % Modify the Weights
      Weights = (tmp -
indexReplicationCount)/numberMultinomialResampledWeights;

      % Replicate the Weights according to the replcationCount
      indicesToReplicate = find(indexReplicationCount > 0);

      % The number of times each index should be replicated
      indexReplicationCount = indexReplicationCount(indicesToReplicate);

      % Replicate the indicesToReplicate indexReplicationCount times in the
      % results
      C = arrayfun(@(x, y) repmat(x, [1 y]), indicesToReplicate, ...
      indexReplicationCount, 'UniformOutput', false);
      results(1:residualCount, :) = cell2mat(C);

      % Draw the remaining Weights using the ResamplingScheme.Multinomial
%method
      multinomialResults = ResamplingScheme.Multinomial(Weights, residualCount
+ 1);
      multinomialSelectedIndices = find(multinomialResults > 0);

      % Add the results from the
      results((residualCount + 1):end) = ...
      multinomialResults(multinomialSelectedIndices);
```

```
    end

    % Function: Stratified
    % Input:    Weights (Vector)
    % Output:   Selected Indices (Vector)
    % NOTES:
    function results = Stratified(Weights)
      % Stratified.
      %       fprintf('ResamplingScheme.Stratified\n');

      numberWeights = length(Weights);

      % Initialise the results data structure
      results = NaN(numberWeights, 1);

      % The CDF of the Weights
      empiricalCDF = cumsum(Weights);

      % Generate thresholds
      thresholds = NaN(numberWeights, 1);
      for i = 1:numberWeights,
        randNum = rand;
        thresholds(i) = (randNum + (i-1))/numberWeights;
      end

      %
      for i = 1:numberWeights
        % The first index of the empiricalCDF that exceeds the current
%threshold
        results(i) = find(empiricalCDF > thresholds(i), 1);
      end
    end

    % Function: Systematic
    % Input:    Weights (Vector)
    % Output:   Selected Indices (Vector)
    % NOTES:
    function results = Systematic(Weights)
      % Systematic.
      %       fprintf('ResamplingScheme.Systematic\n');

      numberWeights = length(Weights);

      % Initialise the results data structure
      results = NaN(numberWeights, 1);

      % The CDF of the Weights
      empiricalCDF = cumsum(Weights);

      % Generate thresholds
      randNum = rand;
      thresholds = linspace(0, 1 - 1/numberWeights, numberWeights) + ...
      randNum/numberWeights;

      for i = 1:numberWeights
        % The first index of the empiricalCDF that exceeds the current
%threshold
        results(i) = find(empiricalCDF > thresholds(i), 1);
      end
```

```
    end

    % Function: Get Handle
    % Input:    Option (Integer)
    % Output:   Function Handle (Function Handle)
    % NOTES:
    function value = GetHandle(Option)
      % GetHandle.

      % Return the ResamplingScheme. method corresponding to the Option
      switch Option
        case 1
          fprintf('ResamplingScheme.Multinomial\n');
          value = @(x) ResamplingScheme.Multinomial(x);
        case 2
          fprintf('ResamplingScheme.Residual\n');
          value = @(x) ResamplingScheme.Residual(x);
        case 3
          fprintf('ResamplingScheme.Systematic\n');
          value = @(x) ResamplingScheme.Systematic(x);
        case 4
          fprintf('ResamplingScheme.Stratified\n');
          value = @(x) ResamplingScheme.Stratified(x);
        otherwise
          usageString = '1 = Multinomial, 2 = Residual, 3 = Systematic, 4 =
      Stratifed';
          error('ResamplingScheme:GetHandle:InvalidOption', usageString);
      end
    end
  end
end
```

Appendix G

Cointegration Rank Estimates for FV - TU Pair

Batch Number	Estimated Rank	Frequency of Estimation	Lag
1	0	0.2593	1
1	1	0.5185	1
1	2	0.2222	1
2	0	0.1852	1
2	1	0.5926	1
2	2	0.2222	1
3	0	0.1481	1
3	1	0.6296	1
3	2	0.2222	1
4	0	0.1852	1
4	1	0.6296	1
4	2	0.1852	1
5	0	0.1111	1
5	1	0.7407	1
5	2	0.1481	1
6	0	0.1852	1
6	1	0.5926	1
6	2	0.2222	1
7	0	0.1481	1
7	1	0.5926	1
7	2	0.2593	1
8	0	0.1852	1
8	1	0.5926	1
8	2	0.2222	1
9	0	0.1852	1
9	1	0.5556	1
9	2	0.2593	1
10	0	0.2222	1
10	1	0.5926	1
10	2	0.1852	1
11	0	0.2222	1
11	1	0.5185	1
11	2	0.2593	1
12	0	0.1852	1

12	1	0.5556	1
12	2	0.2593	1
13	0	0.2222	1
13	1	0.4444	1
13	2	0.3333	1
14	0	0.2222	1
14	1	0.5185	1
14	2	0.2593	1
15	0	0.1481	1
15	1	0.5926	1
15	2	0.2593	1
16	0	0.1481	1
16	1	0.5556	1
16	2	0.2963	1
17	0	0.1481	1
17	1	0.5185	1
17	2	0.3333	1
18	0	0.1111	1
18	1	0.6667	1
18	2	0.2222	1
19	0	0.0741	1
19	1	0.7037	1
19	2	0.2222	1
20	0	0.0741	1
20	1	0.7037	1
20	2	0.2222	1
1	0	0.2593	2
1	1	0.4444	2
1	2	0.2963	2
2	0	0.2222	2
2	1	0.5185	2
2	2	0.2593	2
3	0	0.2963	2
3	1	0.4815	2
3	2	0.2222	2
4	0	0.3704	2
4	1	0.4444	2
4	2	0.1852	2
5	0	0.2222	2
5	1	0.5926	2
5	2	0.1852	2
6	0	0.2593	2
6	1	0.4815	2
6	2	0.2593	2
7	0	0.3333	2
7	1	0.4074	2
7	2	0.2593	2
8	0	0.2593	2
8	1	0.5185	2
8	2	0.2222	2
9	0	0.1852	2
9	1	0.5556	2
9	2	0.2593	2
10	0	0.2593	2
10	1	0.5185	2
10	2	0.2222	2
11	0	0.2963	2
11	1	0.4444	2
11	2	0.2593	2

12	0	0.2593	2
12	1	0.4815	2
12	2	0.2593	2
13	0	0.1852	2
13	1	0.4815	2
13	2	0.3333	2
14	0	0.2222	2
14	1	0.5185	2
14	2	0.2593	2
15	0	0.2593	2
15	1	0.5556	2
15	2	0.1852	2
16	0	0.2222	2
16	1	0.5185	2
16	2	0.2593	2
17	0	0.2222	2
17	1	0.5185	2
17	2	0.2593	2
18	0	0.1852	2
18	1	0.6296	2
18	2	0.1852	2
19	0	0.1481	2
19	1	0.7037	2
19	2	0.1481	2
20	0	0.1481	2
20	1	0.6667	2
20	2	0.1852	2
1	0	0.4444	3
1	1	0.2593	3
1	2	0.2963	3
2	0	0.4074	3
2	1	0.3333	3
2	2	0.2593	3
3	0	0.3704	3
3	1	0.4444	3
3	2	0.1852	3
4	0	0.3704	3
4	1	0.4444	3
4	2	0.1852	3
5	0	0.2963	3
5	1	0.5185	3
5	2	0.1852	3
6	0	0.3333	3
6	1	0.4444	3
6	2	0.2222	3
7	0	0.4444	3
7	1	0.2963	3
7	2	0.2593	3
8	0	0.3704	3
8	1	0.4074	3
8	2	0.2222	3
9	0	0.2593	3
9	1	0.4815	3
9	2	0.2593	3
10	0	0.3333	3
10	1	0.5185	3
10	2	0.1481	3
11	0	0.2963	3
11	1	0.4444	3

11	2	0.2593	3
12	0	0.2593	3
12	1	0.4815	3
12	2	0.2593	3
13	0	0.2222	3
13	1	0.4815	3
13	2	0.2963	3
14	0	0.2593	3
14	1	0.5185	3
14	2	0.2222	3
15	0	0.2593	3
15	1	0.5185	3
15	2	0.2222	3
16	0	0.2963	3
16	1	0.5185	3
16	2	0.1852	3
17	0	0.2593	3
17	1	0.5185	3
17	2	0.2222	3
18	0	0.2963	3
18	1	0.5185	3
18	2	0.1852	3
19	0	0.2593	3
19	1	0.5926	3
19	2	0.1481	3
20	0	0.2593	3
20	1	0.5556	3
20	2	0.1852	3
1	0	0.4815	4
1	1	0.2222	4
1	2	0.2963	4
2	0	0.4444	4
2	1	0.2963	4
2	2	0.2593	4
3	0	0.3704	4
3	1	0.4074	4
3	2	0.2222	4
4	0	0.3704	4
4	1	0.4444	4
4	2	0.1852	4
5	0	0.3333	4
5	1	0.5185	4
5	2	0.1481	4
6	0	0.3704	4
6	1	0.3704	4
6	2	0.2593	4
7	0	0.4444	4
7	1	0.2963	4
7	2	0.2593	4
8	0	0.4074	4
8	1	0.3704	4
8	2	0.2222	4
9	0	0.3704	4
9	1	0.4074	4
9	2	0.2222	4
10	0	0.3704	4
10	1	0.4815	4
10	2	0.1481	4
11	0	0.3704	4

11	1	0.4074	4
11	2	0.2222	4
12	0	0.2963	4
12	1	0.4444	4
12	2	0.2593	4
13	0	0.2593	4
13	1	0.4444	4
13	2	0.2963	4
14	0	0.2963	4
14	1	0.4815	4
14	2	0.2222	4
15	0	0.3333	4
15	1	0.4444	4
15	2	0.2222	4
16	0	0.4074	4
16	1	0.4074	4
16	2	0.1852	4
17	0	0.3704	4
17	1	0.4444	4
17	2	0.1852	4
18	0	0.3333	4
18	1	0.4815	4
18	2	0.1852	4
19	0	0.3333	4
19	1	0.5556	4
19	2	0.1111	4
20	0	0.3333	4
20	1	0.5185	4
20	2	0.1481	4

Appendix H

Cointegration Rank Estimates for AUD - CAD Pair

Batch Number	Estimated Rank	Frequency of Estimation	Lag
1	0	0.7333	1
1	1	0.1333	1
1	2	0.1333	1
2	0	0.8000	1
2	1	0.1333	1
2	2	0.0667	1
3	0	0.9000	1
3	1	0.0333	1
3	2	0.0667	1
4	0	0.8333	1
4	1	0.0667	1
4	2	0.1000	1
5	0	0.8667	1
5	1	0.0667	1
5	2	0.0667	1
6	0	0.7667	1
6	1	0.1000	1
6	2	0.1333	1
7	0	0.7667	1
7	1	0.1333	1
7	2	0.1000	1
8	0	0.8667	1
8	1	0.1000	1
8	2	0.0333	1
9	0	0.8333	1
9	1	0.1333	1
9	2	0.0333	1
10	0	0.8333	1
10	1	0.1333	1
10	2	0.0333	1
11	0	0.7667	1
11	1	0.1667	1
11	2	0.0667	1
12	0	0.7333	1

12	1	0.1667	1
12	2	0.1000	1
13	0	0.7667	1
13	1	0.1333	1
13	2	0.1000	1
14	0	0.8333	1
14	1	0.1000	1
14	2	0.0667	1
15	0	0.9000	1
15	1	0.0333	1
15	2	0.0667	1
16	0	0.8000	1
16	1	0.0667	1
16	2	0.1333	1
17	0	0.6667	1
17	1	0.1333	1
17	2	0.2000	1
18	0	0.7333	1
18	1	0.1000	1
18	2	0.1667	1
19	0	0.7667	1
19	1	0.1000	1
19	2	0.1333	1
20	0	0.8000	1
20	1	0.0667	1
20	2	0.1333	1
1	0	0.8333	2
1	1	0.1000	2
1	2	0.0667	2
2	0	0.8333	2
2	1	0.1000	2
2	2	0.0667	2
3	0	0.9000	2
3	1	0.0333	2
3	2	0.0667	2
4	0	0.8667	2
4	1	0.0333	2
4	2	0.1000	2
5	0	0.8667	2
5	1	0.0667	2
5	2	0.0667	2
6	0	0.8333	2
6	1	0.0667	2
6	2	0.1000	2
7	0	0.9000	2
7	1	0.0333	2
7	2	0.0667	2
8	0	0.9333	2
8	1	0.0333	2
8	2	0.0333	2
9	0	0.8667	2
9	1	0.1000	2
9	2	0.0333	2
10	0	0.9333	2
10	1	0.0667	2
10	2	0.0000	2
11	0	0.8333	2
11	1	0.0667	2
11	2	0.1000	2

12	0	0.7667	2
12	1	0.1000	2
12	2	0.1333	2
13	0	0.7667	2
13	1	0.1000	2
13	2	0.1333	2
14	0	0.8667	2
14	1	0.0667	2
14	2	0.0667	2
15	0	0.9000	2
15	1	0.0333	2
15	2	0.0667	2
16	0	0.8667	2
16	1	0.0333	2
16	2	0.1000	2
17	0	0.8000	2
17	1	0.0333	2
17	2	0.1667	2
18	0	0.8000	2
18	1	0.0667	2
18	2	0.1333	2
19	0	0.9000	2
19	1	0.0333	2
19	2	0.0667	2
20	0	0.8000	2
20	1	0.0667	2
20	2	0.1333	2
1	0	0.8000	3
1	1	0.1333	3
1	2	0.0667	3
2	0	0.8333	3
2	1	0.1000	3
2	2	0.0667	3
3	0	0.9667	3
3	1	0.0000	3
3	2	0.0333	3
4	0	0.9000	3
4	1	0.0333	3
4	2	0.0667	3
5	0	0.9000	3
5	1	0.0333	3
5	2	0.0667	3
6	0	0.8667	3
6	1	0.0667	3
6	2	0.0667	3
7	0	0.9000	3
7	1	0.0333	3
7	2	0.0667	3
8	0	0.9333	3
8	1	0.0333	3
8	2	0.0333	3
9	0	0.9333	3
9	1	0.0333	3
9	2	0.0333	3
10	0	0.9333	3
10	1	0.0667	3
10	2	0.0000	3
11	0	0.8333	3
11	1	0.0333	3

11	2	0.1333	3
12	0	0.8000	3
12	1	0.1000	3
12	2	0.1000	3
13	0	0.8000	3
13	1	0.0667	3
13	2	0.1333	3
14	0	0.8667	3
14	1	0.0667	3
14	2	0.0667	3
15	0	0.9000	3
15	1	0.0333	3
15	2	0.0667	3
16	0	0.8667	3
16	1	0.0333	3
16	2	0.1000	3
17	0	0.8333	3
17	1	0.0000	3
17	2	0.1667	3
18	0	0.8333	3
18	1	0.0667	3
18	2	0.1000	3
19	0	0.9000	3
19	1	0.0333	3
19	2	0.0667	3
20	0	0.8000	3
20	1	0.0667	3
20	2	0.1333	3
1	0	0.7667	4
1	1	0.1333	4
1	2	0.1000	4
2	0	0.8333	4
2	1	0.1000	4
2	2	0.0667	4
3	0	0.9667	4
3	1	0.0000	4
3	2	0.0333	4
4	0	0.9333	4
4	1	0.0333	4
4	2	0.0333	4
5	0	0.9333	4
5	1	0.0000	4
5	2	0.0667	4
6	0	0.9000	4
6	1	0.0333	4
6	2	0.0667	4
7	0	0.9000	4
7	1	0.0333	4
7	2	0.0667	4
8	0	0.9333	4
8	1	0.0333	4
8	2	0.0333	4
9	0	0.9333	4
9	1	0.0333	4
9	2	0.0333	4
10	0	0.9333	4
10	1	0.0667	4
10	2	0.0000	4
11	0	0.8333	4

11	1	0.0667	4
11	2	0.1000	4
12	0	0.8000	4
12	1	0.1000	4
12	2	0.1000	4
13	0	0.8333	4
13	1	0.0667	4
13	2	0.1000	4
14	0	0.8667	4
14	1	0.0667	4
14	2	0.0667	4
15	0	0.9000	4
15	1	0.0333	4
15	2	0.0667	4
16	0	0.8667	4
16	1	0.0333	4
16	2	0.1000	4
17	0	0.8333	4
17	1	0.0000	4
17	2	0.1667	4
18	0	0.8333	4
18	1	0.0667	4
18	2	0.1000	4
19	0	0.9000	4
19	1	0.0333	4
19	2	0.0667	4
20	0	0.8000	4
20	1	0.0667	4
20	2	0.1333	4

Appendix I

Cointegration Rank Estimates for NQ - AUD Pair

Batch Number	Estimated Rank	Frequency of Estimation	Lag
1	0	0.8095	1
1	1	0.1905	1
1	2	0.0000	1
2	0	0.9048	1
2	1	0.0476	1
2	2	0.0476	1
3	0	0.9524	1
3	1	0.0000	1
3	2	0.0476	1
4	0	0.9048	1
4	1	0.0000	1
4	2	0.0952	1
5	0	0.9524	1
5	1	0.0000	1
5	2	0.0476	1
6	0	0.9524	1
6	1	0.0000	1
6	2	0.0476	1
7	0	1.0000	1
7	1	0.0000	1
7	2	0.0000	1
8	0	0.9048	1
8	1	0.0000	1
8	2	0.0952	1
9	0	0.9048	1
9	1	0.0000	1
9	2	0.0952	1
10	0	0.9524	1
10	1	0.0000	1
10	2	0.0476	1
11	0	0.8571	1
11	1	0.0000	1
11	2	0.1429	1

12	0	0.8095	1
12	1	0.0000	1
12	2	0.1905	1
13	0	0.8095	1
13	1	0.0000	1
13	2	0.1905	1
14	0	0.8571	1
14	1	0.0000	1
14	2	0.1429	1
15	0	0.9000	1
15	1	0.0000	1
15	2	0.1000	1
16	0	0.8889	1
16	1	0.0556	1
16	2	0.0556	1
17	0	0.7778	1
17	1	0.1667	1
17	2	0.0556	1
18	0	0.8333	1
18	1	0.1667	1
18	2	0.0000	1
19	0	0.8333	1
19	1	0.1667	1
19	2	0.0000	1
20	0	0.9444	1
20	1	0.0556	1
20	2	0.0000	1
1	0	0.8095	2
1	1	0.1905	2
1	2	0.0000	2
2	0	0.9048	2
2	1	0.0476	2
2	2	0.0476	2
3	0	0.9524	2
3	1	0.0000	2
3	2	0.0476	2
4	0	0.8571	2
4	1	0.0476	2
4	2	0.0952	2
5	0	0.9048	2
5	1	0.0476	2
5	2	0.0476	2
6	0	0.9048	2
6	1	0.0000	2
6	2	0.0952	2
7	0	0.9524	2
7	1	0.0000	2
7	2	0.0476	2
8	0	0.9524	2
8	1	0.0000	2
8	2	0.0476	2
9	0	0.9048	2
9	1	0.0000	2
9	2	0.0952	2
10	0	0.9524	2
10	1	0.0000	2
10	2	0.0476	2
11	0	0.9048	2
11	1	0.0000	2

11	2	0.0952	2
12	0	0.8095	2
12	1	0.0000	2
12	2	0.1905	2
13	0	0.8095	2
13	1	0.0000	2
13	2	0.1905	2
14	0	0.9048	2
14	1	0.0000	2
14	2	0.0952	2
15	0	0.9500	2
15	1	0.0000	2
15	2	0.0500	2
16	0	0.8889	2
16	1	0.0556	2
16	2	0.0556	2
17	0	0.7778	2
17	1	0.1667	2
17	2	0.0556	2
18	0	0.8333	2
18	1	0.1667	2
18	2	0.0000	2
19	0	0.8333	2
19	1	0.1667	2
19	2	0.0000	2
20	0	0.9444	2
20	1	0.0556	2
20	2	0.0000	2
1	0	0.7619	3
1	1	0.1905	3
1	2	0.0476	3
2	0	0.9048	3
2	1	0.0476	3
2	2	0.0476	3
3	0	0.9524	3
3	1	0.0000	3
3	2	0.0476	3
4	0	0.8571	3
4	1	0.0476	3
4	2	0.0952	3
5	0	0.9048	3
5	1	0.0476	3
5	2	0.0476	3
6	0	0.9048	3
6	1	0.0000	3
6	2	0.0952	3
7	0	1.0000	3
7	1	0.0000	3
7	2	0.0000	3
8	0	0.9524	3
8	1	0.0000	3
8	2	0.0476	3
9	0	0.9048	3
9	1	0.0000	3
9	2	0.0952	3
10	0	0.9048	3
10	1	0.0000	3
10	2	0.0952	3
11	0	0.9048	3

11	1	0.0000	3
11	2	0.0952	3
12	0	0.8095	3
12	1	0.0000	3
12	2	0.1905	3
13	0	0.8095	3
13	1	0.0000	3
13	2	0.1905	3
14	0	0.9048	3
14	1	0.0000	3
14	2	0.0952	3
15	0	0.9000	3
15	1	0.0000	3
15	2	0.1000	3
16	0	0.8889	3
16	1	0.0556	3
16	2	0.0556	3
17	0	0.7778	3
17	1	0.1667	3
17	2	0.0556	3
18	0	0.8333	3
18	1	0.1667	3
18	2	0.0000	3
19	0	0.8333	3
19	1	0.1667	3
19	2	0.0000	3
20	0	0.9444	3
20	1	0.0556	3
20	2	0.0000	3
1	0	0.7143	4
1	1	0.1905	4
1	2	0.0952	4
2	0	0.9524	4
2	1	0.0476	4
2	2	0.0000	4
3	0	0.9524	4
3	1	0.0000	4
3	2	0.0476	4
4	0	0.8571	4
4	1	0.0476	4
4	2	0.0952	4
5	0	0.9524	4
5	1	0.0476	4
5	2	0.0000	4
6	0	0.9048	4
6	1	0.0000	4
6	2	0.0952	4
7	0	1.0000	4
7	1	0.0000	4
7	2	0.0000	4
8	0	0.9524	4
8	1	0.0000	4
8	2	0.0476	4
9	0	0.9048	4
9	1	0.0000	4
9	2	0.0952	4
10	0	0.9048	4
10	1	0.0000	4
10	2	0.0952	4

11	0	0.8571	4
11	1	0.0000	4
11	2	0.1429	4
12	0	0.7619	4
12	1	0.0000	4
12	2	0.2381	4
13	0	0.7619	4
13	1	0.0000	4
13	2	0.2381	4
14	0	0.8571	4
14	1	0.0000	4
14	2	0.1429	4
15	0	0.9000	4
15	1	0.0500	4
15	2	0.0500	4
16	0	0.8889	4
16	1	0.0556	4
16	2	0.0556	4
17	0	0.7778	4
17	1	0.1667	4
17	2	0.0556	4
18	0	0.8333	4
18	1	0.1667	4
18	2	0.0000	4
19	0	0.8333	4
19	1	0.1667	4
19	2	0.0000	4
20	0	0.9444	4
20	1	0.0556	4
20	2	0.0000	4