
# ENN586: Estimation and decision in a sequential setting.[†]

Jason Ford[‡]

VERSION: 0.5. July 2023.



**Abstract**

These notes provide a first look at estimation and decision problems in a sequential setting. These are problems where the aim is to extract improving meaning, or make decisions, from sequentially arriving measurements or observations. As our interest is robotics we focus on sequential measurements and dynamics described in discrete time.

This material for these weeks of ENN586:

- Introduction to estimation
- Principles of system identification
- The linear filtering problem (the Kalman filter)
- The discrete space filtering problem (the hidden Markov model (HMM) filter)
- The nonlinear filtering problem (the extended Kalman filter and more)
- The alert decisions in a sequential setting (when to stop the robot)
- Application example of integrated navigation

[†] ENN586: Decision and Control. School of Electrical Engineering and Robotics, Queensland University of Technology.

[‡]Please report errors within this document to j2.ford@qut.edu.au

# 1 Introduction to estimation

## 1.1 Modelling

Modelling is the process of building a description of the relation between the input and output of a system of sufficient fidelity for an intended engineering design purpose.

We internally understand our own knowledge of the world, and communicate our knowledge to others, through codified conceptual descriptions that can be called models of reality.

### 1.1.1 History of model and knowledge

- Modern (Western) science began in the "age of reason" (or "the Enlightenment"). Roots in key publications by Newton and Locke, 1686-9.
- Lead to empirical science's basic principle that knowledge should be extracted from observations.
- One important discussion was planetary motion (Galileo, Copernicus, Kepler, Newton, Gauss and Euler) where astronomical observations were used to create profound knowledge of our existence.

In this unit we focus on models of dynamical systems; that is, systems that evolve over time according to some rules. This early human journey to understand our night sky became the journey of building dynamic system models of planetary motion. The tools you learn in this unit could be tasked for this purpose.

### 1.1.2 Knowledge extraction

In modern language, we might say that three basic approaches to building models:

1. First principles: from laws of nature, or similar, then confirmed on data.
2. Black-box: Learnt from data within the informed choice of model or architecture class.
3. Model-free: Learnt hyper-parameter models containing very limited innate priors.

This unit will say some things about the second item (black-box model). There will be other units in your studies that involve the first and the third item.

## 1.2 Components of a system model

Three basic components of a system model are:

1. Input (a signal), $u$
2. System model (an operation), $f(\cdot)$
3. Output (a signal), $y$.

We often represent input and output signals as mathematical objects and we think of the system model as being a generalised function; that is, $y = f(u)$ where $u$ is the input, $f(\cdot)$ is the system and $y$ is the output. Generally, we often think of three components as being in a temporal relationship in which the input signal $u$ goes into the system $f(\cdot)$ which does an operation on the input signal to create the output $y$. In this way, the input is the independent variable, and the output is the dependent variable (the input causes the output, not the other way around).

Sometimes you are given a system model that describes the relationship between input and output, but often the task of an engineer is to build a suitable system model for a design activity. Note: modelling is always conducted in the context of a purpose, which guides how much model fidelity (or accuracy) is required.

### 1.3 Problem Landscape

Examples of robotics settings were models, data, and extraction of knowledge arise include:

- Determining a robot's absolute location from traditional position measurements, such as GPS, INS.
- Determining a robot's relative locations via passive sensors, such as Electro-optic (EO), lidar, and radar.
- Higher-order perceptions such as object recognition SLAM [2, Ch. 2].
- Information for decision making, similar sensor to above.
- Information for control, similar sensor to above.

Whilst there are variations in how words are used, some working definitions for this unit included:

- **Estimation** is the process of determining an estimate of a *static* quantity of interest. Related words include (essentially synonyms): "parameter estimation", "system identification", "adaption" and "learning".
- **Filtering** is the process of estimating a *dynamic* quantity from a sequence of data. Related words include (slight conceptual variations): "predicting" (estimating future quantity values based on current data) and "smoothing" (creating refined estimates of past quantity values using future data).

- **Decision** will mean using available information to make a choice. This can be in the context of a fixed information set or can be in the context of sequentially arriving information.
- **Control** will mean determining a sequence of actions. This is typically in the context of sequential determining actions on the basis of sequentially arriving information. Related words include "feedback", "closed-loop" and a long list of specialised versions.

Note that different fields may use different terminological landscapes. For example, related sequential knowledge extraction and decision-making ideas appear under a variety of names in the fields of statistics, operational research, and econometrics, amongst others. As an example of the type of potential confusion differences in terminological landscapes can cause, in statistics the term "prediction" means something like our "estimation", and instead the term "forecasting" means something like our "prediction".

# 2 Principles of system identification

Estimation of a static quantity from data is variously called "parameter estimation", "identification", and "system identification". In this section, we will present the estimation of a constant from data using terminology suitable for the topic of "adaptive control" (or "learning control") that we discuss later in this unit.

We begin introducing some system properties before discussing some criteria for estimation and a variety of algorithms. We end with a discussion of some problem assumptions and properties of algorithms.

## 2.1 Systems and representations

In this section, we will consider linear input-output models or order $n$ of the form:

$$y_{k+n} + a_{n-1}y_{k+n-1} + \ldots + a_0 y_k = b_{n-1}u_{k+n-1} + \ldots + b_0 u_k \tag{2.1}$$

where $k = 0, 1, \ldots$, $u_k \in R$, $y_k \in R$

### 2.1.1 Latent variable: state space form

$$x_{k+1} = Ax_k + Bu_k \tag{2.2}$$
$$y_k = Cx_k \tag{2.3}$$

where $x_k$ can be considered a latent (or hidden) variable.

### 2.1.2 Creating the latent form from a difference equation

We can define the state vector as

$$x_k = [y_k, y_{k+1}, \ldots, y_{k+n-1}]'. \tag{2.4}$$

Noting that $x^i_{k+1} = x^{i+1}_k$, for $i \leq n-1$, and that $x^n_{k+1} = y_{k+n}$ is described by (2.1) (i.e. make $y_{k+n}$ the subject of the equation), we realised a time-invariant linear discrete-time state space model can be written as

$$x_{k+1} = \begin{bmatrix} 0 & 1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & 1 \\ -a_0 & -a_1 & -a_2 & -a_3 & \ldots & -a_{n-1} \end{bmatrix} x_k + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} (b_{n-1}u_{k+n-1} + \ldots + b_0 u_k) \tag{2.6}$$

$$y_k = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 & 0 \end{bmatrix} x_k \tag{2.7}$$

We often consider the case with $b_0 = 1$ and all other $b_i = 0$.

The proposed state space representations are not unique representations of the input-output behaviour described by provided difference equations or transfer functions. That is, other $A$, $B$ and $C$ combinations will also be valid representations of the provided input-output behaviour.

### 2.1.3 System properties

**Definition 1** *A state space system is (complete)* **state controllable** *if there exists a controller that is able to bring any state from any initial value to a desired value within a finite time.*

A time-invariant state space system is (complete) state controllable if

$$rank\left\{\left[B, AB, \ldots, A^{n-1}B\right]\right\} = N.$$

Note that the state controllability matrix is

$$\Gamma^c_{cont} = \left[B, AB, \ldots, A^{n-1}B\right]$$

(Sometimes called the controllability Grammian).

**Definition 2** *A state space system is (complete)* **output controllable** *if there exists a controller that is able to bring the output to any desired value from any initial value.*

A time-invariant linear state space system is (complete) output controllable if

$$rank\left\{\left[D, CB, CAB, \ldots, CA^{n-1}B\right]\right\} = N.$$

**Definition 3** *A system is* **state observable** *if any initial state can be found from observing the output vector over a finite time period*

A time-invariant state space system is state observable if the matrix

$$\Gamma_{obs}^{c} = \begin{bmatrix} C \\ CA \\ \cdot \\ \cdot \\ CA^{n-1} \end{bmatrix}$$

has rank $N$.

## 2.2 Principles of identification

An identification problem has four elements:

1. The **object** to be identified, typically a model.
2. The measured **data** from which the object is to be identified. Typically available in real-time.
3. The **parameterised class** which is assumed to contain the object
4. A **criterion** that is able to specify the goodness of fit of elements from the parameterised class to the data.

In adaptive control, the identification task is performed in real-time as the system operates and the data is received. In a different engineering context, you might be interested in the offline identification from a collected batch of data, but that is not the situation in the adaptive control problem.

## 2.3 The data and parameterised model class

To place our identification problem into a standard form, we note that in linear systems (of order $n$) the data and model class can be rewritten to the form

$$z_k = \theta' \phi_k$$

where $z_k$ is our (scalar case) measured output data at time $k$, $\phi \in R^{2n}$ is our regressor at time $k$ (quantities impacting the output) and $\theta \in R^{2n}$ is our (unknown) parameters. For a given estimate $\widehat{\theta}$, the expression $e(\widehat{\theta}, k) = z_k - \widehat{\theta}' \phi_k$ is called the estimation error. Vector versions can be considered with suitable modification

If we had $N$ time steps of data, we might consider the identification criteria

$$J_N(\widehat{\theta}) = \sum_{k=1}^{N} e(\widehat{\theta}, k)^2.$$

### 2.3.1 An example of data and parameterised model class

An example of an object of interest ("Plant") and associated measured data ($u(k)$ and $y(k)$) are shown in Figure 1. In this example, the input and output of the underlining system are being shown as corrupted by additive noises ($v(k)$ and $w(k)$) which would make the identification task harder.

The parameterised class might be:

$$y^p_{k+n} + a_{n-1}y^p_{k+n-1} + \ldots + a_0 y^p_k = b_{n-1}u^p_{k+n-1} + \ldots + b_0 u^p_k,$$

where we might group the parameters together in the form $\theta = [a_{n-1}, \ldots, a_0, b_{n-1}, \ldots, b_0]'$ and write the regressor as

$$\phi_{k+n} = [-y_{k+n-1}, \ldots, -y_k, u_{k+n-1}, \ldots, u_k]'.$$

That is, we re-arrange the parameterised class as

$$y^p_{k+n} = -a_{n-1}y^p_{k+n-1} - \ldots - a_0 y^p_k + b_{n-1}u^p_{k+n-1} + \ldots + b_0 u^p_k,$$

so were can write $y^p_{k+n} = \theta\phi_{k+n}$ or $z_k = \theta\phi_k$ (after a re-indexing of $k$ so we can write $z_k = y^p_{k+n}$).

Note that the measured input $u_k$ is used instead of the unmeasured actual plant input $u^p_k$ in the parameterised class model. Also, note that previous measurements of $y_k$ appear in the regressor (and that the previously measured outputs $y_k$ are being used instead of the unmeasured previous plant outputs $y^p_k$).

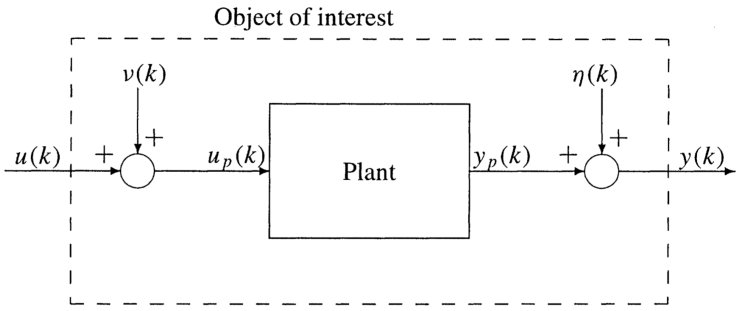


Figure 1: The object of interest and data. Image credit: [15].

## 2.4 Identification criteria and algorithms

### 2.4.1 Least squares identification

If we had $N$ time steps of data, we can define the (classic) least squares identification criteria of:

$$J_N(\theta) \stackrel{\text{def}}{=\!=} \sum_{k=1}^{N} e(\theta, k)^2$$

and define the associated least squares estimate

$$\widehat{\theta}_N \stackrel{\text{def}}{=\!=} \text{argmin} J_N(\theta)$$

consider the least squares estimate

$$\widehat{\theta}_N = R_N^{-1} Z_N$$

where we have

$$R_N \stackrel{\text{def}}{=\!=} \sum_{k=1}^{N} \phi_k \phi_k' \text{ and } Z_N^{-1} \stackrel{\text{def}}{=\!=} \sum_{k=1}^{N} \phi_k z_k'.$$

Here we are assuming the inverse $R_N^{-1}$ exists (more discussion later).

### 2.4.2 Recursive least squares

It can be shown that the Least squares can be rewritten in the following form [15, Ch. 3]

$$\widehat{\theta}_{k+1} = \widehat{\theta}_k + \frac{R_k^{-1}\phi_{k+1}}{1 + \phi_{k+1}' R_k^{-1} \phi_{k+1}} e(\theta_k, k+1)$$

where, via the matrix inversion lemma, we can write

$$R_{k+1}^{-1} = R_k^{-1} - \frac{R_k^{-1}\phi_{k+1}\phi_{k+1}' R_k^{-1}}{1 + \phi_{k+1}' R_k^{-1} \phi_{k+1}}.$$

Note that in practice this form is rarely used in an adaptive control setting.

### 2.4.3 Basic projection algorithm

$$\widehat{\theta}_{k+1} = \widehat{\theta}_k + \frac{1}{\phi_{k+1}'\phi_{k+1}} \phi_{k+1} e(\theta_k, k+1)$$

### 2.4.4 Normalised Least Mean Square (NLMS)

$$\widehat{\theta}_{k+1} = \widehat{\theta}_k + \frac{\mu}{\alpha + \phi_{k+1}'\phi_{k+1}} \phi_{k+1} e(\theta_k, k+1)$$

where $\alpha > 0$ to avoid division by 0 and $\mu$ is the stepsize ($0 < \mu < 2$ so update is well behaved).

### 2.4.5 Least Mean Square (LMS)

$$\widehat{\theta}_{k+1} = \widehat{\theta}_k + \mu \phi_{k+1} e(\theta_k, k+1)$$

where $\mu > 0$ is a small stepsize.

### 2.5 Data, model and algorithm assumptions and properties

**Stable algorithm** A standing assumption is that the identification algorithm being used is stable in the sense that initial conditions or values in the algorithm are forgotten and any internal errors that arise are forgotten.

**Data in model class** We assume the data is generated by a model in the parameterised class

Of course, the model that generated the data being a member of the model class would seem to be a challenging requirement in a real-world setting (given that a model can only be an approximation of the real world). In practice, we are hoping that that model class is rich enough that includes a model close enough that everything works. Saying something with technical precision is beyond the scope of these notes.

**Informative data** We assume data is informative in the sense that the received data sequence will not result in singular matrices or divide-by-zero operations within the identification algorithm. This is often expressed as a **persistency of excitation** condition [15, p. 69]

$$\frac{1}{K}\sum_{k=\ell}^{\ell+K-1}\frac{\phi_k'\phi_k}{1+\phi_k'\phi_k} \geq aI > 0$$

for some $a > 0$, $K \geq 2n$ and all $\leq > 0$.

**Consisteny algorithm** An algorithm is said to be consistent if is convergence to the least squares estimate (this usually means convergence to the true value). According to [15, Ch. 3] RLS and NLMS are consistent.

**Tracking variations algorithm** If used in a time-varying environment, we may be interested in the ability to track changing parameters. RLS is not able to track parameter variations. According to [15, Ch. 3] the NLMS algorithm can track variations.

## 3 Kalman Filtering

Kalman filtering is a very useful and important signal-processing technique that was developed in the 1960s and has impacted almost all engineering fields. Here, to assist with the assignment, we will sketch some of the important features of the discrete-time Kalman filter (KF).

We begin by introducing the dynamics model under consideration before defining a typical stochastic filtering problem. We will then present a KF solution. In later sections, we look at non-linear extensions.

### Remarks

1. A stochastic filtering problem, is a special type of estimation problem for stochastic processes.
2. Here, the term "filtering" is slightly different from the term "filtering" you will have previously encountered (for example, the usage here is different from a low-pass filter which does something to the frequency spectrum of a signal). We use "filter" in the more general sense of an operation that separates something useful (the true signal) from something undesired (the noise).
3. Filtering is typically a recursive process; recursive in the sense that a new piece of information is produced whenever a new measurement arrives. This is the same sense that an IIR (infinite impulse response) filter is recursive.

## 3.1 The State Process

In linear stochastic filtering problems, we always begin by considering the linear stochastic (random) process:

$$x_{k+1} = A_k x_k + v_k \tag{3.1}$$

where $x_k \in R^N$ is a vector, $A_k \in R^{N\times N}$ is a square matrix and $v_k \in R^N$ is a white zero-mean Gaussian noise with known covariance (and uncorrelated to $x_k$). Here the notation $R^{N\times N}$ means the set of matrices with real elements and dimension $N \times N$, and $\in$ means "is an element". Hence "$A_k \in R^{N\times N}$" is read "$A_k$ is a $N \times N$ matrix with real valued elements". This notation has a similar meaning when used for vectors $x_k \in R^N$, etc.

Gaussian noise is a sequence of random variables, where each random variable has a "normal" (or bell-curve) distribution at each time instant (and is independent of the value at other time instants). This sequence of random variables is called a stochastic process. Because of the appearance of this type of noise in the process (3.1), the state $x_k$ also inherits this stochastic nature and is also a stochastic process.

In Kalman filtering, we talk of covariances and will assume that

$$E[v_k v_\ell'] = \begin{cases} Q_k & \text{if } k = \ell \\ 0 & \text{otherwise} \end{cases}$$

where $Q_k \in R^{N\times N}$ and is called our process covariance matrix. Here $E[.]$ is called the expectation operation and is defined as

$$E[f(x)] = \int_{x\in R^N} f(x)P(x)dx$$

for any function $f(x)$, where $P(x)$ is the probability of $x$. Expectations are used to define things like "means" and "averages". For example, $E[x]$ is the mean of $x$. Alternatively, we can think of $Q_k$ as the matrix version of a scalar noise variance, which is a the concept that you have probably encountered previously.

Don't be overly concerned about the technical phrases and definitions used above; you might want to think of $Q_k$ as the energy of the $v_k$. In practice, the model (3.1) can be used to represent most of the dynamics you have encountered during your studies. To illustrate we now introduce an example state process.

**Ex 1: A simple vector state process.**

Often in navigation problems, we might be interested in a state $x_k$ that represents the vehicle's dynamic quantities in a vector (for example, position and velocity coordinates). In this case, we might let the first two elements of $x_k$ represent the Cartesian position in $(x^c, y^c)$ co-ordinates, and the last two elements the velocity vector $(\dot{x}^c, \dot{y}^c)$. To avoid notation confusion, we use the superscript $c$ to remind us that these are cartesian coordinates, not our state/measurement vectors. In this case, $N = 4$ is the dimension of our state process.

To complete our description, we assume that the state process is "noisy" and we might imagine that the noise process $v_k$ represents something like the physical aerodynamic buffering the airframe experience in a fight. The noise might also represent other modelling inaccuracies/errors.

Hence, in a navigation problem, we might choose the notation:

$$x_k = \begin{bmatrix} x_k^c \\ y_k^c \\ \dot{x}_k^c \\ \dot{y}_k^c \end{bmatrix} \qquad A_k = \begin{bmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad Q_k = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 \\ 0 & \sigma_y^2 & 0 & 0 \\ 0 & 0 & \sigma_{\dot{x}}^2 & 0 \\ 0 & 0 & 0 & \sigma_{\dot{y}}^2 \end{bmatrix} \tag{3.2}$$

where $\Delta T$ is the sample period used, $\sigma_x^2$, etc. are the effective "variances" of the process "bumping" components of the state (we have assumed a simplified noise because this matrix is diagonal). We sometimes think of $\sigma_x^2$ as the the energy of the bumping that is driving the state process.

We expand on this example later.

**Remarks**

4. We note that any linear stochastic (random) process can be described by (3.1).
5. We could easily add $z$ components to our state process, which might increase $N = 6$. If required, we could also add acceleration states (but this would further increase the dimension).
6. We can also consider simpler processes, such as scalar processes.

## 3.2 The Measurement Process

The second part of any filtering problem is the measurement process (how measurements are related to the state). An example of a measurement process is that the output voltages from a gyroscope reflect

the measured angular rate. In linear problems, we use the following measurement equation:

$$y_k = H_k x_k + w_k \tag{3.3}$$

where $y_k \in R^P$ is our measurement at time $k$ (that is, there at $P$ measurements), $H_k \in R^{P\times N}$, and $w_k \in R^P$ is a white zero-mean Gaussian noise with known covariance (uncorrelated to $x_k$). We assume that

$$E[w_k w_\ell'] = \begin{cases} R_k & \text{if } k = \ell \\ 0 & \text{otherwise} \end{cases}$$

where $R_k \in R^{P\times P}$ and is called our measurement covariance matrix.

**Ex 2: Simple Measurement Process**

For example, we might consider $y_k = [x_k^{gps}, y_k^{gps}]'$ to be GPS measurements of $x^c$ and $y^c$ coordinates (where we have assumed that the GPS errors $v_k$ are Gaussian - which is not actually true). Then

$$H_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

## Remarks

7. Note, the number of measurements $P$ need not equal the dimension of the state process $N$. Generally, we will have fewer measurements than states. Note that this situation is different from previously encountered "low-pass" filtering approaches that have assumed one measurement per signal.

## 3.3 The State-Space Form

When (3.1) and (3.3) are written together

$$\begin{aligned} x_{k+1} &= A_k x_k + v_k \\ y_k &= H_k x_k + w_k \end{aligned} \tag{3.4}$$

we have a combined description of the problem.

This way of expressing the state and measurement information is called the **State-space form** of the state and measurement processes.

**Ex 3: Combined state and measurement process**

To check that everything makes sense, consider what happens when we combine the state and measurement processes from previous examples (and expand into scalar equations). This model might be useful in the real-world problem of estimating (tracking) the location of an aircraft given GPS (position information) over time.

From Ex 1, we have the 2D dynamics

$$\begin{aligned} x_{k+1} &= A_k x_k + v_k \\ \begin{bmatrix} x_k^c \\ y_k^c \\ \dot{x}_k^c \\ \dot{y}_k^c \end{bmatrix} &= \begin{bmatrix} 1 & 0 & \Delta T & 0 \\ 0 & 1 & 0 & \Delta T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1}^c \\ y_{k-1}^c \\ \dot{x}_{k-1}^c \\ \dot{y}_{k-1}^c \end{bmatrix} + v_k \end{aligned}$$

Hence,

$$\begin{bmatrix} x_k^c \\ y_k^c \end{bmatrix} = \begin{bmatrix} x_{k-1}^c + \Delta T \dot{x}_{k-1}^c \\ y_{k-1}^c + \Delta T \dot{y}_{k-1}^c \end{bmatrix} + \begin{bmatrix} v_k^1 \\ v_k^2 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \dot{x}_k^c \\ \dot{y}_k^c \end{bmatrix} = \begin{bmatrix} \dot{x}_{k-1}^c \\ \dot{y}_{k-1}^c \end{bmatrix} + \begin{bmatrix} v_k^3 \\ v_k^4 \end{bmatrix}.$$

This is actually the 1st order Euler approximation of the ODE describing constant velocity 2D dynamics (plus Gaussian noise). Therefore, Ex 1 was actually the sampled version of a continuous-time description of linear vehicle motion in 2D space (down and cross-range). Further, if we look at the measurement equation given in Ex 2:

$$\begin{aligned} y_k &= H_k x_k + w_k \\ \begin{bmatrix} x_k^{gps} \\ y_k^{gps} \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} x_k^c \\ y_k^c \\ \dot{x}_k^c \\ \dot{y}_k^c \end{bmatrix} + w_k \\ &= \begin{bmatrix} x_k^c \\ y_k^c \end{bmatrix} + w_k \end{aligned}$$

That is, the measurements are the $(x^c, y^c)$ location plus some measurement noise.

Warning: This example is too simple to be applied to real GPS/INS integration problems. This example is included for the purpose of introducing the Kalman filtering technique.

## Remarks

8. This (hopefully) simple example illustrates many of the important features of stochastic filtering problems. At this point, key insights are that:
   - It is often desirable to estimate all components of the $x_k$ vector (position plus velocity),
   - Often, we only measure part of the state directly. Sometimes this is called an indirect measurement of the state.

- Intuitively, we might feel that velocity information can be "estimated" from successive values of position information. The Kalman filter is the "optimal" approach to this sort of strategy.
- Each physical measurement appears only once on the left-hand side of the measurement equation (but can be related to any sub-set of state variables). Conversely, a state variable can appear at multiple locations on the right-hand side of the measurement equation.
- Anything with memory appears as a state. For example, unknown measurement biases, delayed state values influencing measurements, and similar, all appear in the state vector (even though we might think of them as "measurement" related).

## An Alternative Example: General ARMA model to State-space

It was mentioned previously that sometimes processes are intrinsically discrete and have nothing to do with a continuous evolution of time. Such processes are often described in terms of an ARMA model, which was touched on briefly in Section 3.9 (9, 10, 11). We will now continue that discussion and show how the ARMA model can be converted to state-space form. We begin by repeating the ARMA model equation given in Section 3.9:

$$y(k+n) + \alpha_{n-1}y(k+n-1) + \alpha_{n-2}y(k+n-2) + \cdots + \alpha_0 y(k)$$
$$= \beta_m w(k+m) + \beta_{m-1}w(k+m-1) + \cdots + \beta_0 w(k),$$
$$m = n-1 \quad \text{and} \quad k = 0, 1, 2, \ldots \tag{5.3.35}$$

Note that the model is single-sided in $k$ and the order of the MA part is at least one less than the AR part. The reasons for these restrictions will be apparent shortly.

Conversion of the ARMA model to the controllable canonical form is effected by following a procedure analogous to that used in the continuous differential equation case. First, we define an intermediate variable $r(k)$, which is the solution of

$$r(k+n) + \alpha_{n-1}r(k+n-1)$$
$$+ \alpha_{n-2}r(k+n-2) + \cdots + \alpha_0 r(k) = w(k) \tag{5.3.36}$$

Next, we define state variables that are analogous to the phase variables in the continuous case:

$$\begin{aligned} x_1(k) &= r(k) \\ x_2(k) &= r(k+1) \\ &\vdots \\ x_n(k) &= r(k+n-1) \end{aligned} \tag{5.3.37}$$

Using state variables as defined by Eq. (5.3.37) and the difference equation, Eq. (5.3.36), leads to the matrix equation

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ \vdots \\ x_n(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 1 & 0 & \cdots \\ \vdots & & & & \\ -\alpha_0 & -\alpha_1 & -\alpha_2 & \cdots & -\alpha_{n-1} \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_n(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} w(k) \tag{5.3.38}$$

We now need to reconstruct the original $y(k)$ variable as a linear combination of the elements of the state vector. This can be done by considering a superposition of equations of the type given by Eq. (5.3.36) with the index $k$ advanced appropriately. However, it is a bit easier to do this with a block diagram by using $z$ transforms as shown in Fig. 5.6. The analogy between the block diagrams of Figs. 5.2 and 5.6 should be apparent. It should be clear now that the output equation that takes us from the state space back to $y(k)$ is

$$y(k) = [\beta_0 \quad \beta_1 \quad \beta_2 \quad \cdots \quad \beta_n] \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_n(k) \end{bmatrix} \tag{5.3.39}$$

### 3.4 The KF problem

The KF estimation problem can be stated as:

*Given state process (3.1), and measurement process (3.3), and initial state value $\bar{x}_0$, determine the "optimal" estimate of the state $x_k$ at time $k$, given the measurements $y_0, y_1, \ldots, y_k$. Denote this estimate $\widehat{x}_k$. Moreover, if possible, provide a recursive solution, so that the next estimate $\widehat{x}_{k+1}$ (of the state $x_{k+1}$) can be determine from the previous estimate $\widehat{x}_k$ (of the state $x_k$) and the new measurement $y_{k+1}$.*

### Remarks

9. We will not address the meaning of the word "optimal" in these notes; However, the KF estimate of the state $x_k$ at time $k$ (i.e. $\widehat{x}_k$) is the estimator that makes the "optimal" use of all measurement information available until this point in time (i.e. $y_0, y_1, \ldots, y_k$).

### 3.5 Kalman Filter

Without proof, a Kalman filter is the recursive equations:

The KF is a two-step process. The time update is:

$$\begin{aligned} \widehat{x}_k^- &= A_{k-1}\widehat{x}_{k-1} \\ P_k^- &= A_{k-1}P_{k-1}A_{k-1}' + Q_{k-1} \end{aligned} \tag{3.5}$$

where $\widehat{x}_k$ is the estimate at time $k$ (actually $E[x_k|y_0,\ldots,y_k]$), $\widehat{x}_k^-$ is called the one-step-ahead prediction at time $k-1$ (based on measurements until time $k-1$, and $P_k$ (without the superscript) is called the error covariance matrix.
The measurement update is:

$$\begin{aligned} \widehat{x}_k &= \widehat{x}_k^- + K_k(y_k - H_k\widehat{x}_k^-) \\ K_k &= P_k^- H_k'(H_k P_k^- H_k' + R_k)^{-1} \\ P_k &= (I - K_k H_k)P_k^- \end{aligned} \tag{3.6}$$

Here, $K_k$ is known as the Kalman gain (or blend matrix) and $P_k^-$ is the state prediction covariance (a type of estimation error)
This filter is initialized at some best initial estimate $\widehat{x}_0 = \bar{x}_0$, and an estimate of the initial error $P_0 \approx A_0 E[(\bar{x}_0 - x_0)(\bar{x}_0 - x_0)']A_0 + Q_0$. Often, $P_0$ is unknown and we use a large $P_0$.
WARNING: There are many forms of the Kalman filter recursion.

### Remarks

10. The above KF equations (3.5) and (3.6) represent a recursion. Consider time $k$.
    (a) Assume that we have values for $\widehat{x}_{k-1}$ and $P_k$ from the last step, and a new measurement $y_k$.
    (b) At this step, we first calculate $K_k$ and $P_{k+1}$ and then calculate $\widehat{x}_k$.
    (c) We are then prepared for the next step or next measurement, so set $k = k+1$ and return to step (a).
11. The term $K_k$ is called the Kalman gain, or blend matrix, and describes how much new measurement information affects the old estimate. The intuition is that very noisy measurements will have "small" $K_k$. The KF recursions find the "optimal" choice of $K_k$.
12. The virtue of the KF is not so much that it is "optimal", but that it has the right sorts of "tuning" knobs. Engineers often fiddle with $Q_k$ and $R_k$ until the KF produces the right trade-off between stability and performance. Moreover, the KF has good numeric stability properties.

13. The second line of (3.6) involves an operation known as matrix inversion. Often, these equations are written in an alternative form that avoids the need to calculate a matrix inverse at every time step.

14. The filter will "forget" initial choices of $P_0$ and $\bar{x}_o$. Hence, if $P_0$ is not known, you can always try a large value. After the initial iterations of the algorithm, the filter will quickly become independent of your choice for $P_0$.

15. For $k = 0, 1, \ldots$, the covariance $P_k^-$ must be a positive definite symmetric matrix. There will be numeric problems if it loses these properties, These properties can be an issue if $|R_k|_\infty$ is small ($|.|_\infty$ is the largest magnitude of elements of the matrix.)

16. When the filter is producing good estimates, $|P_k^-|$ will be small.

### 3.6 What does the KF mean?

The KF is almost always understood as the two step ("time" and "measurement") update process shown in Figure 2.

The basic idea is that the time update step creates the best state prediction, based on the dynamic model and the estimated state at the previous time instant. We expect that this prediction will be fairly close to the true value. When a measurement arrives it is used to correct (or improve, or refine) our original prediction. This corrected prediction should be even closer to the true value.

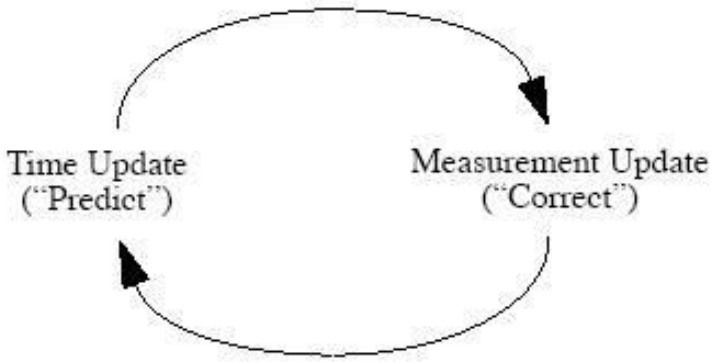


Figure 2: Kalman filter as a two-step process. A time step ahead, followed by a correction by the measurement. Image Credit: Welch and Bishop [5].

To match this representation, we can understand the KF as a two-step process. The time update is:

$$\begin{aligned} \widehat{x}_k^- &= A_{k-1}\widehat{x}_{k-1} \\ P_k^- &= A_{k-1}P_{k-1}A_{k-1}' + Q_{k-1} \end{aligned} \tag{3.7}$$

where $\widehat{x}_k^-$ is called the one-step-ahead prediction at time $k-1$ (based on measurements until time $k-1$, and $P_k$ (without the superscript) is called the error covariance matrix.

The measurement update is:

$$\begin{aligned} \widehat{x}_k &= \widehat{x}_k^- + K_k(y_k - H_k\widehat{x}_k^-) \\ K_k &= P_k^- H_k'(H_k P_k^- H_k' + R_k)^{-1} \\ P_k &= (I - K_k H_k)P_k^- \end{aligned} \tag{3.8}$$

Figure 3 provides a graphical representation of the two steps.

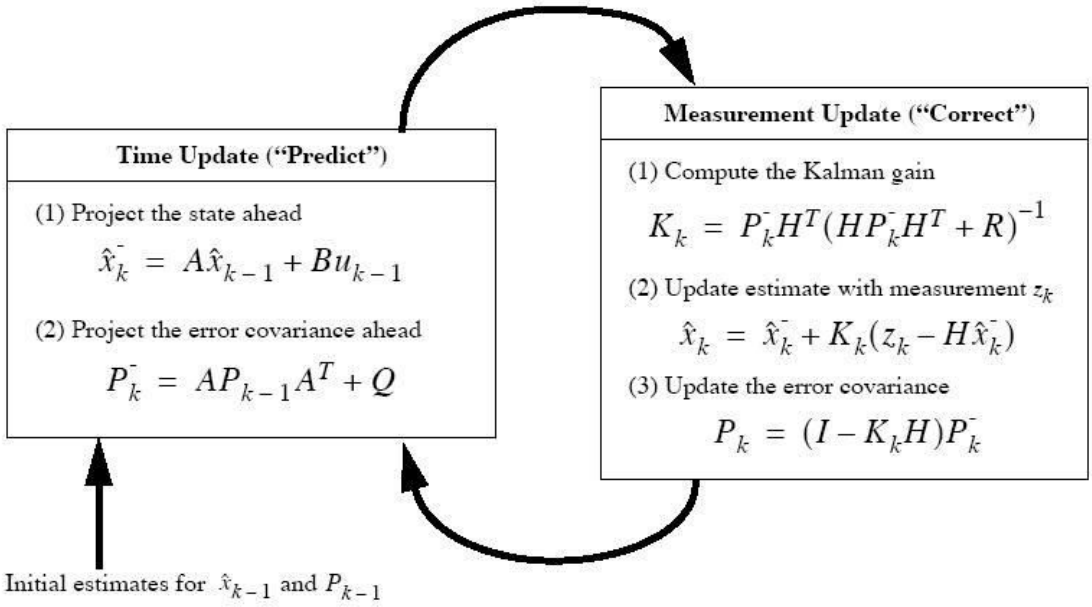


Figure 3: The Kalman filter "time" and "measurement" updates. The $z_k$ shown in the figure is our measurement $y_k$. Different authors use slightly different notations. Image Credit: Welch and Bishop [5].

What is this doing to our understanding of the state? A visual interpretation of a scalar case is shown in Figure 4 where the means $\widehat{x}_k$ (and $\widehat{x}_k^-$) and variances $P_k$ (and $P_k^-$) have been unwrapped as gaussian pdf curves. The change in $\widehat{x}_k^-$ (and $P_k^-$) from sub-figure (a) to $\widehat{x}_k$ (and $P_k$) in (c) is the measurement update, whilst the change in $\widehat{x}_k$ (and $P_k$) from sub-figure (c) to $\widehat{x}_{k+1}^-$ (and $P_{k+1}^-$) (d) is the time update. Then the change from sub-figure (e) to (f) is the next measurement update, and so on.

## Remarks

17. The previous made Remarks 10 to 16 also hold from this version of the Kalman filter.

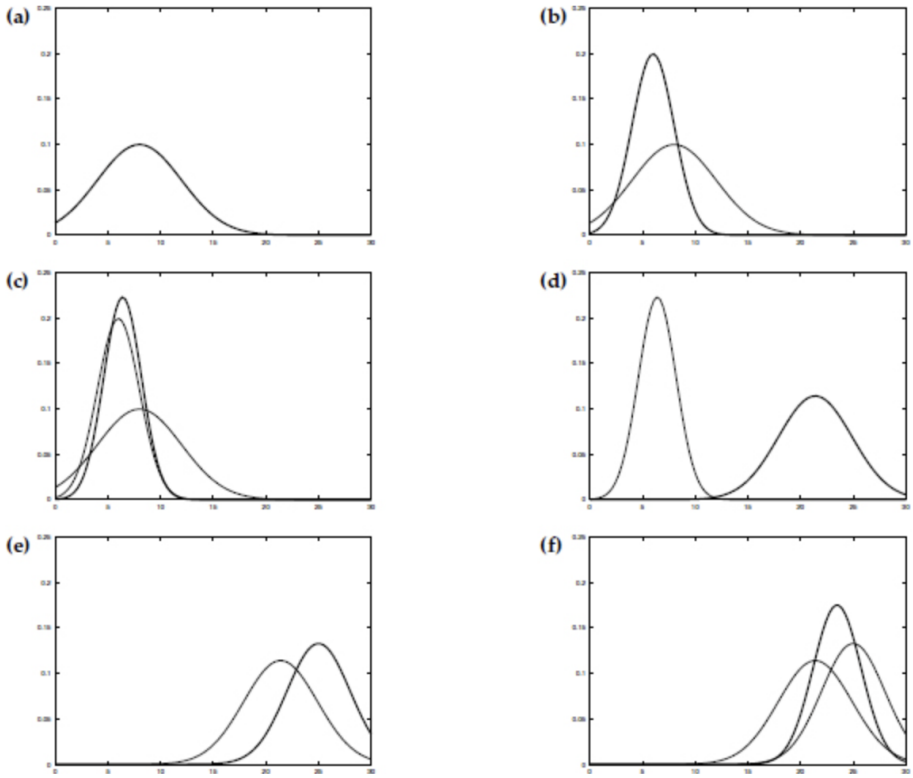


**Figure 3.2** Illustration of Kalman filters: (a) initial belief, (b) a measurement (in bold) with the associated uncertainty, (c) belief after integrating the measurement into the belief using the Kalman filter algorithm, (d) belief after motion to the right (which introduces uncertainty), (e) a new measurement with associated uncertainty, and (f) the resulting belief.

Figure 4: A scalar Kalman filter exampke of the "time" and "measurement" updates (unwrapped into pdf curves). Image Credit: S. Thrun, W. Burgard, and D. Fox [1].

18. This alternative representation of the KF better highlights the important role of system dynamics in estimation problems.

19. This step two interpretation is certainly worth deep consideration. Versions of these time and measurement updates appear in almost all filtering and estimation problems involving dynamics (including the most advanced, non-linear, fancy estimation approaches being developed today).

## 3.7 Compact Representation

This compact version of the filter cause confusion last year.

$$\begin{aligned} \widehat{x}_k &= A_{k-1}\widehat{x}_{k-1} + K_k(y_k - H_k A_{k-1}\widehat{x}_{k-1}) \\ K_k &= P_k^- H_k'(H_k P_k^- H_k' + R_k)^{-1} \\ P_{k+1}^- &= A_k(P_k^- - K_k H_k P_k^-)A_k' + Q_k \end{aligned} \tag{3.9}$$

whereas before $\widehat{x}_k$ is the estimate at time $k$ (actually $E[x_k|y_0,\ldots,y_k]$), $P_k^-$ is the state prediction covariance (a type of estimation error). This filter is initialized at some best initial estimate $\widehat{x}_0 = \bar{x}_0$, and an estimate of the initial error $P_0 \approx A_0 E[(\bar{x}_0 - x_0)(\bar{x}_0 - x_0)']A_0 + Q_0$. Often, $P_0$ is unknown ans we use a large $P_0$.

## 3.8 The Computer Processing Cycle

Typically, a Kalman filter would execute processing as shown in Figure 5. In high data rate applications, it is important to note that the processing must be completed before the next measurement occurs. Also, note that the estimation latency is related to computational difficulty.

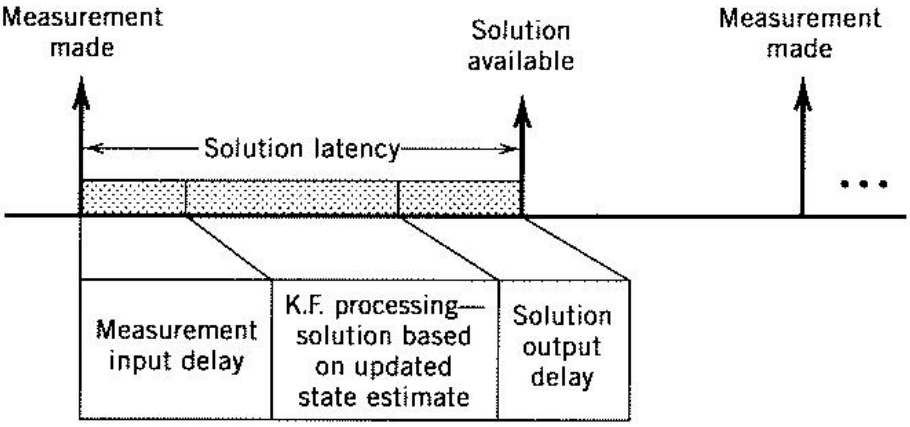


Figure 5: The basic Kalman filter processing cycle Image Credit: Welch and Bishop [5].

**First Kalman Filter Example**

Firstly note that in addition to the ideas in the previous Section 2, with certain model choices the Kalman filter problem and filter recursion simplify to a parameter estimation problem and recursive algorithm, respectively). Specifically, under certain model assumptions the Kalman fitler simplifies to recursive least squares of sub-section 2.4.2.

It will now be informative to examine the role of these KF quantities by considering a constant estimation problem (the simplest KF problem).

This example is taken from [5]. Imagine that we have noisy scalar measures of (scalar) unknown constant. We wish to estimate this constant from the measurements. In this problem, we have noticed that:

- The quantity of interest is a scalar. That is, $x_k \in R$, which means $N = 1$ and $Q \in R$.
- The quantity of interest is constant. That means $A = 1$ and $v_k = 0$. Hence $Q = 0$.
- We have a scalar measurement of the constant. That is, $y_k \in R$, which means $P = 1$.
- With loss of generality, we assume the measurement has unity gain. That is $H = 1$. We also assume that the measurements have a particular noise variance, $\sigma^2 = 0.01$.

Figure 6 shows the evolution of various quantities in our first simulation. In this figure, the $x$ axis represents the evolution of time, and the $y$ axis represents measurements/estimates. The solid constant line presents the true value of the unknown constant. The crosses in the figure represent the noisy measurements. The trajectory shown in the figure is the sequence of KF estimates, and we note that the KF quickly converges to the true value.

We can also plot some of the KFs internal values. For example, in Figure 7 we have plotted the value of the error covariance $P_k$ for this simulation. In this figure, we see that the KF believes that the error has decreased over time. We can visually confirm this is true by noting in Figure 6 that the estimation error seems to decrease. If we wanted, we could plot the actual estimation error and $P_k$ on the same graph to compare.

In realistic situations, it is not possible to know the true value of the noise covariance $R$ and often this is a quantity that the engineer must choose or guess. We will examine the performance of the KF for different choices of $R$, and hence illustrate the influence of $R$ (on KFs applied to measurement data with fixed $R = 0.01$). This situation is known as model mismatch (because the model parameters used in the KF do not match the parameters used to generate the data).

To examine the effect of model miss-match in $R$ on the performance of the filter, we consider a situation with larger $R$ (Figure 8) and a situation with smaller $R$ (Figure 9). Note that the same measurements are used, so the actual measurement noise has not changed (we have only changed the KF's assumptions about the noise).

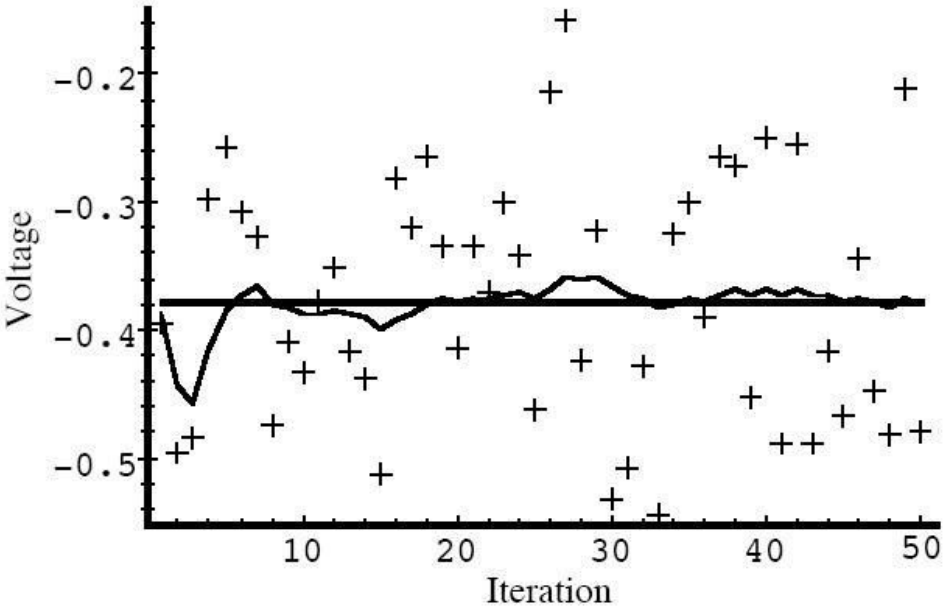


**Figure 3-1.** The first simulation: $R = (0.1)^2 = 0.01$. The true value of the random constant $x = -0.37727$ is given by the solid line, the noisy measurements by the cross marks, and the filter estimate by the remaining curve.

Figure 6: First simulation Image Credit: Welch and Bishop [5].

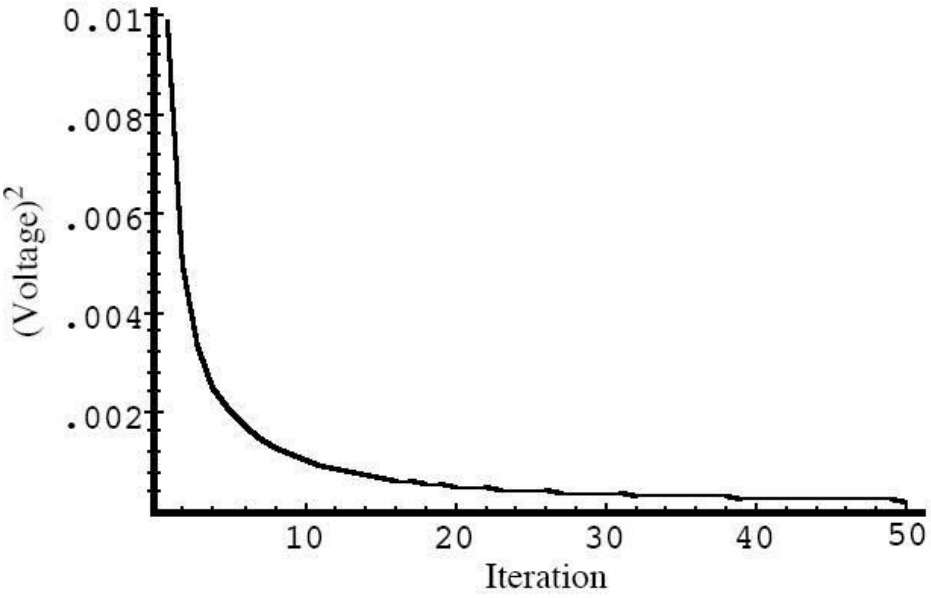


Figure 7: The evolution of $P_k$. Image Credit: Welch and Bishop [5].

Notice that with larger assumed $R$, the filter exhibits slower convergence to the true value, but eventually seems to get close to the constant. In comparison, when a smaller $R$ is assumed, the filter rapidly gets near the true value, but the filter never seems to completely converge to the constant.

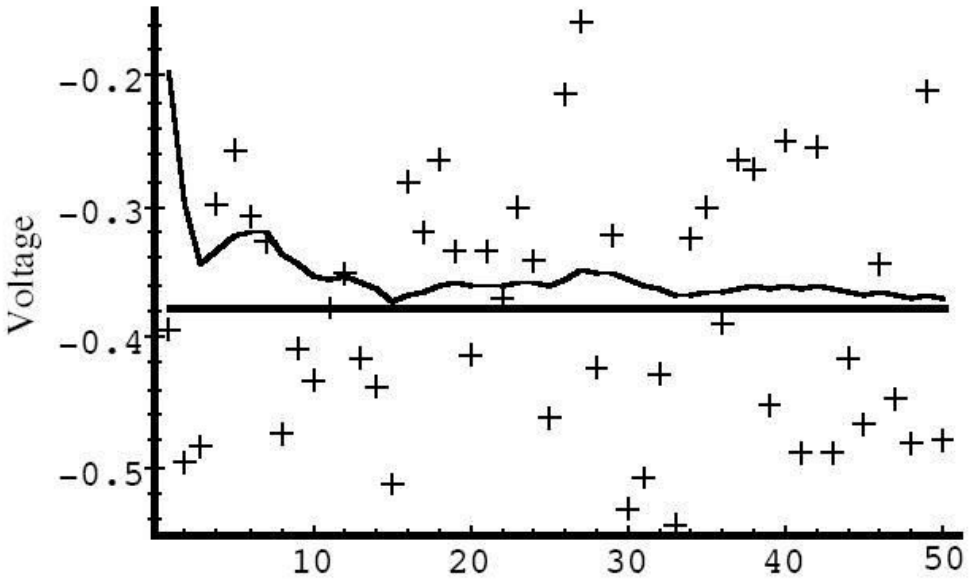


**Figure 3-3.** Second simulation: $R = 1$. The filter is slower to respond to the measurements, resulting in reduced estimate variance.

Figure 8: Second simulation, larger $R$. Image Credit: Welch and Bishop [5].

# 4 HMM Filter

A hidden Markov model (HMM) is an important signal model providing a useful model representation useful in a number of situations. Importantly it has a computationally efficient optimal filter.

Recall from EGH431, we introduced the concept of a Markov chain. Let us consider, at each time instant, the process $X_k \in S$, where $S$ is a finite set of discrete elements having $N$ elements. It will be convenient to define an abstract state space $S \stackrel{\text{def}}{=\!=} \{e_1, e_2, \dots e_N\}$, where $e_i$ are indicator vectors with 1 in the ith element and zeros elsewhere.

In addition to this set, a Markov chain is characterized by the probability of transition between states. We can introduce a **transition probability matrix** defined as follows:

$$A^{i,j} = P(X_k = e_i | X_{k-1} = e_j).$$

where $A^{i,j}$ is the $i, j$ th element of a matrix $A \in R^{N \times N}$. (Note some authors define the transposed version of this in the sense the roles of $i$ and $j$ in A are swapped, including within MATLAB's corresponding

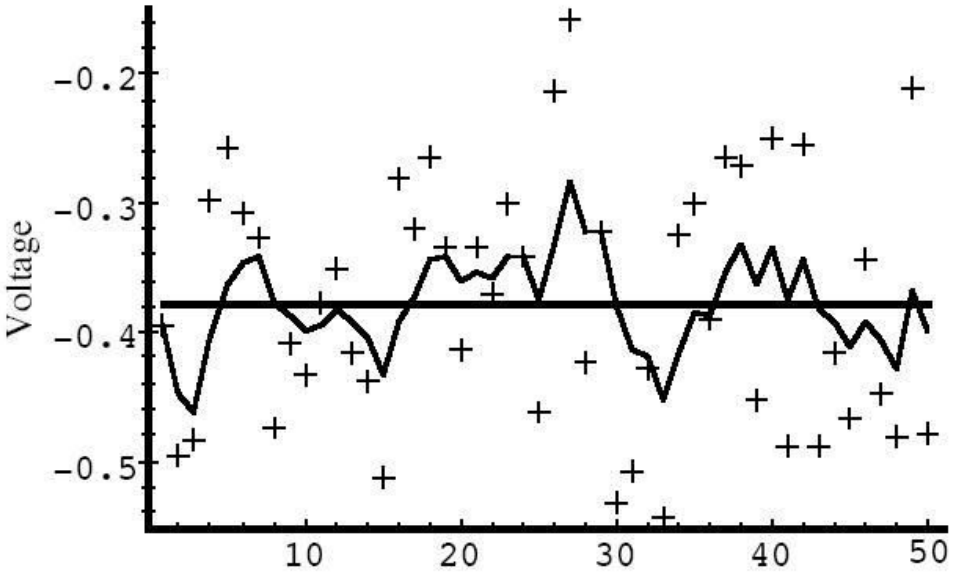


**Figure 3-4.** Third simulation: $R = 0.0001$ . The filter responds to measurements quickly, increasing the estimate variance.

Figure 9: Third simulation, smaller $R$. Image Credit: Welch and Bishop [5].

toolbox).

We assume for each $k > 0$, $X_k$ is observed through a stochastic process $y_k$ generated by conditional observation densities $b(y_k, i) \stackrel{\text{def}}{=\!=} P(y_k|Xk = e_i)$ for $1 \leq i \leq N$. At each time, we have a (diagonal) **measurement probability matrix** that encodes what $y_k$ is telling us about $X_k$

$$B(y_k) \stackrel{\text{def}}{=\!=} diag(b(y_k, 1), b(y_k, 2).\ldots, b(y_k, N)).$$

We denote our conditional posterior information at time $k$ as the vector $\widehat{X}_k \in R^N$ with elements $\widehat{X}_k^i = P(X_k = e_i|y_{[1,k]})$.

**Problem statement** The HMM filtering problem is stated as follows. Assume the HMM described above and given initial state distribution $\widehat{X}_0^i = P(X_0 = e_i)$, for $1 \leq i \leq N$. For each $k \geq 0$, Determine the conditional posterior information $\widehat{X}_k$ from a sequence of measurements $y_k$.

**General solution** For $k > 0$, $\widehat{X}_k$ can be calculated using the **HMM filter** using the following efficient matrix representation:

$$\widehat{X}_k = N_k B(y_k) A \widehat{X}_{k-1} \tag{4.10}$$

where $N_k \stackrel{\text{def}}{=\!=} \langle 1, B(y_k)A\widehat{X}_k\rangle^{-1}$ is a normalisation factor to ensure the posterior probabilities sum to 1, and $\widehat{X}_0$ is initial distribution.

**Further reading** Please see [10] for a tutorial introduction to HMM filters. This filter is also discussed in [1, Section 4.1.1] where it is called The Discrete Bayes Filter Algorithm.

## 4.1 Robotics case studies

HMM filters have been used in many applications including vision-based aircraft detection [7], statistical change detection [8], and more. Moreover, the HMM filter is an important companion for partially observed Markov decision problems that underpin many important decision and control problems arising in robotics and aligned fields.

## 4.2 MATLAB toolboxes

For the special case of measurements from a finite alphabet, MATLAB provides a function called *hmmdecode* that implements the HMM filter [11]. MATLAB also provides a function called *hmmviterbi* which solves a related but slightly different HMM filtering problem [12].

### 4.2.1 Example MATLAB code for HMM filter

```
function [xhat]=hmmfilter(datav,A,means,vars,x0)
% Author Jason J. Ford Apr 2023. For ENN586.
% HMM filter for continuous observations.
% Code assumes Gaussian measurement models, but B lines could be changed
% Assumes scalar measurement but could be modified.
%
% Inputs:
%    datav: 1xT sequence of scalar real measurements.
%    A:     NxN transition probability matrix,
%    means & var: N-vectors, elements are the means & variances of each measurement from each sta
%    x0:    Nx1 vector of initial distribution P(X_o)
%
% Outputs: the posterior information \hat{X}_k

% Extract some dimensions from passed data.
N=length(x0); % Number of Markov states, should also be the size of A.
[Nm,T]=size(datav); %Nm=1 but T is the length of measurements.

% first step of recursion set up manually
B=zeros(N,N);
for i=1:N
    B(i,i)=pdf('Normal',datav(1,1),means(i),sqrt(vars(i))); %need to hard code these,
end
xestu=B*A*x0;  % unnormalised estimate.
xhat(:,1)=inv(sum(xestu))*xestu; % normalise step

%main loop of recursion
for k=2:T
    for i=1:N
        B(i,i)=pdf('Normal',datav(1,k),means(i),sqrt(vars(i))); %need to hard code these,
    end
    xestu=B*A*xhat(:,k-1);  % unnormalised estimate.
    xhat(:,k)=inv(sum(xestu))*xestu; % normalise step
end
```

# 5 General Non-linear filtering

Generally, robotics applications involve departures from the linear Gaussian setting given in the previous section. There are 3 main types of departure:

- The state process can non-linear and/or time-varying. Examples include non-linear relationship between steering and linear motion quantities.
- The measurement process is non-linear. Examples include Radar, and Lidar measurements which are (relative) range and bearing. EO sensors are bearing.
- The process and/or measurement noises are non-gaussian. For example, noises on inertial sensors such as gyroscopes and accelerometers are more complex than Gaussian (temperature and motion dependent bias, statistical characteristics that vary with the sampling rate and average period).
- Filtering performance criteria might not be previous optimisation cost. For example, we might be interested in feature extraction, object recognition or other higher-order perceptions such as localisation or SLAM [2, Ch. 6]

## 5.1 General non-linear filter problem

Assume the following general state and measurement process:

$$\begin{aligned} x_{k+1} &= f(x_k, k) + v_k \\ y_k &= h(x_k, k) + w_k \end{aligned} \tag{5.11}$$

where $f(.,.)$ and $h(.,.)$ are nonlinear time-varying functions and $v_k$ and $w_k$ are *i.i.d.* random variables. This nonlinear system 5.11 constraint our investigation to systems with the following properties:

- The state process does not depend on the measurements.
- Causality: the current state does not dependent on future values of the state or future measurements.
- Markov: the current state depends only on the previous state.
- The measurements are independent in time except for the memory present in the state.

This class of systems covers many of the situations we will encounter.

**Problem statement** The nonlinear filtering problem is stated as follows. For the nonlinear system 5.11, defined for $k \geq 0$ where the initial state $x_0$ is a a random variable with known distribution and $v_k$ and $w_k$ are sequences of random variables with known state $x_0$ is a random variable with known distributions and correlations, from a sequence of measurements $y_k$ determine an estimate for $x_k$.

**General solution** We first will assume by estimate for $x_k$ we are happy to work with a probabilistic lens in the sense that we seek a probability density function description of our knowledge of $x_k$

For this purpose,it will be convenient to interpret 5.11 in a slightly different way. Let $p(x_k|x_{k-1}, k)$ denote the state transition probability encoded in the first equation of 5.11 and let $p(y_k|x_k, k)$ denote the conditional measurement probability encoded in the second equation of 5.11. It will be used to denote the sequence of measurements until time $k$ as $y_{[0,k]}$

It then follows from the application of Bayers rule that we can write down the solution to our estimation problems are the following recursion (also called Bayes filter in [1, Section 2.4]):

$$p(x_k|y_{[0,k-1]}) = \int p(x_k|x_{k-1}, k)p(x_{k-1}|y_{[0,k-1]})dx_{k-1} \tag{5.17}$$

$$p(x_k|y_{[0,k]}) = N_k p(y_k|x_k, k)p(x_k|y_{[0,k-1]}) \tag{5.18}$$

where $N_k$ is a normalisation factor to ensure $\int p(x_k|y_{[0,k]})dx_k = 1$.

These two operations are the non-linear analogy of the time update and the measurement update. The quantity $p(x_k|y_{[0,k-1]})$ in (5.17) is our conditional probability for $x_k$ based on all measurements up to time $k-1$ but not the latest measurement (labelled belief $\overline{bel}(x_k)$ in [1, Section 2.4]), whilst $p(x_k|y_{[0,k]})$ in (5.18) is our (posterior) conditional probability for $x_k$ based on all measurements including the measurement at time $k$ (labelled belief $bel(x_k)$ in [1, Section 2.4])

**What is hard about this?** The filter in the above form can only be implemented in the simplest of situations.

The challenge in most situations is the need to calculate the integral in (5.17) and the scaling in (5.18) for all values of $x_k$. When the state is a real value vector, this filter is termed infinite-dimensional and can only be perfectly calculated in the important but limited case where the posterior information contains enough structure to be represented by a finite number of statistical quantities: the Kalman filter

When the state takes value from a finite set, or can be approximated by a finite set, then the HMM filter might be applicable.

In general situations, most effort is devoted to developing approximations to this filter that offer a reasonable tradeoff between computational effort and performance.

## 5.2 Nonlinear filtering landscape

A brief landscape of non-linear filtering approximation can be found in [13] (for target tracking) and more general coverage can be found at [1, Ch 3].

Very roughly filtering for nonlinear dynamic system techniques of increasing complexity are Extended Kalman filter (EKF), Unscented Kalman filter (UKF), HMM filtering approximation and Particle filters.

In the following, we will briefly discuss the EKF and particle filter approaches. As they are relatively minor variations on other material, we will leave the Unscented Kalman filter (UKF) (see [1, Ch 3.4.2]) and HMM filtering approximation (see [1, Ch 4.1]) to your own reading.

## 5.3 The Extended Kalman Filter (EKF)

In most real-life problems, the state process and the measurement process are actually non-linear. The above KF recursions can be extended to handle non-linear problems using a technique called linearization. We do this now.

Assume the following general state and measurement process:

$$\begin{aligned} x_{k+1} &= f(x_k, k) + v_k \\ y_k &= h(x_k, k) + w_k \end{aligned} \tag{5.23}$$

with the same assumptions about $v_k$ and $w_k$ as in the Kalman filter problem, but here $f(.,.)$ and $h(.,.)$ are nonlinear time-varying functions. Let

$$\bar{A}_k(\bar{x}) = \frac{\partial}{\partial x} f(x,k)\bigg|_{x=\bar{x}} \quad \text{and } \bar{H}_k(\bar{x}) = \frac{\partial}{\partial x} h(x,k)\bigg|_{x=\bar{x}} \tag{5.24}$$

where the $|_{x=\bar{x}}$ is used to highlight that we first take the derivative w.r.t $x$, and then evaluated the answer at the point $x = \bar{x}$. In particular, we don't do things in reverse order.

Note that $\frac{\partial}{\partial x}$ denotes partial derivatives. In partial derivatives, we pretend that everything is constant except the quantity being considered. Moreover, these are matrix derivatives, where a vector derivative is defined as follows:

$$\frac{\partial}{\partial x}\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \frac{\partial a}{\partial x} \\ \frac{\partial b}{\partial x} \\ \frac{\partial c}{\partial x} \end{bmatrix},$$

and has an appropriate generalization to matrices. More information about partial derivatives at this link.

Using these partial derivatives (which have the matrix dimensions, $N \times N$ and $P \times N$ respectively, More about matrix calculus at this link) we can write down the EKF recursion:

The EKF can also be implemented as a two-step process. The time update is:

$$\begin{aligned} \widehat{x}_k^- &= f(\widehat{x}_{k-1}, k-1) \\ P_k^- &= \bar{A}_{k-1}(\widehat{x}_{k-1}) P_{k-1} \bar{A}_{k-1}(\widehat{x}_{k-1})' + Q_{k-1} \end{aligned} \tag{5.25}$$

where $\widehat{x}_k$ is the estimate at time $k$ (actually $E[x_k|y_0, \ldots, y_k]$), $\widehat{x}_k^-$ is called the one-step-ahead prediction at time $k-1$ (based on measurements until time $k-1$, and $P_k$ is called the pseudo error covariance matrix.
The measurement update is:

$$\begin{aligned} \widehat{x}_k &= \widehat{x}_k^- + K_k(y_k - h\left(\widehat{x}_k^-, k\right)) \\ K_k &= P_k^- \bar{H}_k(\widehat{x}_k^-)'(\bar{H}_k(\widehat{x}_k^-) P_k^- \bar{H}_k(\widehat{x}_k^-)' + R_k)^{-1} \\ P_k &= P_k^- - K_k \bar{H}_k(\widehat{x}_k^-) P_k^- \end{aligned} \tag{5.26}$$

Here,$P_k^-$ is the pseudo-state prediction covariance (a type of estimation error).
This filter is initialized at some best initial estimate $\widehat{x}_0 = \bar{x}_0$, and an estimate of the initial error $P_0 \approx A_0 E[(\bar{x}_0 - x_0)(\bar{x}_0 - x_0)']A_0 + Q_0$. Often, $P_0$ is unknown and we use a large $P_0$.

**WARNINGS**

- The EKF is an approximate filter and is sometimes unstable. These stability problems are most significant when the EKF is not initialized near the true state value.
- In the EKF, $P_k^-$ is not an error covariance quantity, but a "pseudo" quantity that has no precise meaning. However, big values are probably bad.

### 5.3.1 A worked EKF design example

**The state-measurement model** Consider the problem of estimating the position of an aircraft that is observed from a fixed radar ground station. Assume that the radar makes range and bearing measurements of the target, and assume that we are interested in tracking the 2D location of the aircraft.

To develop a state process model of the aircraft motion, we assume that the aircraft target is moving with a roughly fixed velocity. We choose a coordinate system with an origin fixed to the location of the radar ground station, with $z$-axis pointing down to the earth's centre of gravity, and with arbitrary $x, y$-axis directions that complete the right-hand rule.

The state process for this aircraft target can be described by (3.1), with vector and matrices as given in (3.2). Assume that $\Delta T = 1$ (sampled every second) and $\sigma_x^2 = \sigma_y^2 = 0$, $\sigma_{\dot{x}}^2 = \sigma_{\dot{y}}^2 = 0.01$ (aero-buffering of the velocity vector, only). We fix the aircraft attitude to 1000 $m$ above the ground station.

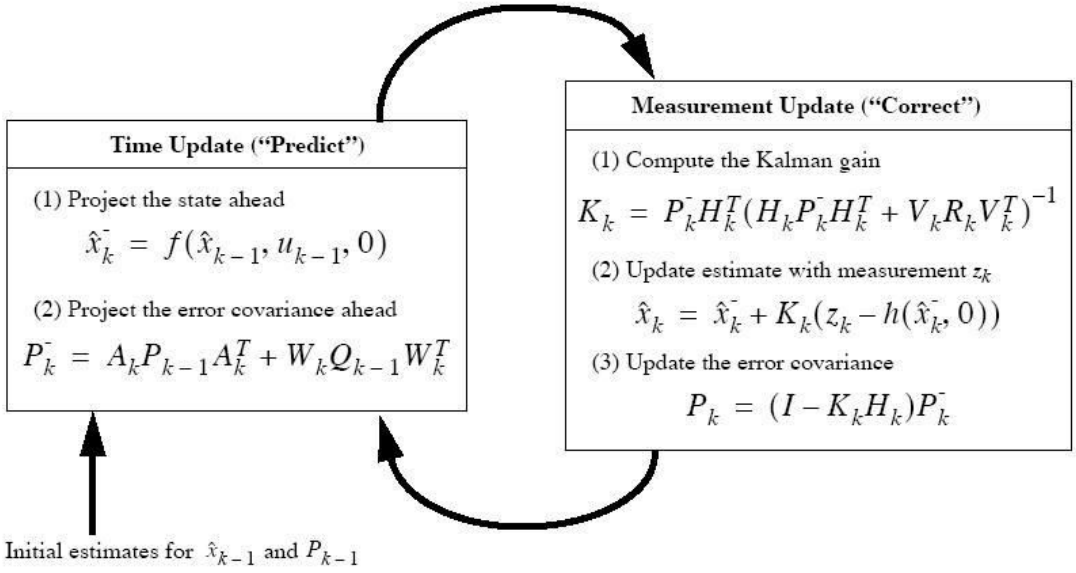


Figure 10: The EKF two step process. The $z_k$ shown in the figure is our measurement $y_k$, and you can assume $V_k = 1$ and $W_k = 1$. Different authors use slightly different notations. Image Credit: Welch and Bishop [5].

The state equation is linear so we can use that $\bar{A}_k = A$.

To develop a measurement process, we will assume that we have non-linear radar range and bearing measurements described as:

$$y_k = h(x_k, k) + w_k = \begin{bmatrix} \sqrt{(x_k^c)^2 + (y_k^c)^2} \\ \tan^{-1}\left(\frac{y_k^c}{x_k^c}\right) \end{bmatrix} + w_k$$

Then, from (5.24), we have that

$$\begin{aligned}
\bar{H}_k(\bar{x}) &= \frac{\partial}{\partial x} h(x,k)\Big|_{x=\bar{x}} \\
&= \begin{bmatrix} \frac{\partial}{\partial x}\sqrt{(x_k^c)^2+(y_k^c)^2}\big|_{x=\bar{x}} \\ \frac{\partial}{\partial x}\tan^{-1}\left(\frac{y_k^c}{x_k^c}\right)\Big|_{x=\bar{x}} \end{bmatrix} \\
&= \begin{bmatrix} \frac{\partial}{\partial x^c}\sqrt{(x_k^c)^2+(y_k^c)^2} & \frac{\partial}{\partial y^c}\sqrt{(x_k^c)^2+(y_k^c)^2} & 0 & 0 \\ \frac{\partial}{\partial x^c}\tan^{-1}\left(\frac{y_k^c}{x_k^c}\right) & \frac{\partial}{\partial y^c}\tan^{-1}\left(\frac{y_k^c}{x_k^c}\right) & 0 & 0 \end{bmatrix}\Bigg|_{x=\bar{x}} \\
&= \begin{bmatrix} \frac{x_k^c}{\sqrt{(x_k^c)^2+(y_k^c)^2}} & \frac{y_k^c}{\sqrt{(x_k^c)^2+(y_k^c)^2}} & 0 & 0 \\ \frac{-y_k^c}{(x_k^c)^2+(y_k^c)^2} & \frac{x_k^c}{(x_k^c)^2+(y_k^c)^2} & 0 & 0 \end{bmatrix}\Bigg|_{x=\bar{x}} \\
&= \begin{bmatrix} \frac{\bar{x}^c}{\sqrt{(\bar{x}^c)^2+(\bar{y}^c)^2}} & \frac{\bar{y}^c}{\sqrt{(\bar{x}^c)^2+(\bar{y}^c)^2}} & 0 & 0 \\ \frac{-\bar{y}^c}{(\bar{x}^c)^2+(\bar{y}^c)^2} & \frac{\bar{x}^c}{(\bar{x}^c)^2+(\bar{y}^c)^2} & 0 & 0 \end{bmatrix}. \qquad (5.27)
\end{aligned}$$

We highlight that the state location $\bar{x}$ is not substituted until the last line. The step from line 3 to line 4 uses the chain rule, and the following results for partial derivatives

$$\frac{\partial x^2}{\partial x^1} = \frac{\partial x^1}{\partial x^2} = 0.$$

and

$$\frac{\partial x^1}{\partial x^1} = \frac{\partial x^2}{\partial x^2} = 1.$$

The first of these relationships does not hold for "total" derivatives.

**The EKF recursion** Assume that we believe the initial target state is $x_0 = [10000\ m, 10000\ m, 1\ m/s, 0\ m/s]$. That is, initially the target has 2D location $(10000\ m, 10000\ m)$ with a 2D velocity of $1\ m/s$ in the direction of the $x$ co-ordinate. Based on our confidence about the initial estimate we set $P_0 = 1000 I_{4\times 4}$, where $I_{4\times 4}$ is the $4 \times 4$ identify matrix.

Substitution into (5.25) and(5.26) gives that the EKF recursion (prove as an exercise) for this problem is

$$\begin{aligned}
\widehat{x}_k &= A\widehat{x}_{k-1} + K_k\left(y_k - h\left(A\widehat{x}_{k-1}, k\right)\right) \\
K_k &= P_k^- \bar{H}_k(\widehat{x}_k^-)'(\bar{H}_k(\widehat{x}_k^-)P_k^-\bar{H}_k(\widehat{x}_k^-)' + R_k)^{-1} \\
P_{k+1}^- &= A(P_k^- - K_k\bar{H}_k(\widehat{x}_k^-)P_k^-)A' + Q_k \qquad (5.28)
\end{aligned}$$

where we use (5.27) and values from (3.2).

Substitution into (5.25) and(5.26) gives the EKF recursion (prove as an exercise) for this problem.

### 5.3.2 A second EKF example with MATLAB code

In the MATLAB example given in Tutorial written by Jose Manuel Rodriguez, Universidad Politecnica de Cataluña, Spain considers the signal model:

$$\begin{aligned} \begin{bmatrix} x^1_{k+1} \\ x^2_{k+1} \end{bmatrix} &= \begin{bmatrix} \sin(x^2_k k) + v_k \\ x^2_k \end{bmatrix} \\ y_k &= \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x^1_k \\ x^2_k \end{bmatrix} + w_k \end{aligned} \tag{5.29}$$

with $\sigma_v^2 = 0.01$ and $\sigma_w^2 = 0.25$. Here $x^1_k$ and $x^2_k$ are described as position and frequency, but their physical basis isn't overly important for gaining an understanding of the EKF.

Hence, in our model notation (5.23), we have the state update and measurement functions

$$\begin{aligned} f(x_k, k) &= \begin{bmatrix} \sin(x^2_k k) \\ x^2_k \end{bmatrix} \\ h(x_k, k) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x_k. \end{aligned} \tag{5.31}$$

Therefore, after consideration of the partial fraction operations, the matrices appearing in the EKF recursions become

$$\bar{A}_k(\bar{x}) = \begin{bmatrix} \cos(\bar{x}^2 k) \times k \\ \bar{x}^2 \end{bmatrix}$$

$$\bar{H}_k(\bar{x}) = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

$$Q = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & 0 \end{bmatrix} \tag{5.32}$$

$$R = \sigma_w^2. \tag{5.33}$$

These equations are implemented in the code you can download from https://au.mathworks.com/matlabcentral/fileex kalman-filter-tutorial.

## 5.4 Particle filter

The particle filter (also known as a Sequential Monte Carlo method) is a non-parametric technique that attempts to represent the posterior probability density of the system state by a system of particles that evolve according to the non-linear system [13]. The particle filter algorithm can be thought of as a variant of the Bayes filter based on importance sampling (a technique where an assumed pdf is used as a proxy for a sampling approximation of the system under study).

A detailed introduction can be found at [14] or [1, Ch. 4.3].

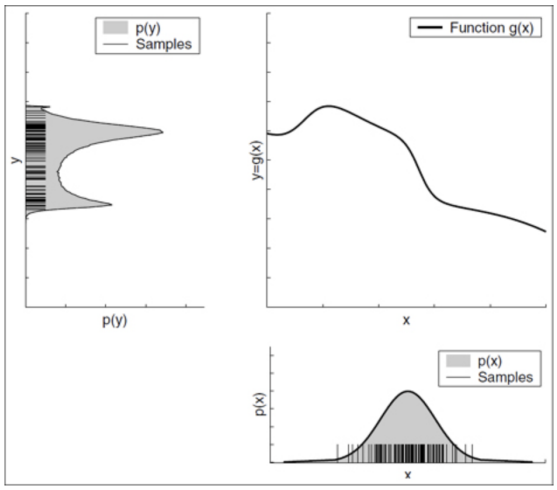


**Figure 4.3** The "particle" representation used by particle filters. The lower right graph shows samples drawn from a Gaussian random variable, $X$. These samples are passed through the nonlinear function shown in the upper right graph. The resulting samples are distributed according to the random variable $Y$.

Figure 11: Partciles used an approximate representation of non-gaussian pdfs (relative to Gaussian sampling). Image Credit: S. Thrun, W. Burgard, and D. Fox [1].

# 6 First look - decision making in sequential setting

Quickly detecting a change in the statistics of a process, whilst you are monitoring the process, is an important signal-processing problem with applications in a diverse range of areas including automatic control, quality control, statistics, target detection and many more. An example use case in robotics applications is the during-operation detection of an actuator fault or the emergence of an obstacle to avoid (e.g. the emergence of an aircraft anywhere in an image that needs to be quickly detected for collision avoidance purposes) [9].

In the simplest version of the decision problem, we might allow for the possibility of change in the statistics of an observed process at some random time. Our decision objective might be to create an alert when we believe a change has occurred in a manner that our alert decision-making strategy has the property of minimising the average detection delay subject to a constraint on the probability of a false alarm [8].

Interestingly, this problem can be considered with some rigour within a statistics optimisation problem. There are a variety of mathematical assumptions to make analysis rigorous but one of the more important problem formulations (the Bayesian formulation) provides a solution that is a remarkably simple and elegant threshold test on the conditional posterior of the fault or event.

### 6.1 Problem statement: Bayesian Quickest Change Detection

Let us assume that the change event $\nu$ is a random variable with a geometric prior (see [8] for meaning). Let $P_\pi$ and $E_\pi$ be associated probability and expectation operations.

We can now state our QCD problem as seeking to quickly detect a change in the statistical properties of $X_k$ in the sense of designing a stopping time $\tau \geq 1$ that minimises the following cost (Bayes risk)

$$J(\tau) \triangleq cE_\pi\left[(\tau-\nu)^+\right] + P_\pi(\tau < \nu) \tag{6.34}$$

where $(\tau-\nu)^+ \triangleq \max(0, \tau-\nu)$ and $c$ is the penalty of each time step that alert is not declared after $\nu$. Here the first term of $J(\tau)$ is a penalty on detection delay and the second term $P_\pi(\tau < \nu)$ is a penalty on false alarms.

### 6.2 Optimal solution: Bayesian Quickest Change Detection

Let us define the conditional posterior probability of no change (i.e. that $k \leq \nu$) based on the measurements received up until the current time $k$ as $\widehat{M}_k^0 = P_\pi(k \leq \nu | y_{[0,k]})$, then the following theorem holds [8].

**Theorem 6.1** *For the cost criteria (6.34) the exactly optimal stopping rule $\tau^*$ is given as*

$$\tau^* \triangleq \{k \geq 1 : \widehat{M}_k^0 \leq h\} \tag{6.36}$$

*for some threshold value $h \in [0,1]$.*

The important interpretation of this result is that the intuitive idea of using the conditional posterior probability of the fault event as a test of when to stop (or change behaviours) is well grounded in theory. Fortunately, there are well-known ways of numerically calculating posteriors such as $\widehat{M}_k^0$ (for example, might involve HMM filter recursion provided earlier in these notes to calculate $\widehat{X}_k$ (which as shown in [8] is an efficient representation of the test statistis).

### 6.3 Other detectors

There are a range of important sequential fault detectors such as CUSUM, but the common feature is the calculation of a test statistic based on measurements so far, and then testing this statistic against a threshold to determine if you should stop.

## 7 Application Example: Integrated Navigation

Some additional filtering and estimation concepts are shown in a more details example application,

## 7.1 Basic Operation of a Complementary Filter

We begin by considering a general continuous-time system model. Consider a situation where we have two independent measurements of the same signal:

$$\begin{aligned} z_1(t) &= s(t) + n_1(t) \\ z_2(t) &= s(t) + n_2(t) \end{aligned}$$

where $n_1(t)$, $n_2(t)$ are noise process (not necessarily gaussian), $s(t)$ is the signal of interest, and $z_1(t)$, $z_2(t)$ are independent measurements.

If the statistical properties (mean and covariance) of the noises $n_1$ and $n_2$ are not known, then the KF is not the appropriate filtering solution. Instead, the signal can be recovered using the filtering approach shown in Figure 12, for some choice of $H$. Note the signal passes through regardless of $H$. Hence, the design problem becomes to determine the $H$ that minimizes the influence of $n_1$ and $n_2$.

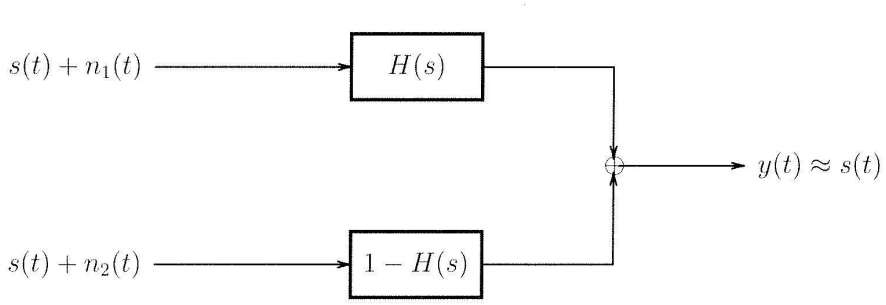


Figure 12: The complementary filter. Image Credit: Bar-shalom [3].

## 7.2 Complementary Filtering for Navigation

Historically, complementary filtering has been a common approach to the integrated navigation problem; for both computational and integrity reasons (onboard computational effort was limited, and complementary filtering approaches tend to limit the impact of bad measurements).

Moreover, due to the lack of GPS integrity information, GPS-aided INS (open-loop) has been preferred over an INS-aided GPS approach. This highlights that integrity considerations have often prevented GPS from being used as the primary navigation device.

### 7.2.1 Open-loop Complementary GPS-aided INS

Consider the open-loop GPS-aided INS shown in Figure 13. Let the abbreviation PVT denote position-velocity-time information (as distinct from raw measurement information). In this figure, we have

separate GPS and INS solutions providing independent state estimates (in PVT-form). The difference between these two PVT solutions is "filtered" to provide an estimate of the errors in the INS-only estimate. This error information can then be used to aid (or correct) the original INS solution.

This configuration is called open-loop because no information is feedback to either the INS or GPS processing blocks.

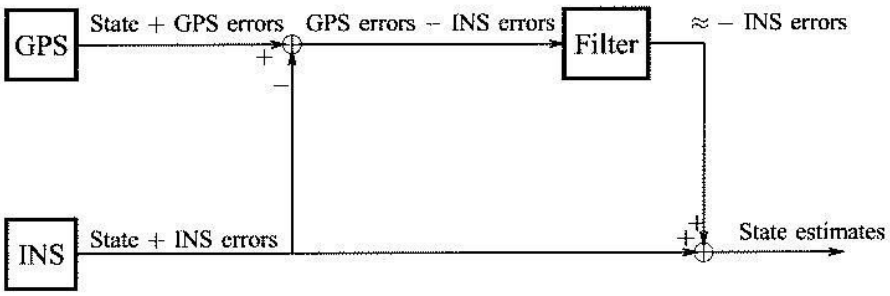


Figure 13: Open-loop GPS-aided INS. Image Credit: Bar-Shalom [3].

To see that this a complementary filtering approach (applied to the GPS/INS integration problem) compare Figure 13 with Figure 12. Note that both the INS and GPS solutions provide independent PVT solutions, and the block marked "filter" functions as the $H$ in the complementary filter.

In KF language, we might say that the INS solution is being used as a nominal trajectory about which linearization is used so that a linearised Kalman filter (the "filter" block in Figure 13) can be applied to GPS measurements.

## Remarks

20. Note, Figure 13 has written "estimate=state + error". An estimate can always be written this way.

### 7.2.2 Closed-loop Complementary GPS-aided INS

Alternatively, consider the closed-loop GPS-aided INS shown in Figure 14. The key difference from the previous configuration is that some error information is feedback to correct the INS solution (inside the INS processing block).

The feedback use of the error information is expected to improve INS performance. We expect the INS to "converge" to the best solution by "bootstraping" our linearisation point onto the best estimates. The key disadvantage of the approach is that any error induced by faulty GPS measurement will persist in the filter, and might "break" the complementary filter. If broken, the filter might need to be reset.

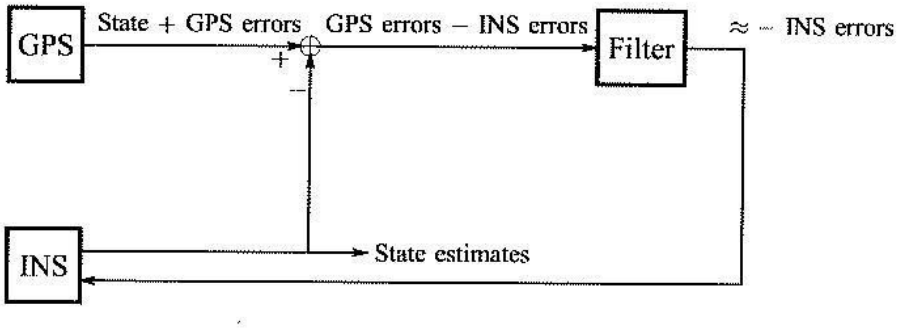


Figure 14: Closed-loop GPS-aided INS. Image Credit: Bar-shalom [3].

### 7.3 Centralised Filtering for Navigation

Centralised filtering is also known as tightly coupled GPS/INS integration. Consider the centralised filtering approach shown in Figure 15.

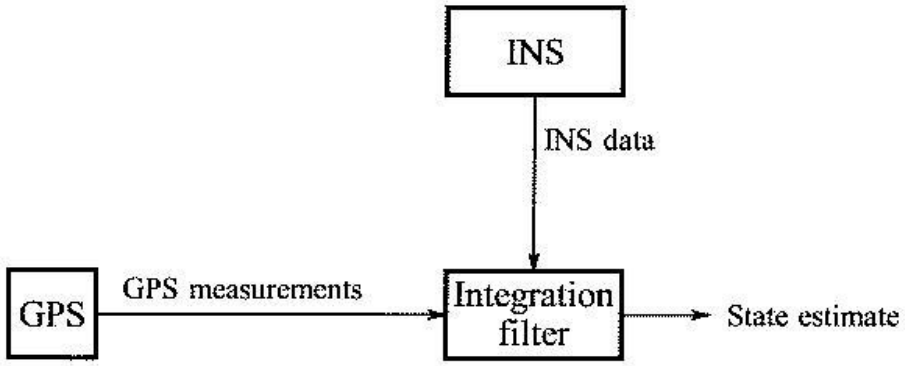


Figure 15: Centralised filtering for GPS/INS integration. Image Credit: Bar-Shalom [3].

This figure shows a conceptually straightforward and theoretically satisfying way to integrate GPS and INS measurements. The term "centralised" describes the fact that all the processing is performed in a central location (not spread out with some processing done in a "INS" processor and other processing done in an "GPS" processor). The phrase "tightly-coupled" highlights that the "INS" and "GPS" components have been incorporated, together, into a unified signal model. The key attribute of centralised filtering approach is the requirement for an integration filter that makes "optimal" use of all the available information (this sounds like a job for a KF!).

We highlight that the "INS data" and "GPS measurements" shown in Figure 15 are raw observables such as inertial rate measurements and pseudo-range measurements. These raw measurements are fed into integration filter that combines these measurements to produce one estimate of all desirable

navigation variables.

A tightly-coupled integration filter can be designed using a stochastic filtering problem that involves the following steps:

1. Develop appropriate dynamic model (discrete-time) of user motion (state process). This model includes the dynamics of error states, such as GPS user clock biases, and INS bias. Sometimes as many as 50 dynamic error states are considered.
2. Develop an appropriate description of the relationship between measurements (GPS and INS raw measurements) and the discrete-time state (measurement process).
3. Linearise both state and measurement process by developing the required EKF partial derivatives (5.24).
4. Implement the EKF recursion (5.25) and(5.26).

Two major disadvantages of a tightly coupled solution are its heavy computational load and its poor fault tolerance. In particular, it is difficult to ensure acceptable performance in situations involving bad measurements.

Further, centralised filtering approaches can have difficulty with:

- The different data rates of INS measurements and GPS measurements.
- The inclusion of additional sensors/measurement devices.

## 7.4 Decentralised Filtering for Navigation

Decentralised filtering approaches, or loosely coupled approaches, to the GPS/INS integration problem can be seen as a compromise between the simplicity of complementary approaches, and the complexity of tightly coupled approaches. Consider the loosely coupled solution shown in Figure 16.

Note in shown decentralised GPS/INS integration filter that:

- The raw GPS measurements are separately processed to the PVT-stage (with some INS aiding in the GPS acquisition process).
- The raw INS measurements are separately processed to the PVT stage (with some feedback of corrected information).
- The GPS solution (in PVT-form) and INS solution (in PVT-form) are combined in the integration filter to produce an integrated navigation solution.

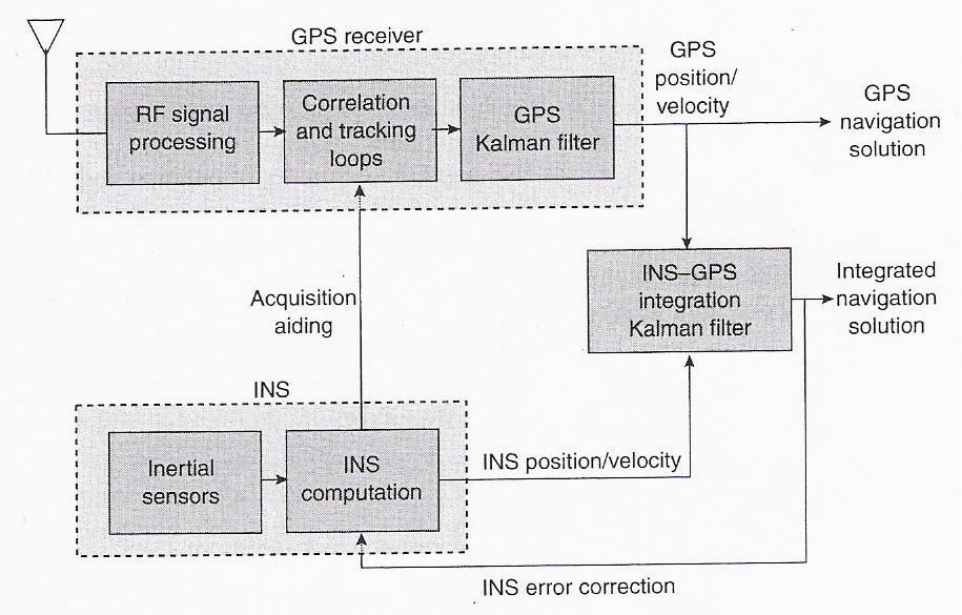


Figure 16: Loosely coupled GPS/INS. Image Credit: Bar-Shalom [3].

This approach lies somewhere between the closed-loop complementary filter and the tightly coupled integration filter. In particular, this approach is different from GPS-aided INS because the navigation solution is the output of an integration filter. Moreover, this approach is different from a tightly coupled approach, because the separated processed GPS and INS solutions (in PVT form) are inputs to the integration filter (not raw GPS and INS measurements).

## 7.5 Extended Kalman Filter(EKF) for GPS example in MATLAB

You might find it helpful to review the Yi Cao (2021). Extended Kalman Filter(EKF) for GPS on MATLAB Central File Exchange.

This example "contains a brief illustration of the principles and algorithms of both the Extended Kalman Filtering (EKF) and the Global Position System (GPS). It is designed to provide a relatively easy-to-implement EKF. It also serves as a brief introduction to the Kalman Filtering algorithms for GPS. In the example for the EKF, we provide the raw data and solution for GPS positioning using both EKF and the Least Square method."

## 7.6 Summary: Integration Approaches in Comparison

In order of increasing complexity, possible integrated navigation solutions are:

- A open-loop complementary filter approach for GPS-aided INS. This would combine the existing

GPS solution (in PVT-form) with an INS solution (in PVT-form). A KF can be used to estimate the correction to stand-alone INS solutions.

- A closed-loop complementary filter approach for GPS-aided INS. In this closed-loop solution, estimated information is feedback to correct the INS solution process (possibly as position/attitude corrections, or as bias correction on raw inertial measurements).
- A loosely-coupled GPS/INS integration. Involves the design of a filter that mixes the GPS solution (in PVT-form) with the INS solution (in PVT-form) to create a new integrated solution. Information is also feedback to INS block.
- Tightly coupled integration.

Students are recommended to initially attempt the implementation of an open-loop complementary filter approach for GPS-aided INS.

## References

[1] S. Thrun, W. Burgard, and D. Fox, Probabilistic robotics. Cambridge, Massachusetts: The MIT Press, 2006. QUT Library online link: https://qut.primo.exlibrisgroup.com/permalink/61QUT_INST/1tofoc8/alma991010282728704001

[2] P. I. Corke, Robotics and control: fundamental algorithms in MATLAB. Cham, Switzerland: Springer, 2022. QUT Library online link: https://qut.primo.exlibrisgroup.com/permalink/61QUT_INST/1g7tbfa/alma991010244357104001

[3] Y. Bar-shalom, X. Rong Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation: Theory, Algorithms, and Software*, Wiley, New York, 2001 (AKA "Bar-shalom's Purple Book", he also has a black book). QUT Library online link: https://qut.primo.exlibrisgroup.com/permalink/61QUT_INST/1g7tbfa/alma991003521119704001

[4] R.G. Brown and P.Y.C. Hwang, *Introduction to Random Signal and Applied Kalman Filtering*, 3rd Ed., Wiley, New York, 1997. This book is also purple. QUT Library online link: https://qut.primo.exlibrisgroup.com/permalink/61QUT_INST/1g7tbfa/alma991010501097104001

[5] G. Welch and G. Bishop, The Kalman Filter, http://www.cs.unc.edu/~welch/kalman/ (Availability checked April 2023).

[6] D.H. Titterton and J.L. Weston, *Strapdown inertial navigation technology*, 2nd Ed, IEE, Stevenage, UK, 2004. QUT Library online link: https://qut.primo.exlibrisgroup.com/permalink/61QUT_INST/1tofoc8/alma991009351423504001

[7] J Lai, J. Ford, L. Mejias Alvarez, P. O'Shea, & R. Walker, "See and avoid using onboard computer vision," In Plamen, A (Ed.) Sense and Avoid in UAS: Research and Applications. John Wiley & Sons, United States, pp. 265-294, 2012. QUT eprints link: https://eprints.qut.edu.au/50789/16/50789a.pdf

[8] Jason J. Ford, Jasmin James, Timothy L. Molloy, "On the informativeness of measurements in Shiryaev's Bayesian quickest change detection", Automatica, Volume 111, 2020, 108645, ISSN 0005-1098, https://doi.org/10.1016/j.automatica.2019.108645. QUT eprints link: https://eprints.qut.edu.au/133612/

[9] J. James, J. J. Ford and T. L. Molloy, "Quickest Detection of Intermittent Signals With Application to Vision-Based Aircraft Detection," in IEEE Transactions on Control Systems Technology, vol. 27, no. 6, pp. 2703-2710, Nov. 2019, doi: 10.1109/TCST.2018.2872468. QUT eprints link: https://eprints.qut.edu.au/122352/

[10] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," in Proceedings of the IEEE, vol. 77, no. 2, pp. 257-286, Feb. 1989, doi: 10.1109/5.18626.

[11] Hidden Markov model posterior state probabilities https://au.mathworks.com/help/stats/hmmdecode.html

[12] Hidden Markov model most probable state path https://au.mathworks.com/help/stats/hmmviterbi.html

[13] J.J. Ford, "Non-linear and robust filtering: From the Kalman filter to the particle filter". Aeronautical and Maritime Research Laboratory, Melbourne, Vic. 2002. https://eprints.qut.edu.au/108495/

[14] Doucet, A., de Freitas, N., Gordon, N. (2001). An Introduction to Sequential Monte Carlo Methods. In: Doucet, A., de Freitas, N., Gordon, N. (eds) Sequential Monte Carlo Methods in Practice. Statistics for Engineering and Information Science. Springer, New York, NY. https://doi.org/10.1007/978-1-4757-3437-9_1 QUT Library online link: https://qut.primo.exlibrisgroup.com/permalink/61QUT_INST/1g7tbfa/alma991009913309504001

[15] I. Mareels and J. W. Polderman, Adaptive Systems An Introduction, 1st ed. 1996. Boston, MA: Birkhäuser Boston, 1996. doi: 10.1007/978-0-8176-8142-5. Link to QUT library ebook https://qut.primo.exlibrisgroup.com/permalink/61QUT_INST/1g7tbfa/alma991010143442104001