

Control, Estimation, Modeling

# Compiled Notes on Control Theory

June 2025

## Table of Contents

1	Goals of Control Theory . . . . .	6
1.1	Applications to Aerospace . . . . .	7
1.2	Timeline of Development. . . . .	7
1.3	Examples of Control Systems . . . . .	7
1.4	Common Definitions . . . . .	7
2	Background . . . . .	8
2.1	Calculus . . . . .	8
2.1.1	Euler's Formula . . . . .	8
2.1.2	Differentiation . . . . .	8
2.1.3	Integration . . . . .	8
2.1.4	Taylor Series. . . . .	8
2.2	Differential Equations . . . . .	8
2.2.1	Ordinary (ODE) . . . . .	8
2.2.2	Partial (PDE) . . . . .	8
2.2.3	Coupled First-order Nonlinear Differential Equations . . . . .	8
2.2.4	Jacobian Matrix . . . . .	8
2.2.5	Convolution . . . . .	8
2.2.6	Complex-plane Geometry . . . . .	9
2.2.7	Fourier Transforms . . . . .	9
2.2.8	Laplace Transforms. . . . .	9
2.2.9	Delayed Differential Equations. . . . .	11
2.2.10	Poles & Zeros . . . . .	11
2.3	Linear Algebra . . . . .	11
2.3.1	Basic Operations . . . . .	11
2.3.2	Matrix Inversion . . . . .	11
2.3.3	Eigenvectors & Eigenvalues . . . . .	11
2.4	Mechanics (Newtonian) . . . . .	12
2.4.1	Linear . . . . .	12
2.4.2	Rotational . . . . .	13
2.4.3	Energy . . . . .	13
2.4.4	Electrical . . . . .	13
2.5	Lagrangian Mechanics . . . . .	14
3	System Identification & Analysis . . . . .	18
3.1	Linear Regression . . . . .	18
3.2	State-space Models . . . . .	18
3.3	Linear, Time-Invariant Systems (LTI). . . . .	18
3.3.1	Properties . . . . .	18
3.3.2	Convolution . . . . .	18
3.4	Linear, Time-Varying Systems (LTV) . . . . .	19
3.4.1	Properties . . . . .	19

3.5	Modal Analysis . . . . .	20
3.5.1	System Poles. . . . .	23
3.5.2	System Zeros . . . . .	23
3.5.3	System Modes for Second-order Systems . . . . .	26
3.6	System Stability . . . . .	26
3.6.1	Bounded Input, Bounded Output Stability. . . . .	26
3.6.2	Routh-Hourwitz Criterion. . . . .	26
3.6.3	Nyquist Criterion . . . . .	26
3.7	Sensitivity . . . . .	26
3.7.1	Cosensitivity . . . . .	26
4	Linear Control Methods . . . . .	27
4.1	Open Loop . . . . .	27
4.2	Closed Loop (Feedback) . . . . .	27
4.2.1	Purpose. . . . .	28
4.3	Bode Plots . . . . .	28
4.4	PID Control . . . . .	28
4.4.1	Gain Margin. . . . .	29
4.4.2	Phase Margin . . . . .	29
4.4.3	Crossover Frequency . . . . .	29
4.4.4	Integrator Anti-windup Term . . . . .	29
4.5	Higher-order Controllers. . . . .	29
4.5.1	Lead-Lag Control . . . . .	29
4.6	Feedforward . . . . .	29
4.7	Multi-Input, Multi-Output (MIMO) . . . . .	29
4.8	Designing the Controller . . . . .	29
4.8.1	Frequency Response Method . . . . .	29
4.8.2	Root Locus Method. . . . .	29
4.8.3	Loop-Shaping Method . . . . .	29
4.8.4	Examples . . . . .	30
5	Nonlinear Methods . . . . .	31
5.1	Linearization . . . . .	32
5.1.1	Taylor Series Expansion . . . . .	32
5.1.2	Jacobian. . . . .	32
5.2	Sliding Mode Control . . . . .	32
6	State Estimation. . . . .	33
6.1	Purpose . . . . .	33
6.2	Observer. . . . .	33
6.3	Measurement Model. . . . .	33
6.4	Filtering . . . . .	33
6.4.1	Calibration . . . . .	33
6.5	Complementary Filter . . . . .	33
6.6	Alpha-Beta Filter . . . . .	33
6.6.1	Alpha-Beta-Gamma Filter . . . . .	33

6.7	Least-Squares Estimator . . . . .	33
6.8	Linear-Quadratic Estimator (Kalman Filter). . . . .	33
6.8.1	Extended Kalman Filter (EKF) . . . . .	33
6.8.2	Unscented Kalman Filter (UKF) . . . . .	33
6.9	Particle Filter . . . . .	33
6.9.1	SIS Filter . . . . .	33
6.9.2	SIR Filter . . . . .	33
6.9.3	The Curse of Dimensionality. . . . .	33
6.10	Ensemble Kalman Filter (EnKF). . . . .	33
6.11	Sensor Fusion. . . . .	33
6.11.1	Madgwick Filter . . . . .	33
6.11.2	Mahoni Filter . . . . .	33
7	Modeling & Simulation . . . . .	34
7.1	Reference Frames . . . . .	34
7.1.1	Inertial Frame . . . . .	34
7.1.2	Body Frame . . . . .	34
7.1.3	North-East-Down Frame . . . . .	34
7.1.4	Intermediate Frames . . . . .	34
7.1.5	Coriolis Forces . . . . .	34
7.2	Determining the State Vector . . . . .	34
7.2.1	Position & Derivatives . . . . .	34
7.2.2	Rotation & Derivatives . . . . .	34
7.2.3	Homogenous Transformation Matrix . . . . .	34
7.3	Flat Earth Equations of Motion . . . . .	35
8	Optimal Control . . . . .	36
8.1	Linear-Quadratic Regulator (LQR). . . . .	36
9	Robust Control . . . . .	37
10	Adaptive Control . . . . .	38
11	Predictive Control. . . . .	39
12	Other Important Topics . . . . .	40
12.1	Frequency Domain vs Time Domain . . . . .	40
12.2	Continuous Time vs Discrete Time . . . . .	40
12.2.1	Z Transform. . . . .	40
12.3	Optimization . . . . .	40
12.3.1	Gradient Descent. . . . .	40
12.4	Singular Value Decomposition . . . . .	40
13	Appendix: Quick Reference . . . . .	41
14	Appendix: MATLAB / Simulink Reference . . . . .	42
14.1	Linearization . . . . .	42
14.1.1	MATLAB . . . . .	42
14.1.2	Simulink . . . . .	42
14.2	Animation . . . . .	42
14.2.1	MATLAB . . . . .	42
14.2.2	Simulink . . . . .	42

15	Appendix: Python Reference . . . . .	43
16	References . . . . .	44

# 1 Goals of Control Theory

Control theory is a branch of engineering that deals with the behavior of dynamical systems and how to influence that behavior through inputs. Typical considerations are:

**Stability** Ensuring the system remains bounded and predictable over time, and preventing oscillations or otherwise chaotic behavior.

**Controllability** Determining whether a system can be driven from an initial state to a desired final state using the available control inputs.

**Observability** Assessing how much of the internal state of a system can be inferred from its outputs. Crucial for systems where direct measurement of all variables isn't possible.

**Optimality** Achieving the desired behavior with minimal cost (energy, time). Involves balancing between speed, accuracy, and resource usage.

**Robustness** Maintaining performance despite uncertainties, disturbances, or modelling inaccuracies. The controller must be able to handle real world imperfections!

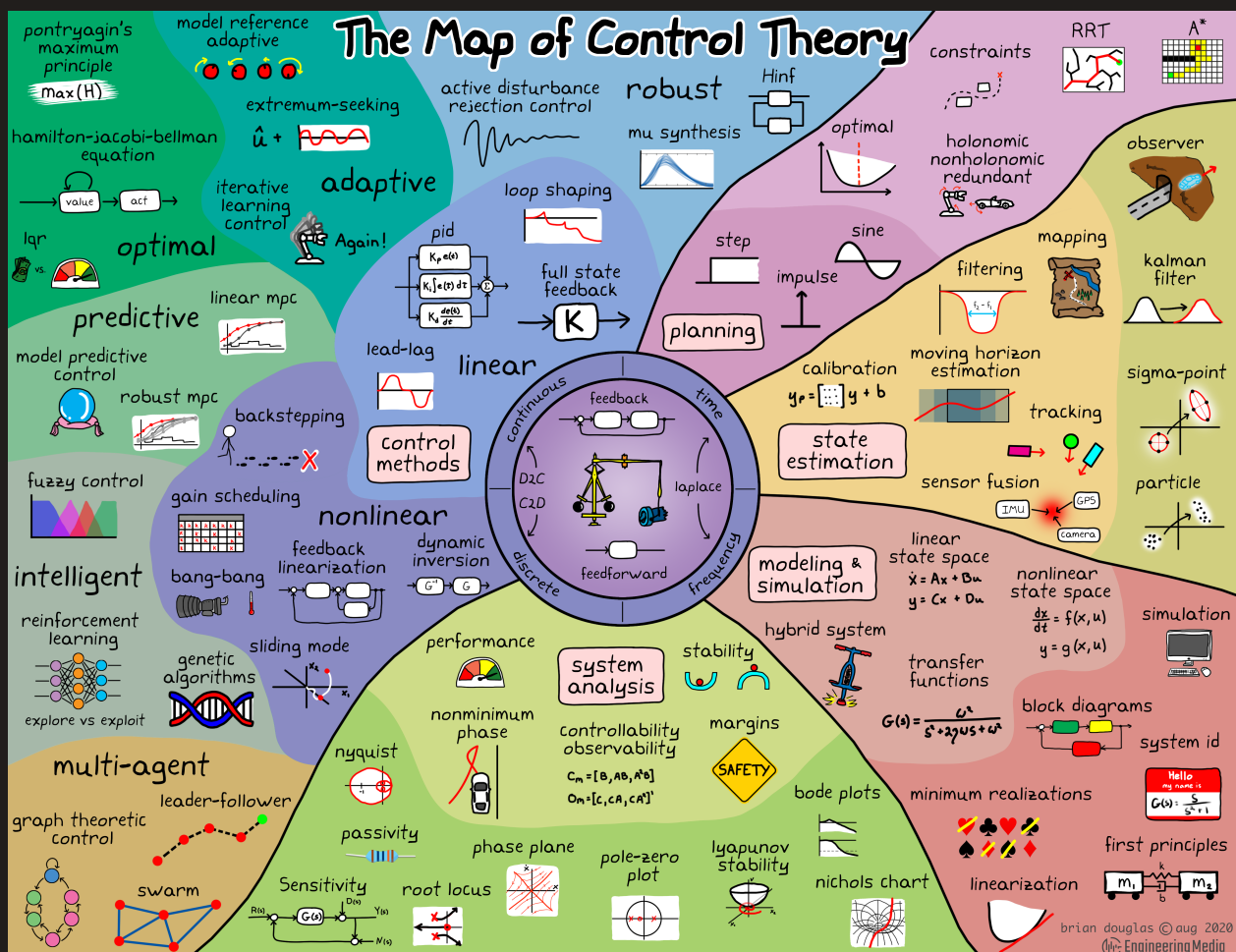


Figure 1: Brian Douglas' Concept Map of Control Theory

Control theory is an interdisciplinary topic. It can be applied to mechanical, electrical, thermal, fluid, really any system. Examples include a swinging pendulum, predator v.s. prey dynamics, RLC circuits, mass-spring-damper systems, etc.

## 1.1 Applications to Aerospace

The main application of control theory in aerospace is in Guidance, Navigation, and Control (GNC). In order from most broad to least:

**Navigation (Statistics)** "Given the measurements we have, where do we think we are?"

**Guidance (Optimization)** "Given where we want to go and where we think we are, which path should we follow?"

**Control (Differential Equations)** "What effort should we apply and how should we control actuators given where we are and the chosen path?"

All require knowledge of dynamics and linear algebra, and can be further broken down into numerous subcategories.

Control theory offers many benefits in the aerospace application.

**Automation** Remove the need for human intervention, preserving the attention of the operator for more important matters.

**Performance** Operate more effectively than a human operator.

**Safeguards / Protections** Prevent the craft from exceeding design limits.

**Deferred Decision-making** Let the human operator make certain flight critical decisions.

## 1.2 Timeline of Development

### Classical Control (prior to 1950's)

- Analysis of linear control systems.
- Controller design using frequency domain representation.

### Advanced Topics (post 1950's)

- Multivariable systems (state domain representation).
- Nonlinear and adaptive control.
- Optimal control.
- Estimation and filtering.
- Stochastic control and reinforcement learning.

## 1.3 Examples of Control Systems

**Falcon 9 Landing** An example of controlling an inverted pendulum. Following a flight trajectory (actual vs desired path).

**X-29 jet** Unstable design leads to control difficulties, but also results in high maneuverability.

## 1.4 Common Definitions

**Dynamical System** A collection of quantities / variables with significance that change over time due to interactions with each other and the environment. Typical form:  
 $\dot{x} = f(x)$ .

**Dimension** The number of quantities / variables to model. There are infinite dimensional systems as well, for example fluid flow.

## 2 Background

Some background information must be understood before delving into control theory.

### 2.1 Calculus

I am not trying to rewrite a textbook here, nor do I have to knowledge to. This will just be a quick refresher on the most important topics.

#### 2.1.1 Euler's Formula

$$e^{ix} = \cos(x) + i \sin(x) \quad (2.1)$$

#### Euler's Identity

$$e^{i\pi} + 1 = 0 \quad (2.2)$$

#### 2.1.2 Differentiation

It can be simply described as the rate of change of a quantity or function. Represented by the slope of a function.

#### 2.1.3 Integration

It can be simply described as the sum of change of a quantity or function. Represented by the area under the curve of a function.

#### 2.1.4 Taylor Series

### 2.2 Differential Equations

#### 2.2.1 Ordinary (ODE)

#### 2.2.2 Partial (PDE)

#### 2.2.3 Coupled First-order Nonlinear Differential Equations

**Coupled** - variables  $x_i$  might appear in equations  $x_j$  for  $i \neq j$ . Control input  $u_i$  may affect multiple states  $x_j$  simultaneously. A single state may be affected by multiple control inputs at once.

**First-order** - Highest derivative is the first derivative.

**Nonlinear** - equations include terms that cannot be expressed as linear combinations (scaling, addition) of  $x$  and  $u$ . In practice, this happens when  $x$  and  $u$  have non-unity exponents or are included inside

**Ordinary** - Equations include a partial w.r.t. only one variable (usually  $t$ , in contrast to a PDE)

#### 2.2.4 Jacobian Matrix

#### 2.2.5 Convolution

Convolution integrals are useful for solving initial value problems with general forcing functions.

$$(f * g)(t) = \int_0^t f(t - \tau)g(\tau)d\tau \quad (2.3)$$



Convolution is a way of combining two functions (signals, images, etc.) to produce a third one that shows how one affects the other. Intuitively, it can be thought of as a blending or filtering process.

### 2.2.6 Complex-plane Geometry

### 2.2.7 Fourier Transforms

#### Fourier Series

**Fourier Transforms** Any periodic function can be represented by a Fourier series (series of periodic functions).

For periodic functions that decay to zero, we can use the Fourier transform.

$$f(w) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t} dt \quad (2.4)$$

This transform converts a continuous time domain signal into a continuous frequency domain signal. In other words, it converts a signal into its component frequencies.

### 2.2.8 Laplace Transforms

The Laplace transform is a more general case of the Fourier transform.

$$F(s) = \int_0^{\infty} f(t)e^{-st} dt \quad (2.5)$$

A system's transfer function is the Laplace transform of the system's impulse response. For a given LTI system, the transfer function is unique, whereas the state space representation may not be. Like the transfer function, the feedthrough  $D$  matrix is always unique.

**Convolution Theorem** Convolution in the time domain is equal to multiplication in the frequency domain.

$$\mathcal{L}\{[x * y](t)\} = X(s) \cdot Y(s) \quad (2.6)$$

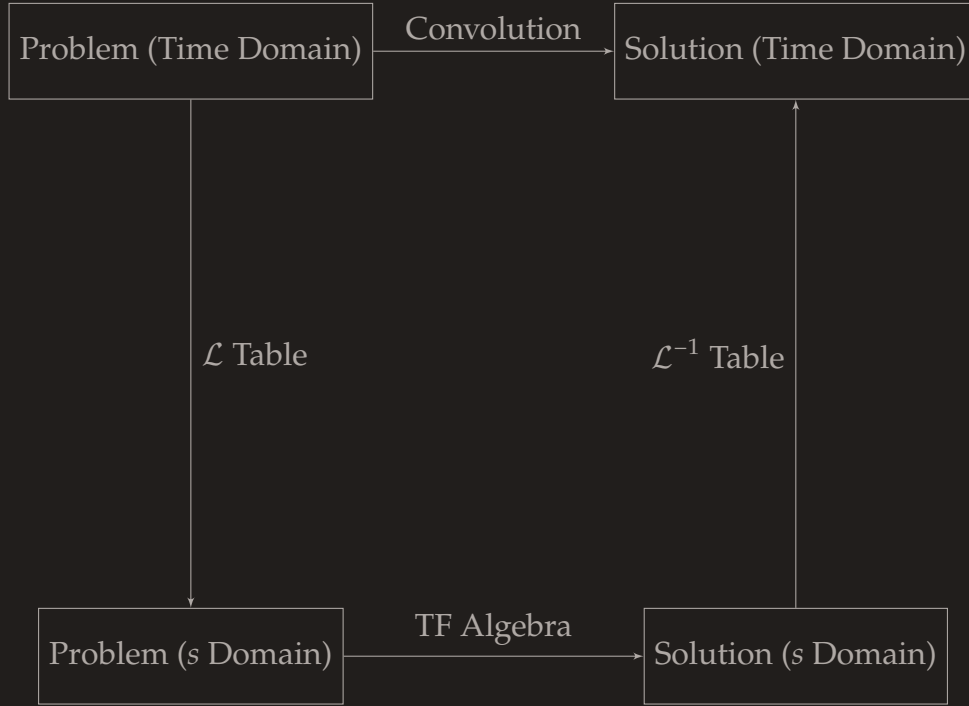


Figure 2: Laplace Problem Approach Flowchart

Two alternative approaches: solve the (convoluted) convolution integral, or transform into the frequency domain, perform simple algebra, and use the inverse transform to find the solution in the time domain.

**Transfer Functions** Applying the Laplace transform to a time domain function will produce the frequency domain representation the input-output relationship of a SISO LTI system. Classical transfer functions do not exist for LTV or nonlinear systems. In the typical state space system:

$$\begin{aligned}\dot{X}(s) &= sX(s) - X(0) = AX(s) + BU(s) \\ Y(s) &= CX(s) + DU(s)\end{aligned}$$

Solving this system gives

$$Y(s) = C(sI - A)^{-1}[BU(s) + X(0)] + DU(s) \quad (2.7)$$

The quantity  $(sI - A)^{-1}$  is called the resolvent of  $A$  and represents the Laplace transform of the state transition matrix. Like the state transition matrix, the resolvent is always invertible, regardless of the invertibility of  $A$ . If the initial condition is zero:

$$T(s) := \frac{Y(s)}{U(s)} = C(sI - A)^{-1}B + D \quad (2.8)$$

### 2.2.9 Delayed Differential Equations

DDE are when the system depends on a delayed version of state.

$$\text{eg. } u(t) = x(t - \tau), \text{ with } \tau > 0 \quad (2.9)$$

These types of equations require specialized solvers.

### 2.2.10 Poles & Zeros

The poles and zeros of a transfer function provide some critical information on the behavior of a system.

#### Poles

#### Zeros

**Relative Degree of Transfer Functions** A transfer function with  $n_p$  poles and  $n_z$  zeros has a relative degree of  $n_p - n_z$ .

**Strictly Proper** A transfer function whose numerator has a lesser order than the denominator (relative degree greater than zero) is called strictly proper.

**Semi-Proper** When the relative degree is zero, the transfer function is bi-proper or semi-proper.

**Improper** When the numerator is a greater order than the denominator, the transfer function is improper.

The feed-through matrix  $D$  is zero when the transfer function is strictly proper, and non-zero when the transfer function is semi-proper. A state-space representation of a system only exists if the transfer function is proper (either strictly proper or semi-proper). No definitions of  $x, A, B, C, D$  can be found for an improper system. Physical systems are always proper, and practically always strictly proper.

## 2.3 Linear Algebra

### 2.3.1 Basic Operations

### 2.3.2 Matrix Inversion

### 2.3.3 Eigenvectors & Eigenvalues

The eigenvalue problem seeks to find the directions  $v$  of the square matrix  $A$  that result in a pure scaling of  $\lambda$  of  $v$  when  $A$  is right-multiplied by  $v$ .

$$\begin{aligned} \lambda &- \text{Eigenvalues} \\ v &- \text{Eigenvectors} \end{aligned}$$

This must satisfy the following equation:

$$Av_i = \lambda_i v_i \quad (2.10)$$

where in general  $\lambda_i \in \mathbb{C}$  is a complex scalar and  $v_i \in \mathbb{C}^{n_x}$  is a complex  $n_x \times 1$  vector. Since we are interested in physical systems, we restrict our attention to real square matrices  $A \in \mathbb{R}^{n_x \times n_x}$ .

- $\lambda_i$  and  $v_i$  can be complex when  $A$  is real.
- When  $A$  is real,  $\lambda_i$  and  $v_i$  will either be both real or both complex.
- When  $A$  is real,  $\lambda_i$  will always come in complex conjugate pairs. A real matrix  $A$  should never have an odd number of complex eigenvalues, and these eigenvalues should be symmetric about the real axis when viewed in the s-plane.

## Finding Eigenvalues

$$\det(\lambda_i I - A) = 0$$

The computation forms a  $n_x^{\text{th}}$ -order polynomial in  $\lambda_i$  when  $n_x$  roots - each root representing possibly repeated eigenvalues. The eigenvector corresponding to an eigenvalue represents the direction associated with the null-space of the matrix  $(\lambda_i I - A)$ . The eigenvectors associated with an eigenvalue can be used to span the null space of  $\lambda_i I - A$ .

**Relating Eigenvalues and Poles**  $\det(sI - A)$  is the characteristic polynomial of the system, the roots of which are both the poles and eigenvalues of the system. This also means the eigenvalues and poles are not only dependent on  $A$  and are thus independent of the input or output.

## 2.4 Mechanics (Newtonian)

Understanding the physics that inform the behavior of a system is critical to modeling and controlling it.

**Newton's 1st Law** A body remains at rest, or in motion at a constant speed in a straight line, unless it is acted upon by a force

**Newton's 2nd Law** At any instant of time, the net force on a body is equal to the body's acceleration multiplied by its mass or, equivalently, the rate at which the body's momentum is changing with time.

**Newton's 3rd Law** If two bodies exert forces on each other, these forces have the same magnitude but opposite directions.

### 2.4.1 Linear

#### Position / Velocity / Acceleration

**Example: Mass-Spring-Damper System** The classical example of a physical linear system is the mass-spring-damper system.

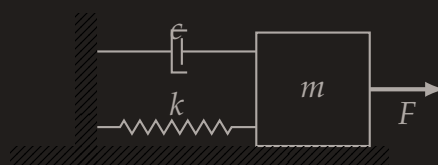


Figure 3: MSD System

Applying Newton's Laws, the governing equation can be derived.

$$\sum F_x = F(t) - c\dot{x} - kx = m\ddot{x} \quad (2.11)$$

where  $F(t)$  is the forcing function (force applied),  $k$  is the spring constant,  $c$  is the damper constant, and  $m$  is the mass of the block.

## Momentum

### 2.4.2 Rotational

**Angular Position / Velocity / Acceleration** An analog to the linear system can be made for a rotational system.

## Angular Momentum

**Coriolis Force** The presense of rotation adds an additional (faux) force. This is described further in (7.1.5).

### 2.4.3 Energy

### 2.4.4 Electrical

Electrical circuits can be modeled very similarly to the mechanical system. The classic example is RLC (resistor-inductor-capacitor) circuits.

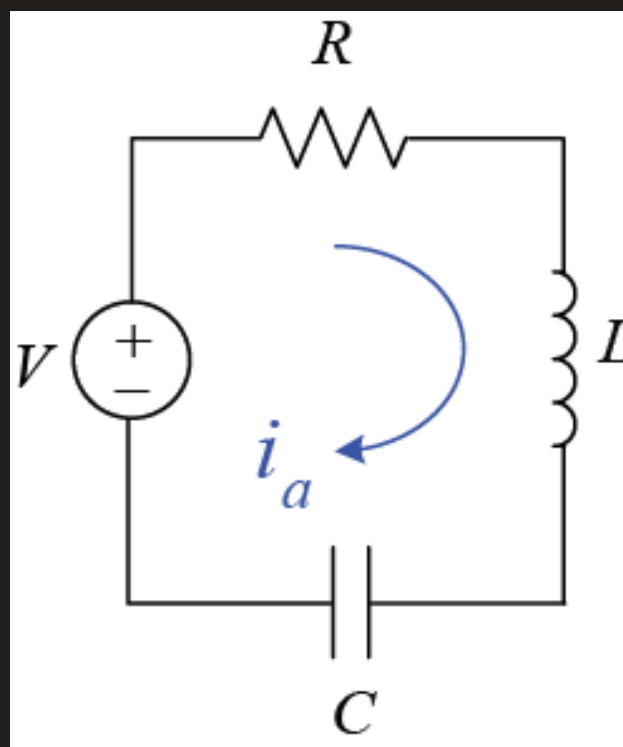


Figure 4: RLC System

Applying the Kirchhoff's Current and Voltage Laws, the governing equation can be derived.

$$V(t) - Ri - L \frac{di}{dt} - \frac{1}{C} \int i dt = 0 \quad (2.12)$$

where  $V(t)$  is the forcing function,  $R$  is the resistance,  $L$  is the inductance, and  $C$  is the capacitance.

## 2.5 Lagrangian Mechanics

Lagrangian mechanics are preferable to use for dynamical systems that have multiple components whose interaction is complicated to express (eg. multi-body systems). This is especially true if internal reactions are not of immediate interest.

$$\text{Action: } S = \int_{t_1}^{t_2} L(q, \dot{q}) dt \quad (2.13)$$

Where  $L(q, \dot{q})$  is the Lagrangian and  $q \in \mathbb{R}^{n_q}$  is a vector of generalized degrees of freedom (positions, angles).  $\dot{q}$  is the vector of corresponding rates of change of the degrees of freedom (linear + angular velocities).

$$L(q, \dot{q}) = T(q, \dot{q}) - V(q) \quad (2.14)$$

$T(q, \dot{q})$  is the kinetic energy function,  $V(q)$  is the potential energy function. Thus the Lagrangian is the difference between the system's total kinetic and total potential energy.

**Principle of Least Action** A conservative system (one that does not gain or lose energy) starting in state  $q(t_1)$  at time  $t_1$  and ending at state  $q(t_2)$  at time  $t_2$  follows a path that makes the action  $S$  stationary (first derivative goes to zero). This is the path of least action. While the path of least action minimizes the action the vast majority of the time, the stationary condition does not guarantee minimization.

**Euler-Lagrange Equations for Conservative Systems** The path of least action can be thought of as a continuous stream of states  $q(t)$  over the time interval. It must satisfy the Euler-Lagrange Equations.

$$\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_j} \right) - \frac{\partial L}{\partial q_j} = 0, \quad j \in \{1, \dots, n_q\} \quad (2.15)$$

This yields  $n_q$  equations of motion, one corresponding to each of the generalized degrees of freedom. This we obtain all the equations of motion necessary to describe a conservative (energy-preserving) dynamical system.

**Euler-Lagrange Equations for Non-conservative Systems** We can modify Eq. 2.15 to account for energy dissipation (eg. friction) and energy (eg. forcing).

$$\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_j} \right) - \frac{\partial L}{\partial q_j} = Q_j - \frac{\partial R}{\partial \dot{q}_j} \quad (2.16)$$

where  $Q_j := Q_j(t)$  is the forcing function and  $R := R(\dot{q})$  is the energy dissipation function. If we plug in the definition for  $V(q)$  from the Least Action Principle, we arrive at:

$$\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}_j} \right) + \frac{\partial R}{\partial \dot{q}_j} + \left( \frac{\partial V}{\partial q_j} - \frac{\partial T}{\partial q_j} \right) = Q_j \quad (2.17)$$

This is the equation of motion for each degree of freedom.

**Example** We can use the classic example of the mass-spring-damper system. The advantages of the Lagrangian method become clear when an inverted pendulum is added to the system.

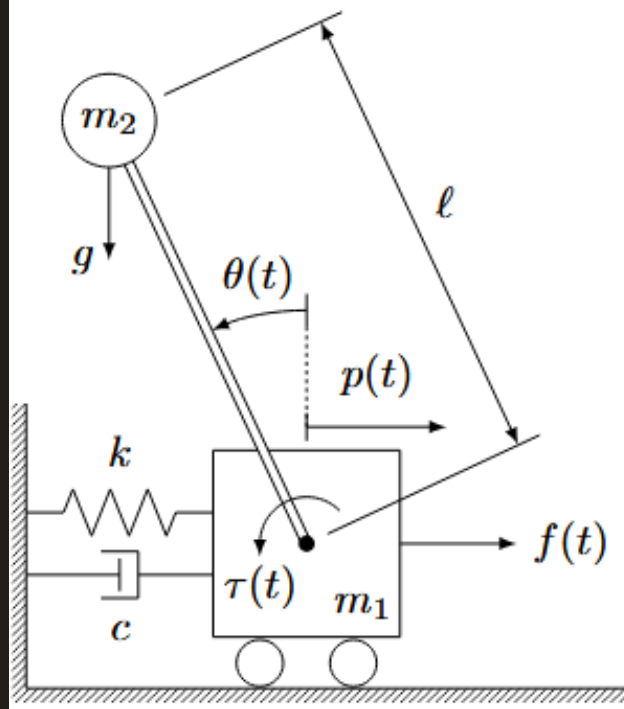


Figure 5: Inverted Pendulum on Cart System

With two bodies (masses) in the system, we can describe the degrees of freedom and the velocities of each as:

$$\begin{aligned} q_1(t) &:= p(t), & \dot{q}_1(t) &= \dot{p}(t), \\ q_2(t) &:= \theta(t), & \dot{q}_2(t) &= \dot{\theta}(t), \\ Q_1(t) &:= f(t), & Q_2(t) &:= \tau(t), \end{aligned}$$

$$v_1 = \begin{bmatrix} \dot{p} \\ 0 \end{bmatrix}, \quad v_2 = \begin{bmatrix} \dot{p} - \ell \dot{\theta} \cos(\theta) \\ -\ell \dot{\theta} \sin(\theta) \end{bmatrix}$$

Now, the components of the Euler-Lagrange equation (Eq. 2.17) can be found to assemble

the complete equations of motion of the system.

$$T_i = \frac{1}{2} m_i v_i^T v_i$$

$$T_1 = \frac{1}{2} m_1 (\dot{p}^2 + 0^2), \quad T_2 = \frac{1}{2} m_2 [(\dot{p} - \ell \dot{\theta} \cos(\theta))^2 + (-\ell \dot{\theta} \sin(\theta))^2]$$

The sum of the kinetic energy of the system is the sum of the kinetic energy of both masses:

$$\begin{aligned} T &= T_1 + T_2 \\ &= \frac{1}{2} m_1 \dot{p}^2 + \frac{1}{2} m_2 [\dot{p}^2 - 2\ell \cos(\theta) \dot{p} \dot{\theta} + \ell^2 \dot{\theta}^2] \end{aligned}$$

We can then differentiate the above equation with respect to the generalized velocities  $\dot{q}_i$ . This gives us the following partial derivatives:

$$\frac{\partial T}{\partial \dot{q}_1} = (m_1 + m_2) \dot{p} - m_2 \ell \dot{\theta} \cos(\theta) \quad \text{and} \quad \frac{\partial T}{\partial \dot{q}_2} = m_2 \ell [\ell \dot{\theta} - \cos(\theta) \dot{p}]$$

Differentiating again, with respect to time:

$$\frac{d}{dt} \left( \frac{\partial T}{\partial \dot{q}_1} \right) = (m_1 + m_2) \ddot{p} - (m_2 \ell) (\ddot{\theta} \cos(\theta) + \sin(\theta) \dot{\theta}^2) \quad \text{and} \quad \frac{d}{dt} \left( \frac{\partial T}{\partial \dot{q}_2} \right) = m_2 \ell (\ell \ddot{\theta} - \ddot{p} \cos(\theta) - \dot{p} \sin(\theta))$$

Then, we also differentiate the total kinetic energy equation with respect to position:

$$\frac{\partial T}{\partial q_1} = 0 \quad \text{and} \quad \frac{\partial T}{\partial q_2} = m_2 \ell \sin(\theta) \dot{\theta} \dot{p}$$

There are two sinks/sources of potential in the system. One is the spring, and the other is the gravitational potential of  $m_2$ . Thus, the total potential energy of the system is:

$$V = \frac{1}{2} k q_1 + m_2 g \ell \cos(q_2)$$

This equation is differentiated with respect to position:

$$\frac{\partial V}{\partial q_1} = k p \quad \text{and} \quad \frac{\partial V}{\partial q_2} = -m_2 g \ell \sin(\theta)$$

There is one source of energy dissipation in the system: the damper. The energy dissipated by this component is given by:

$$R = \frac{1}{2} c \dot{p}^2$$

Differentiating this with respect to velocity:

$$\frac{\partial R}{\partial \dot{q}_1} = c \dot{p} \quad \text{and} \quad \frac{\partial R}{\partial \dot{q}_2} = 0$$



Now that all of the derivatives have been solved for, the nonlinear ODEs can be plugged into the Euler-Lagrange Equation, resulting in the coupled equations:

$$\begin{aligned} f &= (m_1 + m_2)\ddot{p} - m_2\ell \cos(\theta)\ddot{\theta} + m_2\ell \sin(\theta)\dot{\theta}^2 + c\dot{p} + kp \\ \tau &= m_2\ell^2\ddot{\theta} - m_2\ell \cos(\theta)\ddot{p} - m_2g\ell \sin(\theta) \end{aligned}$$

The coupled differential equations can be rearranged to isolate  $\ddot{p}$  and  $\ddot{\theta}$ . In order to do this, the constants  $\alpha(\theta)$  and  $\beta(\theta)$  are defined as follows:

$$\alpha(\theta) := \frac{\cos(\theta)}{\ell} \quad \text{and} \quad \beta(\theta) := \frac{m_2\ell \cos(\theta)}{m_1 + m_2}$$

These constants are used to eliminate the second derivative term in the coupled differential equations. Through some algebraic manipulation, it can be shown that the equations with the isolated second derivative are given by:

$$\begin{aligned} \ddot{p} &= \frac{m_2\ell \sin(\theta)[\alpha(\theta)g - \dot{\theta}^2] - c\dot{p} - kp + f + \alpha(\theta)\tau}{m_1 + m_2[1 - \alpha(\theta)\ell \cos(\theta)]} \\ \ddot{\theta} &= \frac{m_2\ell \sin(\theta)[g - \beta(\theta)\dot{\theta}^2] + \beta(\theta)[f - c\dot{p} - kp] + \tau}{m_2\ell[\ell - \beta(\theta) \cos(\theta)]} \end{aligned}$$

These are the final equations of motion, without solving for the reaction force of the pendulum on the cart!

### 3 System Identification & Analysis

Before we can understand how to control a system, we must first understand how that system changes in response to input. System identification is the process by which a mathematical model of the dynamical system can be constructed. Designing a controller is infinitely easier with such a model.

The system (excluding the controller) is referred to as the "plant".

#### 3.1 Linear Regression

#### 3.2 State-space Models

State space models are a useful way to describe a system. A typical LTI system can be arranged as follows:

$x(t)$  – state     $y(t)$  – output     $u(t)$  – input

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

$$y(t) = Cx(t) + Du(t)$$

$A \in \mathbb{R}^{n_x \times n_x}$     zero input dynamics matrix

$B \in \mathbb{R}^{n_x \times n_u}$     input matrix

$C \in \mathbb{R}^{n_y \times n_x}$     output matrix

$D \in \mathbb{R}^{n_y \times n_u}$     pass-through matrix

#### 3.3 Linear, Time-Invariant Systems (LTI)

##### 3.3.1 Properties

**Homogeneity**  $\begin{cases} x(t_0), u(t) \rightarrow y(t) \\ \alpha x(t_0), \alpha u(t_0) \rightarrow \alpha y(t) \end{cases}$

**Additivity**  $\begin{cases} x_1(t_0), u_1(t) \rightarrow y_1(t) \\ x_2(t_0), u_2(t) \rightarrow y_2(t) \end{cases} \Rightarrow x_1(t_0) + x_2(t_0), u_1(t) + u_2(t) \rightarrow y_1(t) + y_2(t)$

**Time Invariance** For the same input, the response at an time will be the same.

##### 3.3.2 Convolution

The uniformity and superposition allows us to use the response to a known unput to comput the response to a generic input. Convolution is the mathematical operation by which we generate the response to said generic input. It's only because of the time uniformity and superposition that convolution is useful and gets its own name (unlike other mathematical constructs).

In the time domain the definition is:

$$y(t) = [h * u](t) = \int_0^t h(\tau)u(t - \tau)d\tau = \int_0^t h(t - \tau)u(\tau)d\tau \quad (3.1)$$

Where  $h(t)$  is the impulse response of the LTI system impulse response is the response to a unit impulse  $\delta(t)$ .

$$\delta(t) = \begin{cases} \infty, & \text{at } t = 0 \\ 0 & \text{otherwise} \end{cases} \quad (3.2)$$

Assuming zero initial conditions:

$$h(t) = Ce^{At}B + D\delta(t)$$

The impulse response is by definition a time domain concept, as it is defined as the time domain output of an LTI system to a unit time domain impulse in the input.

**Visualization of Convolution** The input can be interpreted as a sequence of impulses offset in time, each setting off its own impulse response that is scaled in magnitude by the magnitude of  $u(t)$  at said time. The output can be understood as the sum of the resulting scaled and time shifted impulse responses.

For any initial condition the output  $y(t)$  of an LTI system due to a sinusoidal input  $u(t)$  will converge to a steady state output  $y_{ss}(t)$  as  $t \rightarrow \infty$ , assuming a steady state output exists. The output will be a sinusoid of the same frequency as  $u(t)$ , with a different magnitude and phase shift. The magnitude and phase shift can be gleaned from the transfer function (specifically from the Bode plot).

Each block in a control diagram represents a convolution. Thus, we need a more intuitive way to perform the operation, which can be done in the frequency domain, which can be done with the Laplace transform.

### 3.4 Linear, Time-Varying Systems (LTV)

For an LTV system, the system is time-varying because at least one of the four system matrices changes with time, not because  $x, y, u$  change with time! The primary shortcoming of LTV systems is their dependence on the initial time. An input and initial condition might have different responses over different time intervals. We cannot guarantee homogeneity and additivity over shifted time intervals.

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

$$y(t) = C(t)x(t) + D(t)u(t)$$

$$A(t) \in \mathbb{R}^{n_x \times n_x} \quad B(t) \in \mathbb{R}^{n_x \times n_u} \quad C(t) \in \mathbb{R}^{n_y \times n_x} \quad D(t) \in \mathbb{R}^{n_y \times n_u}$$

#### 3.4.1 Properties

$$\text{Homogeneity} \quad \begin{cases} x(t_0), u(t) \rightarrow y(t) \\ \alpha x(t_0), \alpha u(t) \rightarrow \alpha y(t) \end{cases}$$

$$\text{Additivity} \quad \begin{cases} x_1(t_0), u_1(t) \rightarrow y_1(t) \\ x_2(t_0), u_2(t) \rightarrow y_2(t) \end{cases} \Rightarrow x_1(t_0) + x_2(t_0), u_1(t) + u_2(t) \rightarrow y_1(t) + y_2(t)$$

Together, these two properties are called superposition. These properties do not hold under time shifts, however.

**State Transition Matrix** Governs zero-input dynamics.

$$\Phi_A(t_0, t_0) = I_{n_x \times n_x}, \quad \dot{\Phi}_A(t_0, t_0) = A(t_0)\Phi_A(t_0, t_0)$$

### 3.5 Modal Analysis

Eigenvalues (2.3.3) are critical for understanding how a system reacts to input. A mode of a system is the time response associated with either a simple eigenvalue and associated eigenvector, or a complex conjugate pair of eigenvalues + eigenvectors. Maximum number of modes is  $n_x$ , and minimum is  $n_x/2$  (which is if all eigenvalues are complex conjugate pairs.)

$$\dot{x}(t) = Ax(t)$$

If we define the initial conditions as:

$$\begin{aligned} x(0) &= v_i \text{ where } v_i \text{ is an eigenvector of } A \\ \dot{x}(0) &= Ax(0) = Av_i = \lambda_i v_i \end{aligned}$$

Since  $\lambda$  is a scalar, this equation implies the state of the system will evolve parallel to  $v_i$ . This means that  $\dot{x}(t)$  will always remain aligned with  $v_i$ , and  $x(t)$  will never come out of alignment.

#### Observations

- The real part of the eigenvalue  $\lambda_i$  dictates the exponential growth or decay of  $x_{mj}(t)$ .
- The imaginary part of eigenvalue  $\lambda_i$  dictates the frequency with which  $x_{mj}(t)$  oscillates.
- The eigenvector  $v_i$  encodes the relative phase and magnitude of each state in  $x(t)$ . The phase information is encoded by the relative phase of the complex elements of the vector  $v_i$ .
- The scalar  $\alpha_0$  determines the initial scaling and phase shift of all the states in  $x(t)$ .

$$X_{mj}(t) = \alpha_0 v_i e^{\lambda_i t} = \alpha_0 v_i e^{\text{Re}(\lambda_i)t} [\cos(\text{Im}(\lambda_i)t) + j \sin(\text{Im}(\lambda_i)t)]$$

Any zero input motion that a given LTI system can exhibit can be expressed as a superposition of the system modes.

#### Example: Conventional Aircraft Modes

Phugoid

Dutch Roll

Subsidence

**Example: System Modes** Take the following 2D mass-spring-damper system:

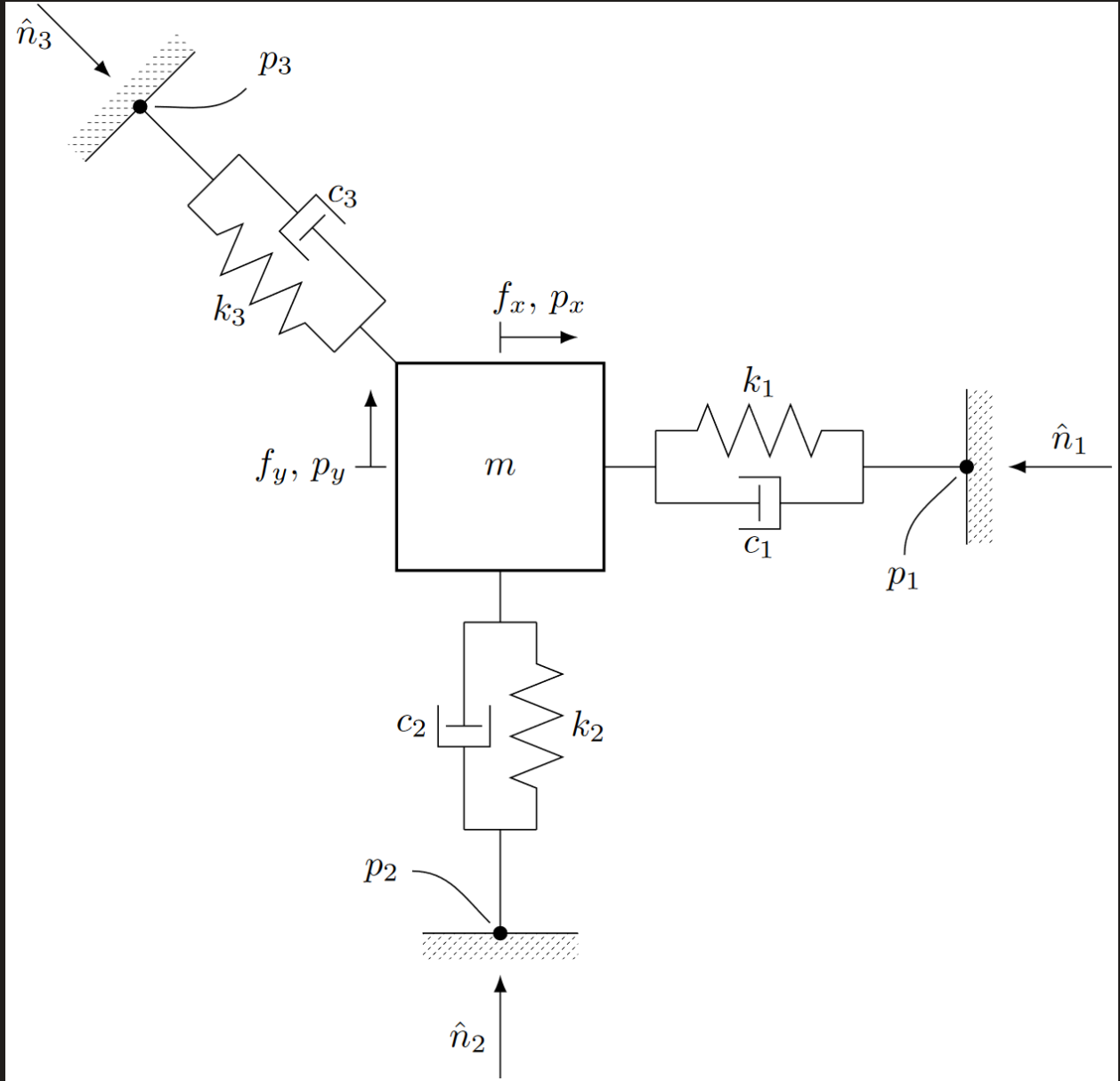


Figure 6: 2D Nonlinear Mass-Spring-Damper System

We can define the state, control input, and desired output vectors as follows:

$$x(t) = \begin{bmatrix} p_x \\ p_y \\ \dot{p}_x(t) \\ \dot{p}_y(t) \end{bmatrix}, \quad u(t) = \begin{bmatrix} f_x \\ f_y \end{bmatrix}, \quad y(t) = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$$

For this system, gravity can be neglected. We can define variables as follows:  $\delta_i$  is the displacement of each mass/damper pair,  $\dot{\delta}_i$  is the rate of displacement of each mass/damper pair,  $\hat{n}_i$  is the unit vector pointing from the attachment point of each mass/damper pair towards the position of the mass. The variables  $k_i$  and  $c_i$  are the spring and damper constants respectively, where  $i$  represents each mass/damper grouping.

In order to find the function describing the dynamics of the system, the state vector is differentiated, because  $\dot{x}(t) = f(x(t), u(t))$ . The second derivative of the position of the mass,  $\ddot{p}$ , can be found using Newton's Laws of Motion, as seen in Eq. 3.3.

$$\dot{x}(t) = \begin{bmatrix} \dot{p}_x(t) \\ \dot{p}_y(t) \\ \ddot{p}_x(t) \\ \ddot{p}_y(t) \end{bmatrix} \Rightarrow \begin{bmatrix} \dot{p}_x(t) \\ \dot{p}_y(t) \\ \sum[k_i \cdot \delta_i \cdot \hat{n}_{x,i}] + \sum[c_i \cdot \dot{\delta}_i \cdot \hat{n}_{x,i}] + f_x \\ \sum[k_i \cdot \delta_i \cdot \hat{n}_{y,i}] + \sum[c_i \cdot \dot{\delta}_i \cdot \hat{n}_{y,i}] + f_y \end{bmatrix} \quad (3.3)$$

Next, the equations of the system can be derived by linearizing about the equilibrium point, represented by  $\bar{x}(t)$  and  $\bar{y}(t)$ . The linearization is done by holding one variable constant about the equilibrium point while the derivative with respect to the other is computed. The system matrices calculated (numerical Jacobian) from the initial system variables given are:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -2.5 & 1.5 & 0 & 0 \\ 1.5 & -3.5 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

MATLAB's built in `eig()` function can be used to find the eigenvalues and corresponding eigenvectors. The eigenvalues and their corresponding eigenvectors are found below. Each row in the eigenvalue vector corresponds to the same indexed column in the eigenvector matrix.

$$\lambda = \begin{bmatrix} -0 + 2.1404i \\ -0 - 2.1404i \\ 0 + 1.1912i \\ 0 - 1.1912i \end{bmatrix}, \quad v = \begin{bmatrix} 0 + 0.2475i & 0 - 0.2475i & 0 - 0.5216i & 0 + 0.5216i \\ 0 - 0.3434i & 0 + 0.3434i & 0 - 0.3760i & 0 + 0.3760i \\ -0.5297 - 0i & -0.5297 + 0i & 0.6213 + 0i & 0.6213 + 0i \\ 0.7350 + 0i & 0.7350 + 0i & 0.4478 + 0i & 0.4478 - 0i \end{bmatrix}$$

The A matrix has two complex conjugate pairs of eigenvectors, meaning that the system has two modes. The modes are found by taking the real part of the complex eigenvectors and plugging them in as the initial state. The first two eigenvectors correspond to the direction seen in Fig. 7a, and the second two correspond to Fig. 7b. Both modes oscillate, although the first mode oscillates faster. Neither exponentially grows for two reasons: there is no forcing input applied, and the absence of damping results in a purely imaginary eigenvalue, which means motion is purely oscillatory with no gain. Neither exponentially decays for the same reason. The relationship between the modes and the damping coefficients is such that when the system is undamped, it is in a state of pure oscillation, and thus the eigenvalues are purely imaginary. Adding a damping coefficient adds a real component to the eigenvalues, which represents the damping effect. Moreover, the damped natural

frequency will be less than that of an undamped system.

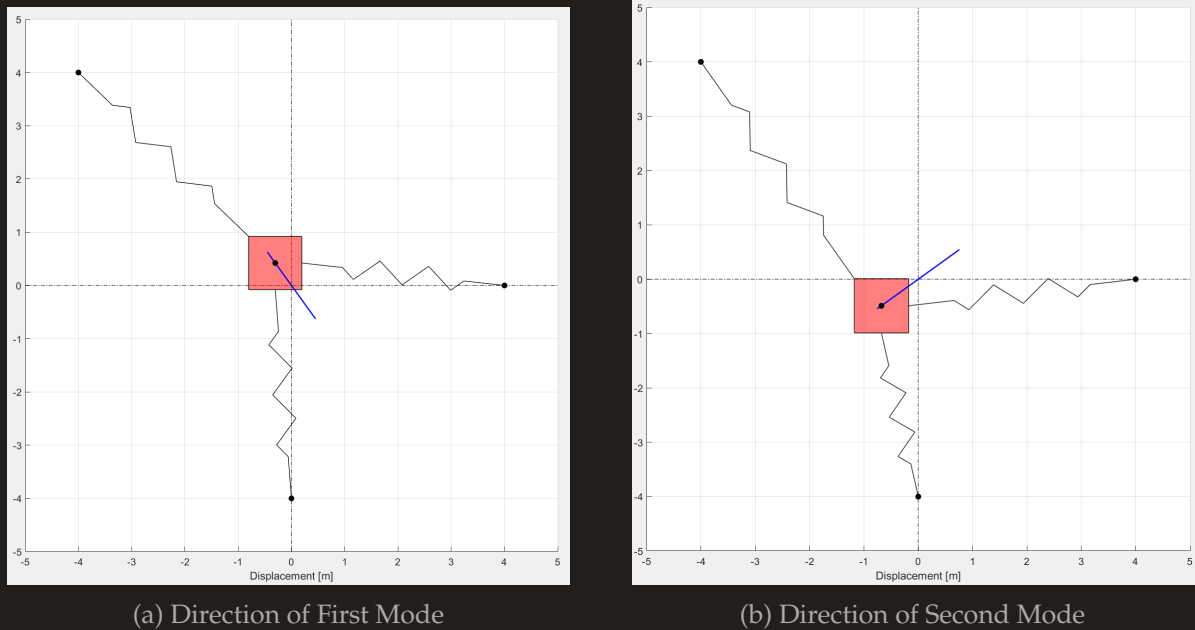


Figure 7: System Modes

In order to excite the system along its mode, the eigenvectors corresponding to either eigenvalue pair can be plugged into the function as an initial state  $x(0)$ . It can be observed that the linear simulations do not match the nonlinear ones. This is due to the imprecision of the linearization operation performed; The farther the mass travels from the equilibrium point, the farther the nonlinear simulation diverges from the linear one. This is because the system matrices derived through linearization are only very accurate around the point about which they were linearized, that being the equilibrium point.

## Natural Frequency

### 3.5.1 System Poles

#### Example: System Poles

### 3.5.2 System Zeros

The presence of a zero in an LTI system is an indication that the system can exhibit zero output even when the state and control input are nonzero. Intuitively: the presence of a zero implies that there are special combinations of non-zero initial conditions and non-zero control input that produce non-zero state time histories and zero output time histories.

$$x(0) \neq 0, u(t) \neq 0 \Rightarrow x(t) \neq 0, y(t) = 0$$

Components of the initial conditions and control input may combine and vanish, resulting in no perceptible contribution to the input. It follows that removing said components from the initial conditions and control input would generate the same output  $y(t)$ .

Not all LTI systems have zeros. Zeros are properties of the system at hand. Unlike poles,

zeros do not influence the BIBO (3.6.1) stability of the system. The BIBO stability is determined by the location of its poles, even in cases that even have zeros in the right half plane.

**Formal Definition of a Zero** Goal: find non-zero initial conditions and control input so that the output is zero for all time.

$$\zeta(t) := \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \neq 0 \Rightarrow y(t) = 0$$

**Minimum Phase vs. Non-minimum Phase Zeros** Minimum phase zeros are zeros that have non-positive real parts. Non-minimum phase zeros are zeros that have positive real parts. The naming convention follow from the Bode plot. If both are arranged symmetrically about the  $j\omega$  axis, they will generate the same Bode magnitude plot, but the minimum phase plot will lie below that of the non-minimum phase plot.

To build intuition on the effect minimum and non-minimum phase zeros have on a system, consider an MSD system, with transfer function:

$$T(s) = \frac{Y(s)}{U(s)} = \frac{s \pm z}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

which exhibits two stable poles and one zero, which is a minimum if the + sign is used, and non-minimum if the - sign is used. We can rearrange the above equation and take its inverse Laplace transform to obtain a time-domain representation of the system.

$$\dot{u}(t) \pm zu(t) = f(t) = \ddot{y}(t) + 2\zeta\omega_n\dot{y}(t) + \omega_n^2 y(t)$$

We have introduced  $f(t)$  to represent the forcing function. In this representation we can understand the zeros as determining the input dynamics that relate the control input  $u(t)$  to the forcing function  $f(t)$ . Likewise, the poles can be understood as determining the output dynamics that relate the forcing function  $f(t)$  to  $y(t)$ .

**Example: System Zero** To demonstrate a system zero, take the example of a triple mass-spring-damper system, in which the input is the forcing applied to  $m_1$ . Defining the output as  $p_2$ , the input that results in a zero output (motion) in  $m_2$  can be found.

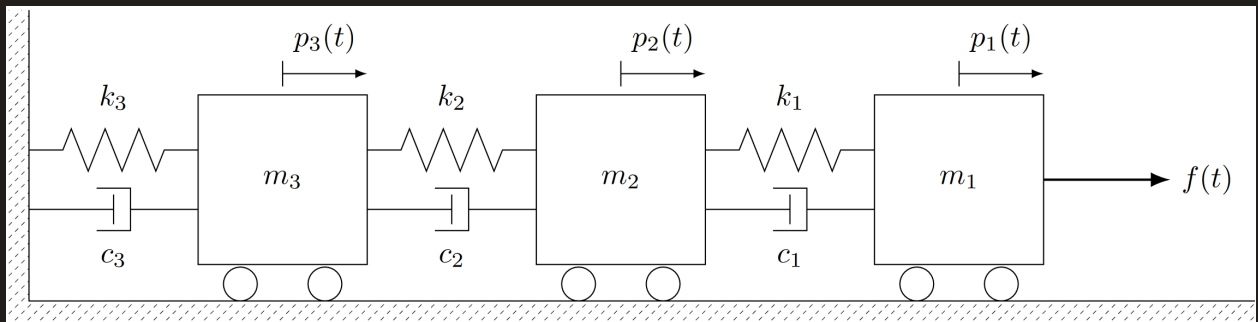


Figure 8: Triple Mass-Spring-Damper System



The equations of motion (derived from Newton's laws) are as follows:

$$\begin{aligned} m_1 \ddot{p}_1(t) &= k_1(p_2(t) - p_1(t)) + c_1(\dot{p}_2(t) - \dot{p}_1(t)) + f(t) \\ m_2 \ddot{p}_2(t) &= k_1(p_1(t) - p_2(t)) + k_2(p_3(t) - p_2(t)) + c_1(\dot{p}_1(t) - \dot{p}_2(t)) + c_2(\dot{p}_3(t) - \dot{p}_2(t)) \\ m_3 \ddot{p}_3(t) &= k_2(p_2(t) - p_3(t)) - k_3 p_3(t) + c_2(\dot{p}_2(t) - \dot{p}_3(t)) - c_3 \dot{p}_3(t) \end{aligned}$$

These equations of motion can then be arranged into the LTI state-space matrix form, giving the following state matrices.

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -k_1/m_1 & k_1/m_1 & 0 & -c_1/m_1 & c_1/m_1 & 0 \\ k_1/m_2 & -(k_1 + k_2)/m_2 & k_2/m_2 & c_1/m_2 & -(c_1 + c_2)/m_2 & c_2/m_2 \\ 0 & k_2/m_3 & -(k_2 + k_3)/m_3 & 0 & c_2/m_3 & -(c_2 + c_3)/m_3 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{m_1} \\ 0 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \end{bmatrix}$$

The system zeros can easily be found by converting the state space system into a zero-pole-gain form using MATLAB's `zpk()` function. The poles of the system are compared to the zeros to establish which pole-zero pairs cancel, and which pole will lead to the desired system behavior. None of the zeros overlap a pole, yet the first zero has only a real component, indicating a lack of oscillation. The second and third zeros are conjugate pairs with only imaginary components. The second zero  $s_0 = 0 + 1.4142i$  was used for the following computations.

The initial condition vector was computed by setting up a generalized eigenvalue problem matrix using the system matrices.

$$A_z = \begin{bmatrix} s_0 I - A & -B \\ C & D \end{bmatrix}$$

The null space of the  $A_z$  matrix corresponds to the initial state  $x_0$  and forcing input  $u_0$ . The values calculated for these variables is as follows:

$$x_0 = \begin{bmatrix} -0.0098 \\ 0.0000 \\ 0.4964 \\ -0.0973 \\ 0.0000 \\ -0.0003 \end{bmatrix}, \quad u_0 = \begin{bmatrix} -0.4768 - 0.1378i \end{bmatrix}$$

The system is then solved using MATLAB's `ode45()` function. The initial state  $x_0$  and input  $u_0$  can be plugged into the system to confirm that  $m_2$  demonstrates zero motion.

### 3.5.3 *System Modes for Second-order Systems*

## 3.6 **System Stability**

### 3.6.1 *Bounded Input, Bounded Output Stability*

### 3.6.2 *Routh-Hourwitz Criterion*

### 3.6.3 *Nyquist Criterion*

## 3.7 **Sensitivity**

### 3.7.1 *Cosensitivity*

## 4 Linear Control Methods

### 4.1 Open Loop

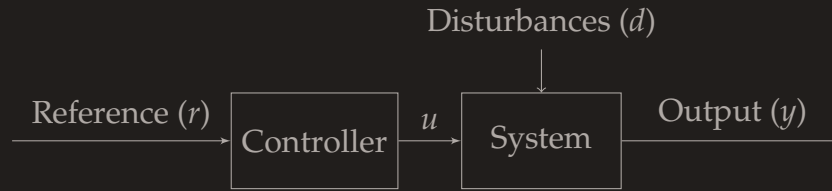


Figure 9: Block Diagram of Open Loop System

The system identification reveals how the inputs affect the outputs and the state of the system. We can work backwards from the desired output to find the required input.

**Example** In the context of a car's cruise control, we would like to achieve a desired speed with our throttle input. With a model of the car, we can determine the effect of the factors at play in the system: Throttle response, engine force, drag force, tire friction, etc. Treating the system as an equation and solving through, the throttle input required can be solved.

### 4.2 Closed Loop (Feedback)

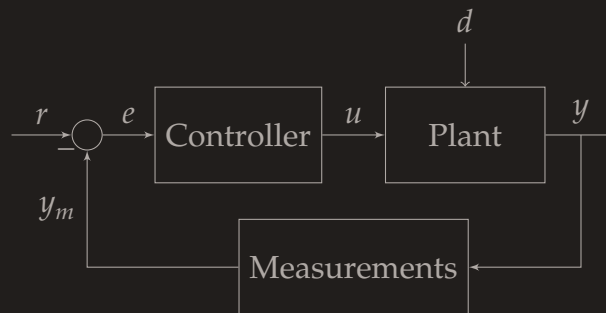


Figure 10: Block Diagram of Feedback Loop

**Example** Returning to the example of the cruise control, imagine that the same system (plant + controller) encounters a hill. Climbing up the slope changes the effect of the gravity force, meaning that our previous throttle input is not longer bringing the desired results. If a feedback system is implemented, the error between the desired speed and the actual speed can be calculated. With this information available, we can design our controller to account for disturbances, such as that hill.

**Disturbance Modeling** In between the plant and the output, disturbances in the system are modeled. An example of a disturbance is the hill previously discussed, but it can also include modeling uncertainties / inaccuracies in the plant.

### 4.2.1 Purpose

**Note:** Feedback control systems may be myopic! In many designs, they do not have the information or sophistication to understand that current decisions might make control of the vehicle impossible later on.

## 4.3 Bode Plots

The Bode plot is way to visualize the transfer function of a system. The Bode plot is a pair of subplots that show the magnitude and phase of a transfer function evaluated for  $s - i\omega$  over the interval  $\omega \in (0, \infty)$ . In other words, it is the 3D magnitude and phase surface sliced along the positive  $i\omega$ -axis. Bode plots are helpful for understanding the steady-state behavior, as it shows the amplification factor and phase shift that the system will exhibit relative to a sinusoidal persistent input.

Bode plots are shown on logarithmic scales, using the product property for ease.

### Logarithm Product Property

$$\log(a \cdot b) = \log(a) + \log(b)$$

**Geometry of Transfer Functions** Transfer functions are usually expressed as fractions of polynomials.

$$T(s) = k \cdot \frac{\prod_{i=1} (s - z_i)}{\prod_{i=1} (s - p_i)}$$

Splitting up these complex numbers using Euler's formula (2.1), we can separate the magnitude and phase of the transfer function.

$$\begin{aligned} \text{Magnitude: } |T(s)| &= |k| \cdot \frac{\prod_{i=1} |s - z_i|}{\prod_{i=1} |s - p_i|} \\ \text{Phase: } \angle T(s) &= \angle k + \sum_{i=1} \angle(s - z_i) - \sum_{i=1} \angle(s - p_i) \end{aligned}$$

These are then plotted on a log scale, after taking the base-10 logarithm and multiplying by 20.

$$20 \log_{10} |T(s)| = 20 \log_{10} |k| + 20 \sum_{i=1} \log_{10} |s - z_i| - 20 \sum_{i=1} \log_{10} |s - p_i|$$

## 4.4 PID Control

The PID controller is one of the simplest types of controllers.

**Proportional Gain (P term)** Applies a control input proportional to the amount of error.

**Integral Gain (I term)** Applies a control input proportional to the integral of error (sum of error over time). This term is useful for countering steady-state error.

**Derivative Gain (D term)** Applies a control input proportional to the rate of change of the error. This term is useful for countering overshoot.

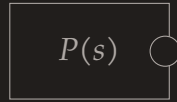


Figure 11: Typical Control Block Diagram for Feedback System

The PID controller can also be represented as a transfer function.

$$K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s} \quad (4.1)$$

#### 4.4.1 Gain Margin

##### Amplification Gain Margin

##### Attenuation Gain Margin

#### 4.4.2 Phase Margin

#### 4.4.3 Crossover Frequency

##### Gain Crossover Frequency

##### Phase Crossover Frequency

#### 4.4.4 Integrator Anti-windup Term

In certain cases where the error does not correct rapidly, the I term may "wind up" (error over time builds up, resulting in large corrections). The large correction applied may destabilize the system. To counter this, an "anti-windup" term can be used, to limit the sum of error the integral term sees.

#### Example

### 4.5 Higher-order Controllers

#### 4.5.1 Lead-Lag Control

### 4.6 Feedforward

Feedforward control can address some of the drawbacks of feedback control, such as the case of myopia (4.2.1). For this reason, feedforward is sometimes referred to as guidance.

### 4.7 Multi-Input, Multi-Output (MIMO)

So far, all of the techniques have addressed mainly single input, single output systems (SISO).

### 4.8 Designing the Controller

#### 4.8.1 Frequency Response Method

#### 4.8.2 Root Locus Method

#### 4.8.3 Loop-Shaping Method

Loop-shaping is somewhat of an enhancement to the frequency response method, using the Nyquist plot in addition to the Bode plot.

#### 4.8.4 Examples

**PID Tuning** Take Figure 11 as the block diagram for the example system for which the PID controller is being tuned.

A PID controller is designed to satisfy the following requirements: A loop gain crossover frequency of 1 Hz, an amplification gain margin of 6dB or better, an attenuation gain margin of 10dB or better, and a phase margin of  $45^\circ$  or better.

## 5 Nonlinear Methods

A nonlinear plant is one that includes nonlinear functions in its dynamics (eg. powers, sine, cosine). Refer to (2.2.3). Many of the assumptions that allow us to solve linear systems do not apply. For example, superposition is generally not true for a nonlinear system. A typical nonlinear dynamical system:

$$\begin{aligned}\dot{x}(t) &= f(t, x(t), u(t)) \\ y(t) &= g(t, x(t), u(t))\end{aligned}$$

where  $t \in \mathbb{R}$  is monotonically increasing.  $f$  defines the dynamics of the system (nonlinear).  $g$  defines the output (measured quantities).

**Example** A classic example is the two dimensional mass-spring-damper system. No forces are present in the springs and dampers when  $p_x = p_y = \dot{p}_x = \dot{p}_y = 0$ . Gravity is neglected for this problem.

$$x = \begin{bmatrix} p_x \\ p_y \\ \dot{p}_x \\ \dot{p}_y \end{bmatrix}, \quad u = \begin{bmatrix} f_x \\ f_y \end{bmatrix}, \quad y = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$$

Defining  $\delta_p$  as the spring displacement, the equations of motion are therefore

$$f(t, x, u) = \begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \sum_{i=1}^3 \frac{k_i}{m} \delta_{p,i} \hat{n}_{x,i} + \sum_{i=1}^3 \frac{c_i}{m} \delta_{\dot{p},i} \hat{n}_{x,i} + \frac{f_x}{m} \\ \sum_{i=1}^3 \frac{k_i}{m} \delta_{p,i} \hat{n}_{y,i} + \sum_{i=1}^3 \frac{c_i}{m} \delta_{\dot{p},i} \hat{n}_{y,i} + \frac{f_y}{m} \end{bmatrix}$$

Assuming that the mass has negligible dimension, and defining  $p = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$ :

$$\begin{aligned}\delta_{p,i} &= ||p_i||_2 - ||p_i - p||_2 \\ \delta_{\dot{p},i} &= -\dot{p}^\top \hat{n}_i \\ \hat{n}_i &= \frac{p - p_i}{||p - p_i||_2}\end{aligned}$$

Note that we have arbitrarily defined the outputs  $y(t)$  of the system to be the  $x, y$  position of the mass. If we were measuring the position and velocity, the choice of output would be different.

## 5.1 Linearization

The easiest way to control a nonlinear system is by making it linear. The Jacobian matrix (2.2.4) can be calculated about the point.

Beware that the accuracy of the linearized model decreases the farther away (in terms of state) you are calculating. If compute power allows,

### 5.1.1 *Taylor Series Expansion*

One method to linearize is to use the Taylor series expansion and retain only the zero-th and first order term.

### 5.1.2 *Jacobian*

Refer to section (2.2.4).

## 5.2 Sliding Mode Control



## 6 State Estimation

### 6.1 Purpose

### 6.2 Observer

### 6.3 Measurement Model

### 6.4 Filtering

Filtering is the real-time subset of estimators.

#### 6.4.1 Calibration

### 6.5 Complementary Filter

### 6.6 Alpha-Beta Filter

#### 6.6.1 Alpha-Beta-Gamma Filter

### 6.7 Least-Squares Estimator

### 6.8 Linear-Quadratic Estimator (Kalman Filter)

#### 6.8.1 Extended Kalman Filter (EKF)

The Kalman filter is limited to linear systems. One method to apply it to nonlinear systems is to linearize the system at the current time step, and perform the same algorithm.

#### 6.8.2 Unscented Kalman Filter (UKF)

### 6.9 Particle Filter

#### 6.9.1 SIS Filter

#### 6.9.2 SIR Filter

#### 6.9.3 The Curse of Dimensionality

### 6.10 Ensemble Kalman Filter (EnKF)

### 6.11 Sensor Fusion

#### 6.11.1 Madgwick Filter

Note the quaternion derivative appears (Eq. 7.1).

#### 6.11.2 Mahoni Filter

## 7 Modeling & Simulation

### 7.1 Reference Frames

#### 7.1.1 Inertial Frame

#### 7.1.2 Body Frame

#### 7.1.3 North-East-Down Frame

#### 7.1.4 Intermediate Frames

### Transforming Between Frames

#### Wind Frame

#### Stability Frame

#### 7.1.5 Coriolis Forces

### 7.2 Determining the State Vector

#### 7.2.1 Position & Derivatives

#### 7.2.2 Rotation & Derivatives

### Euler Angles

### Direction Cosine Matrices

### Quaternions

### Quaternion Derivatives

$$\dot{q} = \frac{1}{2}\Omega q, \quad \text{where } \Omega = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \quad (7.1)$$

$$\text{Recall that } \omega_{b/v}^b = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

**Converting Between Representations** There are different methods based on which representation is being converted from/to. For ease, MATLAB has the built-in functions `eul2dcm`, `dcm2quat`, `quat2eul`. There is also the `angle2quat` function.

#### 7.2.3 Homogenous Transformation Matrix

The idea behind the homogenous transformation matrix is to combine rotation and translation.











## 12 Other Important Topics

### 12.1 Frequency Domain vs Time Domain

Computing the convolution integral is much more difficult in the time domain compared to the frequency domain. Using the frequency domain to analyze a system means that the Laplace transform can be used, simplifying the convolution step significantly. See (2.2.8)

### 12.2 Continuous Time vs Discrete Time

When the controller is calculated by a computer, the system and feedback cannot be evaluated in continuous time. The system can be thought of as proceeding in time-steps.

#### 12.2.1 Z Transform

The Z transform is the corollary to the Laplace Transform, only for discrete time where Laplace is for continuous.

### 12.3 Optimization

#### 12.3.1 Gradient Descent

### 12.4 Singular Value Decomposition





## 14 Appendix: MATLAB / Simulink Reference

This chapter contains practical methods / tutorials to accomplish tasks in MATLAB and Simulink

### 14.1 Linearization

#### 14.1.1 *MATLAB*

#### 14.1.2 *Simulink*

### 14.2 Animation

#### 14.2.1 *MATLAB*

#### 14.2.2 *Simulink*



