Introduction and Deterministic System Models

Dr. Randall Christensen

August 5, 2020

1 Introduction

Chapter 1 of [1] contains an excellent introduction and motivation to the concept of the Kalman filter. You should re-read this chapter when you are feeling confused or need to re-acquire the 'big picture' of what we are accomplishing during this class. Some important concepts from section 1.3 are:

1. The Kalman filter (KF) is simply an optimal, recursive data processing algorithm
2. Optimal can means many things. A high level view of optimality is that the KF incorporates ***all*** information provided to it, i.e. the KF processes all available measurements, regardless of their precision, to estimate the current state of the system
3. Recursive means that the KF doesn't require all previous data to be kept in storage and reprocessed every time a new measurement is taken. This helps immensely with storage and computation requirements, enabling the KF to be used on resource-constrained hardware (e.g. space hardware).

The bottom line is that the KF combines all available measurement data, plus prior knowledge about the system and measuring devices, to produce an estimate of the desired variables in such a manner that the error is minimized statistically.

The assumptions of a KF appear limiting at first, but in practice are very often valid:

1. **Linear system model:** While most interesting problems are nonlinear in nature, the system dynamics and measurement models can be linearized about a carefully chosen 'operating point'. Within a region near the operating point, the dynamics and measurement models are approximately linear. The size of the region is problem-dependent.
2. **White process and measurement noise:** White implies that a sample of a random process at time t_k is completely uncorrelated with (i.e. tells you nothing about) a sample at time t_{k+1}. While a white noise source does not exist, it is a good approximation for realistic systems that are band-limited (e.g. a rotational or translational dynamics of rigid bodies) or sensors with typical sampling times (e.g. 1Hz GPS measurements, 1000Hz Fiber Optic Gyro Measurements)
3. **Gaussian process and measurement noise:** The distribution of a Gaussian random variable is specified by two moments, the first (mean) and the second (variance). In many cases, this is all the information that an engineer has about a random variable. Furthermore, the central limit theorem guarantees that the distribution tends toward Gaussian with a large number of noise sources increases, which is typically the case.

Section 1.5 of [1] contains a simple, yet excellent example of what a KF does. The example is of two navigators lost at sea, one of which is highly-trained and the other not. The untrained navigator takes a position measurement at t_1 with mean and variance (See equation 1-1 and 1-2 of [1])

$$\hat{x}(t_1) = \tag{1}$$

$$\sigma_x^2(t_1) = \tag{2}$$

The corresponding ***conditional*** distribution of position, $f_{x|z_1(x(t_1)|z_1)}$, is a Gaussian with a peak at $\hat{x}$ and a width determined by σ_x (Draw the distribution after the first measurement. Should it be wide or narrow?).

Now suppose the trained navigator takes a measurement, immediately after, i.e. $t_1 \approx t_2$, such that the boat did not move in between measurements. What is the mean and covariance of this measurement?

$$\hat{x}(t_2) = \tag{3}$$

$$\sigma(t_2) = \tag{4}$$

(Draw the conditional distribution of position given the second measurement and label it) Is the conditional distribution wide or narrow? Where would the mean be?

At this point, you have two measurements or observations of the same quantity, $x(t_1)$. Which measurement would you choose? Why? Is there a way to combine the information, i.e. to exploit all available information? We will show later in the course that the conditional density given both z_1 and z_2 is expressed as (see equations 1-3 and 1-4 of [1])

$$\mu = \tag{5}$$

$$1/\sigma^2 = \tag{6}$$

Does the form of 5 and 6 make sense? What would happen to μ if the first navigator was *really* poor? What about σ? What if both navigators were equally skilled? It is important to point out that σ^2 is less than σ_{x_2} even when σ_{x_1} is very large. In other words, even poor data provides some information. (Draw the resulting density in the figure above). Given this density, what is the best estimate of your position, conditioned on both the measurements? (See equation 1-5 of [1])

$$\hat{x}(t_2) = \tag{7}$$

Equation 5 can also be written in the predictor/corrector form (Derive this yourself following the development in equation 1-7 of [1])

$$\hat{x}(t_2) = \tag{8}$$

where K is called the "Kalman gain" and is equal to (see equation 1-8 of [1])

$$K(t_2) = \tag{9}$$

Why would we call equation 8 a predictor/corrector form? What is the best prediction of your position, prior to the second measurement? What is the correction term proportional to? (Label the prediction term and the correction term) Does the form of K agree with your intuition on the relative accuracy of each measurement?

Likewise, equation 6 can also be expressed in terms of the Kalman gain. (See equation 1-9 of [1]). Verify that the form agrees with intuition.

$$\sigma_x^2(t_2) = \tag{10}$$

Equations 8 and 10 define a density that embodies all the information available at $t_1 \approx t_2$. But what if $t_2 > t_1$ such that the boat is able to move. We describe the motion of the both via a differential equation, driven by a random variable, i.e. a random *process*. (See equation 1-10 of [1])

$$\dot{x} \equiv \frac{dx}{dt} = \tag{11}$$

where u is the *constant* nominal velocity of the boat, and w is the noise that drives the differential equation, or "process noise". What might w represent? We will model w as zero mean with a variance of σ_w^2.

Starting with an estimate of your position after the second measurement $\hat{x}(t_2)$, how might you propagate to some future time t_3, knowing that your best estimate of the process noise is zero? Given the presence of the process noise, how might the distribution of your position estimate change over time? (Draw the distribution at t_2, some time t, and at just prior to the next measurement, t_3^-). What would the distribution look like if $t_3 \gg t_2$?

How would you mathematically express your prediction (i.e. "apriori estimate) of position at t_3 and the associated variance? (See equation 1-11 and 1-12 of [1])

$$\hat{x}\left(t_3^-\right) = \tag{12}$$

$$\sigma_x^2\left(t_3^-\right) = \tag{13}$$

Now suppose you make another measurement at t_3. How might you combine that measurement with the your prediction to generate the "best" estimate of your position? What would be the variance of your estimate? (See equaions 1-13 to 1-15 of [1])

$$\hat{x}\left(t_3\right) = \tag{14}$$

$$\sigma\left(t_3\right) = \tag{15}$$

where

$$K\left(t_3\right) = \tag{16}$$

Given the form of equations 12 to 15, answer the following questions

- What is the best estimate of position if the measurement at t_3 is very poor? Very accurate?
- What happens to the apriori variance when the process noise variance is very large?
- What happens to the Kalman gain at t_3 when the process noise is large?
- What is the resulting best estimate of position when process noise is large?

Notwithstanding the complexity of the underlying theory, the Kalman filter is simply an algorithm which repeatedly performs these two steps of state/variance propagation (equations 12 and 13), followed by a measurement update (equations 14 and 15). In this class, you will learn the theory behind the Kalman filter, its extension to nonlinear systems, the Extended Kalman filter, and implement a Monte Carlo simulation to validate correction operation. **The theory you learn and the skills you obtain are broadly applicable to the aerospace field.**

2 Math Preliminaries

This section contains multiple math properties that I have found useful in this class. You are also encouraged to ready Appendix A of [1] which contains addition background material.

Distribution of a transpose

$$(AB)^T = B^T A^T \tag{17}$$

$$(ABC)^T = C^T B^T A^T \tag{18}$$

Commutative Property of Dot Product

$$\underline{\mathrm{a}} \cdot \underline{\mathrm{b}} = (\underline{\mathrm{a}})^T \underline{\mathrm{b}} = (\underline{\mathrm{b}})^T \underline{\mathrm{a}} \tag{19}$$

Anti-commutative Property of Cross Product

$$\underline{\mathrm{a}} \times \underline{\mathrm{b}} = -\underline{\mathrm{b}} \times \underline{\mathrm{a}} \tag{20}$$

Dot product

$$\underline{\mathrm{V}} \cdot \underline{\mathrm{W}} = VW \cos\phi \tag{21}$$

Cross product

$$\underline{\mathrm{V}} \times \underline{\mathrm{W}} = \underline{\mathrm{n}} VW \sin\phi \tag{22}$$

where $\underline{\mathrm{n}}$ is a unit vector (magnitude of 1) perpendicular to $\underline{\mathrm{V}}$ and $\underline{\mathrm{W}}$ whose positive direction is defined by the right hand rule that curls the fingers of the right hand from $\underline{\mathrm{V}}$ to $\underline{\mathrm{W}}$, with a positive $\underline{\mathrm{n}}$ thereby provided in the thumb pointing direction.

Vector magnitude/norm

$$\|\underline{\mathrm{x}}\| = \sqrt{\underline{\mathrm{x}}^T \underline{\mathrm{x}}} \tag{23}$$

Cross Product Matrix

$$\underline{\omega} = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{24}$$

$$\underline{\omega} \times \underline{\mathrm{r}} = (\underline{\omega}\times)\,\underline{\mathrm{r}} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \underline{\mathrm{r}} \tag{25}$$

Dot product expressed as a transpose

$$\underline{\mathrm{a}} \cdot \underline{\mathrm{b}} = \underline{\mathrm{a}}^T \underline{\mathrm{b}} \tag{26}$$

Perpendicular Component of a Vector

$$\underline{\mathrm{V}}^A_\perp = \underline{\mathrm{V}}^A - \left(\underline{\mathrm{V}}^A \cdot \underline{\mathrm{u}}^A\right) \underline{\mathrm{u}}^A \tag{27}$$

The unit vector $\underline{\mathrm{u}}_b$ has the same direction of $\underline{\mathrm{b}}$ and a magnitude of 1.

$$\underline{\mathrm{u}}_b = \frac{\underline{\mathrm{b}}}{|\underline{\mathrm{b}}|} \tag{28}$$

The projection of $\underline{\mathrm{a}}$ onto a line defined by $\underline{\mathrm{u}}_b$ is

$$\underline{\mathrm{a}} = \underline{\mathrm{a}} \cdot \underline{\mathrm{u}}_b \tag{29}$$

Using dot products and projections of a vector $\underline{\mathrm{a}}$, we can express $\underline{\mathrm{a}}$ in any frame (e.g. the b frame)

$$\underline{\mathrm{a}} = (\underline{\mathrm{a}} \cdot \underline{\mathrm{u}}_{xb})\,\underline{\mathrm{u}}_{xb} + \left(\underline{\mathrm{a}} \cdot \underline{\mathrm{u}}_{yb}\right) \underline{\mathrm{u}}_{yb} + (\underline{\mathrm{a}} \cdot \underline{\mathrm{u}}_{zb})\,\underline{\mathrm{u}}_{zb} \tag{30}$$

A time derivative is denoted using "dot" notation, for example

$$\frac{d\underline{\mathrm{x}}}{dt} = \dot{\underline{\mathrm{x}}} \tag{31}$$

The multiplication rule applies to vectors and matrices, just like it does to scalars, but order is important!

$$\frac{d}{dt}\left(A\left(t\right)\underline{\mathrm{x}}\left(t\right)\right) = \dot{A}\underline{\mathrm{x}} + A\dot{\underline{\mathrm{x}}} \tag{32}$$

The chain rule is an important tool when evaluating derivatives of implicit functions

$$\frac{\partial f}{\partial \underline{\mathrm{x}}} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial \underline{\mathrm{x}}} \tag{33}$$

where

$$f = g(\underline{x}) \tag{34}$$

Example of chain rule

$$f = \|\underline{x}\| = \sqrt{\underline{x}^T \underline{x}} \tag{35}$$

$$g(\underline{x}) = \tag{36}$$

$$f(g(\underline{x})) = \tag{37}$$

Partial derivative of a scalar with respect to a vector

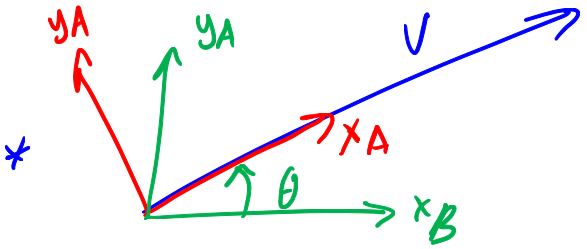
$$\frac{\partial f}{\partial \underline{x}} = \tag{38}$$

Partial derivative of a vector with respect to a vector

$$\frac{\partial \underline{f}}{\partial \underline{x}} = \tag{39}$$

3 Vectors and Coordinate Systems

A vector is a geometric object, independent of a coordinate system/frame of reference. The numerical values we use to describe a vector are define by the frame we express it in. (Draw a picture of a vector, choose a frame, and express the numerical values of that vector. Draw another choice of frame and express the vector in this new frame. Did the vector change?)



$$V^A = [1 \quad 0 \quad 0]^T$$

$$V^B = [\cos\theta \quad \sin\theta \quad 0]^T$$

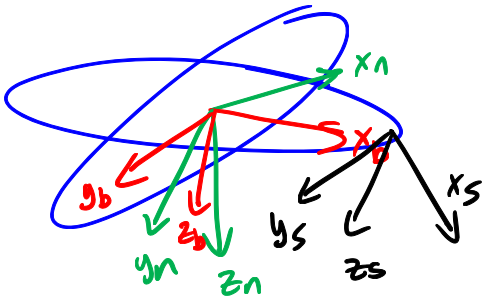
The vector label notation that we will use in this class is based on [2], section 2.1, equation 3.1-1)

$$\underline{v}^a = v_{xa}\underline{u}_{xa} + v_{ya}\underline{u}_{ya} + v_{za}\underline{u}_{za} \tag{40}$$

The choice of coordinate systems is very problem-specific. In some problems, it is common to have as many as 10 coordinate systems! For UAV applications, a typical choice is:

- **b** frame: attached to the body of the aircraft (i.e. moves and rotates with the aircraft)
- **n** frame: same origin as the B frame, but X points north, Y points east, and Z down
- **i** frame: same orientation as the NED frame, but does not move (i.e. attached to the earth)
- **s** frame: a frame similar to the **b** frame, but attached to the sensor.

Draw each of these coordinate systems.



4 Coordinate Systems and Attitude Representations

What is attitude? It is the orientation of one coordinate system with respect to another. (Draw a picture)

There are many ways to represent attitude. Four of the most common representations are included in this section. We begin with a discussion on how a direction cosine matrix is defined, then describe its relationship with three other useful representations.

4.1 Direction Cosine Matrix (DCM) or Transformation Matrix

Equations 3.1-1 through 3.1-11 of [2] define the direction the cosine matrix using the dot-products of the basis vectors for a pair of coordinate systems. See equation 3.1-6 of [2] for an example. In this form, the components of

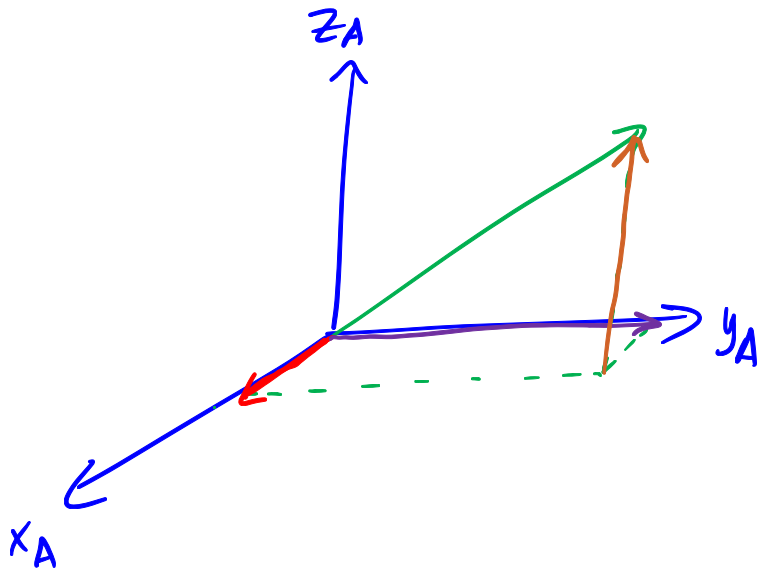
a vector expressed in the A frame are projected onto the B frame. Savage combines equations 3.1-20 with 3.1-12 to yield a more intuitive form of the direction cosine matrix

$$C_B^A = \begin{bmatrix} \left(\underline{\mathbf{u}}_{XA}^B\right)^T \\ \left(\underline{\mathbf{u}}_{YA}^B\right)^T \\ \left(\underline{\mathbf{u}}_{ZA}^B\right)^T \end{bmatrix} \tag{41}$$

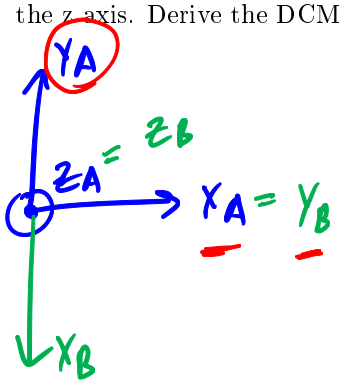
In this form, the direction cosine matrix can be thought of as the matrix which "transforms" a vector in the B frame to the A frame via a series of dot products. This can be illustrated by left-hand multiplying a vector in the B frame, V^B, by C_B^A. (Show that each row of C_B^A is dotted with $\underline{V}^B$. Draw a picture illustrating a vector being projected onto two different coordinate systems)

$$\begin{bmatrix} V_{XA} \\ V_{YA} \\ V_{ZA} \end{bmatrix} = \begin{bmatrix} \left(\underline{\mathbf{u}}_{XA}^B\right)^T \\ \left(\underline{\mathbf{u}}_{YA}^B\right)^T \\ \left(\underline{\mathbf{u}}_{ZA}^B\right)^T \end{bmatrix} \begin{bmatrix} V_{XB} \\ V_{YB} \\ V_{ZB} \end{bmatrix} = \begin{bmatrix} \left(\underline{u}_{XA}^B\right)^T \underline{V}^B \\ \left(\underline{u}_{YA}^B\right)^T \underline{V}^B \\ \left(\underline{u}_{ZA}^B\right)^T \underline{V}^B \end{bmatrix} \tag{42}$$

$$V^A = C_B^A \, V^B$$

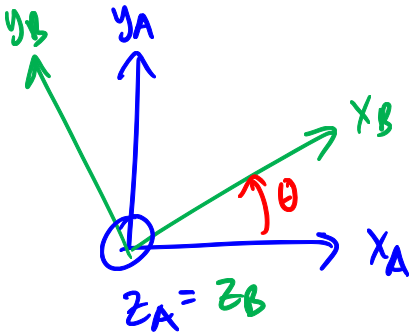


Example 1: Draw two coordinate systems, with coordinate system A rotated from frame B by 270 degrees about the z axis. Derive the DCM from frame B to A. ~~Derive the DCM from A to B.~~



$$C_B^A = \begin{bmatrix} (u_{xA}^B)^T \\ (u_{yA}^B)^T \\ (u_{zA}^B)^T \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Example 2: Draw two coordinate systems, with coordinate system A rotated from frame B by an angle θ about the $\underline{u}_{ZA}^B$ axis. Derive the DCM from frame B to A. Derive the DCM from A to B.



Important properties of DCMs are listed in [2]. These include the following:

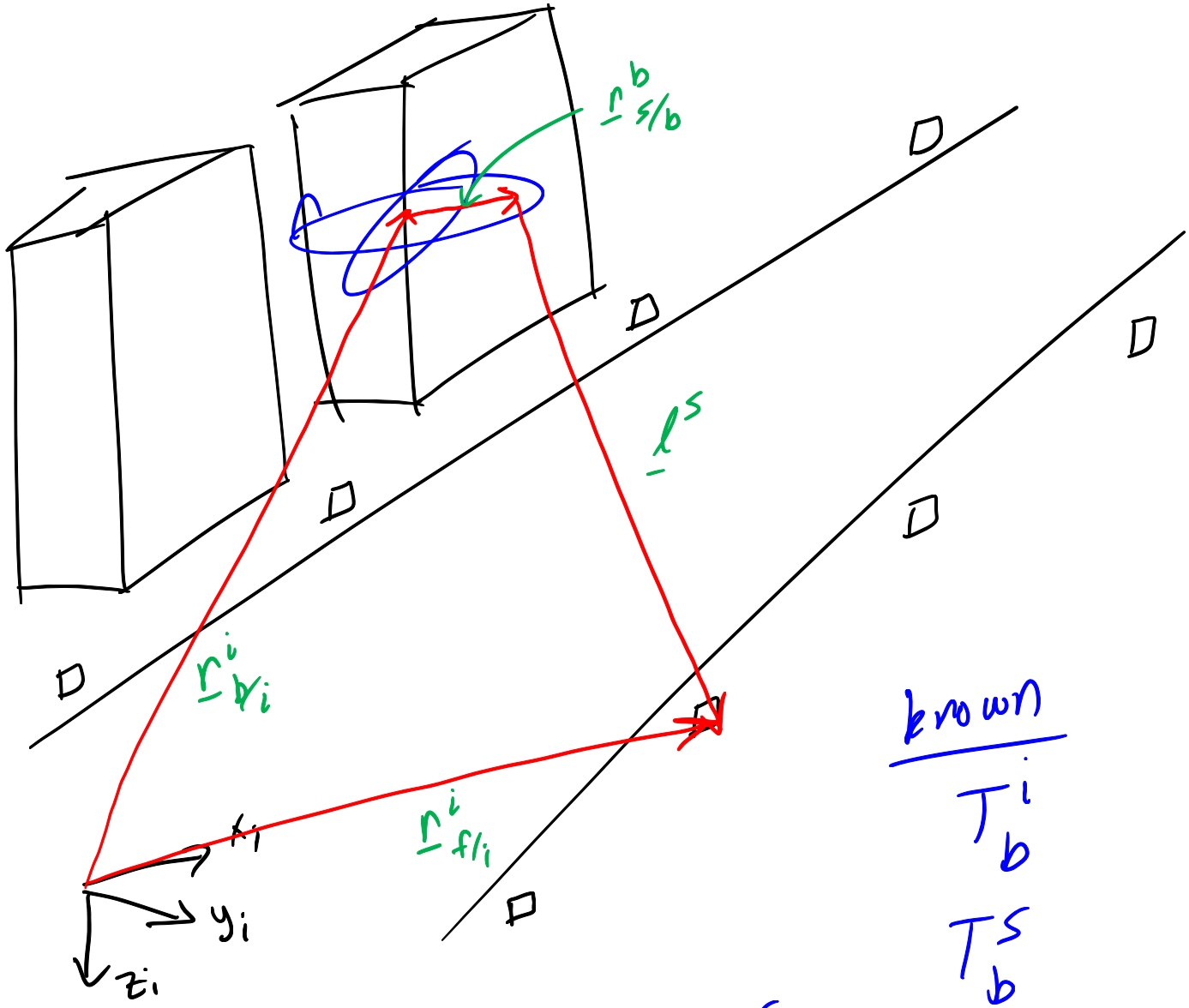
- Ability to transform vectors from one frame to another (equation 3.2.1-2) $\underline{v}^A = C_B^A \underline{v}^B$
- Transpose is equal to its inverse (equation 3.2.1-4) $(C_B^A)^{-1} = (C_B^A)^T = C_A^B$
- Transpose changes the direction of the transformation (equation 3.2.1-3)
- DCM can be cascaded (equation 3.2.1-5) $v^D = C_C^D C_B^C C_A^B v^A$
- A DCM distributes through a cross product (equation 3.1.1-42) $v^B \times u^B = C_A^B (v^A \times u^A)$
- A transformation (not unique) can be solved for given a vector in two coordinate frames (see section 3.2.1.1)
- Rows (and columns) are unit norm and mutually orthogonal

In using a DCM in a simulation or flight code, it is important to maintain the "ortho-normality" of a DCM. Orthogonal refers to the fact that dot product of any two rows (or columns) is always zero. Normal refers to the fact that the magnitude of any row (or column) is always one. One method for "ortho-normalizing" a DCM is the following two step process of computing the ortho-normality error, E_{SYM}, then using it to remove the errors (see section 7.1.1.3 of [2]).

$$E_{SYM} = \frac{1}{2}\left[C_B^A \left(C_B^A\right)^T - I_{3\times 3}\right] \tag{43}$$

$$C_{B,corrected}^A = \left(I_{3\times 3} - E_{SYM}\right) C_B^A \tag{44}$$

Example usage: Draw a UAV flying over an urban canyon, with LOS measurements to landmarks. Derive an expression for the LOS vector, expressed in the sensor frame.



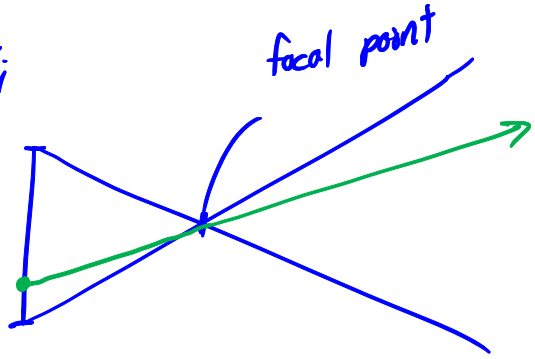
known

T^i_b

T^s_b

Derive an expression for $\underline{l}^s$

$$\underline{r}_{b/i} + \underline{r}_{s/b} + \underline{l} = \underline{r}_{f/i}$$



Solve for $\underline{l}$

$$\underline{l} = \underline{r}_{f/i} - \underline{r}_{b/i} - \underline{r}_{s/b}$$

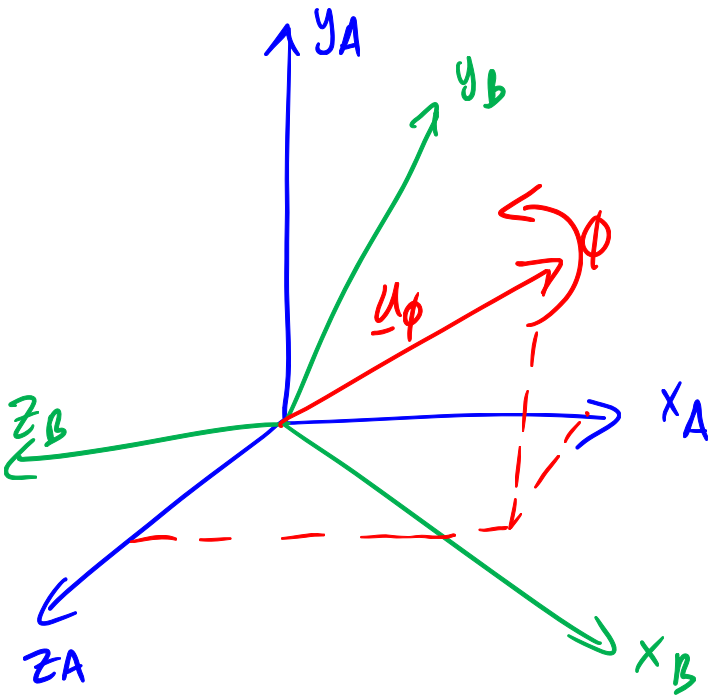
Now choose coordinate system

$$\underline{l}^s = T^s_b T^b_i \underline{r}^i_{f/i} - T^s_b T^b_i \underline{r}^i_{b/i} - T^s_b \underline{r}^b_{s/b}$$

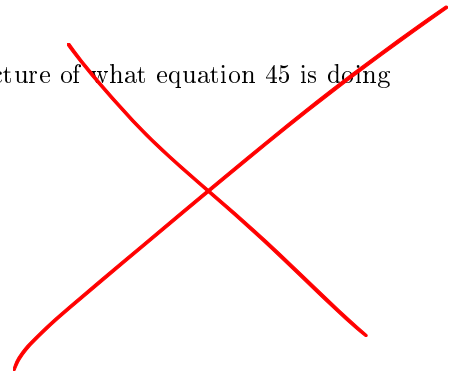
4.2 Rotation Vector

Another useful way to represent the relative orientation of two frames is via the rotation vector. Section 3.2.2 of [2] contains a lengthy derivation of the rotation vector. The end results is that each basis vector (define basis vector) of frame A (i.e. $\underline{\mathrm{u}}_{iA}^{A}$ for $i = X, Y, Z$) can be rotated by an angle ϕ about the axis $\underline{\mathrm{u}}_{\phi}$ to form a new set of basis vectors, which together defines the B frame.

$$\underline{\mathrm{u}}_{iB}^{A} = \left[I + \sin\phi\left(\underline{\mathrm{u}}_{\phi}^{A}\times\right) + (1-\cos\phi)\left(\underline{\mathrm{u}}_{\phi}^{A}\times\right)\left(\underline{\mathrm{u}}_{\phi}^{A}\times\right)\right]\underline{\mathrm{u}}_{iA}^{A} \tag{45}$$



Draw a picture of what equation 45 is doing



Equation 45 can be used to rotate each basis vector of frame A, thus defining the transformation between frame A and B as

$$C_B^A = \left[I + \sin\phi\left(\underline{u}_\phi^A\times\right) + (1-\cos\phi)\left(\underline{u}_\phi^A\times\right)\left(\underline{u}_\phi^A\times\right)\right] \tag{46}$$

Equation 46 shows how to convert a rotation vector to a DCM. Section 3.2.2.2 of [2] derives a method for computing a rotation vector in terms of a DCM.

4.3 Euler Angles

Euler angles characterize the orientation of one coordinate system with respect to another by 3 successive rotations about specified axes. (See first part of section 3.2.3 of [2] for a good description of an Euler angle sequence).

There are many (12) possible combinations of Euler angles (e.g. ZYX, XYZ, XYX, YZY, etc). Each has different singularity points (a.k.a. "gimbal lock" points) when converting from DCM to Euler angles. We will focus on the ZYX Euler angle sequence, which is common for aircraft because it usually avoids the singularity point for typical aircraft orientations. (Draw a picture of a singularity point)

The DCM defined by a ZYX Euler angle sequence can be derived the rotation vector to define in section 4.2.

Step 1: Starting with frame A, create frame A_1 by rotating frame A by an angle ψ, about the Z axis of frame A, i.e. $\underline{u}_{ZA}^A$ (Draw picture and derive $C_{A_1}^A$).

Step 2: Starting with frame A_1, create frame A_2 by rotating frame A_1 by an angle θ about the Y axis of frame A_1, i.e. $\underline{u}_{YA_1}^{A_1}$ (Draw picture and derive $C_{A_2}^{A_1}$).

Step 3: Starting with frame A_2, create frame B by rotating frame A_2 by an angle ϕ about the X axis of frame A_2, i.e. $\underline{u}_{XA_2}^{A_2}$ (Draw picture and derive $C_B^{A_2}$).causing con

Step 4: Compute C_B^A by cascading each Euler angle rotation (derive C_B^A)

The elements of C_B^A are listed in equation 3.2.3.1-2 of [2]. Section 3.2.3.2 of [2] derives a method for computing the ZYX Euler angles from a DCM. This method "gracefully" handles the singularity point.

4.4 Attitude Quaternion

Attitude quaternions are derived by extending imaginary numbers to 3 dimensions. One lengthy derivation of the attitude quaternion is contained in section 3.2.4 of [2]. (Read the beginning of section 3.2.4 for a brief description of the attitude quaternion). Unfortunately, multiple attitude quaternion representations exist, which is the cause of much confusion. The article by [3] provides an excellent history and explanation of vector rotations, transformations, and the corresponding DCM's and quaternions. The option considered in this course is the "Left-handed" quaternion with Hamiltonian multiplication, documented in detail in [2]. Another prominent convention is the "Right-handed" quaternion with non-Hamiltonian multiplication , which is typically encountered in space applications.

4.4.1 Left-Handed Quaternion with Hamiltonian Multiplication

Quaternions can be used much like DCMs, but with different definitions for multiplication (equation 3.2.4-28 of [2]), composition (equation 3.2.4.1-7 of [2]), inversion (equation 3.2.4.13 of [2]), and the transformation of a vector (equation 3.2.4.1-4 of [2]). The attitude quaternion from the B frame to the A frame is defined by a scalar term a and a vector term $\underline{r} = \begin{bmatrix} b & c & d \end{bmatrix}^T$ which have the following relationship to the rotation vector from subsection 4.2.

$$q_B^A = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} \cos\frac{\phi}{2} \\ u_{\phi XA}\sin\frac{\phi}{2} \\ u_{\phi YA}\sin\frac{\phi}{2} \\ u_{\phi ZA}\sin\frac{\phi}{2} \end{bmatrix} = \begin{bmatrix} \cos\frac{\phi}{2} \\ \underline{u}_\phi \sin\frac{\phi}{2} \end{bmatrix} \tag{47}$$

Given two quaternions

$$u = a + \underline{r} \tag{48}$$

$$v = e + \underline{s} \tag{49}$$

where a and e are the scalar portions and $\underline{r}$ and $\underline{s}$ are the vector portions, quaternion multiplication is defined as

$$u \otimes v = \underline{r} \times \underline{s} - \underline{r} \cdot \underline{s} + a\underline{s} + e\underline{r} + ae \tag{50}$$

where the vector and scalar portions must be handled separately. Equation 50 can be written more explicitly as

$$u \otimes v = \begin{bmatrix} ae - \underline{r} \cdot \underline{s} \\ \underline{r} \times \underline{s} + a\underline{s} + e\underline{r} \end{bmatrix} \tag{51}$$

The conjugation of a quaternion (q^*) corresponds to the transposition or inversion of a DCM (recall equation 3.2.1-3 of [2]), in that it reverses the "direction" of the rotation, such that for $q_B^A = \begin{bmatrix} a & b & c & d \end{bmatrix}$

$$q_B^{A*} = q_A^B = \begin{bmatrix} a \\ -b \\ -c \\ -d \end{bmatrix} \tag{52}$$

In other words, the vector portion of the quaternion has been reversed. This can be thought of as negating the direction of the axis of rotation (recall $\underline{u}_\phi$), which changes the direction of the attitude quaternion.

Sections 3.2.4.2 and 3.2.4.3 of [2] contain methods to go from the attitude quaternion to the DCM and from the DCM to the attitude quaternion, respectively. Other important properties of quaternions are:

1. Attitude quaternions have a length of one, i.e. unit norm (see equation 3.2.4.1-2 of [2])
2. Attitude quaternions can be used to transform vectors by converting the vector to a quaternion, then pre and post multiplying (see equation 3.2.4.1-4 of [2]).
3. Attitude quaternions can be cascaded just like DCMs, but using quaternion multiplication rather than matrix multiplication (see equation 3.2.4.1-9 of [2])

Numerical conditioning of a attitude quaternion is easier than for DCM. In the latter, the DCM had to be orthonormalized. In contrast, the only constraint required for an attitude quaternion is that the magnitude be equation to one. Thus maintaining a quaternion inside a simulation or flight code simplifies to

$$q = \frac{q}{|q|} \tag{53}$$

where

$$|q| = \sqrt{a^2 + b^2 + c^2 + d^2} \tag{54}$$

4.4.2 Right-Handed Quaternions and Non-Hamiltonian Multiplication

This section will use the notation in [3], with the exception of the storage of the scalar and vector portions of the quaternion. Based on the concept of a rotation vector, where the B frame is obtained by actively rotating the A frame about an axis $\boldsymbol{n}$ by an angle θ, the "Right-Handed" quaternion is defined as (see eqn 30 of [3])

$$q_A^B = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) \boldsymbol{n}^A \end{bmatrix} \tag{55}$$

Successive transformations behave like their DCM counter parts (i.e. $T_A^C = T_B^C T_A^B$), with the quaternion multiplication defined with the non-Hamiltonian operator. Given two quaternions

$$q_A^B = q_1 + \boldsymbol{q}_1 \tag{56}$$

$$q_B^C = q_2 + \boldsymbol{q}_2 \tag{57}$$

where the bold quantity refers to the vector portion of the quaternion and the non-bold quantity refers to the scalar portion. The composite quaternion is (See eqn 35 of [3])

$$q_A^C = q_B^C \otimes q_A^B = \begin{bmatrix} q_1 q_2 - \boldsymbol{q}_1 \cdot \boldsymbol{q}_2 \\ q_1 \boldsymbol{q}_2 + q_2 \boldsymbol{q}_1 - \boldsymbol{q}_1 \times \boldsymbol{q}_2 \end{bmatrix} \tag{58}$$

The conjugate of a quaternion is analogous to the transpose of a transformation matrix, i.e. it reverses the direction of the transformation. For the quaternion in equation 56

$$q_A^B = \left(q_B^A\right)^* = q_1 - \boldsymbol{q}_1 \tag{59}$$

The quaternion can be converted into the corresponding transformation matrix. Again considering the quaternion in equation 56 (see eqn 39 of [3])

$$T_A^B = I_{3\times 3} - 2q_1 \left[\boldsymbol{q}_1 \times\right] + 2\left[\boldsymbol{q}_1 \times\right]^2 \tag{60}$$

Quaternions can be used to transform vectors by pre and post multiplying a "pure" quaternion containing the vector

$$\delta q\left(T_b^s \delta\boldsymbol{\theta}^b\right) = q_b^s \otimes \delta q\left(\delta\boldsymbol{\theta}^b\right) \otimes q_s^b \tag{61}$$

where

$$\delta q\left(\boldsymbol{\theta}\right) \equiv \begin{bmatrix} 0 \\ \frac{\boldsymbol{\theta}}{2} \end{bmatrix} \tag{62}$$

4.5 When to use which representation?

The decision of which attitude representation to use is dependent partially on preference and partially on what part of the simulation/flight code is being implemented. Some suggestions are given here.

- Euler angles are typically used to report attitude in a way that human's can understand (e.g. heading = 30 deg, pitch of 4 deg, and a roll of 20 deg).
- Quaternions are typically used in spacecraft due to low storage constraints (4 elements, vs 9 elements), but this constraint isn't as important as it used to be.
- Quaternions are also easier to numerically condition (normalize vs. orthonormalize), but the difference in number of computations is also less important than it used to be.
- Direction cosine matrices are more intuitive and can more readily be used to transform vectors or extract the basis vectors of a coordinate system.
- Rotation vectors are typically used in navigation Kalman filters to estimate errors in attitude.

For a good read, see http://www.strapdownassociates.com/Geordie's%20Quaternion.pdf on whether DCMs or quaternions are preferable. He resorts to asking his grandson, and she concludes that it doesn't really matter. I agree.

5 Attitude Kinematics

This section discusses attitude kinematics, which defines the rate of change (derivative) of an attitude representation as a function of the angular rate of the B frame with respect to the A frame. The kinematics are outlined for the DCM, Quaternion, and Euler Angle attitude representations.

5.1 DCM Kinematics

Start by reviewing section 3.3.1 of [2] to understand the following Coriolis relationship which will be used to derive the rate of change of a DCM (from equation 3.3.1-5 of [2])

$$\dot{\underline{V}}^B_{CnstA} = -\underline{\omega}^B_{B|A} \times \underline{V}^B_{CnstA} \tag{63}$$

Walk through the derivation of section 3.3.2 of [2] ending with equation 3.3.2-6

$$\dot{C}^A_B = C^A_B \left(\underline{\omega}^B_{B|A}\times\right) \tag{64}$$

Equation 64 can be used in a simulation to track the changing of the B frame with respect to the A frame given the angular rate of the B frame with respect to the A frame, expressed in the B frame (typical of angular rate sensors used in space, e.g. strapdown fiber optic or ring laser gyros).

5.2 Quaternion Kinematics

Walk through the derivation of section 3.3.4 of [2], ending with equations 3.3.4-14 and 3.3.4-15

$$\dot{q}^A_B = \frac{1}{2} q^A_B \otimes \omega^B_{B|A} \tag{65}$$

$$\omega^B_{B|A} = \begin{bmatrix} 0 \\ \underline{\omega}^B_{B|A} \end{bmatrix} \tag{66}$$

where recall that $\otimes$ refers to quaternion multiplication.

Using the conventions in [3], the quaternion kinematics are defined as

$$\dot{q}^b_i = \frac{1}{2} \begin{bmatrix} 0 \\ \boldsymbol{\omega}^b_{b/i} \end{bmatrix} \otimes q^b_i \tag{67}$$

6 Attitude Error

For this course, the misalignment error in a rotation matrix or quaternion is an important quantity. There are lots of definitions of misalignment error, depending on which frame you consider to be misaligned.

For inertial navigation, the misalignments are typically expressed in a navigation frame. In this case, the definitions of misalignment error for the DCM and quaternion are

$$q^{ned}_b = \begin{bmatrix} 1 \\ -\frac{1}{2}\delta\underline{\theta}^{ned}_b \end{bmatrix} \otimes q^{ned}_{\hat{b}} \tag{68}$$

$$T^{ned}_b = \left[I - \left(\underline{\theta}^{ned}_b \times\right)\right] T^{ned}_{\hat{b}} \tag{69}$$

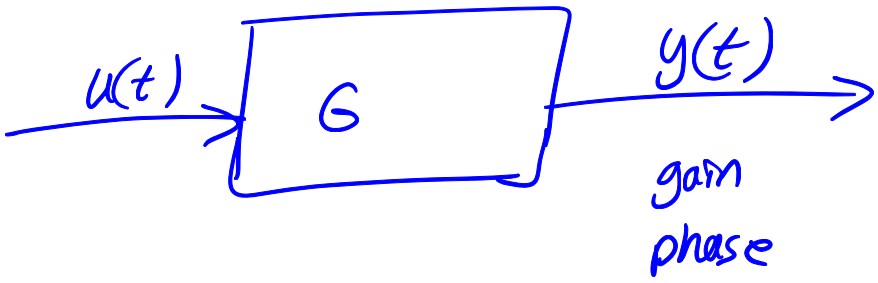
For space applications, the misalignment errors are typically expressed in the body frame.

$$q^b_i = \delta q\left(\delta\boldsymbol{\theta}^b\right) \otimes \hat{q}^b_i \tag{70}$$

$$T^b_i = \left(I_{3\times 3} - \left[\delta\boldsymbol{\theta}^b\times\right]\right) \hat{T}^b_i \tag{71}$$

7 Laplace Transforms

The transfer function describes the steady-state frequency response of a linear, time-invariant differential equation. (Draw a block diagram)



This a fairly limiting case, but is useful when we discuss power spectral density, which describes the steady-state statistics of a time-invariant differential equation driven by a random input. We will review how to go from a differential equation to a transfer function. Given the following differential equations, derive the transfer function.

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = u(t) \tag{72}$$

Goal: Derive $\frac{X(s)}{U(s)}$

$$ms^2X(s) + csX(s) + kX(s) = U(s)$$

$$(ms^2 + cs + k)X(s) = U(s)$$

$$\Rightarrow \frac{X(s)}{U(s)} = \frac{1}{ms^2 + cs + k}$$

8 State Space Representation of Dynamic Systems

A more flexible representation of a physical system is the state-space representation, defined by a vector state differential equation, a vector input, and (sometimes) a vector output equation. (Write general equation for linear, time-varying state space system)

$$\dot{\underline{x}}(t) = A(t)\,\underline{x}(t) + B(t)\,\underline{u}(t)$$

$\underline{x}$: "state"; $\underline{u}$: "input"

$$\underline{z}(t) = C(t)\,\underline{x}(t) + D(t)\,\underline{u}(t)$$

$\underline{z}$: output

As an example, form a (i.e. not unique) linear state-space representation of the differential equation described in equation 72, but with time-varying coefficients[1]. The steps are 1) choose the state vector, 2) choose the input

[1] Maybeck says, "Laplace transform methods are not readily extended to these cases. Such time-varying linear models arise most naturally from perturbations of nonlinear set of relations about a nominal solution to the original nonlinear equations."

vector, 3) choose the output vector (if applicable), and 4) define its derivative. ← output eqn

* $m(t)\ddot{x} + c(t)\dot{x} + k(t)x = F(t)$

① Let $x_1 = x$, $x_2 = \dot{x}$

② Let $u = F$

③ Let $z = x = x_1$

④ $\dot{\underline{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{c}{m}x_2 - \frac{k}{m}x_1 + \frac{u}{m} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix}}_{A(t)} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 1/m \end{bmatrix}}_{B(t)} u$

$z = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{C} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \end{bmatrix}}_{D} u$

* It is important to note that a state-space representation of a system is NOT unique. While the underlying physics/differential equations do not change, the choice of states is up to the designer. Maybeck lists several different state-space forms on pages 28-35. In this class, we will use the "Physical variables" form.

* An advantage of the state-space representation is that it can be easily extended to nonlinear, multiple input, multiple output, time varying differential equations, of the following form (write the nonlinear differential equation with input $\underline{u}$, initial condition $\underline{x}_0$, and output $\underline{z}$)

$$\dot{\underline{x}} = \underline{f}(\underline{x}(t), \underline{u}(t), t), \quad \underline{x}(t_0) = \underline{x}_0$$

$$\underline{z} = \underline{h}(\underline{x}(t), \underline{u}(t), t)$$

Example 2.7 of Maybeck is a good illustration. The steps to forming the state space system are the same as before. (Try it! Draw a picture to understand the equations.)

$$\begin{cases} \ddot{r}(t) = r(t)\dot{\theta}(t) - \dfrac{G}{r^2(t)} + u_r(t) & (73) \\ \ddot{\theta}(t) = -\dfrac{2}{r(t)}\dot{\theta}(t)\dot{r}(t) + \dfrac{1}{r(t)}u_t(t) & (74) \end{cases}$$

① Choose the state vector

$x_1 = r$

$x_2 = \dot{r}$

$x_3 = \theta$

$x_4 = \dot{\theta}$

② Choose the input

$u_1 = u_r$

$u_2 = u_t$

③ Choose output

$z_1 = x = r\cos\theta$

$z_2 = y = r\sin\theta$

④ Define $\underline{f}$ + $\underline{h}$

* $\dot{\underline{x}} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ x_1 x_4 - \frac{G}{x_1^2} + u_1 \\ x_4 \\ -\frac{2}{x_1} x_4 x_2 + \frac{1}{x_1} u_2 \end{bmatrix} = \underline{f}(\underline{x}, \underline{u}, t)$ ✓

$\underline{z} = \begin{bmatrix} x_1 \cos x_3 \\ x_1 \sin x_3 \end{bmatrix} = \underline{h}(\underline{x}, \underline{u}, t)$

While we will not go into the details here, Maybeck discusses requirements for the existence of a solution to the nonlinear differential equation. For physics-based systems, these requirements are almost always satisfied. I.e. although the solution may be difficult to obtain, we can be confident that it exists and has the form

$$\underline{\phi}(t_0) = \underline{x}_0 \quad \text{I.C} \qquad (75)$$

$$\dot{\underline{\phi}}(t) = \underline{\mathrm{f}}(\underline{\phi}(t), \underline{\mathrm{u}}(t), t) \tag{76}$$

9 Linearization Techniques

Once a nominal solution to a nonlinear differential equation is obtained (either analytically or numerically integrated), perturbations about this nominal can be considered. This is done by linearizing the nonlinear system about the nominal solution [2]. We will utilize two methods of linearization in this class, one based on a Taylor series expansion, and one based on perturbation analysis. The multi-variable Taylor series expansion is defined as

$$\underline{\mathrm{f}}(\underline{\mathrm{x}}, \underline{\mathrm{u}}, t) = \underline{\mathrm{f}}(\hat{\underline{\mathrm{x}}} + \delta\underline{\mathrm{x}}, \hat{\underline{\mathrm{u}}} + \delta\underline{\mathrm{u}}, t) = \underline{\mathrm{f}}(\hat{\underline{\mathrm{x}}}, \hat{\underline{\mathrm{u}}}, t) + F_x(t)\,\delta\underline{\mathrm{x}} + F_u(t)\,\delta\underline{\mathrm{u}} + H.O.T. \tag{77}$$

$$\approx \underline{\mathrm{f}}(\hat{\underline{\mathrm{x}}}, \hat{\underline{\mathrm{u}}}, t) + F_x(t)\,\delta\underline{\mathrm{x}} + F_u(t)\,\delta\underline{\mathrm{u}} \tag{78}$$

where F_x and F_u are the "Jacobian" matrices defined as

$$F_x \equiv \left.\frac{\partial \underline{\mathrm{f}}}{\partial \underline{\mathrm{x}}}\right|_{\underline{\mathrm{x}}(t)=\hat{\underline{\mathrm{x}}}(t),\, \underline{\mathrm{u}}(t)=\hat{\underline{\mathrm{u}}}(t)} \tag{79}$$

$$F_u \equiv \left.\frac{\partial \underline{\mathrm{f}}}{\partial \underline{\mathrm{u}}}\right|_{\underline{\mathrm{x}}(t)=\hat{\underline{\mathrm{x}}}(t),\, \underline{\mathrm{u}}(t)=\hat{\underline{\mathrm{u}}}(t)} \tag{80}$$

How does this apply to linearizing a differential equation? The process consists of the following steps: 1) Define nominal state and state perturbations, 2) Derivatives of state perturbations, 3) Define nominal differential equation, 4) Expand the differential equations about the nominal, 5) subtract the nominal. (Show the process in general form).

Let $\dot{x} = f(x, u, t)$

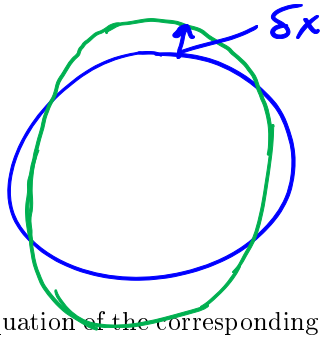
1) Let $x \equiv \hat{x} + \delta x$, $u \equiv \hat{u} + \delta u$

2) $\Rightarrow \dot{x} = \dot{\hat{x}} + \delta\dot{x}$, $\dot{u} = \dot{\hat{u}} + \delta\dot{u}$

3) Let $\dot{\hat{x}} \equiv f(\hat{x}, \hat{u}, t)$

4) $\dot{x} = \dot{\hat{x}} + \delta\dot{x} \approx f(\hat{x}, \hat{u}, t) + f_x\,\delta x + f_u\,\delta u$

5) $\Rightarrow \delta\dot{x} \approx f_x\,\delta x + f_u\,\delta u$



Now derive the linearized perturbation differential equation of the corresponding to example 2.7, repeated below

$$\dot{\underline{\mathrm{x}}} = \begin{bmatrix} x_2 \\ x_1 x_4^2 - \frac{G}{x_1^2} + u_1 \\ x_4 \\ -\frac{2}{x_1} x_4 x_2 + \frac{1}{x_1} u_2 \end{bmatrix} = f(\underline{x}, \underline{u}) \tag{81}$$

[2] For the extended Kalman filter, the "nominal solution" is simply chosen to be the current estimated state.

$$F_x = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} \\ \frac{\partial f_2}{\partial x_1} & \cdots & & \\ \vdots & & & \end{bmatrix}_{4\times 4} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \hat{x}_4^2 + 2\frac{G}{\hat{x}_1^3} & 0 & 0 & 2\hat{x}_1\hat{x}_4 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

$$F_u = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} \\ \frac{\partial f_2}{\partial u_1} & \vdots \\ \vdots & \end{bmatrix}_{4\times 2} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1/\hat{x}_1 \end{bmatrix}$$

The second method of linearization is useful when the Jacobians either don't exist or are difficult to obtain. The procedure consists of the following steps: 1) Define state and input perturbations, 2) Define any needed derivatives, 3) Define the nominal differential equation, 4) Substitute perturbations into differential equation, 5) Cancel out the nominal, 6) Discard second-order or higher terms. This will be illustrated with the following example

$$\begin{bmatrix} \dot{v} \\ \dot{d} \end{bmatrix} = \begin{bmatrix} d + a - \alpha v^2 \\ -\frac{d}{\tau_d} + w \end{bmatrix} \tag{82}$$

1) Let $v \equiv \hat{v} + \delta v$

$d \equiv \hat{d} + \delta d$

$a \equiv \hat{a} + \delta a$

$w \equiv \hat{w} + \delta w$

2) $\Rightarrow \dot{v} = \dot{\hat{v}} + \delta\dot{v}$

$\dot{d} = \dot{\hat{d}} + \delta\dot{d}$

3) $\dot{\hat{v}} = \hat{d} + \hat{a} - \alpha\hat{v}^2$

$\dot{\hat{d}} = -\frac{\hat{d}}{\tau_d} + \hat{w}$

4) $\dot{\hat{v}} + \delta\dot{v} = \hat{d} + \delta d + \hat{a} + \delta a - \alpha(\hat{v} + \delta v)^2$

$\dot{\hat{d}} + \delta\dot{d} = -\frac{1}{\tau_d}(\hat{d} + \delta d) + \hat{w} + \delta w$

5) $\dot{\hat{v}} + \delta\dot{v} = \hat{d} + \delta d + \hat{a} + \delta a - \alpha\hat{v}^2 - \alpha 2\hat{v}\delta v - \alpha\delta v^2$

$\dot{\hat{d}} + \delta\dot{d} = -\frac{1}{\tau_d}\hat{d} - \frac{1}{\tau_d}\delta d + \hat{w} + \delta w$

6) $\delta\dot{v} = \delta d + \delta a - \alpha 2\hat{v}\delta v - \alpha\delta v^2$ ($\delta v^2 \sim 0$)

$\delta\dot{d} = -\frac{1}{\tau_d}\delta d + \delta w$

10 Solutions to Linear Differential Equations

The "state transition matrix" (or STM) is an important quantity in the implementation of an indirect (i.e. error state) extended Kalman filter. Linearization of a non-linear differential equation usually results in time-varying linear differential perturbation equation of the form.

$$\delta\dot{\underline{x}}(t) = F(t)\,\delta\underline{x}(t) + B(t)\,\delta\underline{u}(t)\,; \quad \delta\underline{x}(t_0) = \delta\underline{x}_0 \tag{83}$$

The solution to this differential equation is (see equation 2-52 of Maybeck)

$$\delta\underline{x}(t) = \Phi(t, t_0)\,\delta x_0 + \int_{t_0}^{t} \Phi(t, \tau)\,B(\tau)\,u(\tau)\,d\tau \tag{84}$$

where $\Phi(t, t_0)$ is the STM which "transitions" the state from its initial condition to some time t. How do we determine the STM? It must satisfy the state differential equation AND have a specific initial condition. What does this mean in equation form? (See equations 2-53a,b of Maybeck)

$$\dot{\Phi}(t, t_0) = F(t)\,\Phi(t, t_0)$$

$$\Phi(t_0, t_0) = I$$

This is a nice analytical description of the STM, but how do we determine the STM in a real application? It depends on whether the perturbation system is time-invariant or time-varying. (List the STM for each case, see equation 2-58a in Maybeck for LTI case)

* LTI: $\Phi(t,t_0) = \Phi(t-t_0) = \exp\{F(t-t_0)\}$

LTV: $\Phi(t,t_0) = \exp\left\{\int_{t_0}^{t} F(\tau)d\tau\right\}$

Learis Method: $\Phi(t_k,t_{k-1}) \approx I + \frac{\Delta t}{2}(F_k + F_{k-1}) + \frac{\Delta t^2}{2} F_k F_{k-1}$

The matrix exponential can be analytically defined using the infinite series

$$\exp(F\Delta t) = \sum_{n=0}^{\infty} F^n \frac{\Delta t^n}{n!} = I + F\Delta t + F^2\frac{\Delta t^2}{2} + F^3\frac{\Delta t^3}{3!} + F^4\frac{\Delta t^4}{4!} \quad (85)$$

n=0, n=1, n=2, n>3, n=4

In some cases, the series vanishes after a certain number of terms. In other cases the series is truncated to approximate the exponential. (Provide an example of determining the STM for the errors of an inertial navigation system flying straight and level in the north direction.)

$$\delta\dot{\boldsymbol{x}} = F\delta\boldsymbol{x} \quad (86)$$

where $\delta x = [\delta p \ \delta v \ \delta\theta]^T$ Derive an expression for $\delta p(t)$

For the case of straight-and-level flight, the dynamic coupling matrix of equation is constant,

$$F = \begin{bmatrix} 0_{3\times3} & I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & (\boldsymbol{\nu}^n)\times \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \quad (87)$$

The accelerometer measurements are also constant and expressed in the n frame as

$$\boldsymbol{\nu}^n = \begin{bmatrix} 0 & 0 & -g \end{bmatrix}^T \text{ — constant} \quad (88)$$

Since F is constant, the STM is determined as the matrix exponential ([1], page 42)

$$\Phi(t_{k+1},t_k) = \exp(F\Delta t) = \sum_{n=0}^{\infty} F^n \frac{\Delta t^n}{n!} \quad (89)$$

where $\Delta t = t_{k+1} - t_k$. Substitution of equation 87 into 89 yields the following summation

$$\Phi(t_{k+1},t_k) = I \quad (90)$$ n=0

$$+ \begin{bmatrix} 0_{3\times3} & I_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & (\boldsymbol{\nu}^n)\times \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \Delta t \quad (91)$$ n=1

$$+ \begin{bmatrix} 0_{3\times3} & 0_{3\times3} & (\boldsymbol{\nu}^n\times) \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \frac{\Delta t^2}{2!} \quad (92)$$ n=2

$$+ \begin{bmatrix} 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \frac{\Delta t^3}{3!} \quad (93)$$ n=3

$$+ \cdots \quad (94)$$

where, since the cubed and higher terms are zero, yields the following STM

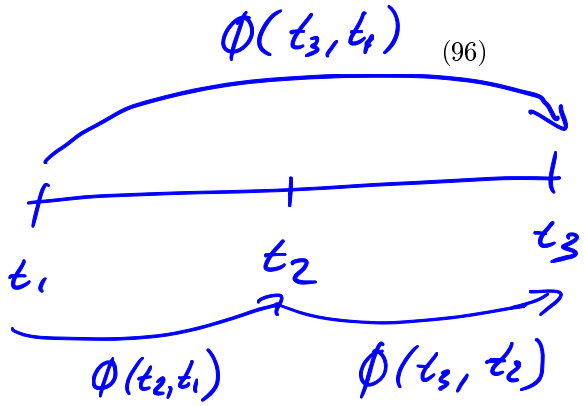
$$\Phi(t_{k+1},t_k) = \begin{bmatrix} I_{3\times3} & I_{3\times3}\Delta t & (\boldsymbol{\nu}^n\times)\frac{\Delta t^2}{2!} \\ 0_{3\times3} & I_{3\times3} & (\boldsymbol{\nu}^n)\times\Delta t \\ 0_{3\times3} & 0_{3\times3} & I_{3\times3} \end{bmatrix} \quad (95)$$

$$\begin{bmatrix} \delta p(t) \\ \delta v(t) \\ \delta\theta(t) \end{bmatrix} = \Phi(t,t_0) \begin{bmatrix} \delta p(t_0) \\ \delta v(t_0) \\ \delta\theta(t_0) \end{bmatrix}$$

$\delta p(t) =$

The desired analytical expression for position errors is obtained from the first row of the STM to yield (An example of this calculation using Matlab is show in stm.m)

$$\delta \boldsymbol{p}^n(t) = \delta \boldsymbol{p}_0^n + \delta \boldsymbol{v}_0^n \Delta t + \boldsymbol{\nu}^n \times \delta \boldsymbol{\theta}_0^n \frac{\Delta t^2}{2!} \tag{96}$$

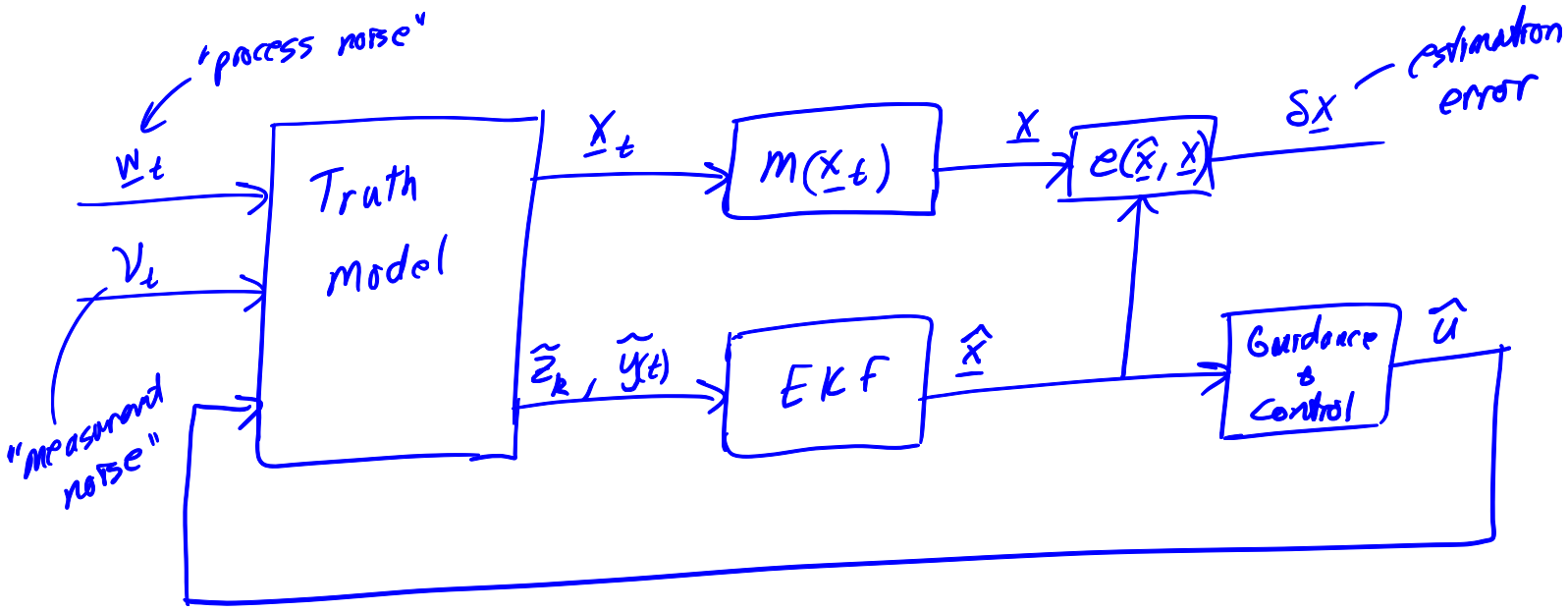


Some useful properties of the STM are the following

1. $\Phi(t, t_0)$ is uniquely defined for all t and t_0 in $[0, \infty)$
2. The STM's can be cascaded, i.e. $\Phi(t_3, t_1) = \Phi(t_3, t_2)\,\Phi(t_2, t_1)$
3. The STM is invertible $\Phi^{-1}(t, t_0) = \Phi(t_0, t)$

11 Truth Model vs. Design Model and Model Replacement

In this class, you will implement a Monte Carlo simulation of the navigation system for you chosen project. The block diagram of a typical guidance, navigation, and control simulation is shown below.



An important concept in the development of a Monte Carlo simulation is the difference between the truth model and the design model. The truth model represents what happens in the real world, or our best description of the real world. The design model is a simplification of the real world, and is used to design the Kalman filter. Examples of these two types of models are various. Sometimes they are identical. Sometimes they differ significantly. In the case of inertial navigation, we also employ a technique called "model replacement" in the design model, where we replace complicated dynamics with a "continuous" sensor. Below is the "typical" approach for inertial navigation, when you have the luxury of owning the entire simulation. (List the truth model and design model equations)

Truth model

$$\dot{\underline{r}}^i_{b/i} = \underline{v}^i_{b/i}$$

$$\dot{\underline{v}}^i_{b/i} = \underline{a}^i_{grav} + \underline{a}^i_{thr} + \underline{\epsilon}^i_{grav} + \underline{w}^i_a$$

$$\dot{q}^b_i = \frac{1}{2}\begin{bmatrix} 0 \\ \underline{\omega}^b_{b/i} \end{bmatrix} \otimes q^b_i$$

$$\dot{\underline{\omega}}^b_{b/i} = I_b^{-1}\left(-\underline{\omega}^b_{b/i} \times I_b \underline{\omega}^b_{b/i} + \underline{T}^b\right) + \underline{w}^b_r$$

$$\dot{\underline{b}}_a = -\frac{1}{\tau_a}\underline{b}_a + \underline{n}_a$$

$$\dot{b}_g = -\frac{1}{\tau_g} b_g + n_g$$

$$\dot{\underline{\epsilon}}^i_{grav} = -\frac{V_\perp}{d_{grav}}\underline{\epsilon}^i_{grav} + \underline{w}^i_{grav}$$

$$\underline{a}^i_{grav} = -\frac{\mu}{\|\underline{r}^i_{b/i}\|^3}\underline{r}^i_{b/i}$$

↑ point mass gravity model

Design model

$$\dot{\underline{r}}^i_{b/i} = \underline{v}^i_{b/i}$$

$$\dot{\underline{v}}^i_{b/i} = T^i_b \underline{v}^b + \underline{a}^i_{grav} + \underline{\epsilon}_{grav}$$

$$\dot{q}^b_i = \frac{1}{2}\begin{bmatrix} 0 \\ \underline{\omega}^b_{b/i} \end{bmatrix} \otimes q^b_i$$

$$\dot{\underline{b}}_a = -\frac{1}{\tau_a}\underline{b}_a + \underline{n}_a \quad \left.\begin{matrix} \\ \\ \end{matrix}\right\} \text{"process noise"}$$

$$\dot{\underline{b}}_g = -\frac{1}{\tau_g} b_g + \underline{n}_g$$

$$\dot{\underline{\epsilon}}_{grav} = -\frac{V_\perp}{d_{grav}} + \underline{w}_{grav}$$

where

$$* \tilde{\underline{v}}^b = T^b_i\left(\underline{a}^i_{thr} + \underline{w}^i_a\right) + \underline{b}_a + \underline{n}_a$$

$$* \tilde{\underline{\omega}}^b_{b/i} = \underline{\omega}^b_{b/i} + \underline{b}_g + \underline{n}_g$$

EKF Propagation

$$\dot{\hat{\underline{r}}}^i_{b/i} = \hat{\underline{v}}^i_{b/i}$$

$$\dot{\hat{\underline{v}}}^i_{b/i} = \hat{T}^i_b \hat{\underline{v}}^b + \hat{\underline{a}}_{grav} + \hat{\underline{\epsilon}}_{grav}$$

$$\dot{\hat{q}}^b_i = \frac{1}{2}\begin{bmatrix} 0 \\ \hat{\underline{\omega}}^b_{b/i} \end{bmatrix} \otimes \hat{q}^b_i$$

$$\dot{\hat{\underline{b}}}_a = -\frac{1}{\tau_a}\hat{\underline{b}}_a$$

$$\dot{\hat{b}}_g = -\frac{1}{\tau_g}\hat{\underline{b}}_g$$

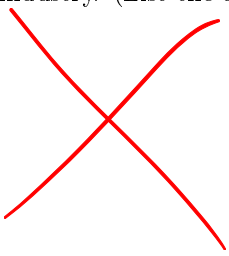
$$\dot{\hat{\underline{\epsilon}}}_{grav} = -\frac{V_\perp}{d_{grav}}\hat{\underline{\epsilon}}_{grav}$$

where

$$\hat{\underline{v}}^b = \tilde{\underline{v}}^b - \hat{\underline{b}}_g$$

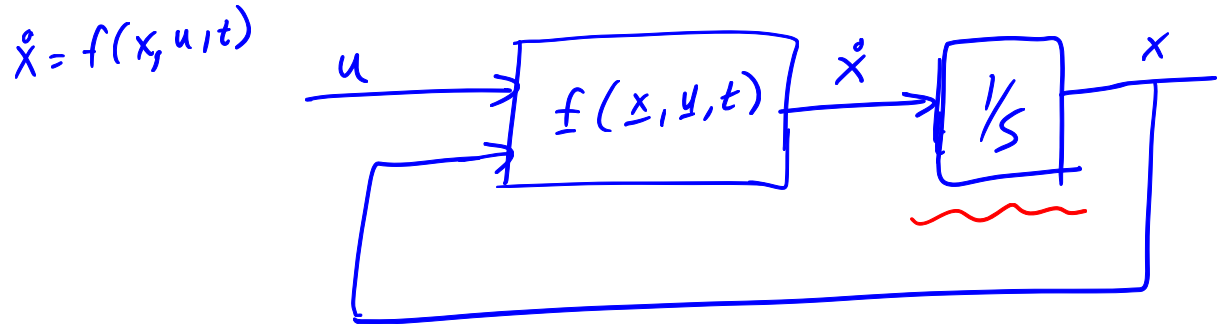
$$\hat{\underline{\omega}}^b_{b/i} = \tilde{\underline{\omega}}^b_{b/i} - \hat{\underline{b}}_g$$

In our specific example of inertial navigation for UAVs, we interfaced with some aerodynamics guys and used their flight simulation. This is more typical in industry. (List the truth model and design model for this case).



12 Numerical Integration of Differential Equations

The analytical solution to the nonlinear differential state space equation is rarely known in practice, so we need a numerical method for solving the differential equation. A typical approach is known as a 4th Order Runge Kutta integrator. (Draw a block diagram of the integrator). There are infinitely-many possible versions of a 4th Order



Runge Kutta numerical integrator. The "Classical Fourth-Order RK" equations for an ordinary differential equation of the form

$$\dot{\underline{x}} = \underline{f}(\underline{x}, \underline{u}, t) \tag{97}$$

$$\underline{x}_{i+1} = \underline{x}_i + \frac{1}{6}(\underline{k}_1 + 2\underline{k}_2 + 2\underline{k}_3 + \underline{k}_4)\, h \qquad \text{where} \quad h = (t_{i+1} - t_i) \tag{98}$$

where

$$\underline{k}_1 = \underline{f}(\underline{x}_i, \underline{u}_i, t_i) \tag{99}$$

$$\underline{k}_2 = \underline{f}(\underline{x}_i + 0.5\underline{k}_1 h, \underline{u}_i, t_i + 0.5h) \tag{100}$$

$$\underline{k}_3 = \underline{f}(\underline{x}_i + 0.5\underline{k}_2 h, \underline{u}_i, t_i + 0.5h) \tag{101}$$

$$\underline{k}_4 = \underline{f}(\underline{x}_i + \underline{k}_3 h, \underline{u}_i, t_i + h) \tag{102}$$

References

[1] P. S. Maybeck, *Stochastic Models, Estimation, and Control.* New York: Navtech Book and Software Store, 1994, vol. 1.

[2] P. G. Savage, *Strapdown analytics.* Maple Plain, Minn.: Strapdown Associates, 2000.

[3] R. Zanetti, "Rotations, Transformations, Left Quaternions, Right Quaternions?" *The Journal of the Astronautical Sciences*, vol. 66, no. 3, pp. 361–381, Sep. 2019. [Online]. Available: http://link.springer.com/10.1007/s40295-018-00151-2