Universidad Carlos III de Madrid

UNIVERSIDAD CARLOS III DE MADRID

DEPARTAMENTO DE TEORÍA DE LA SEÑAL Y COMUNICACIONES

GRADO EN INGENIERÍA DE SISTEMAS AUDIOVISUALES (GISA)

TRABAJO DE FIN DE GRADO

Estudio e implementación de Algoritmos de Inferencia Bayesiana en Sistemas Espacio-Temporales

David Martín Gutiérrez

Madrid, 2016

Supervisado por:
Víctor Elvira Arregui

Índice general

Índice de figuras

Índice de tablas

Abstract

Many problems in engineering require estimation of the state of a system which changes over the time using a set of noisy measurements made on the system. In this project we focus on the state-space approach to modelling dynamic systems. First of all we study the Kalman filter algorithm which achieves the optimal solution in linear and Gaussian models. The Kalman filter minimises the variance of the estimation error. In nonlinear and/or Non-Gaussian models, approximations to the distribution of interest must be performed. We study some suboptimal algorithms such as the Monte Carlo methods and in particular, we focus on the Particle filter which is a Sequential Monte Carlo method. Over the project, several experiments in MATLAB are done with the goal of discussing and comparing the algorithms performances in several situations to demonstrate their theoretical features.

Keywords: *Kalman filter, Particle filter, Monte Carlo methods, Bayesian inference, Resampling, State-Space models, Sequential Importance Sampling.*

Capítulo 1

Introduction

1.1. Project introduction

Since the great discovery of the Kalman filter by E.R. Kalman [47], technology has made progress in several fields and, in the last years, a lot of prediction and estimation applications have been developed including filtering noisy signal, object tracking, financial engineering or navigation systems [6] [8] [9] [10] [14] [35] [40] .

The Kalman filter is known as an optimal recursive data processing algorithm. It can estimate the variables of a wide range of processes. In this project we will affirm that a Kalman filter estimates the states of a linear and/or Gaussian system model. It is an attractive algorithm because it is the only filter which minimises the variance of the estimation error, so in these lineal and Gaussian situations, it is the best solution to the estimation.

However, the Kalman filter is not always the best solution for estimations in general. In nonlinear and non-Gaussian scenarios, the Kalman filter algorithm is no longer the optimal solution. In order to try to resolve this problem, the Extended Kalman filter (EKF) was proposed as an alternative method but its solution was not sufficient [18] [34][44] .

At the end of the century, the Monte Carlo methods were introduced, especially the one called the Particle filter. The Monte Carlo methods described in this project have been developed to provide approximate solutions to resolve the inference problems of interest. In comparison with the Extended Kalman filter, the main advantage of the Monte Carlo methods is that they do not depend on any local linearization technique. The worst part is that they are computationally expensive, but today, the great development of computer systems has allowed the computational power to be increased. Since then, the Particle filter has become more popular due to its good ability to process observations represented by nonlinear and non-Gaussian models. Using this algorithm, the main aim is tracking the distributions that arise in dynamic state-space models. This tracking will consist in exploring the space of the states with randomly generated samples (particles). The distributions of interest will be approximated by the generated particles according to their assigned weight. Particle filter is based on three steps: 1) particle propagation, 2) weight computation and 3) resampling [54].

In this project we will study both Kalman filter and Monte Carlo solution for the problem estimation in different situations. Several simulations will be carried out with the software tool MATLAB in order to explain the results in a visual way and to demonstrate the features of the algorithms.

It would be recommendable to have a basic knowledge in statistics and mathematics because all of these algorithms, both Kalman filter and Particle filter, have a statistical and mathematical basis which can be complicated to understand without a minimum knowledge of these fields.

The complete structure of the project is divided into the following steps:

- Study and knowledge of the stochastic processes.
- Study and analysis of the Kalman filter algorithm.
- Development of the Kalman filter in some estimation problems.
- Study and analysis of the Monte Carlo methods.
- Development of the Particle filter algorithm.
- Documentation.

1.2. Motivation and goals

Nowadays the growth of computerization and the great advance of technology allow to collect a great quantity of data and information which needs to be explored in an effective way. Machine intelligence has allowed for the creation of complex systems which were unthinkable before, in different branches of science, technology and business. Some of these systems are used in applications which are used for weather prediction, navigation systems, object tracking or company share estimation.

Since the discovery of the Kalman filter and the Monte Carlo methods, data analysis for obtaining estimation and prediction results has encouraged the great advance of technology in several fields. Lifestyle improvement of people in many ways is the consequence of good development of these technologies. Statistics and mathematics are the basic tools to achieve the intelligent processing of information. Throughout this project, some 20th century revolutionary technological algorithms will be presented. Nowadays, these algorithms are so essential that they are subject to numerous research projects.

The main goal of this project is to provide a simplified easy-to-follow tutorial in Kalman and Particle filtering in order to understand the great advantages that these algorithms provide to many applications. Different methods which are used to make estimations and predictions of the states of dynamic systems will be explained.

This project focuses on the study of some of these algorithms which were developed in the second half of the 20th century. We will concentrate specially on Kalman filter and Monte Carlo methods, thanks to which estimation and prediction applications have been developed in several fields including telecommunications, medicine and economics. Some pseudocodes will be introduced together with their algorithm implementations. We will review the main advantages and disadvantages of the different methods.

The development of some experiments, with the posterior analysis of the final results, will be our second challenge in the project. Several tests will be done on the algorithms in order to study their performance in the face of different conditions.

1.3. Project planning

The development of the project was split up in several phases in order to make the process simpler and thus achieve the main project goals. In Tabla 1.1, we present the project calendar with the tracking of each item of the development of the project and a little description of the different activities.

Item Number	Item name	Item date	Item description
1	Documentation and study	8th-29th of February	Reading and researching several papers in order to understand the point of the subject of the project
2	1D Kalman filter development	1st – 13th of March	Developments of 1D Kalman filter in MATLAB.
3	1D Kalman filter testing	14th- 21th of March	Period of testing the 1D Kalman filter simulation.
4	2D Kalman filter development	22th-31th of March	Developments of 2D Kalman filter in MATLAB.
5	2D Kalman filter testing	4th-11th of April	Period of testing the 2D Kalman filter simulation.
6	Result Analysis	14th-17th of April	Analyses of the Kalman filter simulations.
7	Documentation about Particle filtering	20th of April -28th of May	Reading and researching several papers in order to understand the point of the Particle filtering.
8	Particle filter development	3th-12th of May	Development of a Particle filter in MATLAB.
9	Particle filter testing	13th-20th of May	Period of testing the Particle filter simulation.
10	Experiments for the 1D and 2D Kalman filter simulation	21th-24th of May	Experimental processing of both 1D and 2D Kalman filter simulations in different environments.
11	Experiments for the Particle filter simulation	25th -29th of May	Experimental processing of Particle filtering simulations in different environments.
12	Result Analysis	20th-22th of June	Analyses of the Particle filter simulations.
13	Basic Latex Course	24th of June-7th of July	Learning the basic functions about Latex Software.
14	Tutorial working	8th of July– 13th of September	Development of the project tutorial.

Tabla 1.1: *Planning Calendar for project development.*

1.4. Software tools

In this project, two software tools have been employed during the development of the same: LAtex and MATLAB. LAtex is a free software tool which is used in document layout and in document edition primarily in scientific research. Thanks to their numerous scientific formulation functions, LAtex provides a simple and efficient way for documents to be written.

MATLAB is a non-free software tool which is used in several fields including: data analyzing and processing, digital signal processing, mathematics, machine learning, etc. Its ability to handle matrix, allows the efficient development and simulation of complex algorithms which, without this software, could not have been studied. Nowadays it is one of the most important software tools in the engineering and statistics fields.

1.5. Socio-economic environment

Since this project is a research tutorial about prediction and estimation methods and not a real world application analysis, this section lacks economical material on which to base its functionality.

The 21th century is destined to become the century of technological challenges. Due to the algorithms such as Kalman filter and the Sequential Monte Carlo methods, a lot of powerful applications have been developed including vehicle tracking, weather prediction, sport results prediction, image processing, etc. One of the most essential and developed technology in several fields these days is known as Big Data or massive data analysis.

The birth of Big Data has taken place because of the increase in the use of three technologies: mobile devices which are connected to Internet, electronic trade and social networks. Between 20012 and 2013, the Internet connections among mobile devices over the world rose more than 500 million, reaching the figure of 7.000 million connections. It is estimated that in the next two years, data traffic will grow by 63 % in the case of smartphones, 87 % in tablets, 30 % in lap-tops and 113 % in M2M devices (machine to machine). M2M is the exchange of data which takes place between devices such as point of sale terminals, intelligent alarms or GPS.

According to the OBS (Online Business School) study made in 2015, nowadays the Big Data technology is ruled by the 7 V's: Variety, Velocity, Volume, Variability, Veracity, Visualization and the Value which the data provides to the organization. In 2014, 73 % of global organizations invested in Big Data or they planned to do so in the next two years. According to data projects started during 2014, it rose from 8 % to 13 % [48].

In 2014 there was a prominent rise in the investments of every business sector, being Media and Communication the main investor: 53 % of the organizations of this sector invested in Big Data technology, and 33 % of the organizations of the sector had planned to do so. If the sector is analyzed in Europe: the most economic sectors with more profit thanks to Big Data were trade (47.000 million euros), industry (45.000 million), State administration (27.000 million) and Health Services(10.000 million). This study is presented by the OBSE in the web sources [38] and [48] . It is predicted that in 2020, more than 30 thousand million devices will be connected to the Internet network

A lot of companies are using Big Data because it is a great ally in the improvement of client experience as well as in the improvement of business process efficiency. Thanks to Big Data technology, the business world is going to change. it will be possible to advance in several fields which will improve different aspects of daily life in society including economy, communications and health services.

1.6. Regulatory framework

Since this project is entirely theoretical, the regulatory framework is not of great importance, as it would have if the project were a software application. However, these algorithms are used as a basic tool in data analysis or Big Data, which is, as already explained, one of the new emerging technologies with more success and development since its beginning. Big Data is defined in the AEPD (Spanish Data Protection Agency) 2014 report as *"the huge amounts of digitalized data that are controlled by companies, public authorities and other big cnterprises, which have technology able to make an extensive analysis of this data, based on the use of algorithms."*

The current Personal Data Protection Act (LOPD)[1] regulates some aspects, but there is not a specific part about Big Data. Nevertheless, a clear limitation in data exchange does exist, in order to prevent a certain set of data being related to a certain user, so his privacy will be protected.

The Big Data regulation in data protection area will be available in the next legislation, which is being discussed among the European member states. Meanwhile, what is certain is that big companies cannot keep over time their users data and cede them to the authorities in order to preserve national security. This practice is reflected in the Data Retention Directive 2006/24/CE, which was repealed by the European Court of Justice last April [11], [19].

In the case of the developed applications belonging to telecom framework, such as navigation systems or signal processing, they must be governed by the General Telecommunications Act. Alternatively, if the final application is in the field of medicine, it should follow that field legal rules.

If we discuss the contents of this project, the Royal Legislative Decree 1/1996, issued on the 12th of April 1996, should be taken into consideration. It establishes the rights and duties of the user, as well as the necessary rules when research on already developed subjects is carried out.

1.7. Project structure

During the development of the project, the contribution of several scientific publications carried out by researchers like R.E. Kalman, Neil Gordon, Arnaud Doucet or Petar Djurić has been necessary. The basic reason for this choice is based on the fact that they did research on the algorithms that will be studied in the project.

Throughout the course of **Chapter 2**, *Procesos estocásticos*, a basic study of the principle features of these processes is done as well as some classifications according to their parameters. This chapter is used as an introduction to the following chapters because the algorithms are based on stochastic processes system estimation.

In **Chapter 3**, *El filtro de Kalman*, the entire study of the Kalman filter is explained. In this chapter we present an explanation including the different phases of the algorithm: prediction and update with an in-depth study of the equations included in each phase. Afterwards, the problem situation and the way to evaluate are demonstrated. Normally the Minimum Square Error (MSE) and its error covariance are used for the evaluation of the algorithm performance. The next step of the chapter is the development of a 1D and 2D Kalman filter in a linear and/or Gaussian situation, in order to evaluate the optimal estimation.

Finally the advantages and disadvantages that Kalman filter has for the evaluation of its performance are explained. The main disadvantage is based on the model conditions: in many engineering problems, the models are nonlinear and/or Non-Gaussian which are

[1] Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal.

difficult to estimate with Kalman filter. Because of that, in **Chapter 4**, *Métodos de Monte Carlo*, the solution for that nonlinear and/or Non-Gaussian situation is presented. In this chapter, we present different Monte Carlo methods. We focus on the Sequential Importance Sampling method which is the most common in estimation and prediction applications. There are pseudocodes for all the algorithms which are studied in the chapter with the aim of allowing other people to develop them.

Next, the problem of particle degeneracy, which negatively affects the working of these methods, is explained. Also the main solution, which consists on a new step known as resampling, is presented. Nowadays, there is some research on resampling and many different resampling techniques have been developed. We present the basic ones and their main features.

In order to evaluate the performance of the Sequential Monte Carlo methods, a Particle filter is implemented in **Chapter 5**, *Desarrollo de un filtro de partículas en MATLAB*. First of all, the Particle filter is tested in a nonlinear and Non-Gaussian model and the main idea is to evaluate the MSE, the computational cost and the number of particles that are necessary to achieve a good performance. Finally, the Particle filter is tested in a complex model discovered by Edward Lorenz. The model is used in the Chaos Theory and is three-dimensional. The goal of that experiment is to evaluate the performance of the Particle filter in a more complex situation.

In **Chapter 6**, *Final conclusions*, the main conclusions which have been obtained during the whole development of the project are explained.

Capítulo 2

Procesos estocásticos

2.1. Introducción a los procesos estocásticos

En muchas ocasiones de la vida cotidiana nos enfrentamos a procesos afectados por eventos fortuitos que inciden en los resultados que esperábamos. Dichos procesos son conocidos como procesos estocásticos. Dichos procesos se desarrollan en el tiempo y se rigen por las leyes de la probabilidad [26].

La teoría de los procesos estocásticos se centra en el estudio y modelización de sistemas de evolucionan a lo largo del tiempo, o del espacio, de acuerdo a unas leyes determinísticas, es decir, de carácter aleatorio. La forma habitual de describir cómo evoluciona un determinado sistema es mediante colecciones de variables aleatorias. De este modo, se puede estudiar la evolución de una variable aleatoria a lo largo del tiempo [27].

Definición 2.1 *Un proceso estocástico es una colección de variables aleatorias* $\{X_t \text{ con } t \in T\}$ *definidas sobre un espacio de probabilidad* $\{\Omega, A, P\}$ *[15].*

Un proceso estocástico es una colección o familia de variables aleatorias $\{X_t \text{ con } t \in T\}$, ordenadas según el subíndice t, que en general se refiere al Tiempo. Para cada instante t se tendrá una variable aleatoria diferente que vendrá representada por X_t. De esta manera, se puede interpretar un proceso estocástico como una sucesión de variables aleatorias cuyas características pueden variar a lo largo del tiempo [33].

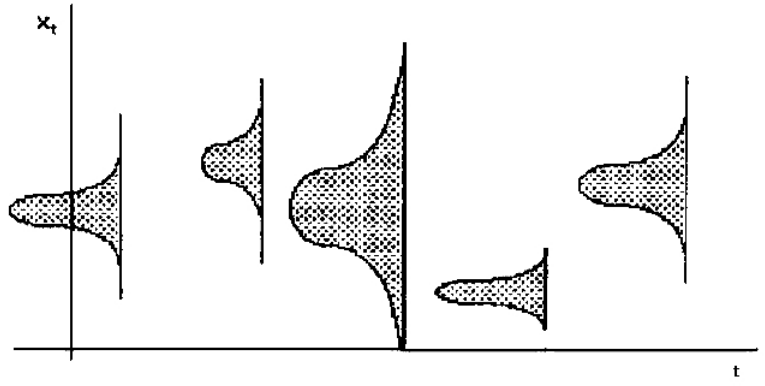


Figura 2.1: *Función de densidad de* X_t *para cada instante* t*. Figura extraída de [33].*

En la Figura 2.1 se representa para cada valor de t, la función de densidad que se corresponde a X_t. A los posibles valores que puede tomar la variable aleatoria se les denominada estados. Es posible tener un espacio de estados discretos y un espacio de estados continuo.

De esta manera es posible que la variable tiempo pueda ser de tipo discreto o de tipo continuo. En el caso del tiempo discreto, se podría dar como ejemplo valores de estado que varían cada cierto tiempo. cada mes, cada semana etc. En el caso del tiempo continuo, los cambios de los estados se podrían realizar en cualquier instante [33].

2.1.1. Trayectoria de un proceso estocástico

Un proceso estocástico puede ser visto como una función aleatoria con un doble argumento $\{X(t,w), t \in \Omega\}$. Si se fija $w = w_0$, se tendrá una realización del proceso $X(\cdot,w_0)$, cuya representación gráfica constituye lo que se denomina **Trayectoria del proceso**. Si lo que se fija es $t = t_0$, se está haciendo referencia a $X_{t_0} = X(\cdot,t_0)$ [15] .

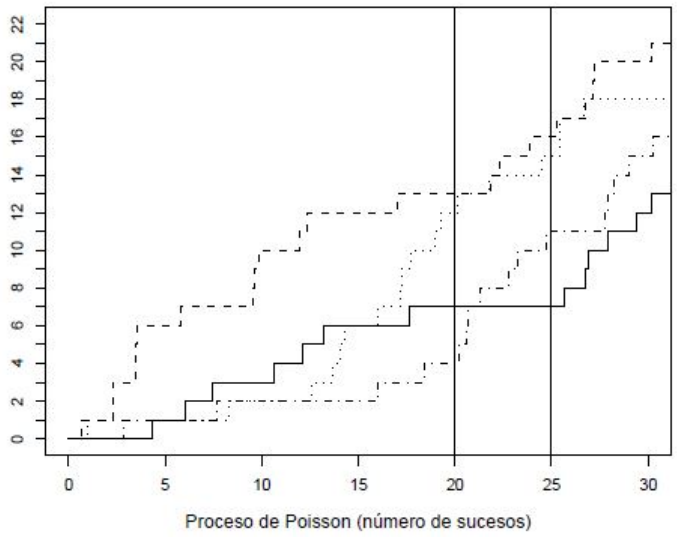


Figura 2.2: *Trayectorias de un proceso de Poisson y realizaciones de las variables N_{20} y N_{25}. Figura extraída de [15].*

En la Figura 2.2, las líneas verticales representan a las variables aleatorias N_{20} y N_{25}. Su intersección con las cuatro trayectorias del proceso, muestra los valores que han tomado dichas variables en cada realización.

2.1.2. Momentos de un proceso estocástico.

Previamente a la clasificación de los procesos estocásticos, es necesario estudiar algunas definiciones de los momentos de un proceso estocástico.

Función media: Se define como $\mu_x(t) = E[X_t]$, t ∈ T.

Para su obtención es necesario tener en cuenta el tipo de variables que forman el proceso. Para el caso discreto,

$$\mu_x(t) = \sum_{X \in D_{X_t}} xP(X_t = x), \tag{2.1}$$

donde D_{X_t} es el soporte de X_t.

En el caso continuo:

$$\mu_x(t) = \int_{-\infty}^{\infty} x f_t(x) \cdot dx \tag{2.2}$$

Función de autocorrelación: Se define a partir del momento conjunto de dos variables aleatorias asociadas a dos tiempos cualesquiera, t_1 y t_2, como

$$R(t_1, t_2) = E[X_{t_1}, X_{t_2}]. \tag{2.3}$$

Para el caso discreto se obtiene mediante la expresión:

$$R(t_1, t_2) = \sum_{X_1 \in D_{X_{t_1}},\ X_2 \in D_{X_{t_2}}} x_1 x_2 P(X_{t_1} = x_1, X_{t_2} = x_2). \tag{2.4}$$

Para el caso continuo se obtiene mediante la expresión:

$$R(t_1, t_2) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x_1 x_2 f_{t_1,t_2}(x_1, x_2) dx_1 dx_2. \tag{2.5}$$

Función de autocovarianza: Se define a partir del momento central conjunto de dos variables asociadas a dos tiempos cualesquiera, t_1 y t_2 como

$$C(t_1, t_2) = E[(X_{t_1} - \mu(t_1))(X_{t_2} - \mu(t_2))], \tag{2.6}$$

con sus correspondientes versiones discreta y continua. De esta manera se deduce la relación entre $C(t_1, t_2)$ y $R(t_1, t_2)$ como se indica a continuación,

$$C(t_1, t_2) = R(t_1, t_2) - \mu(t_1)\mu(t_2). \tag{2.7}$$

Cabe destacar el caso particular en el que $t_1 = t_2 = t$ que da como resultado la varianza $\sigma^2(t) = C(t, t)$. Para concluir, la función de correlación se obtiene mediante

$$\rho(t_1, t_2) = \frac{C(t_1, t_2)}{\sqrt{\sigma^2(t_1)\sigma^2(t_2)}}. \tag{2.8}$$

Dicha expresión verifica $|\rho(t_1, t_2)| \leq 1$. Esta función es llamada también función de autocorrelación. Es importante señalar la importancia de que dichas expresiones sean finitas para que las definiciones tengan sentido [32].

2.2. Clasificación de los procesos estocásticos.

En este trabajo se expondrán dos formas diferentes de clasificar los procesos estocásticos. La primera de ellas tendrá que ver con el tipo de variable (continua o discreta) que se dé en el conjunto de estados y en el índice T. La segunda clasificación se basa en las características probabilísticas de las variables aleatorias.

2.2.1. Clasificación de los P.E en función del índice T y del conjunto de estados.

Una primera clasificación de los procesos estocásticos se puede llevar a cabo en función de cómo sea el conjunto de subíndices T y el tipo de variable aleatoria dado por X_t. A partir de ellos se pueden establecer cuatro tipos diferentes de procesos estocásticos:

- *Discrete Time / Discrete values* (DTDV): procesos con índice numerable y variables discretas.

Estado / Tiempo	t discreto	T continuo
X discreta	Cadena	Proceso de saltos puros
X continua	Proceso de estado discreto	Proceso continuo

Tabla 2.1: *Clasificación según las estructuras del tiempo T y de los estados E.*

- *Discrete Time / Continuous values* (DTCV): procesos con índice numerable y variables continuas.
- *Continuous Time / Discrete values* (CTDV): procesos con índice no numerable y variables discretas.
- *Continuous Time / Continuous values* (CTCV): procesos con índice no numerable y variables continuas [15] .

Una **cadena** es un proceso estocástico donde el tiempo se mueve de manera discreta y la variable aleatoria sólo toma valores discretos en el espacio de estados.

Un **proceso de saltos puros**, es un proceso estocástico donde los cambios de estado ocurren de manera aislada y aleatoria, pero la variable aleatoria sólo toma valores discretos en el espacio de estados.

En un **proceso continuo**, los cambios de estado se producen en cualquier instante y hacia cualquier estado dentro de un espacio continuo de estados.

En el caso de **procesos de estado discreto**, se representa una secuencia de variables indicando el valor del proceso en instantes sucesivos de la siguiente manera :

$$X_0 = x_0,\ X_1 = x_1 \ldots\ldots X_{k-1} = x_{k-1},\ X_k = x_k, \tag{2.9}$$

en la que cada variable $X_i, i = 0, \ldots k$, tiene una distribución de probabilidades que, en general, es diferente del resto de variables aunque pudieran tener características comunes [33] .

En el caso de los procesos de **estado discreto**, el objetivo del estudio es obtener las probabilidades de ocupación de cada estado partiendo de las probabilidades de estado. Por ejemplo, si en el instante temporal $k-1$ se está en el estado x_{k-1} . ¿ Con qué probabilidad se estará en el estado x_k en el instante siguiente k? Para obtener esta probabilidad, se denota como sigue a continuación:

$$P(X_k = x_k | X_{k-1} = x_{k-1}) \tag{2.10}$$

A este tipo de probabilidad condicionada se le denomina **probabilidad de transición**. En los siguientes capítulos se hablará sobre el uso de dichas probabilidades en algoritmos de inferencia Bayesiana. A las probabilidades del tipo $P(X_k = x_k)$ se les denomina **probabilidades de estado** o **probabilidades de ocupación de estado**. Otro tipo de probabilidades de interés es la probabilidad de permanecer en un cierto estado en un instante k sabiendo que en todos los instantes anteriores, desde $k = 0$ hasta $k-1$, son conocidos los estados donde estuvo el proceso. Analíticamente se escribe de la siguiente manera:

$$P(X_k = x_k | X_0 = x_0, X_1 = x_1, \ldots\ldots, X_{k-1} = x_{k-1}) \tag{2.11}$$

Es importante observar que dicha probabilidad depende de todos los estados por los que ha pasado el proceso, mientras que en el caso de la probabilidad de transición de estados sólo depende del estado en el instante anterior [33] .

2.2.2. Clasificación de los P.E según las características probabilísticas de las v.a.

Otra posible clasificación para los procesos estocásticos, está basada en las características que puede tener la variable aleatoria a estudiar.

En la vida real se producen diferentes relaciones entre variables aleatorias constituyendo lo que hemos denominado como proceso estocástico. Cabe destacar la importancia de dichas características a la hora de poder estudiar una determinada variable aleatoria. En este caso, se van a clasificar en base a tres grupos [27] :

- Procesos estacionarios.
- Procesos Markovianos.
- Procesos de incrementos independientes.

Procesos estacionarios.

En una primera aproximación se llamarán procesos estacionarios, aquellos procesos estocásticos cuyo comportamiento sea constante a lo largo del tiempo [33]. Dentro de este grupo se hacen dos distinciones en función del nivel de estabilidad de sus momentos.

Se dice que un proceso estocástico es estacionario en *sentido débil*, si es estable en media y en autocovarianza [33] .

Se dice que un proceso estocástico es estacionario en *sentido estricto*, si las distribuciones marginales de todas las variables son idénticas, y además, la distribución es finita y sólo depende de los retardos. Es decir, si:

$$F_{t_1....t_k}(x_1,...x_k) = F_{t_1+h....t_k+h}(x_1,...x_k), \tag{2.12}$$

para cualquier $K \in T, t_1, ..., t_k, h \in \{t_i, ...t_k\} \forall i$.

Son muchos los procesos que surgen de la repetición periódica de un experimento o fenómeno aleatorio. Se puede pensar que dicha periodicidad influye en el comportamiento probabilístico del proceso. De esta manera surge la noción de **cicloestacionariedad** [15] .

Proceso cicloestacionario Se dice que un proceso X_t es cicloestacionario, si sus momentos de primer y segundo orden son periódicos con periodo T.

$$\mu_X(t) = \mu_X(t + kT)$$
$$R_X(t_1, t + \Gamma) = R_X(t_1 + kT, t + \Gamma + kT).$$

Un ejemplo de proceso cicloestacionario propuesto en [15] se muestra a continuación:

Ejemplo 2.1 *Un modem transmite señales binarias 0 y 1 IID de la siguiente manera,*

- *Para transmitir un 1, emite una señal rectangular de amplitud 1 y duración T.*
- *Para transmitir un 0, emite una señal rectangular de amplitud -1 y duración T.*

Ejemplo 2.2 *Ejemplo de un proceso estacionario. El* ruido blanco *es un claro ejemplo de proceso estacionario. Dicho proceso cuenta con las siguientes características:*

- $E(X_t) = 0$

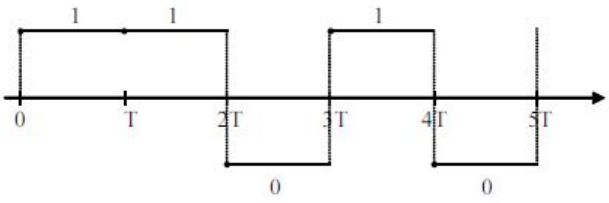


Figura 2.3: *Una trayectoria del proceso de pulsos modulados. Figura extraída de [15] para el Ejemplo 2.1.*

- $Var(X_t) = \sigma^2$
- $C(t_1, t_{1+j}) = C(t_2, t_{2+j}) = R(t_1, t_2)$.

"Se puede interpretar el proceso de ruido blanco como una sucesión de valores sin relación alguna entre ellos, oscilando en torno al cero dentro de un margen constante. En este tipo de procesos, conocer los valores pasados no proporciona ningún tipo de información sobre el futuro puesto que son procesos puramente aleatorios"[33] .

Este tipo de proceso está constituido por variables aleatorias independientes e idénticamente distribuidas (IID). Debido a esto, las distribuciones finito-dimensionales se obtienen a partir de la función de distribución común a todas las X_t del proceso,

$$F_{t_1,t_2,\ldots,t_k}(x_{t_1}, x_{t_2}, \ldots, x_{t_k}) = P(X_{t_1} \leq x_{t_1}, \ldots, X_{t_k} \leq x_{t_k}) = F(x_{t_1})F(x_{t_2})\ldots F(x_{t_k}). \tag{2.13}$$

Las funciones de los momentos tienen expresiones sencillas. En el caso de la media,

$$\mu(t) = E(X_t) = \mu, \tag{2.14}$$

siendo μ la esperanza común a todas las v.a X_t.
En el caso de la función de autocovarianza $C(t_1, t_2)$ es 0 puesto que las medias en todos los instantes son iguales. Para el caso en que $t_1 = t_2 = t$, se obtiene la varianza σ^2 [15] .

Como se ha comentado anteriormente, el ruido blanco es un caso de proceso IID. Dentro de este grupo, existen ciertos procesos de especial interés como el *proceso de Bernoulli*, el proceso Binomial o el proceso suma Gaussiano. Dichos procesos se explican con detalle en [15] [32] [33], pero en ningún caso en dicho documento.

Procesos Markovianos.

La característica principal de los procesos estocásticos markovianos es que la distribución de X_{t+1} sólo depende de la distribución anterior X_t y no de las anteriores X_{t-n} , siendo $n > 0$ [27] . Se expresa de la siguiente manera:

$$P\left(X_{t_k} \leq x_k | X_{t_1} \leq x_1, ..., X_{t_{k-1}} \leq x_{k-1}\right) = P\left(X_{t_k} \leq x_k | X_{t_{k-1}} \leq x_{k-1}\right). \tag{2.15}$$

Cuando el espacio de estados es discreto, se puede escribir como:

$$P\left(X_{t_k} = x_k | X_{t_1} = x_1, ..., X_{t_{k-1}} = x_{k-1}\right) = P(X_{t_k} = x_k | X_{t_{k-1}} = x_{k-1}) \quad \forall k \in \mathbb{k} y \forall t_1 < .... < t_k. \tag{2.16}$$

Propiedad de Markov: Se dice que un proceso cumple la propiedad de Markov si toda la historia pasada del proceso se puede resumir en la posición actual que ocupa para poder calcular la probabilidad de cambiar a otro estado [33] .

Procesos de incrementos independientes.

Se dice que un proceso es de incrementos independientes si $\forall k \in \mathbb{k}$ y $\forall t_1 < .... < t_n$ siendo $t_1 < .... < t_k$ las *v.a*

$$y_1 = X_{t_2} - X_{t_1}$$
$$y_1 = X_{t_3} - X_{t_2}$$
$$...$$
$$y_1 = X_{t_k} - X_{t_{k-1}},$$

son independientes [33].

Proposición 2.1 *Todo proceso de incrementos ortogonales es un proceso markoviano. A continuación se presenta la demostración [27] :*

Se supone un proceso X_1, X_2, X_3, y se debe demostrar que se cumple la igualdad (2.17) *para que sea un proceso markoviano.*

$$P(X_3 = x_3 | X_2 = x_2, X_1 = x_1) = P(X_3 = x_3 | X_2 = x_2). \tag{2.17}$$

Dado que se conoce $X_2 = x_2$ y $X_1 = x_1$

$$P(X_3 = x_3 | X_2 = x_2, X_1 = x_1) =$$
$$P(X_3 - X_2 = x_3 - x_2 | X_2 = x_2, X_2 - X_1 = x_2 - x_1) =$$

dado que los incrementos son independientes,

$$P(X_3 - X_2 = x_3 - x_2 | X_2 = x_2) = P(X_3 = x_3 | X_2 = x_2),$$

quedando demostrado que es un proceso markoviano.

2.2.3. Ejemplos de procesos estocásticos y simulaciones en MATLAB

A continuación se tratará de ilustrar lo explicado en las secciones anteriores a través de simulaciones de procesos estocásticos diferentes, con el objetivo de aclarar ciertos aspectos y características de los mismos.

Proceso de Poisson

El proceso de Poisson pertenece a los llamados procesos de saltos puros clasificados anteriormente. Este proceso estocástico se basa en contar el número de sucesos que ocurren a lo largo del tiempo. El tiempo entre cada par de eventos consecutivos sigue una distribución exponencial de parámetro λ, y cada uno de dichos tiempos entre llegadas se supone independiente [64] .

Dicho proceso satisface las siguientes propiedades [32] :

- Es un proceso de Markov
- Tiene incrementos independientes y estacionarios
- Para cualquier s, $t > 0$, X_{t+s} - $X_s \sim$ Poisson(λt)

- Para cualquier s, $t > 0$, y enteros $0 < i \leq j$, las probabilidades de transición son:

$$P(X_{t+s} = j | X_s = i) = \mathrm{e}^{-\lambda t} \frac{(\lambda t)^{j-i}}{(j-i)!}. \tag{2.18}$$

El proceso de Poisson se utiliza para modelar los llamados *procesos de colas*. En ellos se pueden incluir situaciones tales como saber los clientes que llegan a un banco, las peticiones que llegan a una central, etc.

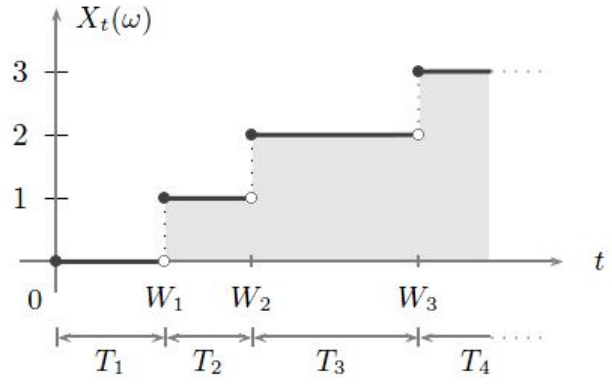


Figura 2.4: *El proceso de Poisson y los tiempos de ocurrencia de eventos. Figura extraída de [32].*

Proceso de Bernoulli

Un proceso de Bernoulli es la repetición de un determinado ensayo de Bernoulli. Debe cumplir que la probabilidad de éxito se mantiene constante ensayo tras ensayo y todos los ensayos deben de ser independientes entre sí [62] . Su función de probabilidad viene dada por:

$$X_t = \begin{cases} +1, & \textit{con probabilidad P} \\ -1, & \textit{con probabilidad (1-P)} \end{cases} \tag{2.19}$$

Cuyos momentos principales son:

$$E[X_t] = P$$
$$var[X_t] = P(1\text{-}P).$$

En la Figura 2.5 se muestra una simulación en MATLAB de un proceso de Bernoulli en 100 instantes temporales.

Ruido blanco Gaussiano.

El ruido blanco es un proceso estocástico de media cero, $\mu(t) = 0$, varianza constante, $\sigma^2(t) = \sigma^2$, y con componentes incorreladas. Debido a esto, las funciones de autocovarianza y autocorrelación son idénticas y valen,

$$R(t_1, t_2) = C(t_1, t_2) = \begin{cases} \sigma^2, & t_1 = t_2 \\ 0, & t_1 \neq t_2 \end{cases} \tag{2.20}$$

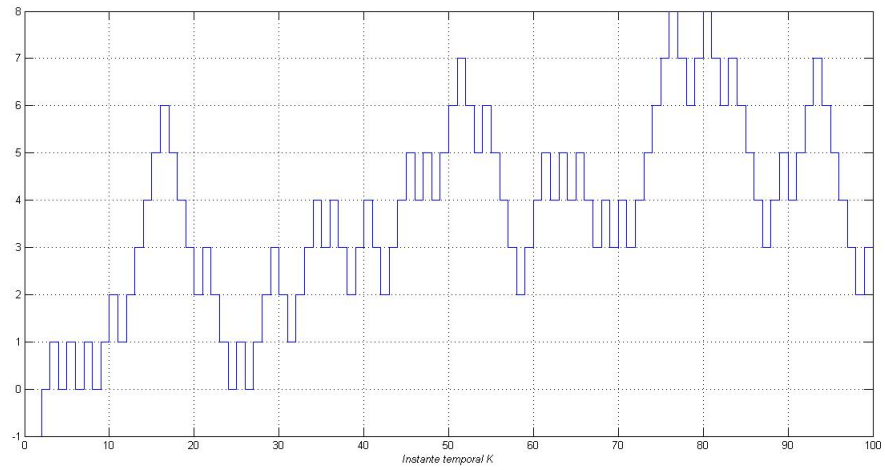


Figura 2.5: *Proceso de Bernoulli en 100 instantes temporales.*

Esto es válido tanto si se trata de una sucesión, t discreto, como si t es continuo [15]. En la Figura 2.6 se muestra una simulación en MATLAB de un ruido blanco con media cero y varianza unidad.

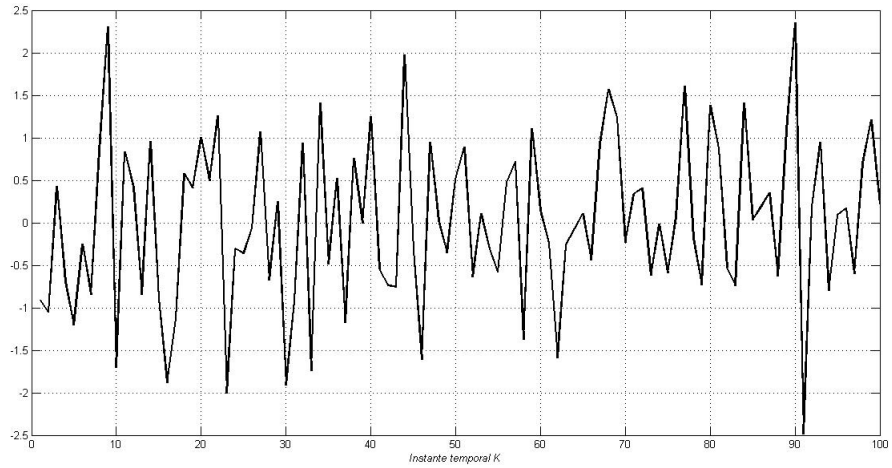


Figura 2.6: *Ruido blanco Gaussiano en 100 instantes temporales.*

Proceso autorregresivo de primer orden

Los proceso estocásticos autoregresivos (AR), son representaciones que describen procesos aleatorios que varían a lo largo del tiempo, como en finanzas, geolocalización, etc. El modelo autoregresivo especifica que la variable de salida depende *linealmente* de sus propios valores anteriores [20] .

Definición 2.2 *Un proceso AR de orden 1 viene dado:*

$$X_t = c + \varphi X_{t-1} + \varepsilon_t, \tag{2.21}$$

donde ε_t es un proceso de ruido blanco con media cero y varianza constante σ_ε^2, φ es el parámetro del modelo y c es una constante. Dicho proceso es estacionario en sentido estricto si $|\varphi| < 1$, ya que se obtiene como la salida de un filtro estable cuya entrada es un ruido blanco. Se debe prestar

atención al caso en que $\varphi = 1$, ya que en esta situación, X_t tendrá una varianza infinita y por lo tanto el proceso dejará de ser estacionario en sentido estricto. Para el caso $\varphi < 1$, la media del proceso es:

$$\mu = \frac{c}{1-\varphi}, \tag{2.22}$$

que para el caso particular $c = 0$, la media del proceso es igualmente cero. La varianza es

$$var(X_t) = \frac{\sigma_\varepsilon^2}{1-\varphi^2}. \tag{2.23}$$

Viendo la definición 2.2, se observa que dicho proceso no es más que un caso especial del modelo autorregresivo de media móvil (*ARMA*) de series temporales [60] .

En la Figura 2.7 se presenta una simulación en MATLAB para un proceso autorregresivo de orden 1 que sigue el siguiente modelo:

$$X_t = \alpha X_{t-1} + \varepsilon_t$$

Siendo α el parámetro de dicho modelo con valor 0.8 y ε_t un proceso de ruido blanco con media cero y varianza unidad. La simulación se ha realizado para 150 instantes temporales.

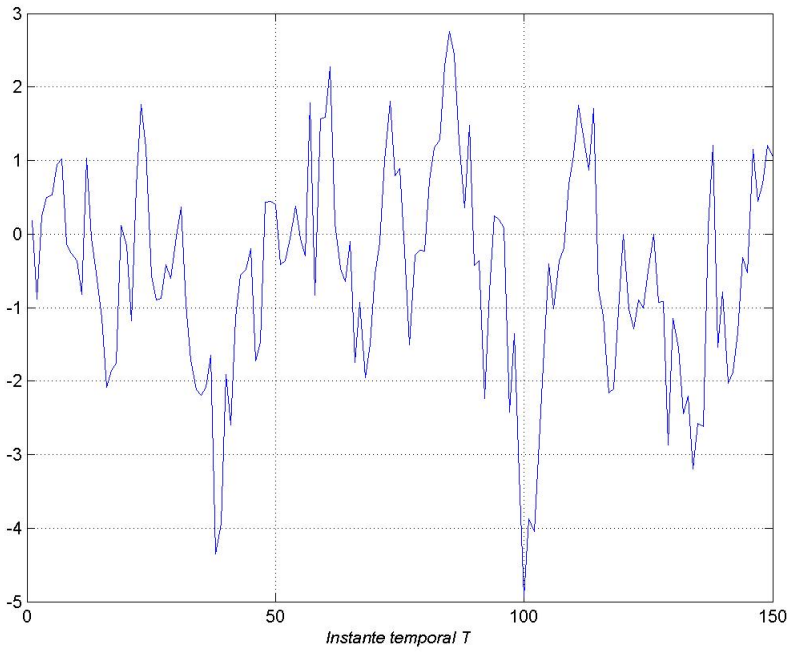


Figura 2.7: *Simulación en MATLAB de un modelo de Autorregresión de orden 1.*

2.3. Modelos espacio temporales

Previamente al estudio de los modelos espacio temporales, es necesario definir un sistema dinámico para poder entender el resto de la sección.

Un sistema dinámico es un sistema cuyo estado evoluciona con el tiempo. Los sistemas físicos en situación no estacionaria son ejemplos de sistemas dinámicos, pero también existen modelos económicos, matemáticos que también lo son. Para caracterizar el comportamiento en dicho estado se determinan los límites del sistema, los elementos y sus relaciones; de tal manera que se puedan elaborar modelos que representen la estructura del sistema [50] [61].

Muchos sistemas dinámicos pueden ser elaborados en lo que se conoce como modelado espacio temporal, para hacer más sencillo su análisis [24] .

Los modelos espacio temporales son modelos matemáticos presentes en sistemas físicos formados por un conjunto de variables de entrada, salida y de estado, relacionadas a través de un sistema de ecuaciones diferenciales de primer orden. El término *espacio-temporal* hace referencia al espacio en el cuál los ejes son los estados de las variables. El estado del sistema puede representarse como un vector dentro de un espacio [43].

Para abstraerse del número de entradas, salidas y estado, las variables se representan como vectores. Además, si el sistema dinámico es lineal, invariante en el tiempo y de dimensión finita, las ecuaciones del sistema se pueden expresar como matrices. En función del tipo de sistema, diferenciaremos entre sistemas continuos e invariantes en el tiempo, sistemas continuos y variantes en el tiempo, sistemas discretos e invariantes en el tiempo y sistemas discretos variantes en el tiempo [65]. En este proyecto nos centraremos en los sistemas discretos e invariantes con el tiempo .

La representación espacio-temporal otorga una manera robusta y eficiente de modelar y analizar sistemas con múltiples entradas y salidas [65].

Los modelos espacio temporales permiten modelar (múltiples) series temporales observadas, $\{y_k\}_{k=1}^{K}$, que guardan relación con el vector de los estados del sistema $\{x_k\}_{k=1}^{K}$, que en la mayor parte de las situaciones no se observa. Ambos conjuntos vienen perturbados por un proceso estocástico de carácter ruidoso [43]. El sistema general sigue las ecuaciones (2.25) y (2.24).

$$y_k = g(x_k, v_k). \tag{2.24}$$

$$x_k = f(x_{k-1}, w_k), \tag{2.25}$$

La ecuación (2.25) se llama *ecuación de medida* y describe la relación entre las series temporales observadas y_k y los estados no observados x_k. En general se asume que las observaciones y_k son medidas perturbadas por un cierto ruido, que se refleja en v_k y cuya naturaleza podría ser de cualquier tipo, no necesariamente tiene que seguir una distribución Gaussiana.

La ecuación (2.24) se conoce como *ecuación de transición* y describe la evolución de los estados del sistema dinámico. Dicha expresión está también perturbada por un proceso estocástico w_k de carácter ruidoso [43]. Como se puede ver en ecuación (2.24), el estado actual del sistema únicamente depende del estado anterior y por tanto, facilita mucho el análisis del sistema dado que no depende de todos los instantes pasados [24] [43] [65].

En general, las funciones f, g así como las covarianzas de los ruidos son desconocidas y tienen que estimarse [24] [43] . Sin embargo, en este documento se van a considerar constantes y conocidas. Además los ruidos se van a asumir Gaussianos y blancos para facilitar la implementación de los experimentos.

Capítulo 3

Filtro de Kalman

3.1. Introducción

Rudolf Emil Kalman, nació en Budapest, Hungría, en 1930. Emigró a América durante la Segunda Guerra Mundial y se doctoró en el M.I.T en ingeniería eléctrica en 1954 [46].

Entre 1960 y 1961 R. E. Kalman, publicó sus primeros informes sobre un filtro predictivo y recursivo que se basaba en el uso de técnicas sobre modelos de estado espacio-temporales y en la recursividad de algoritmos con los que se revolucionó el campo de la estimación. Desde aquel momento, el llamado *Filtro de Kalman* ha sido objeto de numerosas investigaciones y aplicaciones [45] .

Gracias a las investigaciones realizadas sobre dicho filtro, se ha conseguido extender su uso a numerosas aplicaciones esenciales en campos como la biología, telecomunicaciones, sistemas de navegación, econometría, robótica, visión artificial, [4] [6] [7] [9] [17] [23] [28] [35] [45] [51] .

Durante el transcurso de este capítulo se pretende estudiar con detenimiento el algoritmo de Kalman, analizando sus fases para comprender el buen funcionamiento del mismo en diversas aplicaciones que se comentarán al final del capítulo.

El filtro de Kalman es un estimador desarrollado para resolver el problema conocido como "problema lineal-cuadrático", que básicamente, es el problema de estimación de estados instantáneos de un sistema lineal dinámico perturbado por un ruido blanco. El resultado de dicho estimador es estadísticamente óptimo respecto a cualquier otra función cuadrática de estimación de error [34] [35] .

En muchas situaciones reales, no va a ser posible o deseable medir todas las variables que se quieren controlar en el sistema que se está estudiando. Gracias al filtro de Kalman, se consigue hacer inferencia sobre la información que se necesita, a través de lo que se conocen como *observaciones*, que sirven para completar el conocimiento del sistema. Dichas observaciones pueden estar distorsionadas, de nuevo, por un ruido blanco Gaussiano [34] .

El filtro de Kalman se utiliza también para predecir la probabilidad de los procesos de un sistema dinámico que, sin dicha herramienta, serían imposibles de estimar, como los precios de las acciones de una empresa, la posición de un determinado avión etc. [34] . En la práctica, se considera uno de los grandes descubrimientos en la historia de la teoría de estimación estadística, y uno de los mayores descubrimientos del siglo XX.

Principalmente, el filtro de Kalman data de dos pasos que se ejecutan de manera recursiva:

- Predicción
- Actualización

En el primer paso, el estado es predecido a través de un modelo dinámico. En el segundo paso, se corrige la estimación con las observaciones [51]. Como se verá más adelante, *la covarianza del error del estimador se minimiza*. Es por esto por lo que se dice que el *filtro de Kalman es un estimador óptimo* [18] [22] [23] [34] [35] [44] [45] [46] [47] [51] [59] [63].

El proceso se repite de manera recursiva en cada instante temporal. La recursividad de dicho filtro hace referencia a que *no se requiere almacenar todos los datos previos* y reprocesarlos cada vez que llega nueva información.

El filtro de Kalman está basado en varios componentes: el vector de estado, el modelo dinámico, y las observaciones [51] .

3.2. Modelo espacio-temporal lineal y Gaussiano

En la sección de Modelos espacio temporales, se ha estudiado como los procesos estocásticos autorregresivos pueden formularse como modelos espacio temporales. Un modelo espacio temporal es, en principio, cualquier modelo que incluya una observación y_k y un estado x_k del proceso. Las ecuaciones pueden ser no lineales o no Gaussianas. El filtro de Kalman es un algoritmo particular que se utiliza para resolver modelos espacio temporales lineales [43].

El filtro de Kalman busca resolver el problema general de estimación del estado $x \in \mathbb{R}^n$ de un proceso que se desea controlar en tiempo discreto y que está gobernado por la ecuación diferencial estocástica y lineal:

$$x_k = Ax_{k-1} + w_{k-1}, \tag{3.1}$$

con unas observaciones $y \in \mathbb{R}^m$ dadas por

$$y_k = Hx_k + v_k. \tag{3.2}$$

Las variables aleatorias w_k y v_k representan los ruidos del proceso y de medida, respectivamente. Se asume que son independientes entre sí, blancos y con distribuciones de probabilidad

$$p(w_k) \sim N(0, Q) \tag{3.3}$$

$$p(v_k) \sim N(0, R) \tag{3.4}$$

En la práctica, las matrices de la *covarianza del ruido del proceso* Q y la *covarianza del ruido de medida* R, pueden cambiar con cada instante discreto de tiempo, pero se va a asumir que son constantes.

La matriz A de dimensiones $n \times n$ en la ecuación diferencial (3.1), relaciona linealmente el estado en el instante de tiempo discreto anterior n-1 con el estado en el instante actual n en ausencia del ruido del proceso. Destacar que, en la práctica, A puede cambiar con cada instante de tiempo, pero se va a asumir constante en este documento. La matriz H de dimensiones $m \times n$ en la ecuación de medida (3.2), relaciona los estados con las observaciones y_k. En la práctica, la matriz H varía con el tiempo y con las observaciones pero de nuevo, se va a asumir constante [18].

La limitación principal del filtro de Kalman reside en que se deben de dar ciertas condiciones para que su funcionamiento sea el óptimo. Dichas condiciones se enumeran a continuación:

a) Cualquier perturbación ruidosa debe de ser un proceso de ruido blanco y Gaussiano.

b) El modelo implementado debe encajar perfectamente con el sistema real y ser lineal.

c) Las covarianzas de los ruidos deben de ser conocidas.

Cabe destacar que los sistemas lineales y Gaussianos no siempre son posibles en la vida real. Hay diversas ocasiones en las que no existirá linealidad y/o *Gaussianidad* en el sistema y, el filtro de Kalman dejará de ser la solución óptima. En estas situaciones se deberá emplear otro tipo de filtrado que se verá en los próximos capítulos.

3.3. Algoritmo del filtro de Kalman

El filtro de Kalman es un estimador *recursivo*. Esto reduce bastante los problemas, ya que implica que, para estimar el estado actual, sólo es necesario conocer el estado anterior y la observación en el instante temporal actual. Al contrario que en las técnicas de estimación *batch*, no se necesita conocer la historia de las observaciones ni de las estimaciones [63] .

Se define $\hat{x}_k^-$ como la estimación *a priori* del estado en el instante n, dado un cierto conocimiento del proceso previo al instante k, y se denotará $\hat{x}_k \in \mathbb{R}^n$ como la estimación *a posteriori* del estado en el instante k, dadas las observaciones y_k. De esta manera, se pueden definir las estimaciones de los errores *a priori* y *a posteriori* como

$$e_k^- = x_k - \hat{x}_k^-, \tag{3.5}$$

$$e_k = x_k - \hat{x}_k. \tag{3.6}$$

De esta manera, las estimaciones *a priori* y *a posteriori* de la covarianza del error vendrán dadas por las ecuaciones (3.7) y (3.8) [18] .

$$P_k^- = E\left[e_k^- e_k^{-T}\right], \tag{3.7}$$

$$P_k = E\left[e_k e_k^T\right]. \tag{3.8}$$

El filtro de Kalman se puede resumir en dos fases: predicción y actualización. A continuación se estudiará cada una de las fases con más detalle, con ánimo de profundizar lo comentado hasta este momento.

3.3.1. Predicción

En la etapa de predicción, se utiliza la estimación del estado anterior para obtener la estimación del estado actual. Este estado predecido, es conocido como estado *a priori* debido a que, aunque se trata de una estimación del estado actual, no incluye ningún tipo de información sobre la observación del instante actual de manera directa.

Gracias a esta etapa, **se consigue una primera aproximación** tanto del estado a estimar, como de la covarianza del error cuadrático medio ó Minimum Square Error (MSE en inglés) [45] [63] . Esta etapa se rige por las ecuaciones (3.9) y (3.10).

$$\hat{x}_k^- = A\hat{x}_{k-1} + w_{k-1} \tag{3.9}$$

$$P_k^- = AP_{k-1}A^T + Q_k, \tag{3.10}$$

donde la matriz A viene de (3.1), mientras que Q_k viene de (3.3) [18]. Los valores iniciales del filtro se van a suponer conocidos en este documento.

3.3.2. Actualización

Durante el transcurso de esta etapa, se busca **minimizar el error cuadrático medio de la estimación**. El filtro de Kalman tratará de encontrar una ecuación que realice una estimación del estado a posteriori $\hat{x}_k$ como una combinación lineal del valor estimado a priori $\hat{x}_k^-$ y una diferencia ponderada entre una medida real y_k y una predicción de dicha medida $H\hat{x}_k^-$, como se muestra en la ecuación (3.11).

$$\hat{x}_k = \hat{x}_k^- + K_k\left(y_k - H\hat{x}_k^-\right), \tag{3.11}$$

$$K_k = \frac{P_k^- H^T}{HP_k^- H^T + R_k}, \tag{3.12}$$

$$P_k = (I - K_k H)P_k^-. \tag{3.13}$$

El término $\left(y_k - H\hat{x}_k^-\right)$ que aparece en (3.11), es conocido como el factor de innovación o residuo [23]. Dicho factor, refleja la discrepancia existente entre la medida predicha $H\hat{x}_k^-$ y la medida real y_k. Si dicho factor fuese cero, la predicción sería perfecta.

La ecuación (3.12) obtiene un término de gran importancia, puesto que es el elemento consecuente de que se minimice la covarianza del error a posteriori: K_k, parámetro conocido también como *ganancia de Kalman*.

Por un lado se observa que, mientras la covarianza del error de medición R_k se aproxime a cero, la ganancia de Kalman aumenta el peso al residuo más alto. Matemáticamente se puede expresar de la siguiente manera [7] [18] [34] :

$$\lim_{R_k \to 0} (K_k) = H^{-1}. \tag{3.14}$$

Por otro lado, si la covarianza del error a priori $P^-[k]$ se aproxima a cero, la ganancia de Kalman tendería a cero, dado que la estimación a priori sería buena y no se tendría que corregir [18] .

$$\lim_{P_k^- \to 0} (K_k) = 0. \tag{3.15}$$

Observación. Es importante destacar el hecho de que ninguna de las dos fases del filtro de Kalman serían capaces por sí solas de conseguir una buena estimación del estado. Si sólo se emplease la dinámica del sistema se produciría una acumulación paulatina del error y aumentaría la incertidumbre gradualmente. Si por el contrario sólo se empleara la información proveniente de los sensores, se producirían saltos bruscos en la estimación debidos al ruido de medición, incluso en situaciones en las cuáles el sistema fuera estático. Es evidente que, en caso de que no se tenga confianza en el modelo dinámico debido a un alto contenido de ruido, la decisión óptima sería ignorar el valor obtenido por la predicción, y tomar el estado obtenido por el modelo de medida como la estimación a priori del estado oculto [22].

3.3.3. Interpretación del filtro de Kalman

La justificación para (3.11) viene dada en la probabilidad de la estimación *a priori* $\hat{x}_k$ condicionado a todas las observaciones a priori y_k (Regla de Bayes). Por ello, es suficiente con señalar que, el filtro de Kalman mantiene los dos primeros momentos de la distribución

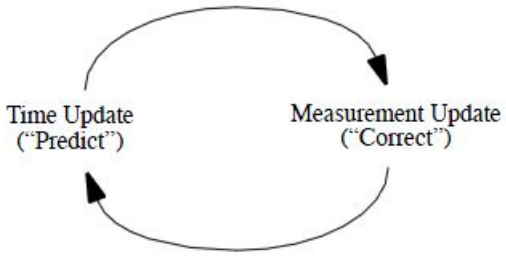


Figura 3.1: *Fases del Algoritmo del filtro de Kalman. Figura extraída de [18].*

del estado,

$$E[x_k] = \hat{x}_k, \tag{3.16}$$

$$E\left[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T\right] = P_k. \tag{3.17}$$

La estimación *a posteriori* dada por (3.11), refleja la media (momento de primer orden) de la distribución del estado, que normalmente sigue una distribución normal si se cumplen (3.3) y (3.4). La estimación de la covarianza *a posteriori* refleja la varianza de la distribución del estado (momento de segundo orden) [18]. En otras palabras,

$$p(x_k \mid y_k) \sim N\left(E[x_k]\ ,\ E\left[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T\right]\right) \tag{3.18}$$

$$= N(\hat{x}_k, P_k). \tag{3.19}$$

En la Figura 3.2 se representa todo el proceso del algoritmo discreto de Kalman, incluyendo las ecuaciones que se implementan en cada una de las fases explicadas con anterioridad.

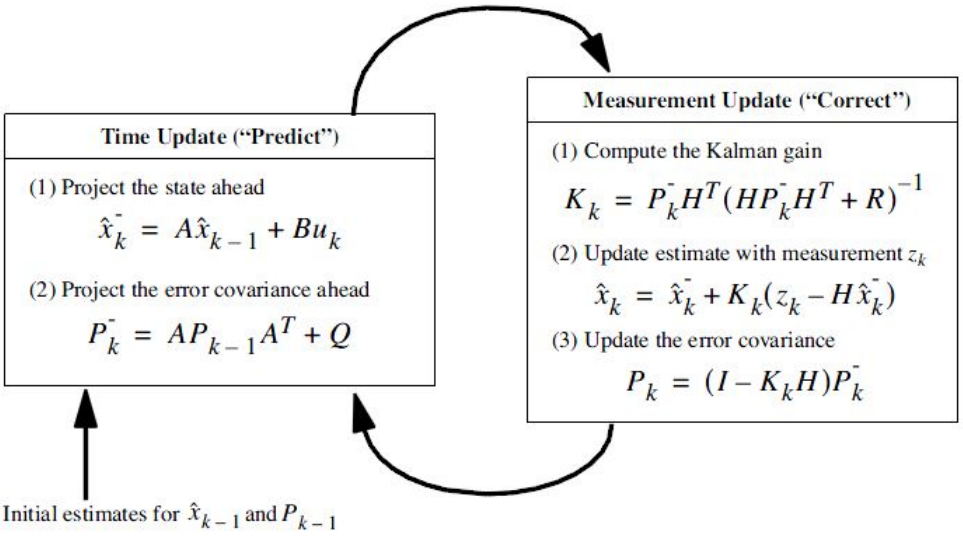


Figura 3.2: *Resumen de las principales ecuaciones del filtro de Kalman. Figura extraída de [18].*

3.4. Estimación de las covarianzas

En muchas ocasiones, es complicado realizar la implementación del filtro de Kalman debido a la dificultad que supone conseguir buenas estimaciones de las matrices de cova-

rianza de los ruidos Q_k, R_k. Diversas investigaciones han tratado de proponer estimaciones de dichas covarianzas haciendo uso de los datos. Una de las técnicas más prácticas y más prometedoras es la técnica de la **autocovarianza mínima cuadrática**, *(ALS)* en inglés [63] .

Dicha técnica esta basada en un software que realiza un conjunto de operaciones rutinarias en los datos para obtener las covarianzas del ruido. ALS fue diseñado para GNU Octave, un potente lenguaje de alto nivel utilizado en muchos campos relacionados con el tratamiento de las señales [16].

3.5. Casos de estudio del filtro de Kalman: simulaciones en MATLAB

Habiendo estudiado previamente la teoría del filtro de Kalman, se introduce ahora una sección de experimentos realizados mediante el software de MATLAB. El objetivo de dicha sección es recoger todos los aspectos vistos en la parte teórica, por medio de ejemplos, para demostrar las características del algoritmo de Kalman.

En esta sección se realizan dos experimentos sobre el filtro de Kalman: el primero de ellos está basado en la implementación de un filtro de Kalman en 1D para resolver un problema general de estimación de estados ocultos de un determinado sistema, mientras que el segundo es una implementación similar al anterior pero en en una situación donde el modelo de interés es de 2D.

Al final de la sección se discute la eficiencia y la robustez del algoritmo en base a las simulaciones realizadas de manera que se vea justificada toda la teoría descrita hasta el momento.

Experimento 3.1 *Se pretende estimar los estados ocultos de una cierta variable aleatoria discreta x_k de la que no se pueden tomar medidas. El sensor del sistema y_k nos aporta observaciones cada k instantes de tiempo discreto. Dichas observaciones tienen una relación con los estados x_k a través de* (3.21)*. El estado oculto actual x_k se relaciona a través de* (3.20) *con el estado anterior.*

$$x_k = Ax_{k-1} + w_{k-1}, \tag{3.20}$$

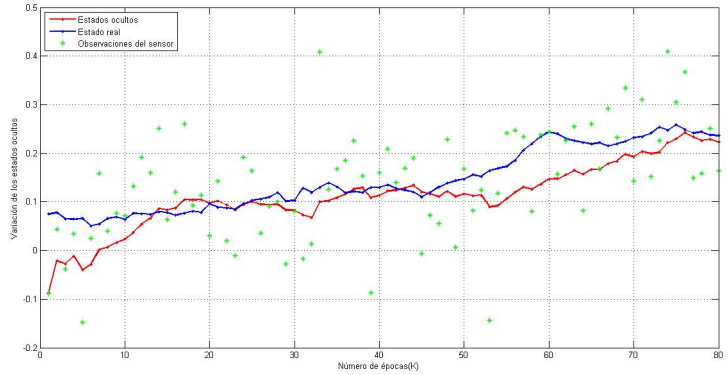
$$y_k = Hx_k + v_k. \tag{3.21}$$

Los valores de las matrices A, H y las matrices de covarianza de los ruidos Q_k , R_k se conocen. También se suponen conocidos los valores iniciales del estado de la variable aleatoria x_0 y la matriz de covarianza del error P_0. En la Tabla 3.1 se muestran los valores utilizados durante la simulación:

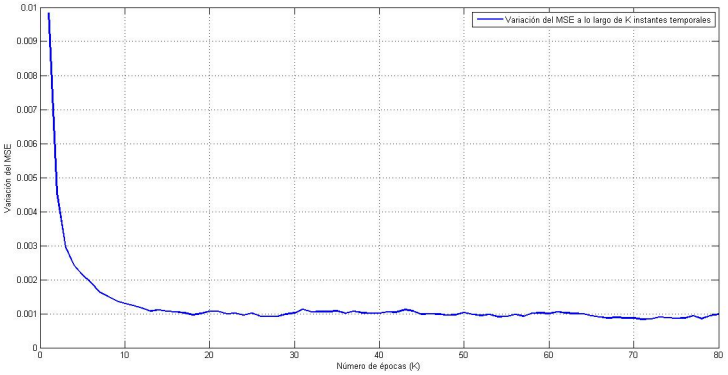
En la Figura 3.3(a) se recoge la estimación de los estados ocultos de una variable aleatoria cuando se utiliza el filtro de Kalman. En verde están representadas las observaciones, en azul los estados reales y en rojo los estados estimados por el filtro de Kalman.

Para poder demostrar que el filtro de Kalman funciona de manera óptima, hay que detenerse en las Figuras 3.3(b)-3.3(d). Al comienzo de la simulación las predicciones de los estados no son del todo buenas al no tener suficiente información de los mismos. Por ello, el filtro de Kalman tiene que corregir en gran medida dichos valores haciendo uso de la ganancia de Kalman representada en la Figura 3.3(d). En la parte teórica se habló de cómo en la etapa de corrección el filtro de Kalman minimizaba el MSE gracias a dicha ganancia.

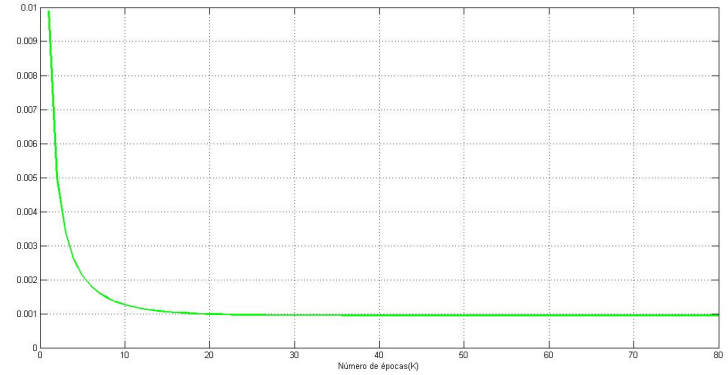
De esta manera, dado que al principio del experimento la estimación *a priori* $\hat{x}_k^-$ es mala, la estimación *a posteriori* $\hat{x}_k$ lo será también. Por consecuente, la covarianza del error *a posteriori* P_k, representada en la Figura 3.3(c), será alta y por tanto, el filtro de Kalman tendrá que corregir dicha predicción aumentando el valor de la ganancia de Kalman K_k. A medida que



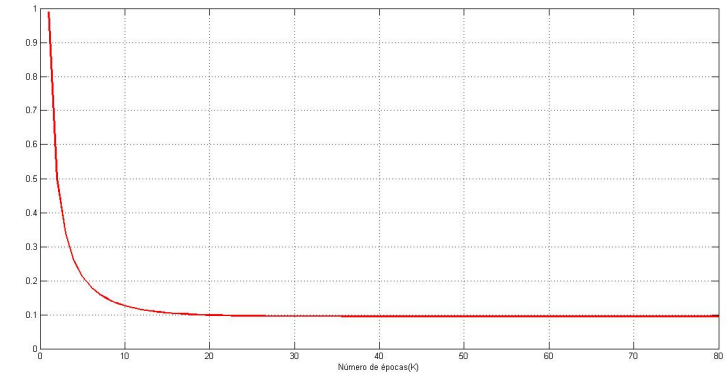
(a) Estimación del estado oculto a posteriori $\hat{x}_k$.



(b) MSE.



(c) Covarianza del error *a posteriori* P_k.



(d) Ganancia de Kalman K_k.

Figura 3.3: *Resultados gráficos recopilados durante la simulación del Experimento 3.1.*

Parámetros del algoritmo durante la simulación	Valor
Número de simulaciones	400
Número de épocas (k)	80
A	1
H	1
X_0 (media del estado inicial)	0.1
P_0 (Matriz de covarianza del estado inicial)	$\mathbb{I}$ (Matriz Identidad)
Q_k	0.0001
R_k	0.01

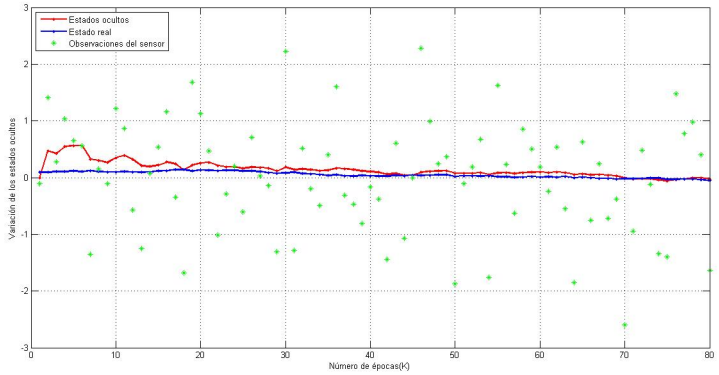
Tabla 3.1: *Parámetros empleados en la simulación de MATLAB para el Experimento 3.1.*

el filtro de Kalman recibe más información del sistema, las estimaciones *a priori* mejoran y por tanto también lo hacen las estimaciones *a posteriori*. Por ello a medida que aumenta el número de iteraciones, tanto la ganancia de Kalman K_k como la covarianza del error *a posteriori* P_k, se ven reducidas hasta que llega un punto en que se estabilizan. Es en ese punto donde se produce la minimización del error.

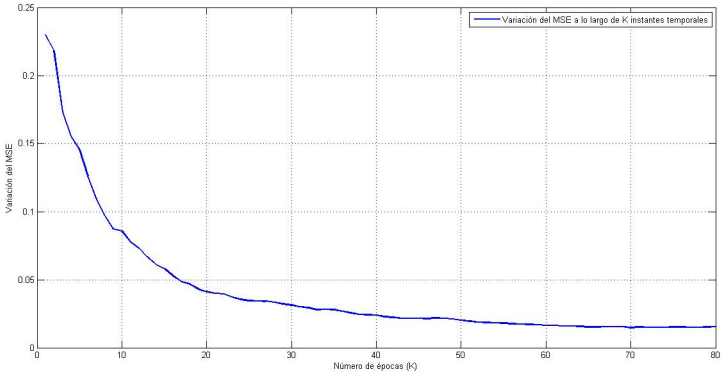
Se puede demostrar la ecuación (3.14) viendo que R_k toma un valor cercano a cero en nuestra simulación (0.01), teniendo como consecuencia que la ganancia de Kalman K_k tome el valor de H^{-1} que, en el caso de dicho experimento, es 1. Viendo la Figura 3.3(d), se visualiza el valor 1 como el valor máximo que toma la ganancia durante la simulación. Por otro lado, se puede demostrar la ecuación (3.15), observando cómo cuando la covarianza *a posteriori* del error P_k toma valores cercanos a cero en la Figura 3.3(c), la ganancia de Kalman K_k tiende a cero también, quedando demostrada la ecuación mencionada. Gracias a la ganancia de Kalman K_k, el error *a posteriori* se minimiza, logrando que el filtro de Kalman sea un algoritmo óptimo para casos lineales y/o Gaussianos.

Experimento 3.2 *A continuación, se aumenta el valor de la covarianza del error de medida R_k a 1. El resto de valores se mantienen como en el Experimento 3.1. El objetivo del experimento es ver la influencia del ruido de medida en la ejecución del filtro de Kalman y comparar las diferencias con el Experimento 3.1.*

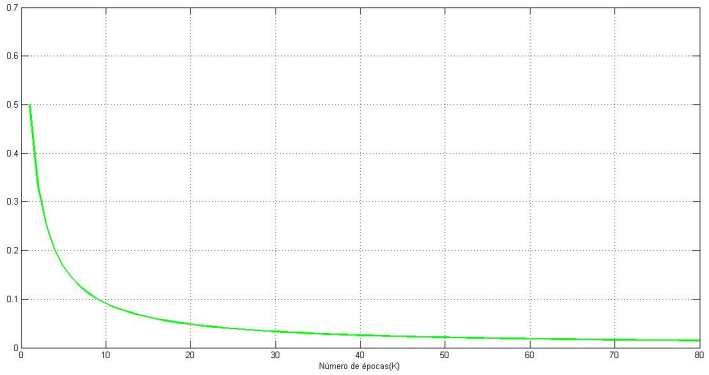
En la Figura 3.4(a), se recoge el resultado del algoritmo del filtro de Kalman tras la simulación. En verde aparecen las observaciones que, como se puede apreciar, fluctúan de manera elevada entorno a los valores reales de los estados del proceso. Esta elevada fluctuación es provocada por el aumento del ruido de medida debido a que la matriz de covarianza R se ha elevado al valor unidad.



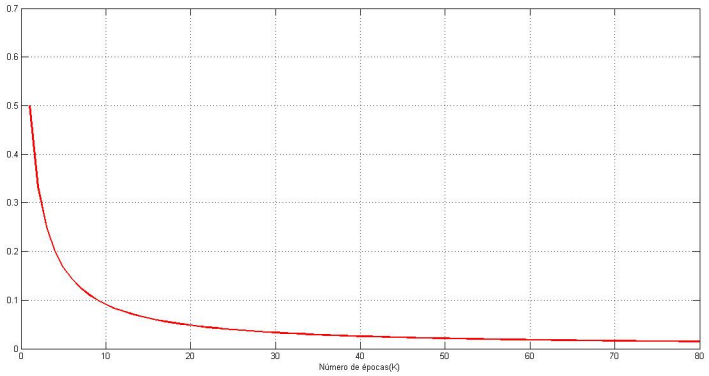
(a) Estimación del estado oculto a posteriori $\hat{x}_k$.



(b) MSE.



(c) Covarianza del error *a posteriori* P_k.



(d) Ganancia de Kalman K_k.

Figura 3.4: *Resultados gráficos recopilados durante la simulación del Experimento 3.2.*

En este caso la estimación *a posteriori* x_k no es demasiado imprecisa debido a que el ruido del proceso es muy pequeño. Si se tiene en cuenta la ecuación (3.12), un aumento de R_k hará que, por un lado, las observaciones y_k sean más ruidosas y, por otro, que la ganancia de Kalman K_k sea menor que en el Experimento 3.1. Por tanto, la estimación *a posteriori* del estado x_k será peor que en el Experimento 3.1, que se ve reflejado en que el MSE de la Figura 3.4(b), toma valores más altos que en el caso de la Figura 3.3(b).

A medida que aumenta el número de épocas k, el MSE disminuye gracias a que se minimiza debido al uso de la ganancia de Kalman, alcanzando de nuevo el objetivo de ser un algoritmo óptimo.

Experimento 3.3 *En el siguiente experimento se aumenta el valor de la covarianza del error del proceso Q_k a 1. El resto de valores se mantienen como en el Experimento 3.1. El objetivo del experimento es observar el efecto que produce aumentar Q_k en la estimación de los estados ocultos llevada a cabo por el filtro de Kalman y comparar y contrastar los resultados con el Experimento 3.1.*

Aumentando el ruido del proceso, la estimación del estado *a priori* $\hat{x}_k^-$ vendrá perturbada por dicho ruido y por tanto, la covarianza del error *a priori* P_k^- tomará valores más altos. En la etapa de actualización la ganancia k_k, influenciada por P_k^- como se aprecia en (3.12), tomará valores más altos. En concreto, dado que R_k toma el valor de 0,01, siendo cercano a cero, se vuelve a cumplir (3.14) y K_k tenderá a la unidad en el inicio de la simulación, valor que se ve reflejado en la Figura 3.5(d). Destacar que dicho valor no llega a la unidad como ocurría en el Experimento 3.1 debido al aumento del ruido del proceso Q_k.

En este caso, como K_k toma valores más altos, la covarianza del error *a posteriori* P_k aumentará con respecto al Experimento 3.2. La forma picuda que toma el MSE en la Figura 3.5(b) se debe a que el número de simulaciones no es demasiado alto[1] y el MSE no está bien promediado. Sin embargo si que se aprecia que los valores entorno a los que fluctúa, son mucho menores a los del Experimento 3.2.

Experimento 3.4 *El objetivo de este experimento es implementar un filtro de Kalman en 2D para conseguir estimar los estados ocultos de una determinada variable del sistema: $\ddot{z}_k$. Para ello se cuenta con un modelo y con unas observaciones con las que se debe lograr el objetivo. También se van a comparar y contrastar los resultados con el Experimento 3.1. El modelo dinámico vendrá dado por la siguiente expresión :*

$$\bar{x}_k = \begin{bmatrix} \dot{z}_k \\ \ddot{z}_k \end{bmatrix} = \begin{bmatrix} 1 & 0{,}1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{z}_{k-1} \\ \ddot{z}_{k-1} \end{bmatrix} + \bar{w}_{k-1}.$$

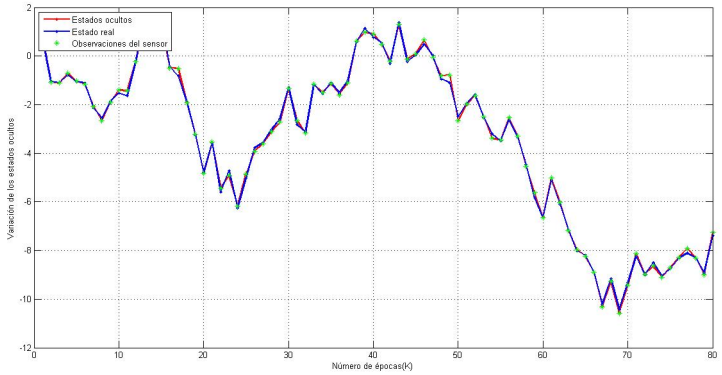
El modelo de medida u observación sigue la siguiente expresión:

$$\bar{y}_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{z}_k \\ \ddot{z}_k \end{bmatrix} + \bar{v}_k,$$

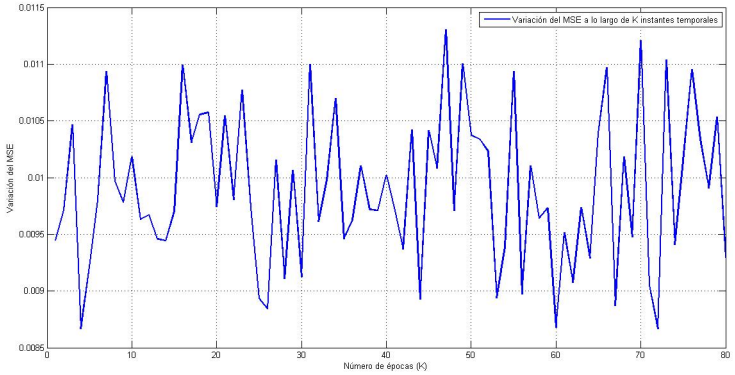
siendo $\dot{z}_k$ y $\ddot{z}_k$ variables aleatorias discretas a estimar. Viendo el modelo de observación es evidente que, sólo se toman observaciones de la variable $\dot{z}_k$. Por tanto, la variable $\ddot{z}_k$ es desconocida. Los valores iniciales de los estados ocultos y de la matriz de covarianza del error a posteriori vienen dados por:

$$\bar{x}_0 = \begin{bmatrix} 0{,}5 \\ 0{,}5 \end{bmatrix}$$

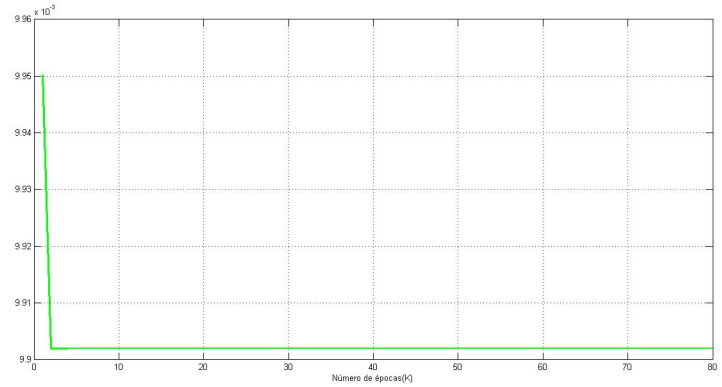
[1] Debido a las limitaciones de carga computacional de la máquina empleada en las simulaciones, el número de las mismas no ha podido aumentarse hasta conseguir el deseado, siendo por tanto, una dificultad añadida al experimento.



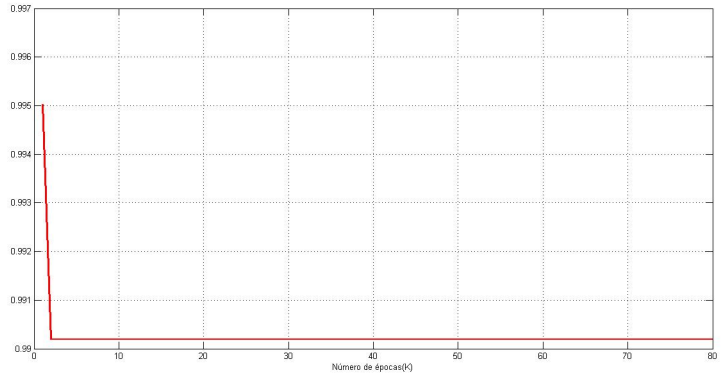
(a) Estimación del estado oculto a posteriori $\hat{x}_k$.



(b) MSE.



(c) Covarianza del error *a posteriori* P_k.



(d) Ganancia de Kalman K_k.

Figura 3.5: *Resultados gráficos recopilados durante la simulación del Experimento 3.3.*

Parámetros del algoritmo durante la simulación	Valor
Número de simulaciones	500
Número de épocas (k)	80
A	1
H	1
X_0 (media del estado inicial)	[0.5, 0.5]
P_0 (Matriz de covarianza del estado inicial)	10 x $\mathbb{I}$ (Matriz Identidad de dimensión 2)
Q_k	0.0001
R_k	0.01

Tabla 3.2: *Parámetros empleados en la simulación de MATLAB para el Experimento 3.4.*

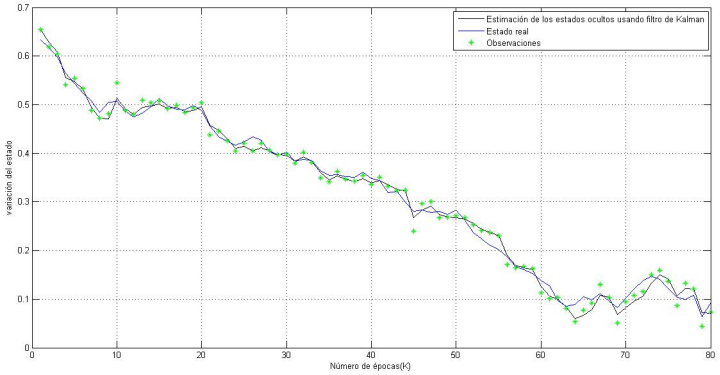
$$\bar{P}_0 = 10\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

El resto de valores utilizados en la simulación se contemplan en la Tabla 3.2.

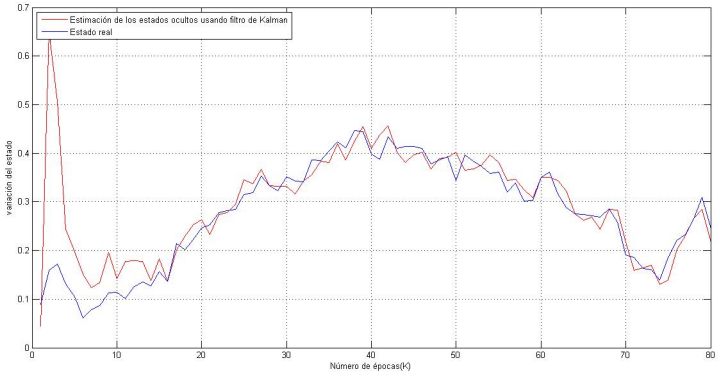
En la Figura 3.6(a) está representada la estimación del filtro de Kalman de los estados de $\dot{z}_k$, de la cuál se toman observaciones. En verde están representadas las observaciones, en azul los estados reales, y en rojo las estimaciones de los estados ocultos. Sin embargo, para $\ddot{z}_k$, el sistema no nos proporciona observaciones, pasando a ser una variable oculta que deseamos conocer. Por ello, en la Figura 3.6(b), sólo aparecen representados los estados reales y los estimados por el filtro de Kalman en azul y rojo, respectivamente.

Es evidente que el filtro de Kalman realiza una buena estimación de $\dot{z}_k$ dado que posee conocimiento sobre dicha variable y además no está perturbada prácticamente por el ruido del proceso ya que, toma valores pequeños. En la Figura 3.6(d), se representan las ganancias de Kalman aplicadas para las estimaciones de $\dot{z}_k$ y $\ddot{z}_k$, en azul y verde, respectivamente. Para $\dot{z}_k$, K_k toma valores muy pequeños dado que al tener información sobre dicha variable, no necesita hacer una corrección muy severa. Sin embargo, para el caso de $\ddot{z}_k$, K_k debe tomar valores altos en el inicio del experimento para mejorar la estimación a priori $\bar{x}_k^-$.

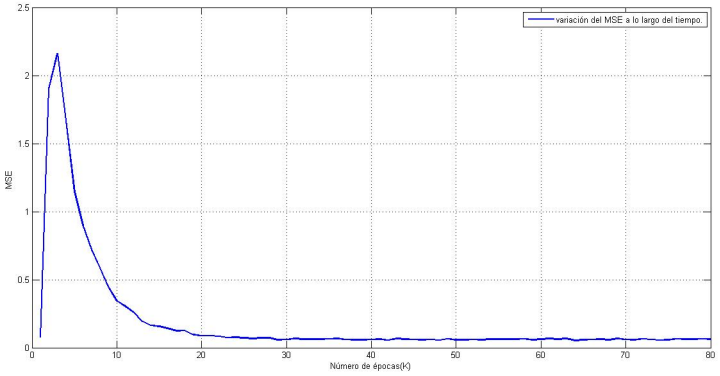
El MSE, representado en la Figura 3.6(c), marca la diferencia entre el valor real de $\ddot{z}_k$ y su valor estimado y, tomará por tanto, valores muy altos en el inicio del experimento, pero irá disminuyendo a medida que el número de épocas k avanza. De nuevo, gracias a la influencia de K_k, el MSE será mínimo y por tanto, se vuelve a demostrar que el algoritmo es óptimo, como ya ocurría en los Experimentos 3.1-3.3.



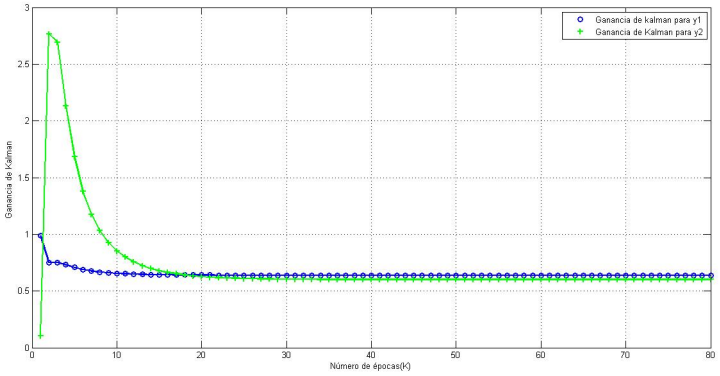
(a) Estimación del estado oculto a posteriori de $\dot{z}_k$.



(b) Estimación del estado oculto a posteriori de $\ddot{z}_k$.



(c) MSE.



(d) Ganancia de Kalman $\bar{K}_k$.

Figura 3.6: *Resultados gráficos recopilados durante la simulación del Experimento 3.4.*

3.6. Ventajas y Desventajas del Filtro de Kalman

Ventajas

El filtro de Kalman evita la influencia de posibles cambios estructurales en la estimación. La estimación recursiva se inicia con una muestra inicial y se van actualizando las estimaciones incorporando sucesivamente una observación nueva hasta cubrir la totalidad de los datos. Esto conlleva a que **la estimación más reciente de los coeficientes esté afectada por toda la historia de la serie**, lo cual, en caso de que ocurrieran cambios en el modelo, **podría sesgarla**. Este sesgo se puede corregir con las estimaciones secuenciales, pero a cambio de un mayor error estándar. De esta manera, el filtro de Kalman, así como los métodos recursivos, utiliza toda la historia de la serie pero con la ventaja de que **busca estimar una trayectoria estocástica de los coeficientes** en lugar de una determinista, eliminando el posible sesgo de la estimación ante la presencia de cambios estructurales.

El filtro de Kalman utiliza el método de mínimos cuadrados para generar recursivamente un estimador del estado al momento k, que es lineal, insesgado y de **varianza mínima**. El filtro de Kalman está en línea con el teorema de Gauss-Markov aportándole robustez al algoritmo, para poder resolver un amplio rango de problemas en inferencia estadística.

Se distingue por su habilidad para **predecir el estado** de un modelo en el pasado, presente y futuro, aún cuando la naturaleza del sistema modelado es desconocida. El modelado dinámico de un sistema es una de las características claves que distingue el método de Kalman. Los modelos lineales dinámicos son modelos con una transición lineal desde un periodo al próximo que pueden describir la gran mayoría de los modelos utilizados en trabajos de series de tiempo [7].

Desventajas

Una de las principales desventajas del filtro de Kalman es la **necesidad de conocer los valores iniciales de la media y de la varianza del vector estado** para iniciar el algoritmo recursivo. A día de hoy todavía no existe una manera óptima de hacerlo.

En un enfoque bayesiano, el filtro de Kalman requiere que se especifiquen a priori valores de los coeficientes iniciales y de sus respectivas varianzas. Una manera de hacerlo es obtener esa información estimando un modelo similar al deseado pero con coeficientes fijos para un subperiodo muestral. Por otra parte, se necesitan especificar las varianzas para lo cuál Doan, Litterman y Sims en el artículo [**?**] en 1984, sugirieron varianzas muy pequeñas y proporcionales en relación con las obtenidas para los coeficientes iniciales.

Se ha visto en secciones anteriores que el filtro de Kalman es **óptimo bajo ciertas condiciones** provocando que su uso sea muy restrictivo, como la condición de que las variables aleatorias se puedan modelar como Gaussianas. Todo ello limita su estudio y su aplicación [7].

Cuando se desarrolla para modelos autorregresivos, **los resultados se ven condicionados por la información pasada** de la variable en cuestión. El pronóstico con series de tiempo representa la fuerza o inercia que actualmente presenta el sistema, y son eficientes únicamente a corto plazo [7] .

3.7. Aplicaciones

En la actualidad, el uso de técnicas de estimación está aumentando, puesto que se ha visto que mejora en gran medida los problemas relacionados con el análisis de datos y de variables que varían a lo largo del tiempo. En el caso del filtro de Kalman, se ha extendido a

aplicaciones de **sistemas de navegación** [18] [34] [35] [45] [49] donde a partir de los sensores de velocidad de un cierto vehículo, se puede realizar un seguimiento de la posición del mismo a lo largo del tiempo. Muchos de los sistemas de navegación actuales, no sólamente utilizan el *Sistema de Posición Global* o GPS en inglés, sino también un *sistema de navegación inercial* o INS en inglés, para ayudar al conductor a encontrar su destino. Juntos, estos dos sistemas se complementan y permiten mejorar la precisión y la confianza de la navegación, especialmente cuando la señal del GPS se degrada o se interrumpe en zonas de baja cobertura. Para esta aplicación, el filtro de Kalman constituye una herramienta de corrección y predicción de la trayectoria INS gracias a las observaciones dadas por la señal GPS [45] .

También se emplea en aplicaciones relacionadas con el mundo económico como en la econometría [17], donde los usos más particulares se encuentran en la estimación de modelos ARIMA(Modelos Autorregresivos Integrados de Media Móvil), la modelación con parámetros que cambian en el tiempo y la modelación de componentes no obervables [7] .

Aplicaciones como la visión artificial, el procesamiento de señales, la navegación marina, el modelado de temas demográficos, la ciencia del tiempo y sus predicciones, la manufacturación y otras muchas más utilizan, en su base lógica, el algoritmo del filtro de Kalman [18] [23] [34] [44].

3.8. Conclusiones

El filtro de Kalman es una herramienta muy robusta y que se puede aplicar a distintas áreas, obteniendo resultados óptimos bajo ciertas condiciones. Se ha estudiado cómo, cuando el sistema es lineal y Gaussiano, el algoritmo de Kalman es capaz de minimizar el error cuadrático medio (MSE) a través de lo que se denomina en la literatura como la ganancia de Kalman. Con las dos fases del algoritmo: predicción y actualización, es capaz de predecir de manera óptima los estados ocultos de una variable perteneciente a un sistema de interés.

Debido a sus desventajas cuando el sistema deja de ser lineal, se han investigado alternativas basadas en el propio algoritmo, como el llamado *Extended Kalman Filter* en inglés, o el *Unscented Kalman Filter*, que buscan resolver el problema de estimación en modelos no lineales y/o no *Gaussianos*.

En el siguiente capítulo, se estudiarán los métodos de Monte Carlo y en particular, se empleará el método **Sequential Importance Sampling**, también conocido como filtro de partículas, para modelos no *Gaussianos* y/o no lineales.

Capítulo 4

Métodos de Monte Carlo

4.1. Motivación

El uso del filtro de Kalman en problemas de estimación es limitado dadas las condiciones de *Gaussianidad* y de linealidad de las que precisa para su correcto funcionamiento. En la mayoría de sistemas reales, la linealidad o la *Gaussianidad* no son planteables y se deben utilizar otros métodos para poder tratar de resolver los problemas.

Una de las primeras alternativas planteadas fue una variación del filtro de Kalman: el filtro de Kalman Extendido o *Extended Kalman Filter* (EKF en inglés). El EKF es la versión no lineal del filtro de Kalman, que linealiza la estimación de la media y la covarianza actuales.

Las aproximaciones por serie de Taylor realizadas en el algoritmo del *EKF*, puede llevar a representaciones muy pobres de distribuciones de probabilidad de interés. Cuando no existe linealidad en los modelos, dichas distribuciones tienden a ser multi-modales. Las aproximaciones Gaussianas en estos casos pueden verse deterioradas debido a la no linealidad existente en los modelos. De hecho, si el modelo del ruido no es Gaussiano, el *EKF* puede llevar a resultados erróneos [25].

Los métodos secuenciales de Monte Carlo, *Sequential Monte Carlo Methods* (*SMC* en inglés), ofrecen un camino para superar estos problemas. Permiten obtener una representación completa de la distribución a posteriori, de tal manera que, cualquier estimación estadística como la media, la moda o la varianza, pueden ser computadas de manera sencilla. La aplicación del seguimiento en el área de la visión artificial, es un claro ejemplo de una situación donde las aproximaciones del filtro de Kalman no son capaces de aproximar distribuciones multi-modales, mientras que con los SMC, el proceso es posible [2] [25] .

Al contrario que los métodos *Grid* [36] [59], los cuáles no se estudiarán en este documento, los métodos SMC son muy flexibles, fáciles de implementar, paralelizables y aplicables en gran cantidad de situaciones. Los avances tecnológicos de los últimos tiempos, han permitido aumentar la potencia computacional de las máquinas a un precio bajo. Gracias también a las numerosas investigaciones realizadas sobre dichos métodos, se han logrado avances que han permitido mejorar aplicaciones en campos como la economía, la robótica, las telecomunicaciones o la biología [2] [5] [10] [14] [40].

Durante este capítulo se hablará de distintos métodos de Monte Carlo de los cuáles, en los últimos años, se han escrito una gran cantidad de informes científicos. El método *Importance Sampling* (IS), el *Bootstrap Filter*, comúnmente denominado filtro de partículas, son algunos ejemplos de métodos de Monte Carlo que se emplean a día de hoy en diversos campos.

4.2. Presentación del problema

Como medida de simplicidad, se va a restringir el estudio a señales modeladas como modelos espacio-temporales, no Gaussianos, no lineales y Markovianos, a pesar de que los métodos de Monte Carlo se pueden aplicar a muchos más casos. La señal no observada (estado oculto) $\{x_t; t \in N\}$, $\mathbf{x}_t \in X$, se modela como un *proceso de Markov* con $P(x_0)$ como distribución inicial, y $p(x_t|x_{t-1})$ como ecuación de transición [2] .

Las observaciones $\{y_t; t \in N\}$, $\mathbf{y}_t \in Y$, se asume que son condicionalmente independientes dado el proceso $\{x_t; t \in N\}$ y la distribución marginal $p(y_t|x_t)$. En resumidas cuentas, el modelo se describe mediante

$$\begin{aligned} p(x_0) & \\ p(x_t|x_{t-1}) & \quad \text{para} \quad t \geq 1, \\ p(y_t|x_t) & \quad \text{para} \quad t \geq 1. \end{aligned}$$

Se denotará $x_{0:t} \triangleq \{x_0, \ldots, x_t\}$ e $y_{1:t} \triangleq \{y_1, \ldots, y_t\}$ como la señal y las observaciones, respectivamente, hasta el instante t.

El objetivo es estimar a lo largo del tiempo y recursivamente, la *distribución a posteriori* $p(x_{0:t}|y_{1:t})$, sus características asociadas incluyendo la *distribución marginal* $p(x_t|y_{1:t})$, conocida como la *distribución de filtrado*, y las esperanzas

$$I(f_t) = E_{p(x_{0:t}|y_{1:t})}\left[f_t(x_{0:t})\right] \triangleq \int f_t(x_{0:t})\, p(x_{0:t}|y_{1:t})\, dx_{0:t}, \tag{4.1}$$

para una función de interés $f_t : X^{t+1} \to \mathbb{R}^{nf_t}$ integrable respecto a $p(x_{0:t}|y_{1:t})$. Como ejemplos de funciones apropiadas se incluyen la media condicional, en cuyo caso $f_t(x_{0:t}) = x_{0:t}$, o la covarianza condicional de x_t, donde $f_t(x_{0:t}) = x_t x_t^T - E_{p(x_t|y_{1:t})}\left[X_t\right]\ E^T_{p(x_t|y_{1:t})}[X_t]$.

En cualquier instante t, la distribución a posteriori viene dada por el teorema de Bayes,

$$p(x_{0:t}|y_{1:t}) = \frac{p(y_{1:t}|x_{0:t})p(x_{0:t})}{\int p(y_{1:t}|x_{0:t})p(x_{0:t})dx_{0:t}}. \tag{4.2}$$

Es posible obtener de manera directa una fórmula recursiva para la distribución conjunta $p(x_{0:t}|y_{1:t})$,

$$p(x_{0:t+1}|y_{1:t+1}) = p(x_{0:t}|y_{1:t})\frac{p(y_{1:t+1}|x_{t+1})p(x_{t+1}|x_t)}{p(y_{t+1}|y_{1:t})}. \tag{4.3}$$

La distribución marginal $p(x_t|y_{1:t-1})$ también satisface la siguiente recursión:

1) Predicción

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1}. \tag{4.4}$$

2) Actualización

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{\int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t}. \tag{4.5}$$

Estas expresiones y recursiones son, en general, **imposibles de calcular**, puesto que incluyen integrales complejas de altas dimensiones. Por esta razón, a mediados de los 60, se desarrollaron gran cantidad de estudios para tratar de abarcar dicho problema **aproximando las expresiones** anteriores. Los grandes avances surgidos a partir de de 1980, con

el incremento de la potencia computacional, han hecho posible importantes logros en los métodos de filtrado Bayesiano (Kitagawa 1987) [2] [37]. Este problema es conocido como **problema de filtrado de partículas** [1] [2] .

4.3. Métodos de Monte Carlo

Para resolver los problemas comentados en la sección anterior, muchas disciplinas científicas y de ingeniería se han dedicado al estudio y a la investigación de métodos de integración de Monte Carlo (MC). La principal ventaja que ofrecen es que no están sujetos a la linealidad o *Gaussianidad* del modelo, además de proporcionar atractivas propiedades de convergencia [2].

Durante el estudio de esta sección se verá que, cuando se tiene un número elevado de muestras de distribuciones a posteriori, no es complicado obtener buenas aproximaciones de las integrales vistas en las ecuaciones (4.3), (4.4) y (4.5), que se ha comentado que son complejas de resolver dadas sus dimensiones. Sin embargo, obtener muestras de estas distribuciones directamente, raramente es posible. Por lo tanto, se debe recurrir a métodos alternativos de Monte Carlo, como puede ser el método conocido en inglés como *Importance Sampling* (IS) o *muestreo según importancia* como se podría traducir al castellano [1]. Haciendo esta técnica recursiva de Monte Carlo, se obtiene el método conocido como *Sequential importance sampling* (SIS). Desgraciadamente, se puede ver cómo, a medida que aumenta t, el método SIS comienza a degradarse. Para solucionar dicho problema se debe introducir un paso más en cada iteración. Se basa en una técnica conocida como *resampling*, que se introduce en [37] y que se verá con más detenimiento en otra sección. Gracias a estos estudios se han generado algoritmos estables como describe Pierre Del Moral en [39].

4.4. Standard Monte Carlo sampling

Se asume que se puede generar una simulación de N muestras aleatorias, independientes e idénticamente distribuidas, conocidas como *partículas*, $\{x_{0:t}^{(i)}; i = 1,\dots N\}$ de acuerdo con $p(x_{0:t}|y_{1:t})$. Una estimación empírica de dicha distribución viene dada por

$$P_N(dx_{0:t}|y_{0:t}) = \frac{1}{N}\sum_{i=1}^{N}\delta_{x_{0:t}^{(i)}}(dx_{0:t}), \tag{4.6}$$

donde $\delta_{x_{0:t}^{(i)}}(dx_{0:t})$ denota la masa de la delta de Dirac localizada en $x_{0:t}^{(i)}$. Se obtiene de manera directa la siguiente estimación de $I(f_t)$

$$I_N = \int f_t(x_{0:t})P_N(dx_{0:t}|y_{1:t}) = \frac{1}{N}\sum_{i=1}^{N} f_t(x_{0:t}^{(i)}). \tag{4.7}$$

Esta estimación es insesgada y, si la varianza a posteriori de $f_t(x_{0:t})$ satisface que $\sigma_{f_t}^2 \triangleq E_{p(x_{0:t}|y_{1:t})}[f_t^2(x_{0:t})] - I^2(f_t) < +\infty$, entonces la varianza de $I_N(f_t)$ es igual a $var(I_N(f_t)) = \frac{\sigma_{f_t}^2}{N}$. De esta manera se cumple:

$$I_N(f_t) \longrightarrow I(f_t), \quad si \quad N \longrightarrow +\infty, \tag{4.8}$$

[1] Dada la complejidad de traducir ciertos nombres propios asignados a técnicas científicas, se utilizará para hacer referencia a éstos, la nomenclatura inglesa (por ejemplo: Importance Sampling)

donde $\longrightarrow$ indica que converge. Además si $\sigma_{f_t}^2 < +\infty$, entonces utilizando el teorema central del límite se obtiene que

$$\sqrt{N}[I_N(f_t) - If_t] \Longrightarrow \quad N(0, \sigma_{f_t}^2) \quad si \quad N \longrightarrow +\infty, \tag{4.9}$$

donde se denota $\Longrightarrow$ como la convergencia a la distribución. Es evidente la ventaja que nos ofrece este método. Dadas una muestras aleatorias $\{x_{0:t}^{(i)}; i = 1,\dots N\}$, se puede estimar cualquier cantidad $I(f_t)$ y el grado de convergencia de dicha estimación va a ser **independiente del dimensionado de las integrales**. Por otro lado, un método de integración numérico determinista tiene un grado de convergencia que decrece cuando la dimensión de la integral aumenta [2] .

Desafortunadamente, suele ser imposible hacer un muestreo eficiente de una distribución a posteriori $p(x_{0:t}|y_{1:t})$ en cualquier instante t, ya que $p(x_{0:t}|y_{1:t})$ puede seguir un modelo no estándar o ser multivariante. Para solucionar estos problemas se han investigado algoritmos basados en cadenas de Markov de Monte Carlo y que se estudiarán en esta sección.

4.5. Importance sampling

4.5.1. Batch importance sampling

Es posible aproximar la distribución a posteriori con una función con un número finito discreto de muestras. El problema reside en que no es posible, en la mayoría de los casos, conseguir muestras directamente de las distribuciones a posteriori.

Una alternativa para resolver los problemas del *Standard Monte Carlo sampling*, consiste en utilizar el método *Batch Importance sampling*. Previamente al estudio, se debe introducir la distribución conocida en inglés bajo el nombre de *importance sampling distribution* $\pi(x_{0:t}|y_{1:t})$. Asumiendo que se quiere evaluar $I(f_t)$, y siempre y cuando el soporte de $\pi(x_{0:t}|y_{1:t})$ incluya el soporte de $p(x_{0:t}|y_{1:t})$, se obtiene la identidad

$$I(f_t) = \frac{\int f_t(x_{0:t})\, w(x_{0:t})\, \pi(x_{0:t}|y_{1:t})\, dx_{0:t}}{\int w(x_{0:t})\, \pi(x_{0:t}|y_{1:t})\, dx_{0:t}},$$

donde $w(x_{0:t})$ es conocido, en terminología inglesa, como *importance weight* y que hace referencia a los pesos de las muestras [2]. Sigue la siguiente expresión:

$$w(x_{0:t}) = \frac{p(x_{0:t}|y_{1:t})}{\pi(x_{0:t}|y_{1:t})}. \tag{4.10}$$

De esta manera, si es posible realizar una simulación de N partículas idénticamente distribuidas e independientes (i.i.d) $\{x_{0:t}^{(i)},\ i = 1,\dots,N\}$ de acuerdo a $\pi(x_{0:t}|y_{1:t})$, una manera de estimar por Monte Carlo $I(f_t)$ es

$$\hat{I}_N(f_t) = \frac{\frac{1}{N}\sum_{i=1}^{N} f_t\left(x_{0:t}^{(i)}\right) w\left(x_{0:t}^{(i)}\right)}{\frac{1}{N}\sum_{j=1}^{N} w\left(x_{0:t}^{(i)}\right)} = \sum_{i=1}^{N} f_t\left(x_{0:t}^{(i)}\right) \tilde{w}_t^{(i)}, \tag{4.11}$$

donde $\tilde{w}_t^{(i)}$ hace referencia a los pesos normalizados dado por

$$\tilde{w}_t^{(i)} = \frac{w\left(x_{0:t}^{(i)}\right)}{\sum_{j=1}^{N} w\left(x_{0:t}^{(i)}\right)}. \tag{4.12}$$

Para una N finita, $\hat{I}_N(f_t)$ es un estimador sesgado, pero tiende a cero con N, i.e, es asintóticamente insesgado. Esto quiere decir que, para que el estimador $\hat{I}_N(f_t)$ se aproxime de manera adecuada a $I(f_t)$, el número de partículas N debe tender a infinito. Bajo ciertas condiciones, se puede obtener un teorema central del límite con un ratio de convergencia independiente de la dimensión de la integral [2] .

Este método de integración puede ser interpretado como un método de muestreo, donde la distribución a posteriori $p(x_{0:t}|y_{1:t})$ se aproxima como

$$\hat{P}_N(dx_{0:t}|y_{1:t}) = \sum_{i=1}^{N} \tilde{w}_t^{(i)} \, \delta_{x_{0:t}^{(i)}}(dx_{0:t}), \tag{4.13}$$

e $\hat{I}_N(f_t)$ no es otra cosa que la función $f_t(x_{0:t})$ integrada con respecto a la medida empírica $\hat{P}_N(dx_{0:t}|y_{1:t})$:

$$\hat{I}_N(f_t) = \int f_t(x_{0:t}) \hat{P}_N(dx_{0:t}|y_{1:t})$$

Batch importance sampling es un método general de Monte Carlo. Sin embargo, en su forma más simple, *no es adecuado para realizar estimaciones recursivas*. Esto se debe a que necesita recibir toda la información proveniente de las observaciones $y_{1:t}$ para poder estimar $p(x_{0:t}|y_{1:t})$. En general, en cada instante de tiempo, se dispone de nuevos datos y_{t+1}, luego, se requiere recalcular los pesos sobre la secuencia de estado completa. La complejidad del cálculo hace que esta operación se vea incrementada con el tiempo. Para poder resolver dicho problema, será necesario establecer ciertas estrategias que se comentarán en las siguientes secciones [2].

4.5.2. Sequential importance Sampling

El método *importance sampling* se puede modificar para aproximar $p(x_{0:t}|y_{1:t})$ sin tener que modificar el pasado de las trayectorias $\{x_{0:t-1}^{(i)}; i = 1, \dots N\}$ [2] [25].

Esto significa que la *importance function* $\pi(x_{0:t}|y_{1:t})$ en el instante t, admite como distribución marginal en el instante $t-1$, la *importance function* $\pi(x_{0:t-1}|y_{1:t-1})$, tal que

$$\pi(x_{0:t}|y_{1:t}) = \pi(x_{0:t-1}|y_{1:t-1}) \; \pi(x_t|x_{0:t-1}, y_{1:t}).$$

Iterando, se obtiene

$$\pi(x_{0:t}|y_{1:t}) = \pi(x_0) \prod_{k=1}^{t} \pi(x_k|x_{0:k-1}, y_{1:k}). \tag{4.14}$$

En este punto, es necesario recordar que en todo momento se está asumiendo que **los estados corresponden a procesos Markovianos** y que las observaciones son condicionalmente independientes dados los estados [25]. Por ello,

$$p(x_{0:t}) = p(x_0) \prod_{k=1}^{t} p(x_k|x_{k-1}), \tag{4.15}$$

$$p(y_{1:t}|x_{0:t}) = \prod_{k=1}^{t} p(y_k|x_k). \tag{4.16}$$

Se puede apreciar que la *importance function* permite evaluar de forma recursiva en el tiempo los pesos de la ecuación (4.12) . Si se sustituyen las ecuaciones (4.15) y (4.16) en la ecuación (4.12) se obtiene la estimación recursiva de los pesos como sigue [25] :

$$\tilde{w}_t^{(i)} \propto \tilde{w}_{t-1}^{(i)} \frac{p\left(y_t|x_t^{(i)}\right)\left(x_t^{(i)}|x_{t-1}^{(i)}\right)}{\pi\left(x_t^{(i)}|x_{0:t-1}^{(i)}, y_{1:t}\right)}. \tag{4.17}$$

La ecuación (4.17), ofrece un mecanismo para secuencializar la fase de actualización de los pesos. Dado que se puede muestrear la *importance function* y evaluar las probabilidades de verosimilitud y de transición, lo que se debe hacer es generar a priori un conjunto de muestras y, de manera iterativa, ir calculando sus pesos. Este proceso es el que se conoce como **Sequential Importance Sampling** (SIS), permitiendo obtener el tipo de estimación que se describía en la ecuación 4.11 [25].

Un caso particular de este estudio se presenta cuando se adopta la distribución a priori como *importance distribution*. En ese caso, la *importance function* quedaría de la siguiente manera:

$$\pi(x_{0:t}|y_{1:t}) = p(x_{0:t}) = p(x_0) \prod_{k=1}^{t} p(x_k|x_{k-1}).$$

En esta situación, los pesos satisfacen la expresión

$$\tilde{w}_t^{(i)} \propto \tilde{w}_{t-1}^{(i)} p\left(y_t|x_t^{(i)}\right).$$

Este método es bastante atractivo, pero no es más que una versión restringida del método *importance sampling*. Desgraciadamente, dicho método se ha visto que es ineficiente en espacios de altas dimensiones. A medida que t aumenta, el problema se presentará de la misma manera en el *sequential importance sampling* [2] . En el Algoritmo 1 está reflejado el pseudocódigo para implementar el SIS.

Elección de la *importance function*

La elección de la *importance function*, es uno de los diseños más críticos en los algoritmos de *importance sampling*. La preferencia de que la *importance function* minimizase la varianza de los pesos, fue propuesta por Doucet en 1997. El resultado de aquel argumento fue el siguiente:

Proposición 4.1 *La importance function* $\pi(x_t|x_{0:t-1}, y_{1:t}) = p(x_t|x_{0:t-1}, y_{1:t})$, *minimiza la varianza de los pesos condicionada a* $x_{0:t-1}$ *y a* $y_{1:t}$.

Esta elección de la importance function ha sido propuesta por otros investigadores como Liu y Chen en 1995, Zaritskii en 1975. Sin embargo, la distribución:

$$\pi(x_t|x_{0:t-1}, y_{1:t}) = p(x_t|x_{0:t-1}), \tag{4.18}$$

es la más popular, investigada por autores como Gordon en 1993, Avitzour en 1995, Kitagawa en 1996 ó Beadle y Djurić en 1997 [25] . A pesar de tener una variación mayor que la óptima importance function $p(x_t|x_{0:t-1}, y_{1:t})$, *es más sencilla de implementar porque no se incorporan las observaciones más recientes.*

Como se refleja en la Figura 4.1, si se comete un error al utilizar la información más reciente disponible para proponer los nuevos valores de los estados, solamente un conjunto pequeño de partículas sobrevivirá. Por ello es importante desplazar las partículas hacia la región de máxima verosimilitud [25] .

Algoritmo 1 *Sequential Importance Sampling (SIS)*

for $K = 0, 1, 2, \ldots$ **do**
  for $i = 1, \ldots, N$ **do**
    Muestrear $x_t^{(i)} \sim \pi\left(x_t | x_{0:t-1}^{(i)}, y_{0:t}\right)$
    Obtener un conjunto $x_{0:t}^{(i)} \triangleq \left(x_{0:t-1}^{(i)}, x_t^{(i)}\right)$
  end for
  for $i = 1, \ldots, N$ **do**
    Evaluar los pesos de cada una de las muestras:

$$\tilde{w}_t^{*(i)} = \tilde{w}_{t-1}^{*(i)} \frac{p\left(y_t | x_t^{(i)}\right) p\left(x_t^{(i)} | x_{t-1}^{(i)}\right)}{\pi\left(x_t^{(i)} | x_{0:t-1}^{(i)}, y_{0:t}\right)}$$

  end for
  for $i = 1, \ldots, N$ **do**
    Normalizar los pesos según

$$\tilde{w}_t^{(i)} = \frac{w_t^{*(i)}}{\sum_{j=1}^{N} w_t^{(*j)}}$$

  end for
end for

El problema de la degeneración en el algoritmo SIS

El algoritmo SIS discutido tiene una limitación importante: **la varianza de los pesos** en (4.10) **se incrementa estocásticamente a lo largo del tiempo.**

Proposición 4.2 *Proposición 1 de [3]. La varianza incondicionada de los pesos, aumenta a lo largo del tiempo. Para entender por qué el hecho de que la varianza de los pesos aumente se convierte en un problema, se va a suponer que se quiere realizar un muestreo de la distribución a posteriori. En este caso, el objetivo es conseguir una densidad de probabilidad que sea muy cercana a la densidad de probabilidad a posteriori. Cuando esto ocurre se obtiene los resultados de media y varianza dados por* (4.19) *y* (4.20) *respectivamente:*

$$E\left[\frac{P\left(x_{0:t} | y_{1:t}\right)}{\pi\left(x_{0:t} | y_{1:t}\right)}\right] = 1, \tag{4.19}$$

$$var\left(\frac{p\left(x_{0:t} | y_{1:t}\right)}{\pi\left(x_{0:t} | y_{1:t}\right)}\right) = E\left[\left(\frac{P\left(x_{0:t} | y_{1:t}\right)}{\pi\left(x_{0:t} | y_{1:t}\right)} - 1\right)^2\right] = 0. \tag{4.20}$$

En otras palabras, se espera que la varianza sea próxima a cero para obtener estimaciones razonables. Por lo tanto, un incremento de la varianza tendrá un efecto muy dañino en la precisión de las simulaciones. En la práctica, el problema de degeneración, puede ser observado monitorizando los pesos. Típicamente, lo que se observa es que, después de unas cuantas iteraciones, uno de los pesos normalizados tiende a la unidad, mientras que el resto de ellos tienden a cero [25] [55].

En la Figura 4.2, se representan dos situaciones en las que aparece el efecto negativo de la degeneración de partículas por el elevado incremento de la varianza. Se observa como sólamente sobreviven aquellas partículas que están en la región de la verosimilitud o likelihood. Por ello, es importante elegir una importance function adecuada al modelo a estimar.

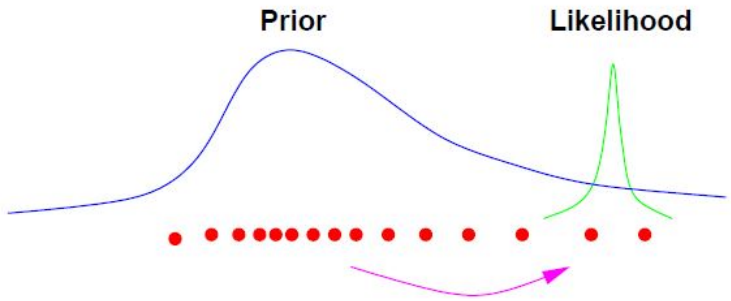
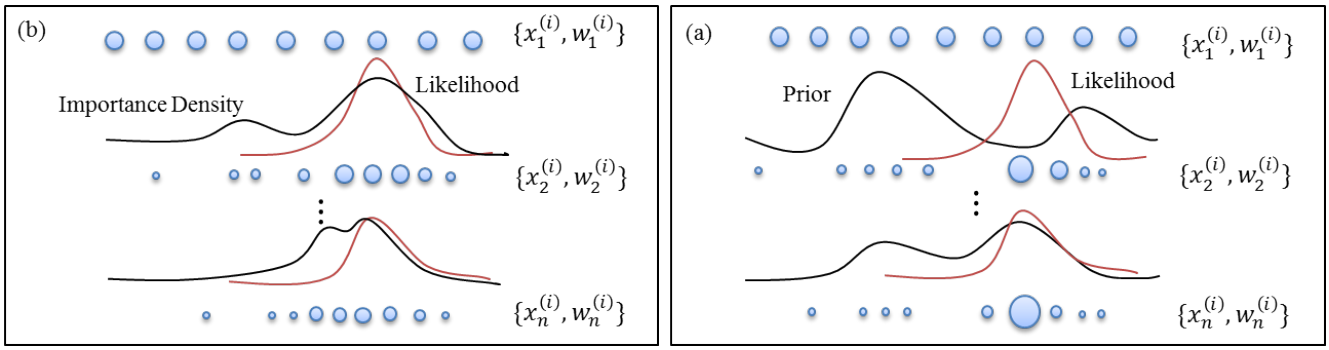


Figura 4.1: *La importance distribution óptima permite mover las muestras de la distribución a priori, a las regiones de la distribución de máxima verosimilitud. Esto es primordial en caso de que la verosimilitud se sitúe en una de las colas de la distribución a priori. Figura extraída de [25].*



(a) Situación a) (b) Situación b)

Figura 4.2: *Representación gráfica de dos situaciones donde aparece el problema de la degeneración de partículas en dos importance functions distintas. Figura extraída de [50].*

4.6. El *Bootstrap filter (BPF)*

El principal problema del método SIS es que, a medida que aumenta t, la distribución de los pesos $\tilde{w}_t^{(i)}$ se degeneran más y más. Prácticamente, después de unas pocas iteraciones, sólo una partícula tiene un peso asociado distinto a cero. El algoritmo, de esta manera, falla a la hora de representar la distribución a posteriori deseada. Para evitar el problema de la degeneración, es necesario añadir un paso de selección [2].

La idea clave del *bootstrap filter*, reside en **eliminar aquellas partículas cuyo peso asociado $\tilde{w}_t^{(i)}$ sea bajo, y replicar aquellas partículas cuyo peso asociado sea elevado**. Más formalmente, se dice que se sustituyen los pesos empíricos de la distribución $\hat{P}_N(dx_{0:t}|y_{1:t}) = \sum_{i=1}^N \tilde{w}_t^{(i)}\, \delta_{x_{0:t}^{(i)}}(dx_{0:t})$ por los pesos de la medida,

$$P_N(dx_{0:t}|y_{1:t}) = \frac{1}{N}\sum_{i=1}^{N} N_t^{(i)} \delta_{x_{0:t}^{(i)}}(dx_{0:t}),$$

donde $N_t^{(i)}$ es un número que representa el número de veces que la partícula $x_{0:t}^{(i)}$ ha aparecido en el muestreo de la función de distribución muestreada. Esto es un número entero tal que $\sum_{i=1}^N N_t^{(i)} = N$. Si $N_t^{(j)} = 0$, entonces la partícula $x_{0:t}^{(j)}$ muere. Los $N_t^{(i)}$ se escogen tal que $P_N(dx_{0:t}|y_{0:t})$ se aproxime a $\hat{P}_N(dx_{0:t}|y_{1:t})$ en el sentido de que, para cualquier función f_t,

$$\int f_t(x_{0:t})\; P_N(dx_{0:t}|y_{1:t}) \approx \int f_t(x_{0:t})\; \hat{P}_N(dx_{0:t}|y_{1:t}). \tag{4.21}$$

Después del paso de selección, las partículas que sobreviven $x_{0:t}^{(i)}$, es decir, aquellas cuyo $N_t^{(i)} > 0$, son las que entonces, se distribuyen de manera aproximada de acuerdo a $p(x_{0:t}|y_{1:t})$ [2] .

Hay muchas maneras de seleccionar $N_t^{(i)}$, la más popular fue la que introdujo Neil Gordon en 1993. En ella, se obtienen las partículas supervivientes tras haber muestreado N veces la distribución discreta $\hat{P}_N(dx_{0:t}|y_{1:t})$. Esto es el equivalente de muestrear el número de índices N_t^i de acuerdo a una distribución multinomial de parámetros $\tilde{w}_t^{(i)}$. La ecuación (4.21) se satisface en el sentido de que se puede comprobar fácilmente que, para una función f_t con $\| f_t \| = sup_{x_{0:t}} | f_t(x_{0:t}) |$, existe C tal que

$$E\left[\left(\int f_t(x_{0:t})\ P_N(dx_{0:t}|y_{1:t}) - \int f_t(x_{0:t})\ \hat{P}_N(dx_{0:t}|y_{1:t})\right)^2\right] \leq \frac{C\ \| f_t \|^2}{N}.$$

Para la implementación del algoritmo BPF, se deben seguir los pasos que se describen en el pseudocódigo del Algoritmo 2.

Algoritmo 2 *Bootstrap Particle Filter (BPF)*

1. *Inicialización*, t = 0.
for $i = 1,\ldots,N$ **do**
  Muestrear $x_0^{(i)} \sim p(x_0)$
end for
Avanzar a la siguiente iteración t = 1.
for $t = 1, 2, \ldots$ **do**
  2. *Ejecución del algoritmo importance sampling.*
  for $i = 1,\ldots,N$ **do**
    Muestrear $\tilde{x}_t^{(i)} \sim p\left(x_t|x_{t-1}^{(i)}\right)$.
    Almacenar el conjunto $\tilde{x}_{0:t}^{(i)} = \left(\tilde{x}_{0:t-1}^{(i)}, \tilde{x}_t^{(i)}\right)$
  end for
  for $i = 1,\ldots,N$ **do**
    Evaluar los pesos según :

$$\tilde{w}_t^{(i)} = p\left(y_t|\tilde{x}_t^{(i)}\right) \tag{4.22}$$

  end for
  Normalizar los pesos según (4.12)
  3. *Selección de partículas o resampling.*
  Volver a muestrear remplazando N partículas $\left(x_{0:t}^{(i)};\ i = 1,\ldots,N\right)$ del conjunto $\left(\tilde{x}_{0:t}^{(i)};\ i = 1,\ldots,N\right)$ de acuerdo a sus pesos.
end for

4.7. Resampling

Como se ha visto, tanto el algoritmo IS como el SIS ofrecen estimaciones cuya varianza crece típicamente de manera exponencial con n. Las técnicas del *resampling* son el ingrediente clave para que se resuelvan los problemas estudiados en los métodos SMC [1] [25].

El *resampling* es una técnica utilizada para evitar el problema de la degeneración de las partículas propagadas, modificando la medida aleatoria X_t en $\tilde{X}_t$, y mejorando la exploración del estado en el instante $t+1$. Para reducir el problema de la degeneración durante esta etapa de *resampling*, es importante que las medidas aleatorias se aproximen tan bien como puedan a la distribución original, evitando que haya sesgos que dificulten la mejora del problema. Aunque la aproximación $\tilde{X}_t$ va a ser similar a la original X_t, **el conjunto de partículas de $\tilde{X}_t$, va a ser diferente al de X_t** [54] .

La técnica del *resampling* indica que las partículas procedentes de X_t con pesos altos son más probables de dominar en $\tilde{X}_t$ que aquellas partículas cuyos pesos sean pequeños. Consecuentemente, en el siguiente instante temporal, se generarán nuevas partículas en la región de las partículas con pesos altos. Esta es la razón por la que los métodos SMC, mejoran tras la aplicación de dicha técnica. El foco de exploración cambia en las partes del espacio con masas de mayor probabilidad. Debido al *resampling*, la propagación de las partículas de $\tilde{X}_t$, tendrá pesos con menor discriminación que si la propagación fuera de las partículas de X_t.

Formalmente, el *resampling* es un proceso en el cuál se cogen muestran de una medida original aleatoria $X_t = \left\{x_t^{(m)}, w_t^{(m)}\right\}_{m=1}^{M}$, para crear una nueva medida aleatoria $\tilde{X}_t = \left\{\tilde{x}_t^{(m)}, \tilde{w}_t^{(m)}\right\}_{m=1}^{M}$. Después, se sustituyen las medida aleatoria X_t por $\tilde{X}_t$. Durante el proceso, algunas partículas procedentes de la distribución X_t se ven replicadas y, dado que son más probables, sus pesos son mayores. Las partículas procedentes de $\tilde{X}_t$, se utilizan para propagar nuevas partículas y por tanto, serán las *partículas padre* de $x_{t+1}^{(m)}$.

Es evidente que, para la aproximación de $p(x_{0:t}|y_{1:t})$, es mejor utilizar la distribución original X_t en vez de $\tilde{X}_t$. También se observa que el número de partículas obtenidas tras el *resampling* N, no es necesariamente igual al número de partículas propagadas. En métodos tradicionales de *resampling* se ha mantenido constancia entre el número de partículas propagadas M y el número de partículas obtenidas tras el resampling N, de tal manera que normalmente $M = N$. Finalmente, comentar que en muchos de los métodos de *resampling*, todos los pesos de las partículas después del proceso son iguales [54].

El método *Sequential importance resampling* (SIR), utiliza este paso para mejorar las prestaciones del método SIS de Monte Carlo. En este documento, nos centramos en el filtro de partículas, que es uno de los métodos SIR más comunes a día de hoy.

En la Figura 4.3, se aprecia de una manera más gráfica lo comentado hasta el momento sobre el proceso de *resampling*.

4.7.1. Efectos negativos del resampling

Sin embargo, el *resampling* también presenta efectos indeseados. Uno de ellos es el **empobrecimiento del muestreo**. Cuando se realiza el proceso de *resampling*, aquellas partículas con pesos bajos probablemente serán eliminadas, y por tanto, **la diversidad de las partículas se reduce** [37] . Por ejemplo, si unas pocas partículas de X_t tienen gran parte del peso, muchas partículas obtenidas tras el *resampling* serán iguales, y por ello, el número de partículas diferentes en $\tilde{X}_t$ será pequeño.

El otro efecto indeseado está en la **alta carga computacional** que dicho método añade al método SIS. Una buena mejora sería tratar de paralelizar el proceso.

Los efectos secundarios del *resampling*, han impulsado a los investigadores a investigar sobre métodos más avanzados de *resampling*. Estos métodos poseen varias características, incluyendo la variación del número de partículas, la restricción de la igualdad de los pesos en las partículas obtenidas tras el *resampling*, la abstinencia de descartar aquellas partículas

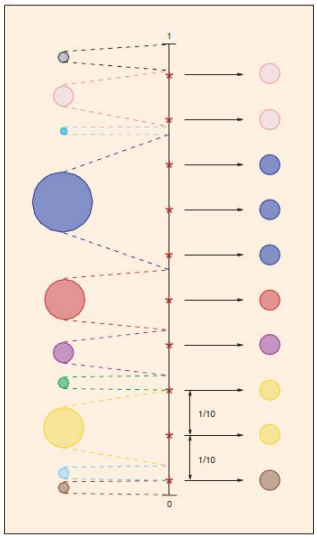


Figura 4.3: *Esquema descriptivo del resampling. Figura extraída de [41].*

con pesos bajos y la introducción de métodos paralelos.

Cuando se quiere implementar el proceso de *resampling*, se deben tomar muchas decisiones tales como elegir la distribución a muestrear, la estrategia de muestreo, determinar la frecuencia o el tamaño de muestreo [54]. Durante el capítulo que repercute al desarrollo de un método SIS aplicando *resampling*, se realizará un estudio minucioso de lo comentado hasta ahora.

En la Figura 4.4 se observan gráficamente los problemas de degeneración y de empobrecimiento que aparecen por la ausencia o el excesivo uso, respectivamente, de la técnica *resampling*. El tamaño de los círculos representa los pesos de las partículas. En la segunda fila, los círculos conectados comparten el mismo estado tras la etapa del *resampling*. Sólo las partículas representadas por los círculos verdes tienen pesos significativos, el resto, pintadas en rojo, serán eliminadas tras la etapa del *resampling*. El problema de la degeneración se produce porque los pesos se distribuyen con demasiada dispersión mientras que el del empobrecimiento viene por un exceso de partículas en una región concentrada [55].

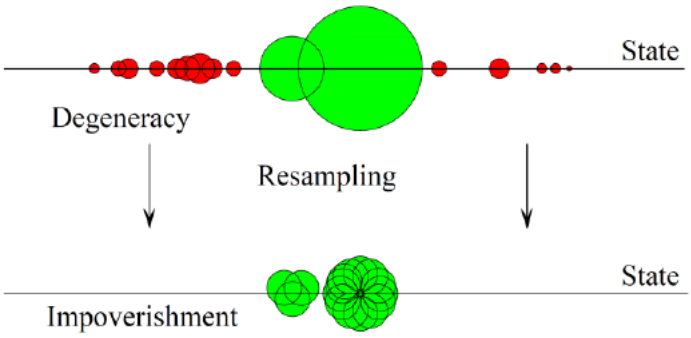


Figura 4.4: *Ejemplo de una situación donde se observan tanto el problema de degeneración como el de empobrecimiento. Figura extraída de [55].*

4.7.2. Resampling adaptativo

El *resampling* tiene el efecto de eliminar aquellas partículas cuyos pesos son bajos y replicar aquellas con pesos altos. Sin embargo, este es el coste de introducir un aumento en la

varianza. Si los **pesos normalizados** de las partículas tienen poca varianza, entonces **realizar el *resampling* es innecesario**. Consecuentemente, en la práctica es más razonable ejecutar el *resampling* solo cuando la varianza de los pesos normalizados[2] es superior a un cierto umbral. Se suele evaluar monitorizando la variabilidad de los pesos haciendo uso del criterio conocido como **Tamaño de Muestreo Eficaz** o *Effective Sample Size (ESS)* en inglés, que viene dado por,

$$ESS = \left(\sum_{i=1}^{K} \left(\bar{w}_t^{(i)} \right)^2 \right)^{-1}. \tag{4.23}$$

El ESS toma valores entre 1 y K, y usando este criterio, sólo se realizará el *resampling* cuando el ESS, esté por debajo de un cierto umbral $N_t = K/2$. Un criterio alternativo puede ser la **entropia de los pesos** $\bar{w}_t^{(i)}$, cuyo máximo valor ocurre cuando $\bar{w}_t^{(i)} = 1/K$. En este caso, el *resampling* se ejecuta cuando la entropía está por debajo de un umbral [1] [30] [31] . En el próximo capítulo, se realizarán experimentos con distintos umbrales para evaluar el ESS y determinar en qué momentos es mejor ejecutar el *resampling*. En el Algoritmo 3 se muestra el pseudocódigo necesario para implementar el método de *Sequential Importance Resampling* (SIR) haciendo uso del resampling adaptativo.

4.7.3. Clasificación de los métodos de resampling

Los distintos métodos de *resampling* se pueden clasificar de varias maneras. Una de ellas es la propuesta en [54], donde se dividen los métodos en función de su implementación en paralelo o secuencial. Simplemente recordar que las implementaciones en paralelo representan dos o más implementaciones secuenciales ejecutadas simultáneamente.

Las estrategias secuenciales se clasifican a su vez en base a si el *resampling* se realiza sobre una o dos distribuciones o sobre varias distribuciones obtenidas a partir de un grupo de partículas.

Otra categoría hace referencia a las estrategias especiales. Tal y como dice el nombre, estas estrategias especiales tienen características para separar el muestreo en simple o compuesto. En este documento se estudiarán los métodos pertenecientes al muestreo de distribuciones simples. Para completar más información referente a métodos de *resampling* avanzados, consultar [54]. En la Tabla 4.5 se describe la clasificación comentada.

A continuación se comentan algunas curiosiadades a tener en cuenta sobre las clasificaciones de los métodos de *resampling* propuestas en [54] :

1. Un tipo de clasificación se realiza en base a la distribución usada en el *resampling*. Principalmente esta distribución se representa como X_t. Otras aproximaciones incluyen el muestreo de una distribución aproximada. Este documento está centrado en los casos en los que el *resampling* no se hace sobre X_t.

2. El *resampling* puede actuar de la misma manera para todas las partículas, pero también es posible hacerlo de manera diferente en distintas partes del espacio del muestreo.

3. Existen métodos que agrupan partículas de alguna manera antes de ejecutar el *resampling*. En lo que aquí respecta, se distinguen entre **esquemas de muestreo de una distribución** (no se agrupan las partículas), **técnicas que combinan partículas adyacentes** (muy común cuando se trata de implementaciones en paralelo), y **técnicas que**

[2] A día de hoy hay investigadores que trabajan calculando el ESS con los pesos normalizados y otros que lo calculan con los pesos sin normalizar. En este trabajo, el ESS se ha calculado habiendo normalizado los pesos previamente.

Algoritmo 3 *SIR con Resampling Adaptativo.*

1. *Inicialización, t = 0.*

for $i = 1,\ldots,N$ **do**
Muestrear $x_0^{(i)} \sim p(x_0)$.
end for
Avanzar a la siguiente iteración t = 1.
for t = 1,2,... **do**
2. *Ejecución del algoritmo importance sampling.*
for $i = 1,\ldots,N$ **do**
Muestrear $\tilde{x}_t^{(i)} \sim p\left(x_t|x_{t-1}^{(i)}\right)$.
Almacenar el conjunto $\tilde{x}_{0:t}^{(i)} = \left(\tilde{x}_{0:t-1}^{(i)}, \tilde{x}_t^{(i)}\right)$
end for
for $i = 1,\ldots,N$ **do**
Evaluar los pesos (*importance weights*) según :

$$\tilde{w}_t^{(i)} = p\left(y_t|\tilde{x}_t^{(i)}\right) \tag{4.24}$$

end for
Normalizar los pesos según (4.12)
Evaluar ESS usando (4.23).
if $ESS \geq N_T$ **then**
for $i = 1,\ldots,N$ **do**
Almacenar el conjunto $x_{0:t}^{(i)} = \tilde{x}_{0:t}^{(i)}$.
end for
else
3. *Selección de partículas o resampling*
Replicar/ Eliminar muestras $\tilde{x}_{0:t}^{(i)}$ con altos/bajos pesos $\tilde{w}_t^{(i)}$, respectivamente, para obtener N muestras aleatorias $x_{0:t}^{(i)}$ distribuidas de forma aproximada de acuerdo a $p\left(x_{0:t}^{(i)}|y_{1:t}\right)$.
for $i = 1,\ldots,N$ **do**
Generar los pesos según $w_t^{(i)} = \tilde{w}_t^{(i)} = \frac{1}{N}$
end for
end if
end for

CLASSIFICATIONS	R1 BASED ON THE DISTRIBUTION x_t	R2 RESAMPLING OF ALL THE PARTICLES IN THE SAME WAY	R3 GROUPING	R4 USING INFORMATION FROM THE CURRENT TIME INSTANT
Sequential implementation	YES	YES/NO	YES/NO	YES/NO
■ SINGLE DISTRIBUTION SAMPLING [1], [7], [8], [10]	YES	YES	NO	YES
■ COMPOUND-SAMPLING		YES/NO FOR MANY: DIFFERENT OR NO RESAMPLING PER GROUP	BASED ON COMPOUND CRITERIA	
• THRESHOLDS/GROUPING-BASED RESAMPLING [11], [12], [24]				YES/NO FOR [11] AND [17]
• RESAMPLING THAT TAKES INTO ACCOUNT THE STATE [16]				YES
■ SPECIAL STRATEGIES		YES		
• MODIFIED RESAMPLING [11]	NO		NO	YES/NO FOR [18]
• VARIABLE-SIZE RESAMPLING [27]	YES			YES
• ROUGHENING [1]				
Parallel implementation			BASED ON ADJACENT PARTICLES	
■ MAPPING TO SPECIFIC HARDWARE PLATFORMS [31]–[33], [35]–[38] ■ DISTRIBUTED RESAMPLING [32], [40] ■ NORMALIZATION-FREE RESAMPLING [38], [39], [41]				

Figura 4.5: *Clasificación de los distintos métodos de resampling según su implementación secuencial o en paralelo. Figura extraída de [54].*

agrupan partículas para remuestrearlas en base a algún criterio predefinido (conocidas como *muestreo compuesto*).

4. El *resampling* se puede clasificar también en base a si solo las partículas del instante actual están implicadas en el proceso de *resampling*, que es una aproximación muy común, si las partículas de instantes anteriores se tienen en cuenta, o si las futuras partículas generadas deben considerarse en el *resampling*. También, el *resampling* puede ser clasificado en base a si solamente se tienen en cuenta los pesos, que es la aproximación estándar, o si se tiene en cuenta el estado de las partículas y su peso.

5. El *resampling* puede ser aplicado en instantes de tiempo determinados. Se han implementado compensaciones como el *roughening* para mejorar la actuación del proceso.

6. Hay métodos de *resampling* estocásticos y deterministas. Los métodos deterministas dirigen siempre un mismo conjunto de partículas para el mismo conjunto de partículas de entrada [54].

En los métodos de muestreo de distribuciones simples, el *resampling* es aplicado a todas las partículas siguiendo un procedimiento de muestreo de una distribución simple. El número esperado de veces $N_t^{(m)}$ que la m-enésima partícula es muestreada es proporcional a $w_t^{(m)}$. Dentro de este grupo se distinguirá entre métodos tradicionales y métodos variantes, aunque éstos últimos no se comentarán en profundidad en este documento.

4.7.4. Tipos de resampling

- **Resampling multinomial**.

  La idea principal de este método es generar N números aleatorios independientes $u_t^{(n)}$ de una distribución uniforme $(0,1]$ y utilizarlos para seleccionar partículas de X_t. En la n-enésima selección, la partícula $x_t^{(m)}$ es elegida al cumplirse la siguiente condición:

$$Q_t^{(m-1)} < u_t^{(n)} \leq Q_t^{(m)}, \tag{4.25}$$

  donde

$$Q_t^{(m)} = \sum_{k=1}^{m} w_t^{(k)}. \tag{4.26}$$

  Por consiguiente, la probabilidad de seleccionar la partícula $x_t^{(m)}$ es la misma que la de $u_t^{(n)}$, estando en el intervalo comprendido por la suma acumulada de los pesos normalizada visto en la ecuación (4.25). Este método satisface la condición de ser una estimación insesgada.

  El *resampling* multinomial tambiés es conocido como el *muestreo aleatorio simple*. Dado que el muestreo de cada partícula es aleatorio, los límites superior e inferior del número de veces en que una partícula se vuelva a muestrear son cero (no se muestrea) y N_t (muestreado N_t veces), respectivamente. Esto nos da la máxima varianza de las partículas muestreadas [54] .

  La complejidad computacional del resampling multinomial es del orden $O(NM)$, donde el factor M surge de la búsqueda de un m requerido en la ecuación (4.25). Dado que se ha visto que este método es ineficiente, se ha investigado sobre cómo se podría reducir la complejidad computacional. Además se han desarrollado métodos que contaremos a continuación donde la varianza del número de veces en que una partícula se vuelve a muestrear, se reduce [54] . En el Algoritmo 4 se presenta un esquema básico para el desarrollo de dicho método.

Algoritmo 4 *Resampling Multinomial.*

$\left[\left\{\tilde{x}_t^{(n)}\right\}_{n=1}^{N}\right] = \text{Resample}\left[\left\{x_t^{(m)}, w_t^{(m)}\right\}_{m=1}^{M}, N\right]$
$\left[\left\{Q_t^{(n)}\right\}_{n=1}^{N}\right] = \text{CumulativeSum}\left[\left\{w_t^{(m)}\right\}_{m=1}^{M}\right]$
$n = 0$
while $(n \leq N)$ **do**
  $u \sim U(0,1]$
  $m = 1$
  while $\left(Q_t^{(m)} < u\right)$ **do**
    $m = m + 1$
  end while
  $n = n + 1$
  $\tilde{x}_t^{(n)} = x_t^{(n)}$
end while

- **Resampling estratificado/sistemático**.

  El método estratificado, divide el conjunto completo de partículas en subconjuntos denominados *estratos*. Esto divide el intervalo $(0,1]$ en N subintervalos distribuidos tal que $(0,1/N] \cup \ldots \cup (1-1/N,1]$.

  Los números aleatorios $\left\{u_t^{(n)}\right\}_{n=1}^{N}$ se distribuyen independientemente en cada uno de los intervalos,

$$u_t^{(n)} \sim U\left(\frac{n-1}{N}, \frac{n}{N}\right], \qquad n = 1,2,\ldots,N, \tag{4.27}$$

  y después, se implementa el método basado en suma acumulada de los pesos normalizados visto en la ecuación (4.25).

  El *resampling* sistemático también explora la idea de dividir las partículas en estratos, pero de manera diferente. Ahora $u_t^{(1)}$ se obtiene de una distribución uniforme $(0,1/N]$, y el resto de los números aleatorios u se obtienen de manera determinista,

$$\begin{aligned} u_t^{(1)} &\sim (0,1/N], \\ u_t^{(n)} &= u_t^{(1)} + \frac{n-1}{N}, \qquad n = 2,3,\ldots,N. \end{aligned} \tag{4.28}$$

  Tanto el método sistemático como el estratificado presentan una complejidad del orden $O(N)$. Es importante destacar que el método sistemático es más eficiente que el estratificado debido a que se generan menos números aleatorios.

  Los límites superior e inferior del número de veces en que la partícula m-enésima es remuestreada en el método sistemático son $\lfloor Nw_t^{(m)} \rfloor$ y $\lfloor Nw_t^{(m)} \rfloor + 1$ respectivamente, donde $\lfloor x \rfloor$ representa la parte entera del número más alto que no excede a x. Por otro lado, en el método estratificado son $\max\left(\lfloor Nw_t^{(m)} \rfloor - 1, 0\right)$ y $\lfloor Nw_t^{(m)} \rfloor + 2$, respectivamente, porque las variables $\left\{u_t^{(n)}\right\}_{n=1}^{N}$ no son equidistantes y, en lugar de $\Delta u = u_t^{(n)} - u_t^{(n-1)}$ para $n = 2,3,\ldots,N$, varía entre 0 y $2/N$.

  Cuando Δu tiende a 0, una partícula cuyo peso sea pequeño (cercano a 0 pero mayor a Δu), puede ser remuestreada dos veces y cuando Δu es mayor a $2/N$, una partícula con un peso comprendido entre $1/N$ y Δu puede ser descartada. Esto indica que **la varianza del número de veces en que una partícula puede volver a ser muestreada por el método sistemático, es menor que en el caso del método estratificado** [54] . En el Algoritmo 5 se expone el pseudocódigo de los métodos estratificado y sistemático.

- **Resampling residual**.

  El método residual consta de dos pasos. El primero se basa en hacer una réplica determinista de cada partícula cuyo peso sea mayor a $1/N$, y el segundo consiste en hacer un muestreo aleatorio usando los residuos de los pesos. Se representará $N_t^{(m)}$ como el número de veces que se ha replicado la partícula $x_t^{(m)}$. Con el *resampling* residual, la m-enésima partícula es remuestreada $N_t^{(m)} + R_t^{(m)}$ veces, donde $N_t^{(m)}$ y $R_t^{(m)}$ son los números de las réplicas del primer y segundo paso respectivamente, y donde $N_t^{(m)} = \lfloor Nw_t^{(m)} \rfloor$.

Algoritmo 5 *Resampling Estratificado/Sistemático*

$\left[\left\{\tilde{x}_t^{(n)}\right\}_{n=1}^{N}\right] = \text{Resample}\left[\left\{x_t^{(m)}, w_t^{(m)}\right\}_{m=1}^{M}, N\right]$
$\left[\left\{Q_t^{(n)}\right\}_{n=1}^{N}\right] = \text{CumulativeSum}\left[\left\{w_t^{(m)}\right\}_{m=1}^{M}\right]$
$n = 0$
$m = 1$
Sistemático
$u_0 \sim U(0, 1/N]$
while $(n \leq N)$ **do**
  Estratificado
  $u_0 \sim U(0, 1/N]$
  $u = u_0 + n/N$
  while $\left(Q_t^{(m)} < u\right)$ **do**
    $m = m + 1$
  end while
  $n = n + 1$
  $\tilde{x}_t^{(n)} = x_t^{(n)}$
end while

El número total de partículas replicadas en el primer paso viene dado por,

$$N_t = \sum_{m=1}^{M} N_t^{(m)},$$

y en el segundo paso $R_t = N - N_t$. El residuo de los pesos se obtiene directamente de

$$\hat{w}_t^{(m)} = w_t^{(m)} - \frac{N_t^{(m)}}{N}. \tag{4.29}$$

En el segundo paso, las partículas son distribuidas en base al residuo de los pesos y haciendo uso del *resampling* multinomial u otro método de los vistos anteriormente, donde la probabilidad de seleccionar $x_t^{(m)}$ es proporcional al residuo de los pesos de dicha partícula. El método residual tiene dos bucles cuya complejidad computacional es del orden de $O(M) + O(R_t)$ en cada uno de ellos.

Dado que el primer paso simplemente representa la réplica determinista, la varianza del número de veces en que una partícula se vuelve a muestrear se atribuye solamente al segundo paso. De esta manera, los límites superior o inferior de lo anterior son $\lfloor N w_t^{(m)} \rfloor$ y $\lfloor N w_t^{(m)} \rfloor + R_t$, respectivamente, si el segundo paso es implementado mediante el método multinomial [54]. En el Algoritmo 6 se presenta dicho método.

En la Figura 4.6, las flechas representan las localizaciones del muestreo, y una partícula se muestrea si una flecha apunta a dicha partícula [54]. En dicha Figura se puede contemplar de una manera más gráfica la división del espacio muestral en intervalos uniformes o no, en función del método empleado.

Los métodos mencionados anteriormente seguramente sean los más conocidos y los más utilizados. Dichos métodos se han modificado de algún modo para sacar un mejor rendimiento a la implementación de los métodos de Monte Carlo en aplicaciones reales. Uno de

Algoritmo 6 *Resampling Residual.*

$$\left[\left\{\tilde{x}_t^{(n)}\right\}_{n=1}^{N}\right] = \text{Resample}\left[\left\{x_t^{(m)}, w_t^{(m)}\right\}_{m=1}^{M}, N\right]$$

for $m = 1 : M$ **do**

$$N_t^{(m)} = \text{Floor}\left(N_t^{(m)} x\, w_t^{(m)}\right)$$

$$\hat{w}_t^{(m)} = w_t^{(m)} - N_t^{(m)}/N$$

end for

$$\left[\left\{\tilde{x}_t^{(n)}\right\}_{n=1}^{N}, N_t\right] = \text{Replication}\left[\left\{x_t^{(m)}, N_t^{(m)}\right\}_{m=1}^{M}\right]$$

for $m = 1 : M$ **do**

$$\hat{w}_t^{(m)} = \hat{w}_t^{(m)} x\, N/(N - N_t)$$

end for

$$\left[\left\{\tilde{x}_t^{(n)}\right\}_{n=N_t+1}^{N}\right] = \text{(Multinomial)Resampling}\left[\left\{x_t^{(m)}, w_t^{(m)}\right\}_{m=1}^{M}, N - N_t\right]$$

estos métodos elimina la alta carga computacional que el *resampling* multinomial introduce en el segundo paso del residual. Este método se conoce como *resampling residual y sistemático (RSR)*. La complejidad computacional de este método es del orden de $O(N)$.

Para poder desarrollar los métodos de resampling mencionados, se puede hacer uso de [56], donde se facilitan las implementaciones en MATLAB de estos métodos.

Otros métodos como el conocido en inglés como *branching*, adapta el número de partículas que se obtienen tras el *resampling* en vez de mantenerlo siempre constante. En este documento, se va a centrar el estudio de los métodos de *resampling* tradicionales. Para consultar más información sobre estas variaciones en los métodos se recomienda la lectura de [54].

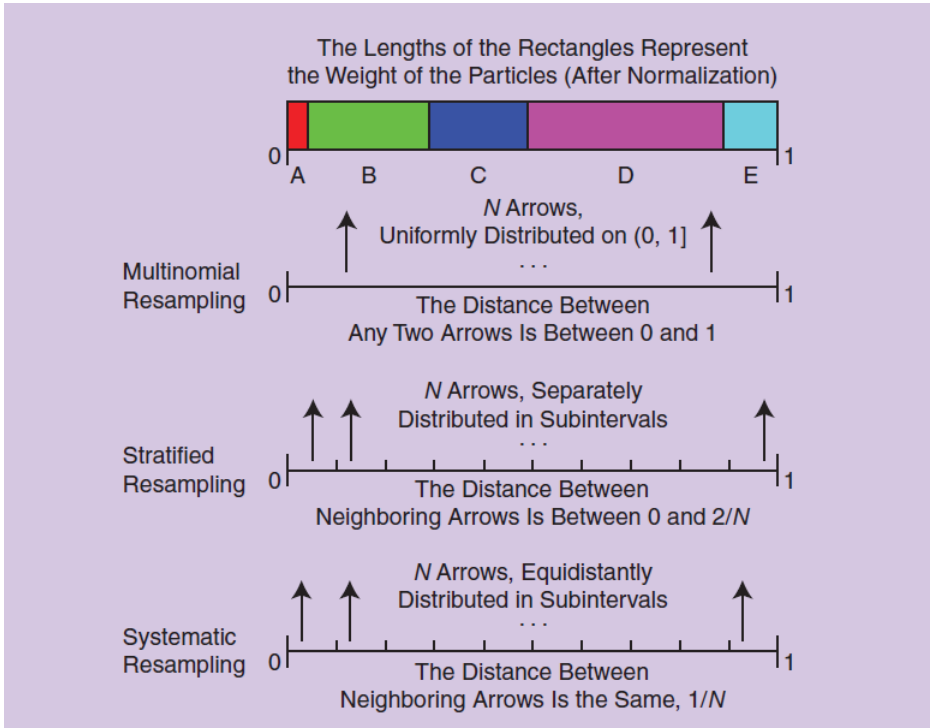


Figura 4.6: *Representación gráfica de los métodos de resampling multinomial, estratificado y sistemático basados en la suma acumulada de los pesos normalizados de las partículas. Figura extraída de [54].*

Capítulo 5

Desarrollo de filtros de partículas en MATLAB

5.1. Introducción

El objetivo de este capítulo es, visualizar gráficamente la resolución de un problema de estimación de estados ocultos de un sistema de interés, dado un cierto modelo que se supone conocido, cuando la situación **no es Gaussiana o lineal**. En este caso, el filtro de Kalman, no funciona adecuadamente dadas estas características y se busca, a través de los métodos de Monte Carlo, solucionar este problema. Para dicha tarea se empleará de nuevo la herramienta MATLAB.

El modelo escogido se ha utilizado en diversas publicaciones sobre filtros de partículas dado su alto conocimiento y su buena respuesta ante dicho filtro.

5.2. Planteamiento del Problema

Se considera el siguiente conjunto de ecuaciones como ejemplo ilustrativo [36] :

$$p(x_k|x_{k-1}) = N(x_k; f_k(x_{k-1}, \omega), Q_{k-1}), \tag{5.1}$$

$$p(y_k|x_k) = N(y_k; h_k(x_k), R_k), \tag{5.2}$$

o equivalentemente,

$$x_k = f_k(x_{k-1}, \omega) + w_{k-1}, \tag{5.3}$$

$$y_k = h_k(x_k) + v_k, \tag{5.4}$$

donde las funciones f_k y h_k son no lineales y cuyas expresiones vienen dadas por

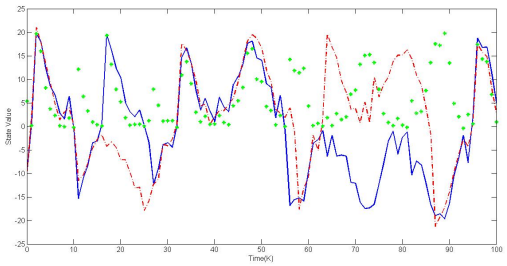
$$f_k(x_{k-1}, \omega) = \frac{x_{k-1}}{2} + \frac{25x_{k-1}}{1 + x_{k-1}^2} + 8cos(1{,}2\omega),$$

$$h_k = \frac{x_k^2}{20}.$$

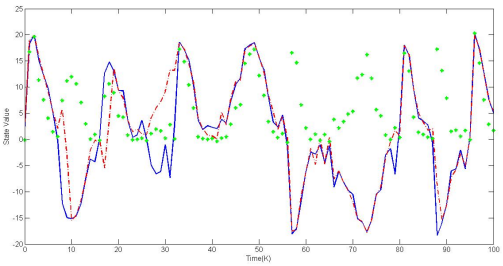
En este caso w_{k-1} y v_k son ruidos aleatorios y Gaussianos de media cero y varianzas Q_{k-1} y R_k, respectivamente y ω es el parámetro de la frecuencia medido en rad/s.

5.3. Simulación de un filtro de partículas

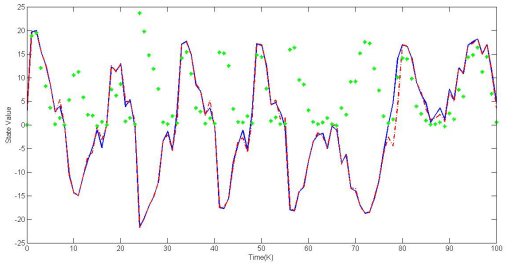
Experimento 5.1 *Se desea implementar un BPF con un número variable de partículas para comprobar el efecto que produce en el MSE y en el coste computacional cuando se aumenta el número de partículas. Con el objetivo de ver el resultado de una manera más precisa, se han utilizado las potencias en base binaria para el número de partículas, de tal manera que,* $M \in \left\{2, 2^2, 2^3, 2^4, \ldots, 2^{10}\right\}$*, siendo* M *el número de partículas. Para cada uno de los valores de* M *se han realizado 100 simulaciones. En la Figura 5.1, se presentan cuatro gráficas pertenecientes a la estimación de los estados ocultos de una determinada variable dadas unas observaciones o medidas, utilizando 2, 16, 128 y 1024 partículas, con ánimo de ver las diferencias en la precisión de la estimación, así como la influencia del coste computacional en la ejecución del algoritmo. En azul, se representa la trayectoria de los estados reales, en rojo, la estimación realizada por el filtro de partículas y en verde, las medidas realizadas por el sensor. La Tabla 5.1 muestra los valores utilizados durante la simulación.*



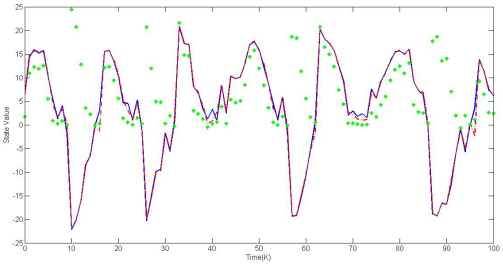
(a) 2 Partículas



(b) 16 Partículas



(c) 128 Partículas



(d) 1024 Partículas

Figura 5.1: *Resultados gráficos de estimación para el Experimento 5.1, empleando el filtro de partículas cuando el número de partículas* $M = 2{,}16{,}128$ *y 1024.*

Para medir la precisión de las estimaciones, se ha representado en la Figura 5.2 el error cuadrático medio (MSE) en función del número de partículas. Se ha promediado temporalmente en cada una de las simulaciones para conseguir un sólo valor promedio asociado a un número de partículas. Se observa gráficamente el efecto positivo que tiene aumentar el número de partículas en la disminución del MSE.

Los resultados[1] de la Tabla 5.2, muestran lo que a priori parecía evidente: a medida que se aumenta el número de partículas, el MSE disminuye pero el coste computacional aumenta. También se refleja que a partir de 2^7 partículas, el MSE ya no mejora demasiado pero el coste computacional se dispara.

En este caso, la mejor solución sería elegir 2^8 partículas durante la simulación para la resolución del problema de estimación, puesto que con dicho valor el MSE es bastante bajo y el

[1] Los valores obtenidos en MATLAB para rellenar la Tabla 5.2 han sido aproximados a un decimal para facilitar la lectura de los resultados.

Parámetros del algoritmo durante la simulación	Valor
Número de simulaciones	100
Número de instantes temporales (K)	100
X_0 (media del estado inicial)	0.1
P_0 (Matriz de covarianza del estado inicial)	2
σ_w^2	2
σ_v^2	0.1

Tabla 5.1: *Valores utilizados durante la simulación del Experimento 5.1.*

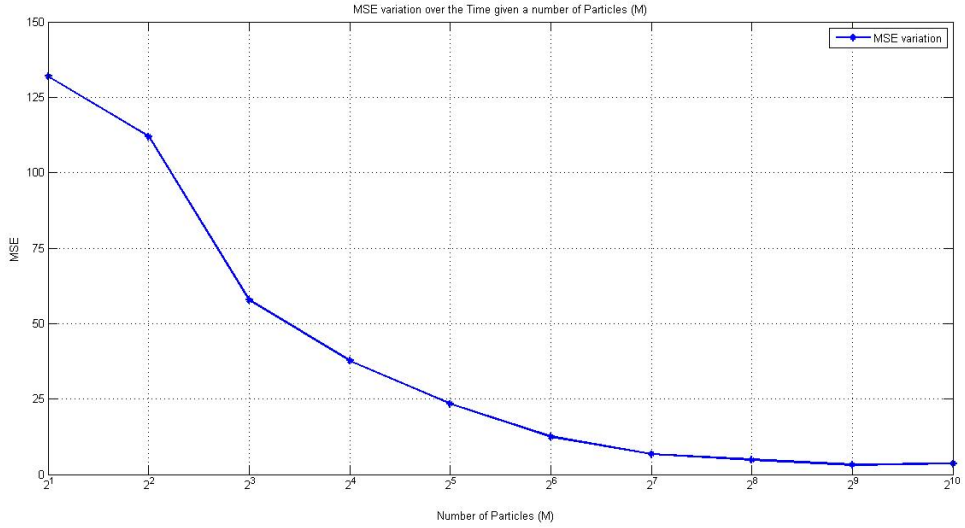


Figura 5.2: *Representación del error cuadrático medio MSE para el Experimento 5.1, en función del número de partículas M.*

coste computacional no se ha disparado demasiado. Aumentar a 2^9 el número de partículas dispararía demasiado el coste computacional.

Elegir el número de partículas óptimo para la estimación suele hacerse de manera *heurística*, y se mantiene fijo a lo largo de la ejecución del algoritmo. Se realizan muchas simulaciones promediando el error y viendo qué situaciones favorecen la implementación del algoritmo y cuáles deterioran las características del mismo.

Para mejorar la robustez del método, se podría adaptar el número de partículas durante la ejecución del algoritmo en función de la complejidad de la estimación. De esta manera se conseguiría minimizar el error cuadrático pero manteniendo una carga computacional menor que si se escogiera durante toda la ejecución un número fijo de partículas. Otra opción es buscar la mejor solución para el tipo de aplicación que se quiere utilizar. En función de las características demandadas por la aplicación, como pudiera ser la precisión o el retardo, se podría emplear un número mayor o menor de partículas. De esta manera, estableciendo un buen criterio entre precisión y retardo, se conseguiría optimizar la eficiencia del algoritmo

Número de partículas	Error Cuadrático Medio (MSE)	Coste computacional (segundos)
2	109.4	2
2^2	80.7	3.9
2^3	56.2	6.3
2^4	36.3	9.6
2^5	14.4	14.5
2^6	9.8	23.2
2^7	5.7	37.4
2^8	3.6	63.4
2^9	2.8	111.6
2^{10}	2.8	204.1

Tabla 5.2: *Resultados del MSE y la carga computacional en función del número de partículas para el Experimento 5.1.*

para dicha aplicación.

Otra posible solución sería emplear el ya conocido ESS. De esta manera, el *resampling* no se tendría que ejecutar siempre, sino que sólo cuando se considere totalmente necesario. Dicha solución será la propuesta en el Experimento 5.2 que se verá a continuación.

En muchas situaciones es innecesario ejecutar el *resampling* ya que éste provoca que el muestreo se empobrezca dado que sólo sobreviven unas pocas partículas. Se ha hablado del papel del *Tamaño de muestreo eficaz* o ESS para evaluar en qué instantes es necesario ejecutar el *resampling* y en cuáles no.

Experimento 5.2 *Se desea realizar un experimento para ver la influencia del ESS en el funcionamiento del filtro de partículas. El objetivo del experimento es establecer un criterio entre el ESS y el MSE viendo los resultados obtenidos. La simulación se ha desarrollado variando el umbral que se compara con el ESS desde cero hasta la unidad, en pasos de* 0,1 *para cada número de partículas M.*

En la Figura 5.3 está representado el MSE en distintos colores en función de su valor. El color rojo simboliza que el MSE toma valores muy altos, mientras que el color azul representa una disminución del mismo. Se vuelve a ver la relación entre el número de partículas y el valor del MSE. En el caso del ESS, **si toma el valor de cero, quiere decir que no se ha ejecutado el *resampling*** en ninguna iteración, mientras que **si el ESS toma el valor 1, quiere decir que en cada iteración, el algoritmo sí que ejecuta el paso del *resampling***. Al poder contemplar directamente la relación entre la precisión y el retardo, es sencillo aplicar dicho criterio a una aplicación para obtener los máximos beneficios posibles. Se han utilizado pasos de 0,1 para poder apreciar la relación entre el MSE y el ESS, de tal manera que se observa cómo **a medida que el ESS aumenta, el MSE disminuye para un número de partículas menor**. La principal ventaja de esta gráfica es que permite relacionar tres parámetros importantes del filtro de partículas a la vez: el MSE, el ESS y el número de partículas M. Además, existe una relación directamente proporcional entre el número de partículas y el tiempo de ejecución del algoritmo. Si el ESS aumenta, el MSE disminuye para un número de partículas menor, pero, al tener que ejecutar más veces el *resampling*, el coste computacional final

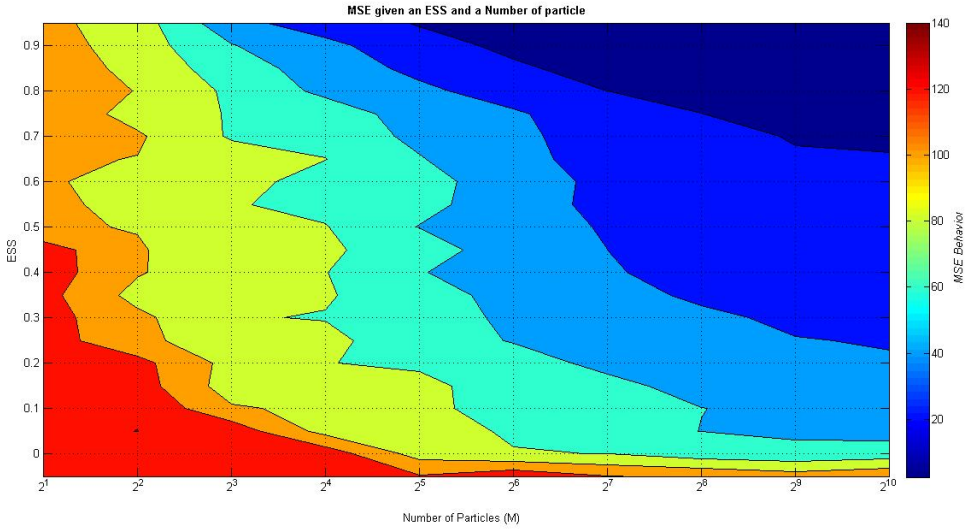


Figura 5.3: *Comparación del MSE en función del umbral ESS y del número de partículas M obtenida durante el Experimento 5.2.*

aumentará. Existe un debate entre utilizar más partículas y un ESS menor, o utilizar menos partículas y aumentar el ESS.

Para este experimento la mejor solución estaría en utilizar un umbral de entre 0.70-0.80 para un número de partículas superior a 2^6. Se puede ver en la Figura 5.3 cómo un número de menor de partículas implicaría una disminución considerable del MSE, y un valor superior a 0.8 en el umbral de comparación del ESS supondría un aumento considerable del coste computacional puesto que el *resampling* se ejecutaría más veces.

Es evidente que la elección de estos parámetros en muchos casos es heurística y, por tanto, en función del tipo de aplicación se deben ajustar dichos valores a los que se consideren oportunos para el funcionamiento correcto de la misma. No siempre será mejor obtener la estimación más precisa si ello implica un tiempo de ejecución demasiado alto y, en otras situaciones donde se busque más precisión en la estimación y no importe tanto el tiempo de ejecución, se puede aumentar M o el umbral del ESS, hasta obtener los resultados deseados.

Experimento 5.3 *En este experimento se va a estudiar qué ocurre con la estimación llevada a cabo por el filtro de partículas, cuando el resampling se realiza cada w iteraciones, siendo w el tamaño de la ventana. Dado que hay 100 iteraciones en total, la ventana* $w \in \{1, 6, 11, 16, \ldots, 101\}$. *Cuando* $w = 1$, *el resampling se ejecuta en todas las iteraciones y, cuando la ventana toma el valor máximo, 101, no se ejecuta nunca, puesto que, evidentemente, la ventana es mayor al número de iteraciones. De esta manera se podrá contemplar lo que pasa cuando la ventana se va aumentando de tamaño, y el resampling deja de ejecutarse en todas las iteraciones. El objetivo de dicho experimento es ver si el resampling podría ejecutarse tras un número fijo de iteraciones sin añadir un deterioro en la estimación.*

Si se observa la Figura 5.4, se ve claramente que no existe ningún tipo de uniformidad como sucedía en el Experimento 5.2 y, por tanto, no se aprecia una relación directa entre el MSE, el número de partículas necesario para una estimación razonable y el tamaño de la ventana para la ejecución del *resampling*. Se observa un funcionamiento peor que en el caso de utilizar el ESS, dado que no se consigue llegar a un error tan pequeño como ocurría en el Experimento 5.2. Se aprecian casos en los que el MSE es bastante bajo y el tamaño de w grande, teniendo como consecuencia que el *resampling* no se ejecuta demasiadas veces y que por tanto, el coste computacional sería menor. Sin embargo, los resultados de la Figura 5.3

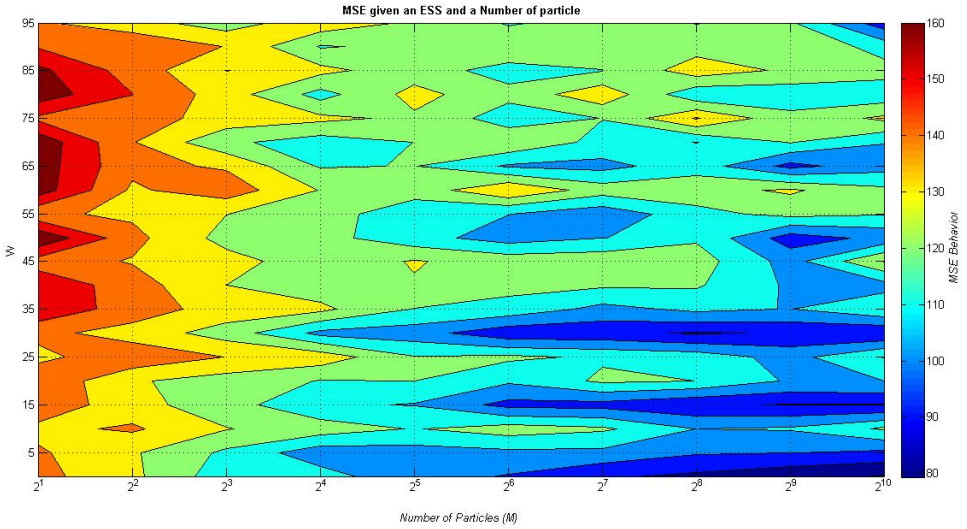


Figura 5.4: *Comparación del MSE en función del tamaño de ventana W y del número de partículas M obtenida durante el Experimento 5.3.*

son muchísimo mejores ya que emplean el uso de la varianza de los pesos y es un método adaptativo, mientras que en el caso de utilizar un tamaño de ventana w fijo para ejecutar el *resampling*, no se está teniendo en cuenta ningún tipo de análisis estadístico, obteniendo, por lo tanto, resultados peores.

Numerosos investigadores han propuesto nuevos métodos de *resampling* con ánimo de disminuir la alta carga computacional que conlleva ejecutar dicho paso en el filtro de partículas. El principal objetivo de las investigaciones se basa en modificar algunos métodos tradicionales como el acumulado, multinomial, residual, estratificado o el sistemático, con ánimo de mejorar las características del algoritmo BPF.

Experimento 5.4 *La idea de este nuevo experimento es englobar los resultados de los tiempos de ejecución de cada uno de los métodos mencionados, durante la simulación del filtro de partículas. Para ello, se ha ejecutado el filtro de partículas con un umbral para el ESS de 0.75, utilizando un método de resampling de los mencionados y evaluando el MSE y el coste computacional que implica dicho método. Este proceso se ha realizado para cada uno de los métodos de resampling. El objetivo del estudio es encontrar un método que minimice el error lo máximo posible y además con un coste computacional bajo.*

Observando la Figura 5.5, se aprecia cómo la curva del MSE para cada uno de los métodos de *resampling* es similar y, claramente se observa una mejora del MSE a medida que el número de partículas aumenta.

Viendo la Tabla 5.3, se aprecia que todos los métodos dan un MSE aproximado similar, lo que en principio es una ventaja, puesto que el uso de uno u otro no va a suponer un aumento del error en la estimación y, por tanto, una pérdida en la precisión. Sin embargo, observando la columna que representa el coste computacional, se puede apreciar claramente cómo ciertos métodos, como el estratificado o el sistemático, son más eficientes que el resto y proporcionan una mayor velocidad en la simulación del algoritmo, que para ciertas aplicaciones será necesario.

En conclusión, se puede afirmar que, para obtener un coste computacional bajo pero manteniendo el MSE lo más pequeño posible, sería idóneo **emplear el método estratificado** o el **sistemático**, ambos comentados en profundidad en la sección correspondiente al *resampling*. Para un conocimiento mayor sobre variaciones de estos métodos tradicionales y sobre

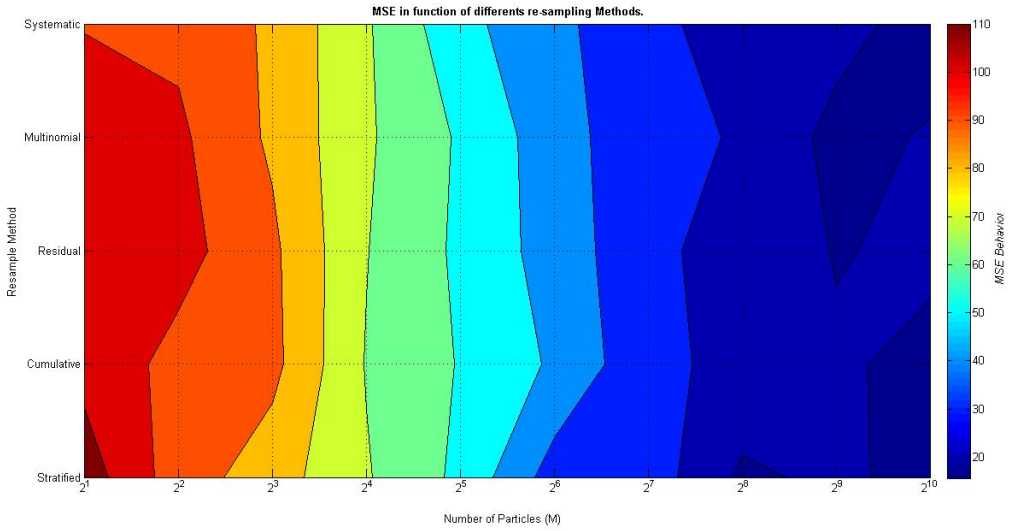


Figura 5.5: *Comparación del MSE en función del método de resampling utilizado y del número de partículas M obtenida durante el Experimento 5.4.*

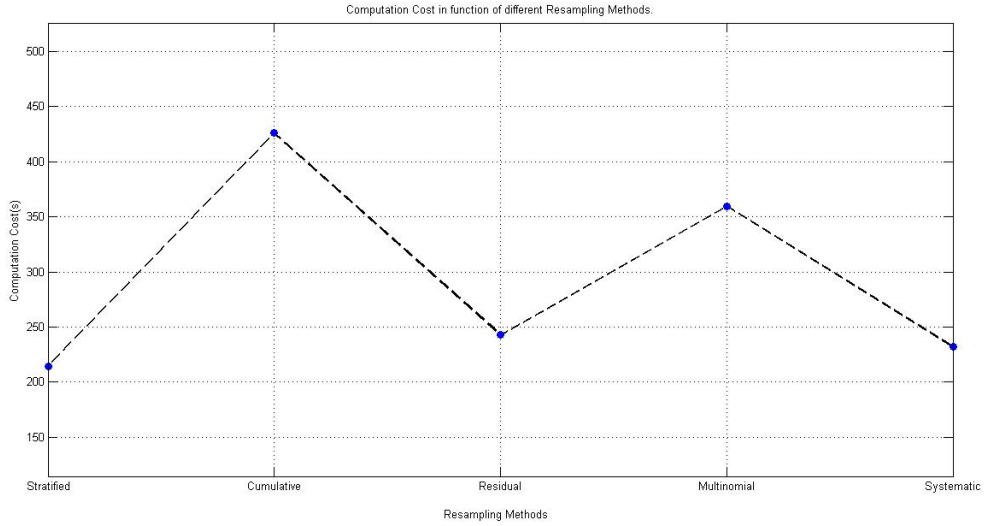


Figura 5.6: *Resultados de los costes computacionales del algoritmo en función del método de resampling utilizado obtenidos durante el Experimento 5.4.*

Método de resampling	Nº ejecuciones del resampling	Coste computacional (segundos)	MSE (en promedio)
Acumulado	87765	425.7	57.3
Residual	87780	242.9	57.5
Estratificado	87654	214.6	55.1
Sistemático	87826	232.0	55.0
Multinomial	87892	359.3	57.3

Tabla 5.3: *Resultados del MSE y el coste computacional en función del método de resampling utilizado obtenidos durante el Experimento 5.4.*

sus carácterísticas, sería recomendable la lectura de [54] dado que, en este documento, se ha puesto especial atención a los métodos tradicionales, no así a las variaciones de dichos métodos, cuyas características les hacen más populares en el uso del filtro de partículas, ya que mejoran las características propias de los métodos tradicionales.

Experimento 5.5 *En este experimento se desea comparar el funcionamiento del algoritmo subóptimo del filtro de partículas en el modelo lineal empleado en el filtro de Kalman dado por las ecuaciones* (3.20) *y* (3.21).

El principal objetivo es poder apreciar la diferencia de los resultados del MSE en el caso de utilizar el filtro de partículas y en el caso de utilizar el filtro de Kalman, así como, comparar los tiempos de ejecución y el número de partículas para el cuál, el filtro de partículas consigue prácticamente los mismos resultados que el filtro de Kalman.

Para llevar a cabo la simulación en MATLAB, se han modificado en el código las ecuaciones no lineales (5.3) *y* (5.4) *del modelo del filtro de partículas, por las ecuaciones lineales* (3.20) *y* (3.21) *empleadas en el modelo del filtro de Kalman. Durante la simulación se ha empleado el método de resampling estratificado y el umbral de comparación del ESS se ha establecido en 0.75.*

Durante la simulación, se ha ido aumentando en potencias de 2 el número de partículas como se ha hecho en el Experimento 3.1. En la Tabla 5.4 se enuncian los elementos empleados durante la simulación. Para que tenga sentido el experimento, se ha realizado una nueva simulación del filtro de Kalman idéntica a la realizada en el Experimento 3.1, pero empleando los valores que se muestran en la Tabla 5.4.

Parámetros del algoritmo durante la simulación	Valor
Número de simulaciones	100
Número de instantes temporales (K)	100
X_0 (media del estado inicial)	0.1
P_0 (Matriz de covarianza del estado inicial)	I (Matriz Identidad)
σ_w^2	0.01
σ_v^2	0.0001
H	1
F	1

Tabla 5.4: *Valores utilizados durante la simulación del Experimento 5.5 donde el filtro de Partículas se emplea para la estimación el modelo lineal del filtro de Kalman.*

Tras ejecutar la simulación del filtro de partículas empleando el modelo de Kalman, se han recogido los resultados correspondientes al MSE y al coste computacional en función del número de partículas en la Tabla 5.5.

Después se ha realizado la simulación del filtro de Kalman con los mismos valores. En la Tabla 5.6 se muestran los resultados de dicho experimento[2].

[2]Dado que en el filtro de Kalman el MSE se da en función del número de épocas, se ha hecho un promedio temporal para obtener un solo valor. Dicho valor se va comparando con cada uno de los valores del MSE obtenidos para el filtro de partículas.

Para ver de una manera más visual la mejora de las estimaciones con el aumento del número de partículas, se han recopilado algunas gráficas de la simulación que se muestran en la Figura 5.7. En verde están representadas las observaciones, el rojo la estimación realizada por el filtro de partículas y en azul, los valores reales de los estados. En la Figura 5.8(a) queda reflejado la mejora del MSE cuando el número de partículas aumenta considerablemente y, en la Figura 5.8(b) se muestra la principal consecuencia del aumento del número de partículas: el alto coste computacional.

Número de partículas	Error Cuadrático Medio (MSE)	Coste computacional (segundos)
2	0.4	2.6
2^2	0.2	6.4
2^3	0.03	12.2
2^4	0.006	18.6
2^5	0.003	35.5
2^6	0.002	56.0
2^7	0.002	97.0
2^8	0.001	185.2
2^9	0.001	345.9
2^{10}	0.001	616.8

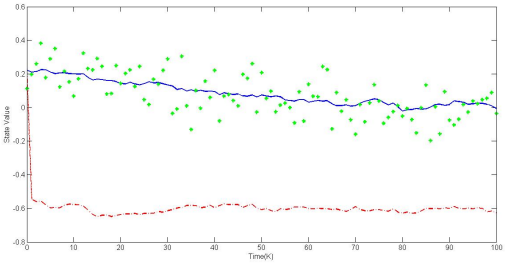
Tabla 5.5: *Resultados del MSE y la carga computacional en función del número de partículas para la simulación del Experimento 5.5 donde se emplea el filtro de partículas para estimar el modelo lineal del filtro de Kalman.*

Prestaciones del filtro de Kalman	Resultados obtenidos
Error Cuadrático Medio MSE	$10^{-4} \approx 0$
Coste computacional (segundos)	1.1

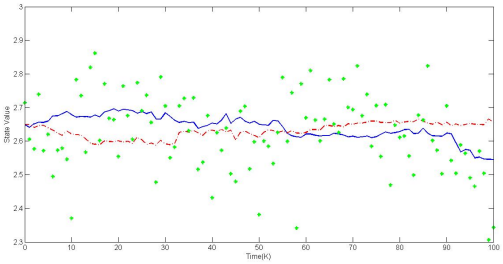
Tabla 5.6: *Resultados del MSE y la carga computacional para la simulación del filtro de Kalman durante el Experimento 5.5.*

Viendo los resultados obtenidos tras las simulaciones, se puede demostrar cómo el filtro de Kalman es un algoritmo óptimo bajo un modelo lineal y Gaussiano y cómo los métodos de Monte Carlo pueden llegar a conseguir resultados similares aumentando el coste computacional.

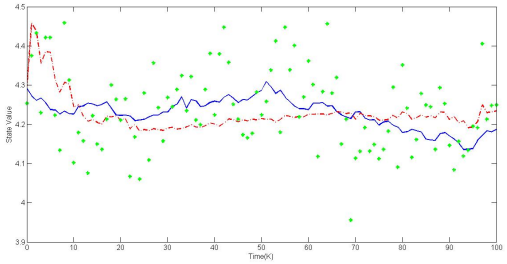
El MSE en promedio que se obtiene aplicando el filtro de Kalman tiende a cero y tiene un coste computacional de 1 segundo aproximadamente. Viendo la Tabla 5.5, se puede ver cómo no se llega a ese nivel de MSE con un coste computacional igual para el caso del filtro de partículas. Para que dicho método obtuviera las mismas prestaciones que el filtro de Kalman, tendrían que utilizarse más de 1000 partículas, provocando que el coste computacional se disparara a 600 segundos, siendo mucho mayor que el empleado por el filtro de Kalman.



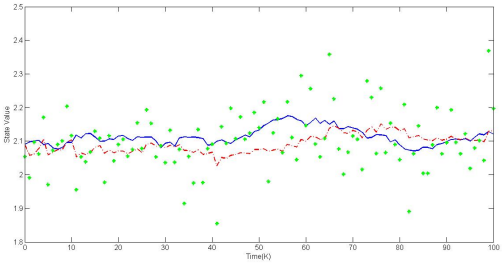
(a) 2 Partículas



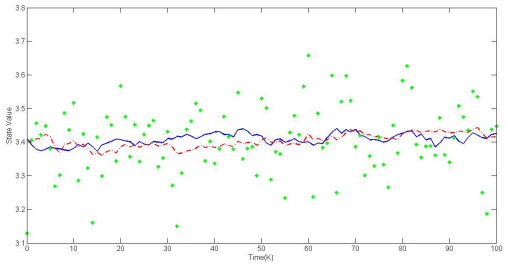
(b) 8 Partículas



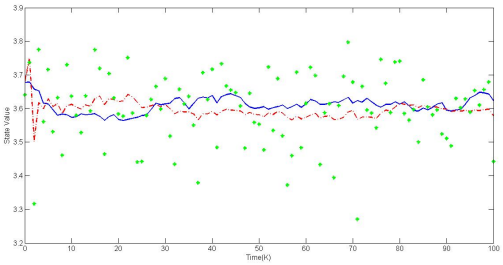
(c) 32 Partículas



(d) 128 Partículas

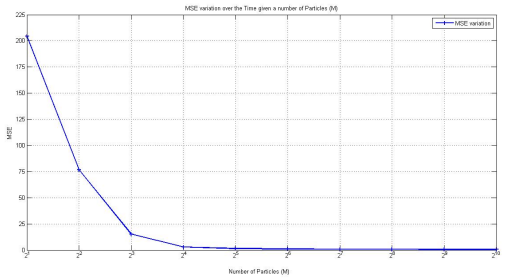


(e) 512 Partículas

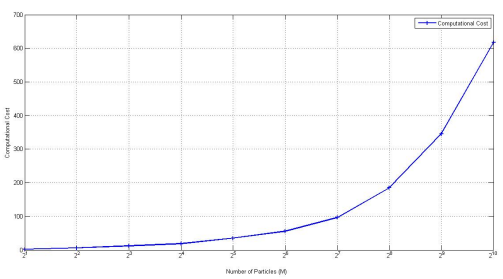


(f) 1024 Partículas

Figura 5.7: *Gráficas obtenidas durante el Experimento 5.5 para la estimación de los estados ocultos a posteriori de x_k, utilizando el filtro de partículas bajo un modelo lineal y Gaussiano.*



(a) MSE.



(b) Coste Computacional.

Figura 5.8: *Resultados obtenidos del Experimento 5.5 para el MSE y el coste computacional durante la simulación del filtro de partículas bajo el modelo lineal y Gaussiano de Kalman.*

Lógicamente si se está en una situación general y no se da por supuesta la existencia de un modelo lineal y Gaussiano, se podría utilizar el filtro de partículas ya que, a pesar de ser un algoritmo subóptimo, consigue unos resultados similares si hablamos de precisión en la estimación. No obstante habría que establecer un criterio entre dicha precisión y el coste computacional.

5.4. Estudio del filtro de partículas en un modelo de 3-D: El modelo de Lorenz

5.4.1. El modelo de Lorenz

El modelo de Lorenz se describe con un sistema de tres ecuaciones diferenciales en tres dimensiones y con una conexión con los fluídos de la atmósfera. Lo más importante es que exhibe dos características principales de los sistemas dinámicos no lineales: **la existencia de un atractor** y **caos** [12] [13]. La primera característica implica que el estado del sistema se aproximará a un conjunto especial, llamado atractor, dadas unas carácterísticas iniciales, tras un tiempo suficientemente largo, mientras que el segundo significa que, dos estados próximos, aunque sufran ligeras perturbaciones, convergerán en el tiempo. Por tanto, el estado del sistema en un futuro lejano es impredecible [12]. En esta sección, se mostrarán los resultados de la simulación del filtro de partículas dado el modelo tridimensional de Lorenz.

Se considera un problema de seguimiento de un estado del sistema tridimensional de Lorenz con ruido aditivo y dinámico, observaciones y ruido aditivo de medida. En concreto, se considera un proceso estocástico tridimensional $\{X(S)\}_{S\in(0,\infty)}$ que toma valores en $\mathbb{R}^3$, cuya dinámica se describe a través de un sistema de ecuaciones diferenciales estocásticas (5.5),

$$\begin{aligned} \partial X_1 &= -s(X_1 - Y_1) + \partial W_1 \\ \partial X_2 &= rX_1 - X_2 - X_1X_3 + \partial W_2 \\ \partial X_3 &= X_1X_2 - bX_3 + \partial W_3, \end{aligned} \tag{5.5}$$

donde $\{W_i(S)\}_{S\in(0,\infty)}$, $i = 1, 2, 3$ son procesos de Wiener independientes y unidimensionales y,

$$(s, r, b) = \left(10, 28, \frac{8}{3}\right),$$

son parámetros estáticos del modelo usados en la literatura dado que lideran el comportamiento caótico [13]. Aquí se usará una version discreta en el tiempo del último sistema usando un esquema Euler-Maruyama con paso de integración $\Delta = 10^{-3}$, quedando el modelo,

$$X_{1,n} = X_{1,n-1} - \Delta s(X_{1,n-1} - X_{2,n-1}) + \sqrt{\Delta}U_{1,n} \tag{5.6}$$

$$X_{2,n} = X_{2,n-1} + \Delta(rX_{1,n-1} - X_{2,n-1} - X_{1,n-1}X_{3,n-1}) + \sqrt{\Delta}U_{2,n} \tag{5.7}$$

$$X_{3,n} = X_{3,n-1} + \Delta(X_{1,n-1}X_{2,n-1} - bX_{3,n-1}) + \sqrt{\Delta}U_{3,n}, \tag{5.8}$$

donde $\{U_i(S)\}_{S\in(0,\infty)}$, $i = 1, 2, 3$ son secuencias independientes de variables aleatorias de tipo normal, idénticamente distribuidas, de media cero y varianza unitaria [57].

El sistema (5.6)-(5.8) se observa parcialmente cada 200 instantes discretos de tiempo. Específicamente, se almacena una secuencia escalar de observaciones $\{Y_t\}_{t=1,2,\ldots,T}$ de la forma,

$$Y_t = X_{1,200t} + V_t, \tag{5.9}$$

donde el ruido de observación $\{V_t\}_{t=1,2,\ldots,T}$, es una secuencia de variables aleatorias normales, idénticamente distribuidas, de media cero y varianza $\sigma^2 = \frac{1}{2}$. El conjunto $X_n = (X_{1,n}, X_{2,n}, X_{3,n}) \in \mathbb{R}^3$ es el vector de estados. El modelo dinámico dado por las ecuaciones (5.6)-(5.8), define la transición kernel $p(x_n|x_{n-1})$ y, las observaciones del modelo dadas por la ecuación (5.9), dan lugar a la función de verosimilitud

$$p(y_t|x_{1,200t}) \propto exp\left\{\frac{-1}{2\sigma^2}(y_t - x_{1,200t})^2\right\}. \tag{5.10}$$

El objetivo es el seguimiento de la secuencia de las medidas de la probabilidad conjunta a posteriori Π_t, $t = 1, 2, \ldots, T$ para $\left\{\hat{X}_t\right\}_{t=1,\ldots,T}$, donde $\hat{X}_t = X_{200t}$. Destacar que uno puede escoger una muestra $\hat{X}_t = \hat{x}_{t-1}$ realizando sucesivas simulaciones

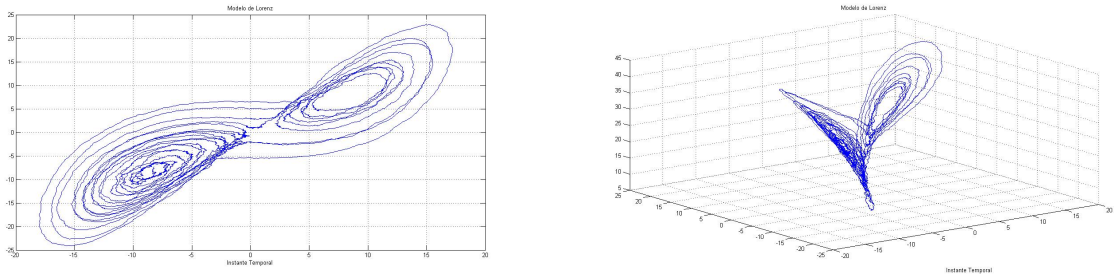
$$\tilde{x}_n \sim p(x_n|\tilde{x}_{n-1}), \quad n = 200(t-1)+1,,200t, \tag{5.11}$$

donde $\tilde{x}_{200(t-1)} = \hat{x}_{t-1}$ y $\hat{x}_t = \tilde{x}_{200t}$. La medida a priori de las variables de estado sigue una distribución normal, específicamente,

$$X_0 \sim N\left(x_*, \upsilon_0^2 \tau_3\right), \tag{5.12}$$

donde $x_* = (-5{,}9165; -5{,}5233; 24{,}5723)$ es la media y $\upsilon_0^2\tau_3$ es la matriz de covarianza de X_0, con $\upsilon_0^2 = 10$ y siendo τ_3, la matriz identidad tridimensional [57].

En la Figura 5.9, se puede visualizar el modelo de Lorenz real, conocido como "alas de mariposa"por la forma que dicho modelo toma. Dicha forma puede haber inspirado el nombre del **efecto mariposa** en la **Teoría del Caos**.



(a) Modelo de Lorenz 2D (b) Modelo de Lorenz 3D

Figura 5.9: *Modelo de Lorenz en 2D y 3D respectivamente.*

5.4.2. Simulación

Experimento 5.6 *El objetivo de la simulación es mostrar cómo el algoritmo estámdar BPF, permite estimar el modelo tridimensional de Lorenz. Con dicho propósito, se aplica el BPF para realizar un seguimiento de las medidas de la probabilidad a posteriori del sistema dado por las ecuaciones* (5.6)-(5.8)*, implicadas en el modelo tridimensional de Lorenz comentado anteriormente [57].*

Se genera una secuencia de $T = 2000$ *observaciones sintéticas,* $\{y_t; t = 1, \ldots, 2000\}$*, distribuidas en intervalos de 400 segundos (en tiempo continuo), correspondientes a* $4x10^5$ *instantes discretos de tiempo en el esquema de Euler-Maruyama (una observación cada 200 iteraciones).*

Número de partículas	MSE (en promedio)	Coste computacional (segundos)
2000	1.5	28.7
1000	2.4	15.8
500	5.6	7.0
250	61.6	3.8
125	80.8	3.1
60	108.1	2.4

Tabla 5.7: *Resultados obtenidos durante el Experimento 5.6 para el MSE en función del número de partículas.*

Dado que la aproximación de tiempo discreto de las ecuaciones (5.6)-(5.8) *es* $n = 200t$*, el paso del resampling se ejecutará cada 200 iteraciones [57].*

La idea de este nuevo experimento es similar a la del Experimento 5.1: encontrar el número de partículas ideal para poder realizar una estimación a posteriori de los estados de las variables ocultas del sistema. En el caso del modelo de Lorenz, dado que es un sistema tridimensional, se busca estimar los estados de $X_{1,n}, X_{2,n}$ *y* $X_{3,n}$*.*

Inicialmente se ha simulado el algoritmo con tan solo 4 partículas. Después se ha ido aumentando dicho número para ver la mejora en la precisión y en la eficiencia del algoritmo. En la Figura 5.10 se puede ver en 2D[3] la repercusión del aumento del número de partículas a la hora de estimar los estados ocultos de las variables $X_{1,n}$ y $X_{1,n}$ del sistema (5.6)-(5.8).

Para terminar con el estudio, se presenta la Tabla 5.7, donde queda presente la influencia del número de partículas escogido para realizar la estimación, en el error cuadrático medio MSE[4].

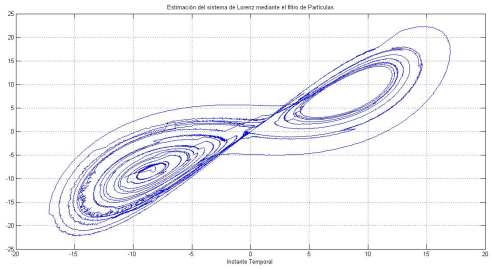
5.4.3. Conclusiones

En el Experimento 5.6 se ha realizado un estudio de la influencia del número de partículas en la estimación de los estados ocultos del sistema diferencial y tridimensional de Lorenz, para determinar la robustez del filtro de partículas (BPF) en una situación de varias dimensiones y mayor complejidad. Se ha visto como el BPF realiza una estimación muy precisa para 2000 partículas aproximadamente, tras probar heurísticamente en varias simulaciones.

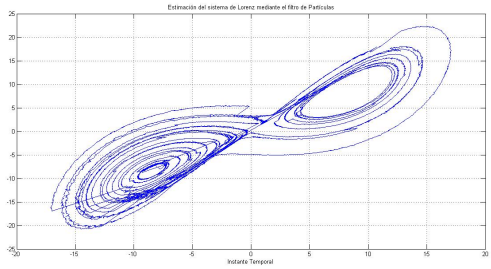
Es importante destacar, por tanto, la gran robustez que tiene el filtro de partículas cuando la situación deja de ser Gaussiana y lineal. Sin embargo, no hay que olvidar que los métodos de Monte Carlo son métodos subóptimos puesto que no cumplen el Principio de Ortogonalidad necesario para minimizar el MSE, como ocurría en el caso del filtro de Kalman. En el caso de los métodos de Monte Carlo, se realizan aproximaciones muestrales de distribuciones.

[3]No se han mostrado gráficas en 3D de la simulación, dado que es más complicado ver la mejora de la estimación. El Experimento 5.6 se ha realizado en 2D, mostrando por tanto los resultados de las estimaciones de las variables $X_{1,n}, X_{2,n}$. La estimación de $X_{3,n}$, no se muestra en ninguna gráfica aunque durante la simulación fue necesario su cálculo.

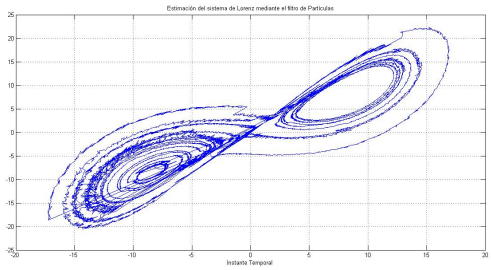
[4]Es importante destacar que el coste computacional en el Experimento 5.6 es mucho menor que en el resto de experimentos dado que no se tenía que ejecutar una gráfica en todas las iteraciones de cada simulación, como se ha necesitado hacer en otros experimentos, si no que sólamente se ha generado una gráfica por simulación.



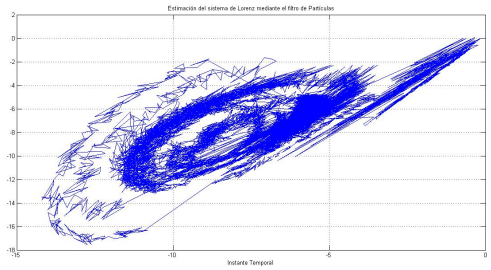
(a) 2000 Partículas



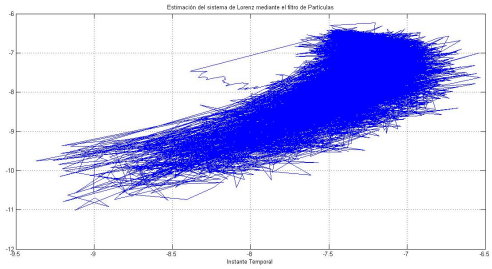
(b) 1000 Partículas



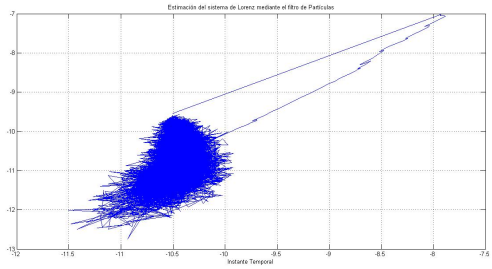
(c) 500 Partículas



(d) 250 Partículas



(e) 125 Partículas



(f) 60 Partículas

Figura 5.10: *Gráficas recogidas durante la simulación de la estimación de las variables $X_{1,n}$ y $X_{2,n}$ del modelo de Lorenz, haciendo uso del filtro de partículas, para el Experimento 5.6.*

El problema fundamental reside, como ya ocurría en los experimentos vistos, en establecer el mejor criterio entre el número de partículas y el coste computacional. Se ha demostrado cómo, a medida que el número de partículas aumenta, el coste computacional lo hace también, resultando en muchos casos una barrera en situaciones de tiempo real.

Elegir el número de partículas óptimo no es fácil, y a día de hoy se siguen realizando investigaciones para su hallazgo. En este documento se ha estudiado la búsqueda de este número de partículas de manera heurística y, manteniendo dicho valor durante toda la simulación. En [57] [58], se propone adaptar el número de partículas de manera online durante la simulación. Este proceso disminuiría el tiempo de ejecución, ya que solamente en aquellos momentos donde se necesitasen más partículas, la carga computacional aumentaría. Por tanto, se estaría aumentando la eficiencia del algoritmo. Otro camino que se ha propuesto en [30] [31] y en este documento como posible solución ha sido el empleo del ESS, gracias al cuál, comparando con un umbral determinado, se podía intuir si el *resampling* era o no necesario en cada una de las ejecuciones, reduciendo por tanto, el coste computacional.

5.5. Estimación y presupuesto del proyecto

Este proyecto se basa en el desarrollo de algoritmos de estimación de modelos espacio temporales. Dichos algoritmos se pueden implementar en diversas aplicaciones como se ha comentado anteriormente. En esta sección se realizará el estudio del presupuesto del proyecto así como la planificación que se ha seguido para la consecución de los objetivos principales.

En la Tabla 5.8 se puede ver la estimación de la duración total del proyecto. Dicha estimación se realizó previamente a la etapa de documentación con el objetivo de establecer hitos para la consecución de los objetivos del proyecto.

Nombre de la Etapa	Número de semanas
Documentación básica del problema	3 semanas
Desarrollo del filtro de Kalman (1D y 2D)	4 semanas
Periodo de pruebas de los algoritmos	4 semanas
Análisis de resultados del filtro de Kalman	1 semana
Documentación y estudio de los métodos de Monte Carlo	2 semanas
Desarrollo de un filtro de Partículas	2 semanas
Periodo de pruebas del algoritmo	2 semanas
Análisis de resultados del filtro de Partículas	2 semanas
Estudio sobre el modelo de Lorenz	1 semana
Implementación del modelo en el filtrado de Partículas	1 semana
Análisis de resultados obtenidos tras el experimento	1 semana
Curso sobre el software LAtex	2 semanas
Realización de la memoria del proyecto	8 semanas
TOTAL	**33**

Tabla 5.8: *Estimación del proyecto.*

La planificación se dividió en tres tipos de etapas diferentes: por un lado la etapa de documentación y estudio de los algoritmos, por otro lado el desarrollo y la implementación en Matlab de los mismos y por último, el análisis de resultados.

En la Tabla 5.9 se recoge el presupuesto del personal detallado del proyecto teniendo en cuenta que se trata de un proyecto de investigación. Para el cálculo del coste por horas del personal, se ha tenido en cuenta el salario aproximado de un investigador en formación. Dicho salario está comprendido entre los 900-1000 €/mes. De esta manera, haciendo un cálculo del número de horas empleadas a la semana aproximado (80horas) multiplicado por el número de semanas de un mes (4 semanas), se obtiene que el coste por hora del personal es de 7€/hora.

En la Tabla 5.10 se recogen los costes asociados a la amortización de los equipos y herramientas empleadas durante el proyecto. El cálculo de la amortización sigue la ecuación (5.13).

Personal	Grado	Categoría	Nº horas	Coste por hora (€)	Coste final (€)
David Martín Gutiérrez	Investigador en formación	Análisis de los modelos.	75	7	525
		Desarrollo de algoritmos y experimentos.	360	7	2.520
		Documentación y redacción.	250	7	1.750
Total	-	-	**685**	-	**4.795**

Tabla 5.9: *Presupuesto del personal.*

Descripción	Coste(€)	Tiempo de uso (meses)	Factor de utilización (0.1-1)	Periodo de vida útil (meses)	Coste imputable(€)
Portátil Sony Vaio PCG-71811M	850	48	1	96	425
Matlab (Standard Version)	2000	6	0.9	Licencia vitalicia	0
Total	-	-	-	-	**425**

Tabla 5.10: *Amortización de los productos.*

$$\text{Coste Amortizacion} = \frac{\text{Coste } x \text{ Tiempo de uso } x \text{ Factor de Utilización}}{\text{Periodo de vida útil}} \tag{5.13}$$

Categoría de presupuesto	Coste (€)
Personal	4.795
Amortización	425
TOTAL	**5.220**

Tabla 5.11: *Presupuesto total.*

Por último, el cálculo total del presupuesto del proyecto se recoge en la Tabla 5.11, donde se suma el coste total del personal así como el coste de amortización. El presupuesto para este proyecto está tasado en 5.220€.

Capítulo 6

Final conclusions

6.1. Project Conclusions

In this section we will review all the conclusions which have been obtained during the development of the project. Especially we will focus on the results of the experiments.

First of all, we have demonstrate the optimal performance of the kalman filter in Experimento 3.1-3.4 which were done in **Chapter 3**, *El filtro de Kalman*. In these cases, the model was suposed to be linear and Gaussian and only the Kalman filter could minimise the minimum square error, so it achieved the optimal solution for the estimation of systems states that can only be observed indirectly. In fact, in Experimento 5.5, a Particle filter was used to estimate the Kalman filter model and we have seen how this algorithm needed much more time to work than the Kalman filter did in order to have the same MSE performance.

However, if the system does not fit nicely into a linear or Gaussian model, the Kalman filter performance is not so accurate. There are some alternative solutions for that such as the Extended Kalman filter (EKF), but it does not work very well in many cases so, in this project, we have considered other options like Monte Carlo methods; especially we have focused on the Sequential Monte Carlo methods, in particular the Particle filter.

We have seen in **Chapter 4**, *Métodos de Monte Carlo*, that Particle filter makes the work of the Kalman filter in those nonlinear and/or non-Gaussian environments. The main difference is that, the Particle filter uses simulation methods to generate estimates of the state and measures instead of deriving analytic equations as the Kalman filter does. The basic concept of the method is that the approximation of the posterior probability of the state is carried out by the generation of a huge number of weight particles, using Monte Carlo methods. The particles are no longer uniformly distributed over the state but instead concentrated in regions of high probability.

In **Chapter 4**, *Métodos de Monte Carlo*, we have explained that it is necessary to resolve the degeneracy problem of some Monte Carlo methods with the resampling technique. However the idea of implement the resampling step in each iteration is under discussion. We have introduced the Effective Size Sampling (ESS) in order to demonstrate that it is not always necessary to implement the resampling step. Only when the variance of the particles weight starts to rise, it would be fine to do it. However, we have to measure the ESS with a threshold. Thus in Experimento 5.2, the ESS threshold is modified from zero to one, in order to present the best criteria among the number of particles, the threshold value, the MSE and the computational cost. In this situation we have seen that the bigger the number of particles and the threshold are, the better of the Particle filter performance is, but the computational cost increases with the rise of the threshold.

Over the **Chapter 5**, *Desarrollo de un filtro de partículas en MATLAB*, some experiments were development with the goal of verifying the great performance of the Particle filter

in nonlinear and/or Non-Gaussian situations. In Experimento 5.1, the number of particles is increased every iteration in order to explain the positive effect that it has in the MSE. The worst part is that, again, the bigger the number of particles is, the more expensive the computational cost can be. So, good criteria between the number of particles and the MSE have to be established.

The criteria for this experiment are established in a heuristic way. For example, in this case, after doing a lot of simulation with different parameters, we have seen that using 2^8 particles and taking a threshold between 0,7 – 0,8, the MSE is lower and the computational cost is not too large. If the threshold takes a value really close to zero, the computational cost decreases considerably but the performance of the algorithm is worse.

Finally, we have concluded that the value of these parameters should depend on the features of the final application. It will depend on the need of more accuracy in the estimation or more velocity. Then, a new experiment is taken place in the project. In Experimento 5.3, the final goal is to investigate if the resampling can be implemented according to a fix window w. In this way if w gets higher so the resampling step is implemented fewer times and viceversa. Looking at the final results, we can see that there are not any relationship between the number of particles, the size of w and the performance of the Particle filter. Finally, Experimento 5.2 has more sense than Experimento 5.3 because in the first one a statistic study is made and it is an adaptative method and in the second one, you cannot achieve the same results of performance so, it is the worst method.

Different resampling methods are studied in this project. The idea of Experimento 5.4 is to evaluate the influence of these methods in the performance of the algorithm. After tested all of them in the Particle filter simulation, we have concluded that using the stratified or the systematic method, the final results of the tests are better because of the lower computational cost. The MSE is similar in all of the studied methods.

The final experiment, which is developed in Experimento 5.6, has as a goal, the evaluation of the estimation performance of the Particle filter in a more complicated situation. In this case, the model implemented is the one studied in the Chaos theory researched by Edward Lorenz. This is a three-dimensional dynamical system generated by three differential equations. The Particle filter is used to estimate the states of this complex system. We have seen how accurate the performance of the Particle filter with 2000 particles is. However, when the number of particles goes down, the performance is worse, but we did know it because of the rest of the experiments. The important thing is that Particle filter can estimate a complicated system provided that the number of particles approaches to infinite.

Nowadays, the computational cost problem that Monte Carlo methods had can be resolve with the new advances of technology.

6.2. Future Lines

Nowadays, analysis and processing data technology is making a huge progress. There are a lot to research in the Monte Carlo methods field yet, in order to improve the performance of their methods. In the case of the Particle filtering, there are new investigations about new techniques for the resampling step whose features are more powerful than the traditional ones. The better the resampling is, the better the final performance of the Particle filter will be.

What is more, setting up a fixed number of particles over the simulation does not have any sense because the complexity of the estimation is not kept in mind. The idea of this is that the more complexity of the estimation is, the more number of particles should be used. One of the proposal solutions would be the online adapting of the number of particles over the simulation. In this case, an evaluation of the MSE could be done every iteration and if

the MSE is less than in the previous iteration, the number of particles could be decreased in the followings. On the other hand if the MSE is higher than in the last iteration, the number of particles should be increased in the next one in order to improve the performance. A more effective but difficult way of doing the online adapting of the number of particles is proposed in [57] [58].

Taking on these improvements, the performance of the particle filter would be considerably increased and it would be possible to develop more powerful applications in the future.

Apéndice A

Project Summary

The main goal of this project is to make a non-dificult Kalman filter and Particle filter tutorial in order to understand the great advantages that these algorithms provide to many applications. Some methods which are used to make estimations and predictions of the states of dynamic systems will be explained.

Since the great discovery of the Kalman filter by E.R. Kalman, technology has made progress in several fields and in the last years, a lot of prediction and estimation applications have been developed including filtering noisy signal, object tracking, navigation systems, etc.

Many problems in engineering require the estimate of the state of a system which changes over time using a set of noisy measurements made on the system. In this project we have focussed on the state-space approach to modelling dynamic systems.

In order to understand the state-space models, it is neccesary to know about the stochastic processes. A stochastic process is a family of random variables X_t, where t is a parameter running over a suitable index set T. In this project, t represents time. Over time X can take many values which define its trayectory. These trayectories are the states that we want to estimate. There are a lot of ways to classify these processes but we have focussed on two. The first one depends on the discrete or continuous form for both the time index T and the states. In this category we have focussed on the discrete states and discrete time which is called discrete state process. In the second one, the stochastic processes are classified according to their probabilistic random variables features. In this category, the Markov processes are the most important ones for this project. The stochastic processes are used in the modelling estimation problem which is resolved in this project.

First of all, we will study the Kalman filter. In 1960, R.E. Kalman published his paper describing a recursive solution to the discrete linear filtering problem. The Kalman filter addresses the general problem of trying to estimate the state $x \in \mathbb{R}^n$ of a discrete-time controlled process that is governed by some stochastic differential equation.

$$x_k = Ax_{k-1} + w_{k-1} \tag{A.1}$$

In order to do the estimate, we have some measurements which we will consider as $y \in \mathbb{R}^m$ and whose model is given by

$$y_k = Hx_k + v_k. \tag{A.2}$$

It is important to say that both the state model and the measurement model are perturbed by aleatory, white and Gaussian noises which are represented by v_k and w_k.

In the Kalman filter process there are two work phases: prediction step and update step. In the first one, the current state and the error covariance are projected forward in time and their estimation is used to obtain the *a priori* estimates for the next time step.

$$\hat{x}_k^- = A\hat{x}_{k-1} + w_{k-1} \tag{A.3}$$

$$P_k^- = AP_{k-1}A^T + Q_k. \tag{A.4}$$

In the second one, a new measurement is incorporated into the *a priori* estimate in order to obtain an improved *a posteriori* estimate. It can seen that the first step is a *predictor* and the second one a *corrector*. This recursive nature is one of the most interesting features of the Kalman filter.

$$\hat{x}_k = \hat{x}_k^- + K_k\left(y_k - H\hat{x}_k^-\right) \tag{A.5}$$

$$K_k = \frac{P_k^- H^T}{HP_k^- H^T + R_k} \tag{A.6}$$

$$P_k = (I - K_k H)P_k^-. \tag{A.7}$$

The Kalman filter estimator criteria should accomplish two requirements in order to be an accurate estimator. In the first place, the average value of the state estimate should be equal to the average value of the true state. In the second place, the state estimate should not vary so much from the true state, thus we want to find an estimator with the smallest possible error variance. The Kalman filter satisfies these two criteria but only if the model is linear and Gaussian and the noises are independent, white and Gaussian. The Kalman gain K_k is the neccesary tool to correct the *a priori* estimate and then, minimise the covariance error P_k.

At the end of the Kalman filter explanation, some experiments 3.1-3.4, have been done in order to evaluate the performance of the Kalman filter in a 1D and 2D situations. In these cases, the model was suposed to be linear and Gaussian and only the Kalman filter could minimise the minimum square error (MSE), so the Kalman filter achieves the optimal solution for the estimation of system states that can only be observed indirectly.

With the aim of demonstrating the optimal solution of the Kalman filter, in Experimento 5.5, a Particle filter was used to estimate the Kalman filter model and we have seen how this algorithm needed much more time to work than the Kalman filter did in order to have the same MSE performance.

The main problem of the Kalman filter resides in that there are many situations where the models are nonlinear and/or Non-Gaussian. In these cases, the optimal solution proposed by Kalman does not work and we have to use another tool in order to estimate the hidden states.

Since it is impossible to obtain analytic solutions to the inference problems of interest in many situations, some researchers have investigated different algorithms from Kalman filter. One standard approximation method is called Extended Kalman Filter (EKF). This method is the nonlinear version of the Kalman filter which linearizes an estimate of the current mean and covariance. The main problem of this algorithm is that if the initial estimate of the state is wrong, or if the process is modelled incorrectly, the filter may quickly diverge, owing to its linerization. In this project the EKF has not been studied in depth because there are better solutions for nonlinear and Non-Gaussian situations.

As is known, real data can be more complex and sometimes elements of non-Gaussianity or high dimensionality can appear. In **Chapter 4**, Monte Carlo methods are studied in order to resolve these environments. Specially, Sequential Monte Carlo methods (SMC) will be studied. The main goal of SMC, is to approach posterior distributions by a large collection of N random weighted samples (particles). The main advantage of these methods is that under weak assumptions, they provide asymptotically consistent estimates of the target distributions of interest so, if N approaches to infinity, the estimate is stronger.

First of all, we will talk about the Standar Monte Carlo methods and their disvantages because of the high dimensionality and the unsuitability in recursive estimation problems.

In order to resolve these problems, two different methods have been researched: Batch Importance Sampling (BIS) and Sequential Importance sampling (SIS). The first one is not adequate for recursive estimation so we have focussed on the second one. The solution of SIS involves a good choice of an importance distribution. The SIS method has two principle steps: particle propagation and weight computation.

Both IS and SIS suffer the problem which is known as *degeneracy problem*. The problem is caused because the estimate variance is increased exponentially with n. The main effect of that problem is shown in the particle weights because most of the particles have a weight which approaches zero, and only a few particles have a non-zero weight. In order to resolve this problem, some researchers have developed a technique called *resampling*. The algorithm which uses this technique is called Bootstrap Particle filter (BPF), and in the final chapter, we will focus on that one in order to do the software development.

The BPF has three principle steps: particle propagation, weight computation and resampling. The first two amount to the generation of particles and assignment of weights and the last one replaces one set of particles and their weights with another set. The resampling is an essential step in order to prevent the aforementioned problems.

The aim of the resampling is to prevent the degeneracy of the propagated particle. The idea is that the particles with large weights are more likely to dominate in the sampling distribution than the particles with small weights. Consequently, more particles will be generated in the region of large weights. However, resampling is not always perfect and undesired effects, such as the impoverishment of the sampling and the increase of the computional cost, can be introduced. The first undesired effect is caused because the diversity of the particles is reduced. The second one is casused because the resampling step involves more computational operations which increase the implementation time of the algorithm.

In this project we have studied several resampling methods which have been developed to resolve the indesired effects of resampling. In particular, we have studied the traditional methods including multinomial, accumulative, residual, sistematic and stratified methods. The impoverishment problem is decreased using some of these methods but we will evaluate their computational complexity, which can be a problem in the final development of the algorithm.

As we have said, resampling removes particles whose weights are low and multiplies particles with high weights and, in some cases, can be harmful for the estimates. So, additional variance is introduced in the system. In this way, it is important to analyze when it is necesssary to implement the resampling step. If particles have normalised weights with a small variance, then the resampling step is not necessary. Consequently, we will only carry out the resampling step if the variance of the normalised weights is superior to a threshold. This is what is known as Effective Sample Size (ESS) and it is given by

$$ESS = \left(\sum_{i=1}^{K} \left(w_t^{(i)} \right)^2 \right)^{-1}. \tag{A.8}$$

The ESS takes values from 1 and N and resampling will be carried out when the threshold is typically N/2. In this project we have normalised the ESS in order to make the study easier, so,

$$\bar{ESS} = \frac{ESS}{M}, \tag{A.9}$$

where M is the number of particles. Thanks to that, the $\bar{ESS}$ takes values from 0 to 1 and we will analyze, in the simulation chapter, what would be the best value in order to achieve the best performance.

Finally in **Chapter 5**, *Desarrollo de una filtro de partículas (SMC) en MATLAB*, we have developed a Bootstrap Particle filter (BPF) with the aim of evaluating the minimum Square Error (MSE) performance in several situations.

The main goal of this chapter is the achievement of the best performance in order to estimate the hidden states in a nonlinear or Non-Gaussian situation. First of all, we have introduced the problem to be resolved and the algorithm description. Then we have done the first simulation in MATLAB. The main idea of Experimento 5.1 is to increase the number of particles in order to improve the performance of the Particle filter. We have seen how if the number of particles is increased, the performance is better and consequently, the MSE is decreased. The negative side is that the computional cost increases if the number of particles does so. Because of this fact, we have to establish a good criteria in order to achieve the best solution. Finally we have concluded that with 2^8 particles, the Particle filter performance is sufficiently accurate and the MSE is quite low.

The next experiment was developed to evaluate the resampling role in the final performance of the Particle filter. In Experimento 5.2, the main idea was to increase the threshold to compare the ESS value from 0 to 1, in order to evaluate the MSE and the computational cost according to the number of particles used. When the threshold value increased, the MSE decreased with a lesser number of particles. This means that the algorithm performance is considerably good. The worst thing is that if the threshold takes a bigger value, the resampling step is carried out more times so the computational cost is increased. On the other hand, if the threshold value approaches zero, the performance is not so good because the resampling step is not carried out so often, but the computational cost is low. Once again having a good criteria becomes a necessary task but in this case among the threshold to compare the ESS, the number of particles, the MSE and the computational cost. We have concluded that with more than 2^7 particles and with a threshold between $0{,}7 - 0{,}8$ to compare the ESS, the final performance of the Particle filter is sufficiently accurate.

Then we did a similar experiment in Experimento 5.3, but this time, the resampling step was carried out each w iteration, where w was the window which pointed out if the resamplig had to be done or not. If w is greater, the resampling step is carried out few times and viceversa. The main objective of this experiment is to compare the results with Experimento 5.2 in order to evaluate the influence of the ESS in the final performance of the Particle filter. After doing this experiment, we have seen how the Particle filter performance is worse than the ESS one and the principle cause is because the window experiment is not adaptative and sometimes it is necessary to implement the resampling step but in Experimento 5.3 it is not carried out and the performance does not work well. Maybe on some occasions, this situation could be better than the ESS one, but not in a real time application or in some applications which involve an accurate estimate.

The next experiment has to do with the resampling methods aforementioned. The main idea of this experiment is to evaluate the performance of the Particle filter according to the resampling method used. After evaluating all the tradicional resampling methods in Experimento 5.4, we have seen how some traditional methods are better than others just because the computational cost is decreased and the MSE is equal in all of them. We have concluded that the sistematic and the stratified methods are the most optimal resampling methods in order to achieve the best solution to the problem.

We have seen how optimal the Kalman filter solution was. Monte Carlo methods are suboptimal solutions; nevertheless they can achieve the same performance as a Kalman filter in linear and Gaussian environments. In order to demonstrate that, in Experimento 5.5, we have simulated the Particle filter in the Kalman filter model and we have compared the results with the Kalman filter ones. If we used 2^10 particles or more, we could obtain the same performance as the Kalman filter apart from the computational cost which in the

Particle filter case, was much higher than in the Kalman filter one.

The final experiment developed in Experimento 5.6 has the aim of evaluating the performance of the Particle filter in a more complex situation. The model used in this experiment was the one belonging to the Chaos Theory, which was discovered by Edward Lorenz. This model consists in a system of three differential equations which is a more complex situation than the model used in the rest of the experiments. The three-dimensional model has been evaluated by the Particle filter according to a number of particles. We have seen how with 2000 particles the estimate of the Lorenz model is very accurate so the MSE is low. Once again, the computational cost is increased. If we decreased the number of particles, the final performance would be worse.

After doing all the experiments, we can conclue that the more complex the situation is, the greater number of particles is needed to achieve a good performance. In real time situations, the computational cost cannot be too high, so, the number of particles should not increase too much. However, because of great tecnological advances, the computational power has been increased so in many cases a great number of particles could be used in order to get better results.

Nowadays, both Kalman filter and Sequential Monte Carlo Methods (SMC) are used very much in several fields, including engineering, econometrics, artificial vision, robotics, biological science or tracking. This is the reason why they are very attractive for many researchers and companies. Many of them are improving the traditional resampling methods in order to decrease the computational cost and the impoverishment problem. Moreover, there are several things that can be improved in Particle filtering such as a good choice of the number of particles and of the resampling method. In this project we have done it in a heuristic way, but there are few researchers that are testing how to adapt the number of particles in the simulation. This would be a great advance because using a fixed number of particles during the whole simulation can be a problem in a real time application because of the computational cost. Both online adapting of the number of particles and the improvement of the traditional resampling methods would be two ways to increase the good performance of these methods in several applications which could be very useful in the future.

Bibliografía

[1] A. Doucet & A. M. Johansen, "A Tutorial on Particle Filtering and Smoothing: Fifteen years later", Instituto de Matemáticas Estadísticas, Tokyo, Japón. Departamento de Estadística, Universidad de Warwick, Coventry, UK, Diciembre 2008.

[2] A. Doucet, N. De Freitas & N. Gordon, *An Introduction to the Sequential Monte Carlo Methods*. New York, NY, USA, *Springer*, 2001.

[3] A. Doucet, S. Godsill & C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering", Statistics and Computing, vol. 10, no. 3, pp. 197-208, Julio 2000.

[4] A. H. Mohamed, & K. P. Schwarz. *Adaptive Kalman filtering for INS/GPS*, *Journal of geodesy*, 73(4), pp. 193-203, 1999.

[5] A. Kalaitzis, "Monte Carlo Methods for Computer Vision", Escuela de Informática, Universidad de Edimburgo, UK, Mayo 2009.

[6] A. Kelly, "A 3D State Space Formulation of a Navigation Kalman Filter for Autonomous Vehicles", The Robotics Institute, Universidad de Carnegie Mellon, USA, Mayo 2006.

[7] A. S. Ramírez, "El filtro de Kalman", Departamento de Investigaciones Económicas, Banco Central de Costa Rica, San José, Costa Rica, Julio 2003.

[8] A. T. Cemgil, & B. Kappen, "Monte Carlo methods for tempo tracking and rhythm quantization". Universidad de Nijmegen, Holanda, *Journal of Artificial Intelligence Research*, 18(1), pp. 45-81, 2003.

[9] B. R. Murty & W.T. Federer, "Use of Kalman Filter. Part II. Applications in Biology", Departamento de biometría y estadística, Universidad de Cornell, Ithaca, USA, Marzo 2001.

[10] C. J. Mode, *Applications of Monte Carlo Methods in Biology, Medicine and other Fields of Science*, *InTech*, 2011.

[11] D. López Rincón. (Abril 2016) Internet de las cosas y Big data. [Online]. Disponible en: http://tecnologia.elderecho.com/tecnologia/internet_y_tecnologia/Internet-cosas-Big-data_11_945430003.html.

[12] D. T. Pham, "Stochastic Methods for Sequential Data Assimilation in Strongly Nonlinear Systems", *Monthly Weather Review*, vol. 129, no. 5, pp. 1194-1207, Mayo 2001.

[13] E. N. Lorenz, "Deterministic Nonperiodic Flow", Journal of the Atmospheric Sciences, vol. 20, no. 2, pp. 130-141, Marzo 1963.

[14] F. Dellaert, "A Sample of Monte Carlo Methods in Robotics and Vision", Instituto de tecnología de Georgia, Atlanta, USA.

[15] F. Montes Suay, "Procesos estocásticos para ingenieros: Teoría y aplicaciones", Departamento de Estadística e Investigación Operativa, Universidad de Valencia, Valencia, España, 2011.

[16] F. V. Lima (2008). Autocovariance Least-Squares (ALS) Package. Departamento de Ingeniería Química, Universidad de Wisconsin, Madison, WI, USA. [Online]. Disponible en: http://jbrwww.che.wisc.edu/software/als/index.html.

[17] G. K. Pasricha, "Kalman filter and its economic applications", Universidad de California, USA, 2006.

[18] G. Welch & G. Bishop, "An introduction to the Kalman Filter", Departamento de Ciencias de la Computación, Universidad de Carolina del Norte, Chapel Hill, NC, USA, 2001.

[19] I. Araguás Fuentes. (Octubre 2014) Big Data: cinco grandes retos en seguridad y privacidad. [Online]. Disponible en: http://www.valoresdigital.es/big-data-cinco-grandes-retos-en-seguridad-y-privacidad/.

[20] I. Molina Peralta & R. Fried, "Procesos Autorregresivos", en Series Temporales, Departamento de Estadística, Universidad Carlos III, Madrid. Departamento de Técnicas Estadísticas, Universidad de Dortmund, Dortmund, Alemania.

[21] I. Molina Peralta and R. Fried, "Introducción a los procesos estocásticos", en Series Temporales, Departamento de Estadística, Universidad Carlos III, Madrid. Departamento de Técnicas Estadísticas Universidad de Dortmund, Dortmund, Alemania.

[22] J. A. Camarena Ibarrola, "El filtro de Kalman", Universidad Michoacana de San Nicolás de Hidalgo, Morelia, México.

[23] J. Ancizar Castañeda Cárdenas, A. Nieto Arias & V. A. Ortiz Bravo, "Análisis y Aplicación del Filtro de Kalman a una señal con ruido aleatorio", Universidad Tecnológica de Pereira, Pereira, Colombia, Abril 2013.

[24] J. D. Hamilton, "State-Space Models", in: R. F. Engle & D.L McFadden (Eds.), *Handbook of Econometrics*, Volumen IV, North-Holland, Amsterdam, 1994. Chapter 50.

[25] J. F. G. de Freitas, "Bayesian Methods for Neural Networks", Departamento de Ingeniería, Trinity College, Universidad de Cambridge, Cambridge, UK.

[26] J. L. Romero Palma, "Introducción a los procesos estocásticos", Matemáticas, Universidad Nacional Abierta, 2015.

[27] J. M. Marín Diazaraque, "Introducción a los procesos estocásticos", Departamento de Estadística, Universidad Carlos III, Madrid, España, 2011.

[28] J. Wendel, C. Schaile & G.F. Trommer, "Direct Kalman Filtering of GPS/INS for Aerospace Applications, Universidad de Karlsruhe, Alemania.

[29] K. M. Hangos, R. Lakner, & M. Gerzson. *Intelligent Control Systems: An Introduction with Examples*. New York: Kluwer Academic Publishers, 2001.

[30] L. Martino, V. Elvira, and F. Lozada, .Alternative Effective Sample Size measures for Importance Sampling", *IEEE Workshop on Statistical Signal Processing (SSP 2016)*, Mallorca, Spain, June, 2016.

[31] L. Martino, V. Elvira, & F. Louzada, .Effective Sample Size for Importance Sampling Based on the Discrepancy Measures", Signal Processing, vol. 131, pp. 386-401.

[32] L. Rincón, "Introducción a los procesos estocásticos", Departamento de Matemáticas, Facultad de Ciencias UNAM, Ciudad de México, México, 2008, pp. 27-39.

[33] M. C. Ruiz Abellón, "Procesos estocásticos", Departamento de Matemáticas y Estadística Aplicada, Universidad Politécnica de Cartagena, Cartagena, España.

[34] M. S. Grewal & A. P. Andrews, *Kalman filtering theory and practice using Matlab*. 2nd ed. Wiley-Interscience, John Wiley & Sons Inc, 2001.

[35] M. S. Grewal, V. D. Henderson & R. S. Miyasako, "Application of Kalman filtering to the calibration and alignment of inertial navigation systems", *IEEE Transactions on Automatic Control*, vol. 36, no. 1, pp. 3-13, 1991.

[36] M. Sanjeev Arulampalam, S. Maskell, N. Gordon & T. Clapp, "A tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking", *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp.174-188, Febrero 2002.

[37] N. J. Gordon, D. J. Salmond & A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation", en *IEE Proceedings - Radar and Signal Processing*, vol. 140, no. 2, Abril 1993.

[38] OBS Online Business School. (2015) En 2020, más de 30 mil millones de dispositivos estarán conectados a Internet. [Online]. Disponible en: http://www.obs-edu.com/es/noticias/estudio-obs/en-2020-mas-de-30-mil-millones-de-dispositivos-estaran-conectados-internet.

[39] P. Del Moral, "Nonlinear Filtering: Interacting Particle Resolution", Universidad Paul Sabatier, Toulouse, Francia, 1996.

[40] P. Glasserman, *Monte Carlo Methods in Financial Engineering*, *Springer*,Verlag, New York, 2004.

[41] P. M. Djurić, J. H. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. Bugallo & J. Míguez, "Particle Filtering", en *IEEE Signal Processing Magazine*, Septiembre 2003.

[42] P. Muse, E. López & L. Di Martino, "Filtro de Kalman. Tratamiento Estadístico de Señales", Departamento de Procesamiento de Señales, Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay Mayo 2015. [Online]. Disponible en: https://eva.fing.edu.uy/pluginfile.php/72056/mod resource/content/2/kalman handout.pdf.

[43] P. Pichler, "State Space Models and the Kalman Filter", Seminar paper prepared for 40461 vektorautoregressive Methoden by Prof. R. Kunst, Enero 2007.

[44] R. G. Brown & P. Y.C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering with MATLAB Exercises*, 4th ed. John Wiley & Sons Inc, 2012.

[45] R. Kleinbauer, "Kalman filtering implementation with Matlab", Universidad de Stuttgart, Stuttgart, Alemania, Noviembre 2004.

[46] R. Molina Soriano, "Bases del Filtro de Kalman", Departamento de Ciencias de Computación, Universidad de Granada, Granada, España.

[47] R. E. Kalman, "A new Approach to Linear Filtering and Prediction Problems", Transactions of the ASME–Journal of Basic Engineering, 82 (Series D): 35-45, 1960.

[48] S. Abad, "Estudio OBS: Big Data en Números", OBS Online Business School, 2014. [Online]. Disponible en: http://www.obs-edu.com/es/noticias/estudio-obs/el-volumen-de-datos-generado-por-smartphones-crecera-un-63-los-proximos-cuatro-anos.

[49] S. F. Schmidt, "Applications of State Space Methods to Navigation Problems", en *Advanced Control Systems*, vol. 3, pp. 293-340, C. T. Leondes (Ed.), Academic Press, New York, 1966.

[50] S. H. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering (Studies in Nonlinearity)*. USA: Westview Press, 2014.

[51] S. Pereira Ruiz, "Localización de Robots mediante el filtro de Kalman", Departamento de Ingeniería de Sistemas y Automática, Escuela Técnica Superior de Ingenieros, Universidad de Sevilla, Sevilla, España, 2010.

[52] Simulation and Optimization Research Laboratory (SimLab), "Importance Density Selection in Particle Filtering", Departamento de Ingeniería Industria, Universidad de Miami, Miami, FL, USA. [Online]. Disponible en: http://www.coe.miami.edu/simlab/pf_importance_density.html.

[53] T. Doan, R. B. Litterman & C. A. Sims, "Forecasting and Conditional Projection Using Realistic Prior Distributions", National Bureau of Economic Research (NBER) Working Papers, no. 1202, Septiembre 1983.

[54] T. Li, M. Bolic & P. M. Djurić, "Resampling Methods for Particle Filtering", *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 70-86, Mayo 2015.

[55] T. Li, S. Sun, T. P. Sattar & J. M. Corchado, "Fight sample degeneracy and impoverishment in particle filters: A review of intelligent approaches", Expert Systems with Applications, vol. 41, no. 8, pp. 3944-3954, Enero 2014.

[56] T. Li. (2015) Resampling Codes for Particle Filtering. [Online]. Disponible en: https://sites.google.com/site/tianchengli85/matlab-codes/resampling-methods.

[57] V. Elvira, J. Míguez & P. M. Djurić, "Adapting the Number Of Particles in Sequential Monte Carlo Methods through an Online Scheme for Convergence Assessment", arXiv preprint arXiv:1509.04879, Septiembre 2015.

[58] V. Elvira, J. Miguez, & P. M. Djurić, "A Novel Algorithm for Adapting the Number of Particles in Particle Filtering", en *IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM 2016)*, Rio de Janeiro, Brasil, Julio 2016.

[59] Z. Chen, "Bayesian filtering: From Kalman filters to particle filters, and beyond", Statistics, vol. 182, no. 1, 1-69, Febrero 2003.

[60] J. A. Gras, *Diseños de series temporales: técnicas de análisis* (Vol. 46). Edicions Universitat Barcelona, 2001.

[61] Math Insight, "The idea of a dynamical system". [Online]. Disponible en: http://mathinsight.org/dynamical_system_idea.

[62] E. M. Mancinelli "Proceso de Bernoulli", Departamento de Matemática: Escuela de Ciencias Exactas y Naturales, Universidad Nacional de Rosario, Argentina.

[63] (2016) Kalman Filter. [Online]. Disponible en: https://en.wikipedia.org/wiki/Kalman_filter.

[64] (2016) Proceso de Poisson. [Online]. Disponible en: https://es.wikipedia.org/wiki/Proceso_de_Poisson.

[65] E. Cheever, "State Space Representations of Linear Physical Systems". [Online]. Disponible en: http://lpsa.swarthmore.edu/Representations/SysRepSS.html.

[66] (Acceso junio 2016). Kalman Filter to Estimate a first Order System. [Online]. Disponible en: http://www.swarthmore.edu/NatSci/echeeve1/Ref/Kalman/Ex2ScalarKalman.html.