
Overview of Kalman Filter Theory and Navigation Applications

Day 1

Michael L. Carroll

Mar 03, 2004

Course Overview

- Topic Overview
- Course Structure
- The Daily Topics
- Instructor Background

Course Overview: Topic Overview

- Kalman Filtering
 - General Theory Overview w/o proofs
 - Practical Aspects of Application
 - Matlab and Simulink Examples
- Combat Talon I Kalman Filter

Course Overview: Course Structure

- 5 days, 4 segments per day
- Segment 1: 8:30 - 10:00
- Break 10:00 - 10:15
- Segment 2: 10:15 - 11:45
- Lunch 11:45 - 1:00

Course Overview: Course Structure

- Segment 3: 1:00 - 2:30
- Break 2:30 - 2:45
- Segment 4: 2:45 - 4:15

Course Overview: Course Structure: Segment Structure

- Concept
- Example
- Exercise

Course Overview: The Daily Topics

- Day 1: The Basic Kalman Equations, Part I – Heuristic Overview and State Dynamics
- Day 2: The Basic Kalman Equations, Part II – Probability, Statistics and Random Processes
- Day 3: Strapdown Navigation and Inertial Error Dynamics
- Day 4: General Code Implementation Issues
- Day 5: Details of the Combat Talon Navigation Kalman Filter

Course Overview: Instructor Background

- Michael Carroll
- Masters Degree in Mathematics from University of California at Santa Barbara
- Senior Software Systems Engineer for SAIC
- Software Architect on C-130 Avionics Modernization Program at Boeing in Long Beach, CA

Course Overview: Instructor Background

Formerly ...

- Worked with Calibration and Alignment filters for ballistic missile inertial measurement units (IMU)
- Led Combat Talon I software development effort at Lockheed Aircraft Service Company (LASC) in Ontario, CA
- Led Quiet Knight I software development effort at LASC
- Engineering Program Manager for Quiet Knight II at LASC

Course Overview: Instructor Background

Formerly ...

- Founder and CEO of CyberStrategies, Inc in Upland, CA
- GPS/INS Navigation Analyst at BEI Systron Donner in Concord, CA
- Have taught formal Kalman filtering courses numerous times

Day 1: The Basic Kalman Equations, Part I Segments

- Understanding the Equations: Heuristic Overview
- Taking the Equations Apart
- Differential Equations, Difference Equations and Dynamic Systems
- State Space Concepts

Day 1, Segment 1

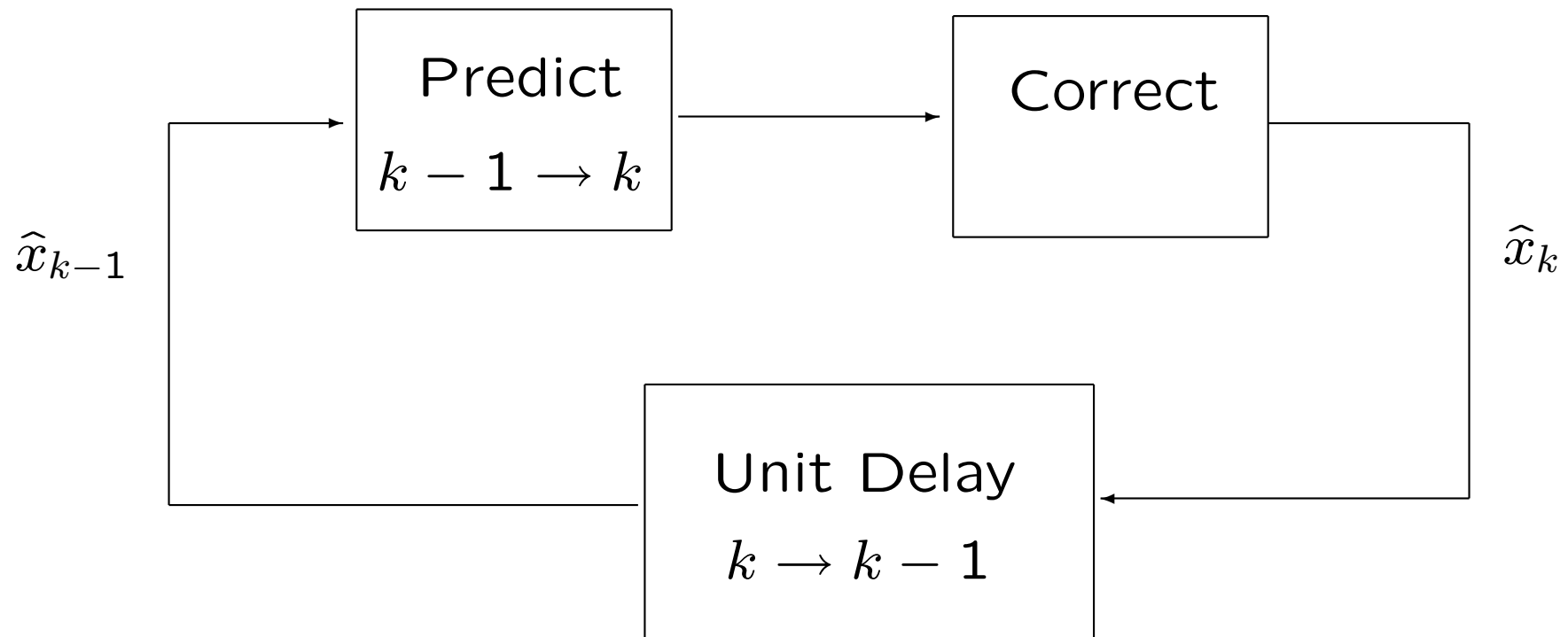
Understanding the Equations: Heuristic Overview Topics

- Recursive Predictor-Corrector Algorithms
- Running Averages
- Extrapolation (Prediction)
- Gain Computation
- Update

Recursive Predictor-Corrector Algorithm

- Prediction:
 - Kalman filter **extrapolates** the solution from the previous iteration to the present time
- Correction:
 - Kalman filter uses measurement information to **update** or **correct** the prediction
- And the cycle begins anew!

Kalman Filter Predictor Corrector Model



Recursive Predictor-Corrector Algorithm

- Algorithm
 - Computational procedure
 - Given initial values of some quantities, new quantities are computed
- Recursive
 - The new quantities are the same as the old quantities
 - New quantities become initial values for next cycle

Simple Example: Running Averages

- Suppose we want to estimate some quantity like the resistance of a single resistor.
- Suppose we have a voltmeter with which to make measurements.
- Thus we have two distinct quantities: the "true" value x of the resistor and the measurement z given by the voltmeter.
- We are going to make an indefinite number of measurements:
 z_1, z_2, \dots

Simple Example: Running Averages

- The **state** that we are trying to estimate is the resistance of the resistor.
- We don't know what the state is. Otherwise, we wouldn't need to take measurements.
- Our **state dynamics** or **process model** is very simple: The state is 1-dimensional and does not change with time:

$$x_{k+1} = x_k$$

Simple Example: Running Averages

- What is the best way to estimate the state of our system?
- In the absence of any other information, we should probably just average the measurements.
- We assume that the averages are unbiased, i.e., the measurement errors have zero mean.

Simple Example: Running Averages

- Simple running average is like a Kalman filter with trivial dynamics ($x_{k+1} = x_k$)

Given an infinite sequence of random measurements z_1, z_2, \dots , compute at each stage the best estimate of the true value of the underlying quantity: At the $N + 1^{\text{st}}$ step, your best guess would be the average of all previous values:

$$(1) \quad \hat{x}_{N+1} = \frac{1}{N} \sum_{k=1}^N z_k = \frac{1}{N} (z_1 + z_2 + \dots + z_N)$$

Simple Example: Running Averages

- Why is the average of all previous measurements the "best" estimate?
- The dynamics tell us that the quantity to be estimated is not changing.
- The only variability stems from the measurements, i.e., the measurement noise.

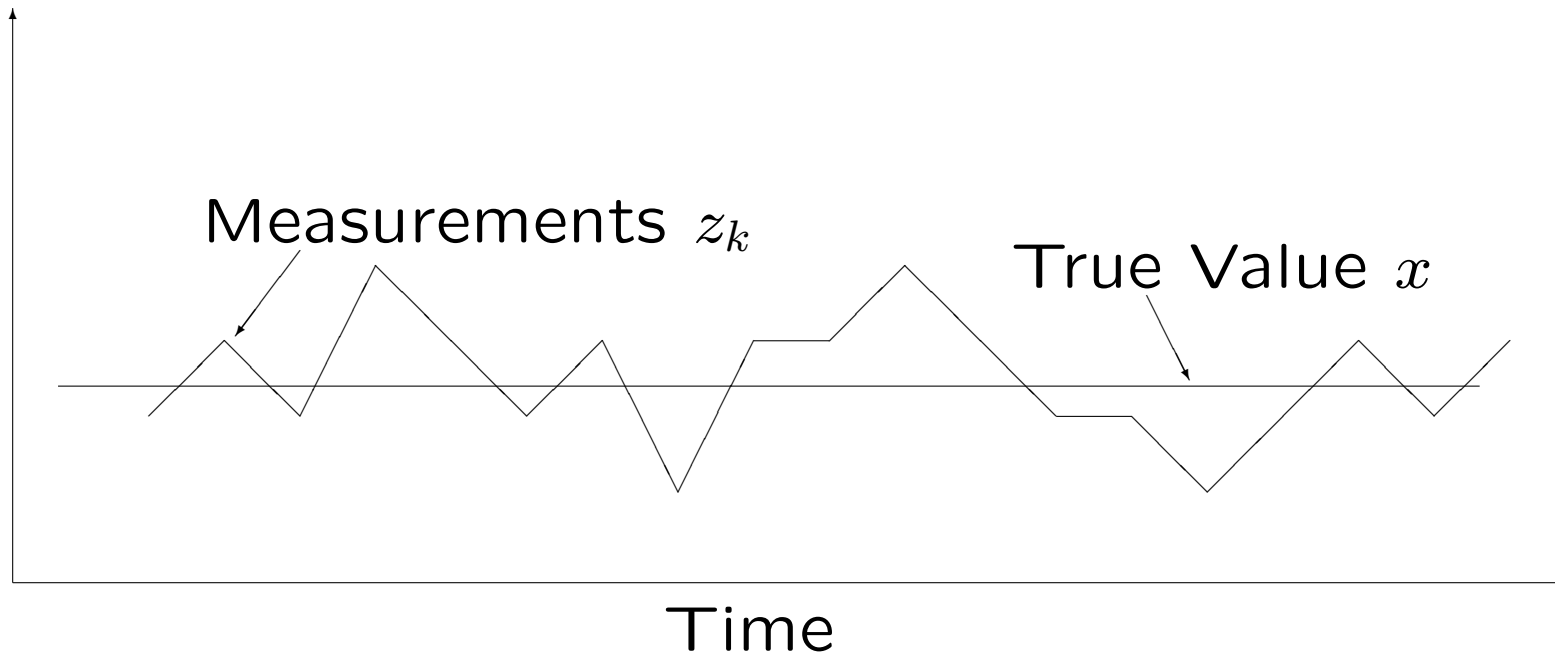
Recursive Running Average

- Therefore, as much as possible, we want to rid ourselves of the noise by averaging.
- Assume that noise is zero-mean, i.e., its average tends to zero over the long run.

Recursive Running Average

- If we had a bias, we wouldn't call that noise. That would be a systematic error on the part of our voltmeter. We'd have to calibrate it out.
- Noise can't be calibrated out; we have to average it out over time.

Ohms



Recursive Running Average

- Now suppose we are lazy and don't want to recalculate the complete sum with each new measurement.
- We just want to use the previously calculated value and add a small adjustment:

Recursive Running Average

$$\begin{aligned}\hat{x}_{N+1} &= \frac{1}{N} \sum_{k=1}^N z_k && \text{by def. Eq. (1)} \\ &= \frac{1}{N} \sum_{k=1}^{N-1} z_k + \frac{1}{N} z_N && \text{separate last term from sum} \\ &= \frac{1}{N} \cdot \frac{N-1}{N-1} \sum_{k=1}^{N-1} z_k + \frac{1}{N} z_N && \text{multiply by 1} \\ &= \frac{N-1}{N} \cdot \frac{1}{N-1} \sum_{k=1}^{N-1} z_k + \frac{1}{N} z_N && \text{rearrange} \\ &= \frac{N-1}{N} \hat{x}_N + \frac{1}{N} z_N && \text{by def. Eq. (1) again} \\ &= \hat{x}_N - \frac{1}{N} \hat{x}_N + \frac{1}{N} z_N && \text{distribute and simplify} \\ &= \hat{x}_N + \frac{1}{N} (z_N - \hat{x}_N). && \text{rearrange and factor out } 1/N\end{aligned}$$

Recursive Running Average

- We now have a recursive algorithm for computing a running average
 - Recipe: Take new measurement, subtract off old average and weight the resulting residual by a gain of $1/N$.
 - Add this weighted residual to the old average and, voilà, you have the new average!

Recursive Running Average

- Note that you do not have to remember all the previous values or averages, only the last average.
- Note also that as N goes to ∞ , the gain $1/N$ goes to 0. (The filter is going to sleep!)

$$(2) \quad \hat{x}_{N+1} = \hat{x}_N + \frac{1}{N} (z_N - \hat{x}_N)$$

Recursive Running Average

- The residual $z_N - \hat{x}_N$ is the difference between the actual new measurement and what we expected or predicted the new measurement to be.
- Why is \hat{x}_N our predicted measurement?
- Answer: In the absence of new information, the previous estimate is the only thing we have to go on!

Extrapolation (Prediction)

- In the running average example, the dynamics were trivial, i.e., $x_{k+1} = x_k$
- This means that the state never really changes. It's always equal to its previous value.
- We just don't know what the exact value is.
- Example: Measuring the same resistor over and over again with a voltmeter to estimate the resistor's resistance value.

Extrapolation (Prediction)

- In more general Kalman filtering, we complicate the situation by adding two more elements:
 - State dynamics: $x_{k+1} = f(x_k)$
 - Process noise: $x_{k+1} = f(x_k) + w_k$
- The form of the function f can be simple or complicated.
- The process noise is assumed to be Gaussian white noise (to be defined later).

Extrapolation (Prediction)

- General Kalman filtering further complicates things by allowing the state variable to be vector-valued.
- This turns the function f into a vector-valued function, often a matrix (possibly time-dependent)
- The process noise must therefore also become a vector-valued variable.

Extrapolation (Prediction)

- Note that the presence of a random forcing function like white noise turns the dynamical problem into a stochastic one.
- If the white noise were zero, then the problem would be deterministic.
- We'll see what this means when we take a look at random variables and random processes.

Extrapolation (Prediction)

- Note that we did not mention a control vector u which would also be part of the forcing function.
- We are generally going to ignore the presence of control.
- The control vector is usually considered to be deterministic although in reality you can't control things perfectly! (E.g., elevator deflection angle might be a control quantity and its precise value may not be known!)

Kalman Gain Computation

- Before we can incorporate the information contained in both the old estimate and the new measurement, we have to first calculate the Kalman gain K_N at the N^{th} step.
- $K_N = \frac{1}{N}$ in our running average example.
- If we have N pieces of information, it seems intuitive that we weight each piece by $1/N$.

Kalman Gain Computation

- What we are weighting is the measurement residual, also called the **innovation**: $z_N - \hat{x}_N$.
- The innovation contains all the new information, i.e., the new measurement minus what we already knew, i.e., the previous estimate.
- The gain computation occurs after the extrapolation but before updating our estimate with measurement information.

Measurement Update (Correction)

- In the running average example, our update implicitly assumed that our measurement was corrupted by white noise.
- A model for this would be: $z_k = x_k + v_k$, where v_k is a Gaussian white measurement noise sequence.
 - We'll explore what we mean by "Gaussian" later in the course.

Measurement Update (Correction)

- Because we assume that our measurement noise was white Gaussian and zero mean, our best guess was simply the mean value of the measurements.
- Remember the resistor example: our voltmeter is not perfect. Each measurement will differ slightly from the previous one, even though we are measuring the same resistor each time.

Measurement Update (Correction)

- General Kalman filtering complicates the measurement model by adding two new elements:
 - The measurement may have a more complicated relationship to the underlying system state: $z_k = h(x_k) + v_k$, where h could be a non-linear function.
 - Just as the state x is allowed to be a vector variable, so the measurement z is also allowed to be a vector variable.

Day1, Segment 1 - Exercises

1. Let's re-familiarize ourselves with vectors and matrices. Addition:

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_n + y_n \end{bmatrix}$$

For vectors in three space, this would be:

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \\ x_3 + y_3 \end{bmatrix}$$

Compute:

$$\begin{bmatrix} -1 \\ 3 \\ 0 \end{bmatrix} + \begin{bmatrix} 7 \\ -1 \\ 5 \end{bmatrix} =$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix} =$$

$$\begin{bmatrix} 0.75 \\ -3 \\ 100 \end{bmatrix} + \begin{bmatrix} 0.75 \\ 0 \\ -1000 \end{bmatrix} =$$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} =$$

$$\begin{bmatrix} 1.1 \\ -3.2 \\ 9.7 \end{bmatrix} + \begin{bmatrix} -0.8 \\ 3.2 \\ 10 \end{bmatrix} =$$

2. Multiplying a row vector times a column vector:

$$\begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = a_1b_1 + a_2b_2 + a_3b_3$$

Now you try it:

$$\begin{bmatrix} 0 & 1 & 3 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ 0 \end{bmatrix} =$$

$$\begin{bmatrix} -1 & 3 & -2 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \\ 2 \end{bmatrix} =$$

$$\begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} =$$

$$\begin{bmatrix} 2x & .5x & -7x \end{bmatrix} \begin{bmatrix} -10 \\ 4 \\ -1 \end{bmatrix} =$$

3. Multiplying a matrix times a column vector:

$$\begin{bmatrix} 1 & 0 & 1 \\ 2 & 7 & 0 \\ 3 & 2 & 5 \end{bmatrix} \begin{bmatrix} 7 \\ -1 \\ 5 \end{bmatrix} =$$

Hint: Multiply the first row of the matrix times the column vector to get the first element of the new column vector. Then multiply by the second row of the matrix, and so on.

$$\begin{bmatrix} 2 & 1 & 0 \\ 0 & 1 & 1 \\ 2 & 2 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ -3 \\ 1 \end{bmatrix} =$$

$$\begin{bmatrix} 3 & 1 & 1 \\ -1 & 1 & -1 \\ 0 & 2 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ -3 \\ 1 \end{bmatrix} =$$

4. What happens if you multiply a column on the left by a row vector on the right?

5. What is the transpose of the following matrix?

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

6. Given the following scalar measurements of some constant scalar process, calculate the average value at each stage k by two different methods: 1) by averaging all past values and 2) by using the recursive algorithm in Eq. (2):

1. $z_1 = 0.76$

2. $z_2 = 0.67$

3. $z_3 = 0.79$

4. $z_4 = 0.81$

5. $z_5 = 0.78$

6. $z_6 = 0.79$

7. $z_7 = 0.81$

8. $z_8 = 0.82$

9. $z_9 = 0.77$

7. You are traveling in a car at 45 mph in a 35 mph speed zone. Your friendly neighborhood policeman is watching you with his radar gun. He is measuring your speed, of course, but needs to report as well your position as a function of time. Write down a dynamics and measurement model for this problem in which the state variable is your positional displacement from some arbitrary initial position. Assume that there is no process noise in your dynamics and that your velocity is not changing. Assume the policeman makes a measurement with his radar gun every 2 seconds.

Hint: You have two variables of interest! 8. How is this problem different from the resistor problem?

Day 1, Segment 2

Taking the Equations Apart

Topics

- Understanding the Notation
- Examples
- Exercises

Understanding the Notation: Kalman Filter Problem Summary

- Given two models:
 1. State Dynamics Model (sometimes called System Model)
 2. Measurement Model
- Find the optimal linear gain K_k with which to weight the measurement residual or innovation in order to compute the optimal estimate of the state (we'll discuss later what we mean by "optimal")

Understanding the Notation: Kalman Filter Problem Summary (General, Non-Linear, Discrete Formulation)

- Given: State Dynamics Model: $x_k = f(x_{k-1}) + w_k$
 - x =state, f =dynamics function, w =process noise
- Given: Measurement Model: $z_k = h(x_k) + v_k$
 - z =measurement, h =geometry function, v =measurement noise
- Find optimal gain K_k that allows us to update our state estimate in the following way:

$$\hat{x}_k = f(\hat{x}_{k-1}) + K_k[z_k - h(f(\hat{x}_{k-1}))]$$

Understanding the Notation: Kalman Filter Problem Summary

- Note that we seek the gain in the form of a multiplier (i.e., a gain matrix)
- This is not the most general form possible. The gain could be a (possibly, non-linear) function of the innovation, perhaps something like $g_k(z_k, \hat{x}_k)$. But we won't pursue that.

Understanding the Notation: Kalman Filter Problem Summary (Linear Discrete Formulation)

- Given: State Dynamics Model: $x_k = \Phi_{k-1}x_{k-1} + w_k$
 - x_k =state, Φ_{k-1} =state transition matrix, w_k =process noise
- Given: Measurement Model: $z_k = H_kx_k + v_k$
 - z_k =measurement, H_k =geometry matrix, v_k =measurement noise
- Find: State estimate in the form

$$\hat{x}_k = \Phi_{k-1}\hat{x}_{k-1} + K_k[z_k - H_k\Phi_{k-1}\hat{x}_{k-1}]$$

Understanding the Notation: Kalman Filter Problem Summary

Slight Notational Change

- The term $\Phi_{k-1}\hat{x}_{k-1}$ is the extrapolated state estimate prior to making the measurement update or correction.
- To distinguish prior estimate from updated estimate, we use the following notation:

$\hat{x}_k(-) = \Phi_{k-1}\hat{x}_{k-1}(+) =$ predicted estimate prior to update

$\hat{x}_k(+)$ = updated state estimate

Understanding the Notation: Kalman Filter Problem Summary

Slight Notational Change

- Some authors use \hat{x}_k^+ and \hat{x}_k^- instead of $\hat{x}_k(+)$ and $\hat{x}_k(-)$
- Also, sometimes we use $x_{k+1} = \Phi_k x_k + w_k$ instead of $x_k = \Phi_{k-1} x_{k-1} + w_k$
 - Doesn't really matter
 - Depends on perspective: pulling from the past vs. pushing into the future

Understanding the Notation: Kalman Filter Problem Summary

- Thus, the solution we seek looks like this:

$$(3) \quad \hat{x}_k(+)=\hat{x}_k(-)+K_k[z_k-H_k\hat{x}_k(-)]$$

Understanding the Notation: Kalman Filter Problem Summary

- We've formulated the discrete problem
- We've indicated the form of the solution: the recursive estimating algorithm with the Kalman gain
- But we haven't yet presented the general solution. Only in the case of a simple, constant, scalar system.
- When we arrive at the general solution, we'll get an added benefit: While trying to come up with $\hat{x}_k(+)$ and K_k , we'll also end up with a recursive estimate of our estimation uncertainty! This is the so-called error covariance matrix.

Understanding Kalman Filter Notation State Dynamics

- State Dynamics Model: $x_k = f(x_{k-1}) + w_k$
- x_k is the **state vector** with n elements

- $x_k = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}_k = \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_n(k) \end{bmatrix}$

Understanding Kalman Filter Notation

State Dynamics

- Discrete State Dynamics
- General, nonlinear dynamics function f too difficult!
- Restrict to matrices: $x_k = \Phi_{k-1}x_{k-1} + w_k$
- Φ_{k-1} is the state transition matrix. It is $n \times n$ in size.

Understanding Kalman Filter Notation

State Dynamics

- State Transition Matrix: $\Phi_{k-1} = \Phi(k, k - 1)$
- Extrapolates state vector from time step $k - 1$ to time step k (pulling from past to present)
- Tells you how the state would evolve in the absence of forcing functions
- Remember: this is still just a model. We don't really know what the state is, because the random forcing function messes things up!

Understanding Kalman Filter Notation State Dynamics

- We are deliberately trying to keep the notation as simple as possible
- We could throw in other factors and terms.
 - A matrix Γ_k to distribute the Gaussian white noise to the states
 - An additional, deterministic control variable u_k , and associated distribution matrix Λ_k .
- But we won't!

The Kalman Filter Solution: State Extrapolation

- Given an estimate $\hat{x}_{k-1}(+)$ at time $k - 1$, we extrapolate it forward in time by the state transition matrix:

$$(4) \quad \hat{x}_k(-) = \Phi_k \hat{x}_{k-1}(+)$$

- Why is this the right extrapolation equation? Because the white noise forcing function is a zero mean process. Thus, on average, the equation is homogeneous and we should therefore use the homogeneous solution, i.e., the state transition matrix.

The Kalman Filter Solution: Covariance Extrapolation

- How certain are we that our solution is right?
- The answer is given by the **estimation error covariance matrix** or often just called the covariance matrix P_k .
- Just like the state estimate, it is propagated forward in time before being corrected:

$$(5) \quad P_k(-) = \Phi_k P_{k-1}(+) \Phi_k^T + \Gamma_k Q_k \Gamma_k^T$$

where Q_k is a matrix called the **process noise covariance matrix**

The Kalman Filter Solution: Covariance Extrapolation

- Generally, we'll assume that the noise distribution matrix Γ_k is the identity matrix. Thus, for covariance extrapolation we'll write instead of (5) the following:

$$(6) \quad P_k(-) = \Phi_k P_{k-1}(+) \Phi_k^T + Q_k$$

- Q_k measures the amount of dispersion in the white noise forcing function w_k
- Later we'll learn how to define P_k .

The Kalman Filter Solution: Covariance Extrapolation

- What does Eq. (6) really mean?
- The term $\Phi_k P_{k-1} (+) \Phi_k^T$ represents the effect that the state dynamics has on the error covariance.
- The other term Q_k represents the increased uncertainty added into each step due to the process noise inherent in the system. This is due to the white noise in the system, possibly due to the aggregate effect of many unmodelled states.

The Kalman Filter Solution: The Kalman Gain

- The heart of the solution is the **Kalman Gain**: K_k

$$(7) \quad K_k = P_k(-)H_k^T [H_k P_k(-)H_k^T + R_k]^{-1}$$

where R_k is the measurement noise covariance matrix (to be defined later) governing the white measurement noise v_k in the measurement model:

$$z_k = H_k x_k + v_k$$

- The discovery of K_k was one of Rudolf Kalman's main contributions.

The Kalman Filter Solution: State Update Equation

- Using the gain, the state update equation (which we've already seen) is

$$(8) \quad \hat{x}_k(+)=\hat{x}_k(-)+K_k[z_k-H_k\hat{x}_k(-)]$$

where z_k is the measurement and H_k is the geometry matrix at time k .

The Kalman Filter Solution: The Covariance Update Equation

- Likewise, the error covariance P_k is corrected using the gain:

$$(9) \quad P_k(+)= [I - K_k H_k] P_k(-)$$

The Five Kalman Equations

$$(10) \quad \text{State Extrapolation: } \hat{x}_k(-) = \Phi_k \hat{x}_{k-1}(+)$$

$$(11) \quad \text{Covariance Extrapolation: } P_k(-) = \Phi_k P_{k-1}(+) \Phi_k^\top + Q_k$$

$$(12) \quad \text{Kalman Gain: } K_k = P_k(-) H_k^\top \left[H_k P_k(-) H_k^\top + R_k \right]^{-1}$$

$$(13) \quad \text{State Update: } \hat{x}_k(+) = \hat{x}_k(-) + K_k [z_k - H_k \hat{x}_k(-)]$$

$$(14) \quad \text{Covariance Update: } P_k(+) = [I - K_k H_k] P_k(-)$$

Examples

- Scalar Example: Resistor Revisited
- Damped Harmonic Oscillator

Scalar Example: Resistor Revisited

- Number of states: 1
- System Dynamics: $x_k = x_{k-1}$
- Measurement Model: $z_k = x_k + v_k$, where v_k is zero mean, Gaussian white noise with variance σ^2
- Thus, $\Phi_k = 1$, $H_k = 1$, $Q_k = 0$, $R_k = \sigma^2$

Scalar Example: Resistor Revisited

- State extrapolation and covariance extrapolation are simple:

$$\hat{x}_k(-) = \hat{x}_{k-1}(+)$$

$$P_k(-) = P_{k-1}(+)$$

where P is a 1×1 matrix or scalar.

- The Kalman gain equation $K_k = P_k(-)H_k^\top [H_k P_k(-)H_k^\top + R_k]^{-1}$ becomes:

$$K_k = P_k(-) (P_k(-) + \sigma^2)^{-1}$$

Scalar Example: Resistor Revisited

- State update equation $\hat{x}_k(+)$ becomes

$$\hat{x}_k(+)=\hat{x}_k(-)+\frac{P_k(-)}{P_k(-)+\sigma^2}(z_k-\hat{x}_k(-))$$

- Covariance update equation $P_k(+)$ becomes

$$P_k(+)=\left(1-\frac{P_k(-)}{P_k(-)+\sigma^2}\right)P_k(-)$$

Damped Harmonic Oscillator

- Number of states = 2. System model:

$$\begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} = \exp \left(\begin{bmatrix} 0 & 1 \\ -k/m & -c/m \end{bmatrix} \Delta t \right) \begin{bmatrix} x_1(k-1) \\ x_2(k-1) \end{bmatrix} + \begin{bmatrix} w_1(k) \\ w_2(k) \end{bmatrix}$$

- Thus, $\Phi_k = \exp \left(\begin{bmatrix} 0 & 1 \\ -k/m & -c/m \end{bmatrix} \Delta t \right)$

- Here $\Delta t = t_k - t_{k-1}$

- Note that exp is a matrix exponential.

Damped Harmonic Oscillator

- Assume that the process noise w has the following statistics:

$$Q_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

- This means that there is unity variance in position uncertainty and unity variance in velocity uncertainty.
- It also means that the uncertainties in position and velocity are uncorrelated (i.e., no non-zero off-diagonal elements in Q_k).

Damped Harmonic Oscillator

- Measurement model:

$$z_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + v_k$$

where the measurement noise v_k has variance σ^2

- Thus $H_k = \begin{bmatrix} 1 & 0 \end{bmatrix}$
- Note that we only have a position measurement.

Damped Harmonic Oscillator

- Φ_k can be approximated through series expansion of

$$\exp \begin{bmatrix} 0 & \Delta t \\ -k/m\Delta t & -c/m\Delta t \end{bmatrix}$$

- Or we can use Matlab's `expm` function for specific values of k, c, m , and Δt .

Damped Harmonic Oscillator

- Because Φ and Q are matrices, the state and covariance extrapolation equations do not simplify from their general matrix form.

- However, the gain equation $K_k = P_k(-)H_k^T [H_k P_k(-)H_k^T + R_k]^{-1}$ looks like this:

$$K_k = \begin{bmatrix} p11_k(-) \\ p21_k(-) \end{bmatrix} \left(\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} p11_k(-) \\ p21_k(-) \end{bmatrix} + \sigma^2 \right)^{-1} = \begin{bmatrix} \frac{p11_k(-)}{p11_k(-) + \sigma^2} \\ \frac{p21_k(-)}{p11_k(-) + \sigma^2} \end{bmatrix}$$

where $p11_k(-)$ and $p21_k(-)$ are the (1,1) and (2,1) elements of $P_k(-)$.

Damped Harmonic Oscillator

- State and covariance update equations left as an exercise!

Exercises 1. Crank through the resistor example using $\hat{x}_0 = 100\Omega$, $\sigma = 0.1$, and $P_0 = 0$. Start the recursive loop with the gain equation first. Run through the loop three times.

2. Draw a rough plot of the error covariance, plotting both the extrapolated and updated covariance for any give k at the same time. (This will yield the classic Kalman sawtooth pattern.)

3. Similarly, draw a rough plot of the state estimates.
4. Write out the state update equation for the damped harmonic oscillator, using the Kalman gain derived in the example.
5. Write out the covariance update equation for the damped harmonic oscillator, using the Kalman gain derived in the example.

Day 1, Segment 3

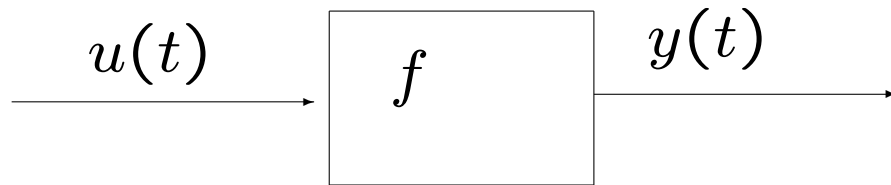
Differential Equations and Difference Equations

Topics

- Input-Output Representation
- Differential Equations and Difference Equations

Input-Output Representation

- Black box approach: $y(t) = f(u(t), t)$



- Output represents the resultant behavior of the dynamic system as a function of the input (and possibly time).
- Output is often one primary scalar variable of interest like position.

Input-Output Representation

- Note that output is not necessarily the same thing as the observation or measurement.
- Observation or measurement is often related directly to the output through a matrix coupling.
- In state space representation, the output is often eliminated in favor of the state vector, and the measurement is given as a function of the state vector directly.

Input-Output Representation

- The derivatives of position and how they relate to the forcing function yields a single differential equation.
- We'll see later how an n^{th} -order, scalar differential equation is related to a vector state space model.

Differential Equations and Difference Equations

- Ordinary Differential Operators
- Ordinary Differential Equations
- Difference Equations

Ordinary Differential Operators: Vectors and Functions

- Vectors are functions and functions are vectors

- A vector $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ in 3-space is really a function $x : \{1, 2, 3\} \rightarrow \mathbb{R}$

- We just write them differently than functions, i.e., instead of $x(i)$ we write x_i .

Ordinary Differential Operators: Vectors and Functions

- And since the domain $\{1, 2, 3\}$ of the function x is so small, we can exhibit all values of x explicitly in one fell swoop, i.e.,

as $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$

- But, by the same token, ordinary real functions $f : \mathbb{R} \rightarrow \mathbb{R}$ are vectors
- They just have so many components that we can't exhibit them all at once

Ordinary Differential Operators: Vectors and Functions

- We can add vectors by adding their components: $z = x + y$ if and only if $z_i = x_i + y_i$ for each index i
- By the same token, we can add functions componentwise: $h = f + g$ if and only if $h(t) = f(t) + g(t)$ for all t in the domain
- Componentwise multiplication of functions can also be defined (although this does not come up as often for ordinary vectors): $h = f \cdot g$ if and only if $h(t) = f(t) \cdot g(t)$ for all t in the domain

Ordinary Differential Operators: Vectors and Functions

- Finally, we can multiply functions by scalars, just like we do for vectors: $g = a \cdot f$ if and only if $g(t) = a \cdot f(t)$ for all $t \in \mathbb{R}$.
- The upshot of all this is that the set of all real functions is a vector space!
- But we are really more interested in a subspace of this vector space, i.e., we really want to restrict our attention to the set of all **differentiable** functions on some closed interval.

Ordinary Differential Operators: Differentiable Functions

- Let $I = [a, b]$ be any closed interval of the real line
- Consider the set $\mathbb{R}^I = \{f : I \rightarrow \mathbb{R}\}$ of all functions mapping $I = [a, b]$ into the real numbers
- Then, as we have just pointed out, \mathbb{R}^I is a vector space over the reals

Ordinary Differential Operators

- We define the **derivative** of a function f at the point $t \in [a, b]$ as the limit as $s \rightarrow t$ of the difference quotient,

$$\frac{f(s) - f(t)}{s - t},$$

assuming this limit exists.

- Note that not all functions have derivatives!
- Those that do have a derivative defined at every point of I we call **differentiable** on I .

Ordinary Differential Operators

- This operation turns out to be a linear operator on the vector space of functions that are differentiable on $I = [a, b]$.
- In fact, we want to consider those functions that can be differentiated as many times as we like
- Thus, we consider the space of infinitely differentiable, real-valued functions defined on the closed real interval $[a, b]$:
 $C^\infty[a, b]$

Ordinary Differential Operators

- The operation of taking the derivative will enable us to develop deterministic differential equations.
- Later we will define derivatives of stochastic processes in order to introduce and solve stochastic differential equations. You cannot always take the derivative (or even integrate) a random process – and sometimes not even the sample functions of a random process – in the ordinary sense.

Ordinary Differential Operators

- Since the derivatives of all orders are assumed to exist for all functions $f \in C^\infty[a, b]$, the limit of the difference quotient always exists for such functions.
- Thus, we assign to each f a new function Df given by

$$(15) \quad [Df](t) = \lim_{s \rightarrow t} \frac{f(s) - f(t)}{s - t}$$

- This assignment defines a transformation

$$D : C^\infty[a, b] \rightarrow C^\infty[a, b]$$

that is a linear operator.

Ordinary Differential Operators

- D is linear: $D[af + bg] = a[Df] + b[Dg]$
- Sometimes we get lax and simply write $Df(t)$ instead of $[Df](t)$. However, keep in mind that $Df(t)$ says "Differentiate f and then evaluate the resulting function at the point t ."
- D obeys all the usual rules of differential calculus such as the product rule and the chain rule:

$$D(fg) = [Df]g + f[Dg]$$

$$D(f \circ g) = [Df] \circ g \cdot Dg$$

Ordinary Differential Operators: Examples

- If id is the identity function, i.e., $\text{id}(t) = t$, then $D\text{id} = 1$. (You probably know this as $\frac{dt}{dt} = 1$; Note that 1 is the function whose value at every t is 1.)
- Using the product rule, we have, for example,

$$\begin{aligned} D\text{id}^2 &= D[\text{id} \cdot \text{id}] \\ &= D\text{id} \cdot \text{id} + \text{id} \cdot D\text{id} \\ &= 1 \cdot \text{id} + \text{id} \cdot 1 \\ &= \text{id} + \text{id} \\ &= 2\text{id} \end{aligned}$$

(In more familiar notation this is just $\frac{dt^2}{dt} = 2t$.)

Ordinary Differential Operators

- Note that we sometimes write d/dt or $\frac{d}{dt}$ for the operator D when we want to emphasize the independent variable t . Or, instead of $\frac{dy}{dt}$ we write simply \dot{y} .
- However, that is really unnecessary, adds no new information, and is just extra baggage.
- Nevertheless, we shall do it sometimes because it may be more familiar.

Ordinary Differential Operators

- Since there is really only one generating operator D that we are interested in here, we are less interested in multiplication than we are in function composition or iteration of this operator. For instance, $D^2 f = [D \circ D](f) = D(Df)$.
- By induction we have $D^n = \underbrace{D \circ \dots \circ D}_n$.
- Just as real-valued functions inherit algebraic operations from those of their range spaces, so too do differential operators:

$$[aD^m + bD^n](f) = a(D^m f) + b(D^n f)$$

Ordinary Differential Operators

- We can therefore generate polynomials in D :

$$p(D) = a_n D^n + \cdots + a_1 D + a_0 I,$$

where I is the identity operator on $C^\infty[a, b]$. We usually just drop the identity operator, since it amounts to multiplying by 1.

- Such polynomials in D are still linear operators.
- Thus, we have rediscovered the group ring $\mathbb{R}(\text{Diff})$, where Diff in this case is the infinite cyclic group generated by the differential operator D . We'll call this group ring the **polynomial ring of differential operators**.

Ordinary Differential Equations

- With polynomials of differential operators we can now define ordinary differential equations.
- An equation of the form

$$p(D)f = q(D)g$$

where $f, g \in C^\infty[a, b]$ and $p(D), q(D) \in \mathbb{R}(\text{Diff})$, is called an ordinary differential equation with constant coefficients.

Ordinary Differential Equations

- It is probably more familiar if we use y instead of f and u instead of g , and if we show the independent variable, say t , explicitly:

$$a_n \frac{d^n y}{dt^n} + \cdots + a_1 \frac{dy}{dt} + a_0 y = b_m \frac{d^m u}{dt^m} + \cdots + b_1 \frac{du}{dt} + b_0 u$$

or

$$(16) \quad \sum_{i=0}^n a_i \frac{d^i y}{dt^i} = \sum_{i=0}^m b_i \frac{d^i u}{dt^i}$$

- For our purposes, the function $u = u(t)$ is called the **input**, while the function $y = y(t)$ is called the **output**. The independent variable t is thought of as time.

Ordinary Differential Equations

- In (16), u is ordinarily assumed to be known or given, while y is unknown and sought.
- Any function y satisfying (16) is called a solution of the differential equation.
- A differential equation such as (16) is called **homogeneous** when the input function u is identically zero for all $t \in [a, b]$.

Ordinary Differential Equations

- Thus far we have only dealt with linear differential equations with constant coefficients.
- We could allow the coefficients to be time-dependent functions. However, for most of our applications, this will not be necessary.

Ordinary Differential Equations: Examples

- From physics we have Hooke's law: the restoring force of a spring is proportional to the distance stretched: $-ky = m\frac{d^2y}{dt^2}$.
- This is just an application of Newton's $F = ma$, where a is the acceleration: $a = d^2y/dt^2$.
- Rewriting this in the form of (16) we have

$$m\frac{d^2y}{dt^2} + ky = 0$$

- Therefore, this is a homogeneous equation of order 2.

Ordinary Differential Equations

- We have used y rather than x as in most physics books, because we want to be consistent with (16).
- It is common to rewrite such equations so that the leading coefficient is 1. Thus we have:

$$\frac{d^2y}{dt^2} + \frac{k}{m}y = 0.$$

- It is easily verified that $\sin \omega_0 t$, where $\omega_0 = \sqrt{\frac{k}{m}}$, satisfies this differential equation.

Ordinary Differential Equations

- Equation (16)

$$\sum_{i=0}^n a_i \frac{d^i y}{dt^i} = \sum_{i=0}^m b_i \frac{d^i u}{dt^i}$$

is called **time-invariant**, because the coefficients are constant and also because the independent variable t does not appear explicitly in the equation.

- Note that (16) is linear in terms of both the input and the output, because the polynomial differential operator on each side of the equation is a linear operator.

Ordinary Differential Equations

- This double linearity means, on the one hand, that the output y due to several inputs acting at the same time is equal to the sum of the outputs due to each input acting alone.
- On the other hand, it also means that if y_1 and y_2 are both solutions to the homogeneous version of (16), then so is $c_1y_1 + c_2y_2$ where c_1 and c_2 are any real constants.
- A maximal set of n such solutions to the homogeneous equation that are also linearly independent (as vectors) is called a **fundamental set**.

Ordinary Differential Equations

- You can find a fundamental set by solving the **characteristic equation** associated with the differential equation. This equation is nothing other than the original polynomial equation in D , but with D replaced by an unknown such as z .

- Thus, the characteristic equation for (16) is

$$a_n z^n + \cdots + a_1 z + a_0 = 0$$

- We use z because the roots of this equation are in general complex (thanks, once again, to Dr. Gauss).

Ordinary Differential Equations

- In solving the characteristic equation, two cases emerge:
 1. The roots of the characteristic equation are all distinct.
 2. Some roots are repeated, i.e., each root r_i has multiplicity n_i .

Ordinary Differential Equations

- In case 1, a fundamental set is easily constructed:

$$y_1(t) = \exp(r_1 t), \dots, y_n(t) = \exp(r_n t)$$

where the r_i are the distinct roots.

- In case 2, each root r_i of multiplicity n_i contributes n_i functions to the fundamental set:

$$y_{i1}(t) = \exp(r_i t), y_{i2}(t) = t \exp(r_i t), \dots, y_{in_i}(t) = t^{n_i-1} \exp(r_i t)$$

Ordinary Differential Equations

- To fully solve the differential equation, the initial conditions need to be taken into account.
- These are the values of the function and its first $n - 1$ derivatives at the start of the time interval.
- We will specialize our interval $[a, b]$ to the case in which $a = 0$ and $b = T$, some unspecified time in the future.

Ordinary Differential Equations

- The **free response** is the solution to the homogeneous equation in which the initial conditions are specified but the input is zero, i.e., there is no forcing function.:

$$(17) \quad y(0), Dy(0), \dots, D^{n-1}y(0)$$

- The **forced response** reverses this situation; it allows the input function to be non-zero, but it requires all the initial conditions in (17) to be zero.
- The **total response** is the sum of the free and forced responses.

Ordinary Differential Equations: Example of Free, Forced and Total Responses

- Consider $D^2y + 3Dy - 1y = u$ with initial conditions: $y(0) = 1, Dy(0) = -1$.
- The roots (found using Matlab's roots function) are $r_1 = -3.3028$ and $r_2 = 0.3028$, and they are distinct.
- Therefore, a fundamental set is $\{\exp(-3.3028t), \exp(0.3028t)\}$.

Ordinary Differential Equations: Example of Free, Forced and Total Responses

- With the fundamental set and the initial conditions we can form a set of n linear equations in n unknowns (here, $n = 2$):

$$\begin{aligned}c_1 y_1(0) + c_2 y_2(0) &= y(0) \\c_1 D y_1(0) + c_2 D y_2(0) &= D y(0)\end{aligned}$$

or

$$\begin{aligned}1 \cdot c_1 + 1 \cdot c_2 &= 0 \\-3.3028 \cdot c_1 + 0.3028 \cdot c_2 &= -1\end{aligned}$$

Ordinary Differential Equations

- The previous linear system is of the form $Ax = b$. Using Matlab's matrix division operator \backslash , the solution is $x = A \backslash b$ or

$$c_1 = 0.2773$$

$$c_2 = -0.2773$$

Ordinary Differential Equations

- The total response can also be decomposed into transient and steady state responses
- The **transient** response is that part of the total response that decays to 0 as $t \rightarrow \infty$.
- The **steady state response** is the difference between the total response and the transient response.

Linear Time-Invariant Differential Equations

- Linear Time-Invariant (LTI) Differential Equations

- Linear: Polynomial differential operators are linear:

$$p(D)[af + bg] = ap(D)f + bp(D)g$$

- Time-Invariant means the coefficients are constant.
- Homogeneous solution of corresponding state equation is a matrix exponential. (More on that later when we discuss the state transition matrix.)

Ordinary Difference Equations

- Auto-Regressive Moving Average (ARMA)
- n^{th} derivative corresponds to n -step advance:

$$(18) \quad \sum_{i=0}^n a_{n-i} y(k+n-i) = \sum_{i=0}^m b_{m-i} u(k+m-i)$$

Ordinary Difference Equations

- The left side of Eq. (18) is the Auto-Regressive part
- The right side of Eq. (18) is the Moving Average part
- Hence, ARMA

Ordinary Difference Equations: Example

- Let $n = 3$, $m = 2$ and consider the difference equation:

$$\begin{aligned} & -3y(k+3) + 1.5y(k+2) \\ & \quad + 7y(k+1) - 3.2y(k) \\ & \qquad \qquad \qquad = 23u(k+2) - 5u(k+1) + 0.5u(k) \end{aligned}$$

Day 1, Segment 4
State Space Dynamics
Topics

- Differential Equations and Dynamic Systems
- Block Diagrams and State Space
- State Transition Matrix

Dynamic Systems

- The time-dependent behavior of physical systems can often be modeled by differential equations.
- Such systems are often called **dynamic systems** or **dynamical systems**.

Dynamic Systems

- Considering Newton's law $F = ma$ once again, we see that the force F is the resultant force acting on the mass m .
- However, the resultant force is simply the vector sum of all the forces, both those internal to the system as well as the input forces that are external.
- The restoring force, $-ky$ in the simple oscillator problem, is a force internal to the system. So we don't consider it an input *per se*.

Dynamic Systems

- The response or output of the system is motion, more specifically, accelerated motion given by a .
- Thus, the simple harmonic oscillator is a dynamic system, one in which there is only an internal force determining system behavior: $F = -ky$.
- An external forcing function may also be present, e.g.:

$$\frac{d^2y}{dt^2} + \frac{k}{m}y = \sin \omega t$$

Dynamic Systems

- The most general second-order equation for a mechanical system of this sort according to Newton's law is:

$$(19) \quad \frac{d^2y}{dt^2} = \frac{1}{m}F(y, \dot{y}, t)$$

- If we can separate the forcing function $F(y, \dot{y}, t)$ into a sum of two functions $F_1(y, \dot{y})$ and $F_2(t)$, then we can rearrange this equation into the input-output form of equation (16):

$$\frac{d^2y}{dt^2} - \frac{1}{m}F_1(y, \dot{y}) = \frac{1}{m}F_2(t)$$

Dynamic Systems

- In engineering applications, we are often interested not only in modeling the dynamics of the system, but also in measuring the output or behavior of the system.
- But the output is not always directly observable.
- Thus, in specifying dynamic systems, we often add a second equation, usually algebraic or functional in nature, that relates the output y (and possibly the input u) to some measured or observed quantities.

$$z = h(y) + g(u)$$

Dynamic Systems: The Concept of State

- In keeping with the linear spirit of things, we will restrict ourselves for now to linear functions h .
- Also, we will now introduce the state space concept and define z as a function of the state.
- Intuitively, the concept of state involves in some sense the instantaneous internal arrangement of a system. In practical situations, the state variables may represent attributes of the system that we wish to know something about.

Dynamic Systems: The Concept of State

- Mechanical systems like a moving particle have three simultaneous attributes: position, velocity, and acceleration.
- Newton tells us that acceleration is often a function of position, velocity and perhaps time.
- However, as we just saw, a force that acts on a system as a function of time only should be considered an external forcing function, i.e., an input to the system.
- Thus we are led to think of position and velocity as the primary state variables of interest.

State Space Representation

- A general n -th order linear differential equation can be expressed as a system of first-order differential equations.
- Ignoring the input u for the moment, consider:

$$p(D)y = D^n y + a_{n-1}D^{n-1}y + \cdots + a_1 Dy + a_0 y,$$

where we have put the polynomial in normal form, i.e., such that its leading coefficient is 1.

State Space Representation

- If we consider y and its first $n - 1$ derivatives as the states of this system, then we have n first order equations. Setting $x_1 = y$, we write

$$x_1(t) = y(t)$$

$$x_2(t) = Dy(t)$$

...

$$x_n(t) = D^{n-1}y(t).$$

- The functions $x_i(t)$ are called the **state variables**.

State Space Representation

- Taking the derivative of these n state variables and making use of the original differential equation yields:

$$Dx_1(t) = x_2(t)$$

$$Dx_2(t) = x_3(t)$$

...

$$Dx_n(t) = -a_0x_1(t) - a_1x_2(t) - \dots - a_{n-1}x_n(t).$$

- These n equations can be written more compactly in vector matrix form: $Dx(t) = Ax(t)$ or $\dot{x}(t) = Ax(t)$.

State Space Representation

- In the equation $\dot{x}(t) = Ax(t)$, we have:

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix}$$

and

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & 0 & 1 \\ -a_0 & -a_1 & \cdots & -a_{n-2} & -a_{n-1} \end{bmatrix},$$

the **companion matrix** of the differential equation.

State Space Representation

- If there is an input function u and it is present via its derivatives as in equation (16), then we can write:

$$(20) \quad \dot{x} = Ax + Bu$$

forming B in much the same way we formed A , and suppressing the time variable t .

- Equation (20) expresses the system's **state dynamics**.
- Later we will include white noise as an input. However, that will make the differential equations stochastic, and we are not yet ready for that.

State Space Representation

- To obtain the original output y as a function of the internal state of the system, we require one additional vector-matrix equation:

$$(21) \quad y = Cx + Du$$

where the matrix D represents a **direct feedthrough** of the input to the output.

- The output equation is derived directly from the b_i coefficients in the original matrix differential equation.

State Space Representation

- Kalman filter theory generally assumes $D = 0$ in the output equation and uses a measurement model instead:

$$(22) \quad z = Hx$$

- Note that H implicitly includes C as a factor.
- Note that we have not placed any restrictions on the size of the measurement column vector z . As long as H has n columns it can have as many rows as desired.

State Space Representation: Example

- Let's consider once again the simple harmonic oscillator:

$$m \frac{d^2 y}{dt^2} + ky = 0$$

and convert this to state space form:

$$x_1 = y$$

$$x_2 = Dy$$

and $\dot{x}_2 = -\frac{k}{m}x_1 + 0 \cdot x_2$.

- Therefore, in matrix and vector form:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k/m & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

- For a measurement model, we could consider a camera with a strobe flash that periodically records the instantaneous position of the sliding mass.
- In this case, only the position would be observable and we would write the measurement or observation model as:

$$z = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

(Remember that we are ignoring both process and measurement noise.)

State Space Representation: Matlab / Simulink Example

- Four parameters A, B, C, D from Matlab workspace
- Measurement model added external to Simulink's state space block

State Space Representation

- State space representation not derived only from single n^{th} order differential equation
- Sometimes state coupling comes more directly through internal dynamic coupling. E.g.,
 - coupled harmonic oscillators
 - complicated RLC networks

Ordinary Difference Equations

- For difference equations, a state space model can be derived using the companion matrix exactly as for differential equations
- Matlab / Simulink has a subsystem block in its Discrete block library in which the four matrices for an input-output model can be specified in the workspace.

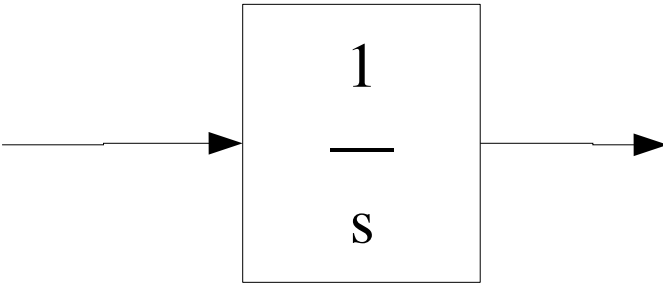
State Space Representation: Summary

- The state variables are often provided by the 0^{th} through $n - 1^{\text{st}}$ derivatives of a single n^{th} plus internal dynamic coupling
- Equivalently, they are the outputs of the n integrators of the system.
- They represent implicit or explicit internal coupling of the system.
- Similarity transformation of state vector yields equivalent representation. Changing coordinate systems.

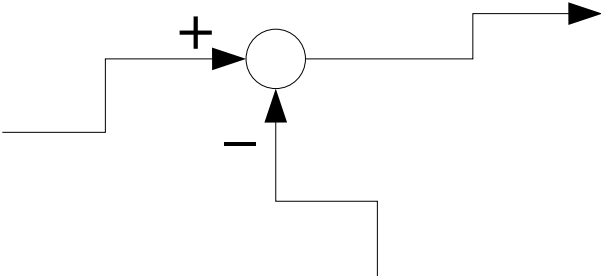
Block Diagrams depict ...

- System signal flow
 - summation and takeoff points
- Various transformation blocks
 - integrators (continuous) and delayors (discrete)
 - differentiators
 - scalors or gain multipliers

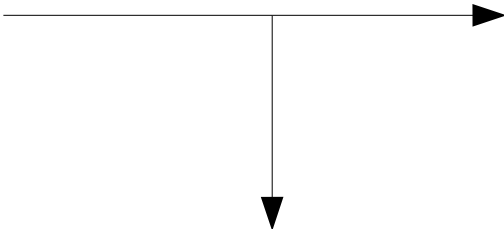
Integration Block



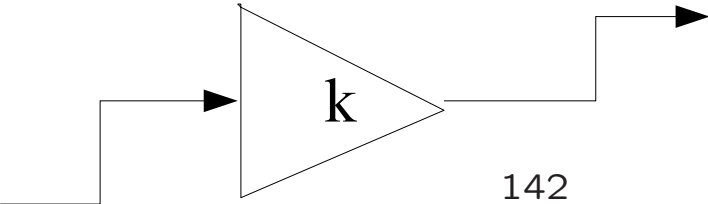
Summation Point

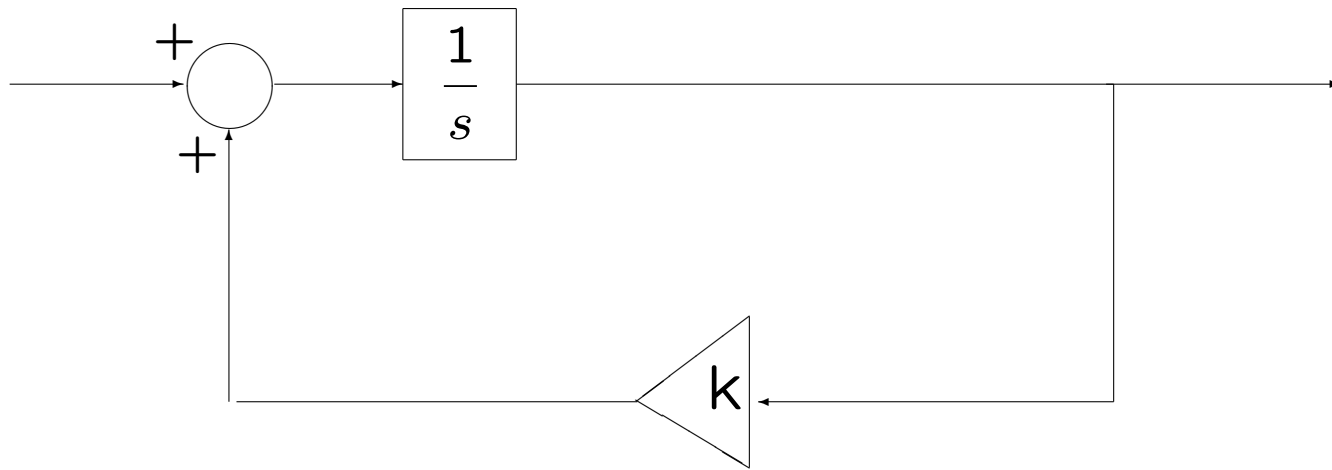


Takeoff Point



Scalar or Gain Multiplier





Deriving States from Diff. Eqs. and Block Diagrams

- Continuous states can be identified as outputs of integrators
- Discrete states can be identified either as sampled continuous states or as outputs of delayors

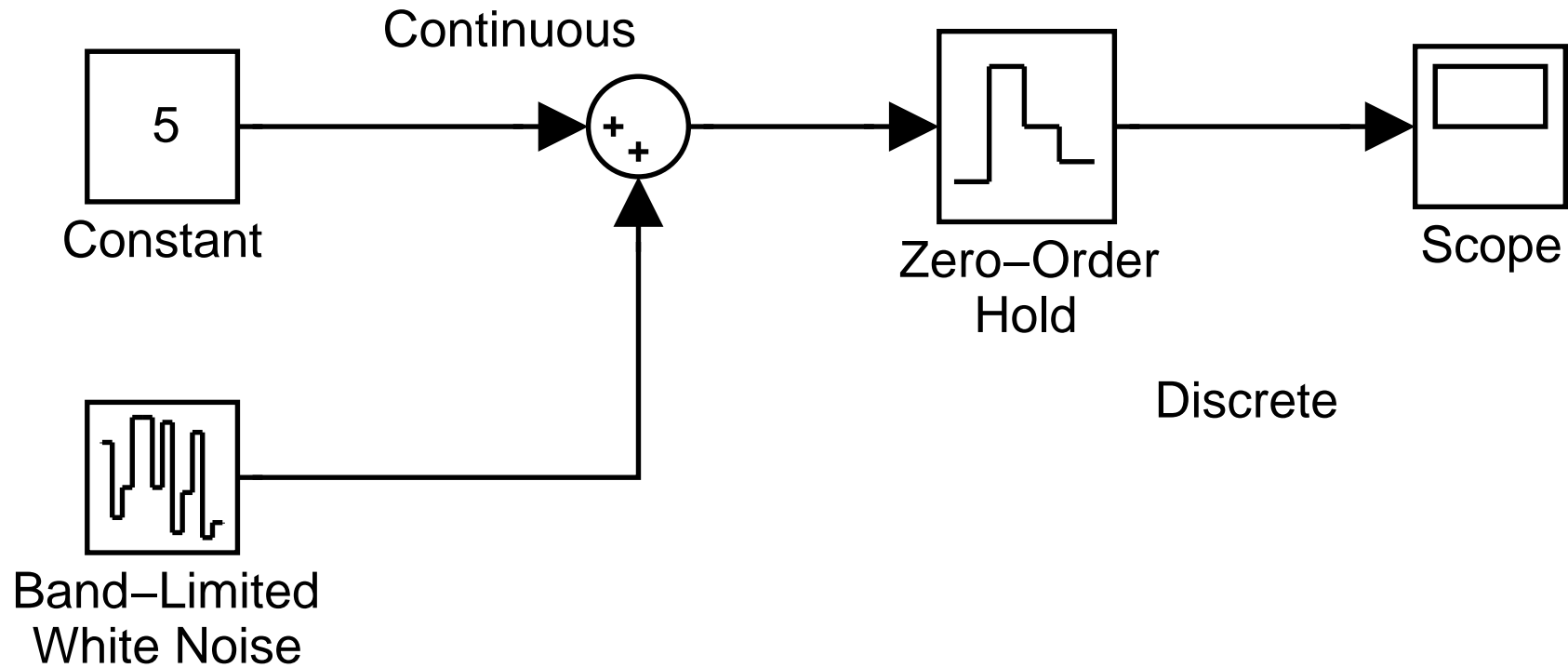
Examples

- Scalar Constant Dynamics
- Scalar Constant Dynamics with Recursive Estimator
- Scalar Ramp Dynamics with Recursive Estimator
- Damped Harmonic Oscillator

Scalar Constant Dynamics

- Matlab Model
- Very Simple
- Additive Noise
- No feedback loops

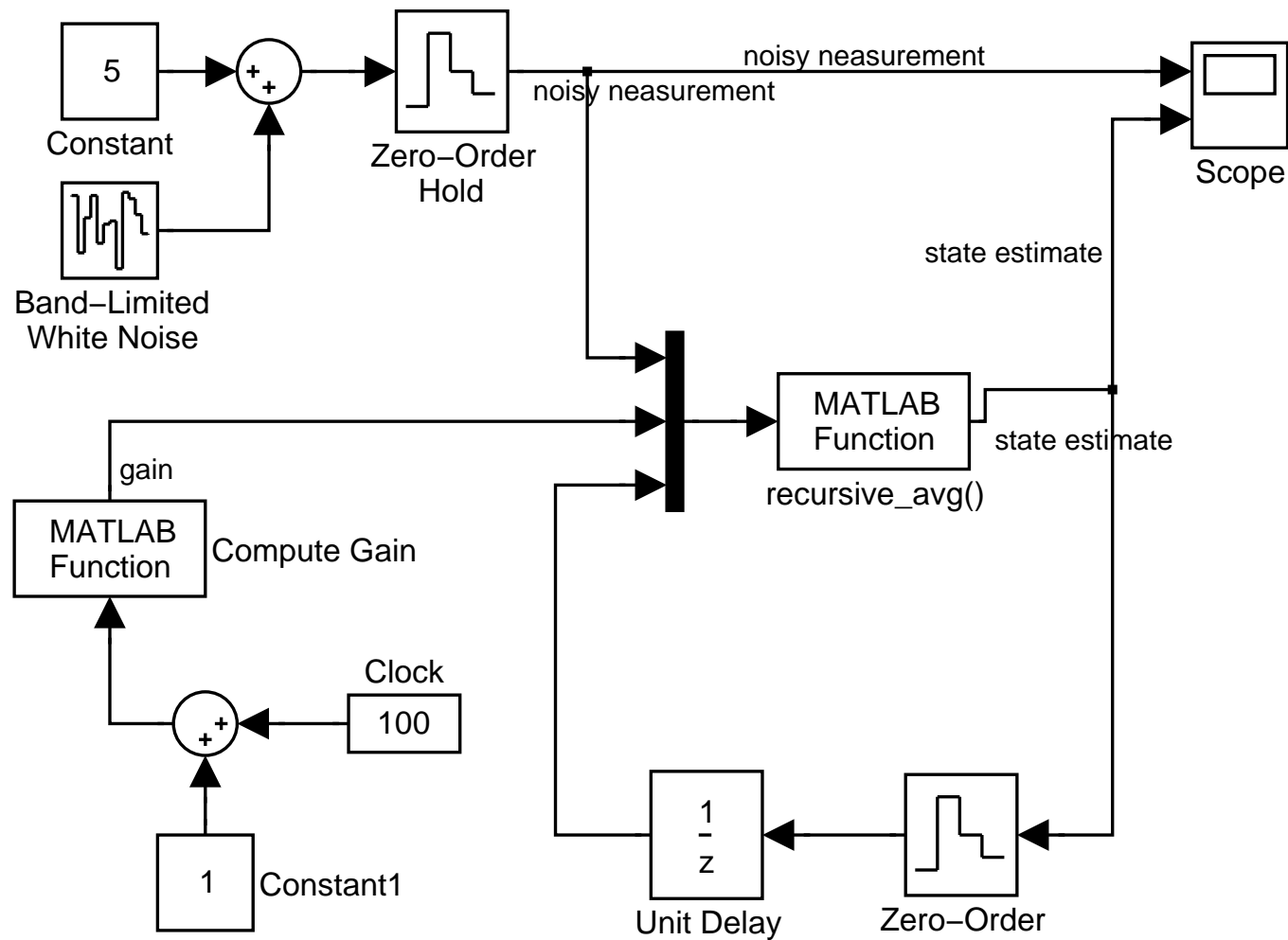
Scalar, Constant Dynamics plus White Process Noise



Scalar Constant Dynamics with Recursive Estimator

- Matlab Model
- Very Simple Dynamics
- Additive Noise
- Estimation loop implementing recursive simple averaging
- Simple open loop gain input

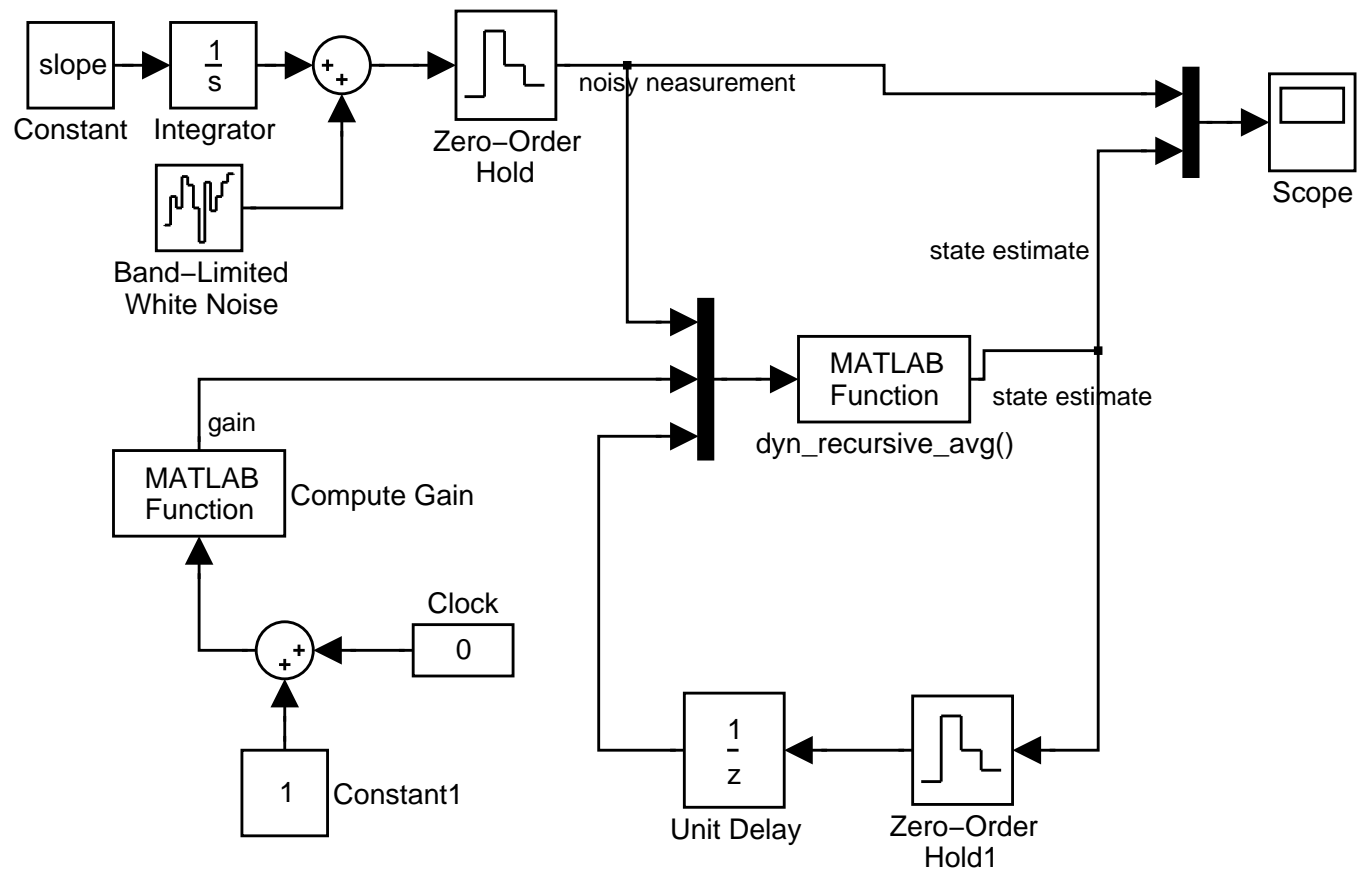
Scalar, Constant Dynamics plus White Process Noise with Estimation



Scalar Ramp Dynamics with Recursive Estimator

- Matlab Model
- Scalar Dynamics with known ramp
- Additive Noise
- Estimation loop implementing recursive simple averaging
- Simple open loop gain input

Scalar, Ramp Dynamics plus White Process Noise with Dynamic Average Estimation



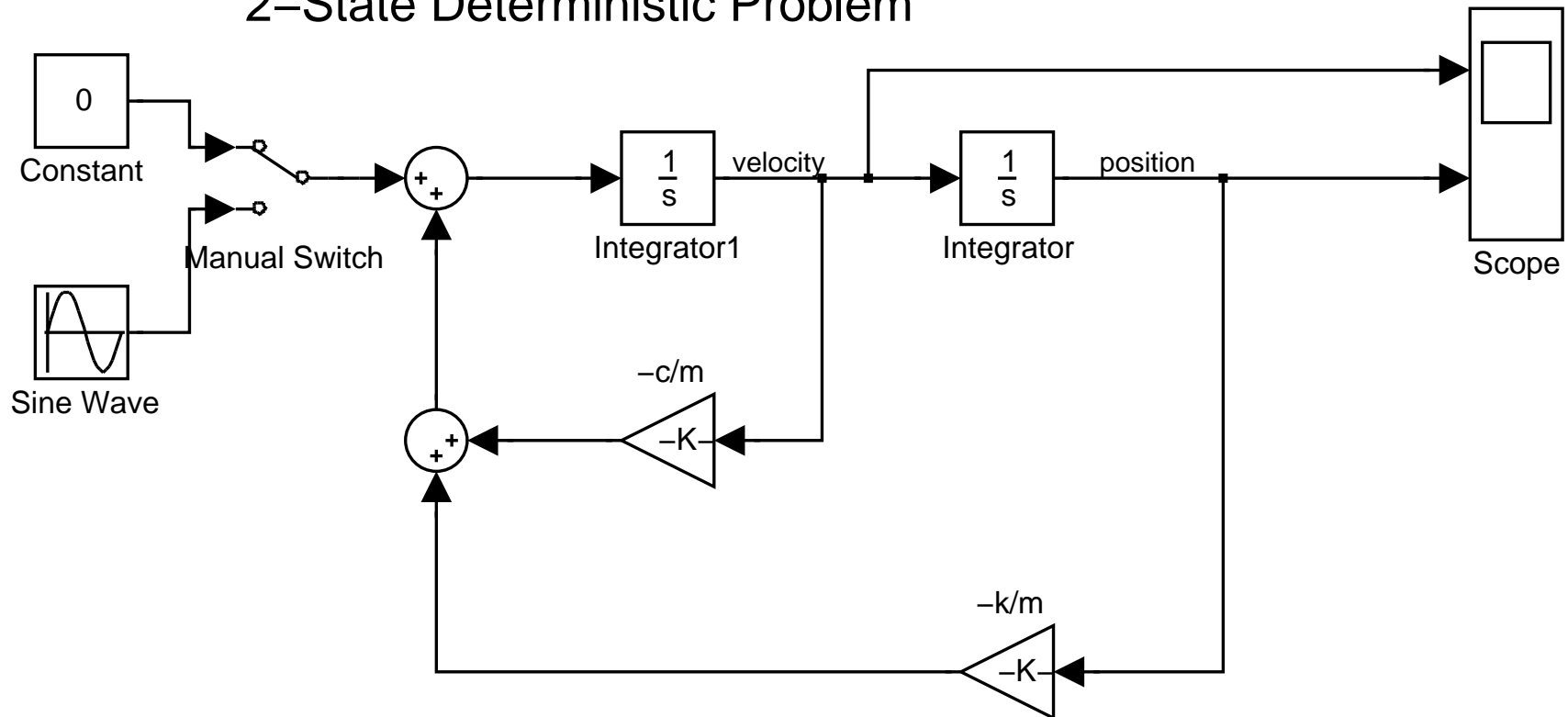
Observe how the sampling rate affects the ability of the filter to follow the input.

Vector Dynamics: Damped Harmonic Oscillator

- Matlab Model
- 2-dimensional Vector Dynamics
- Deterministic, Continuous Model

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k/m & -c/m \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Damped Harmonic Oscillator 2-State Deterministic Problem



Vector Dynamics: Damped Harmonic Oscillator

- Discrete model would look like this

$$\begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} = \exp \left(\begin{bmatrix} 0 & 1 \\ -k/m & -c/m \end{bmatrix} \Delta t \right) \begin{bmatrix} x_1(k-1) \\ x_2(k-1) \end{bmatrix}$$

- Here $\Delta t = t_k - t_{k-1}$
- Note that exp is a matrix exponential.
- In the continuous case, the state transition matrix is a matrix exponential whenever the system dynamics matrix is a constant (i.e., time invariant) matrix.

State Transition Matrix

- $\Phi_k = \Phi(k, k - 1)$
- Solution of homogeneous equation for continuous systems
- If discrete system is derived from continuous system, then Φ_k satisfies:

$$x_k = \Phi_k x_{k-1}$$

Exercises

1. Convert the differential equation

$$5\frac{d^3y}{dt^3} - 0.3\frac{dy}{dt} + 7y = u(t)$$

into a state space system model

2. Convert the difference equation

$$100y(k + 9) = \sum_{i=0}^2 b_i u(k + 2 - i)$$

3. Write the output equations for the above systems as well!