SES 598: Space Robotics and AI

Lecture 4-5 - January 22-27, 2026,
State space models, State estimation, Kalman Filtering, Assignment 0-1

WEEK	TOPICS	LECTURES & ASSIGNMENTS	RELATED RESOURCES
1-3 (Jan 13-Jan 30) State estimation and Controls	• Least squares and maximum likelihood estimation (MLE) • State space models and linear dynamical systems • State-estimation with Kalman and particle filters • PID control, linear quadratic regulator (LQR), and model predictive control (MPC) • Entry descent and landing (EDL), guidance navigation and control (GNC), and attitude determination and control system (ADCS)	Assignment 1: First-Order Boustrophedon Navigator (Lawnmower pattern) using ROS2 (Due: Jan 27, 2026)	**Papers:** • MPC for Quadrotor Flight • Mars 2020 EDL • Psyche Mission GNC **Tutorials:** • Parameter Estimation Tutorial
4-5 (Feb 3-Feb 20) Computer Vision and 3D Reconstruction	• Image formation and camera models	Assignment 2: Optimal Control of	**Papers:** • DUSt3R

Recap

Least-squares estimation

Joint and conditional probabilities

Maximum Likelihood estimation

Bayes rule (what is a Bayes filter?)

Assignments

Hawaii/pacific plate velocity estimation

Lawnmower survey with first order system, 2D

Cart-pole balancing (2nd order), 2D

Drone based mapping (2+ order, 3D)

2009 IEEE International Conference on Robotics and Automation
Kobe International Conference Center
Kobe, Japan, May 12-17, 2009

3D Tree Reconstruction from Laser Range Data

Jonathan Binney and Gaurav S. Sukhatme

***Abstract*— We present a method for reconstructing 3D models of tree branch structure from laser range data. Our approach is probabilistic, and uses general knowledge of tree structure to guide an iterative reconstruction process. Our goal is to recover parameters such as branch locations, angles, radii, and lengths, as well as connectivity information between branches. These parameters can then be fed into functional-structural plant models to study the relationships between the structure of a plant, its environment, and its internal biology. In this paper we present an algorithm for finding these parameters, and results on both simulated and real datasets.**

I. Introduction

Studies of tree architecture in relation to tree biology and environmental factors often require detailed metric data about the tree structure, such as branch structure, lengths, radii, and angles, as well as leaf areas and positions. [1]. Using this information, computer models of tree function can be created [2], giving valuable information about tree development.

Acquiring this metric information can be extremely time consuming. In one study of an adult walnut tree, [3], measuring the tree architecture by hand using a 3D magnetic measuring device which was moved along the branches took several weeks. We hope to provide a method of acquiring detailed metric information about the branch architecture of a tree which is relatively quick and requires significantly less manual labor. In this paper we address the measurement of the branch structure, but we plan to extend our technique to provide data about the leaves of the tree as well.

This problem is well suited to some of the methods used in probabilistic robotics. Using sensors to infer things about its environment is one of the core problems for a robot. This task is often broken down into a number of sub-problems, including localizing the sensors at each step, associating the sensor data with a part of the robot's environment model, and fitting the model to the data. One example of a method which solves the data association and model fitting problem is [4], which fits a set of planes to laser range data. For the tree problem, the model which we want to fit to sensor data is much more complex.

The method presented in this paper works on laser range data, but can be extended to any sensor for which we can calculate the likelihood given a specific set of tree model parameters.

Jonathan Binney (binney@usc.edu) and Gaurav S. Sukhatme (gaurav@usc.edu) are with the Robotic Embedded Systems Laboratory and Department of Computer Science at the University of Southern California (USC). This work was funded in part by the National Science Foundation under grants CNS-0540420, CNS-0331481, and CCF-0120778 and a gift from the Okawa Foundation. Jonathan Binney was supported by a doctoral fellowship from USC.

II. Related Work

Previous work on reconstructing tree models from sensor data falls into two main types. The first focuses on methods for creating visually appealing, but not necessarily accurate, models. The second focuses on accurately determining some parameters of the tree, but often requires a large amount of user interaction.

A method for reconstructing realistic looking tree models from laser scan data is described in [5]. The laser data is first converted to a point cloud, and then a graph-based technique is used to find the rough branching structure. In order to make the tree more visually appealing, fake branches are added according to the expected structure of the tree.

A graph-based technique is also employed in [6] to find overall branch structure, but images are used as input instead of laser range scans. A structure from motion algorithm is used to create a 3D point cloud from the images. The point cloud and the raw images are used to find the branching structure. Visible branches are copied to fill occluded areas and create a visually appealing tree.

The algorithm in [7] and [8] takes laser data as input, and creates a voxel-based occupancy grid representation of the data. It uses morphological operations to find the underlying branching structure, and fits cylinders for the branches. The algorithm is used to extract metric parameters of the tree (e.g. branch radius).

An approach based on instrumented photographs is described in [9] which generates a model that approximates the appearance of a specific tree, although not its exact structure.

Our work is most similar to [10], which uses a generative statistical model to fit trees to image data. It uses a straight cylinder to model each branch, and generates possible trees using L-systems. In [10] complete trees are generated, and then compared against each image.

We use a generative statistical approach similar to [10], because it allows us to use complex sensor models without solving a complex inverse problem. It also allows us to make use of prior knowledge about tree structure in an intuitive way, by including that information in the generative model of the tree. Unlike [10], we use a detailed model of the tree which allows for curved branches. We use laser range data instead of images because it gives us accurate depth information. By making use of priors on the tree structure, we hope to create a method which is tolerant to noise.

III. Approach

In order to reconstruct a tree from sensor data, we use a model of the tree to generate likely hypotheses, and then use a sensor model to evaluate the likelihood of each hypothesis.

978-1-4244-2789-5/09/$25.00 ©2009 IEEE

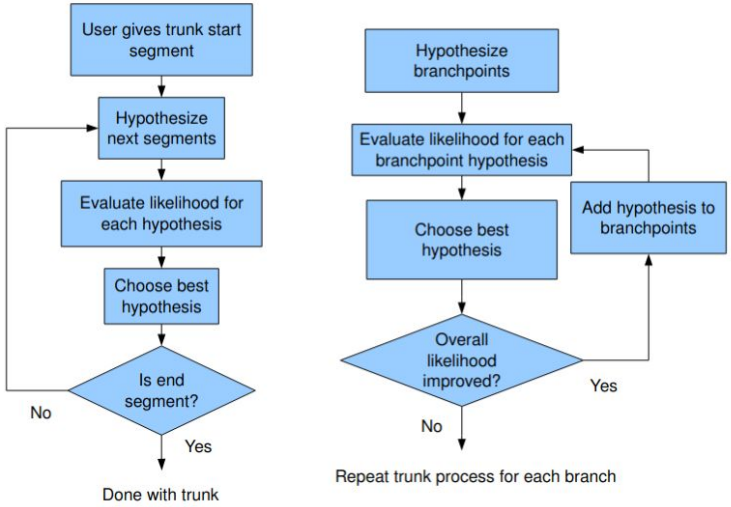
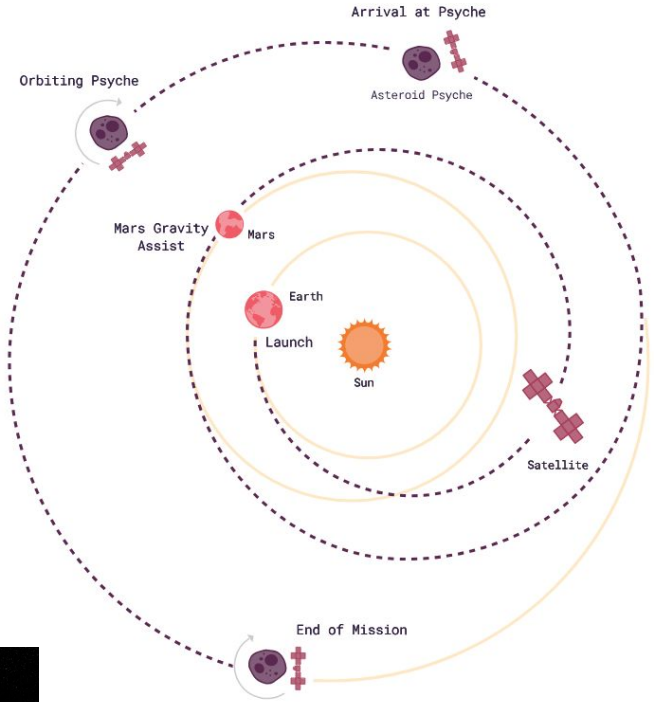
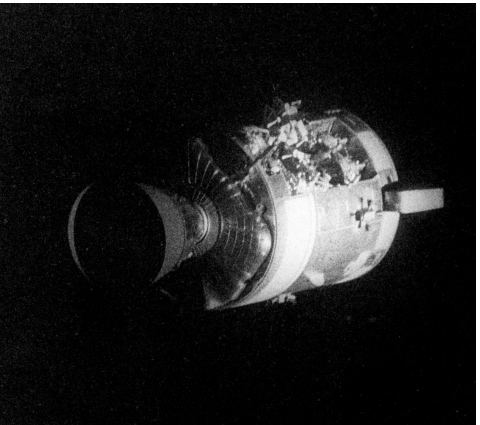


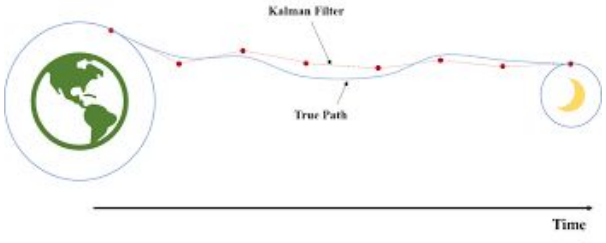
Fig. 1. Overview of Algorithm

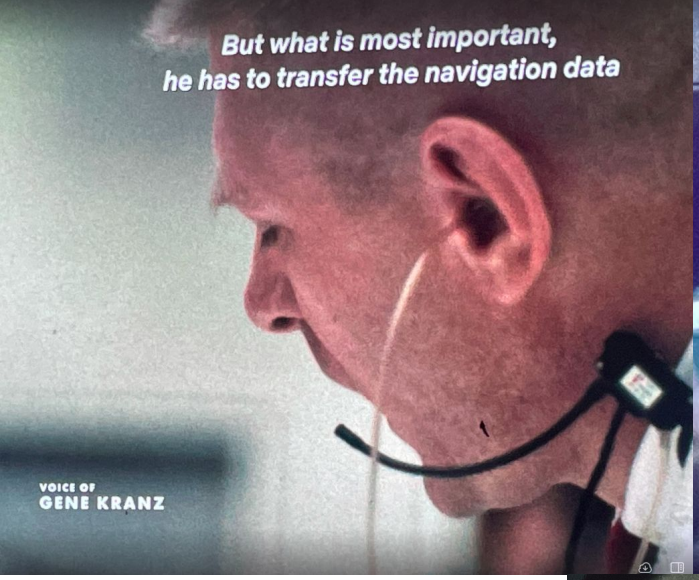
Example of MLE being used.
Expectation Maximization (EM algo) is used to do MLE
K-means _could_ be considered a special case of EM algorithm

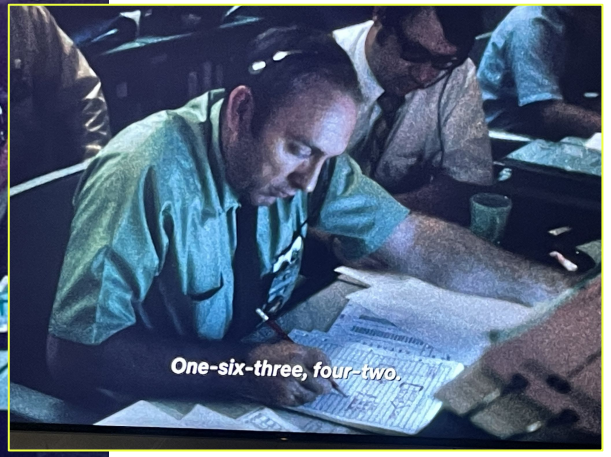
Guidance, Navigation, and Control (GNC)



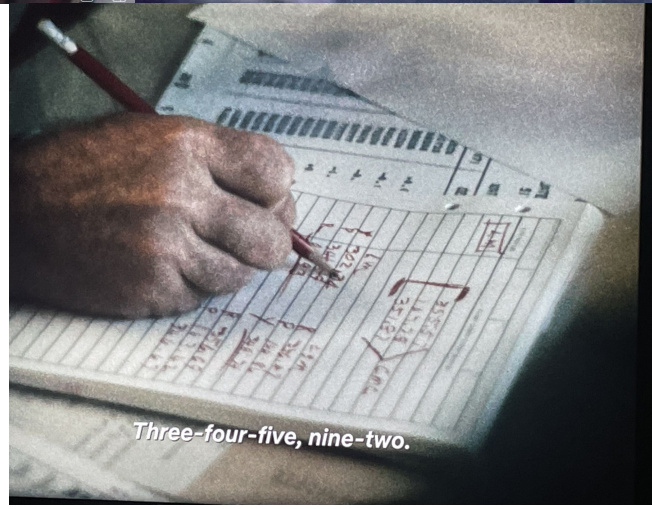


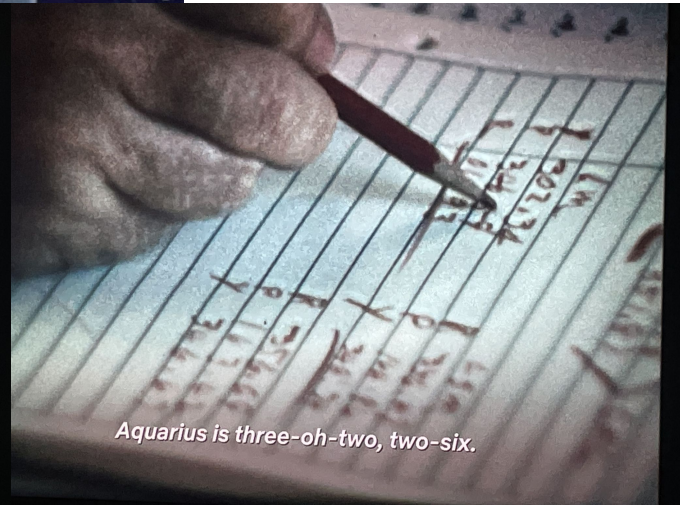






What's happening at mission control?

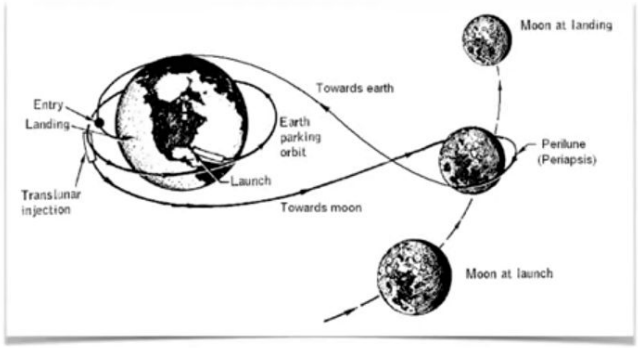




Kalman Filter

Rudolf Kalman received the National Medal of Science on Oct. 7, 2009, from President Barack Obama.

Apollo Guidance Computer



The (extended) Kalman Filter became widely known after its use in the Apollo Guidance Computer for circumlunar navigation.

A New Approach to Linear Filtering and Prediction Problems[1]

R. E. KALMAN
Research Institute for Advanced Study,[2] Baltimore, Md.

The classical filtering and prediction problem is re-examined using the Bode-Shannon representation of random processes and the "state transition" method of analysis of dynamic systems. New results are:

(1) The formulation and methods of solution of the problem apply without modification to stationary and nonstationary statistics and to growing-memory and infinite-memory filters.

(2) A nonlinear difference (or differential) equation is derived for the covariance matrix of the optimal estimation error. From the solution of this equation the coefficients of the difference (or differential) equation of the optimal linear filter are obtained without further calculations.

(3) The filtering problem is shown to be the dual of the noise-free regulator problem. The new method developed here is applied to two well-known problems, confirming and extending earlier results.

The discussion is largely self-contained and proceeds from first principles; basic concepts of the theory of random processes are reviewed in the Appendix.

Introduction

AN IMPORTANT class of theoretical and practical problems in communication and control is of a statistical nature. Such problems are: (i) Prediction of random signals; (ii) separation of random signals from random noise; (iii) detection of signals of known form (pulses, sinusoids) in the presence of random noise.

In his pioneering work, Wiener [1][3] showed that problems (i) and (ii) lead to the so-called Wiener-Hopf integral equation; he also gave a method (spectral factorization) for the solution of this integral equation in the practically important special case of stationary statistics and rational spectra.

Many extensions and generalizations followed Wiener's basic work. Zadeh and Ragazzini solved the finite-memory case [2]. Concurrently and independently of Bode and Shannon [3], they also gave a simplified method [2] of solution. Booton discussed the nonstationary Wiener-Hopf equation [4]. These results are now in standard texts [5-6]. A somewhat different approach along these main lines has been given recently by Darlington [7]. For extensions to sampled signals, see, e.g., Franklin [8], Lees [9]. Another approach based on the eigenfunctions of the Wiener-Hopf equation (which applies also to nonstationary problems whereas the preceding methods in general don't), has been pioneered by Davis [10] and applied by many others, e.g., Shinbrot [11], Blum [12], Pugachev [13], Solodovnikov [14].

In all these works, the objective is to obtain the specification of a linear dynamic system (Wiener filter) which accomplishes the prediction, separation, or detection of a random signal.[4]

Present methods for solving the Wiener problem are subject to a number of limitations which seriously curtail their practical usefulness:

(1) The optimal filter is specified by its impulse response. It is not a simple task to synthesize the filter from such data.

(2) Numerical determination of the optimal impulse response is often quite involved and poorly suited to machine computation. The situation gets rapidly worse with increasing complexity of the problem.

(3) Important generalizations (e.g., growing-memory filters, nonstationary prediction) require new derivations, frequently of considerable difficulty to the nonspecialist.

(4) The mathematics of the derivations are not transparent. Fundamental assumptions and their consequences tend to be obscured.

This paper introduces a new look at this whole assemblage of problems, sidestepping the difficulties just mentioned. The following are the highlights of the paper:

(5) *Optimal Estimates and Orthogonal Projections.* The Wiener problem is approached from the point of view of conditional distributions and expectations. In this way, basic facts of the Wiener theory are quickly obtained; the scope of the results and the fundamental assumptions appear clearly. It is seen that all statistical calculations and results are based on first and second order averages; no other statistical data are needed. This difficulty (4) is eliminated. This method is well known in probability theory (see pp. 75–78 and 148–155 of Doob [15] and pp. 455–464 of Loève [16]) but has not yet been used extensively in engineering.

(6) *Models for Random Processes.* Following, in particular, Bode and Shannon [3], arbitrary random signals are represented (up to second order average statistical properties) as the output of a linear dynamic system excited by independent or uncorrelated random signals ("white noise"). This is a standard trick in the engineering applications of the Wiener theory [2–7]. The approach taken here differs from the conventional one only in the way in which linear dynamic systems are described. We shall emphasize the concepts of *state* and *state transition;* in other words, linear systems will be specified by systems of first-order difference (or differential) equations. This point of view is

[1] This research was supported in part by the U. S. Air Force Office of Scientific Research under Contract AF 49 (638)-382.

[2] 7212 Bellona Ave.

[3] Numbers in brackets designate References at end of paper.

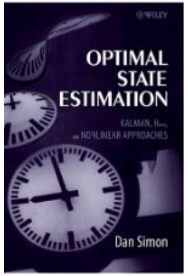
[4] Of course, in general these tasks may be done better by nonlinear filters. At present, however, little or nothing is known about how to obtain (both theoretically and practically) these nonlinear filters.

Contributed by the Instruments and Regulators Division and presented at the Instruments and Regulators Conference, March 29–April 2, 1959, of THE AMERICAN SOCIETY OF MECHANICAL ENGINEERS.

NOTE: Statements and opinions advanced in papers are to be understood as individual expressions of their authors and not those of the Society. Manuscript received at ASME Headquarters, February 24, 1959. Paper No. 59—IRD-11.

Transactions of the ASME–Journal of Basic Engineering, 82 (Series D): 35-45. Copyright © 1960 by ASME

Kalman Filter - Part 1, Prof. Jonathan Kelly, Univ. of Toronto https://www.youtube.com/watch?v=LioOvUZ1MiM



Good pathway to estimation and controls foundation

Least squares estimation

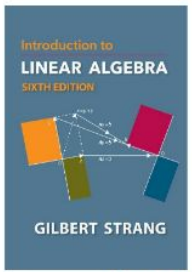
The most probable value of the unknown quantities will be that in which the sum of the squares of the differences between the actually observed and the computed values multiplied by numbers that measure the degree of precision is a minimum.
—Karl Friedrich Gauss [Gau04]

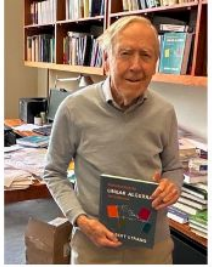
In this chapter, we will discuss least squares estimation, which is the basic idea of Karl Gauss's quote above.[1] The material in this chapter relies on the theory of the previous two chapters, and will enable us to derive optimal state estimators later in this book.

Section 3.1 discusses the estimation of a constant vector on the basis of several linear but noisy measurements of that vector. Section 3.2 extends the results of Section 3.1 to the case in which some measurements are more noisy than others; that is, we have less confidence in some measurements than in others. Sections 3.1 and 3.2 use matrices and vectors whose dimensions grow larger as more measurements are obtained. This makes the problem cumbersome if many measurements are available. This leads us to Section 3.3, which presents a recursive way of estimating a constant on the basis of noisy measurements. Recursive estimation in this chapter is a method of estimating a constant without increasing the computa-

[1] Gauss published his book in 1809, although he claimed to have worked out his theory as early as 1795 (when he was 18 years old).

Optimal State Estimation, First Edition. By Dan J. Simon
ISBN 0471708585 ©2006 John Wiley & Sons, Inc.





Foundational knowledge, general purpose

4.3 Least Squares Approximations

1 Solving $A^TA\hat{x} = A^Tb$ gives the projection $p = A\hat{x}$ of b onto the column space of A.

2 When $Ax = b$ has no solution, $\hat{x}$ is the "least-squares solution": $||b - A\hat{x}||^2 =$ minimum.

3 Setting partial derivatives of $E = ||Ax - b||^2$ to zero $\left(\frac{\partial E}{\partial x_i} = 0\right)$ also produces $A^TA\hat{x} = A^Tb$.

4 To fit points $(t_1, b_1), \ldots, (t_m, b_m)$ by a straight line, A has columns $(1, \ldots, 1)$ and $(t_1, \ldots, t_m)$.

5 In that case A^TA is the 2 by 2 matrix $\begin{bmatrix} m & \Sigma t_i \\ \Sigma t_i & \Sigma t_i^2 \end{bmatrix}$ and A^Tb is the vector $\begin{bmatrix} \Sigma b_i \\ \Sigma t_i b_i \end{bmatrix}$.

It often happens that $Ax = b$ has no solution. The usual reason is: *too many equations*. The matrix A has more rows than columns. There are more equations than unknowns (m is greater than n). The n columns span a small part of m-dimensional space. Unless all measurements are perfect, b is outside that column space of A. Elimination reaches an impossible equation and stops. But we can't stop just because measurements include noise!

To repeat: We cannot always get the error $e = b - Ax$ down to zero. When e is zero, x is an exact solution to $Ax = b$. *When the length of e is as small as possible, $\hat{x}$ is a least squares solution.* Our goal in this section is to compute $\hat{x}$ and use it. These are real problems and they need an answer.

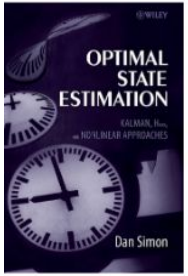
The previous section emphasized p (the projection). This section emphasizes $\hat{x}$ (the least squares solution). They are connected by $p = A\hat{x}$. The fundamental equation is still $A^TA\hat{x} = A^Tb$. Here is a short unofficial way to reach this "*normal equation*":

When $Ax = b$ has no solution, multiply by A^T and solve $A^TA\hat{x} = A^Tb$.

Example 1 A crucial application of least squares is fitting a straight line to m points. Start with three points: *Find the closest line to the points* $(0, 6), (1, 0)$, *and* $(2, 0)$.

No straight line $b = C + Dt$ goes through those three points. We are asking for two numbers C and D that satisfy three equations: $n = 2$ and $m = 3$. Here are the three equations at $t = 0, 1, 2$ to match the given values $b = 6, 0, 0$:

$t = 0$	The first point is on the line $b = C + Dt$ if	$C + D \cdot 0 = 6$
$t = 1$	The second point is on the line $b = C + Dt$ if	$C + D \cdot 1 = 0$
$t = 2$	The third point is on the line $b = C + Dt$ if	$C + D \cdot 2 = 0$.



Good pathway to estimation and controls foundation

Least squares estimation

The most probable value of the unknown quantities will be that in which the sum
the squares of the differences between the actually observed and the computed valu
multiplied by numbers that measure the degree of precision is a minimum.
—Karl Friedrich Gauss [Gau0

In this chapter, we will discuss least squares estimation, which is the basic ide
of Karl Gauss's quote above.[1] The material in this chapter relies on the theor
of the previous two chapters, and will enable us to derive optimal state estimato
later in this book.

Section 3.1 discusses the estimation of a constant vector on the basis of sever
linear but noisy measurements of that vector. Section 3.2 extends the results
Section 3.1 to the case in which some measurements are more noisy than other
that is, we have less confidence in some measurements than in others. Sections 3.
and 3.2 use matrices and vectors whose dimensions grow larger as more measur
ments are obtained. This makes the problem cumbersome if many measurement
are available. This leads us to Section 3.3, which presents a recursive way of e
timating a constant on the basis of noisy measurements. Recursive estimation i
this chapter is a method of estimating a constant without increasing the comput

[1]Gauss published his book in 1809, although he claimed to have worked out his theory as ear
as 1795 (when he was 18 years old).

Optimal State Estimation, First Edition. By Dan J. Simon
ISBN 0471708585

3.1 ESTIMATION OF A CONSTANT

In this section, we will determine how to estimate a constant on the basis of several noisy measurements of that constant. For example, suppose we have a resistor but we do not know its resistance. We take several measurements of its resistance using a multimeter, but the measurements are noisy because we have a cheap multimeter. We want to estimate the resistance on the basis of our noisy measurements. In this case, we want to estimate a constant scalar but, in general, we may want to estimate a constant vector.

To put the problem in mathematical terms, suppose x is a constant but unknown n-element vector, and y is a k-element noisy measurement vector. How can we find the "best" estimate $\hat{x}$ of x? Let us assume that each element of the measurement vector y is a linear combination of the elements of x, with the addition of some measurement noise:

$$\begin{aligned} y_1 &= H_{11}x_1 + \cdots + H_{1n}x_n + v_1 \\ &\vdots \\ y_k &= H_{k1}x_1 + \cdots + H_{kn}x_n + v_k \end{aligned} \tag{3.1}$$

This set of equations can be put into matrix form as

$$y = Hx + v \tag{3.2}$$

Now define ϵ_y as the difference between the noisy measurements and the vector $H\hat{x}$:

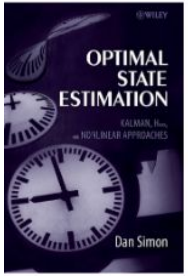
$$\epsilon_y = y - H\hat{x} \tag{3.3}$$

ϵ_y is called the measurement residual. As Karl Gauss wrote [Gau04], the most probable value of the vector x is the vector $\hat{x}$ that minimizes the sum of squares between the observed values y and the vector $H\hat{x}$. So we will try to compute the $\hat{x}$ that minimizes the cost function J, where J is given as

$$\begin{aligned} J &= \epsilon_{y1}^2 + \cdots + \epsilon_{yk}^2 \\ &= \epsilon_y^T \epsilon_y \end{aligned} \tag{3.4}$$

J is often referred to in control and estimation books and papers as a cost function, objective function, or return function. We can substitute for ϵ_y in the above equation to rewrite J as

$$\begin{aligned} J &= (y - H\hat{x})^T (y - H\hat{x}) \\ &= y^T y - \hat{x}^T H^T y - y^T H\hat{x} + \hat{x}^T H^T H\hat{x} \end{aligned} \tag{3.5}$$



Good pathway to estimation and controls foundation

Least squares estimation

The most probable value of the unknown quantities will be that in which the sum of the squares of the differences between the actually observed and the computed values multiplied by numbers that measure the degree of precision is a minimum.
—Karl Friedrich Gauss [Gau04]

In this chapter, we will discuss least squares estimation, which is the basic idea of Karl Gauss's quote above.[1] The material in this chapter relies on the theory of the previous two chapters, and will enable us to derive optimal state estimators later in this book.

Section 3.1 discusses the estimation of a constant vector on the basis of several linear but noisy measurements of that vector. Section 3.2 extends the results of Section 3.1 to the case in which some measurements are more noisy than others; that is, we have less confidence in some measurements than in others. Sections 3.1 and 3.2 use matrices and vectors whose dimensions grow larger as more measurements are obtained. This makes the problem cumbersome if many measurements are available. This leads us to Section 3.3, which presents a recursive way of estimating a constant on the basis of noisy measurements. Recursive estimation in this chapter is a method of estimating a constant without increasing the computa-

[1]Gauss published his book in 1809, although he claimed to have worked out his theory as early as 1795 (when he was 18 years old).

3.2 WEIGHTED LEAST SQUARES ESTIMATION

In the previous section, we assumed that we had an equal amount of confidence in all of our measurements. Now suppose we have more confidence in some measurements than others. In this case, we need to generalize the results of the previous section to obtain weighted least squares estimation. For example, suppose we have several measurements of the resistance of an unmarked resistor. Some of the measurements were taken with an expensive multimeter with low noise, but other measurements were taken with a cheap multimeter by a tired student late at night. We have more confidence in the first set of measurements, so we should somehow place more emphasis on those measurements than on the others. However, even though the second set of measurements is less reliable, it seems that we could get at least *some* information from them. This section shows that we can indeed get some information from less reliable measurements. We should never throw away measurements, no matter how unreliable they may be.

To put the problem in mathematical terms, suppose x is a constant but unknown n-element vector, and y is a k-element noisy measurement vector. We assume that each element of y is a linear combination of the elements of x, with the addition of some measurement noise, and the variance of the measurement noise may be different for each element of y:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} H_{11} & \cdots & H_{1n} \\ \vdots & \ddots & \vdots \\ H_{k1} & \cdots & H_{kn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} v_1 \\ \vdots \\ v_k \end{bmatrix}$$
$$E(v_i^2) = \sigma_i^2 \quad (i = 1, \ldots, k) \tag{3.11}$$

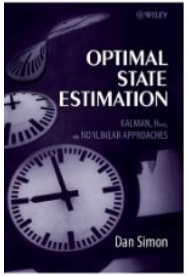
We assume that the noise for each measurement is zero-mean and independent. The measurement covariance matrix is

$$R = E(vv^T) = \begin{bmatrix} \sigma_1^2 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & \sigma_k^2 \end{bmatrix} \tag{3.12}$$

Now we will minimize the following quantity with respect to $\hat{x}$.

$$J = \epsilon_{y1}^2/\sigma_1^2 + \cdots + \epsilon_{yk}^2/\sigma_k^2 \tag{3.13}$$

Note that instead of minimizing the sum of squares of the ϵ_y elements as we did in Equation (3.4), we will minimize the *weighted* sum of squares. If y_1 is a relatively noisy measurement, for example, then we do not care as much about minimizing the difference between y_1 and the first element of $H\hat{x}$ because we do not have much



Good pathway to estimation and controls foundation

Least squares estimation

The most probable value of the unknown quantities will be that in which the sum of the squares of the differences between the actually observed and the computed values multiplied by numbers that measure the degree of precision is a minimum.

—Karl Friedrich Gauss [Gau04]

In this chapter, we will discuss least squares estimation, which is the basic idea of Karl Gauss's quote above.[1] The material in this chapter relies on the theory of the previous two chapters, and will enable us to derive optimal state estimators later in this book.

Section 3.1 discusses the estimation of a constant vector on the basis of several linear but noisy measurements of that vector. Section 3.2 extends the results of Section 3.1 to the case in which some measurements are more noisy than others; that is, we have less confidence in some measurements than in others. Sections 3.1 and 3.2 use matrices and vectors whose dimensions grow larger as more measurements are obtained. This makes the problem cumbersome if many measurements are available. This leads us to Section 3.3, which presents a recursive way of estimating a constant on the basis of noisy measurements. Recursive estimation in this chapter is a method of estimating a constant without increasing the computa-

[1] Gauss published his book in 1809, although he claimed to have worked out his theory as early as 1795 (when he was 18 years old).

Optimal State Estimation, First Edition. By Dan J. Simon
ISBN 0471708585

Equation (3.15) gives us a way to compute the optimal estimate of a constant, but there is a problem. Note that the H matrix in (3.15) is a $k \times n$ matrix. If we obtain measurements sequentially and want to update our estimate of x with each new measurement, we need to augment the H matrix and completely recompute the estimate $\hat{x}$. If the number of measurements becomes large, then the computational effort could become prohibitive. For example, suppose we obtain a measurement of a satellite's altitude once per second. After one hour has passed, the number of measurements is 3600 and growing. The computational effort of least squares estimation can rapidly outgrow our resources.

In this section, we show how to *recursively* compute the weighted least squares estimate of a constant. That is, suppose we have $\hat{x}$ after $(k-1)$ measurements, and we obtain a new measurement y_k. How can we update our estimate without completely reworking Equation (3.15)?

A linear recursive estimator can be written in the form

$$\begin{aligned} y_k &= H_k x + v_k \\ \hat{x}_k &= \hat{x}_{k-1} + K_k(y_k - H_k\hat{x}_{k-1}) \end{aligned} \tag{3.20}$$

That is, we compute $\hat{x}_k$ on the basis of the previous estimate $\hat{x}_{k-1}$ and the new measurement y_k. K_k is a matrix to be determined called the estimator gain matrix. The quantity $(y_k - H_k\hat{x}_{k-1})$ is called the correction term. Note that if the correction term is zero, or if the gain matrix is zero, then the estimate does not change from time step $(k-1)$ to k.

Before we compute the optimal gain matrix K_k, let us think about the mean of the estimation error of the linear recursive estimator. The estimation error mean can be computed as

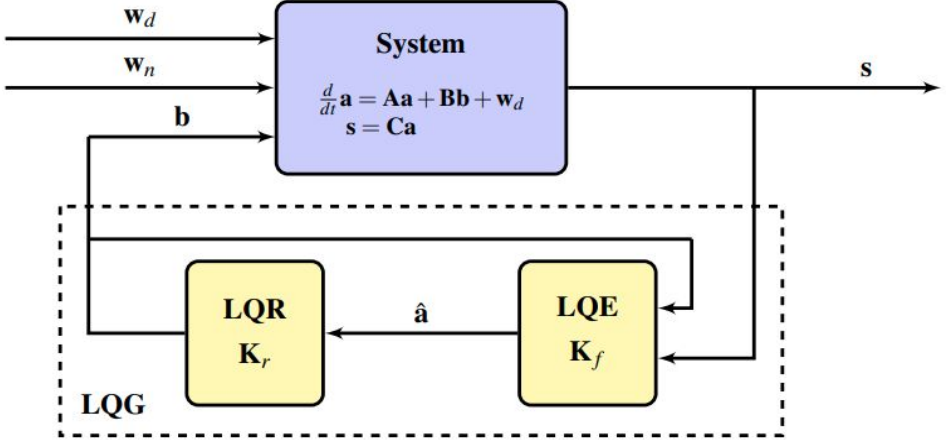
$$\begin{aligned} E(\epsilon_{x,k}) &= E(x - \hat{x}_k) \\ &= E[x - \hat{x}_{k-1} - K_k(y_k - H_k\hat{x}_{k-1})] \\ &= E[\epsilon_{x,k-1} - K_k(H_k x + v_k - H_k\hat{x}_{k-1})] \\ &= E[\epsilon_{x,k-1} - K_k H_k(x - \hat{x}_{k-1}) - K_k v_k] \\ &= (I - K_k H_k)E(\epsilon_{x,k-1}) - K_k E(v_k) \end{aligned} \tag{3.21}$$

So if $E(v_k) = 0$ and $E(\epsilon_{x,k-1}) = 0$, then $E(\epsilon_{x,k}) = 0$. In other words, if the measurement noise v_k is zero-mean for all k, and the initial estimate of x is set equal to the expected value of x [i.e., $\hat{x}_0 = E(x)$], then the expected value of $\hat{x}_k$ will be equal to x_k for all k. Because of this, the estimator of Equation (3.20) is called an unbiased estimator. Note that this property holds regardless of the value of the gain matrix K_k. This is a desirable property of an estimator because it says that, *on average*, the estimate $\hat{x}$ will be equal to the true value x.

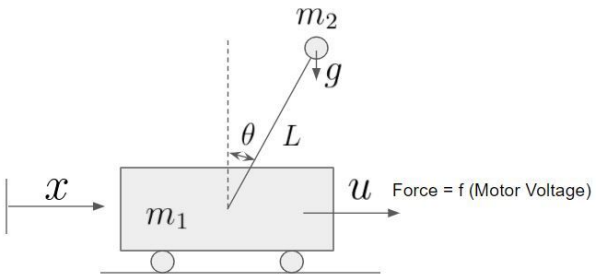
Next we turn our attention to the determination of the optimal value of K_k. Since the estimator is unbiased regardless of what value of K_k we use, we must

Linear Quadratic Gaussian (LQG)

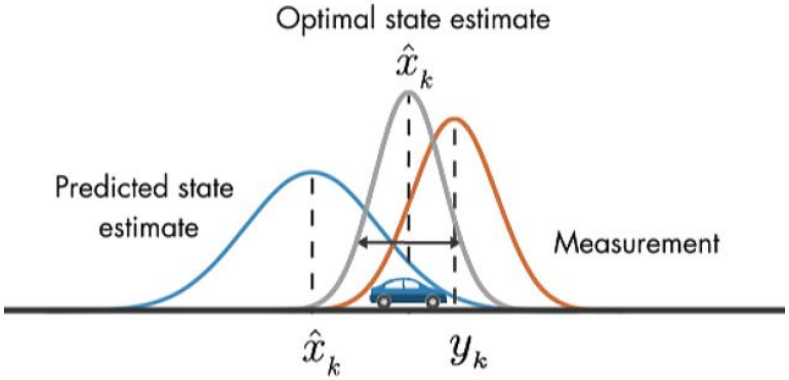
Linear model,
quadratic cost,
Gaussian noise:
LQG = LQR + LQE

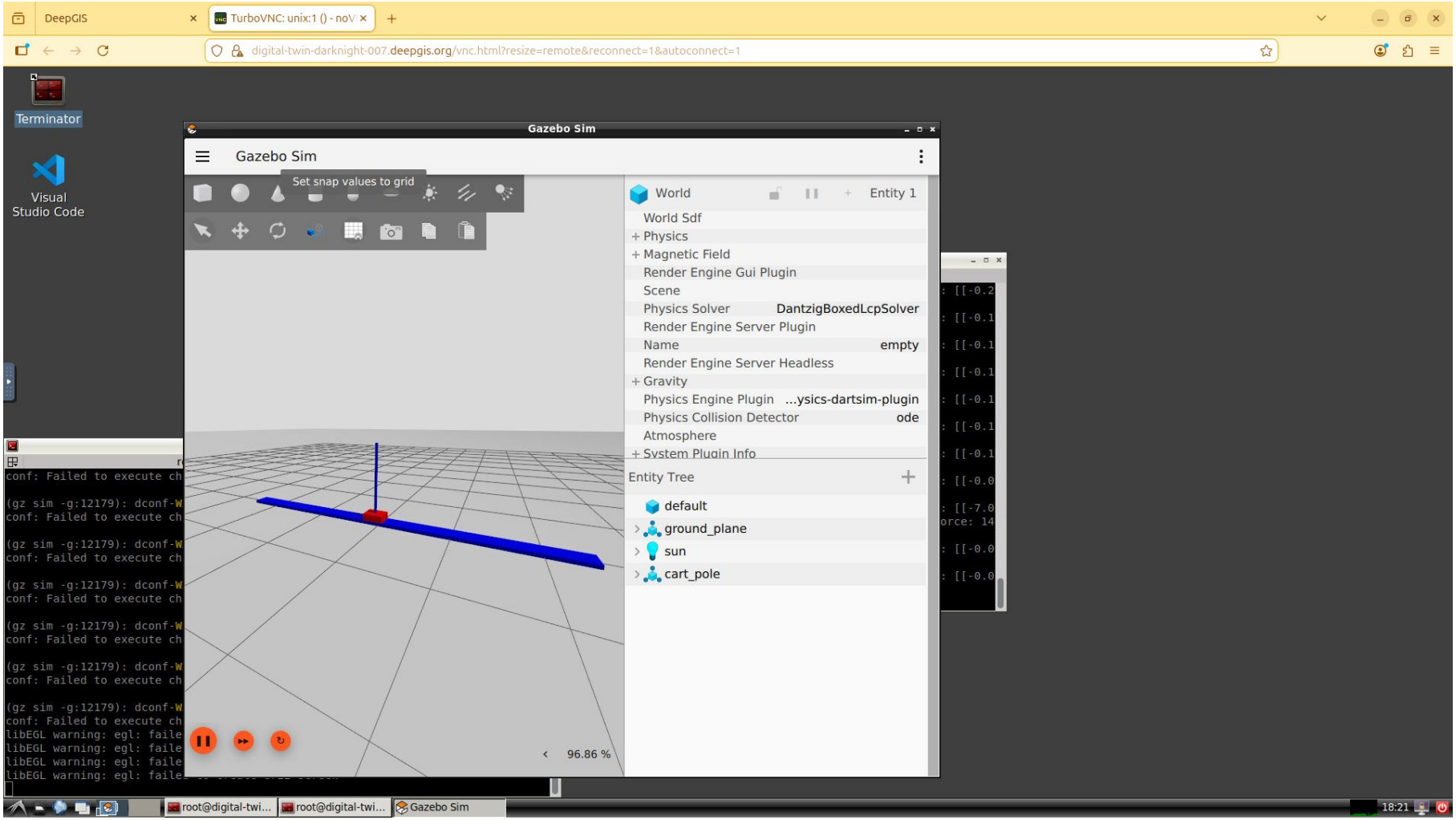


controller ⟺ state-estimator
can be designed separately



$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} \rightarrow \frac{d}{dt}\mathbf{x} = f(\mathbf{x}) \xrightarrow{\frac{Df}{Dx}|\dot{x}} \dot{\mathbf{x}} = A\mathbf{x} + Bu$$




DeepGIS
TurboVNC: unix:1 () - noV
digital-twin-darknight-007.deepgis.org/vnc.html?resize=remote&reconnect=1&autoconnect=1
Terminator
Visual Studio Code
Gazebo Sim
Gazebo Sim
Set snap values to grid
World
Entity 1
World Sdf
Physics
Magnetic Field
Render Engine Gui Plugin
Scene
Physics Solver DantzigBoxedLcpSolver
Render Engine Server Plugin
Name empty
Render Engine Server Headless
Gravity
Physics Engine Plugin ...ysics-dartsim-plugin
Physics Collision Detector ode
Atmosphere
System Plugin Info
Entity Tree
default
ground_plane
sun
cart_pole
96.86 %
conf: Failed to execute ch
(gz sim -g:12179): dconf-W
libEGL warning: egl: faile
root@digital-twi...
root@digital-twi...
Gazebo Sim
18:21

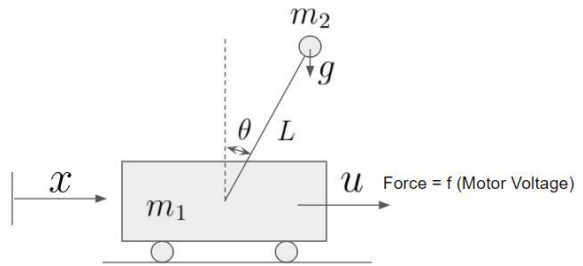
Types of control

Open-loop control
- No sensing

Feedback control (closed-loop)
- Sense error, determine control response.

Feedforward control (closed-loop)
- Sense disturbance, predict resulting error, respond to predicted error before it happens.

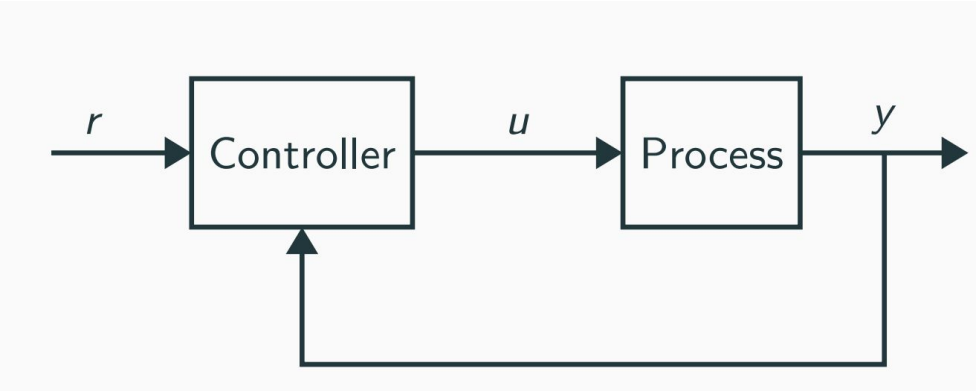
Model-predictive control (closed-loop)
- Plan trajectory to reach goal.
- Take first step.
- Repeat



$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} \rightarrow \frac{d}{dt}\mathbf{x} = f(\mathbf{x}) \xrightarrow{\frac{Df}{Dx}|\dot{x}} \dot{\mathbf{x}} = A\mathbf{x} + Bu$$

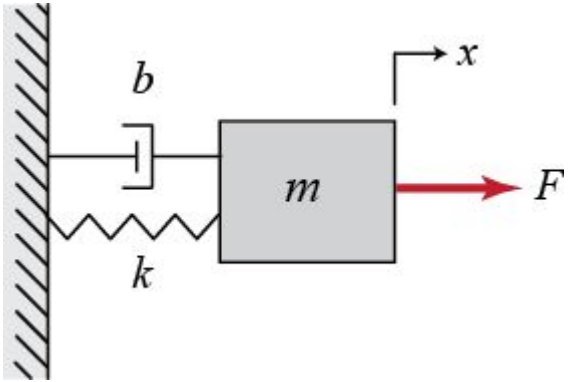
Closed-loop feedback control

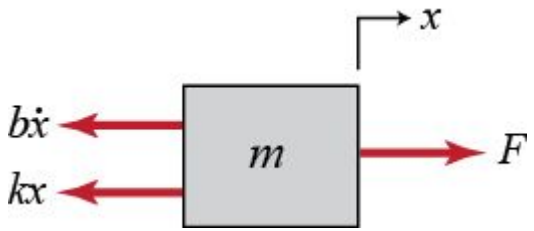
Reference value r
Control signal u
Measured signal/output y



The problem/purpose: Design a controller such that the output follows the reference signal as good as possible

Second order dynamical systems – spring damper mass





$$\Sigma F_x = F(t) - b\dot{x} - kx = m\ddot{x}$$

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F(t)$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

Cart pole problem

Inertia of Pole: $\frac{ml^2}{12}$

Pole Kinetic Energy: $\frac{m(13\dot{\theta}^2 l^2 + 24 * cos(\theta)\dot{\theta}\dot{x}l + 12\dot{x}^2) + 12\dot{x}^2}{24}$

Lagrangian: $\frac{(M+m)\dot{x}}{2} + \frac{13m\dot{\theta}^2 l^2}{24} - mglcos(\theta) + ml\dot{x}\dot{\theta}$

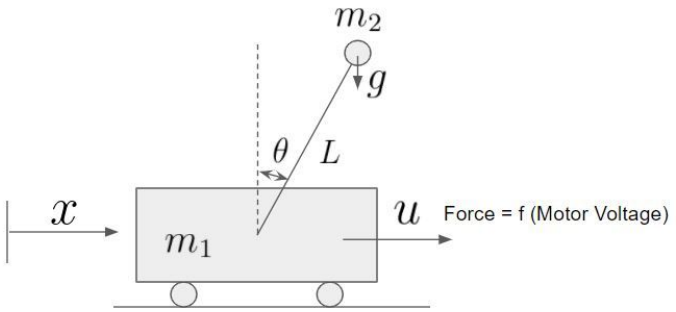
The nonlinear dynamics can then be represented as:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \frac{13ml\,sin(\theta)\dot{\theta}^2 + 13F - 12mg\,cos(\theta)sin(\theta)}{13M + 13m - 12m\,cos(\theta)^2} \\ \dot{\theta} \\ \frac{-12(ml\,cos(\theta))sin(\theta)\dot{\theta}^2 + Fcos(\theta) + mgsin(\theta)) + Mgsin(\theta)}{l*(13M + 13m - 12m\,cos(\theta)^2} \end{bmatrix}$$

The linearized dynamics are represented by:

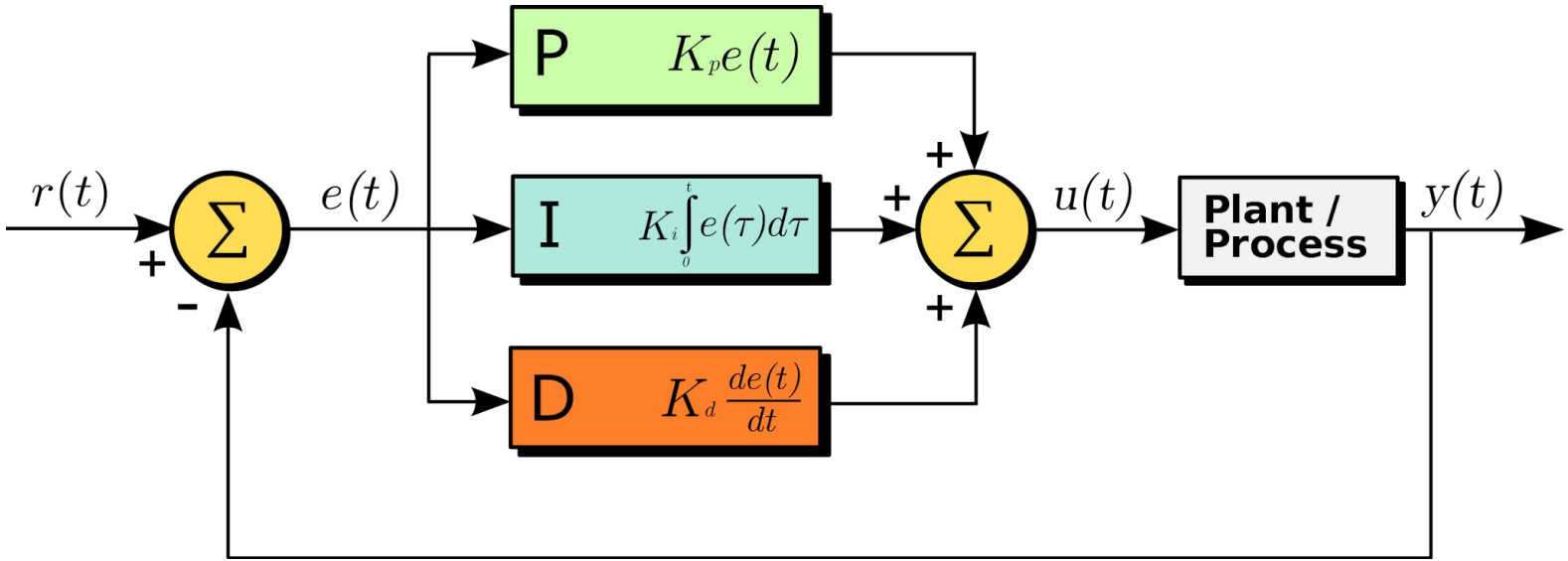
$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-12mg}{13M+m} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{12(mg+Mg)}{l(13M+m)} & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ \frac{13}{13M+m} \\ 0 \\ \frac{-12}{l(13M+m)} \end{bmatrix}$$

Linearized with $x = 0, \dot{x} = 0, \theta = 0, \dot{\theta} = 0$



$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} \rightarrow \frac{d}{dt}\mathbf{x} = f(\mathbf{x}) \xrightarrow{\frac{Df}{Dx}|\dot{x}} \dot{\mathbf{x}} = A\mathbf{x} + Bu$$

PID control

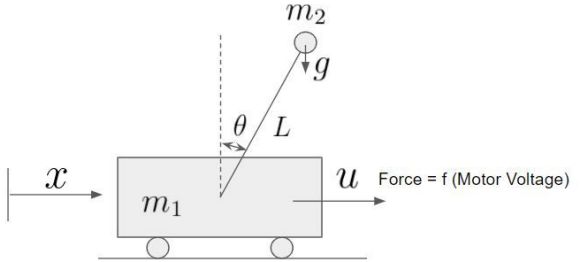

P
$K_p e(t)$
I
$K_i \int_0^t e(\tau)d\tau$
D
$K_d \frac{de(t)}{dt}$
$r(t)$
$e(t)$
$u(t)$
$y(t)$
Plant / Process
+
-
+
+
+

Linear quadratic regulator (LQR)

quadratic cost function

$$J(U) = \sum_{\tau=0}^{N-1} \left(x_\tau^T Q x_\tau + u_\tau^T R u_\tau \right) + x_N^T Q_f x_N$$

$$Q = Q^T \geq 0, \qquad Q_f = Q_f^T \geq 0, \qquad R = R^T > 0$$

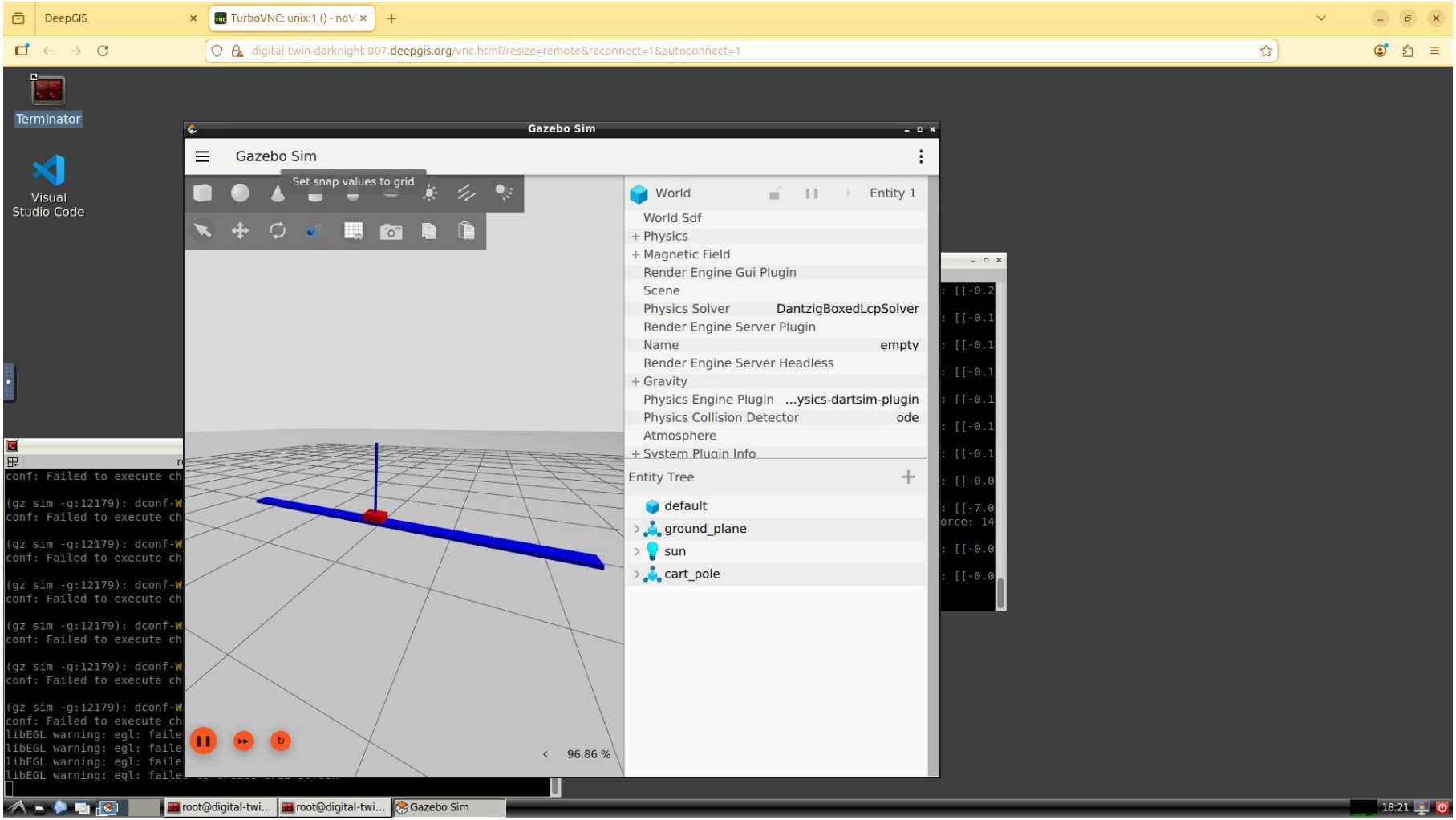


1. N is called time horizon
2. First term measures state deviation second term measures input size or actuator authority
3. Last term measures final state deviation
4. Q, R set relative weights of state deviation and input usage
5. R > 0 means any (nonzero) input adds to cost J

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} \rightarrow \frac{d}{dt}\mathbf{x} = f(\mathbf{x}) \xrightarrow{\frac{Df}{Dx}|\dot{x}} \dot{\mathbf{x}} = A\mathbf{x} + Bu$$

LQR problem: find $u_0^{\text{lqr}}, \ldots, u_{N-1}^{\text{lqr}}$ that minimizes $J(U)$

$$\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t)$$


DeepGIS
TurboVNC: unix:1 () - noV
digital-twin-darknight-007.deepgis.org/vnc.html?resize=remote&reconnect=1&autoconnect=1
Terminator
Visual Studio Code
Gazebo Sim
Gazebo Sim
Set snap values to grid
World
Entity 1
World Sdf
Physics
Magnetic Field
Render Engine Gui Plugin
Scene
Physics Solver DantzigBoxedLcpSolver
Render Engine Server Plugin
Name empty
Render Engine Server Headless
Gravity
Physics Engine Plugin ...ysics-dartsim-plugin
Physics Collision Detector ode
Atmosphere
System Plugin Info
Entity Tree
default
ground_plane
sun
cart_pole
96.86 %
conf: Failed to execute ch
(gz sim -g:12179): dconf-W
libEGL warning: egl: faile
root@digital-twi...
root@digital-twi...
Gazebo Sim
18:21

LQR control

For continuous-time systems, `lqr` computes the state-feedback control $u = -Kx$ that minimizes the quadratic cost function

$$J(u) = \int_0^\infty (x^T Q x + u^T R u + 2x^T N u) dt$$

subject to the system dynamics $\dot{x} = Ax + Bu$.

In addition to the state-feedback gain K, `lqr` returns the solution S of the associated algebraic Riccati equation

$$A^T S + SA - (SB + N)R^{-1}(B^T S + N^T) + Q = 0$$

and the closed-loop poles $P = eig(A - BK)$. The gain matrix K is derived from S using

$$K = R^{-1}(B^T S + N^T).$$

For discrete-time systems, `lqr` computes the state-feedback control $u_n = -Kx_n$ that minimizes

$$J = \sum_{n=0}^{\infty} \{x^T Q x + u^T R u + 2x^T N u\}$$

subject to the system dynamics $x_{n+1} = Ax_n + Bu_n$.

What's a good cost function?
i.e., Q and R matrices

How much do you care about (accumulated) control expenditure, and errors in state?

https://www.mathworks.com/help/control/ref/lti.lqr.html

LQR control

For continuous-time systems, `lqr` computes the state-feedback control $u = -Kx$ that minimizes the quadratic cost function

$$J(u) = \int_0^\infty (x^T Q x + u^T R u + 2x^T N u) dt$$

subject to the system dynamics $\dot{x} = Ax + Bu$.

In addition to the state-feedback gain K, `lqr` returns the solution S of the associated algebraic Riccati equation

$$A^T S + SA - (SB + N)R^{-1}(B^T S + N^T) + Q = 0$$

and the closed-loop poles $P = eig(A - BK)$. The gain matrix K is derived from S using

$$K = R^{-1}(B^T S + N^T).$$

For discrete-time systems, `lqr` computes the state-feedback control $u_n = -Kx_n$ that minimizes

$$J = \sum_{n=0}^{\infty} \{x^T Q x + u^T R u + 2x^T N u\}$$

subject to the system dynamics $x_{n+1} = Ax_n + Bu_n$.

What's a good cost function?
i.e., Q and R matrices

How much do you care about (accumulated) control expenditure, and errors in state?

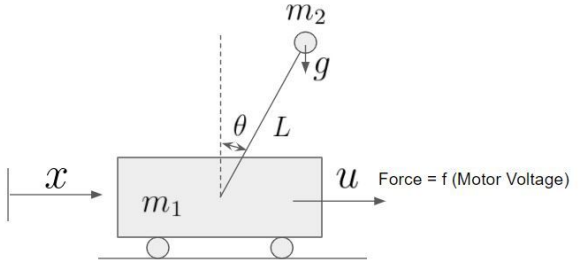
https://www.mathworks.com/help/control/ref/lti.lqr.html

Linear quadratic regulator (LQR)

quadratic cost function

$$J(U) = \sum_{\tau=0}^{N-1} \left(x_\tau^T Q x_\tau + u_\tau^T R u_\tau \right) + x_N^T Q_f x_N$$

$$Q = Q^T \geq 0, \qquad Q_f = Q_f^T \geq 0, \qquad R = R^T > 0$$



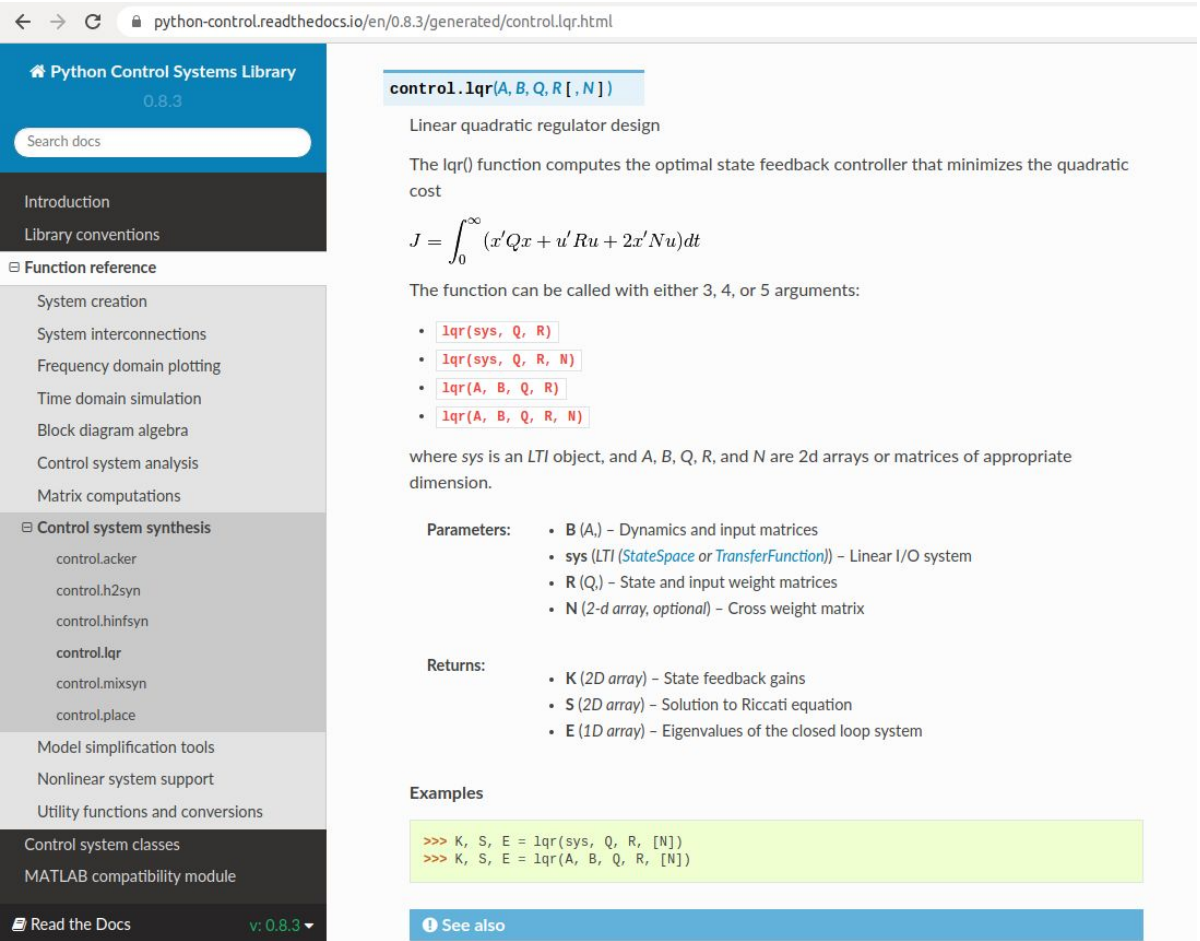
1. N is called time horizon
2. First term measures state deviation second term measures input size or actuator authority
3. Last term measures final state deviation
4. Q, R set relative weights of state deviation and input usage
5. R > 0 means any (nonzero) input adds to cost J

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} \rightarrow \frac{d}{dt}\mathbf{x} = f(\mathbf{x}) \xrightarrow{\frac{Df}{Dx}|\dot{x}} \dot{\mathbf{x}} = A\mathbf{x} + Bu$$

LQR problem: find $u_0^{\text{lqr}}, \ldots, u_{N-1}^{\text{lqr}}$ that minimizes $J(U)$

$$\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t)$$

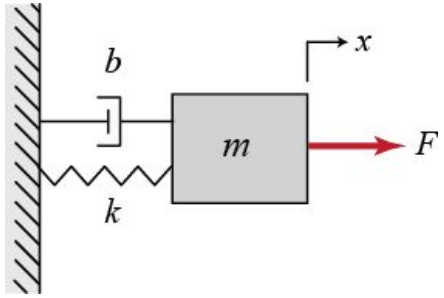
LQR control

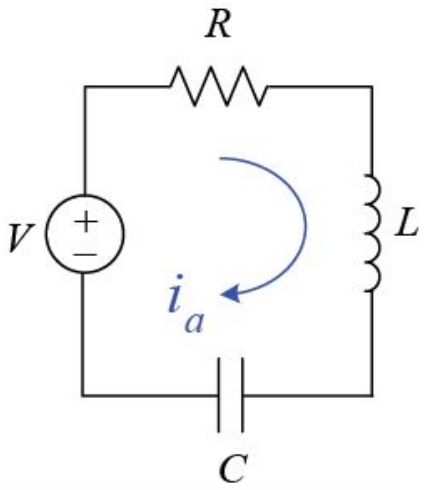


What's a good cost function?
i.e., Q and R matrices

How much do you care about control expenditure, and errors in state?

Are these systems stable?





Gilbert Strang

6.3 Systems of Differential Equations

1 If $A\boldsymbol{x} = \lambda\boldsymbol{x}$ then $\boldsymbol{u}(t) = e^{\lambda t}\boldsymbol{x}$ will solve $\dfrac{d\boldsymbol{u}}{dt} = A\boldsymbol{u}$. Each λ and $\boldsymbol{x}$ give a solution $e^{\lambda t}\boldsymbol{x}$.

2 If $A = X\Lambda X^{-1}$ then $\boxed{\boldsymbol{u}(t) = e^{At}\boldsymbol{u}(0) = Xe^{\Lambda t}X^{-1}\boldsymbol{u}(0) = c_1e^{\lambda_1 t}\boldsymbol{x}_1 + \cdots + c_ne^{\lambda_n t}\boldsymbol{x}_n.}$

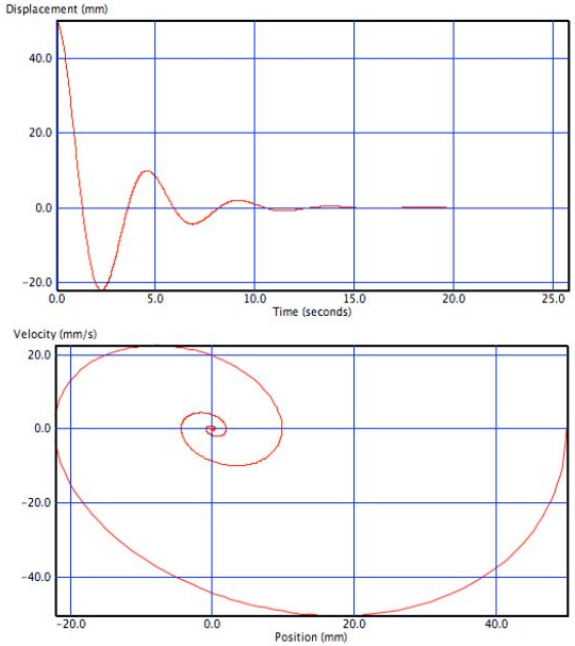
3 A is **stable** and $\boldsymbol{u}(t) \to \boldsymbol{0}$ and $e^{At} \to 0$ when all eigenvalues of A have real part < 0.

4 **Matrix exponential** $e^{At} = I + At + \cdots + (At)^n/n! + \cdots = Xe^{\Lambda t}X^{-1}$ if A is diagonalizable.

5 **Second order equation** $u'' + Bu' + Cu = 0$ is equivalent to **First order system** $\begin{bmatrix} u \\ u' \end{bmatrix}' = \begin{bmatrix} 0 & 1 \\ -C & -B \end{bmatrix}\begin{bmatrix} u \\ u' \end{bmatrix}$.

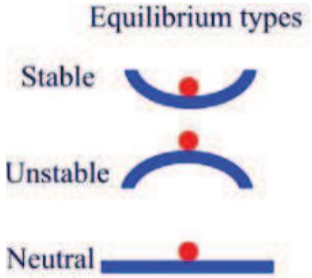
Two views of dynamic behavior

- Time plot (t,x)
- Phase portrait (x,v)



Roughly Speaking: A system of differential equations is Stable if

- small inputs produce small outputs (**Bounded-Input Bounded-Output**)
- Disturbances tend to decay (**Asymptotic Stability**)



For aircraft, we will also define **Static Stability** and **Dynamic Stability**. However, the terms *Static* and *Dynamic* refer to which equations of motion we use, and not properties of the motion itself.

Eigendecomposition of linear dynamical systems

$$my'' + by' + ky = 0$$

$$dy/dt = y'$$
$$dy'/dt = -ky - by'$$

converts to

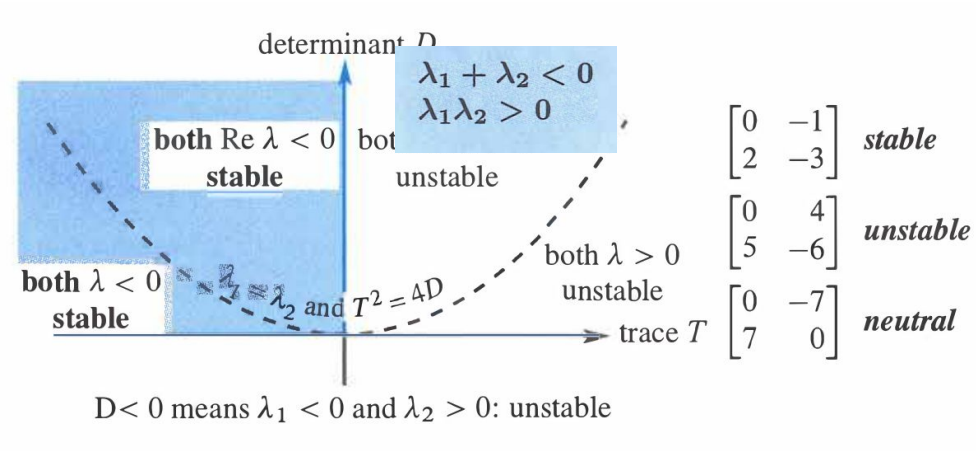
$$\frac{d}{dt}\begin{bmatrix} y \\ y' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k & -b \end{bmatrix}\begin{bmatrix} y \\ y' \end{bmatrix} = Au$$

Elastic force

Stability through determinant and trace

Find roots of this equation

$$A - \lambda I = \begin{bmatrix} -\lambda & 1 \\ -k & -b-\lambda \end{bmatrix} \quad \text{has determinant} \quad \lambda^2 + b\lambda + k = 0.$$

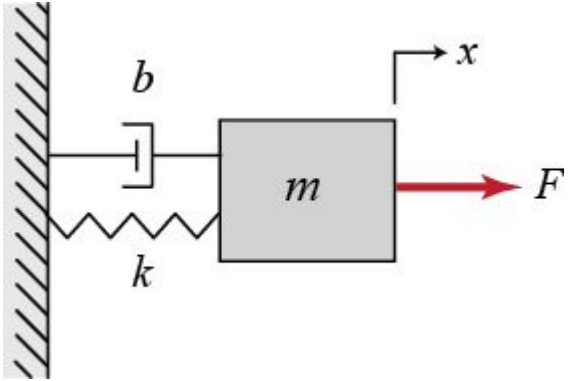


Observability and controllability

In many dynamical systems, control does not affect the complete state of the dynamical system but only a part of it.

In real processes we often can observe only a part of the complete state of the dynamical system.

Important to determine whether or not observation, and control of the complete state of the dynamical system is possible.



- Rocket launching, satellite control, control of aircraft
- Biological systems, e.g., sugar level in blood
- Defence, anti-missiles problems
- Economy, regulating inflation rate
- Ecology, predator-prey system

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} F(t)$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

Controllability

If controllability matrix is full rank, then, it is possible to steer dynamical system from an arbitrary initial state to an arbitrary final state using the set of admissible controls.

$$Q_c = [B, AB, A^2B, \ldots, A^{n-1}B]$$

controllability matrix

THEOREM 1. *Dynamical system* (1) *is controllable if and only if*

$$rank\ W = n.$$

Equivalent: $Q_c Q_c^\top$ is invertible

controllability Grammian

Controllability and Observability summary

$$Q_c = [B, AB, A^2B, \ldots, A^{n-1}B]$$

controllability matrix

If controllability matrix is full rank, then, it is possible to steer dynamical system from an arbitrary initial state to an arbitrary final state using the set of admissible controls.

$$Q_o = \begin{pmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{pmatrix}$$

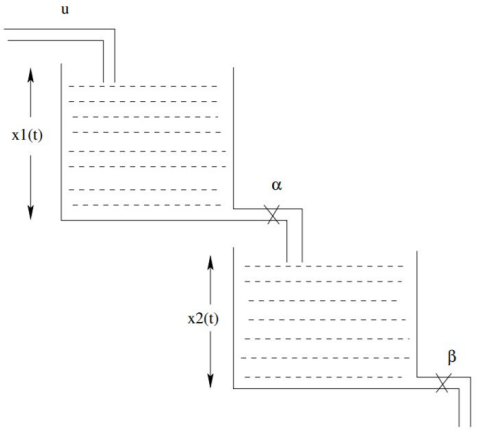
observability matrix

If observability matrix is full rank, system is observable, current values of all of its state variables can be determined through output sensors.

$$\frac{dx_1}{dt} = -\alpha x_1 + u$$
$$\frac{dx_2}{dt} = \alpha x_1 - \beta x_2$$

$$\frac{d}{dt}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -\alpha & 0 \\ \alpha & -\beta \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u$$

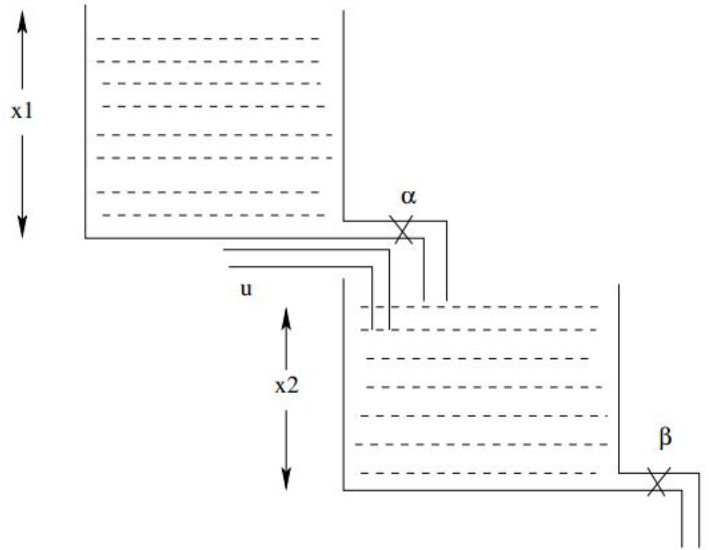
Example: Tank Problem :



Let $x_1(t)$ be the water level in Tank 1 and $x_2(t)$ be the water level in Tank 2. Let α be the rate of outflow from Tank 1 and β be rate of outflow from Tank 2. Let u be the supply of water to the system. The system can be modelled into the following differential equations:

$$\frac{dx_1}{dt} = -\alpha x_1$$
$$\frac{dx_2}{dt} = \alpha x_1 - \beta x_2 + u$$

$$\frac{d}{dt}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} -\alpha & 0 \\ \alpha & -\beta \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u$$



Can both tanks be controlled for each model?
Are tank water levels fully controllable for both models?

Matrix exponentials and LTI, controllability

$$\frac{d}{dt}x = Ax + Bu \qquad x(0) = x_0$$

$$x(t) = e^{At}x_0 + \int_0^t e^{A(t-s)}Bu(s)\,ds$$

$$e^A = I + A + \frac{1}{2}A^2 + \ \ldots\ + \frac{1}{n!}A^n + \ \ldots$$

Theorem: "A system is controllable if and only if the matrix

$$\overline{C} = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix}$$ Controllability Matrix

is full-rank."

Equivalent:

$Q_c Q_c^\top$ is invertible

Matrix exponentials and LTI systems, controllability

The time derivative of the Controllability Gramian

$$P(t) = \int_0^t e^{A(t-\tau)} B B^T e^{A^T(t-\tau)} d\tau$$

is given by (Leibniz's rule)

$$\frac{d}{dt} P(t) = e^{A(t-\tau)} B B^T e^{A^T(t-\tau)}\Big|_{\tau=t} +$$

$$+ \int_0^t \frac{d}{dt}\left[e^{A(t-\tau)}\right] B B^T e^{A^T(t-\tau)} d\tau + \int_0^t e^{A(t-\tau)} B B^T \frac{d}{dt}\left[e^{A^T(t-\tau)}\right] d\tau$$

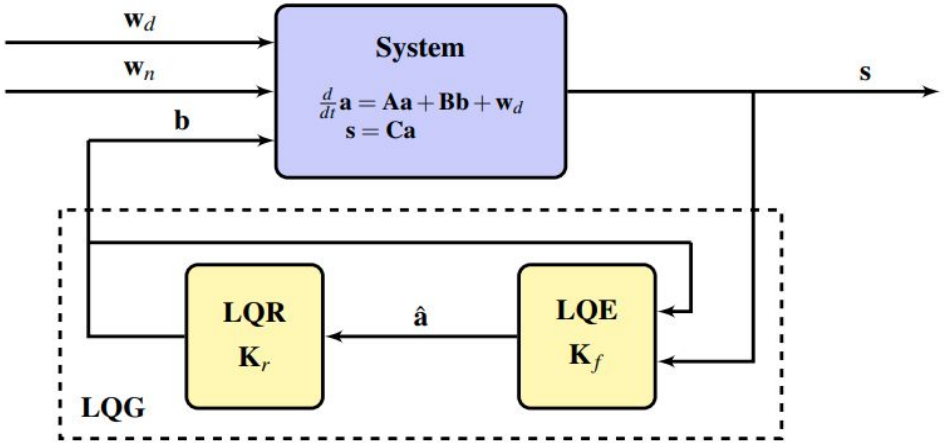
$$= B B^T + \int_0^t \frac{d}{dt}\left[e^{A(t-\tau)}\right] B B^T e^{A^T(t-\tau)} d\tau + \int_0^t e^{A(t-\tau)} B B^T \frac{d}{dt}\left[e^{A^T(t-\tau)}\right] d\tau$$

Then, the Controllability Gramian satisfies

$$\dot{P}(t) = B B^T + A P(t) + P(t) A^T, \qquad P(0) = 0$$

Lyapunov Equation

Linear Quadratic Gaussian (LQG)



controller ⟺ state-estimator

can be designed separately

world has disturbances
sensors have noise

$$\begin{aligned} \mathbf{x}(k+1) &= A\mathbf{x}(k) + B\mathbf{u}(k) + \mathrm{q} \\ \mathbf{y}(k) &= C\mathbf{x}(k) + D\mathbf{u}(k) + \mathrm{r} \end{aligned}$$

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$$

$$\mathbf{u} = -K\mathbf{x}$$

$$\dot{\mathbf{x}} = A\mathbf{x} - BK\mathbf{x}$$

Linear model, quadratic cost, Gaussian noise:
LQG = LQR + Kalman filter

$$\boxed{\dot{\mathbf{x}} = (A - BK)\mathbf{x}}$$

$$Q_o = \begin{pmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{pmatrix}$$

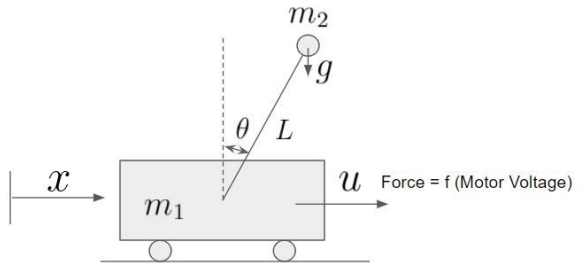
observability matrix

If observability matrix is full rank, system is observable, current values of all of its state variables can be determined through output sensors.

The linearized dynamics are represented by:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-12mg}{13M+m} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{12(mg+Mg)}{l(13M+m)} & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ \frac{13}{13M+m} \\ 0 \\ \frac{-12}{l(13M+m)} \end{bmatrix}$$

Linearized with $x = 0, \dot{x} = 0, \theta = 0, \dot{\theta} = 0$



What if
Only x is observed ?
Only \theta is observed ?

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} \rightarrow \frac{d}{dt}\mathbf{x} = f(\mathbf{x}) \xrightarrow{\frac{Df}{Dx}|\dot{x}} \dot{\mathbf{x}} = A\mathbf{x} + Bu$$

Recursive least-squares (RSL) vs ordinary least-squares (OLS)

Feature	Kalman Filter	Least Squares
Approach	Recursive estimation	Batch processing
System Type	Dynamic systems	Static systems
Data Handling	Processes data sequentially	Processes all data at once
Uncertainty Handling	Explicit noise modeling (process & measurement noise)	Assumes fixed noise structure
Computational Complexity	Lower for online estimation	Higher due to matrix inversion
Adaptive Capability	Updates with new data	No built-in adaptability

Kalman filter involves dynamical system, where system evolves over time

Kalman filter

Bayes filter

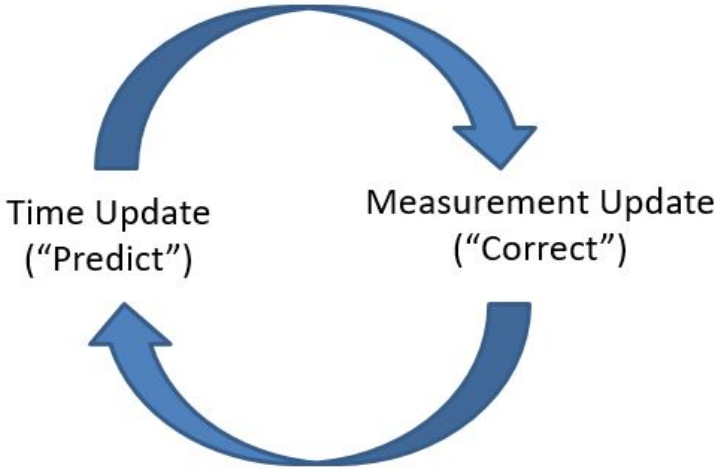
Filtering and prediction for linear systems

State at time t, mean μ_t and the covariance Σ_t

Markov assumption

Sensor updates

What is the posterior?



- **The estimator is a linear dynamical system**
- Algebraic Ricatti equation – stability of error rate, converge to 0
- Think similar gain as LQR case Kc, here Kalman gain Kf
- Can use LQR command to solve algebraic Ricatti eqn to get Kalman gain (or use LQE command!)

Kalman filter

Bayes filter

Filtering and prediction for linear systems

State at time t, mean μ_t and the covariance

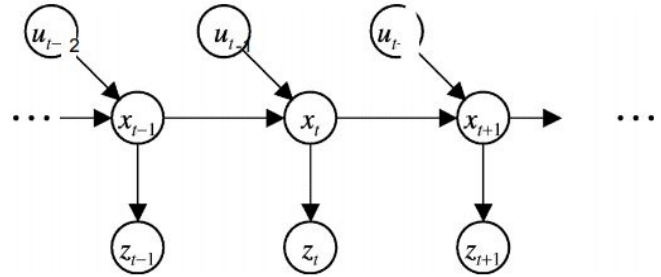
Markov assumption

Sensor updates

What is the posterior?

Graphical model

$X_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$



$$
\begin{aligned}
X_{t+1} &= A_t X_t + B_t u_t + \varepsilon_t \quad \varepsilon_t \sim \mathcal{N}(0, Q_t) \\
Z_t &= C_t X_t + d_t + \delta_t \quad \delta_t \sim \mathcal{N}(0, R_t)
\end{aligned}
$$

State-space representation

Frequency domain vs Time domain filters

Feature	Wiener Filter	Kalman Filter
Approach	Linear filtering based on stationary statistics.	Recursive estimation based on dynamic models.
System Model	Assumes a stationary signal and noise process.	Handles time-varying (non-stationary) processes.
Implementation	Typically implemented in the frequency domain (batch processing).	Works iteratively in the time domain (recursive).
Optimality	Optimal in the minimum mean square error (MMSE) sense for stationary signals.	Optimal for linear dynamic systems with Gaussian noise.
State Estimation	No explicit state-space representation.	Based on a state-space model with prediction and correction.
Computational Complexity	Lower (applies a filter once to the whole data).	Higher (recursively updates estimates over time).
Causality	Often requires the entire signal (non-causal processing).	Operates in real-time (causal).
Noise Assumptions	Requires known power spectral densities of signal and noise.	Assumes Gaussian noise with known covariance.
Best Suited For	Image and signal processing with stationary noise.	Real-time tracking and control in dynamic systems.

Fun facts – factor graphs can represent many things

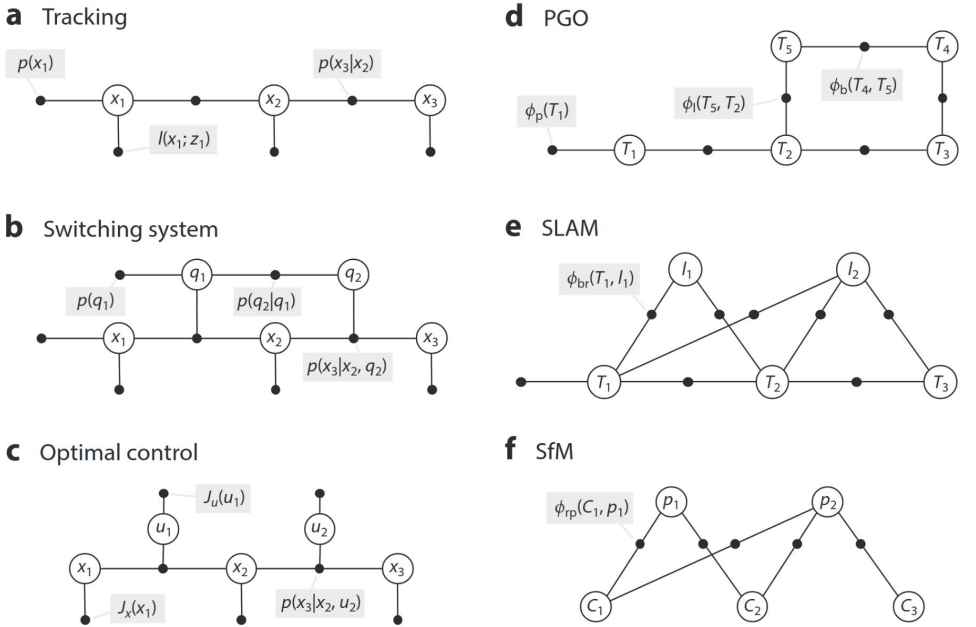


Figure 2

A tour of factor graphs. Abbreviations: PGO, pose graph optimization; SfM, structure from motion; SLAM, simultaneous localization and mapping.

Factor Graphs: Exploiting Structure in Robotics

Annual Review of Control, Robotics, and Autonomous Systems

Vol. 4:141-166 (Volume publication date May 2021)
First published as a Review in Advance on January 12, 2021
https://doi.org/10.1146/annurev-control-061520-010504

Frank Dellaert[1,2]

[1]School of Interactive Computing, Georgia Institute of Technology, Atlanta, Georgia 30332, USA; email: frank.dellaert@gatech.edu
[2]Google AI, Mountain View, California 94043, USA

Sections

Abstract

Many estimation, planning, and optimal control problems in robotics have an optimization problem at their core. In most of these optimization problems, the objective to be maximized or minimized is composed of many different factors or terms that are local in nature—that is, they depend only on a small subset of the variables. A particularly insightful way of modeling this locality structure is to use the concept of factor graphs, a bipartite graphical model in which factors represent functions on subsets of variables. Factor graphs can represent a wide variety of problems across robotics, expose opportunities to improve computational performance, and are beneficial in designing and thinking about how to model a problem, even aside from performance considerations. I discuss each of these three aspects in detail and review several state-of-the-art robotics applications in which factor graphs have been used with great success.

Keywords

simultaneous localization and mapping, SLAM, estimation, motion planning, graphical models, factor graphs

https://www.annualreviews.org/doi/pdf/10.1146/annurev-control-061520-010504

Motion models

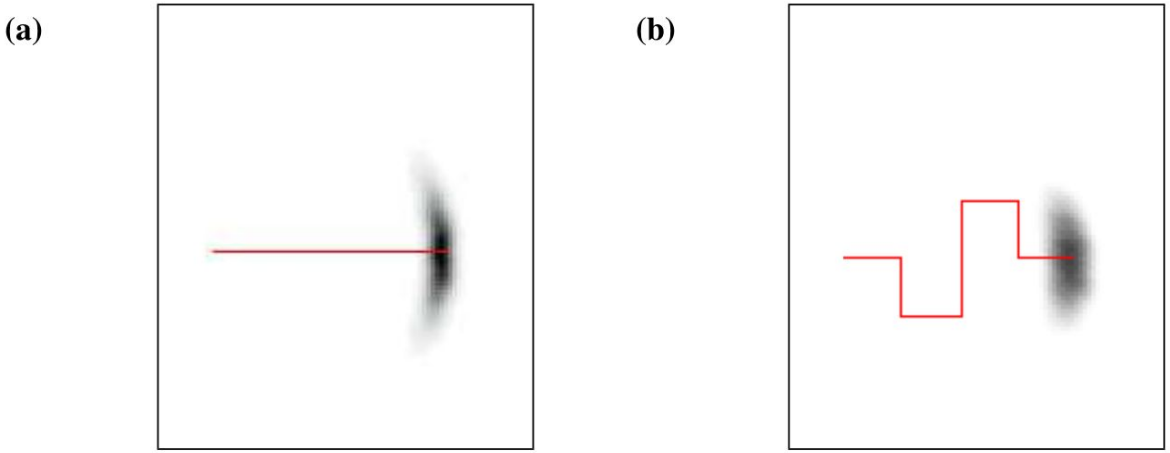


Figure 5.2 The motion model: Posterior distributions of the robot's pose upon executing the motion command illustrated by the solid line. The darker a location, the more likely it is. This plot has been projected into 2D. The original density is three-dimensional, taking the robot's heading direction θ into account.

Probabilistic Robotics
by Sebastian Thrun

Motion models

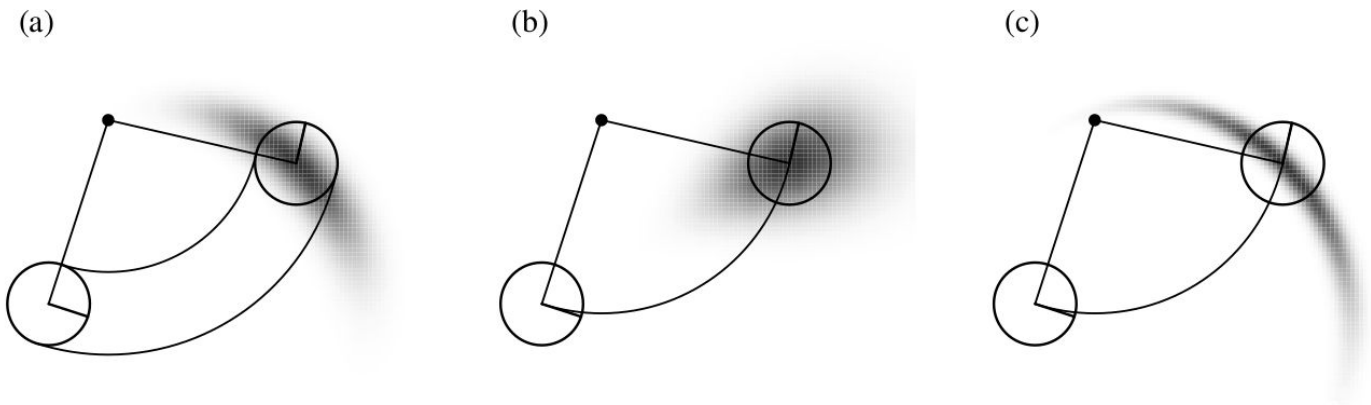


Figure 5.3 The velocity motion model, for different noise parameter settings.

Probabilistic Robotics
by Sebastian Thrun

Motion models

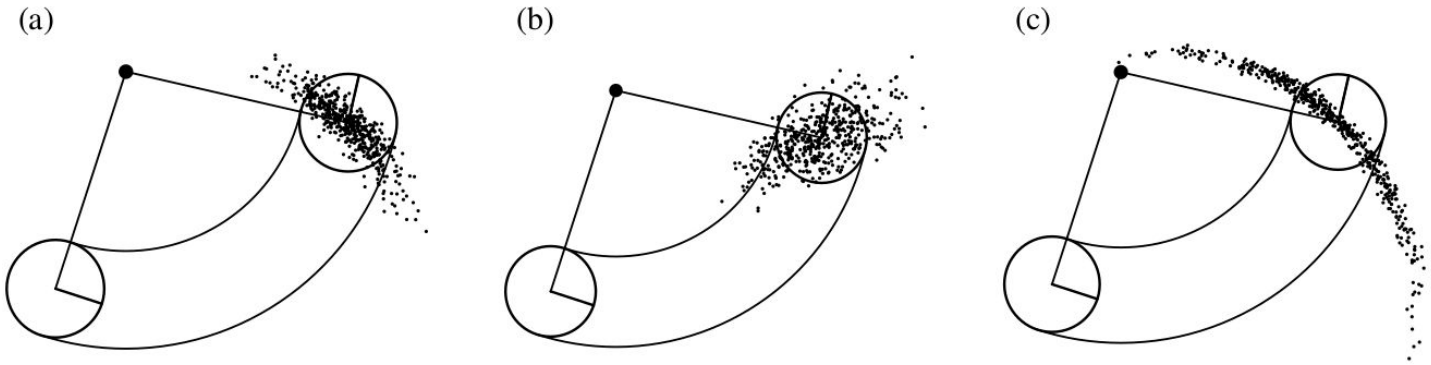


Figure 5.4 Sampling from the velocity motion model, using the same parameters as in Figure 5.3. Each diagram shows 500 samples.

Probabilistic Robotics
by Sebastian Thrun

Motion models - non sensing robot

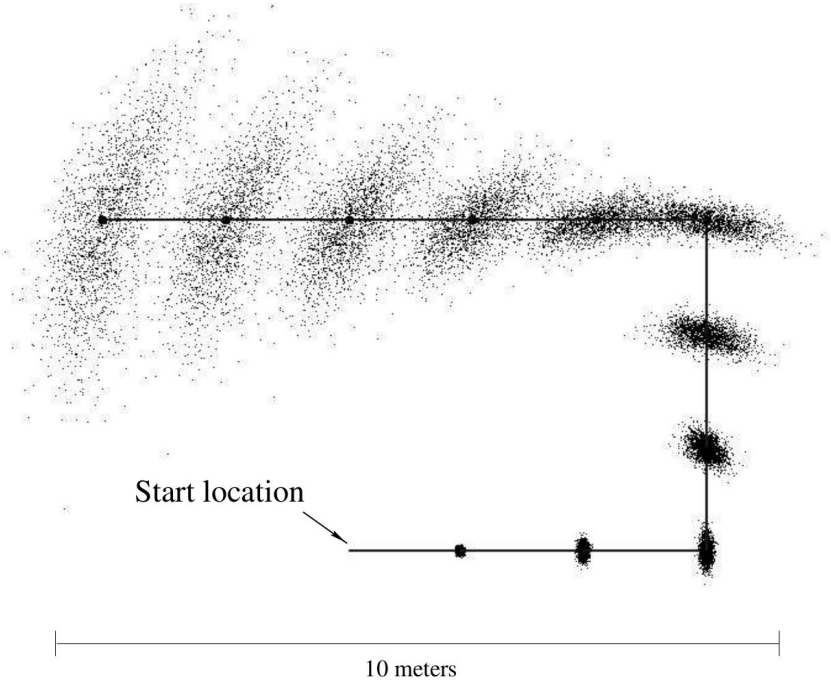


Figure 5.10 Sampling approximation of the position belief for a non-sensing robot. The solid line displays the actions, and the samples represent the robot's belief at different points in time.

Probabilistic Robotics
by Sebastian Thrun

Localization examples

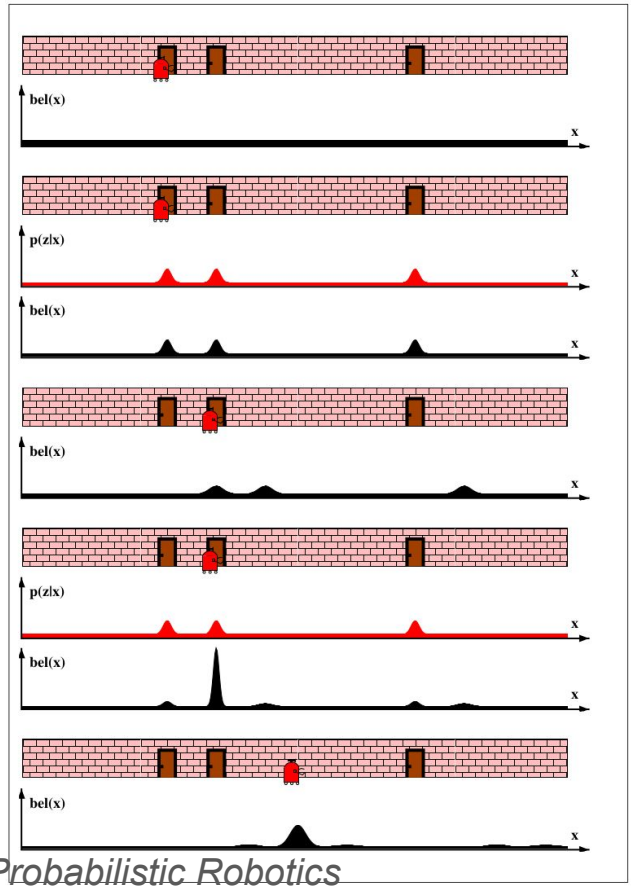


Figure 1.1 Schematic illustration of grid-based localization: A mobile robot during global localization.

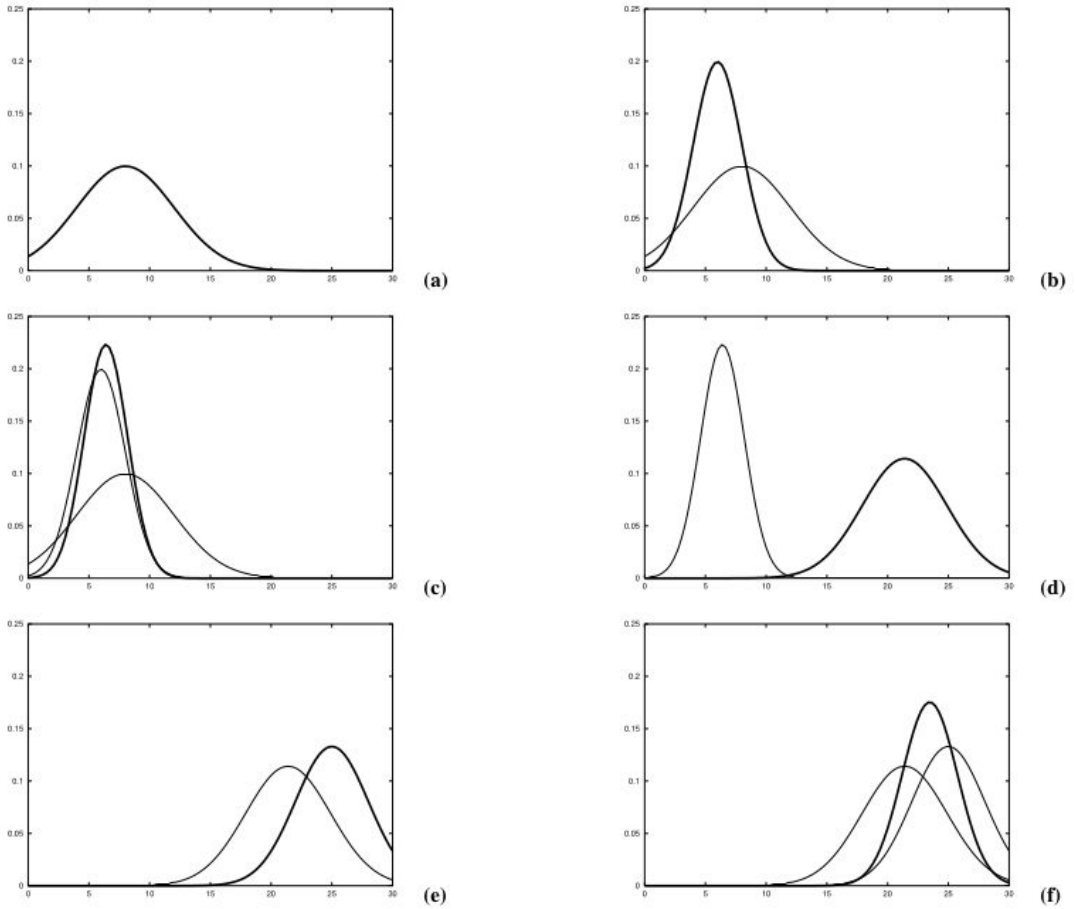


Figure 3.2 Illustration of Kalman filters: (a) initial belief, (b) a measurement (in bold) with the associated uncertainty, (c) belief after integrating the measurement into the belief using the Kalman filter algorithm, (d) belief after motion to the right (which introduces uncertainty), (e) a new measurement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

Probabilistic Robotics
by Sebastian Thrun

Kalman filter

$$x_t = \begin{pmatrix} x_{1,t} \\ x_{2,t} \\ \vdots \\ x_{n,t} \end{pmatrix} \quad \text{and} \quad u_t = \begin{pmatrix} u_{1,t} \\ u_{2,t} \\ \vdots \\ u_{m,t} \end{pmatrix}$$

Bayes filter

Filtering and prediction for linear systems

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

State at time t, mean μ_t and the covariance Σ_t

State transition probability

Markov assumption

Sensor updates

$$p(x_t \mid u_t, x_{t-1})$$

What is the posterior?

$$p(x_t \mid u_t, x_{t-1}) \tag{3.4}$$

$$= \det(2\pi R_t)^{-\frac{1}{2}} \exp\left\{-\tfrac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1}(x_t - A_t x_{t-1} - B_t u_t)\right\}$$

Sebastian Thrun Chapter 3.2

Kalman filter

Process model $$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

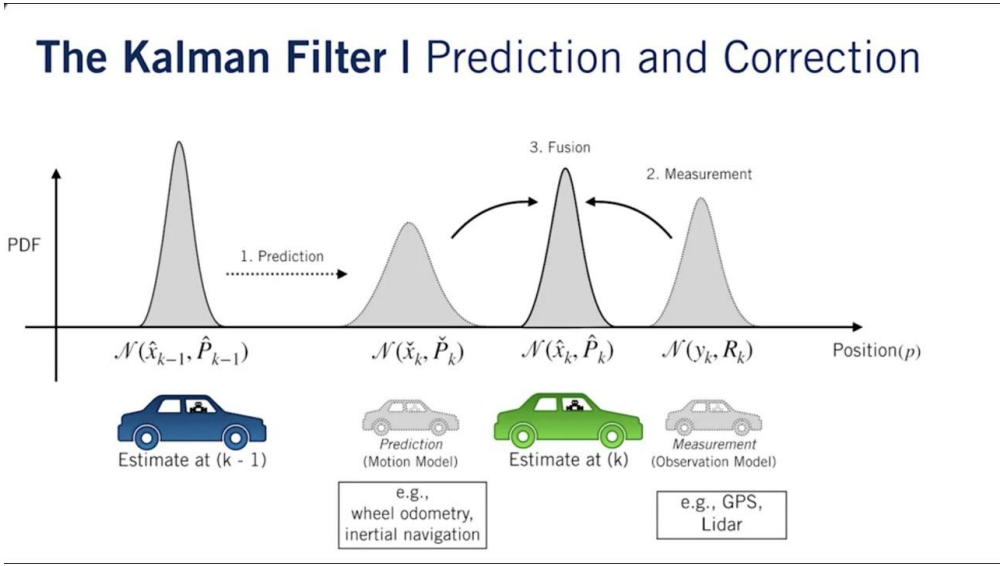
Measurements (observation model) $$z_t = C_t x_t + \delta_t$$

Initial belief $$bel(x_0) = p(x_0) = \det(2\pi\Sigma_0)^{-\frac{1}{2}} \exp\left\{-\tfrac{1}{2}(x_0 - \mu_0)^T \Sigma_0^{-1}(x_0 - \mu_0)\right\}$$

posterior $$p(z_t \mid x_t) = \det(2\pi Q_t)^{-\frac{1}{2}} \exp\left\{-\tfrac{1}{2}(z_t - C_t\, x_t)^T\, Q_t^{-1}\, (z_t - C_t\, x_t)\right\} \quad (3.6)$$

Predict-correct

Kalman gain applied on discrepancy between predicted and observed states



Algorithm Kalman_filter$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$**:**

$$\bar{\mu}_t = A_t\ \mu_{t-1} + B_t\ u_t$$
$$\bar{\Sigma}_t = A_t\ \Sigma_{t-1}\ A_t^T + R_t$$
$$K_t = \bar{\Sigma}_t\ C_t^T (C_t\ \bar{\Sigma}_t\ C_t^T + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t (z_t - C_t\ \bar{\mu}_t)$$
$$\Sigma_t = (I - K_t\ C_t)\ \bar{\Sigma}_t$$
return μ_t, Σ_t

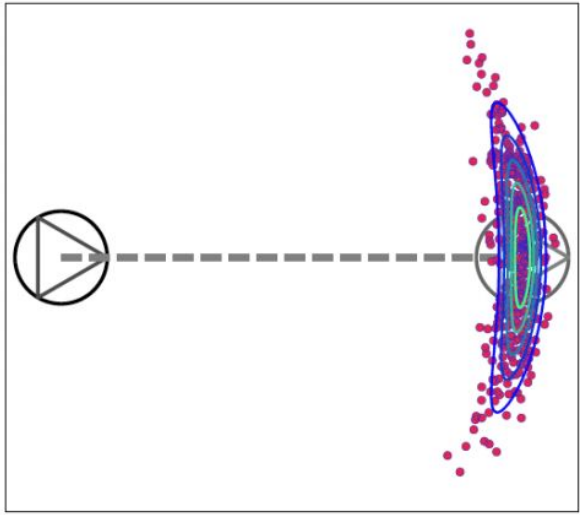
Kalman Filter - Part 1, Prof. Jonathan Kelly, Univ. of Toronto

https://www.youtube.com/watch?v=LioOvUZ1MiM

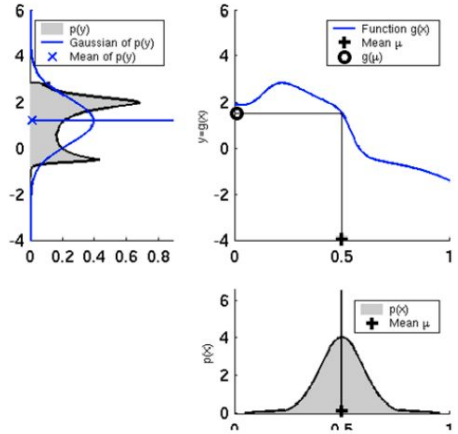
Kalman filtering options

Extended Kalman filter (EKF)

Unscented Kalman filter (UKF)



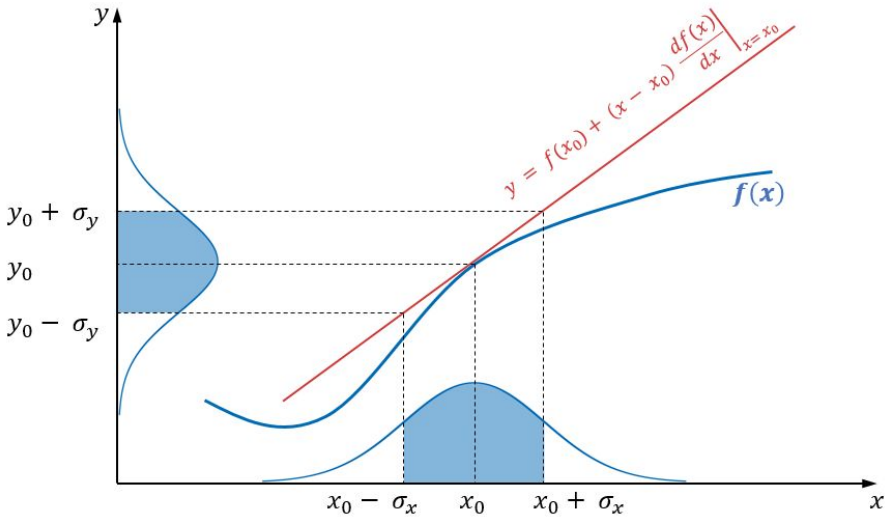
Non-linear, non-Gaussian, multi-modal world



Effect of nonlinear transformation of Gaussian random variable.

Extended Kalman Filter (EKF)

- Linearize at estimated state (Jacobian matrix)
- State propagation and observation model used directly
- Jacobians can be computed offline
- Covariance propagation and update based on linearized model
- Optimality of state estimation not guaranteed, understate estimates the covariances (too optimistic)



$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \cdots$$
$$\cdots + \frac{h^{(n-1)}}{(n-1)!}f^{(n-1)}(x) + \frac{h^n}{n!}f^n(x+\lambda h)$$

$$g'(u_t, x_{t-1}) \quad := \quad \frac{\partial g(u_t, x_{t-1})}{\partial x_{t-1}}$$

Taylor series expansion

1: **Algorithm Extended_Kalman_filter**$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$**:**
2: $\bar{\mu}_t = g(u_t, \mu_{t-1})$
3: $\bar{\Sigma}_t = G_t\ \Sigma_{t-1}\ G_t^T + R_t$
4: $K_t = \bar{\Sigma}_t\ H_t^T (H_t\ \bar{\Sigma}_t\ H_t^T + Q_t)^{-1}$
5: $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
6: $\Sigma_t = (I - K_t\ H_t)\ \bar{\Sigma}_t$
7: return μ_t, Σ_t

https://www.cc.gatech.edu/~bboots3/STR-Spring2018/Lectures/Lecture16/Lecture16_notes.pdf

Extended Kalman Filter (EKF)

$$g'(u_t, x_{t-1}) \quad := \quad \frac{\partial g(u_t, x_{t-1})}{\partial x_{t-1}}$$

$$\begin{aligned} g(u_t, x_{t-1}) &\approx g(u_t, \mu_{t-1}) + \underbrace{g'(u_t, \mu_{t-1})}_{=: G_t} (x_{t-1} - \mu_{t-1}) \\ &= g(u_t, \mu_{t-1}) + G_t \, (x_{t-1} - \mu_{t-1}) \end{aligned} \tag{3.50}$$

$$\begin{aligned} & p(x_t \mid u_t, x_{t-1}) \\ &\approx \det (2\pi R_t)^{-\frac{1}{2}} \; \exp \Big\{ -\tfrac{1}{2} \, [x_t - g(u_t, \mu_{t-1}) - G_t \, (x_{t-1} - \mu_{t-1})]^T \\ & \quad R_t^{-1} \, [x_t - g(u_t, \mu_{t-1}) - G_t \, (x_{t-1} - \mu_{t-1})] \Big\} \end{aligned} \tag{3.51}$$

Extended Kalman Filter (EKF)

$$\begin{aligned} h(x_t) &\approx h(\bar{\mu}_t) + \underbrace{h'(\bar{\mu}_t)}_{=: H_t} (x_t - \bar{\mu}_t) \\ &= h(\bar{\mu}_t) + H_t \, (x_t - \bar{\mu}_t) \end{aligned} \tag{3.52}$$

with $h'(x_t) = \frac{\partial h(x_t)}{\partial x_t}$. Written as a Gaussian, we have

$$\begin{aligned} p(z_t \mid x_t) = \det(2\pi Q_t)^{-\frac{1}{2}} \exp\Big\{ -\tfrac{1}{2} \, [z_t - h(\bar{\mu}_t) - H_t \, (x_t - \bar{\mu}_t)]^T \\ Q_t^{-1} \, [z_t - h(\bar{\mu}_t) - H_t \, (x_t - \bar{\mu}_t)] \Big\} \end{aligned} \tag{3.53}$$

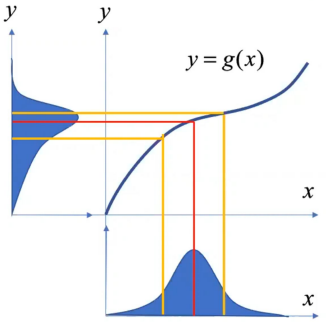
Kalman Filter Algorithms

1: **Algorithm Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)**:**
2: $\bar{\mu}_t = A_t\ \mu_{t-1} + B_t\ u_t$
3: $\bar{\Sigma}_t = A_t\ \Sigma_{t-1}\ A_t^T + R_t$
4: $K_t = \bar{\Sigma}_t\ C_t^T (C_t\ \bar{\Sigma}_t\ C_t^T + Q_t)^{-1}$
5: $\mu_t = \bar{\mu}_t + K_t(z_t - C_t\ \bar{\mu}_t)$
6: $\Sigma_t = (I - K_t\ C_t)\ \bar{\Sigma}_t$
7: return μ_t, Σ_t

1: **Algorithm Extended_Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)**:**
2: $\bar{\mu}_t = g(u_t, \mu_{t-1})$
3: $\bar{\Sigma}_t = G_t\ \Sigma_{t-1}\ G_t^T + R_t$
4: $K_t = \bar{\Sigma}_t\ H_t^T (H_t\ \bar{\Sigma}_t\ H_t^T + Q_t)^{-1}$
5: $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
6: $\Sigma_t = (I - K_t\ H_t)\ \bar{\Sigma}_t$
7: return μ_t, Σ_t

	Kalman filter	EKF
state prediction (Line 2)	$A_t\ \mu_{t-1} + B_t\ u_t$	$g(u_t, \mu_{t-1})$
measurement prediction (Line 5)	$C_t\ \bar{\mu}_t$	$h(\bar{\mu}_t)$

Unscented Kalman Filter - (Julier and Uhlmann [1997])



- Different method to compute covariance matrices
- Does not use the Riccati equations or cov propagation update laws
- Unscented transform
  - find a set of deterministic vectors called sigma points whose ensemble mean and covariance are equal to Z and P
  - apply known nonlinear function y = h(x) to each deterministic vector to obtain transformed vectors.

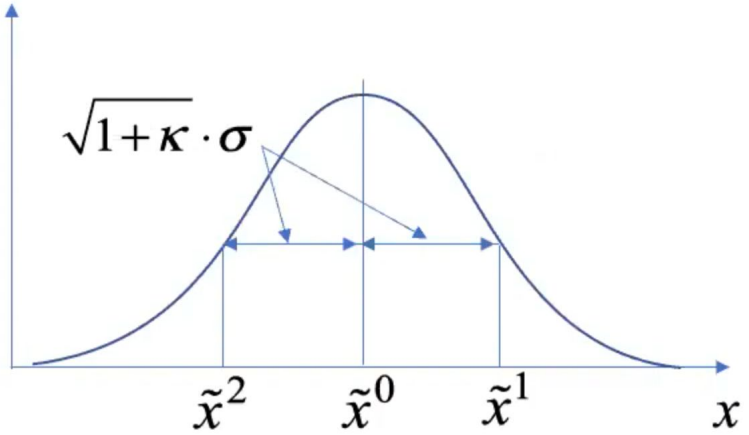
$$p(x) = \frac{1}{\sqrt{2\pi}\cdot\sigma}\exp\left(-\frac{(x-\bar{x})^2}{\sigma^2}\right)$$

$$\tilde{x}^0 = \bar{x} \qquad W_0 = \frac{\kappa}{1+\kappa}$$

$$\tilde{x}^1 = \bar{x} + \sqrt{1+\kappa}\cdot\sigma \qquad W_1 = W_2 = \frac{1}{2(1+\kappa)}$$

$$\tilde{x}^2 = \bar{x} - \sqrt{1+\kappa}\cdot\sigma$$

where κ is a parameter of sigma points to be tuned, and W_i is the weight of the i^{th} sigma point used for computing mean and variance.

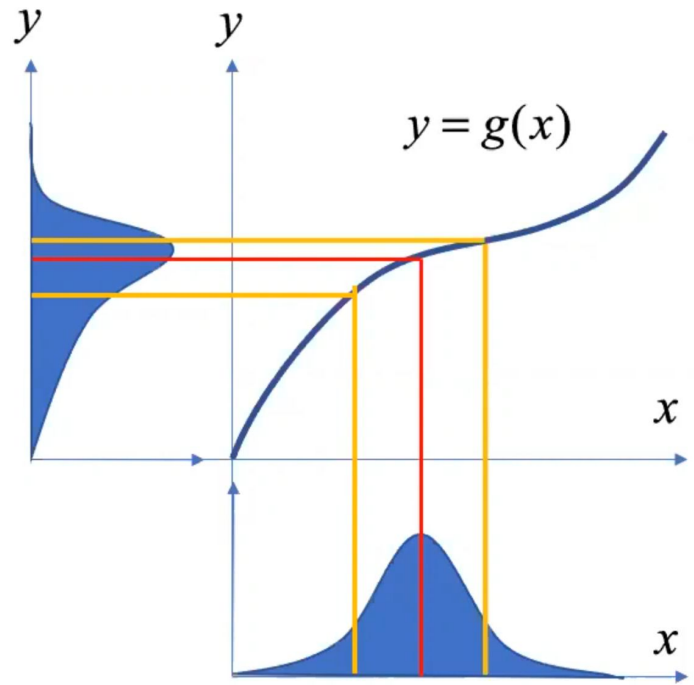


Unscented Kalman Filter - (Julier and Uhlmann [1997])

- Weighted mean of sigma points agree with true mean of the Gaussian distribution

$$\sum_{i=0}^{2} W_i \tilde{x}^i = \frac{\kappa}{1+\kappa}\bar{x} + \frac{1}{2(1+\kappa)}\left\{(\bar{x} + \sqrt{1+\kappa}\cdot\sigma) + (\bar{x} - \sqrt{1+\kappa}\cdot\sigma)\right\}$$

$$= \frac{\kappa}{1+\kappa}\bar{x} + \frac{2}{2(1+\kappa)}\bar{x} = \bar{x}$$



- Weighted variance of sigma points agree with true variance of the Gaussian distribution

$$\sum_{i=0}^{2} W_i (\tilde{x}^i - \bar{x})^2 = \frac{\kappa}{1+\kappa}(\bar{x} - \bar{x}) + \frac{1}{2(1+\kappa)}\left\{(\bar{x} + \sqrt{1+\kappa}\cdot\sigma - \bar{x})^2 + (\bar{x} - \sqrt{1+\kappa}\cdot\sigma - \bar{x})^2\right\}$$

$$= \frac{2}{2(1+\kappa)}(\sqrt{1+\kappa}\cdot\sigma)^2 = \sigma^2$$

Unscented Kalman Filter - (Julier and Uhlmann [1997])

- Weighted mean of sigma points agree with true mean of the Gaussian distribution

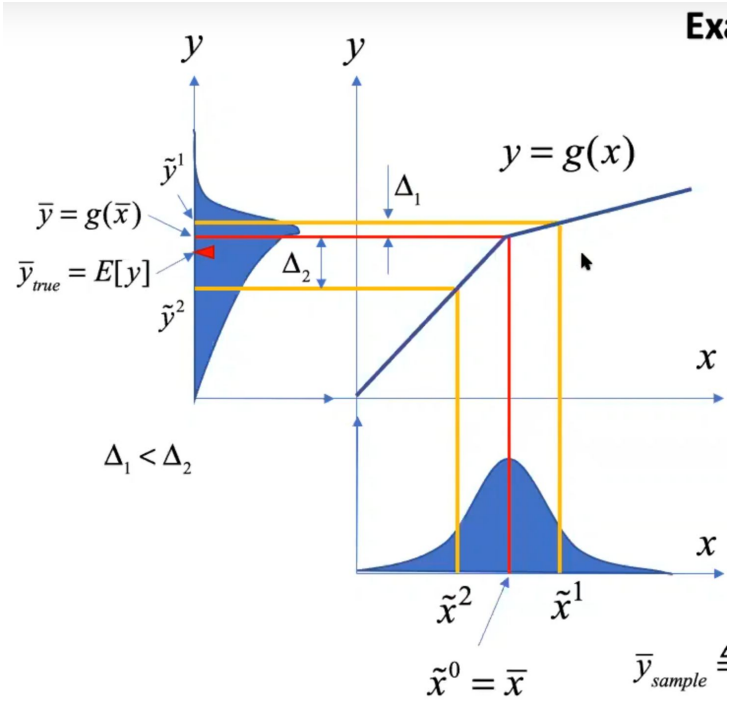
$$\sum_{i=0}^{2} W_i \tilde{x}^i = \frac{\kappa}{1+\kappa}\bar{x} + \frac{1}{2(1+\kappa)}\left\{(\bar{x} + \sqrt{1+\kappa}\cdot\sigma) + (\bar{x} - \sqrt{1+\kappa}\cdot\sigma)\right\}$$

$$= \frac{\kappa}{1+\kappa}\bar{x} + \frac{2}{2(1+\kappa)}\bar{x} = \bar{x}$$

- Weighted variance of sigma points agree with true variance of the Gaussian distribution

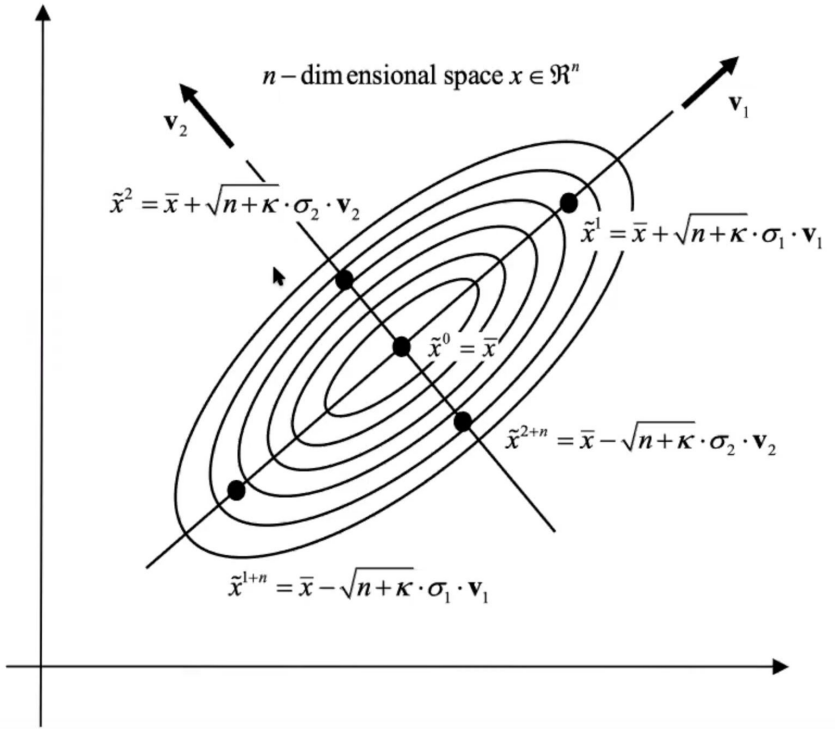
$$\sum_{i=0}^{2} W_i (\tilde{x}^i - \bar{x})^2 = \frac{\kappa}{1+\kappa}(\bar{x} - \bar{x}) + \frac{1}{2(1+\kappa)}\left\{(\bar{x} + \sqrt{1+\kappa}\cdot\sigma - \bar{x})^2 + (\bar{x} - \sqrt{1+\kappa}\cdot\sigma - \bar{x})^2\right\}$$

$$= \frac{2}{2(1+\kappa)}(\sqrt{1+\kappa}\cdot\sigma)^2 = \sigma^2$$



Unscented Kalman Filter - (Julier and Uhlmann [1997])

For n dimensional Gaussian, 2n+1 sigma points can be used



Recursive algorithm for unscented Kalman filter

- Given $\hat{x}_{t-1}$ and P_{t-1}, sample sigma points by computing eigenvalues and eigen vectors of P_{t-1};
- Propagate the sigma points through the nonlinear model to obtain $\tilde{x}^{i*}_{t|t-1} = f(\tilde{x}^i_{t-1}, t-1)$;
- From the (2n+1) sigma points compute the mean and variance:

$$\hat{x}_{t|t-1,sample} = \sum_{i=0}^{2n} W_i \hat{x}^{i*}_{t|t-1} \qquad P_{t|t-1,sample} = \sum_{i=0}^{i=2n} W_i (\tilde{x}^i_{t|t-1} - \hat{x}_{t|t-1,sample})(\tilde{x}^i_{t|t-1} - \hat{x}_{t|t-1,sample})^T + Q_{t-1}$$

- Sample again (2n+1) sigma points for $P_{t|t\text{-}1,sample}$;
- Transform the propagated sigma points to output estimate $\tilde{y}^i_t$ based on the nonlinear measurement equation, and compute the estimated output

$$\hat{y}_{t,sample} = \sum_{i=0}^{2n} W_i \tilde{y}^i_t \qquad \tilde{y}^i_t = h(\tilde{x}^i_{t|t-1}, t)$$

- Evaluate the innovation covariance and the cross covariance by using (2n+1) points of propagated output estimates to find the Kalman gain

$$K_t = P_{xy} P_y^{-1} \qquad P_y = \sum_{i=0}^{2n} W_i (\tilde{y}^i_t - \hat{y}_{t,sample})(\tilde{y}^i_t - \hat{y}_{t,sample})^T + R_t \qquad P_{xy} = \sum_{i=0}^{2n} W_i (\tilde{x}^i_{t|t-1} - \hat{x}_{t|t-1,sample})(\tilde{y}^i_t - \hat{y}_{t,sample})^T$$

- Update the state estimate with the Kalman gain;

$$\hat{x}_t = \hat{\mathbf{x}}_{t|t-1,sample} + K_t[y_t - \hat{y}_{t,sample}]$$

- Update the a posteriori covariance;

$$P_t \cong P_{t|t-1,sample} - K_t P_y K_t^T$$

- Set $t = t + 1$, and repeat the above process.

Unscented Kalman Filter - (Julier and Uhlmann [1997])

- Given $\hat{x}_{t-1}$ and P_{t-1}, sample sigma points by computing eigenvalues and eigen vectors of P_{t-1};
- Propagate the sigma points through the nonlinear model to obtain $\tilde{x}^{i*}_{t|t-1} = f(\tilde{x}^{i}_{t-1}, t-1)$;
- From the (2n+1) sigma points compute the mean and variance:

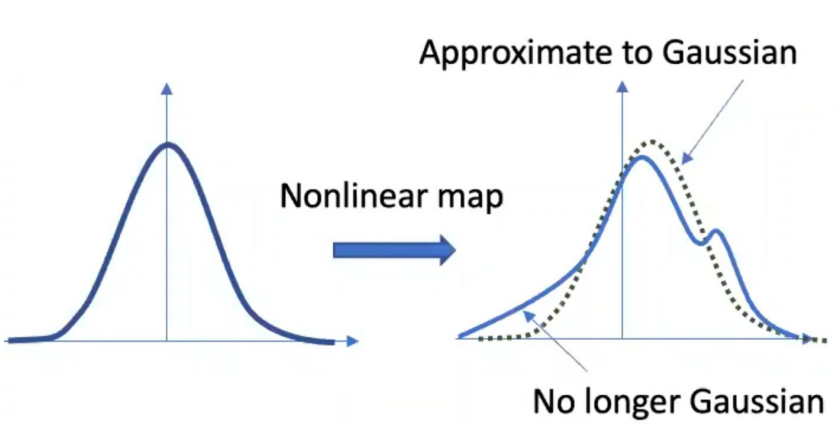
$$\hat{x}_{t|t-1,sample} = \sum_{i=0}^{2n} W_i \hat{x}^{i*}_{t|t-1} \qquad P_{t|t-1,sample} = \sum_{i=0}^{i=2n} W_i (\tilde{x}^{i}_{t|t-1} - \hat{x}_{t|t-1,sample})(\tilde{x}^{i}_{t|t-1} - \hat{x}_{t|t-1,sample})^T + Q_{t-1}$$

- Sample again (2n+1) sigma points for $P_{t|t-1,sample}$;
- Transform the propagated sigma points to output estimate $\tilde{y}^{i}_{t}$ based on the nonlinear measurement equation, and compute the estimated output

$$\hat{y}_{t,sample} = \sum_{i=0}^{2n} W_i \tilde{y}^{i}_{t} \qquad \tilde{y}^{i}_{t} = h(\tilde{x}^{i}_{t|t-1}, t)$$

- Evaluate the innovation covariance and the cross covariance by using (2n+1) points of propagated output estimates to find the Kalman gain

$$K_t = P_{xy} P_y^{-1} \qquad P_y = \sum_{i=0}^{2n} W_i (\tilde{y}^{i}_{t} - \hat{y}_{t,sample})(\tilde{y}^{i}_{t} - \hat{y}_{t,sample})^T + R_t \qquad P_{xy} = \sum_{i=0}^{2n} W_i (\tilde{x}^{i}_{t|t-1} - \hat{x}_{t|t-1,sample})(\tilde{y}^{i}_{t} - \hat{y}_{t,sample})^T$$



- Update the state estimate with the Kalman gain;

$$\hat{x}_t = \hat{x}_{t|t-1,sample} + K_t[y_t - \hat{y}_{t,sample}]$$

- Update the a posteriori covariance;

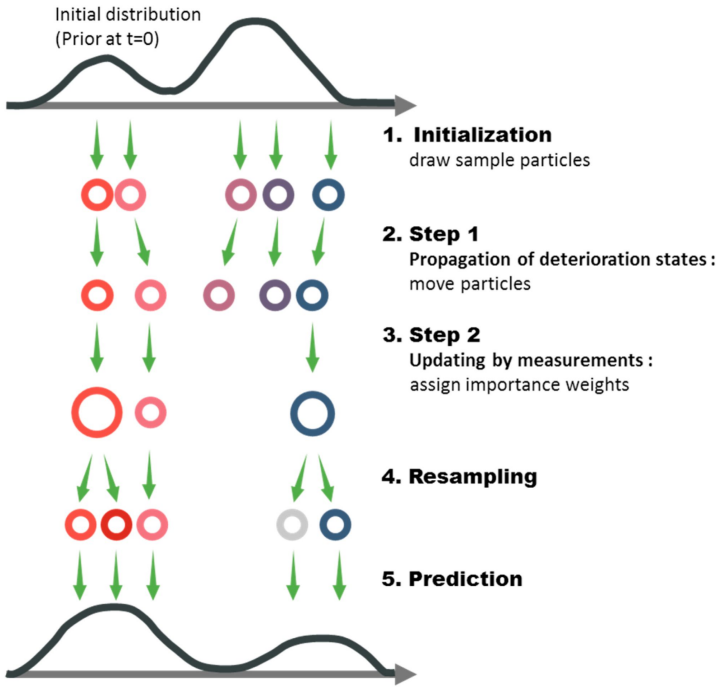
$$P_t \cong P_{t|t-1,sample} - K_t P_y K_t^T$$

- Set $t = t + 1$, and repeat the above process.

- No Jacobian, no partial derivatives are needed.
- The estimated covariance using sigma points is more accurate than the Jacobian-based one.
- ***Caveat!*** The distribution of random variables after transformed through nonlinear equations, e.g. $f(x,t)$, $h(x,t)$, is no longer Gaussian, although the original distribution was Gaussian. Unscented Kalman Filter approximates this distribution to a Gaussian and characterizes with mean and covariance. Although this approximation is accurate to the 2nd order, the discrepancy from a complete Gaussian may grow, as the process is repeated.

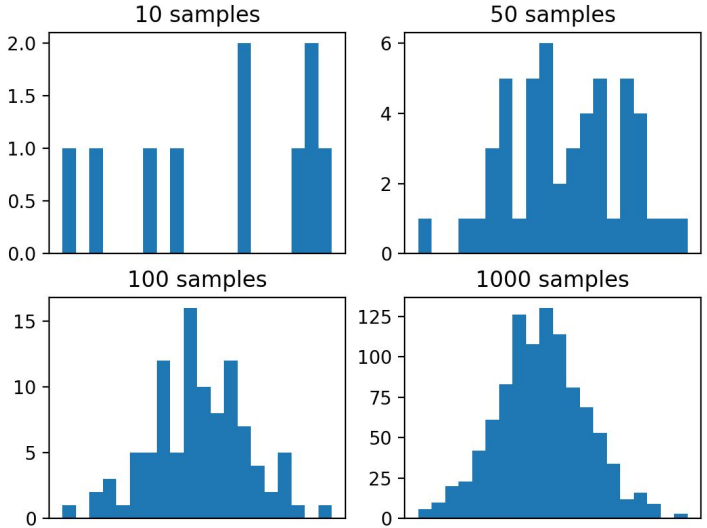
Non-parametric filters

- No fixed functional form
- Approximate posterior by finite number of values
- Can represent a broader space of distributions



Sebastian Thrun
Chapter 4

Non-parametric representation of probability distributions

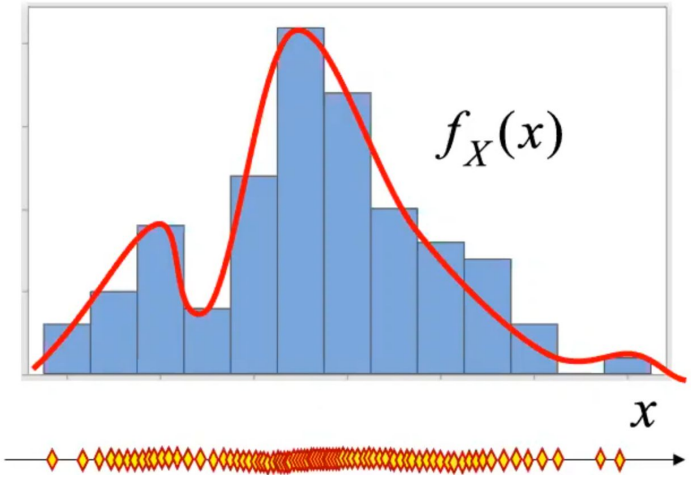


Histograms

Monte Carlo approximation helps in the estimation of mean

$$\mu = \mathbb{E}(f(\boldsymbol{X}))$$

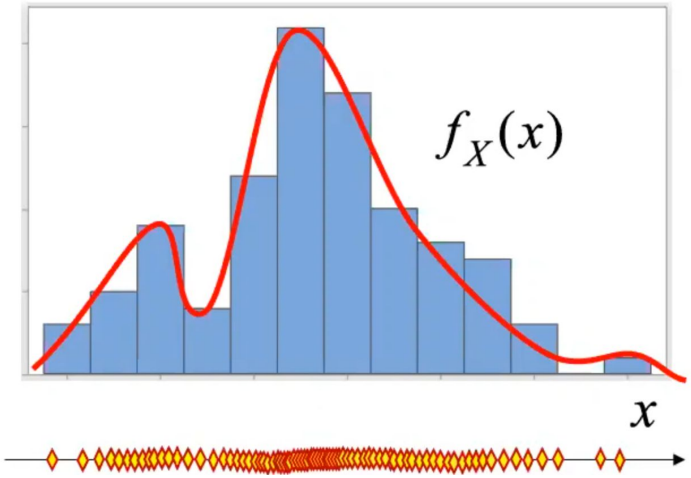
Non-parametric representation of probability distributions



$$\mu = \mathbb{E}(f(\boldsymbol{X}))$$

Draw **particles** that capture probability distribution

Particle filtering



Particles

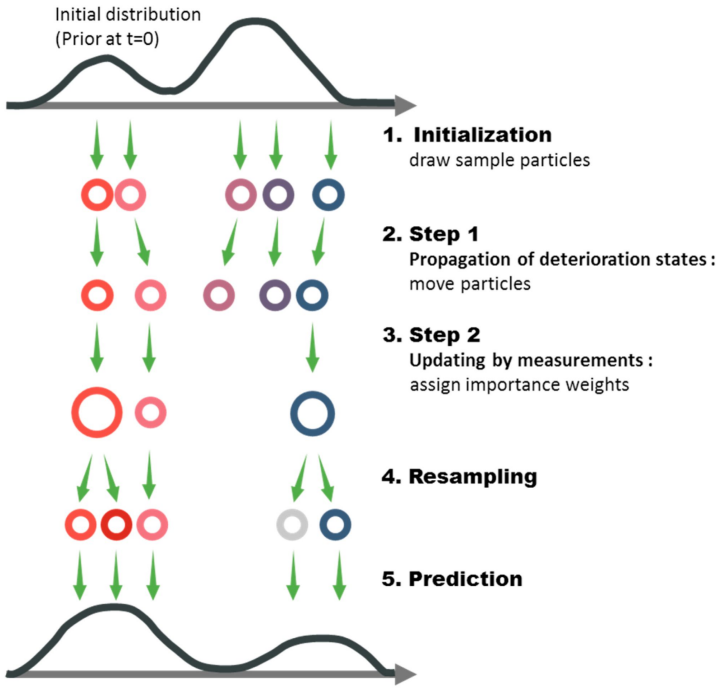
- Bayes filter, non-parametric
- Draw particles that capture probability distribution
- Compute how each point behaves under transformations

Monte Carlo approximation helps in the estimation of mean

$$\mu = \mathbb{E}(f(\boldsymbol{X}))$$

Importance sampling

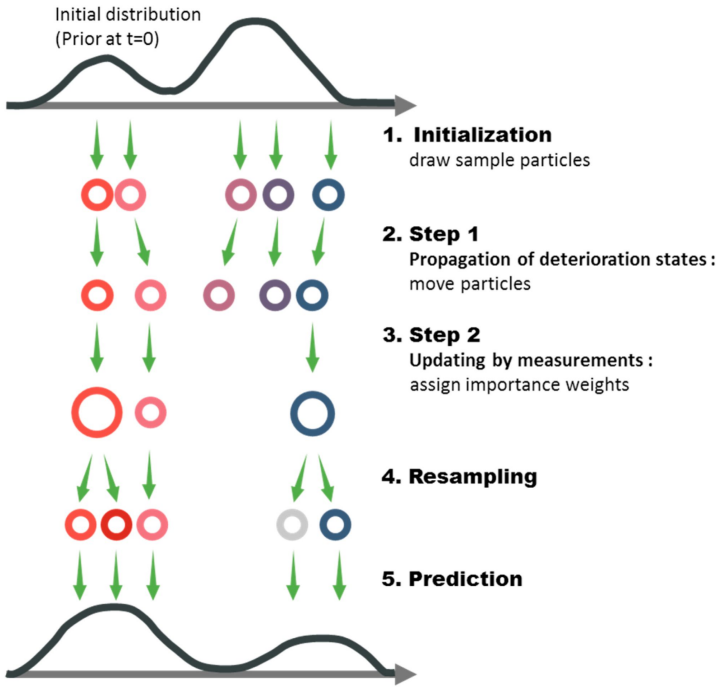
- Get samples from the interesting or important region
- Sample from a distribution that overweights important region
- Adjust estimate to account for having sampled from this other distribution
- Study one distribution while sampling from another



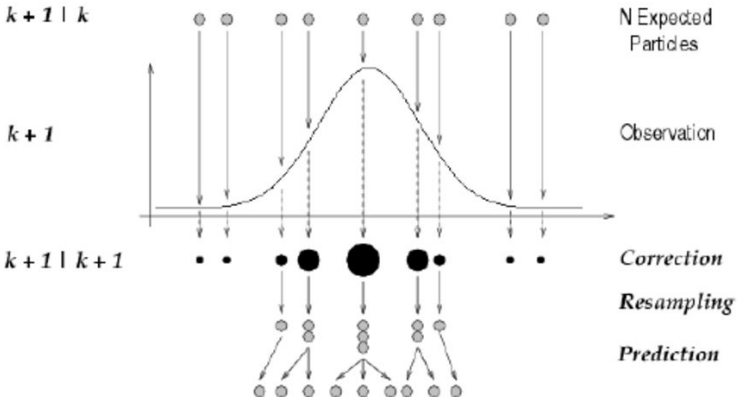
Importance sampling

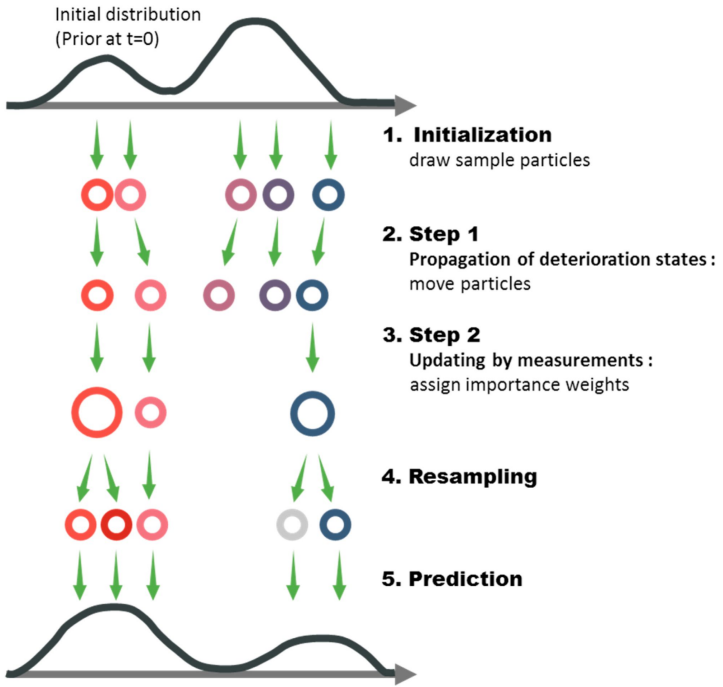
- Get samples from the interesting or important region
- Sample from a distribution that overweights important region
- Adjust estimate to account for having sampled from this other distribution
- Study one distribution while sampling from another

$$\mu = \int_{\mathcal{D}} f(\boldsymbol{x}) p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \int_{\mathcal{D}} \frac{f(\boldsymbol{x}) p(\boldsymbol{x})}{q(\boldsymbol{x})} \, q(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \mathbb{E}_q \left(\frac{f(\boldsymbol{X}) p(\boldsymbol{X})}{q(\boldsymbol{X})} \right)$$



Particle filtering





Particle filtering

Belief

$$\mathcal{X}_t := x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]} \qquad x_t^{[m]} \sim p(x_t \mid z_{1:t}, u_{1:t})$$

Measurement update

$$w_t^{[m]} = p(z_t \mid x_t^{[m]})$$

Sebastian Thrun
Chapter 4.2

Particle filtering

Belief

$$\mathcal{X}_t := x_t^{[1]}, x_t^{[2]}, \ldots, x_t^{[M]}$$

$$x_t^{[m]} \sim p(x_t \mid z_{1:t}, u_{1:t})$$

Measurement update

$$w_t^{[m]} = p(z_t \mid x_t^{[m]})$$

Algorithm Particle_filter$(\mathcal{X}_{t-1}, u_t, z_t)$**:**

$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$

for $m = 1$ to M do

  sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$

  $w_t^{[m]} = p(z_t \mid x_t^{[m]})$

  $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$

endfor

for $m = 1$ to M do

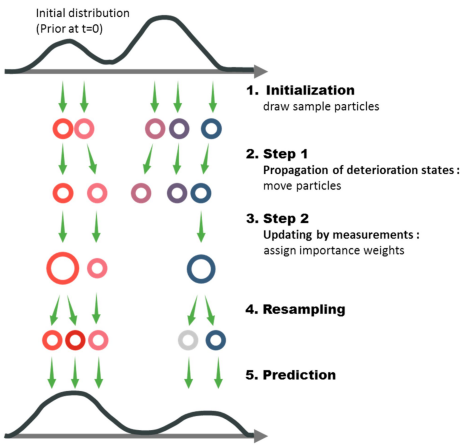
  draw i with probability $\propto w_t^{[i]}$

  add $x_t^{[i]}$ to $\mathcal{X}_t$

endfor

return $\mathcal{X}_t$

belief $bel(x_t)$



Algorithm Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)**:**

$\bar{\mu}_t = A_t\ \mu_{t-1} + B_t\ u_t$

$\bar{\Sigma}_t = A_t\ \Sigma_{t-1}\ A_t^T + R_t$

$K_t = \bar{\Sigma}_t\ C_t^T (C_t\ \bar{\Sigma}_t\ C_t^T + Q_t)^{-1}$

$\mu_t = \bar{\mu}_t + K_t(z_t - C_t\ \bar{\mu}_t)$

$\Sigma_t = (I - K_t\ C_t)\ \bar{\Sigma}_t$

return μ_t, Σ_t

Algorithm Extended_Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$)**:**

$\bar{\mu}_t = g(u_t, \mu_{t-1})$

$\bar{\Sigma}_t = G_t\ \Sigma_{t-1}\ G_t^T + R_t$

$K_t = \bar{\Sigma}_t\ H_t^T (H_t\ \bar{\Sigma}_t\ H_t^T + Q_t)^{-1}$

$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$

$\Sigma_t = (I - K_t\ H_t)\ \bar{\Sigma}_t$

return μ_t, Σ_t

Algorithm Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$)**:**

$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$

for $m = 1$ to M do

  sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$

  $w_t^{[m]} = p(z_t \mid x_t^{[m]})$

  $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$

endfor

for $m = 1$ to M do

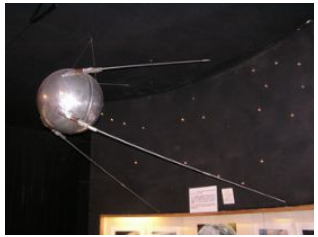
  draw i with probability $\propto w_t^{[i]}$

  add $x_t^{[i]}$ to $\mathcal{X}_t$

endfor

return $\mathcal{X}_t$

Space Race



Year	Nazi Germany	U.S.S.R	U.S.
1944	V2 rocket		
1957		Sputnik	Vanguard (Flopnik, Kaputnik)
1957		Leica	
1958			NASA is born
1961		Yuri Gagarin	Alan Shepard
1961			Kennedy - land a man on the moon and return safely to Earth
1960s			Nova (abandoned - cost and complexity)
1960s			Three stage solution - Saturn
1961			Saturn 1 tested
1965-1966			Gemini missions
1968			Apollo 7, first human (Apollo) test mission
1969			Apollo 11
1972			Apollo 17

350k people involved - F1 engines, no SRBs


MISSION TO PSYCHE:
THE JOURNEY BEGINS

Pixhawk/PX4 architecture