Control Systems: Classical, Neural, and Fuzzy

Oregon Graduate Institute

Lecture Notes - 1998

Eric A. Wan
Preface

This material corresponds to the consolidated lecture summaries and handouts for Control Systems: Classical, Neural, and Fuzzy. The class was motivated from a desire to educate students interested in neural and fuzzy control. Often students and practitioners studying these subjects lack the fundamentals. This class attempts to provide both a foundation and appreciation for traditional control theory, while at the same time, putting newer trends in neural and fuzzy control into perspective. Yes, this really is only a one quarter class (though not everything was covered each term). Clearly, the material could be better covered over two quarters and is also not meant as a substitute for more formal courses in control theory.

The lecture summaries are just that. They are often terse on explanation and are not a substitute for attending lectures or reading the supplemental material. Many of the summaries were initially formatted in LaTeX by student\(^1\) “scribes” who would try to decipher my handwritten notes after a lecture. Subsequent years I would try to make minor corrections. Included figures are often copied directly from other sources (without permission). Thus, these notes are not for general distribution. This is the first draft of a working document - beware of typos!

## Contents

Preface \hspace{0.5cm} i

Class Information Sheet \hspace{0.5cm} ix

### I Introduction

0.1 Basic Structure \hspace{0.5cm} 1
0.2 Classical Control \hspace{0.5cm} 1
0.3 State-Space Control \hspace{0.5cm} 2
0.4 Advanced Topics \hspace{0.5cm} 3
   0.4.1 Dynamic programming \hspace{0.5cm} 3
   0.4.2 Adaptive Control \hspace{0.5cm} 4
   0.4.3 Robust Control \( H_{\infty} \) \hspace{0.5cm} 4
0.5 History of Feedback Control \hspace{0.5cm} 5

1 Neural Control \hspace{0.5cm} 6
   1.0.1 Reinforcement Learning \hspace{0.5cm} 7
1.1 History of Neural Networks and Neural Control \hspace{0.5cm} 9

2 Fuzzy Logic \hspace{0.5cm} 9
   2.1 History \hspace{0.5cm} 9
   2.2 Fuzzy Control \hspace{0.5cm} 10
   2.3 Summary Comments \hspace{0.5cm} 11

### II Basic Feedback Principles \hspace{0.5cm} 12

1 Dynamic Systems - “Equations of Motion” \hspace{0.5cm} 12

2 Linearization \hspace{0.5cm} 19
   2.1 Feedback Linearization \hspace{0.5cm} 19
   2.2 Small Signal Linearization \hspace{0.5cm} 19

3 Basic Concepts \hspace{0.5cm} 20
   3.1 Laplace Transforms \hspace{0.5cm} 20
      3.1.1 Basic Properties of Laplace Transforms \hspace{0.5cm} 20
   3.2 Poles and Zeros \hspace{0.5cm} 21
   3.3 Second Order Systems \hspace{0.5cm} 22
      3.3.1 Step Response \hspace{0.5cm} 23
   3.4 Additional Poles and Zeros \hspace{0.5cm} 25
   3.5 Basic Feedback \hspace{0.5cm} 26
   3.6 Sensitivity \hspace{0.5cm} 28
   3.7 Generic System Tradeoffs \hspace{0.5cm} 30
   3.8 Types of Control - PID \hspace{0.5cm} 31
   3.9 Steady State Error and Tracking \hspace{0.5cm} 33

4 Appendix - Laplace Transform Tables \hspace{0.5cm} 35
III Classical Control - Root Locus 38

1 The Root Locus Design Method 38
   1.1 Introduction ........................................... 38
   1.2 Definition of Root Locus .................................. 38
   1.3 Construction Steps for Sketching Root Loci .................. 39
   1.4 Illustrative Root Loci .................................... 44
   1.5 Some Root Loci Construction Aspects ......................... 45
   1.6 Summary .................................................. 46

2 Root Locus - Compensation 46
   2.1 Lead Compensation ........................................ 47
       2.1.1 Zero and Pole Selection ............................ 48
   2.2 Lag Compensation ......................................... 48
       2.2.1 Illustration ......................................... 48
   2.3 The "Stick on a Cart" example ................................ 49
   2.4 Extensions ............................................... 51

IV Frequency Design Methods 52

1 Frequency Response 52

2 Bode Plots 52
   2.1 Stability Margins ........................................ 56
   2.2 Compensation ............................................ 58
       2.2.1 Bode's Gain-Phase Relationship ..................... 58
       2.2.2 Closed-loop frequency response ..................... 59
   2.3 Proportional Compensation ................................ 60
       2.3.1 Proportional/Differential Compensation .............. 63
       2.3.2 Lead Compensation ................................... 64
       2.3.3 Proportional/Integral Compensation .................. 66
       2.3.4 Lag Compensation .................................... 67

3 Nyquist Diagrams 68
   3.0.5 Nyquist Examples ...................................... 70
   3.1 Stability Margins ........................................ 74

V Digital Classical Control 76

1 Discrete Control - Z-transform 76
   1.1 Continuous to Discrete Mapping ............................ 76
   1.2 ZOH ...................................................... 77
   1.3 Z-plane and dynamic response .............................. 80

2 Root Locus control design 82
   2.1 Comments - Latency ....................................... 84
3 Frequency Design Methods
   3.1 Compensator design ........................................... 84
   3.2 Direct Method (Ragazzini 1958) .............................. 85

4 Z-Transform Tables .................................................. 85

VI State-Space Control ................................................. 87

1 State-Space Representation ......................................... 87
   1.1 Definition ..................................................... 87
   1.2 Continuous-time .............................................. 88
   1.3 Linear Time Invariant Systems ................................. 88
   1.4 "Units" of F in physical terms ................................ 88
   1.5 Discrete-Time .................................................. 88
      1.5.1 Example 2 - "analog computers" ......................... 89
   1.6 State Space Vs. Classical Approach .......................... 89
   1.7 Linear Systems we won't study ................................ 90
   1.8 Linearization .................................................. 91
   1.9 State-transformation .......................................... 92
   1.10 Transfer Function ............................................. 92
       1.10.1 Continuous System ...................................... 92
       1.10.2 Discrete System ......................................... 93
   1.11 Example - what transfer function don't tell us. .......... 94
   1.12 Time-domain Solutions ........................................ 95
   1.13 Poles and Zeros from the State-Space Description .......... 98
   1.14 Discrete-Systems ............................................. 100
       1.14.1 Transfer Function ...................................... 100
       1.14.2 Relation between Continuous and Discrete .......... 100

2 Controllability and Observability ................................ 101
   2.1 Controller canonical form ...................................... 101
   2.2 Duality of the controller and observer canonical forms .... 102
   2.3 Transformation of state space forms .......................... 102
   2.4 Discrete controllability or reachability ..................... 106
       2.4.1 Controllability to the origin for discrete systems .. 107
   2.5 Observability .................................................. 107
   2.6 Things we won't prove ......................................... 108

3 Feedback control ..................................................... 109
   3.1 Zeros .......................................................... 110
   3.2 Reference Input ............................................... 112
   3.3 Selection of Pole Locations ................................... 114
       3.3.1 Method 1 : Dominant second-order poles ............... 114
       3.3.2 Step Response Using State Feedback .................... 115
       3.3.3 Gain/Phase Margin ....................................... 117
       3.3.4 Method 2 : Prototype Design ............................. 119
       3.3.5 Method 3 : Optimal Control and Symmetric Root Locus .. 119
3.4 Discrete Systems .................................................. 127

4 Estimator and Compensator Design .................................. 134
  4.1 Estimators ......................................................... 134
    4.1.1 Selection of Estimator poles \( \alpha_c \) .................... 135
  4.2 Compensators: Estimators plus Feedback Control .............. 136
    4.2.1 Equivalent feedback compensation ....................... 138
    4.2.2 Bode/root locus ........................................... 138
    4.2.3 Alternate reference input methods ....................... 146
  4.3 Discrete Estimators ............................................. 147
    4.3.1 Predictive Estimator ...................................... 147
    4.3.2 Current estimator .......................................... 147
  4.4 Miscellaneous Topics ........................................... 148
    4.4.1 Reduced order estimator ................................... 148
    4.4.2 Integral control ........................................... 148
    4.4.3 Internal model principle ................................... 149
    4.4.4 Polynomial methods ........................................ 149

5 Kalman ............................................................... 150

6 Appendix 1 - State-Space Control .................................. 151

7 Appendix - Canonical Forms ......................................... 153

VII Advanced Topics in Control ..................................... 157

1 Calculus of Variations - Optimal Control ......................... 157
  1.1 Basic Optimization ............................................ 157
    1.1.1 Optimization without constraints ....................... 157
    1.1.2 Optimization with equality constraints ................ 157
    1.1.3 Numerical Optimization ................................... 159
  1.2 Euler-Lagrange and Optimal control ................................ 159
    1.2.1 Optimization over time .................................... 159
    1.2.2 Miscellaneous items ....................................... 161
  1.3 Linear system / ARE ............................................ 161
    1.3.1 Another Example ........................................... 162
  1.4 Symmetric root locus ............................................ 162
    1.4.1 Discrete case .............................................. 163
    1.4.2 Predictive control ......................................... 164
    1.4.3 Other applied problems .................................... 165

2 Dynamic Programming (DP) - Optimal Control .................... 167
  2.1 Bellman ........................................................... 167
  2.2 Examples: ....................................................... 168
    2.2.1 Routing ..................................................... 168
    2.2.2 City Walk .................................................. 168
  2.3 General Formulation ............................................ 168
  2.4 LQ Cost .......................................................... 169
# Fuzzy Logic & Control

## 1 Fuzzy Systems - Overview

## 2 Sample Commercial Applications

## 3 Regular Set Theory

## 4 Fuzzy Logic

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Definitions</td>
<td>255</td>
</tr>
<tr>
<td>4.2 Properties</td>
<td>256</td>
</tr>
<tr>
<td>4.3 Fuzzy Relations</td>
<td>257</td>
</tr>
<tr>
<td>4.4 Boolean Functions</td>
<td>257</td>
</tr>
<tr>
<td>4.5 Other Definitions</td>
<td>258</td>
</tr>
<tr>
<td>4.6 Hedges</td>
<td>258</td>
</tr>
</tbody>
</table>

## 5 Fuzzy Systems

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 A simple example of process control</td>
<td>260</td>
</tr>
<tr>
<td>5.2 Some Strategies for Defuzzification</td>
<td>262</td>
</tr>
<tr>
<td>5.3 Variations on Inference</td>
<td>262</td>
</tr>
<tr>
<td>5.4 7 Rules for the Broom Balancer</td>
<td>264</td>
</tr>
<tr>
<td>5.5 Summary Comments on Basic Fuzzy Systems</td>
<td>264</td>
</tr>
</tbody>
</table>

## 6 Adaptive Fuzzy Systems

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 Fuzzy Clustering</td>
<td>265</td>
</tr>
<tr>
<td>6.1.1 Fuzzy logic viewed as a mapping</td>
<td>265</td>
</tr>
<tr>
<td>6.1.2 Adaptive Product Space Clustering</td>
<td>266</td>
</tr>
<tr>
<td>6.2 Additive Model with weights (Kosko)</td>
<td>266</td>
</tr>
<tr>
<td>6.3 Example Ad-Hoc Adaptive Fuzzy</td>
<td>267</td>
</tr>
<tr>
<td>6.4 Neuro-Fuzzy Systems</td>
<td>267</td>
</tr>
<tr>
<td>6.4.1 Example: Sugene Fuzzy Model</td>
<td>268</td>
</tr>
<tr>
<td>6.4.2 Another Simple Example</td>
<td>269</td>
</tr>
</tbody>
</table>

## 7 Neural & Fuzzy

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1 NN as design Tool</td>
<td>270</td>
</tr>
<tr>
<td>7.2 NN for pre-processing</td>
<td>270</td>
</tr>
<tr>
<td>7.3 NN corrective type</td>
<td>271</td>
</tr>
<tr>
<td>7.4 NN for post-processing</td>
<td>271</td>
</tr>
<tr>
<td>7.5 Fuzzy Control and NN system ID</td>
<td>271</td>
</tr>
</tbody>
</table>

## 8 Fuzzy Control Examples

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1 Intelligent Cruise Control with Fuzzy Logic</td>
<td>272</td>
</tr>
<tr>
<td>8.2 Fuzzy Logic Anti-Lock Brake System for a Limited Range Coefficient of Friction Surface</td>
<td>273</td>
</tr>
</tbody>
</table>

## 9 Appendix - Fuzziness vs Randomness

275
<table>
<thead>
<tr>
<th>Exercises and Simulator</th>
<th>277</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW 1 - Continuous Classical Control</td>
<td>277</td>
</tr>
<tr>
<td>HW 2 - Digital Classical Control</td>
<td>277</td>
</tr>
<tr>
<td>HW 3 - State-Space</td>
<td>277</td>
</tr>
<tr>
<td>HW 4 - Double Pendulum Simulator - Linear SS Control</td>
<td>277</td>
</tr>
<tr>
<td>Midterm</td>
<td>277</td>
</tr>
<tr>
<td>Double Pendulum Simulator - Neural Control</td>
<td>277</td>
</tr>
<tr>
<td>Double Pendulum Simulator Documents</td>
<td>277</td>
</tr>
<tr>
<td>0.1 System description for double pendulum on a cart</td>
<td>278</td>
</tr>
<tr>
<td>0.2 MATLAB files</td>
<td>280</td>
</tr>
</tbody>
</table>
Applications of modern control systems range from advanced aircraft, to processes control in integrated circuit manufacturing, to fuzzy washing machines. The aim of this class is to integrate different trends in control theory.

Background and perspective is provided in the first half of the class through review of basic classical techniques in feedback control (root locus, bode, etc.) as well as state-space approaches (linear quadratic regulators, Kalman estimators, and introduction to optimal control). We then turn to more recent movements at the forefront control technology. Artificial neural network control is presented with emphasis on nonlinear dynamics, backpropagation-through-time, model reference control, and reinforcement learning. Finally, we cover fuzzy logic and fuzzy systems as a simple heuristic based, yet often effective, alternative for many control problems.

Instructor: Eric A. Wan
Room 581, Phone (503) 690-1164, Internet: ericwan@ece.ogi.edu.

Web Page: http://www.ece.ogi.edu/~ericwan/ECE553/

Grading: Homework-45%, Midterm-25%, Project-30%

Prerequisites: Digital signal processing, statistics, MATLAB programming.

Required Text:

Recommended Texts:

(General Control)

(Optimal Control)


(Adaptive Control)


(Robust / $H_\infty$ Control)


(Neural and Fuzzy Control)


Tentative Schedule

• Week 1
  – Introduction, Basic Feedback Principles

• Week 2
  – Bode Plots, Root Locus, Discrete Systems

• Week 3
  – State Space Methods, Pole Placement, Controllability and Observability

• Week 4
  – Optimal Control and LQR, Kalman filters and LQG

• Week 5
  – Dynamic Programming, Adaptive Control or Intro to Robust Control

• Week 6
  – Neural Networks, Neural Control Basics

• Week 7
  – Backpropagation-through-time, Real-time-recurrent learning, Relations to optimal control

• Week 8
  – Reinforcement Learning, Case studies

• Week 9
  – Fuzzy Logic, Fuzzy Control

• Week 10
  – Neural-Fuzzy, Other

• Finals Week
  – Class Presentations
Part I
Introduction

What is "control"? Make x do y.

Manual control: human-machine interface, *e.g.*, driving a car.

Automatic control (our interest): machine-machine interface (thermostat, moon landing, satellites, aircraft control, robotics control, disk drives, process control, bioreactors).

0.1 Basic Structure

Feedback control:

This is the basic structure that we will be using. In the digital domain, we must add D/A and A/D converters:

where ZOH (zero-order hold) is associated with the D/A converter, *i.e.*, the value of the input held constant until the next value is available.

0.2 Classical Control

What is "classical" control? From one perspective, it is any type of control that is non-neural or non-fuzzy. More conventionally, it is control based on the use of transfer functions, \( G(s) \) (continuous) or \( G(z) \) (discrete) to represent linear differential equations.
If \( G(z) = b(z)/a(z) \), then the roots of \( a(z) \) (referred to as \( \text{poles} \)) and the roots of \( b(z) \) (referred to as \( \text{zeros} \)), determine the open-loop dynamics. The closed-loop dynamic response of the system is:

\[
\frac{y}{r} = \frac{DG}{1 + DG}
\]

(1)

What is the controller, \( D(z) \)? Examples:

- \( k \) - (proportional control)
- \( k_s \) - (differential control)
- \( k/s \) - (integral control)
- \( \text{PID} \) (combination of above)
- lead/lag

Classical Techniques:

- Root locus
- Nyquist
- Bode plots

0.3 State-Space Control

State-space, or "modern", control returns to the use of differential equations:

\[
\begin{align*}
\dot{x} &= Ax + bu \\
y &= Hx
\end{align*}
\]

(2)  
(3)

Any linear set of linear differential equations can be put in this standard form were \( x \) is a \textit{state-vector}.

Control law: given \( x \), control \( u = -kx + r \)
In large part, state-space control design involves finding the control law $k$. Common methods include: 1) selecting a desired set of closed loop pole location or 2) optimal control, which involves the solution of some cost function. For example, the solution of the LQR (linear quadratic regulator) problem is expressed as the solution of $k$ which minimizes

$$\int_{-\infty}^{\infty} (x^T Q x + u^T R u) dt$$

and addresses the optimal trade-off between tracking performance and control effort.

We can also address issues of:

- controllability
- observability
- stability
- MIMO

How to find the state $x$? Build an estimator:

![Diagram of control system](image)

with noise (stochastic), we will look at Kalman estimators which yields the LQG (Linear Quadratic Gaussian)

### 0.4 Advanced Topics

#### 0.4.1 Dynamic programming

Used to solve for general control law $u$ which minimizes some cost function. This may be used, for example, to solve the LQR problem, or some nonlinear $u = f(x)$ where the system to be controlled is nonlinear. Also used for learning trajectory control (terminal control).

Type classes of control problems:

a) *Regulators* (fixed point control) or tracking,
b) *Terminal control* is concerned with getting from point a to point b. Examples: a robotics manipulator, path-planning, etc.

Terminal control is often used in conjunction with regulator control, where we first plan an optimal path and then use other techniques to track that path.

Variations:

a) *Model predictive control*: This is similar to LQR (infinite horizon) but you use a finite horizon and resolve using dynamic programming at each time step: \( \int_0^N (u^2 + Qy^2)dt \). Also called receding horizon problem. Popular in process control.

b) *Time-optimal control*: Getting from point a to point b in minimum time. *e.g.* solving the optimal trajectory for an airplane to get to a desired location. It is not a steady ascent. The optimal is to climb to an intermediary elevation, level off, and then swoop to the desired height.

c) *Bang-bang control*: With control constraints, time-optimal control may lead to bang-bang control (hard on / hard off).

### 0.4.2 Adaptive Control

Adaptive control may be used when the system to be controlled is changing with time. May also be used with nonlinear system when we still want to use linear controllers (for different regimes).

In it’s basic form, adaptive control is simply gain scheduling: switch in pre-determined control parameters.

Methods include: self-tuning regulators and model reference control

![Adaptive Control Diagram](image)

### 0.4.3 Robust Control / \( H_\infty \)

Robust control deals with the ability of a system to work under uncertainty and encompasses advanced mathematical methods for dealing with the same. More formally, it minimizes the maximum singular value of the discrepancies between the closed-loop transfer function matrix and the desired
loop shape subject to a closed-loop stability constraint. It is a return to transfer function methods for MIMO systems while still utilizing state-space techniques.

0.5 History of Feedback Control

- Antiquity - Water clocks, level control for wine making etc. (which have now become modern flush toilets)

- 1624, Drebble - Incubator (The sensor consisted of a riser filled with alcohol and mercury. As the fire heats up the box, the alcohol expands and the riser floats up lowering the damper on the flue.)

![Sketch of Drebble's incubator for hatching chicken eggs. (Adapted from Mayr, 1970.)](image)

- 1728, Watt - Flyball governor

![Diagram of Watt's flyball governor](image)

- 1868, Maxwell - Flyball stability analysis (differential equations → linearization → roots of “characteristic equation” need be negative). 2nd and 3rd order systems.

- 1877, Routh - General test for stability of higher order polynomials.

- 1890, Lyapunov - Stability of non-linear differential equations (introduced to state-space control in 1958)

- 1910, Sperry - Gyroscope and autopilot control
• 1927, Black - Feedback amplifier, Bush - Differential analyzer (necessary for long distance telephone communications)

• 1932, Nyquist stability Criterion

• 1938, Bode - Frequency response methods.

• 1936 - PID control methods

• 1942, Wiener - Optimal filter design (control plus stochastic processes)

• 1947 - Sampled data systems

• 1948, Evans - Root locus (developed for guidance control of aircraft)

• 1957, Bellman - Dynamic Programming

• 1960, Kalman - Optimal Estimation

• 1960, State-space or "modern" control (this was motivated from work on satellite control, and was a return to ODE's)

• 1960's - MIMO state-space control, adaptive control

• 1980's - Zames, Doyle - Robust Control

1 Neural Control

Neural control (as well as fuzzy control) was developed in part as a reaction by practitioners to the perceived excessive mathematical intensity and formalism of "classical" control. Although neural control techniques have been inappropriately used in the past, the field of neural control is now starting to mature.

Early systems: open loop: $C \approx P^{-1}$
Feedback control with neural networks:

- Train by back-propagation-through-time (BPTT) or real-time-recurrent-learning (RTRL).
- May have similar objectives and cost functions (as in LQR, minimum-time, model-reference, etc.) as classical control, but solutions are iterative and approximate.
- ”Approximately optimal control” - optimal control techniques and strategies constrained to be approximate by nature of training and network architectural constraints.
- Often little regard for ”dynamics” - few theoretical results regarding stability, controllability, etc.

Some advantages:
- MIMO = SISO
- Can handle non-linear systems
- Generates good results in many problems

1.0.1 Reinforcement Learning

A related area of neural network control is reinforcement learning (“approximate dynamic programming”)

Dynamic programming allows us to do a stepwise cost-minimization in order to solve a more complicated trajectory optimization. Bellman Optimality Condition: An optimal policy has the property that whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.
Short-term, or step-wise, maximization of $J^*$ leads to long term maximization of $u(k)$. (Unfortunately, problems grows exponentially with number of variables).

To minimize some utility function along entire path (to find control $u(k)$), need only minimize individual segments:

$$J^*(x_{k-1}) = \min_{u_k} [r(x_k, u_k) + J^*(x_k)]$$

In general, dynamic programming problems can be extremely difficult to solve. In reinforcement learning, an adaptive critic is used to get an approximation of $J^*$.

(These methods are more complicated than other neural control strategies, are hard to train, have many unresolved issues, but offer great promise for future performance.)
1.1 History of Neural Networks and Neural Control

- 1943, McCulloch & Pitts - Model of artificial neuron
- 1949, Hebb - simple learning rule
- 1957, Rosenblatt - Perceptron
- 1957, Bellman - Dynamic Programming (origins of reinforcement learning, 1960 - Samuel’s learning checker game)
- 1959, Widrow & Hoff - LMS and Adalines (1960 - “Broom Balancer” control)
- 1983 Barto, Anderson, Sutton - Adaptive Heuristic Critics
- 1989 Nguyen, Werbos, Jordan, etc. - Backpropagation-Through-Time for control
- 1990’s Increased sophistication, applications, some theory, relation to “classical” control recognized.

2 Fuzzy Logic

2.1 History

- 1965, L. Zadeh - Fuzzy Sets
- 1974, Mamdani & Assilan - Steam engine control using fuzzy logic, other examples.
- 1980’s Explosion of applications from Japan (fuzzy washing machines).
- 1990’s Adaptive Fuzzy Logic, Neural-Fuzzy, etc.
2.2 Fuzzy Control

1. No regard for dynamic, stability, mathematical modeling

2. Simple to design

3. Combines heuristic "linguistic" rule-based knowledge with smooth control

4. Elegant "gain scheduling" and interpolation

Fuzzy logic makes use of "membership functions" which introduce an element of ambiguity, e.g., the "degree" to which we may consider zero to be zero.

Because of its use of intuitive rules, fuzzy logic can be used appropriately in a wide range of control problems where mathematical precision is not important, but it is also often misused. Newer methods try to incorporate ideas from neural networks to adapt the fuzzy systems (neural-fuzzy).
2.3 Summary Comments

Classical (and state-space) control is a mature field that utilizes rigorous mathematics and exact modeling to exercise precise control over the dynamic response of a system.

Neural control overlaps much with classical control. It is less precise in its formulation yet may yield better performance for certain applications.

Fuzzy control is less rigorous, but is a simple approach which generates adequate results for many problems.

There is a place and appropriate use for all three methods.
Part II
Basic Feedback Principles

This handout briefly describes some fundamental concepts in Feedback Control.

1 Dynamic Systems - “Equations of Motion”

Where do system equations come from?

- Mechanical Systems

\[ F = m \cdot a \]
\[ \text{Force} = \text{Mass} \cdot \text{acceleration} \]

\[ \ddot{x} + \frac{b}{m_1} (\dot{x} - \dot{y}) + \frac{k}{m_1} (x - y) = \frac{u}{m_1} \]
\[ \ddot{y} + \frac{b}{m_2} (\dot{y} - \dot{x}) + \frac{k}{m_2} (y - x) = 0 \]
Rotational Systems

\[ T = I \alpha \]

(Torque) = (moment of inertia) \times (angular acceleration)

- Satellite

![Satellite diagram]

\[
J_1 \ddot{\theta}_1 + d (\dot{\theta}_1 - \dot{\theta}_3) + k (\theta_1 - \theta_2) = T_c
\]
\[
J_2 \ddot{\theta}_2 + d (\dot{\theta}_2 - \dot{\theta}_1) + k (\theta_2 - \theta_1) = 0
\]

- Pendulum

![Pendulum diagram]

\[ T = m e^2 \ddot{\theta} + m g l \sin \theta \]
- “Stick on cart” / Inverted Pendulum (linearized equations)

\[
(I + m_p l^2) \ddot{\theta} - m_p g l \theta = m_p l \dddot{x} \\
(m_c + m_p) \dddot{x} + b \dddot{x} - m_p l \ddot{\theta} = u
\]
\[
\begin{bmatrix}
\dot{u} \\
\dot{w} \\
\dot{q} \\
\dot{\theta}
\end{bmatrix}
= \begin{bmatrix}
X_u & X_w & -W_0 & -g \cos \theta_0 \\
Z_u & Z_w & U_0 & -g \sin \theta_0 \\
M_u & M_w & M_q & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
u \\
w \\
q \\
\theta
\end{bmatrix}
+ \begin{bmatrix}
X_{se} \\
Z_{se} \\
M_{se} \\
0
\end{bmatrix} \delta e.
\]

where

\(u\) = forward velocity perturbation in the aircraft in x direction (see Fig. 6.6(b)).

\(w\) = velocity perturbation in the aircraft in z direction (also proportional to the angle of attack, \(\alpha = w/U_0\)).

\(q\) = angular rate about positive y axis, or pitch rate.

\(\theta\) = pitch-angle perturbation with respect to horizontal.

\(X\) = normalized aerodynamic force derivatives in x direction.

\(Z\) = normalized aerodynamic force derivatives in z direction.

\(M\) = normalized aerodynamic moment derivatives.

\(W_0\) = reference flight-condition velocity in z axis (usually 0 for an aircraft flying horizontally).*

\(U_0\) = reference flight-condition velocity in x axis (that is, the forward velocity of the aircraft).*

\(g\) = acceleration of gravity (32.2 ft/s² or 10 m/s²).

\(\theta_0\) = angle of the x axis from horizontal in the reference flight condition.

\(\delta e\) = movable tail-section, or "elevator," angle for pitch control.

For the Boeing 747 in horizontal flight, \(U = 830\) ft/s at 20,000 ft (Mach 0.8) with a weight of 637,000 lb, we have

\[
\begin{bmatrix}
\dot{u} \\
\dot{w} \\
\dot{q} \\
\dot{\theta}
\end{bmatrix}
= \begin{bmatrix}
-0.00643 & 0.0263 & 0 & -32.2 \\
-0.0941 & -0.624 & 820 & 0 \\
-0.00222 & -0.00153 & -0.668 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
u \\
w \\
q \\
\theta
\end{bmatrix}
+ \begin{bmatrix}
0 \\
-32.7 \\
-2.08 \\
0
\end{bmatrix} \delta e.
\]
- Electrical Circuits

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resistor</td>
<td>$v = Ri$</td>
</tr>
<tr>
<td>Capacitor</td>
<td>$i = C \frac{dv}{dt}$</td>
</tr>
<tr>
<td>Inductor</td>
<td>$v = L \frac{di}{dt}$</td>
</tr>
<tr>
<td>Voltage source</td>
<td>$v = v_s$</td>
</tr>
<tr>
<td>Current source</td>
<td>$i = i_s$</td>
</tr>
</tbody>
</table>

**Operational Amplifier**

$u_o = A_o (v_+ - v_-)$
- Electro-mechanical

\[ L_a \frac{di}{dt} + R_a i_a = V_a - 1c e \dot{\theta} \]

\[ J_m \ddot{\theta} + b (\theta_m - \dot{\theta}_e) + k (\theta_m - \Theta_s) = k_e i_e \]

\[ J_e \ddot{\theta} + b (\theta_e - \theta_m) + k (\theta_e - \Theta_e) = 0 \]

- Heat Flow

\[ q = \text{heat flow} \]

\[ \dot{T}_I = \frac{1}{C_S} \frac{1}{2} = \frac{1}{C_T} (\frac{1}{R_1} + \frac{1}{R_2}) (T_0 - T_I) \]

\[ C_T = \text{Thermal capacitance} \]

\[ T_0 = \text{Temperature outside} \]

\[ T_I = \text{Temperature inside} \]

\[ R_1, R_2 = \text{Thermal resistances} \]
- Incompressible Fluid Flow

\[
\begin{align*}
\dot{m} \ddot{x} &= A_p - F_0 \\
\omega &= \frac{1}{k} \left( p_1 - p_2 \right) \frac{\dot{x}}{x}
\end{align*}
\]

flow rate pressures

- Bio-reactor

\[
\dot{h} = \frac{A}{x} \left( \text{Win} - \text{Wout} \right)
\]

(Bio-reactor)

\[
\begin{align*}
\frac{dC_2}{dt} &= -C_2 \omega + C_1 (1-C_2) e^{C_2/\gamma} \frac{1+\beta}{1+\beta-C_2} \\
\frac{dC_1}{dt} &= -C_1 \omega + C_1 (1-C_2) e^{C_2/\gamma}
\end{align*}
\]

- $C_1$ - cell mass
- $C_2$ - substrate conversion
- $\gamma$, $\beta$ - determine rate of cell growth & nutrient consumption
2 Linearization

Consider the pendulum system shown.

\[\tau = ml^2 \ddot{\theta} + mgl \sin \theta \quad (4)\]

Two methods are typically used in the linear approximation of non-linear systems.

2.1 Feedback Linearization

\[\tau = mgl \sin \theta + u \quad (5)\]

then,

\[ml^2 \ddot{\theta} = u \leftarrow \text{Linear always!} \quad (6)\]

This method of linearization is used in Robotics for manipulator control.

2.2 Small Signal Linearization

\[\ddot{\theta} = \tau - \sin \theta \quad \text{for } m = g = l = 1. \quad (7)\]

\(\ddot{\theta}\) is a function of \(\tau\) and \(\theta \); \(f(\tau, \theta)\). Using a Taylor’s Expansion,

\[f(\tau, \theta) = f(0, 0) + \left[ \frac{\partial f}{\partial \tau} \right]_{0,0} \tau + \ldots \text{higher order terms} \quad (8)\]

\(f(0,0) = 0\) at equilibrium point.

So,

\[\ddot{\theta} = 0 + (\cos \theta) \left|_{0,0} \right. \theta + \tau = -\theta + \tau\]

and for an inverted pendulum

\[\ddot{\theta} = 0 + (\cos \theta) \left|_{0,\pi} \right. \theta + \tau = \theta + \tau\]

These linearized system models should then be tested on the original system to check the acceptability of the approximation.

Linearization does not always work, as can be seen from systems below.

Consider the following function \(\dot{y}_1 = y_1^3\) versus \(\dot{y}_2 = -y_2^3\),
Linearizing both systems yields \( \dot{y}_1 = 0 \). However, this is not correct since it would imply that both systems have similar responses.

3 Basic Concepts

3.1 Laplace Transforms

\[
F(s) = \int_0^\infty f(t) e^{-st} dt
\]  

(9)

The inverse is usually computed by factorization and transformation to the time domain.

<table>
<thead>
<tr>
<th>( f(t) )</th>
<th>( L[f(t)] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \delta(t) )</td>
</tr>
<tr>
<td>( \frac{1}{s} )</td>
<td>( 1(t) )</td>
</tr>
<tr>
<td>( \frac{1}{s^2} )</td>
<td>( t )</td>
</tr>
<tr>
<td>( \frac{1}{s+a} )</td>
<td>( e^{-at} )</td>
</tr>
<tr>
<td>( \frac{1}{s+a}^2 )</td>
<td>( t e^{-at} )</td>
</tr>
<tr>
<td>( \frac{a}{s^2+a^2} )</td>
<td>( \sin at )</td>
</tr>
<tr>
<td>( \frac{a}{s^2+a^2} )</td>
<td>( \cos at )</td>
</tr>
<tr>
<td>( \frac{1}{s^2+a^2} )</td>
<td>( e^{-at} \cos at )</td>
</tr>
</tbody>
</table>

Table 1: Some Standard Laplace Transforms

3.1.1 Basic Properties of Laplace Transforms

Convolution

\[
y(t) = h(t) * u(t) = \int_{-\infty}^{\infty} h(\tau) u(t - \tau) d\tau
\]

(10)

\[
Y(s) = H(s) U(s)
\]

(11)
Derivatives

\[ \dot{y} \leftrightarrow sY(s) - y(0) \]  
(12)

\[ y^{(n)} + a_{n-1} y^{(n-1)} + \ldots + a_1 \dot{y} + a_0 y = b_m u^m + b_{m-1} u^{m-1} + \ldots + b_0 u \]  
(13)

\[ Y(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_0}{s^n + a_{n-1} s^{n-1} + \ldots + a_0} U(s) \]  
(14)

for \( y(0) = 0 \).

\[ \frac{Y(s)}{U(s)} = H(s) = \frac{b(s)}{a(s)} \]  
(15)

H(s) - Transfer Function

Example:

A two mass system represented by two second order coupled equations:

\[
\ddot{x} + \frac{b}{m_1}(\ddot{x} - \dot{y}) + \frac{k}{m_1}(x - y) = \frac{U}{m_1}
\]

\[
\ddot{y} + \frac{b}{m_2}(\ddot{y} - \dot{x}) + \frac{k}{m_2}(y - x) = 0
\]

\[
s^2 X(s) + \frac{b}{m_1} (s X(s) - s Y(s)) + \frac{k}{m_1} (X(s) - Y(s)) = \frac{U(s)}{m_1}
\]

\[
s^2 Y(s) + \frac{b}{m_2} (s Y(s) - X(s)) + \frac{k}{m_2} (Y(s) - X(s)) = 0
\]

After some algebra we find that, \[ \frac{Y(s)}{U(s)} = \frac{\frac{b + k}{m_1 s^2 + ts + k}}{\frac{b + k}{m_2 s^2 + ts + k}} = \frac{\frac{\sqrt{b + k}}{m_1}}{\frac{\sqrt{b + k}}{m_2}} \]

3.2 Poles and Zeros

Consider the system,

\[ H(s) = \frac{2s + 1}{s^2 + 3s + 2} \]

\[ b(s) = 2 \left( s + \frac{1}{2} \right) \rightarrow \text{zero} : \ -\frac{1}{2} \]

\[ a(s) = (s + 1) (s + 2) \rightarrow \text{poles} : \ -1, -2 \]
\[ H(s) = \frac{-1}{s+1} + \frac{3}{s+2} ; h(t) = -e^{-t} + e^{-2t} \]

The response can be empirically estimated by an inspection of the pole-zero locations. For stability, poles have to be in the Left half of the s-Plane (LHP). Control is achieved by manipulating the pole-zero locations.

3.3 Second Order Systems

\[ H(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \]

\( \xi \) = damping ratio  
\( \omega_n \) = natural frequency

\( s = -\sigma - j\omega_d; \quad \sigma = \xi \omega_n; \quad \omega_d = \omega_n \sqrt{1 - \xi^2} \quad (16) \)

Damping = 0 \( \rightarrow \) oscillation  
1 \( \rightarrow \) smooth damping to final level  
\( \leq 1 \) \( \rightarrow \) some oscillation; can realize faster response
3.3.1 Step Response

Step Response = \frac{H(s)}{s} \quad (17)

y(t) = 1 - e^{-\sigma t}(\cos \omega_d t + \frac{\sigma}{\omega_d} \sin \omega_d t) \quad (18)

Rise time : t_r \approx \frac{1.8}{\omega_n} \quad (19)

Overshoot : M_p \approx 1 - \frac{\xi}{0.6} \quad 0 \leq \xi \leq 0.6 \quad (20)

t_s = \frac{4.6}{\sigma} \quad (21)

Typical values are :

\begin{align*}
M = 0.16 & \text{ for } \xi = 0.5 \\
M = 0.4 & \text{ for } \xi = 0.707
\end{align*}
(a) Step responses of second-order systems; (b) impulse responses of second-order systems.
Design Specifications

For specified $t_r$, $M_p$ and $t_s$

$$\omega_n \geq \frac{1.8}{t_r} \quad (22)$$

$$\xi \geq 0.6 \left(1 - M_p\right) \quad (23)$$

$$\sigma \geq \frac{4.6}{t_s} \quad (24)$$

3.4 Additional Poles and Zeros

$$H_1(s) = \frac{2}{(s + 1)(s + 2)} = \frac{2}{s + 1} - \frac{2}{s + 2}$$

$$H_2(s) = \frac{2(s + 1.1)}{1.1(s + 1)(s + 2)} = \frac{0.18}{s + 1} + \frac{1.64}{s + 2}$$

Here, a zero has been added near one of the poles. The 1.1 factor in the denominator adjusts the DC gain, as can be seen from the Final Value Theorem:

$$\lim_{t \to \infty} y(t) = \lim_{s \to 0} sY(s) \quad (25)$$

Note, the zero at -1.1 almost cancels the influence of the Pole at -1. As $s \to 0$, the DC gain is higher by a factor of 1.1, hence the factor in the denominator above.

- As Zero approaches origin $\to$ increased overshoot.
- Zero in RHP results in non-minimum phase system and a direction reversal in the time response.
- Poles dominated by those nearest the origin.
- Poles / Zeros can "cancel" in LHP relative to step response, but may still affect initial conditions.
3.5 Basic Feedback

\[ E(s) = R(s) - Y(s) \]  \hspace{1cm} (26)

\[ Y(s) = k H(s) E(s) \]  \hspace{1cm} (27)

\[ Y(s) = k H(s) \left( R(s) - Y(s) \right) \]  \hspace{1cm} (28)

\[ (1 + k H(s))Y(s) = k H(s) R(s) \]  \hspace{1cm} (29)

\[ \frac{Y(s)}{R(s)} = \frac{k H(s)}{1 + k H(s)} ; \quad \frac{E(s)}{R(s)} = \frac{1}{1 + k H(s)} \]  \hspace{1cm} (30)
With $k$ large, \[ \frac{kH(s)}{1 + kH(s)} \rightarrow \frac{kH(s)}{kH(s)} \rightarrow 1 \] (31)

We need to consider the system dynamics. Large $k$ may make the system unstable. Example:

\[
\frac{1}{s(s + 2)}
\]

The closed loop characteristic equation:

\[
s^2 + 2s + k = 0
\]

Example 2:

\[
H(s) = \frac{1}{s \left( (s + 4)^2 + 16 \right)}
\]

The closed loop characteristic equation:

\[
s \left[ (s + 4)^2 + 16 \right] + k = 0
\]
3.6 Sensitivity

\[ S_Y^X \triangleq \frac{\partial Y}{\partial X} = \frac{X}{Y} \frac{\partial Y}{\partial X} \]  (32)

\( S_Y^X \) is the “sensitivity of \( Y \) with respect to \( X \)”. Denotes relative changes in \( Y \) to relative changes in \( X \)

\[ H(s) = \frac{kG(s)}{1 + kG(s)} \]  (33)

\[ S_H^K = \frac{K}{H} \frac{\partial H}{\partial K} = \frac{1}{1 + KG} \to 0 : |KG| \text{ large} \]  (34)

Thus we see that feedback reduces the parameter sensitivity. Example:

By Final Value Theorem, the final state is = 1. But, suppose we are off with the plant gain \( k_o \to k_o + \delta k_o \), then,

\[ \text{output} = \frac{1}{k_o} (k_o + \delta k_o) = \delta + 1 \]  (35)

Thus, a 10% error in \( k_o \Rightarrow 10\% \text{ error in Speed control} \).
\[
\frac{k \frac{k_0}{s + 1}}{1 + \frac{k k_0}{s + 1}} = \frac{k k_0}{s + 1 + k k_0}
\]

Final values \( = \frac{k k_0}{1 + k k_0} \)

Set \( k_2 = \frac{1 + k k_0}{k k_0} \)

Sensitivity for \( k_0 \to k_0 + \delta k_0 \)

Final value \( = k_2 \left[ \frac{k (k_0 + \delta k_0)}{1 + k (k_0 + \delta k_0)} \right] \)

for \( k \) large \( \to \) reduced sensitivity.

Example:

- Integral Control

Final value
\[
\lim_{s \to 0} s \frac{1}{s} \left[ \frac{kk_0}{s(s + 1) + kk_0} \right] = 1
\]  

0 Steady State Error - Always!

- **Proportional Control**

  Increasing \( k \), increases speed of response.

- **Integral Control**

  Slower response, but more oscillations.

### 3.7 Generic System Tradeoffs

For example:

\[ D(s) = +k \text{ ; Proportional Control} \]

\[
Y(s) = \frac{DG}{1 + DG} R(s) + \frac{G}{1 + DG} \Omega(s) + \frac{-DG}{1 + DG} \Gamma(s)
\]  

Error

\[
E(s) = \frac{1}{1 + DG} R(s) + \frac{-G}{1 + DG} \Omega(s) + \frac{-1}{1 + DG} \Gamma(s)
\]

- **Disturbance rejection**

  \[ |DG| \gg 1 \]
• Good tracking: E small

\[ |DG| \gg 1 \]

• Noise Immunity

\[ |DG| \leftarrow \text{small} \]

3.8 Types of Control - PID

• Proportional

\[ D(s) = k \, ; \, u(k) = ke(t). \]  \hspace{1cm} (47)

• Integral

\[ D(s) = \frac{k}{T_i s} \, ; \, u(t) = \frac{k}{T_i} \int_0^t e(t) dt \]  \hspace{1cm} (48)

⇒ Zero steady state error (may slow down dynamics)

• Derivative

\[ D(s) = k T_D s \]  \hspace{1cm} (49)

→ increases damping and improves stability. Example:

\[ G(s) = \frac{1}{s^2 + as} \]

\[ = \frac{1}{s (s + a)} \]
With derivative

\[ \text{Pole-Zero Cancellation} \]
\[ \text{Closed loop pole trapped here} \]

But what about initial conditions?

\[ (\ddot{y}) + a (\dot{y}) = u(t) \]

\[ s^2 Y(s) + s y(0) + a s Y(s) + a y(0) = U(s) \]

\[ Y(s) = \frac{s + a}{s(s + a)} y(0) + \frac{U(s)}{s(s + a)} \]

Relative to initial value,

\[ Y(s) = \frac{y(0)}{1 + \frac{k}{s+a}} \]

\[ \lim_{t \to \infty} y(t) = \lim_{s \to 0} s \left( \frac{y(0)}{1 + \frac{k}{s+a}} \right) = \frac{y(0)}{1 + \frac{k}{a}} \neq 0 \]

Thus, the effect due to initial conditions is not negligible.

- **PID - Proportional - Integral - Derivative Control**

\[ k_P + k_D s + \frac{k_I}{s} \]

Parameters are often tuned using "Ziegler - Nichols PID tuning"
Figure 1:

3.9 Steady State Error and Tracking

The reference input to a control system is often of the form:

\[ r(t) = \frac{t^k}{k!}1(t) \]  

(50)

\[ R(s) = \frac{1}{s^{k+1}} \]  

(51)

In most cases, the reference input will not be a constant but can be approximated as a linear function of time for a time span long enough for the system to reach steady state. The error at this point of time is called the steady-state error.

The type of input to the system depends on the value of \( k \), as follows:

- \( k=0 \) implies a step input (position)
- \( k=1 \) implies a ramp input (velocity)
- \( k=2 \) implies a parabolic input (acceleration)

The steady state error of a feedback control system is defined as:

\[ e_\infty \triangleq \lim_{s \to 0} sE(s) \]  

(52)

\[ e_\infty = \lim_{s \to 0} sE(s) \]  

(53)

where \( E(s) \) is the Laplace transform of the error signal and is defined as:

\[ E(s) = \frac{1}{1 + D(s)G(s)} \]  

(54)

\[ E(s) = \frac{1}{1 + D(s)G(s)} \frac{1}{s^{k+1}} \]  

(55)

\[ e_\infty = \lim_{s \to 0} \frac{1}{s^k} \left[ \frac{1}{1 + D(s)G(s)} \right] = 0, \infty, \text{or a constant} \]  

(56)

Thus, the steady state error depends on the reference input and the loop transfer function.

System Type

The system type is defined as the order \( k \) for which \( e_\infty \) is a constant. This also equals number of open loop poles at the origin. Example:
\[ D(s)G(s) = \frac{k(1 + 0.5s)}{s(1 + s)(1 + 2s)} \text{ is of type 1} \]  
\[ D(s)G(s) = \frac{k}{s^3} \text{ is of type 3} \]  

**Steady-state error of system with a step-function input \( k=0 \)**  
- **Type 0 system:**  
  \[ e_\infty = \frac{1}{1 + DG(0)} = \frac{1}{1 + K_p} \]  
  where \( K_p \) is called the *closed loop DC gain* or the *step-error constant* and is defined as:  
  \[ K_p = \lim_{s \to 0} D(s)G(s) \]  
- **Type 1 or higher system:**  
  \[ e_\infty = 0 \]  
  (i.e., \( DG(0) = \infty \), due to pole at origin)  

**Steady-state error of system with a ramp-function input \( k=1 \)**  
- **Type 0 system:**  
  \[ e_\infty = \infty \]  
- **Type 1 system:**  
  \[ e_\infty = \lim_{s \to 0} \frac{1}{s[1 + DG(s)]} = \lim_{s \to 0} \frac{1}{sDG(0)} = \frac{1}{K_v} \]  
  where, \( K_v \) is called the *velocity constant.*  
- **Type 2 or higher system:**  
  \[ e_\infty = 0 \]
## Appendix - Laplace Transform Tables

<table>
<thead>
<tr>
<th>Number</th>
<th>( F(s) )</th>
<th>( f(t), t \geq 0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>( \delta(t) )</td>
</tr>
<tr>
<td>2</td>
<td>( 1/s )</td>
<td>( 1(t) )</td>
</tr>
<tr>
<td>3</td>
<td>( 1/s^2 )</td>
<td>( t )</td>
</tr>
<tr>
<td>4</td>
<td>( 2!/s^3 )</td>
<td>( t )</td>
</tr>
<tr>
<td>5</td>
<td>( 3!/s^4 )</td>
<td>( t^3 )</td>
</tr>
<tr>
<td>6</td>
<td>( m!/s^{m-1} )</td>
<td>( t^m )</td>
</tr>
<tr>
<td>7</td>
<td>( 1/(s + a) )</td>
<td>( e^{-at} )</td>
</tr>
<tr>
<td>8</td>
<td>( 1/(s + a)^2 )</td>
<td>( te^{-at} )</td>
</tr>
<tr>
<td>9</td>
<td>( 1/(s + a)^3 )</td>
<td>( \frac{1}{2!}t^2e^{-at} )</td>
</tr>
<tr>
<td>10</td>
<td>( 1/(s + a)^m )</td>
<td>( \frac{1}{(m - 1)!}t^{m-1}e^{-at} )</td>
</tr>
<tr>
<td>11</td>
<td>( \frac{a}{s(s + a)} )</td>
<td>( 1 - e^{-at} )</td>
</tr>
<tr>
<td>12</td>
<td>( \frac{a}{s^2(s + a)} )</td>
<td>( \frac{1}{a}(at - 1 + e^{-at}) )</td>
</tr>
<tr>
<td>13</td>
<td>( \frac{b - a}{(s + a)(s + b)} )</td>
<td>( e^{-at} - e^{-bt} )</td>
</tr>
<tr>
<td>14</td>
<td>( \frac{s}{(s + a)^2} )</td>
<td>( (1 - at)e^{-at} )</td>
</tr>
<tr>
<td>15</td>
<td>( \frac{a^2}{s(s + a)^2} )</td>
<td>( 1 - e^{-at}(1 + at) )</td>
</tr>
<tr>
<td>16</td>
<td>( \frac{(b - a)s}{(s + a)(s + b)} )</td>
<td>( be^{-bt} - ae^{-at} )</td>
</tr>
<tr>
<td>17</td>
<td>( \frac{a(s^2 + a^2)}{s(s^2 + a^2)} )</td>
<td>( \sin at )</td>
</tr>
<tr>
<td>18</td>
<td>( \frac{s}{s(s^2 + a^2)} )</td>
<td>( \cos at )</td>
</tr>
<tr>
<td>19</td>
<td>( \frac{s + a}{(s + a)^2 + b^2} )</td>
<td>( e^{-at}\cos bt )</td>
</tr>
<tr>
<td>20</td>
<td>( \frac{b}{(s + a)^2 + b^2} )</td>
<td>( e^{-at}\sin bt )</td>
</tr>
<tr>
<td>21</td>
<td>( \frac{a^2 + b^2}{s[(s + a)^2 + b^2]} )</td>
<td>( 1 - e^{-at}(\cos bt + \frac{a}{b}\sin bt) )</td>
</tr>
<tr>
<td>( f(t) )</td>
<td>( F(s) )</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Two-sided</strong></td>
<td><strong>One-sided</strong></td>
</tr>
<tr>
<td>0. ( Cx(t) )</td>
<td>( CX(s) )</td>
<td>( CX(s) )</td>
</tr>
<tr>
<td>1. ( \dot{x}(t) = \frac{dx}{dt} )</td>
<td>( sX(s) )</td>
<td>( sX(s) - x(0^-) )</td>
</tr>
<tr>
<td>2. ( \frac{d^2x}{dt^2} )</td>
<td>( s^2X(s) )</td>
<td>( s^2X(s) - e^{-t}x(0^-) - e^{-2t}x(0^-) - \cdots - \frac{d^{n-1}x}{dt^{n-1}} \bigg</td>
</tr>
<tr>
<td>3. ( x(t) )</td>
<td>( \frac{1}{s} X(s) )</td>
<td>( \frac{1}{s} X(s) + \frac{1}{s} [x(t)]_{t=0^-} )</td>
</tr>
<tr>
<td>4. ( \delta(t) )</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5. ( u(t) )</td>
<td>( \frac{1}{s} )</td>
<td></td>
</tr>
<tr>
<td>5.1. ( u(t - \tau) )</td>
<td>( \frac{1}{s} e^{-\tau s} )</td>
<td></td>
</tr>
<tr>
<td>5.2. ( x(t - \tau) )</td>
<td>( e^{-\tau s}X(s) )</td>
<td></td>
</tr>
<tr>
<td>6. ( u(t)e^{-\sigma t} )</td>
<td>( \frac{1}{s + \sigma} )</td>
<td></td>
</tr>
<tr>
<td>7. ( u(t) \sin \omega t )</td>
<td>( \frac{\omega}{s^2 + \omega^2} )</td>
<td></td>
</tr>
<tr>
<td>8. ( u(t) \cos \omega t )</td>
<td>( \frac{s}{s^2 + \omega^2} )</td>
<td></td>
</tr>
<tr>
<td>9. ( u(t) \sin (\omega t + \psi) = u(t) \cos (\omega t + \psi - \frac{\pi}{2}) )</td>
<td>( \frac{\omega}{(s + \sigma_1)(s^2 + \omega_0^2)} )</td>
<td></td>
</tr>
<tr>
<td>( t )</td>
<td>( f(t) )</td>
<td>( F(s) )</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>
| 10. \( u(t)e^{-\omega t} \min (\omega t + \psi) = u(t)e^{-\omega t} \cos \left( \omega t + \phi - \frac{\psi}{2} \right) \) | \(
\frac{\omega}{\omega_0} \frac{s + \omega_0}{s + s_1 + j\omega_0} - \frac{\omega}{\omega_0} \frac{s + \omega_0}{s + s_1} \delta \left( s^2 + 2s_0 s + \omega_0^2 \right) \)
\( \tau = \omega_0/\omega_0 \) | \( -s_1 - s_1 \)
| & \( \frac{1}{\sqrt{s}} \) | \( -s_1 \) |
| 11. \( u(t)(1 - e^{-\omega t}) \) | \( \frac{s}{s(s + \omega_0)} \) | \( -s_1 \) |
| 12. \( u(t)(e^{-\omega t} - e^{-\omega t}) \) | \( \frac{-\omega_0 + \omega_0}{s + s_0}(s + \omega_0) \) | \( -s_1 \) |
| 13. \( u(t)e^{-\omega t} \) | \( \frac{1}{(s + \omega_0)^2} \) | \( -s_1 \) |
| 14. \( u(t) \left[ e^{-\omega t} + \frac{h_0}{\omega_0} \min (\omega t - \psi) \right] \) | \( \frac{h_0^2}{(s + s_0)(s^2 + \omega_0^2)} \) | \( -s_1 \) |
Part III
Classical Control - Root Locus

1 The Root Locus Design Method

1.1 Introduction

- The poles of the closed-loop transfer function are the roots of the characteristic equation, which determine the stability of the system.
- Studying the behavior of the roots of the characteristic equation will reveal stability and dynamics of the system, keeping in mind that the transient behavior of the system is also governed by the zeros of the closed-loop transfer function.
- An important study in linear control systems is the investigation of the trajectories of the roots of the characteristic equation—or, simply, the root loci—when a certain system parameter varies.
- The root locus is a plot of the closed-loop pole locations in the s-plane (or z-plane).
- It provides an insightful method for understanding how changes in the gain of the system feedback influence the closed loop pole locations.

\[ \text{Figure 2:} \]

1.2 Definition of Root Locus

The root locus is defined as a plot of the solution of:

\[ 1 + k D(s)G(s) = 0 \]  \hspace{1cm} (65)

We can think of \( D(s)G(s) \) as having the following form:

\[ D(s)G(s) = \frac{b(s)}{a(s)} \]  \hspace{1cm} (66)

Then with feedback,

\[ \frac{k D(s)G(s)}{1 + k D(s)G(s)} = \frac{kb(s)}{a(s) + kb(s)} \]  \hspace{1cm} (67)

- The zeros of the open loop system do not move.
• The poles move as a function of \( k \).

The root locus of \( D(s)G(s) \) may also be defined as the locus of points in the \( s \)-plane where the phase of \( D(s)G(s) \) is \( 180^\circ \). This is seen by noting that \( 1 + kD(s)G(s) = 0 \) if \( kD(s)G(s) = -1 \), which implies that the phase of \( D(s)G(s) = 180^\circ \).

At any point on the \( s \)-plane,

\[
\angle G = \Sigma \angle \text{due to zeros} - \Sigma \angle \text{due to poles}
\]  

(68)

Example:
For example, consider the open-loop transfer function:

\[
\frac{s + 1}{s[((s + 2)^2 + 4)(s + 5)]}
\]

The pole-zero plot is shown below:

- Assign a test point \( s_o = -1 + 2j \).
- Draw vectors directing from the poles and zeros to the point \( s_o \).
- If \( s_o \) is indeed a point on the root locus, then equation 68 must equal \(-180^\circ\).
- \( \angle G = \psi_1 - (\phi_1 + \phi_2 + \phi_3 + \phi_4) \)

1.3 Construction Steps for Sketching Root Loci

• Step #1: Mark the poles and zeros of the open-loop transfer function.
• Step #2: Plot the root locus due to poles on the real axis.
  - The root locus lies to the left of the odd number of real poles and real zeros.
  - A single pole or a single zero on the real axis introduces a phase shift of \( 180^\circ \) and since the left half of the \( s \)-plane is taken, the phase shift becomes negative.
Further, a second pole or a second zero on the real axis will introduce an additional phase shift of $180^\circ$, making the overall phase shift equal to $360^\circ$. Hence, the region is chosen accordingly.

Here are a few illustrations:

\[
\text{Step } \#3: \text{ Plot the asymptotes of the root loci.}
\]

- Asymptotes give the behavior of the root loci as $k \to \infty$.
- For $1 \cdot G(s) = 0$, as $k \to \infty$, $G(s)$ must approach 0. Typically, this should happen at the zeros of $G(s)$, but, it could also take place if there are more poles than zeros. We know that $b(s)$ has order $m$ and $a(s)$ has order $n$; i.e., let:

\[
\frac{b(s)}{a(s)} = \frac{s^m + b_1 s^{m-1} + \ldots}{s^n + a_1 s^{n-1} + \ldots}
\]

(69)

So, if $n > m$ then as $s \to \infty$, $G(s) \to 0$. That is, as $s$ becomes large, the poles and zeros approx. cancel each other. Thus

\[
1 + kG(s) \approx 1 + k \frac{1}{(s - \alpha)^{n-m}}
\]

(70)

\[
\alpha = \frac{\sum p_i - \sum z_i}{n - m}
\]

(71)

where, $p_i = \text{poles}$, $z_i = \text{zeros}$, and $\alpha$ is the centroid.

- There are $n - m$ asymptotes, where $n$ is the number of zeros and $m$ is the number of poles, and since $\angle \! G(s) = 180^\circ$, we have:

\[
(n - m) \phi_l = 180^\circ + (l)360^\circ
\]

(72)

For instance, $n - m = 3 \Rightarrow \phi_l = 60^\circ, 180^\circ, 300^\circ$.

- The $\phi$s give the angle where the asymptotes actually go to. Note that the angle of departure may be different and will be looked into shortly.
- The asymptotes of the root loci *intersect on the real axis at* $\alpha$ and this point of intersection is called the *centroid*. Hence, in other words, the asymptotes are centered at $\alpha$.

- The following figures a few typical cases involving asymptotes (which have the same appearance as multiple roots):

\[\text{Step \#4: Compute the angles of departure and the angles of arrival of the root loci. (optional - just use MATLAB)}\]

- The angle of departure or arrival of a root locus at a pole or zero denotes the angle of the tangent to the locus near the point.

- The root locus begins at poles and goes either to zeros or to $\infty$, along the radial asymptotic lines.

- To compute the angle of departure, take a test point $s_o$ very near the pole and compute the angle of $G(s_o)$, using equation 72. This gives the angle of departure $\phi_{dep}$ of the asymptote.

\[
\text{Also, } \phi_{dep} = \Sigma \psi_i - \Sigma \phi_i - 180^\circ - 360^\circ l
\]

where, $\Sigma \phi_i = \text{sum of the angles to the remaining poles}$ and $\Sigma \psi_i = \text{sum of the angles to all the zeros}$.

- For a multiple pole of order $q$:

\[
q \phi_{dep} = \Sigma \psi_i - \Sigma \phi_i - 180^\circ - 360^\circ l
\]  

In this case, there will be $q$ branches of the locus, that depart from that multiple pole.
The same process is used for the angle of arrival $\Psi_{arr}$:

$$q \psi_{arr} = \Sigma \phi_i - \Sigma \psi_i + 180^\circ + 360^\circ l$$

(75)

where, $\Sigma \phi_i$ = sum of angles to all poles, $\Sigma \psi_i$ = sum of all angles to remaining zeros, and $q$ is the order of the zero at the point of arrival.

- **Step #5: Determine the intersection of the root loci with the imaginary axis.** (optional)
  - The points where the root loci intersect the imaginary axis of the $s$-plane ($s = jw$), and the corresponding values of $k$, may be determined by means of the Routh-Hurwitz criterion.
  - A root of the characteristic equation in the RHP implies that the closed-loop system is unstable, as tested by the R-H criterion.
  - Using the R-H criterion, we can locate those values of $K$, for which an incremental change will cause the number of roots in the RHP to change. Such values correspond to a root locus crossing the imaginary axis.

- **Step #6: Determine the breakaway points or the saddle points on the root loci.** (optional)
  - Breakaway points or saddle points on the root loci of an equation correspond to *multiple-order roots of the equation.*

Figure 3:
- Figure 3(a) illustrates a case in which two branches of the root loci meet at the breakaway point on the real axis and then depart from the axis in opposite directions. In this case, the breakaway point represents a double root of the equation, when the value of $K$ is assigned the value corresponding to the point.
- Figure 3(b) shows another common situation when two complex-conjugate root loci approach the real axis, meet at the breakaway point and then depart in opposite directions along the real axis.

- In general, a breakaway point may involve more than two root loci.

- Figure 3(c) illustrates such a situation when the breakaway point represents a fourth-order root.

- A root locus diagram can have more than one saddle point. They need not always be on the real axis and due to conjugate symmetry of root loci, the saddle points not on the real axis must be in complex-conjugate pairs.

- All breakaway points must satisfy the following equations:

\[
\frac{dG(s)D(s)}{ds} = 0 \quad (76)
\]

\[
1 + KG(s)D(s) = 0 \quad (77)
\]

- The angles at which the root loci arrive or depart from a saddle point depends on the number of loci that are involved at the point. For example, the root loci shown in Figures 3(a) and 3(b) all arrive and break away at 180° apart, whereas in Figure 3(c), the four root loci arrive and depart with angles 90° apart.

- In general, n root loci arrive or leave a breakaway point at 180°/n degrees apart.

- The following figure shows some typical situations involving saddle points.
1.4 Illustrative Root Loci

- More examples:

\[ G(s) = \frac{(s + 3)(s + 1 \pm j)}{s(s + 1)(s + 2)^3(s + 4)(s + 5 \pm 2j)} \] (78)
1.5 Some Root Loci Construction Aspects

From the standpoint of designing a control system, it is often useful to learn the effects on the root loci when poles and zeros of \( D(s)G(s) \) are added or moved around in the s-plane. Mentioned below are a few brief properties pertaining to the above.

**Effects of Adding Poles and Zeros to \( D(s)G(s) \)**

- *Addition of Poles*: In general, adding a pole to the function \( D(s)G(s) \) in the left half of the s-plane has the effect of pushing the root loci toward the right-half plane.

- *Addition of Zeros*: Adding left-half plane zeros to the function \( D(s)G(s) \) generally has the effect of moving and bending the root loci toward the left-half s-plane.

**Calculation of gain \( k \) from the root locus**

Once the root loci have been constructed, the values of \( k \) at any point on the loci can be determined. We know that

\[
1 + kG(s) = 0
\]

\[
kG(s) = -1
\]

(79)
\[ k = -\frac{1}{G(s)} \quad (80) \]

\[ k = \frac{1}{|G(s)|} \quad (81) \]

Graphically, \( k = \frac{\text{distance from } s_o \text{ to zeros}}{\text{distance from } s_o \text{ to poles}} \quad (82) \)

1.6 Summary

- The Root Locus technique presents a graphical method of investigating the roots of the characteristic equation of a linear time-invariant system when one or more parameters vary.
- The steps of construction listed above should be adequate for making a reasonably adequate plot of the root-locus diagram.
- The characteristic-equation roots give exact indication on the absolute stability of the system. The zeros of the closed-loop transfer function also govern the dynamic performance of the system.

2 Root Locus - Compensation

If the process dynamics are of such a nature that a satisfactory design cannot be obtained by a gain adjustment alone, then some modification of compensation of the process dynamics is indicated. Two common methods for dynamic compensation are lead and lag compensation.

- Compensation with a transfer function of the form
  \[ D(s) = \frac{s + z_i}{s + p_i} \quad (83) \]
  is called lead compensation if \( z_i < p_i \) and lag compensation if \( z_i > p_i \).  
- Compensation is typically placed in series with the plant in the feedforward path as shown in the following figure:

![Figure 6:](image)

- It can also be placed in the feedback path and in that location, has the same effect on the overall system poles.
2.1 Lead Compensation

- Lead compensation approximates PD control.
- It acts mainly to lower the rise time.
- It decreases the transient overshoot, hence improving the system damping.
- It raises the bandwidth.
- It has the effect of moving the locus to the left.

Illustration: consider a second-order system with transfer function

\[ KG(s) = \frac{K}{s(s + 1)} \]  

\( G(s) \) has the root locus shown by the solid line in figure 7. Let \( D(s) = s + 2 \). The root locus produced by \( D(s)G(s) \) is shown by the dashed line. This adds a zero at \( s = -2 \). The modified locus is hence the circle.

\[ D(s) = s + 2 \]

\[ D(s) = \frac{s + 2}{s + 20} \]  

The following figure shows the resulting root loci when \( p = 10 \) and \( p = 20 \).
2.1.1 Zero and Pole Selection

- Selecting exact values of $z_i$ and $p_i$ is done by trial and error.
- In general, the zero is placed in the neighborhood of the closed-loop control frequency $\omega_n$.
- The pole is located at 3 to 20 times the value of the zero location.
- If the pole is too close to the zero, then the root locus moves back too far towards its uncompensated shape and the zero is not successful in doing its job.
- If the pole were too far to the left, then high-frequency noise amplification would result.

2.2 Lag Compensation

After obtaining satisfactory dynamic performance, perhaps by using one or more lead compensators, the low-frequency gain of the system may be found to be low. This indicates an integration at near-zero frequencies, and is achieved by lag compensation.

- Lag compensation approximates PI control.
- A pole is placed near $s = 0$ (low frequency). But, usually a zero is included near the pole, so that the pole-zero pair, called a dipole does not significantly interfere with the dynamic response of the overall system.
- Choose $D(s) = \frac{s + z}{s + p}$, $z > p$, where the values of $z$ and $p$ are small (e.g., $z = 0.1$ and $p = 0.01$).
- Since $z > p$, the phase is negative, corresponding to a phase lag.
- It improves the steady-state error by increasing the low-frequency gain.
- Lag compensation however decreases the stability.

2.2.1 Illustration

Again, consider the transfer function, as in equation 84.

- Include the lead compensation $D_1(s) = \frac{s + 2}{s + 20}$ that produced the locus as in figure.
- Raise the gain until the closed-loop roots correspond to a damping ratio of $\zeta = 0.707$. At this point, the root-locus gain is found to be 31.
• The velocity constant is thus \( K_v = \lim_{s \to 0} sKDG = (31/10) = 3.1 \)

• Now, add a lag compensation of:

\[
D_2(s) = \frac{s + 0.1}{s + 0.01}
\]  \hspace{1cm} (86)

This increases the velocity constant by about 10 (since \( z/p = 10 \)) and keeps the values of both \( z \) and \( p \) very small so that \( D_2(s) \) would have very little effect on the dynamics of the system. The resulting root locus is as shown in Figure 9.

![Root Locus Diagram](image)

**Figure 9:**

• The very small circle near the origin is a result of the lag compensation.

• A closed-loop root remains very near the lag compensation zero at \(-1\), which will correspond to a very slow decaying transient, which has a small magnitude because the zero will almost cancel the pole in the transfer function. However, the decay is so slow that this term may seriously influence the settling time.

• It is thus important to place the lag pole-zero combination at as high a frequency as possible without causing major shifts in the dominant root locations.

• The transfer function from a plant noise to the system error will not have the zero, and thus, disturbance transients can be very long in duration in a system with lag compensation.

2.3 The "Stick on a Cart" example

![Stick on a Cart Diagram](image)

After normalization, we have:

\[
\ddot{\theta} - \dot{\theta} = u
\]  \hspace{1cm} (87)
\[ \Rightarrow s^2 \theta(s) - \theta(s) = U(s) \]  
\[ \frac{\theta(s)}{U(s)} = \frac{1}{s^2 - 1} \]

This results in the following model of the system: The root locus diagram is shown below. The

![Root Locus Diagram](image)

Figure 10:

above figure indicates an unstable system, irrespective of the gain.

- With lead compensation.

![Lead Compensation Diagram](image)

Figure 11:
• The root loci of the lead-compensated system are now in the LHP, which indicates a stable system.

• A slight variation may result in a system as shown below, which may be more satisfactory. This system may tend to be slower than the one considered above, but may have better damping.

2.4 Extensions

Extensions to Root Locus include time-delays, zero-degree loci, nonlinear functions, etc...
Part IV  
Frequency Design Methods

1  Frequency Response

Most of this information is covered in the Chapter 5 of the Franklin text.

Frequency domain methods remain popular in spite of other design methods such as root locus, state space, and optimal control. They can also provide a good design in the face of plant uncertainty in the model.

We start with an open-loop transfer function

\[
\frac{Y(s)}{U(s)} = G(s) \rightarrow G(jw)
\]  

Assume

\[
u(t) = \sin(\omega t)
\]  

For a linear system,

\[
y(t) = A\sin(\omega t + \phi)
\]

The magnitude is given by

\[
A = |G(jw)| = |G(s)|_{s=jw}
\]

And the phase by

\[
\phi = \arctan \frac{\text{Im} G(jw)}{\text{Re} G(jw)} = \angle G(jw)
\]

2  Bode Plots

We refer to two plots when we talk about Bode plots

- Magnitude plot - $\log_{10}$ magnitude vs. $\log_{10} \omega$
- Phase plot - phase vs. $\log_{10} \omega$

Given a transfer function in $s$

\[
KG(s) = K \frac{(s + z_1)(s + z_2)\ldots}{(s + p_1)(s + p_2)\ldots}
\]

then there is a corresponding frequency response in $jw$

\[
KG(jw) = K \frac{(jw\tau_1 + 1)(jw\tau_2 + 1)\ldots}{(jw)^n(jw\tau_a + 1)\ldots}
\]

The magnitude plot then shows

\[
\log_{10} KG(jw) = \log_{10} K' + \log_{10} |jw\tau_1 + 1| + \ldots - n\log_{10} |jw| - \log_{10} |jw\tau_a + 1| - \ldots
\]

The phase plot shows

\[
\angle KG(jw) = \angle K + \angle(jw\tau_1 + 1) + \ldots - n90^\circ - \angle(jw\tau_a + 1) - \ldots
\]

There are three different terms to deal with in the previous equations:
\[ K(jw)^n \]
\[ (jw\tau + 1)^\pm 1 \]
\[ \left( \frac{jw}{\omega_n} \right)^2 + 2\xi \frac{jw}{\omega_n} + 1 \right)^2 \]

1. \( K(jw)^n \)

\[ \log_{10} K |(jw)^n| = \log_{10} K + n \log_{10} |jw| \]  

This term adds a line with slope \( n \) through \((1,1)\) on the magnitude plot, and adds a phase of \( n \times 90 \) to the phase plot. (see figure)

2. \((jw\tau + 1)^\pm 1\)

When \( w\tau \ll 1 \) then the term looks like 1. When \( w\tau \gg 1 \) the term looks like \( jw\tau \). This term adds a line with slope 0 for \( w < \frac{1}{\tau} \) and a line with slope \( \pm 1 \) for \( w < \frac{1}{\tau} \) to the magnitude plot, and adds \( \pm 90^\circ \) of phase when \( w > \frac{1}{\tau} \) to the phase plot. (see figures)

3. \( \left( \frac{jw}{\omega_n} \right)^2 + 2\xi \frac{jw}{\omega_n} + 1 \right)^2 \)

This term adds overshoot in the plots. (see figures)
(a) Magnitude and
(b) phase of
\[ \frac{1}{(j\omega/\omega_n)^2 + 2\zeta (j\omega/\omega_n) + 1} \]
See figures for sample Bode plots for the transfer function

$$G(s) = \frac{2000(s + 0.5)}{s(s + 10)(s + 50)}$$

(100)

Note that the command BODE in Matlab will create Bode plots given a transfer function.
2.1 Stability Margins

We can look at the open-loop frequency response to determine the closed-loop stability characteristics. The root locus plots $|KG(s)| = 1$ and $\angle G(s) = 180^\circ$. There are two measures that are used to denote the stability characteristics of a system. They are the gain margin (GM) and the phase margin (PM).

The gain margin is defined to be the distance between $|KG(j\omega)|$ and the magnitude $= 1$ line on the Bode magnitude plot at the frequency that satisfies $\angle KG(j\omega) = -180^\circ$. See figures for examples showing how to determine the GM from the Bode plots. The GM can also be determined from a root locus plot as

$$GM = \frac{|K| \text{ at } s = j\omega}{|K| \text{ at current design point}}$$

(101)

The phase margin is defined as the amount by which the phase of $G(j\omega)$ exceeds $-180^\circ$ at the frequency that satisfies $|KG(j\omega)| = 1$.

The damping ratio can be approximated from the PM as

$$\zeta \approx \frac{PM}{100}$$

(102)

See figures for examples showing how to determine the PM from the Bode plots. Also, there is a graph showing the relationship between the PM and the overshoot fraction, $M_p$. Note that if both the GM and the PM are positive, then the system is stable. The command MARGIN in Matlab will display and calculate the gain and phase margins given a transfer function.
Damping ratio versus phase margin (PM).

\[ \zeta = \text{PM}/100 \]

Transient response overshoot and frequency response resonant peak versus phase margin (PM) for second-order system.

Determination of \( K_c \) from the Bode plot.
2.2 Compensation

2.2.1 Bode’s Gain-Phase Relationship

\[ \angle G(j\omega) = \frac{1}{\pi} \int_{-\infty}^{\infty} \left( \frac{dM}{du} \right) W(u) \, du \text{ radians} \]

\[ \angle G(j\omega) \approx n \times 90^\circ \]

where

- \( M = \log \text{magnitude, } \ln|G(j\omega)|, \)
- \( u = \log \text{normalized frequency} = \ln(\omega/\omega_0), \)
- \( dM/du \equiv \text{slope } n, \text{ as defined in Eq.}(5.26), \)
- \( W(u) = \text{weighting function} = \ln(\coth|u|/2). \)

When designing controllers using Bode plots we should be aware of the Bode gain-phase relationship. That is, for any stable, minimum phase system, the phase of \( G(j\omega) \) is uniquely determined by the magnitude of \( G(j\omega) \) (see above). A fair approximation is

\[ \angle G(j\omega) \approx n \times 90^\circ \quad (103) \]

where \( n \) is the slope of the curve of the magnitude plot. As previously noted, we desire \( \angle G(j\omega) > -80^\circ \) at \( |KG(j\omega)| = 1 \) so a rule of thumb is to adjust the magnitude response so that the slope at crossover \( (\|KG(j\omega)\| = 1) \) is approximately -1, which should give a nice phase margin.

Consider the system

\[ G(s) = \frac{1}{s^2} \quad (104) \]

Clearly the slope at crossover = -2. We can use a PD controller

\[ D(s) = K(T_D s + 1) \quad (105) \]

and pick a suitable \( K \) and \( T_D = \frac{1}{w_3} \) to cause the slope at crossover to be -1. Figures below contain plots of the open-loop and the compensated open-loop Bode plots.
2.2.2 Closed-loop frequency response

For the open-loop system, we will typically have

\[
|G(jw)| \gg 1 \quad \text{for } w \ll w_c, \quad (106)
\]

\[
|G(jw)| \ll 1 \quad \text{for } w \gg w_c \quad (107)
\]

where \( w_c \) is the crossover frequency. We can then approximate the closed-loop frequency response by

\[
|F| = \left| \frac{G(jw)}{1 + G(jw)} \right| \approx \begin{cases} 
1 & \text{for } w \ll w_c \\
|G| & \text{for } w \gg w_c
\end{cases} \quad (108)
\]

For \( w = w_c \), \( |F| \) depends on the phase margin. Figure 5.44 shows the relationship of \( |F| \) on the PM for several different values for the PM. For a PM of 90°, \( |F(jw_c)| = 0.7071 \). Also, if the PM = 90° then bandwidth is equal to \( w_c \). There is a tradeoff between PM and bandwidth.
2.3 Proportional Compensation

Stability example:
(a) system definition and
(b) root locus.
Frequency-response magnitude and phase.

\[ KG(j\omega) \]

\[ G(j\omega) \]

PM (Phase margin)

PM = 0°

PM = +80°

PM = -35°
Bode Diagrams

Gm=0.0206 dB (at 1 rad/sec), Pm=21.386 deg. (at 0.68233 rad/sec)
2.3.1 Proportional/Differential Compensation

A PD controller has the form

\[ D(s) = K(T_Ds + 1) \quad (109) \]

and can be used to add phase lead at all frequencies above the breakpoint. If no change in gain of low-frequency asymptote, PD compensation will increase crossover frequency and speed of response. Increasing frequency-response magnitude at the higher frequencies will increase sensitivity to noise. The figures below show the effect of a PD controller on the frequency response.

\[ D(s) = K(T_Ds + 1) \]
2.3.2 Lead compensation

A lead controller has the form

\[ D(s) = K \frac{T s + 1}{\alpha T s + 1} \]  

(110)

where \( \alpha \) is less than 1. The maximum phase lead then occurs at

\[ w = \frac{1}{\sqrt{\alpha T}} \]  

(111)

Lead compensation adds phase lead at a frequency band between the two breakpoints, which are usually selected to bracket the crossover frequency. If no change in gain of low-frequency asymptote, lead compensation will increase the crossover frequency and speed of response over the uncompensated system. If gain of low-frequency asymptote is reduced in order not to increase crossover frequency, the steady-state errors of the system will increase. Lead compensation acts approximately like a PD compensator, but with less high frequency amplification.
An example of lead control is given in the figures. Given an open-loop transfer function

\[ G(s) = \frac{1}{s(s + 1)} \]  

(112)
design a lead controller that has a steady-state error of less than 10% to a ramp input. Also the system is to have an overshoot \((M_p)\) of less than 25%. The steady-state error is given by

\[ e(\infty) = \lim_{s \to 0} \left[ \frac{1}{1 + D(s)G(s)} \right] R(s) \]  

(113)

where \( R(s) = \frac{1}{s^2} \) for a unit ramp, which reduces to

\[ e(\infty) = \lim_{s \to 0} \left\{ \frac{1}{s + D(s)\left[ \frac{1}{s+1} \right]} \right\} = \frac{1}{D(0)} \]  

(114)

so we can pick a \( K = 10 \). Also, using the relationship between PM and \( M_p \) we can see that we need a PM of 45° to meet the overshoot requirement. We then experiment to find \( T \) and \( \alpha \) and it turns out that the desired compensation is

\[ D(s) = 10 \left( \frac{s}{\frac{5}{10}} + 1 \right) \]  

(115)
2.3.3 Proportional/Integral Compensation

A PI controller has the form

$$D(s) = \frac{K}{s} \left( s + \frac{1}{T_I} \right)$$

(116)

PI control increases frequency-response magnitude at frequencies below the breakpoint thereby decreasing steady-state errors. It also contributes phase lag below the breakpoint, which must be kept at a low enough frequency so that it does not degrade stability excessively. Figures show how PI compensation will effect the frequency response.
2.3.4 Lag Compensation

A lag controller has the form

\[ D(s) = K \frac{T s + 1}{\alpha T s + 1} \]  

where \( \alpha \) is greater than 1. Lag compensation increases frequency-response magnitude at frequencies below the two breakpoints thereby decreasing steady-state errors. With suitable adjustments in loop gain, it can alternatively be used to decrease the frequency-response magnitude at frequencies above the two breakpoints so that the crossover occurs at a frequency that yields an acceptable phase margin. It also contributes phase lag between the two breakpoints, which must be kept at low enough frequencies so that the phase decrease does not degrade stability excessively. Figures show the effect of lag compensation on frequency response.
Question: Does increasing gain $> 1$ or $< 1$ at $180^\circ$ cause stability or instability?
Answer: Usually instability, but in some cases the opposite holds. Use root locus or Nyquist methods to see.

3 Nyquist Diagrams

A Nyquist plot refers to a plot of the magnitude vs. phase, and can be useful when Bode plots are ambiguous with regard to stability. A Nyquist plot results from evaluating some transfer function $H(s)$ for values of $s$ defined by some contour (see figures). If there are any poles or zeros inside the contour then the Nyquist plot will encircle the origin one or more times. The closed-loop transfer function has the form

$$\frac{Y}{R} = \frac{KG(s)}{1 + KG(S)} \quad (118)$$

The closed-loop response is evaluated by looking at

$$1 + KG(S) = 0 \quad (119)$$

which is simply the open-loop response, $KG(s)$, shifted to the right by 1. Thus $1 + KG(S)$ encircles the origin iff $KG(s)$ encircles $-1$. We can define the contour to be the entire RHP (see figures). If there are any encirclements while evaluating our transfer function, we know that the system is unstable.
A clockwise encirclement of -1 indicates the presence of a zero in the RHP while a counter-clockwise encirclement of -1 indicates a pole in the RHP. The net # of clockwise encirclements is

\[ N = Z - P \] (120)

Alternatively, in order to determine whether an encirclement is due to a pole or zero we can write

\[ 1 + KG(s) = 1 + K \frac{b(s)}{a(s)} = \frac{a(s) + Kb(s)}{a(s)} \] (121)

So the poles of \( 1 + KG(s) \) are also the poles of \( G(s) \). Since the number of RHP poles of \( G(s) \) are known, we will assume that an encirclement of -1 indicates an unstable root of the closed-loop system. Thus we have the number of closed-loop RHP roots

\[ Z = N + P \] (122)
3.0.5 Nyquist Examples

Several examples are given. The first example has the transfer function

\[ G(s) = \frac{1}{(s + 1)^2} \]  \hspace{1cm} (123)

The root locus shows that the system is stable for all values of \( K \). The Nyquist plot is shown for \( K = 1 \) and it does not encircle \(-1\). Plots could be made for other values of \( K \), but it should be noted that an encirclement of \(-1\) by \( KG(s) \) is equivalent to an encirclement of \( \frac{1}{K} \) by \( G(s) \). Since \( G(s) \) only crosses the negative real axis at \( G(s) = 0 \), it will never encircle \( \frac{1}{K} \) for positive \( K \).
The second example has the transfer function

\[ G(s) = \frac{1}{s(s + 1)^2} \]  

and is stable for small values of \( K \). As can be seen from the Nyquist plot, larger values of \( K \) will cause two encirclements. The large arc at infinity in the Nyquist plot is due to the pole at 0. Two poles at \( s = 0 \) would have resulted in a full 360° arc at infinity.
The third example given has the transfer function

\[ G(s) = \frac{s + 1}{s(\frac{1}{10} - 1)^2} \]  

For large values of \( K \), there is one counterclockwise encirclement (see figures), so \( N = -1 \). But since \( P = 1 \) from the RHP pole of \( G(s) \), \( Z = 0 \) and there are no unstable system roots. When \( K \) is small, \( N = 1 \) which indicates that \( Z = 2 \), that there are two unstable roots in the closed-loop system.
3.1 Stability Margins

Now we are interested in defining the gain and phase margins in terms of how far the system is from encircling the -1 point. The gain margin is defined as the inverse of $|KG(j\omega)|$ when the plot crosses the negative real axis. The phase margin is defined to be the difference between the phase of $G(j\omega)$ and $-180^\circ$ when the plot crosses the unit circle. Their determination is shown graphically in the figures below. A problem with these definitions is that there may be several GM's and PM's indicated by a Nyquist plot. A proposed solution is the vector margin which is defined to be the distance to the -1 point from the closest approach of the Nyquist plot. The vector margin can be difficult to calculate though.

Recall the definition of sensitivity

$$S = \frac{1}{1 + GD}$$

(126)

The sensitivity minimized over $\omega$ is equal to the inverse of the vector margin. A similar result holds for a MIMO system, where

$$\min \frac{1}{\sigma(S(j\omega))} = \frac{1}{\|S(j\omega)\|_\infty}$$

(127)

where $\sigma$ is the maximum singular value of the matrix and $\| \cdot \|_\infty$ is the infinity norm.
Nyquist plot of a complex system.

Definition of the vector margin on the Nyquist plot.

$M$ circle
Vector margin
Part V
Digital Classical Control

1 Discrete Control - Z-transform

\[ X(z) = \sum_{k=-\infty}^{\infty} x(k)z^{-k} \]

- Examples

- Step response
  \[ 1(k) \rightarrow \frac{1}{1 - z^{-1}} \]

- Exponential decay
  \[ r^k \rightarrow \frac{z}{z - r} \]

- Sinusoid
  \[ \left[ r^k \cos kr \right] 1(k) \rightarrow \frac{z(z - r \cos \theta)}{z^2 - 2r \cos \theta z + r^2} \]

- For stability \( \| \text{poles} \| < 1 \)

- Final value theorem
  \[ \lim_{k \to \infty} x(k) = \lim_{z \to 1} (1 - z^{-1}) X(z) \]

1.1 Continuous to Discrete Mapping

1. Integration Methods

\[ s \leftarrow \frac{z - 1}{T} \text{ forward method} \]
\[ s \leftarrow \frac{z - 1}{Tz} \text{ backward method} \]
\[ s \leftarrow \frac{2z - 1}{Tz + 1} \text{ trapezoidal/bilinear transformation method} \]
2. Pole-zero mapping

3. ZOH equivalent

\[
\begin{array}{c}
\text{u} & \xrightarrow{\text{D/A}} & G(s) & \xrightarrow{\text{A/D}} & y \\
\text{ZOH} & & \text{sample} & \text{sample (linear & time varying)} & \text{(with ant-alias filter)} \\
\end{array}
\]

Gives exact representation between \(u(k), y(k)\) (does not necessarily tell you what happens between samples)

1.2 ZOH

\[
\text{Impulse-response}
\]

\[
\text{sampler - linear, time-variant}
\]

Input \(= 1(t) - 1(t - T)\)

\[
Y(s) = (1 - e^{-Ts}) \frac{G(s)}{s}
\]

\[
Y(z) = z\text{-transform of samples of } y(t)
\]

\[
G(z) = Z\{y(kT)\} = Z\left\{L^{-1}\{Y(s)\}\right\}
\]

\[\triangleq\ Z\{Y(s)\}\text{ for shorthand}\]

\[
= Z\left\{(1 - e^{-Ts}) \frac{G(s)}{s}\right\}
\]

\[
= (1 - z^{-1})Z\left\{\frac{G(s)}{s}\right\}
\]

- Example 1:

\[
G(s) = \frac{a}{s + a}
\]

\[
\frac{G(s)}{s} = \frac{a}{s(s + a)} = \frac{1}{s} - \frac{1}{s + a}
\]

\[
L^{-1}\left\{\frac{G(s)}{s}\right\} = 1(t) - e^{-sT}1(t)
\]

after sampling \(= 1(kT) - e^{-aT}1(kT)\)

Z-transform \(= \frac{1}{1 - z^{-1}} - \frac{1}{1 - e^{-aT}z^{-1}} = \frac{z(1 - e^{-aT})}{(z - 1)(z - e^{-aT})}\)

\[
G(z) = \frac{1 - e^{-aT}}{z - e^{-aT}}
\]
pole at $a \rightarrow$ pole at $e^{-aT}$, no zero $\rightarrow$ no zero

- Example 2:

\[
G(s) = \frac{1}{s^2} \\
G(z) = (1 - z^{-1}) \left\{ \frac{1}{s^3} \right\} \\
G(z) = \frac{T^2(z + 1)}{2(z - 1)^2} \quad \text{Using Table lookup}
\]

pole at 0 $\rightarrow$ pole at $e^{-0T} = 1$
zero at $\infty$ $\rightarrow$ zero at $-1$

- In general,

\[
z = e^{\theta T} \quad \text{for poles}
\]

zeros cannot be mapped by the same relationship.

- For 2nd order systems - given parameters $(M_p, t_r, t_s, \xi)$

  The poles $s_1, s_2 = -a \pm jb$

  map to $z = re^{\pm j\theta}, r = e^{-aT}, \theta = bT$

  smaller the value of $T$ $\rightarrow$ closer to the unit circle, i.e. closer to $z = 1$

Additional comments on ZOH

\[
U(k) \rightarrow Y(k)
\]

What we really want to control is the continuous output $y(t)$.

Consider sinusoid $\rightarrow$ sampled $\rightarrow$ ZOH
As seen in the figure,

- fundamental frequency is still the same
- but introduction of phase delay - reduced phase margin (bad for control)
- adds higher harmonics.

\[
\frac{1 - e^{-sT}}{s} \rightarrow s = -j\omega \Rightarrow e^{-\frac{j\omega T}{2}}T\text{sinc}\left(\frac{\omega T}{2}\right)
\]
As seen in the figure, extra harmonics excite $G(s)$, this may cause aliasing and other effects. Therefore we need to add an anti-aliasing filter as shown

1.3 $Z$-plane and dynamic response

Descriptions of corresponding lines in $s$-plane and $z$-plane.

<table>
<thead>
<tr>
<th>$s$-plane</th>
<th>Symbol</th>
<th>$z$-plane</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s = j\omega$</td>
<td>$\times \times$</td>
<td>$</td>
</tr>
<tr>
<td>Real frequency axis</td>
<td></td>
<td>Unit circle</td>
</tr>
<tr>
<td>$s = \sigma \geq 0$</td>
<td>$\square \square \square$</td>
<td>$z = r \geq 1$</td>
</tr>
<tr>
<td>$s = \sigma \leq 0$</td>
<td>$\bigcirc \bigcirc \bigcirc$</td>
<td>$z = r, 0 \leq r \leq 1$</td>
</tr>
<tr>
<td>$s = -\zeta \omega_n + j \omega_n \sqrt{1 - \zeta^2}$</td>
<td>$\triangle \triangle \triangle$</td>
<td>$z = re^{j\theta}$ where $r = \exp(-\zeta \omega_n T)$</td>
</tr>
<tr>
<td>$s = -\zeta \omega_n + j \omega_n \sqrt{1 - \zeta^2}$</td>
<td></td>
<td>$= e^{-aT}$</td>
</tr>
<tr>
<td>Constant damping ratio</td>
<td></td>
<td>$\theta = \omega_n T \sqrt{1 - \zeta^2} = bT$</td>
</tr>
<tr>
<td>if ( \zeta ) is fixed and ( \omega_n ) varies</td>
<td></td>
<td>Logarithmic spiral</td>
</tr>
<tr>
<td>$s = \pm j(\pi/T) + \sigma, \sigma \leq 0$</td>
<td>$\cdots \cdots$</td>
<td>$z = -r$</td>
</tr>
</tbody>
</table>

Plot of $z = e^{sT}$ for various $s$:
Typically in digital control we can place poles anywhere in the left-half plane, but because of oscillation, it is preferred to work with positive real part only.

The $z$-plane grid is given by "zgrid" in MATLAB.
Higher order systems

- as pole comes closer to \( z=1 \), the system slows down
- as zero comes closer to \( z=1 \), causes overshoot
- as pole and zero come close to each other, they tend to cancel each other

**Dead-beat control.** In digital domain we can do several things not possible in continuous domain, e.g., pole-zero cancellation, or suppose we set all closed loop poles to \( z = 0 \).

Consider closed loop response

\[
\frac{b_1 z^3 + b_2 z^2 + b_3 z + b_4}{z^4} = b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4}
\]

All poles are at 0. This is called dead-beat control.

## 2 Root Locus control design

2 methods:

1. Emulation
   
   Transform \( D(s) \to D(z) \)

   This is generally considered an older approach.

2. Direct design in the \( z \)-domain - The following steps can be followed:

   (a)

   \( ZOH + G(s) \to G(z) \)

   (b) design specifications

   \( M_p, t_r, t_s \to \) pole locations in \( z \)-domain

   (c) Compensator design : Root locus or Bode analysis or Nyquist analysis in \( Z \)-domain

---

The mapping of \( s \)-plane specifications to the \( z \)-plane. (a) Damping; (b) frequency; (c) settling time.
\[
\frac{y}{r} = \frac{D(z)G(z)}{1 + D(z)G(z)}
\]

Solution of the characteristic equation

\[1 + kD(z)G(z) = 0\]

gives the root locus. The rules for plotting are exactly the same as the continuous case.

Ex. 1.

\[
\frac{a}{s(s + a)}
\]

\[G(z) \text{ with ZOH } = k\frac{z + 0.9}{(z - 1)(z - e^{-aT})}\]

* not great damping ratio as K increases
* lead control - add zero to cancel pole

Better Lead design (gives better damping at higher K - better Kv)
2.1 Comments - Latency

\( u(k) \) is a function of \( y(k) \) (e.g. lead difference equation).

- computation time (in computer) introduces delay.
- Worst case latency - to see the effect of a full 1 clock delay \((z^{-1})\), add a pole at the origin i.e

![Diagram](image)

- To overcome this problem we may try to sample faster than required.
  sample faster \(\rightarrow\) complex control
  sample slower \(\rightarrow\) more latency

3 Frequency Design Methods

\[ z = e^{j\omega} \]

Rules for construction of Bode plot and Nyquist plots are the same as in continuous domain. Definitions for GM and PM are the same. However, plotting by “hand” is not practical. (Use \texttt{dbode} in MATLAB.)

3.1 Compensator design

Proportional \( k \rightarrow k \)

Derivative \(-ks \rightarrow k \frac{1 - z^{-1}}{T} = k \frac{z - 1}{z} \)

PD \( \rightarrow k \frac{z - \alpha}{z} \)

Integral \( \frac{k}{s} \rightarrow \frac{k}{1 - z^{-1}} \)

PID \( D(z) = k_p \left( 1 + \frac{k_d z}{z-1} + \frac{k_p (z-1)}{z} \right) \)
3.2 Direct Method (Ragazzini 1958)

Closed Loop Response

\[ H(z) = \frac{DG}{1 + DG} \]

\[ D = \frac{1}{G(z)} \frac{H(z)}{1 - H(z)} \]

- Sometimes if \( H(z) \) is not chosen properly, \( D(z) \) may be non-causal, non-minimal phase or unstable.

- Causal \( D(z) \) (cannot have a pole at infinity) implies that \( H(z) \) must have zero at infinity of the same order as zero of \( G(z) \) at infinity.

- For stability of \( D(z) \),
  - \( 1 - H(z) \) must contain as zeros, all poles of \( G(z) \) that are outside the unit circle
  - \( H(z) \) must contain as zeros all zeros of \( G(z) \) that are outside the unit circle

- Adding all these constraints to \( H(z) \), gives rise to simultaneous equations which are tedious to solve.

- This method is somewhat similar to a method called Pole Placement, though it is not as easy as one might think. It does not necessarily yield good phase-margin, gain-margin etc. There are more effective method is State-Space Control.

4 Z-Transform Tables

Let \( \mathcal{F}_1(s) \) be the Laplace transform of \( f_1(t) \) and \( \mathcal{F}_1(z) \) be the z-transform of \( f_1(kT) \).

<table>
<thead>
<tr>
<th>Number</th>
<th>Laplace Transform</th>
<th>Samples</th>
<th>z-Transform</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \alpha F_1(s) + \beta F_2(s) )</td>
<td>( \alpha f_1(kT) + \beta f_2(kT) )</td>
<td>( \alpha F_1(z) + \beta F_2(z) )</td>
<td>Discrete convolution corresponds to product of z-transforms</td>
</tr>
<tr>
<td>2</td>
<td>( \mathcal{F}_1(e^{sT}) \mathcal{F}_2(s) )</td>
<td>( \sum_{t=-\infty}^\infty f_1(\ell T) f_2(kT - \ell T) )</td>
<td>( F_1(z) F_2(z) )</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>( e^{+nT} \mathcal{F}_1(s) )</td>
<td>( f(kT + nT) )</td>
<td>( z^n F(z) )</td>
<td>Shift in time</td>
</tr>
<tr>
<td>4</td>
<td>( \mathcal{F}(s + a) )</td>
<td>( e^{-atkT} f(kT) )</td>
<td>( F(e^{aT} z) )</td>
<td>Shift in frequency</td>
</tr>
<tr>
<td>5</td>
<td>( \lim_{k \to \infty} f(kT) )</td>
<td>( \lim_{z \to 1} F(z) )</td>
<td>If all poles of ( (z - 1)F(z) ) are inside the unit circle and ( F(z) ) converges for ( 1 \leq</td>
<td>z</td>
</tr>
<tr>
<td>6</td>
<td>( \mathcal{F}(s/\omega_n) )</td>
<td>( f(\omega_n kT) )</td>
<td>( F(z; \omega_n) )</td>
<td>Time and frequency scaling</td>
</tr>
<tr>
<td>7</td>
<td>( \mathcal{F}_1(kT) \mathcal{F}_2(\omega_n) )</td>
<td>( f_1(kT) f_2(\omega_n kT) )</td>
<td>( \frac{1}{2\pi j} \oint \mathcal{F}_1(\zeta) \mathcal{F}_2(z/\zeta) \frac{dz}{\zeta} )</td>
<td>Time product</td>
</tr>
<tr>
<td>8</td>
<td>( \mathcal{F}_3(s) = \mathcal{F}_1(s) \mathcal{F}_2(s) )</td>
<td>( \int_{-\infty}^\infty f_1(\tau) f_2(kT - \tau) d\tau )</td>
<td>( F_3(z) )</td>
<td>Continuous convolution does not correspond to product of z-transforms</td>
</tr>
</tbody>
</table>
Table B.2

<table>
<thead>
<tr>
<th>Number</th>
<th>( F(s) )</th>
<th>( f(nT) )</th>
<th>( F(z) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>--</td>
<td>( 1, n = 0; 0 \neq n \neq 0 )</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>--</td>
<td>( 1, n = k; 0 \neq k )</td>
<td>( z^{-k} )</td>
</tr>
<tr>
<td>3</td>
<td>( \frac{1}{s} )</td>
<td>( 1(nT) )</td>
<td>( z )</td>
</tr>
<tr>
<td>4</td>
<td>( \frac{1}{s^2} )</td>
<td>( nT )</td>
<td>( \frac{z}{T^2} )</td>
</tr>
<tr>
<td>5</td>
<td>( \frac{1}{s^3} )</td>
<td>( \frac{1}{2!}(nT)^2 )</td>
<td>( \frac{T^2}{2}(z + 1) )</td>
</tr>
<tr>
<td>6</td>
<td>( \frac{1}{s^4} )</td>
<td>( \frac{1}{3!}(nT)^3 )</td>
<td>( \frac{T^3}{6}(z^2 + 4z + 1) )</td>
</tr>
<tr>
<td>7</td>
<td>( \frac{1}{s^m} )</td>
<td>( \lim_{a \rightarrow 0} \frac{(-1)^{m-1} \partial^{m-1} e^{-anT}}{(m-1)! \partial a^{m-1} z - e^{-aT}} )</td>
<td>( \frac{z}{T} )</td>
</tr>
<tr>
<td>8</td>
<td>( \frac{1}{s + a} )</td>
<td>( e^{-anT} )</td>
<td>( \frac{z}{z - e^{-aT}} )</td>
</tr>
<tr>
<td>9</td>
<td>( \frac{1}{(s + a)^2} )</td>
<td>( nT e^{-anT} )</td>
<td>( \frac{z}{z - e^{-aT}} )</td>
</tr>
<tr>
<td>10</td>
<td>( \frac{1}{(s + a)^3} )</td>
<td>( \frac{1}{2}(nT)^2 e^{-anT} )</td>
<td>( \frac{2}{z - e^{-aT}} )</td>
</tr>
<tr>
<td>11</td>
<td>( \frac{1}{(s + a)^m} )</td>
<td>( \frac{(-1)^{m-1} \partial^{m-1} (e^{-anT})}{(m-1)! \partial a^{m-1} z - e^{-aT}} )</td>
<td>( \frac{z}{z - e^{-aT}} )</td>
</tr>
<tr>
<td>12</td>
<td>( \frac{a}{s(s + a)} )</td>
<td>( 1 - e^{-anT} )</td>
<td>( \frac{z(1 - e^{-aT})}{(z - 1)(z - e^{-aT})} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number</th>
<th>( F(s) )</th>
<th>( f(nT) )</th>
<th>( F(z) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>( a )</td>
<td>( \frac{1}{a}(anT - 1 + e^{-anT}) )</td>
<td>( \frac{z[anT - 1 + e^{-aT}z + (1 - e^{-aT} - aTe^{-aT})]}{z - e^{-aT}} )</td>
</tr>
<tr>
<td>14</td>
<td>( b - a )</td>
<td>( e^{-anT} - e^{-bnT} )</td>
<td>( \frac{(z - e^{-aT})z}{z - e^{-bT}} )</td>
</tr>
<tr>
<td>15</td>
<td>( \frac{s}{(s + a)^2} )</td>
<td>( (1 - anT)e^{-anT} )</td>
<td>( \frac{z}{z - e^{-aT}} )</td>
</tr>
<tr>
<td>16</td>
<td>( \frac{1}{(s + a)^2} )</td>
<td>( 1 - e^{-anT}(1 + anT) )</td>
<td>( \frac{z}{z - e^{-aT}} )</td>
</tr>
<tr>
<td>17</td>
<td>( \frac{(b - a)s}{(b + a)s} )</td>
<td>( be^{-bnT} - ae^{-anT} )</td>
<td>( \frac{z}{z - e^{-bT}} )</td>
</tr>
<tr>
<td>18</td>
<td>( \frac{a}{s^2 + a^2} )</td>
<td>( \sin anT )</td>
<td>( z )</td>
</tr>
<tr>
<td>19</td>
<td>( \frac{a}{s^2 + a^2} )</td>
<td>( \cos anT )</td>
<td>( z )</td>
</tr>
<tr>
<td>20</td>
<td>( \frac{(s + a)^2 + b^2}{b} )</td>
<td>( e^{-anT}\cos bnT )</td>
<td>( \frac{z}{z - e^{-aT}\cos bT} )</td>
</tr>
<tr>
<td>21</td>
<td>( \frac{(s + a)^2 + b^2}{a^2 + b^2} )</td>
<td>( e^{-anT}\sin bnT )</td>
<td>( \frac{z}{z - e^{-aT}\cos bT} )</td>
</tr>
<tr>
<td>22</td>
<td>( \frac{1}{s((s + a)^2 + b^2)} )</td>
<td>( 1 - e^{-anT}(\cos bnT + \frac{a}{b}\sin bnT) )</td>
<td>( \frac{z}{z - 2e^{-aT}(\cos bnT)z + e^{-2aT}} )</td>
</tr>
</tbody>
</table>
Part VI
State-Space Control

1 State-Space Representation

- Algebraic based method of doing control.
- Developed in the 1960’s.
- Often called “Modern” Control Theory.

Example

\[ f = ma \]
\[ \mu = \frac{dx^2}{dt} = mx \]

Let \( x_1 = x \) (position)
\[ x_2 = \dot{x} \) (velocity)\]

\[
\begin{pmatrix}
\dot{x}_1 \\
\dot{x}_2
\end{pmatrix} =
\begin{pmatrix}
0 & 1 \\
0 & 0
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2
\end{pmatrix} +
\begin{pmatrix}
0 \\
\frac{1}{m}
\end{pmatrix} u
\]

\[ \dot{x} = Fx + Gu \rightarrow \text{state equation} \]
linear in \( x, u \)

\[ x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \rightarrow \text{State} \]

Any linear, finite state, dynamic equation can be expressed in this form.

1.1 Definition
The state of a system at time \( t_0 \) is the minimum amount of information at \( t_0 \) that (together with \( u(t), t \geq t_0 \)) uniquely determines behavior of the system for all \( t \geq t_0 \).
1.2 Continuous-time

First order ODE

\[
\dot{x} = f(x, u, t) \\
y = h(x, u, t)
\]

1.3 Linear Time Invariant Systems

\[
\begin{align*}
\dot{x} &= Fx + Gu \\
y &= Hx + Ju \\
x(t) &\in \mathbb{R}^n \text{ (state)} \\
u(t) &\in \mathbb{R} \text{ (input)} \\
y(t) &\in \mathbb{R} \text{ (output)} \\
F &\in \mathbb{R}^{n\times n} \\
G &\in \mathbb{R}^{1\times n} \\
H &\in \mathbb{R}^{m\times n} \\
J &\in \mathbb{R}
\end{align*}
\]

Sometimes we will use \(A, B, C, D\) instead of the variables \(F,G, H,J\). Note that dimensions change for MIMO systems.

1.4 "Units" of \(F\) in physical terms

\[
\frac{1}{\text{time}} = \text{freq}
\]

- relates derivative of a term to the term
- \(\dot{x} = ax\), large \(a\) \(\Rightarrow\) faster the response.

1.5 Discrete-Time

\[
\begin{align*}
x_{k+1} &= Fx_k + Gu_k \\
y_k &= Hx_k + Ju_k
\end{align*}
\]

Sometimes we make use of the following variables,

\[
\begin{align*}
F &\rightarrow \phi, A \\
G &\rightarrow \Gamma, B \\
H &\rightarrow C \\
J &\rightarrow D
\end{align*}
\]
1.5.1 Example 2 - "analog computers"

\[
\ddot{y} + 3\dot{y} - y = u \\
\frac{Y(s)}{U(s)} = \frac{1}{s^2 + 3s - 1}
\]

- Never use differentiation for implementation.
- Instead, use integral model.

\[
\begin{pmatrix}
\dot{x}_1 \\
\dot{x}_2
\end{pmatrix} = \begin{pmatrix}
-3 & 1 \\
1 & 0
\end{pmatrix} \begin{pmatrix}
x_1 \\
x_2
\end{pmatrix} + \begin{pmatrix}
1 \\
0
\end{pmatrix} u
\]

\[
y = \begin{pmatrix}
0 & 1
\end{pmatrix} \begin{pmatrix}
x_1 \\
x_2
\end{pmatrix}
\]

Later we will show standard (canonical) forms for implementing ODE's or TF's.

1.6 State Space Vs. Classical Approach

- SS is an internal description of the system whereas Classical approach is an external description.
- SS gives additional insight into the control problem and may tell us things about a system that a TF approach misses (e.g. controllability, observability, internal stability).
- Generalizes well to MIMO.
- Helps in optimal control design / estimation.
- Design
  - Classical - given specifications are natural frequency, damping, rise time, GM, PM

Design a compensator (Lead/Lag PID) \hspace{1cm} R.L./Bode \hspace{1cm} find closed loop poles which hopefully meet specifications
- SS - given specifications are pole/zero locations or other constraints.

1.7 Linear Systems we won’t study

1. 

\[ y(t) = u(t - 1) \]

This simple continuous system of a 1 second delay is infinite dimensional.

\[ x(t) = [u(t)]_{t-1}^t \text{ (state } [0, 1] \rightarrow R) \]

2. Cantilever beam

\[ y = \text{deflection} \]

Infinite dimensional, distributed system, PDE
1.8 Linearization

\[ \dot{\theta} = \tau - \sin \theta \]
\[ x_1 = \theta, \ x_2 = \dot{\theta} \]

Non-linear state-space representation.

\[
\dot{x} = \begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 \\ -\sin x_1 + u \end{pmatrix} = f(x, u) = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}
\]

Small-signal approximation (linearization)

\[
f(x, u) = f(0, 0) + \frac{\partial f}{\partial x}(0, 0)x + \frac{\partial f}{\partial u}(0, 0)u + \ldots
\]

\(f(0, 0) = 0\), assume 0 is an equilibrium point i.e. \(\dot{x}(t) = 0\)

\[
\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \ldots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \ldots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \ldots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} = F
\]

\[
\frac{\partial f}{\partial u} = \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \vdots \\ \frac{\partial f_n}{\partial u} \end{bmatrix} = G
\]

Output

\[
y = h(x, u) \approx \frac{\partial h}{\partial x}x + \frac{\partial h}{\partial u}u
\]

\[
\dot{\hat{x}} = \begin{pmatrix} x_2 \\ -\sin x_1 + u \end{pmatrix}
\]

\[
F = \begin{bmatrix} 0 & 1 \\ -\cos x_1 & 0 \end{bmatrix}, \ G = \begin{bmatrix} 0 \\ 1 \end{bmatrix}
\]

91
Equilibrium at
\[ x = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \text{ or } \begin{pmatrix} \pi \\ 0 \end{pmatrix} \]

1.9 State-transformation

\[ ODE \rightarrow F, G, H, J \]

Is \( x(t) \) Unique? NO

Consider,
\[
\begin{align*}
z &= T^{-1}x \\
x &= Tz \\
\text{Given } x \rightarrow z, z \rightarrow x
\end{align*}
\]

\( x, z \) contain the same amount of information, thus \( z \) is a valid state

\[
\begin{align*}
\dot{x} &= Fx + Gu \\
y &= Hx + Ju \\
\dot{z} &= T\dot{z} = FTz + Gu \\
\dot{z} &= T^{-1}FTz + T^{-1}Gu \\
y &= HTz + Ju
\end{align*}
\]

- There are an infinite number of equivalent state-space realizations for a given system
- State need not represent physical quantities

\[
T^{-1} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}
\]

\[
z = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 - x_2 \\ x_1 + x_2 \end{bmatrix}
\]

- A transformation \( T \) may yield an \( A, B, C, D \) which may have a "nice" structure even though \( z \) has no physical intuition. We'll see how to find transformation \( T \) later.

1.10 Transfer Function

1.10.1 Continuous System

\[
\begin{align*}
\dot{X} &= AX + BU \\
Y &= CX + DU
\end{align*}
\]

Take Laplace-transform:
\[ sX(s) - X(0) = AX(s) + BU(s) \]
\[ Y(s) = CX(s) + DU(s) \]

\[ (sI - A)X(s) = X(0) + BU(s) \]
\[ X(s) = (sI - A)^{-1}X(0) + (sI - A)^{-1}BU(s) \]
\[ Y(s) = [C(sI - A)^{-1}B + D]U(s) + C(sI - A)^{-1}X(0^-) \]

If \( X(0^-) = 0 \),
\[ \frac{Y(s)}{U(s)} = G(s) = C(sI - A)^{-1}B + D. \]

How to compute \((sI - A)^{-1}\):
\[ (sI - A)^{-1} = \frac{\text{ADJ}(sI - A)}{\text{det}(sI - A)}. \]

If \( \text{det}(sI - A) \neq 0 \), \((sI - A)^{-1}\) exists, and poles of the system are eigenvalues of \( A \).

**Example:**
For \( A = \begin{bmatrix} -3 & 1 \\ 1 & 0 \end{bmatrix} \) \( B = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \) \( C = [1 \ 0] \) and \( D = 0 \),
\[ G(s) = C(sI - A)^{-1}B \]
\[ = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} s + 3 & -1 \\ -1 & s \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \]
\[ = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} s & -1 \\ -1 & s + 3 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \]
\[ = \frac{1}{s^2 + 3s - 1}. \]

**1.10.2 Discrete System**

\[ X_{k+1} = \Phi X_k + \Gamma U_k \]
\[ Y_k = HX_k. \]

Take Z-transform, we get
\[ X(z) = (zI - \Phi)^{-1}\Gamma U(z) + (zI - \Phi)^{-1}zX(0) \]
\[ Y(z) = H(zI - \Phi)^{-1}\Gamma U(z) + H(zI - \Phi)^{-1}zX(0). \]
1.11 Example - what transfer function don't tell us.

The purpose of this exercise is to show what transfer functions do not tell us. For the system in Figures below,

\[
\begin{align*}
\begin{array}{c}
\text{u} \\
\hline
\text{s - 1} \\
\hline
\frac{1}{s + 1} \\
\hline
\text{y}
\end{array}
\end{align*}
\]

The state-space equations of the system are as follows:

\[
\begin{align*}
\begin{bmatrix}
x_1' \\
x_2'
\end{bmatrix} &= \begin{bmatrix}
-1 & 0 \\
1 & 1
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2
\end{bmatrix} + \begin{bmatrix}
-2 \\
1
\end{bmatrix} U \\
y &= \begin{bmatrix}
0 & 1
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}.
\end{align*}
\]

Then we have

\[
\begin{align*}
\begin{bmatrix}
x_1(s) - x_1(0) \\
x_2(s) - x_2(0)
\end{bmatrix} &= \begin{bmatrix}
-1 & 0 \\
1 & 1
\end{bmatrix} \begin{bmatrix}
x_1(s) \\
x_2(s)
\end{bmatrix} + \begin{bmatrix}
-2 \\
1
\end{bmatrix} u(s) \\
x_1(s) &= \frac{1}{s + 1} x_1(0) - \frac{2}{s + 1} u(s) \\
x_2(s) &= \frac{1}{s - 1} \left[ x_2(0) - \frac{1}{2} x_1(0) \right] + \frac{1}{2(s + 1)} x_1(0) + \frac{1}{s + 1} u(s).
\end{align*}
\]

Take \( L^{-1} \)-transform,

\[
\begin{align*}
x_1(t) &= e^{-t} x_1(0) - 2e^{-t} u(t) \\
y(t) &= x_2(t) \\
&= e^{t} \left[ x_2(0) - \frac{1}{2} x_1(0) \right] + \frac{1}{2} e^{-t} x_1(0) + e^{-t} u(t).
\end{align*}
\]

When \( t \to \infty \), the term \( e^{t} \left[ x_2(0) - \frac{1}{2} x_1(0) \right] \) goes to \( \infty \) unless \( x_2(0) = \frac{1}{2} x_1(0) \). The system is uncontrollable.

Now consider the system below:

\[
\begin{align*}
\begin{array}{c}
\text{u} \\
\hline
\frac{1}{s - 1} \\
\hline
\frac{s - 1}{s + 1} \\
\hline
\text{y}
\end{array}
\end{align*}
\]
we have
\[
\begin{bmatrix}
  x_1' \\
  x_2'
\end{bmatrix}
= 
\begin{bmatrix}
  1 & 0 \\
  -2 & -1
\end{bmatrix}
\begin{bmatrix}
  x_1 \\
  x_2
\end{bmatrix}
+ 
\begin{bmatrix}
  1 \\
  0
\end{bmatrix}
 u
\]
\[
y = 
\begin{bmatrix}
  1 & 1
\end{bmatrix}
\begin{bmatrix}
  x_1 \\
  x_2
\end{bmatrix}
\]

Take \(L^{-1}\)-transform,
\[
x_1(t) = e^t x_1(0) + e^t u(t)
\]
\[
x_2(t) = e^{-t} x_2(0) + x_1(0) - e^{-t} x_1(0) + e^{-t} u(t) - e^t u(t)
\]
\[
y(t) = x_1(t) + x_2(t)
\]
\[
= e^{-t} (x_1(0) + x_2(0)) + e^{-t} u(t)
\]

The system state blows up, but the system output \(y(t)\) is stable. The system is unobservable. As we will see, a simple inspection of \(A, B, C, D\) tells us observability or controllability.

1.12 Time-domain Solutions

For a system
\[
\dot{X} = AX + BU
\]
\[
Y = CX \, ,
\]
\[
X(s) = (sI - A)^{-1} X(0) + (sI - A)^{-1} BU(s)
\]
\[
Y(s) = CX(s) \, .
\]

The term \((sI - A)^{-1} X(0)\) corresponds to the “homogeneous solution”, and the term \((sI - A)^{-1} BU(s)\) is called “particular solution”.

Define the “resolvent matrix” \(\phi(s) = (sI - A)^{-1}\), then \(\phi(t) = L^{-1}\{ (sI - A)^{-1} \}\), and
\[
X(t) = \phi(t) X(0) + \int_0^t \phi(\tau) BU(t - \tau) d\tau \, .
\]

Because
\[
\phi(s) = (sI - A)^{-1}
\]
\[
= \frac{1}{s} (I - \frac{A}{s})^{-1}
\]
\[
= \frac{1}{s} + \frac{A}{s^2} + \frac{A^2}{s^3} + \cdots \, ,
\]
we have

\[ \phi(t) = 1 + At + \frac{(At)^2}{2!} + \frac{(At)^3}{3!} + \cdots \]

\[ = \sum_{k=0}^{\infty} \frac{(At)^k}{k!} \]

\[ = e^{At}. \]

Compare the solution of \( \dot{X} = AX \) with the solution of \( \dot{x} = ax \),

\[ X(t) = e^{At}X(0) \]

\[ x = e^{at}x(0) \]

we observed that they have similar format.

To check whether \( X(t) = e^{At}X(0) \) is correct, we calculate \( de^{At}/dt \)

\[ \frac{de^{At}}{dt} = \frac{d}{dt} \left( \sum_{k=0}^{\infty} \frac{(At)^k}{k!} \right) = A \sum_{k=1}^{\infty} \frac{(At)^{k-1}}{(k-1)!} = Ae^{At}. \]

From above equation and \( X(t) = e^{At}X(0) \), we have

\[ \dot{X} = \frac{dX}{dt} = Ae^{At}X(0) = AX. \]

The matrix exponential \( e^{At} \) has the following properties:

\[ e^{A(t_1 + t_2)} = e^{At_1} \cdot e^{At_2} \]

\[ e^{(A+B)t} = e^{At} \cdot e^{Bt}, \text{ only if } (AB = BA). \]

The conclusion of this section:

\[ X(t) = e^{At}X(0) + \int_0^t e^{As}BU(t - \tau)d\tau \]

\[ Y(t) = CX(t). \]

If the matrix \( A \) is diagonal, that is

\[ A = \begin{bmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \ddots \end{bmatrix}, \]

then

\[ e^{At} = \begin{bmatrix} e^{\lambda_1 t} & & \\ & e^{\lambda_2 t} & \\ & & \ddots \end{bmatrix}. \]

If \( A \) is not diagonal, we use the following Eqn. to transform \( A \) to a diagonal matrix,

\[ A = T^{-1}AT, \]
where the columns of T are eigenvectors of A.
Then
\[ e^{At} = T e^{At} T^{-1}. \]

**Example:**
For matrix \( A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \) \((sI - A)^{-1} = \begin{bmatrix} 1/s & 1/s^2 \\ 0 & 1/s \end{bmatrix},\) we have
\[ e^{At} = L^{-1} \{(sI - A)^{-1}\} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}. \]

Calculating \( e^{At} \) using power series
\[
e^{At} = I + At + \frac{(At)^2}{2} + \cdots
= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & t \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} + \cdots
= \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}.
\]

**Note:** there is no best way to calculate a matrix exponential. A reference is “19 Dubious method of calculating \( e^{At} \)” by Cleve and Moller.

Consider \( e^{-100} = 1 - 100 + \frac{100^2}{2!} - \frac{100^3}{3!} + \cdots. \)

**State-transition matrices**
\[
e^{Ft} e^{F(-t)} = e^{F(t-t)} = I
( e^{Ft} )^{-1} = e^{-Ft}.
\]

Thus,
\[
X(t_0) = e^{Ft_0} X(0)
X(0) = e^{-Ft_0} X(t_0)
X(t) = e^{Ft} e^{-Ft_0} X(t_0)
\]
\[ = e^{F(t-t_0)} X(t_0). \]

The \( e^{F(t-t_0)} \) is the state-transition matrix \( \phi(t - \tau) \). The last equation relates state at time \( t_0 \) to state at time \( t \).

For a time variant system,
\[
X(t) = A(t) X(t),
\]
\[
\frac{\partial X(t)}{\partial t} = \phi(t, t_0) X(t_0)
\]
\[
\frac{\partial \phi(t, t_0)}{\partial t} = A(t) \phi(t, \tau).
\]
Because
\[
\phi(t, t) = I \\
\phi(t_3, t_1) = \phi(t_3, t_2) \phi(t_2, t_1) \\
I = \phi(t, \tau) \phi(\tau, t),
\]
we have
\[
[\phi(t, \tau)]^{-1} = \phi(\tau, t).
\]

### 1.13 Poles and Zeros from the State-Space Description

For a transfer function
\[
G(s) = \frac{Y(s)}{U(s)} = \frac{b(s)}{a(s)},
\]
a pole of the transfer function is the “natural frequency”, that is the motion of \(aY\) for input \(U \equiv 0\). In the state form we have,
\[
\dot{X} = AX \text{ and } X(0) = V_i.
\]
Assume \(X(t) = e^{st}V_i\), then we have
\[
\dot{X}(t) = s_i e^{st}V_i \\
= AX(t) \\
= Ae^{st}V_i.
\]
So,
\[
s_i V_i = AV_i,
\]
that is \(s_i\) is an eigenvalue of \(A\) and \(V_i\) is an eigenvector of \(A\). We conclude poles = \(\text{eig}(A)\).

A zero of the system is a value \(s = s_i\), such that if input \(u = e^{st}U_i\), then \(y \equiv 0\). For a system in state form
\[
\dot{X} = FX + GU \\
Y = HX,
\]
a zero is a value \(s = s_i\), such that
\[
U(t) = e^{st}U(0) \\
X(t) = e^{st}X(0) \\
Y(t) \equiv 0.
\]
Then we have
\[
\dot{X} = s_i e^{st}X(0) \\
= Fe^{st}X(0) + Ge^{st}U(0), \text{ or}
\]
\[
\begin{bmatrix}
  s_i - F \\
  -G
\end{bmatrix}
\begin{bmatrix}
  X(0) \\
  U(0)
\end{bmatrix} = 0.
\]
We also have
\[ Y = HX + JU = 0, \text{ or} \]
\[ \begin{bmatrix} H & J \end{bmatrix} \begin{bmatrix} X(0) \\ U(0) \end{bmatrix} = 0. \]

Combine them together, we have
\[ \begin{bmatrix} sI - F & -G \\ H & J \end{bmatrix} \begin{bmatrix} X(0) \\ U(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \]

Non-trivial solution exists for \[ \det \begin{bmatrix} sI - F & -G \\ H & J \end{bmatrix} \neq 0. \] (For mimo systems, look to where this matrix loses rank.)

The transfer function
\[ G(s) = H(sI - F)^{-1}G = \frac{\det \begin{bmatrix} sI - F & -G \\ H & J \end{bmatrix}}{\det[sI - F]}. \]

Example:
For a transfer function
\[ G(s) = \frac{s + 1}{(s + 2)(s + 3)} = \frac{s + 1}{s^2 + 5s + 6}. \]

In state form, the system is
\[ F = \begin{bmatrix} -5 & -6 \\ 1 & 0 \end{bmatrix}, \quad G = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 1 \end{bmatrix}. \]

The poles of the system are found by solving \( \det[sI - F] = 0 \), that is
\[ \det[sI - F] = \det \begin{bmatrix} s + 5 & 6 \\ -1 & s \end{bmatrix} = 0 \text{ or} \]
\[ s(s + 5) + 6 = s^2 + 5s + 6 = 0 \]

The zeros of the system are found by solving \( \det \begin{bmatrix} sI - F & -G \\ H & J \end{bmatrix} = 0 \), that is
\[ \det \begin{bmatrix} sI - F & -G \\ H & J \end{bmatrix} = \det \begin{bmatrix} s + 5 & 6 & -1 \\ -1 & s & 0 \\ 1 & 1 & 0 \end{bmatrix} = -1(-1 - s) = s + 1 \]
1.14 Discrete-Systems

1.14.1 Transfer Function

\[ X_{k+1} = \Phi X_k + \Gamma U_k \]
\[ Y_k = HX_k. \]

Then

\[
\frac{Y(z)}{U(z)} = H(zI - \Phi)^{-1}\Gamma = \frac{\det \begin{bmatrix} zI - \Phi & -\Gamma \\ H & 0 \end{bmatrix}}{\det[zI - \Phi]}. 
\]

Example: Calculating \( X_n \) from \( X_0 \).

\[
\begin{align*}
X_1 &= \Phi X_0 + \Gamma U_0 \\
X_2 &= \Phi(X_0 + \Gamma U_0) + \Gamma U_1 \\
X_3 &= \Phi(\Phi X_0 + \Gamma U_0) + \Gamma U_1 + \Gamma U_2 \\
&\quad \vdots \\
X_n &= \Phi^n X_0 + \sum_{i=1}^{n} \Phi^{i-1} \Gamma U_{n-i}
\end{align*}
\]

1.14.2 Relation between Continuous and Discrete

ZOH equivalent:

\[
\dot{X} = FX + GU \\
Y = HX + JU \\
X(t) = e^{F(t-t_0)} X(t_0) + \int_{t_0}^{t} e^{F(t-\tau)} GU(\tau) d\tau.
\]

Let’s solve over one sampling period: \( t = nT + T \) and \( t_0 = nT \), we have

\[
X(nT + T) = e^{FT} X(nT) + \int_{nT}^{(n+1)T} e^{F(T-T-\tau)} GU(\tau) d\tau.
\]
Because $U(\tau) = U(nT) = \text{const}$ for $nT \leq \tau \leq nt + T$,

$$X(nT + T) = e^{FT} X(nT) + \{ \int_{nT}^{nT+T} e^{F(nT+T-\tau)} d\tau \} GU(nT).$$

Let $\tau_2 = nT - T - \tau$, we have

$$X_{n+1} = e^{FT} X_n + \{ \int_{0}^{T} e^{F\tau} d\tau \} GU_n.$$ 

So we have

$$\Phi = e^{FT},$$
$$\Gamma = \{ \int_{0}^{T} e^{F\tau} d\tau \} G,$$
$$H \leftrightarrow H,$$
$$J \leftrightarrow J.$$

2 Controllability and Observability

2.1 Controller canonical form

$$\frac{Y(s)}{U(s)} = \frac{b(s)}{a(s)} = \frac{b_1 s^2 + b_2 s + b_3}{s^3 + a_1 s^2 + a_2 s + a_3}$$

$$Y(s) = b(s) \left[ \frac{U(s)}{a(s)} \right] = b(s) \zeta(s)$$

$$U(s) = a\zeta = \left( s^3 + a_1 s^2 + a_2 s + a_3 \right) \zeta(s)$$

$$u = \xi^{[3]} + a_1 \dot{\xi} + a_2 \ddot{\xi} + a_3 \xi$$

$$y = b_1 \dot{\xi} + b_2 \ddot{\xi} + b_3 \xi$$

![Controller canonical form diagram]

**Figure 13:**
\[
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2 \\
\dot{x}_3
\end{bmatrix} =
\begin{bmatrix}
-a_1 & -a_2 & -a_3 \\
1 & 0 & 0 \\
0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix}
+ \begin{bmatrix}
1 \\
0 \\
0
\end{bmatrix} u
\]

\[C' = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix}\]
\[D = 0\]

2.2 Duality of the controller and observer canonical forms

\[
A_c = A_o^T
\]
\[
B_c = C_o^T
\]

\[
G(s) = C_c(sI - A_c)^{-1}B_c
\]
\[
G(s) = [C_c(sI - A_c)^{-1}B_c]^T
\]
\[
B_c^T(sI - A_c^T)^{-1}C_c^T = C_o(sI - A_o)^{-1}B_o
\]

**Question:** Can a canonical form be transformed to another arbitrary form?

\[\text{A, B, C} \quad \rightarrow \quad \text{A}_c, \text{B}_c, \text{C}_c \quad \rightarrow \quad \text{A}_o, \text{B}_o, \text{C}_o\]

**Figure 14:**

2.3 Transformation of state space forms

Given F, G, H, J, find transformation matrix T, such that

\[
A = \begin{bmatrix}
-a_1 & -a_2 & -a_3 \\
1 & 0 & 0 \\
0 & 1 & 0
\end{bmatrix}
\]
\[
B = \begin{bmatrix}
1 \\
0 \\
0
\end{bmatrix}
\]

Let

\[
T^{-1} = \begin{bmatrix} t_1 \\
t_2 \\
t_3 \end{bmatrix}
\]
\[ A = T^{-1}FT \]
\[ AT^{-1} = T^{-1}F \]
\[ B = T^{-1}G \]

\[
\begin{bmatrix}
-a_1 & -a_2 & -a_3 \\
1 & 0 & 0 \\
0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
t_1 \\
t_2 \\
t_3
\end{bmatrix}
= 
\begin{bmatrix}
t_1 \\
t_2 \\
t_3
\end{bmatrix}
\begin{bmatrix}
F \\
F \\
F
\end{bmatrix}
\]

\[
\begin{bmatrix}
t_1 \\
t_2
\end{bmatrix}
= 
\begin{bmatrix}
t_1F \\
t_2F \\
t_3F
\end{bmatrix}
\]

\[ t_1 = t_2F \]
\[ t_2 = t_3F \]

need to determine \( t_3 \)

\[ B = T^{-1}G \]

\[
\begin{bmatrix}
1 \\
0 \\
0
\end{bmatrix}
= 
\begin{bmatrix}
t_1G \\
t_2G \\
t_3G
\end{bmatrix}
\]

\[ t_1G = 1 = t_3F^2G \]
\[ t_2G = 0 = t_3FG \]
\[ t_3G = 0 \]

\[ t_3 \begin{bmatrix} G & FG & F^2G \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} = e_{3}^{T} \]

basis function

In general form,

\[ t_n \begin{bmatrix} G & FG & F^2G & \ldots & F^{n-1}G \end{bmatrix} = e_{n}^{T} \]

controllability matrix \( C(F,G) \)

If \( C \) is full rank, we can solve for \( t_n \)

\[ T^{-1} = \begin{bmatrix} t_n F^{n-1} \\
\end{bmatrix} \]

where

\[ t_n = e_{n}^{T} C^{-1} \]
**Definition:** A realization \( \{F, G\} \) can be converted to controller canonical form iff \( \mathcal{C}^{-1}(F, G) \) exists \iff system is controllable.

**Corollary:** \( \mathcal{C} \) is full rank \iff system is controllable

**Exercise:** Show existence of \( \mathcal{C}^{-1} \) is unaffected by an invertible transformation \( T, x = Tz \).

**Bonus question:** What’s the condition for a system observable?

Given \( F, G, H, J \), solve for \( A_o, B_o, C_o, D_o \). By duality

\[
A_c = A_o^T \\
B_c = C_o^T
\]

iff \( \mathcal{C}^{-1}(F^T, H^T) \) exists,

\[
\begin{bmatrix}
H^T & F^T H^T & \cdots & F^{n-1} H^T
\end{bmatrix}^T = \begin{bmatrix}
H \\
H F \\
H F^{n-1}
\end{bmatrix} = \mathcal{O}(F, H)
\]

**Observability matrix**

**Exercise**

![Diagram](image)

**Figure 15:**

\[
F = \begin{bmatrix}
-1 & 0 \\
1 & 1
\end{bmatrix}
\]

\[
G = \begin{bmatrix}
-2 \\
1
\end{bmatrix}
\]

\[
H = \begin{bmatrix}
0 & 1
\end{bmatrix}
\]

\[
\mathcal{C} = \begin{bmatrix}
-2 & 2 \\
1 & -1
\end{bmatrix}
\]

\[det(\mathcal{C}) = 2 - 2 = 0 \implies \text{not controllable}\]

\[
\mathcal{O} = \begin{bmatrix}
0 & 1 \\
1 & 1
\end{bmatrix}
\]
\[ \text{det}(\mathcal{O}) = -1 \implies \text{observable} \]

**Exercise:**

\[
\begin{array}{ccc}
& & 1 \\
& \downarrow u & \downarrow & \downarrow & \downarrow \frac{s-1}{s+1} & \downarrow y \\
& & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
& & & & \downarrow & \downarrow & \downarrow & \downarrow \\
& & & & & & & \\
\end{array}
\]

**Answer:** You will get the reverse conclusion to the above exercise.

**Conclusion:** A pole zero cancellation implies the system is either uncontrollable or unobservable, or both.

**Exercise:** Show controllability canonical form has \( \mathcal{C} = I \implies \text{controllable} \)

**Comments**

- Controllability does not imply observability, observability does not imply controllability.
- Given a transfer function,
  \[ G(s) = \frac{b(s)}{a(s)} \]
  We can choose any canonical form \( w \) like. For example, either controller form or observer form. But once chosen, we cannot necessarily transform between the two.

- Redundant states
  \[
  \begin{align*}
  \dot{x} &= Fx + Gu \\
y &= Hx
  \end{align*}
  \]
  Let \( x_{n+1} = Lx \)
  \[
  \mathcal{T} = \begin{bmatrix} x \\ x_{n+1} \end{bmatrix} \implies \text{unobservability}
  \]

**Definition:** A system \((F, G)\) is "controllable" if \( \exists \) a control signal \( u \) to take the states \( x \), from arbitrary initial conditions to some desired final state \( x_F \), in finite time, \( \iff C^{-1} \exists \text{ exists.} \)

**Proof:** Find \( u \) to set initial conditions
Suppose \( x(0^-) = 0 \) and \( u(t) = \delta(t) \)

\[
X(s) = (sI - A)^{-1}bU(s) = (sI - A)^{-1}b
\]

\[
I.V.T. = x(0^+) = \lim_{s \to \infty} sX(s) = \lim_{s \to \infty} s(sI - A)^{-1}b = \lim_{s \to \infty} (I - \frac{A}{s})^{-1}b = b
\]
Thus an impulse brings the state to \( b \) (from 0)
Now suppose \( u(t) = \delta^{(k)}(t) \)

\[
U(s) = s^k 
\]

\[
X(s) = \frac{1}{s} (I - \frac{A}{s})^{-1} b s^k = \frac{1}{s} (I + A s + \frac{A^2}{s^2} + \ldots) b s^k
\]

\[
= b s^{k-1} + A b s^{k-2} + \ldots + A^{k-1} b + \frac{A^k b}{s} + \frac{A^{k+1} b}{s^2} + \ldots
\]

\[
x(t) = b \delta^{(k-1)} + A b \delta^{(k-2)} + A^{k-1} b \delta \text{ due to impulse at } t=0;
\]

\[
x(0^+) = \lim_{s \to \infty} s \left\{ \frac{A^k b}{s} + \frac{A^{k+1} b}{s^2} + \ldots \right\} = A^k b
\]

if \( u(t) = \delta^{(k)}(t) \Rightarrow x(0^+) = A^k b \)

Now let

\[
u(t) = g_1 \delta(t) + g_2 \delta(t) + \ldots + g_n \delta^{(n-1)}(t), \text{ by linearity,}
\]

\[
x(0^+) = g_1 b + g_2 Ab + \ldots + g_n A^{n-1} b = \begin{bmatrix} b & Ab & A^2 b & \ldots & A^{n-1} b \end{bmatrix} \begin{bmatrix} g_1 \\ \vdots \\ g_n \end{bmatrix}
\]

\[
x_{\text{desired}} = x(0^+)
\]

\[
\begin{bmatrix} g_1 \\ \vdots \\ g_n \end{bmatrix} = C^{-1} x(0^+) \text{ if } C^{-1} \text{ exists}
\]

and we can solve for \( g_i \) to take \( x(0^+) \) to any desired vector

**Conclusion:** We have proven that if a system is controllable, we can instantly move the states from any given state to any other state, using impulsive inputs. *(only if proof is harder)*

### 2.4 Discrete controllability or reachability

A discrete system is reachable iff \( \exists \) a control signal \( u \) that can take the states from arbitrary initial conditions to some desired state \( x_F \), in finite time steps.

\[
x_k = A^k x_0 + \sum_{i=0}^{k} A^{i-1} b u_{k-1}
\]

\[
x_N = A^N x_0 + \begin{bmatrix} b & Ab & A^2 b & \ldots & A^{N-1} b \end{bmatrix} \begin{bmatrix} u(N-1) \\ u(N-2) \\ \vdots \\ u(0) \end{bmatrix} = C^{-1}(x_d - A^N x(0))
\]

106
2.4.1 Controllability to the origin for discrete systems

- $|C| \neq 0$ is a sufficient condition
- Suppose $u(n)=0$, then $x(n) = 0 = A^n x(0)$ if $A$ is a nilpotent matrix (all the eigenvalues equal to zero).
- In general, if $A^n x_0 \in \mathbb{R}[C]$, range of $C$

**Advanced question:** If a continuous system $(A, B)$ is controllable, is the discrete system $(\phi, \Gamma)$ also controllable?

**Answer:** Not necessarily!

![Sampling slower](image1)

Figure 16:

2.5 Observability

A system $(F, H)$ is observable $\iff$ for any $x(0)$, we can deduce $x(0)$ by observation of $y(t)$ and $u(t)$ in finite time $\iff O(F, H)^{-1}$ exists

For state space equation

\[
\begin{align*}
\dot{x} &= A x + B u \\
y &= C x \\
\dot{y} &= C \dot{x} = CA x + CB u \\
\ddot{y} &= C \ddot{x} = CA(A x + Bu) + CB \ddot{u} \\
&\vdots \\
y^{(k)} &= CA^k x + CA^{k-1} Bu + \ldots + CB u^{k-1} \\
\begin{bmatrix}
y(0) \\
\dot{y}(0) \\
\ddot{y}(0)
\end{bmatrix} &=
\begin{bmatrix}
C \\
CA \\
CA^2
\end{bmatrix}
\begin{bmatrix}
x(0) \\
\end{bmatrix}
+ 
\begin{bmatrix}
0 & 0 & 0 \\
CB & 0 & 0 \\
CAB & CB & 0
\end{bmatrix}
\begin{bmatrix}
u(0) \\
\dot{u}(0) \\
\ddot{u}(0)
\end{bmatrix}
\end{align*}
\]
\[ x(0) = O^{-1}(A, C) \begin{bmatrix} y(0) \\ \vdots \\ y^{n-1} \end{bmatrix} + T \begin{bmatrix} u(0) \\ \vdots \\ u^{n-1}(0) \end{bmatrix} \]

if \( O^{-1} \) exists, we can deduce \( x(0) \)

**Exercise**: solve for discrete system

\[ x(k) = O^{-1} \begin{bmatrix} y(k) \\ \vdots \\ y(k + n - 1) \end{bmatrix} + T \begin{bmatrix} u(k) \\ \vdots \\ u(k + n - 1) \end{bmatrix} \]

### 2.6 Things we won’t prove

- If \( A, B, C \) is observable and controllable, then it is minimal, i.e., no other realization has a smaller dimension. This implies \( a(s) \) and \( b(s) \) are co-prime.

- The unobservable space

\[ \mathcal{N}(\mathcal{O}) = \{ x \in \mathbb{R}^n | \mathcal{O} x = 0 \} \] is the null space of \( \mathcal{O} \)

solutions of \( \dot{x} = Ax \) which start in unobservable space cannot be seen by looking at output (i.e., we stay in unobservable space).

If \( x(0) \in \mathcal{N}(\mathcal{O}) \), then \( y = C x = 0 \) for all \( t \).

- The controllable space,

\[ \mathcal{R}(\mathcal{C}) = \{ x = C v | v \in \mathbb{R}^n \} \] is the range space of \( \mathcal{C} \)

If \( x(0) \in \mathcal{R}(\mathcal{C}) \) and \( \dot{x} = Ax + Bu \), then \( x(t) \in \mathcal{R}(\mathcal{C}) \forall t \) for any \( u(t) \) the control can only move the state in \( \mathcal{R}(\mathcal{C}) \)

- PBH (Popov-Belevitch-Hautus) Rank Tests

  1. A system \( \{ A, B \} \) is controllable iff \( \text{rank } [sI - A, B] = N \) for all \( s \).

  2. A system \( \{ A, C \} \) is observable iff

\[ \text{rank } \begin{bmatrix} C \\ sI - A \end{bmatrix} = N \] for all \( s \).

- These tests tell us which modes are uncontrollable or unobservable.

- Grammian Tests

\[ P(T, t) = \int_t^T \phi(T, \lambda)B(\lambda)B'(\lambda)\phi'(T, \lambda)d\lambda, \] where \( \phi \) is the state transition matrix.

A system is controllable iff \( P(T, t) \) is non-singular for some \( T > t \). For time-invariant system,

\[ P(T) = \int_0^T e^{-At}B B^T e^{-A^T} dt \]
3 Feedback control

\[
\begin{align*}
    u &= -k x \\
    \dot{x} &= Fx + Gu = (F - Gk)x
\end{align*}
\]

Poles are the eigenvalues of \((F - Gk)\).

Suppose we are in controller canonical form

\[
F - Gk = \begin{bmatrix}
-a_1 - k_1 & -a_2 - k_2 & -a_3 - k_3 \\
1 & 0 & 0 \\
0 & 1 & 0
\end{bmatrix}
\]

\[
det(sI - F + Gk) = s^3 + (a_1 + k_1)s_2 + (a_2 + k_2)s + a_3 + k_3
\]

We can solve for \(k\) so that the closed loop has arbitrary characteristic eigenvalues. This is called "pole placement problem". Simply select \(a_1, a_2, a_3\), which are the coefficients of the desired characteristic equation,

\[
s^3 + a_1 s^2 + a_2 s + a_3 = \alpha_c(s)
\]

Since we know

\[
\alpha_c(s) = det(sI - F + Gk)
\]

we can solve for \(k\) by matching the coefficients of the two equations. Suppose now that \(\{F, G\}\) are not in controller canonical form, find transformation \(T\), such that \(x = Tz\) and

\[
\dot{z} = Az + Bu \text{ are in controller canonical form}
\]

Then find \(k_z\) so that \(det(sI - A + Bk_z) = \alpha(s)\). Feedback \(u = -k_z z = -k_z T^{-1} x\), thus, we get

\[
k_z = k_x T^{-1}
\]

**Conclusion:** \(T^{-1}\) can be found \(\iff\) \(C^{-1}\) exists, which means system is controllable. Putting all the steps together,

\[
(F, G) \xrightarrow{T} (A, B) \xrightarrow{\text{match coefficients}} k_z \xrightarrow{T^{-1}} k_x
\]

Closed form - Ackerman's formula

\[
k_x = c^T_n C^{-1} (F, G) \alpha_c(F)
\]

\[
\alpha_c(F) = F^n + \alpha_1 F^{n-1} + \ldots + \alpha_n I
\]
In MATLAB

\[ k = \text{Ack}er(F, G, P) \] is vector of the desired poles,

This is not good numerically, since we need to calculate \( C^{-1} \).

Instead, we use

\[ k = \text{place}(F, G, P) \]

This is based on a set of orthogonal transformations.

**Summary:**

\[
\begin{align*}
  u &= -Kx \\
  \dot{x} &= Fx + Gu = (F - GK)x
\end{align*}
\]

Poles are \( \text{eig}(F - GK) = \alpha_c(s) \).

Can choose arbitrary \( \alpha_c(s) \) given \( F, G \) controllable.

\[ \text{K} = \text{place}(F, G, P) \]

where \( P \) is the vector of poles.

### 3.1 Zeros

\[
\begin{align*}
  \dot{x} &= Fx + Gu \\
  \dot{z} &= (F - GK)x + G\tilde{N}r \\
  y &= Hx
\end{align*}
\]

Zeros from \( r \rightarrow y \) are given by

\[
\det \begin{bmatrix}
  sI - F + GK & -G\tilde{N} \\
  H & 0
\end{bmatrix}
\]

Multiply column two by \( K/\tilde{N} \) and add to first column.

\[
\det \begin{bmatrix}
  sI - F & -G\tilde{N} \\
  H & 0
\end{bmatrix}
\]

→ Same as before
→ Zeros unaffected by State Feedback (same conclusion as in classical control)
Example:

\[ G(s) = \frac{(s+2)(s-3)}{s^3 + s - 10} \]
\[ C(sI - A + GK)^{-1}B = \frac{1}{s+1} \]

→ unobservable

Does it affect controllability? No

\( F - GK \to \) still in Controller Canonical Form.

Example

\[
\begin{bmatrix}
1 & 1 \\
0 & 2
\end{bmatrix}
\begin{bmatrix}
\dot{x} \\
x
\end{bmatrix}
+ 
\begin{bmatrix}
1 \\
0
\end{bmatrix}
\begin{bmatrix}
u
\end{bmatrix}
\]

\[
det(sI - A + BK) = \begin{vmatrix}
 s - 1 + k_1 & -1 + k_2 \\
 0 & s - 2
\end{vmatrix} = (s - 1 + k_1) (s - 2)
\]

\( \lambda = +2 \) check with PBH

Example

\[
X
\]

\[
\begin{align*}
\dot{x} &= -x + u \\
u &= -Kx \\
\dot{x} &= -(1 + K)x \\
x(t) &= e^{-(1+K)t}, \quad x(0) = 1 \\
u &= -ke^{-(1+K)t}, \quad x(0) = 1
\end{align*}
\]
• As $K \to \infty$ (large gain), $u$ looks more and more like $\delta(t)$ which instantaneously moves $x$ to 0. (recall proof for controllability)

3.2 Reference Input

\[
\begin{align*}
\dot{x} &= Fx + Gu \\
y &= Hx \\
r &= 	ext{ref}
\end{align*}
\]

1. Assume $r \to \text{constant as } t \to \infty$ (e.g. step)

2. Want $y \to r$ as $t \to \infty$

Let,

\[
\begin{align*}
u &= -K(x - x_{ss}) + u_{ss} \\
x_{ss} &= Nx_r \\
u_{ss} &= Nu_r
\end{align*}
\]
\[
\overline{N} = (KN_x + N_u)
\]

At steady state,

\[
\begin{align*}
x & \rightarrow x_{ss} \\
\dot{x} & \rightarrow 0 \\
u & \rightarrow u_{ss}
\end{align*}
\]

\[
\begin{align*}
0 &= Fx_{ss} + Gu_{ss} \\
y_{ss} &= Hx_{ss} = r
\end{align*}
\]

\[
FN_x r + GN_u r = 0 \\
HN_x r = r
\]

Divide by \( r \).

\[
\begin{bmatrix}
F & G \\
H & 0
\end{bmatrix}
\begin{bmatrix}
N_x \\
N_u
\end{bmatrix} =
\begin{bmatrix}
0 \\
1
\end{bmatrix}
\]

\[
\begin{bmatrix}
N_x \\
N_u
\end{bmatrix} =
\begin{bmatrix}
F & G \\
H & 0
\end{bmatrix}^{-1}
\begin{bmatrix}
0 \\
1
\end{bmatrix}
\]

**Example:** Two mass system

![Diagram of two mass system](image)

We know, that in steady state

\[
\begin{bmatrix}
x_1 \\
\dot{x}_1 \\
x_2 \\
\dot{x}_2
\end{bmatrix} \rightarrow
\begin{bmatrix}
1 \\
0 \\
1 \\
0
\end{bmatrix}
\]

Hence, 

\[
N_x = \begin{bmatrix}
1 \\
0 \\
1 \\
0
\end{bmatrix}, \quad and \quad N_u = 0
\]
For SISO systems

\[
\frac{\dot{y}}{\dot{r}} = H(sI - F + GK)^{-1}GN
\]

\[
G(s) |_{s=0} = 1 \quad \Rightarrow \quad N = \frac{1}{H(-F + GK)^{-1}G}
\]

3.3 Selection of Pole Locations

Suppose F, G are controllable, where should the poles be placed? There are three common methods for placing the poles.

3.3.1 Method 1: Dominant second-order poles

\[
\frac{\dot{y}}{\dot{r}} = \frac{\omega_n^2}{s^2 + 2\xi\omega_ns + \omega_n^2}
\]

Example

Choose 2nd order dominant pole locations:

\[
\begin{align*}
\omega_n &= 1 \\
\xi &= 0.5 \\
\alpha_c &= s^2 + s + 1 \\
p_d &= \frac{-1}{2} \pm \frac{\sqrt{3}}{2}i
\end{align*}
\]
Take other 2 poles to be
\[ \omega_n = 4, \quad \xi = 0.5, \quad (4 \times p_d) \]
\[ p = \begin{pmatrix}
-1/2 + \sqrt{3}i/2 \\
-1/2 - \sqrt{3}i/2 \\
-2 + 2\sqrt{3}i \\
-2 - 2\sqrt{3}i
\end{pmatrix} \]

MATLAB:

\[ K = \text{place}(F, G, P) \]
\[ K = \begin{bmatrix}
19.23 & 4.96 & -1.6504 & 16.321
\end{bmatrix} \]

effort is the price of moving poles.

### 3.3.2 Step Response Using State Feedback

\[ \dot{x} = Fx + Gu \\
u = -Kx + N r \\
y = Hx \\
\dot{y} = (F - GK)x + NGr \]

where \( N \) is a scale factor.

MATLAB:

\[ y = \text{step}(F - GK, NG, H, 0, 1, t) \]

Example

% MATLAB .m file for Two Mass System. State Feedback, E. Wan

% Parameters
M = 1
m = .1
k = 0.091
b = 0.0036

% State-Space Description
F = [0 1 0 0; -k/b M k/b M ; 0 0 1; k/m b/m -k/m -b/m]
G = [0 1/M 0 0]'
H = [0 0 1 0]
J = 0

% Desired pole locations
Pd = roots([1 1 1])
P = [Pd 4*Pd]
K = place(F, G, P)

% Closed loop pole locations
pc = eig(F-G*K)

% Reference Design
Nx = [1 0 1 0]';
Nb = K*Nx

% Step Response
t = 0:.004:15;
subplot(211)
[Y, X] = step(F-G*K, G*Nb, H, 0, 1, t);
plot(t, H*X,'- -');
subplot(212)
plot(t, -K*X + Nb,' - -');
title('2-Mass control ref. step')

% Standard Root Locus vs Plant Gain
%axis('square');
rlocus(F, G, K, 0)
%title('Plant/State-Feedback Root Locus')
rcl = rlocus(F, G, K, 0, 1)
%hold
%plot(rcl,' * ')
%axis([-4 4 -4 4]);

% Bode plot of loop transfer function
%margin(F, G, K, 0)
3.3.3 Gain/Phase Margin

MATLAB:

\[ \text{bode}(F, G, K, 0) \]

\[ \text{margin}(F, G, K, 0) \]
Root Locus

MATLAB:

\[ rlocus(F, G, K, 0) \]

\[ \text{Open loop} - (F, G, H) \]

\[ \text{Closed loop} - (F - GK, NG, H) \]
Open loop for stability check

\[(F, G, K)\]

\[T.F. = K(sI - F)^{-1}G\]

### 3.3.4 Method 2: Prototype Design

Select prototype responses with desirable dynamics. There are several sets of these:
- **Butterworth**
  \[
  \frac{1}{1 + (\omega/\omega_c)^{2n}}
  \]
- **Bessel**
- **ITAE**: Minimizes the Integral of the Time multiplied by the Absolute value of Error,
  \[
  J = \int_0^t t |e| \, dt
  \]

(35 years ago - by twiddling and trial and error)

All these methods set pole locations and ignore the influence of plant zeros.

### 3.3.5 Method 3: Optimal Control and Symmetric Root Locus

Suppose we want to minimize

\[
J = \int_0^\infty e^2 \, dt
\]

Optimal Control Law for this is a bunch of \(\delta\)'s and its derivatives. But the \(\delta\)'s are not practical.

Linear Quadratic Constraints

\[
J = \int_0^{t_f} \left[ x^T(t)Qx(t) + u^T(t)Ru(t) \right] \, dt + x^T(t_f)Sx(t_f)
\]

where \(Q\), \(R\) and \(S\) are positive definite weighting matrices. \(x \to 0\) is equivalent to \(e \to 0\).

It turns out that

\[
u(t) = -K(t)x(t)
\]

which is time varying state control.
The linear feedback $K(t)$ may be solved through Dynamic Programming (next chapter). A general “optimal” control has the form:

$$J = \int_{0}^{t_f} [L(x, u, t)] dt + L_2(x(t_f), u(t_f))$$

which may be difficult to solve. For the special case

$$J = \int_{0}^{\infty} \left[ x^T Q x + u^T R u \right] dt$$

we have the Linear Quadratic Regulator.

and

$$u = -kx \implies \text{fixed feedback control.}$$

**SRL**

$$Q = \rho^2 H^T H$$

$$R = I$$

$$\implies J = \int_{0}^{\infty} \left[ \rho^2 y^2 + u^2 \right] dt$$

Solution of $u = -Kx$ leads to the optimal value of $K$, which places the closed loop poles at the stable roots of the equation,

$$1 + \rho^2 G^T(-s)G(s) = 0 \quad \text{where} \quad G(s) = \frac{Y(s)}{U(s)} \quad \text{(open loop)}$$

This is called the Symmetric Root Locus. The locus in the LHP has a mirror image in the RHP. The locus is symmetric with respect to the imaginary axis.

- $\rho = 0$: expensive control, since control effort has to be minimized
- $\rho = \infty$: cheap control, since $y^2$ has to be minimized, hence actuators may approach delta functions.

**Examples**

$$G(s) = \frac{1}{s^2}$$

$$G(-s) = \frac{1}{(-s)^2} = \frac{1}{s^2}$$
Example: Two Mass system

MATLAB:

\[ K = LQR(F, G, Q, R) \]

where

\[ Q = p^2 H^T H \]
\[ R = I \]

% MATLAB .m file for Two Mass System. LQR State Feedback, E. Wan

% Parameters
M = 1
m = .1
k = 0.091
b = 0.0036

% State-Space Description
F = [0 1 0 0; -k/M -b/M k/M b/M; 0 0 0 1; k/m b/m -k/m -b/m]
G = [0 1/M 0 0]
H = [0 0 1 0]
J = 0

% LQR Control Law for p^2 = 4
K = lqr(F,G,H'*H*4,1)

% Closed loop pole locations
pc = eig(F-G*K)
% Check natural frequency
a_pc = abs(eig(F-G*K))

% Symmetric Root Locus for LQR design
% subplot(211)
% A = [F zeros(4); -H'*H -F'];
% B = [G ;0;0;0;0];
% C = [0 0 0 0 G'];
% rlocus(A,B,C,0)
% title('SRL for LQR')
% axis('square');

% subplot(212)
% rlocus(A,B,C,0)
% hold
% plot(pc, '*')
% axis([-3 3 -3 3]);
% axis('square');
% The poles pc are on the stable locus for p^2 = 4

% Standard Root Locus vs Plant Gain
% rlocus(-num,den)
% rlocus(F,G,K,0)
% title('Plant/State-Feedback Root Locus')
% rcl = rlocus(F,G,K,0,1)
% hold
% plot(rcl, '*')
% axis('square');

% Bode plot of loop transfer function
margin(F,G,K,0)

% Reference Design
Nx = [1 0 1 0]';
Nb = K*Nx

% Step Response
% t = 0:.004:15;
% [Y,X] = step(F-G*K,G*Nb,H,0,1,t);
% subplot(211);
% step(F-G*K,G*Nb,H,0,1,t);
% title('2-Mass control ref. step')
% subplot(212)
% plot(t,-K*X+Nb)
% title('2-Mass Control effort ')

122
Plant/State–Feedback Root Locus

\[ G_m = -30.03 \text{ dB, (} w = 1.02 \text{)} \quad P_m = 60.1 \text{ deg, (} w = 2.512 \text{)} \]
Example

\[ G(s) = \frac{1}{(s + 1)(s - 1 \pm j)} \]

Choose these poles to stabilize and minimize the control effort.
Fact about LQR: Nyquist plot does not penetrate disk at -1.

\[ P.M > 60 \]

\[ \Rightarrow PM \geq 60° \]

Example

\[ H(s) = \frac{s}{s^2 - 1} \]

But,

\[ 1 + \rho^2 G^T(-s)G(s) = 1 + \rho^2 \frac{s}{s^2 - 1} \frac{(-s)}{s^2 - 1} \]

\[ = 1 - \rho^2 \frac{s^2}{(s^2 - 1)^2} \]
Instead we should plot the Zero Degree Root Locus

![Root Locus Diagram]

Rule: To determine the type of Root Locus \((0^\circ \text{ or } 180^\circ)\) choose the one that does not cross the Imaginary axis.

3.4 Discrete Systems

\[
1 + \rho^2 G^T (z^{-1}) G(z) = 0
\]

\[
z = e^{sT} \rightarrow e^{-sT} = \frac{1}{z}
\]

\[
J = \sum_{k=0}^{\infty} u^2(k) + \rho^2 y^2(k)
\]

Example

\[
\frac{1}{s^2} \rightarrow \frac{z + 1}{(z - 1)^2}
\]

from zero at infinity

![Discrete System Example Diagram]

\[
G(z^{-1}) = \frac{z^{-1} + 1}{(z^{-1} - 1)^2} = \frac{(z + 1)z}{(z - 1)^2}
\]

Example Two mass system.
% MATLAB .m file for Two Mass System. Discrete control
% Eric A. Wan

% Parameters
M = 1;
m = .1;
k = 0.091;
b = 0.0036;

% State-Space Description
F = [0 1 0 0; -k/M -b/M k/M b/M ;0 0 0 1; k/m b/m -k/m -b/m];
G = [0 1/M 0 0]';
H = [0 0 1 0];
J = 0;

% LQR Control Law for p^2 = 4
K = lqr(F,G,H'*H*4,1);

% Closed loop pole locations
pc = eig(F-G*K);

% Reference Design
Nx = [1 0 1 0]';
Nb = K*Nx;

% Step Response
%t = 0:.004:20;
[Y,X] = step(F-G*K,G*Nb,H,0,1,t);
%subplot(211);
%u = -K*X';
%step(F-G*K,G*Nb,H,0,1,t);
%title('2-Mass reference step')
%plot(t,u+Nb)
%title('2-Mass control effort ')

%ZOH discrete equivalent
T = 1/2;
[PHI,GAM] = c2d(F,G,T);
 pz = exp(pc*T);
Kz = acker(PHI,GAM,pz);
Nxz = Nx;
Nbz = Kz*Nx;

% Step Response
%axis([0 20 0 1.5])
%t = 0:.004:20;
%[Y,X] = step(F-G*K,G*Nb,H,0,1,t);
% subplot(211);
% step(F-G*K,G*Nb,H,0,1,t);
% hold
% title('Discrete: 2-Mass reference step, T = 1/2 sec')
% [Y,X] = dstep(PHI-GAM*Kz,GAM*Nbz,H,0,1,20/T);
% td = 0:T:20-T;
% plot(td,Y,'*')
% hold
% dstep(PHI-GAM*Kz,GAM*Nbz,-Kz,Nbz,1,20/T)
% title('ZOH control effort')

% pole locations in s and z plane
% axis('square')
% axis([-2 2 -2 2]);
% subplot(211)
% plot(pc,'*')
% title('s-plane pole locations')
% grid
% axis([-1 1 -1 1]);
% pz = exp(pc/4);
% plot(pz,'*')
% title('z-plane pole locations')
% grid
% T = 1/4:.01:5;
% pz = exp(pz*T);
% plot(pz,'.')

% Discrete Step Response with intersample response
% axis([0 20 0 1.5])
% subplot(211);
% [Y,X] = dstep(PHI-GAM*Kz,GAM*Nbz,H,0,1,20/T);
% td = 0:T:20-T;
% plot(td,Y,'*')
% title('Dead beat control')
% [u,X] = dstep(PHI-GAM*Kz,GAM*Nbz,-Kz,Nbz,1,20/T);
% uc = ones(20,1)*u';
% uc = uc(:,);
% hold
% [PHI2,GAM2] = c2d(F,G,T/20);
% [y2,x2] = dlinsim(PHI2,GAM2,H,0,uc);
% t2d = linspace(0,20,20*20/T);
% plot(t2d,y2)
% hold
% dstep(PHI-GAM*Kz,GAM*Nbz,-Kz,Nbz,1,20/T)
% title('control effort')

% Standard Root Locus vs Plant Gain
% axis([-1 1 -1 1]);
% axis('square');
% rlocus(PHI,GAM,Kz,0)
%title('Plant/State-Feedback Root Locus, T = 1/2')
%rc1 = rlocus(PHI,GAM,Kz,0,1)
%zgrid
%hold
%plot(rc1,'*')

% Standard Root Locus vs Plant Gain
%axis([-3 3 -3 3]);
%axis('square');
%rlocus(F,G,K,0)
%title('Plant/State-Feedback Root Locus')
%rc1 = rlocus(F,G,K,0,1)
%sgrid
%hold
%plot(rc1,'*')

% Bode plot of loop transfer function
%w = logspace(-1,1,200);
%bode(F,G,K,0,1,w);
%[mag,phase] = bode(F,G,K,0,1,w);
%title('Continuous Bode Plot')
%[gm,pm,wg,wc] = margin(mag,phase,w)

% Bode plot of loop transfer function
rad = logspace(-2,log10(pi),300);
dbode(PHI,GAM,Kz,0,T,1,rad/T);
[mag,phase] = dbode(PHI,GAM,Kz,0,T,1,rad/T);
title('Discrete Bode Plot, T = 1/2')
[gmd,pmd,wg,wc] = margin(mag,phase,rad/T)
LQR step response, $T = \frac{1}{2}$

control effort

LQR step response, $T = 1$

control effort
LQR step response, $T = 2$

Deadbeat step response, $T = 1/2$
4 Estimator and Compensator Design

4.1 Estimators

\[ x(t) = \mathcal{O}^{-1} \left\{ \begin{bmatrix} \dot{y} \\ \dot{\hat{y}} \end{bmatrix} - \mathcal{T} \begin{bmatrix} u \\ \hat{u} \end{bmatrix} \right\} \]

Not very practical for real systems

\[ \dot{x} = Fx + Gu + G_1w \]
\[ y = Hx + v \]

where \( w \) is the process noise and \( v \) is the sensor noise. Want an estimate \( \hat{x} \) such that \( (x - \hat{x}) \) is "small"

\[ \text{if } \quad w = v = 0 \quad x - \hat{x} \rightarrow 0 \]

\[ \hat{x} = F\hat{x} + Gu \]

\[ \text{error } \quad x - \hat{x} = \tilde{x} \]
\[ \tilde{x} = \dot{x} - \dot{\hat{x}} = Fx + Gu + G_1w - (F\hat{x} + Gu) \]
\[ = F\tilde{x} + G_1w \]

(open loop dynamics - usually not good)

\[ \hat{y} = y - \hat{y} \]

\[ \dot{\hat{x}} = F\hat{x} + Gu + L(y - H\hat{x}) \]

How to choose \( L \)?
Pole placement

1. Luenberger (observer)
   - Dual of pole-placement for K

2. Kalman
   - Choose $L$ such that the estimates is Maximum Likelihood assuming Gaussian noise statistics. to minimize

Error Dynamics

The error dynamics are given by,

$$\dot{\tilde{x}} = F\tilde{x} + G_1 w + G_1 u + L(y - H\tilde{x})$$

$$\dot{\tilde{x}} = F\tilde{x} + G_1 w + LH\tilde{x} - Lv$$

$$\dot{\tilde{x}} = (F - LH)\tilde{x} + G_1 w - Lv$$

Characteristic Equation:

$$\det(sI - F + LH) = \alpha_e(s)$$

$$\det(sI - F + GK) = \alpha_e(s)$$

$$\alpha_e(s) = \alpha^T_e = \det(sI - F^T + H^T L^T)$$

Need $(F^T, H^T)$ controllable $\implies$ $(F, H)$ observable.

$$L^T = \text{place}(F^T, H^T, P_e)$$

4.1.1 Selection of Estimator poles $\alpha_e$

1. Dominant 2nd Order

2. Prototype

3. SRL

$$\dot{\tilde{x}} = (F - LH)\tilde{x} + G_1 w - Lv$$

- want $L$ large to compensate for $w$ (fast dynamics)
- $L$ provides lowpass filter from $v$, want $L$ small to do smoothing in $v$.

SRL - optimal tradeoff
\[ G_1(s) = \frac{y}{w} \]
\[ 1 + q^2 G_1^T (-s) G_1(s) = 0 \]
\[ q = \frac{\text{process noise energy}}{\text{sensor noise energy}} \]

\[ \Rightarrow \text{steady state solution to Kalman filter} \]

### 4.2 Compensators: Estimators plus Feedback Control

Try the control law \( u = -k \hat{x} \)

With this state estimator in feedback loop the system is as shown below.

What is the transfer function of this system?

The system is described by the following equations.

\[ \dot{x} = Fx + Gu + G_1 w \]
\[ y = Hx + v \]

Estimation

\[ \hat{x} = F\hat{x} + Gu + L(y - H\hat{x}) \]

Control

\[ u = -K\hat{x} \]

\[
\begin{pmatrix}
\dot{\hat{x}} \\
\dot{\hat{\hat{x}}}
\end{pmatrix} =
\begin{pmatrix}
F & \frac{-GK}{LH} \\
LH & F - GK - LH
\end{pmatrix}
\begin{pmatrix}
x \\
\hat{x}
\end{pmatrix} +
\begin{pmatrix}
G_1 \\
0
\end{pmatrix} w +
\begin{pmatrix}
0 \\
L
\end{pmatrix} v
\]

136
\[ \ddot{x} = x - \dot{x} \]

Apply linear transformation

\[
\begin{pmatrix} x \\ \dot{x} \end{pmatrix} = \begin{pmatrix} I & 0 \\ I & -I \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \end{pmatrix}
\]

With this transformation the state equations are:

\[
\begin{pmatrix} \dot{\ddot{x}} \\ \dot{\ddot{x}} \end{pmatrix} = \begin{pmatrix} F - GK & GK \\ 0 & F - LH \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \end{pmatrix} + \begin{pmatrix} G_1 \\ -L \end{pmatrix} w + \begin{pmatrix} 0 \\ G \end{pmatrix} v
\]

The characteristic equation is

\[
C.E = \det \begin{pmatrix} sI - F + GK & -GK \\ 0 & sI - F + LH \end{pmatrix} = \alpha
\]

no cross terms

\[ \alpha = \alpha_x(s)\alpha_e(s)!! \]

This is the Separation Theorem. In other words the closed loop poles of the system is equivalent to the compensator poles plus the estimator poles as if they were designed independently.

Now add reference input

\[ u = -K(\dot{x} - N_x r) + \dot{N}_x r = -K\dot{x} + \dot{N} r \]

What is the effect on \( \ddot{x} \) due to \( r \)?

\[ \ddot{x} = \dot{x} - \ddot{x} = (F - LH)\ddot{x} + G_1w - Lv \]

\( \ddot{x} \) is independent of \( r \), i.e. \( \ddot{x} \) is uncontrollable from \( r \). This implies that estimator poles do not appear in \( r \rightarrow x \) or \( r \rightarrow y \) transfer function.

\[
\frac{y}{r} = \frac{\ddot{N}\alpha_e(s)b(s)}{\alpha_e(s)\alpha_e(s)} = \frac{\ddot{N}b(s)}{\alpha_e(s)}
\]

This transfer function is exactly same as without estimator.
4.2.1 Equivalent feedback compensation

With a reference input the equivalent system is:

\[ D_2(s) \]

\[ u \]

\[ y \]

\[ D_1(s) \]

\[ r \rightarrow D_2(s) \]

\[ u \rightarrow P \]

\[ y \rightarrow \]

\[ D_1(s) \]

\[ r \rightarrow D_2(s) \]

\[ u \rightarrow \]

\[ y \rightarrow \]

\[ D_1(s) \]

\( D_2(s) \) is transfer function introduced by adding reference input.

We have

\[
\dot{x} = F\dot{x} + Gu + L(y - H\dot{x})
\]

\[
u = -K\dot{x} + \tilde{N}r
\]

\[ \Rightarrow \dot{x} = (F - GK - LH)\dot{x} + G\tilde{N}r + Ly \]

From the above equations (considering \( y \) as input and \( u \) as output for \( D_1(s) \)),

\[ D_1(s) = -K(sI - F + GK + LH)^{-1}L \]

(Considering \( r \) as an input and \( u \) as an output for \( D_2(s) \)),

\[ D_2(s) = -K(sI - F + GK + LH)^{-1}G\tilde{N} + \tilde{N} \]

4.2.2 Bode/root locus

The root locus and Bode plots are found by looking at open loop transfer functions. The reference is set to zero and feedback loop is broken as shown below.

\[ r = 0 \rightarrow D_2(s) \]

\[ u \rightarrow kG(s) \]

\[ y \rightarrow \]

\[ D_1(s) \]

\[ r = 0 \rightarrow D_2(s) \]

\[ u \rightarrow \]

\[ y \rightarrow \]

\[ D_1(s) \]

138
\[
D_i(s) = K(sI - F + GK + LH)^{-1}L
\]

\[
G(s) = H(sI - F)^{-1}G
\]

i.e.
\[
\begin{pmatrix}
\dot{x} \\
\dot{\hat{x}}
\end{pmatrix} = \begin{pmatrix}
F & 0 \\
LH & F - GK - LH
\end{pmatrix} \begin{pmatrix}
x \\
\dot{x}
\end{pmatrix} + \begin{pmatrix}
G \\
0
\end{pmatrix} u_{\text{in}}
\]

\[
u_{\text{out}} = \begin{pmatrix}
0 & -K
\end{pmatrix} \begin{pmatrix}
x \\
\dot{x}
\end{pmatrix}
\]

Rootlocus can be obtained by rlocus(A,B,C,0) and Bode plot by bode(A,B,C,0).

Estimator and controller pole locations are given by rlocus(A,B,C,0,1).

- As an exercise, find the state-space representation for input \( (r, w) \) and output \( (u, y) \).

---

%%% MATLAB .m file for Two Mass System. Feedback Control / Estimator

```matlab
% Parameters
M = 1
m = .1
k = 0.091
b = 0.0036

% State-Space Description
F = [0 1 0 0; -k/M -b/M k/M b/M ; 0 0 0 1; k/m b/m -k/m -b/m]
G = [0 1/M 0 0]
H = [0 0 1 0]
J = 0

% LQR Control Law for \( p^2 = 4 \)
K = lqr(F,G,H'*H*4,1)

K =

2.8730  2.4190  -0.8730  1.7076

% Closed loop pole locations
pc = eig(F-G*K)

pc =

-0.3480 + 1.2817i
-0.3480 - 1.2817i
```

---

139
\[-0.8813 + 0.5051i\]
\[-0.8813 - 0.5051i\]

```matlab
% Symmetric Root Locus for LQR design
axis([-3 3 -3 3]);
axis('square');
A = [F zeros(4); -H*H -F];
B = [G ;0;0;0;0];
C = [0 0 0 0 G'];
rlocus(A,B,C,0)
hold
%plot(pc,'*')
title('SRL for LQR')
% The poles pc are on the stable locus for p^2 = 4

% Standard Root Locus vs Plant Gain
axis([-3 3 -3 3]);
axis('square');
rlocus(-num,den)
rlocus(F,G,K,0)
title('Plant/State-Feedback Root Locus')
rcl = rlocus(F,G,K,0,1)
hold
%plot(rcl,'*')

% Bode plot of loop transfer function
w = logspace(-1,1,200);
bode(F,G,K,0,1,w);
[mag,phase] = bode(F,G,K,0,1,w);
title('Bode Plot')
[gm,pm,wg,wc] = margin(mag,phase,w)

% Reference Design
Nx = [1 0 1 0]';
Nb = K*Nx

Nb =

2.0000

% Step Response
t = 0:.004:15;
subplot(211);
step(F-G*K,G*Nb,H,0,1,t);
title('2-Mass control ref. step')
subplot(212);
step(F-G*K,G,H,0,1,t);
title('2-Mass process noise step ')

% Now Lets design an Estimator

% Estimator Design for p^2 = 10000
L = lqe(F,G,H,10000,1)
```

140
L =

    79.1232
    96.9383
    7.8252
    30.6172

% Closed loop estimator poles
pe = eig(F-L*H)

pe =

    -1.1554 + 2.9310i
    -1.1554 - 2.9310i
    -2.7770 + 1.2067i
    -2.7770 - 1.2067i

% Check natural frequency
a.pe = abs(eig(F-L*H))

% Symmetric Root Locus for LQE design
%A = [F zeros(4); -H'*H -F'];
%B = [G ;0;0;0];
%C = [0 0 0 0 G'];
%rlocus(A,B,C,0)
%hold
%plot(pe,'*')
%title('SRL for LQE')
%axis([-3 3 -3 3]);
%axis('square');

% Compensator state description for r,y to u. D1(s) D2(s)
A = [F-G*K-L*H];
B = [G*Nb L];
C = -K;
D = [Nb 0];

% Check poles and zeros of compensator
[Zr,Pr,Kr] = ss2zp(A,B,C,D,1);
[Zy,Py,Ky] = ss2zp(A,B,C,D,2);
[Zp,Pp,Kp] = ss2zp(F,G,H,0,1);

Zr =

    -1.1554 + 2.9310i
    -1.1554 - 2.9310i
    -2.7770 + 1.2067i
    -2.7770 - 1.2067i

Pr =
-0.9163 + 3.9245i
-0.9163 - 3.9245i
-4.2256 + 2.0723i
-4.2256 - 2.0723i

Zy =
  -0.1677 + 0.9384i
  -0.1677 - 0.9384i
  -0.3948

Py =
  -0.9163 + 3.9245i
  -0.9163 - 3.9245i
  -4.2256 + 2.0723i
  -4.2256 - 2.0723i

Zp =
  -25.2778

Pp =

0
-0.0198 + 1.0003i
-0.0198 - 1.0003i
0

% Combined Plant and Compensator state description (Open Loop) D(s)*G(s)
Fol = [F zeros(4) ; L*H F-L*H-G*K];
Gol = [G ; [0 0 0 0]'];
Hol = [0 0 0 0 K];

% An alternate derivation for D(s)*G(s)
[num1,dem1] = ss2tf(A,B,C,D,2);
[num2,dem2] = ss2tf(F,G,H,0,1);
num = conv(num1,num2);
dem = conv(den1,dem2);

% Plant/Compensator Root Locus
%rlocus(-num,dem)
%rlocus(Fol,Gol,Hol,0)
%title('Plant/Compensator Root Locus')
%rcl = rlocus(num,dem,-1)
%hold
%plot(rcl,'*')
%axis([-5 5 -5 5]);
%axis('square');
% Plant/Compensator Bode Plot
w = logspace(-1,1,200);
[bode(Fol,Gol,Hol,0,1,w])
[mag,phase] = bode(Fol,Gol,Hol,0,1,w);
title('Bode Plot')
[gm,pm,wg,wc] = margin(mag,phase,w)

gm =

1.5154

pm =

17.3683

% Closed loop Plant/Estimator/Control system for step response
Acl = [F -G*K ; L*H F -G*K -L*H];
Bcl = [G*Nb G [0 0 0]' ; G*Nb [0 0 0]' L];
Ccl = [H 0 0 0];
Dcl = [0 0 1];
t = 0:.01:15;

% Reference Step
%[Y, X] = step(Acl,Bcl,Ccl,Dcl,1,t);
%subplot(211)
%step(Acl,Bcl,Ccl,Dcl,1,t);
%title('2-Mass System Compensator Reference Step')

% Control Effort
%Xh = [X(:,5) X(:,6) X(:,7) X(:,8)];
%u = -K*Xh';
%subplot(212)
%plot(t,u+Nb)
%title('Control effort')

% Process Noise Step
%[Y, X] = step(Acl,Bcl,Ccl,Dcl,2,t);
%subplot(211)
%step(Acl,Bcl,Ccl,Dcl,2,t);
%title('Process Noise Step')

% Sensor Noise Step
%[Y, X] = step(Acl,Bcl,Ccl,Dcl,3,t);
%subplot(212)
%step(Acl,Bcl,Ccl,Dcl,3,t);
%title('Sensor Noise Step')
4.2.3 Alternate reference input methods

\[
\dot{x} = F\dot{x} + Gu + L(y - H\dot{x})
\]
\[
u = -K\dot{x}
\]

With additional reference the equations are:
\[
\dot{x} = (F - LH - GK)\dot{x} + Ly + Mr
\]
\[
u = -k\dot{x} + Nr
\]

Case 1:

\[M = G\tilde{N}\]
\[N = \tilde{N}\]

This is same as adding normal reference input.

Case 2:

\[N = 0\]
\[M = -L\]
\[\Rightarrow \dot{x} = (F - GK - LH)\dot{x} + L(y - r)\]

This is same as classical error feedback term.

Case 3:

\[M \text{ and } N \text{ are used to place zeros.}\]

Transfer function
\[
\Rightarrow \frac{y}{r} = \frac{N\gamma(s)b(s)}{\alpha_e(s)\alpha_e(s)}
\]

where
\[\gamma(s) = \det(sI - A + \tilde{M}K)\]
\[\tilde{M} = \frac{M}{N}\]

N for unity gain.

Now estimator poles are not cancelled by zeros. The zeros are placed so as to get good tracking performance.
4.3 Discrete Estimators

4.3.1 Predictive Estimator

\[ x_{k+1} = \Phi x_k + \Gamma u_k + \Gamma_1 w \]
\[ y = H x + v \]

Estimator equation

\[ \Rightarrow \bar{x}_{k+1} = \Phi \bar{x}_k + \Gamma u_k + L_p (y_k - \bar{y}_k) \]

This is a predictive estimator as \( \bar{x}_{k+1} \) is obtained based on measurement at \( k \).

\[ \bar{x}_{k+1} = x_{k+1} - \bar{x}_{k+1} = (\Phi - L_p H) \bar{x}_k + \Gamma_1 w - L_p v \]

\[ a_c(z) = det(zI - \Phi + L_p H) \]

Need \( O(\Phi, H) \) to build estimator.

4.3.2 Current estimator

Measurement update:

\[ \hat{x}_k = \bar{x}_k + L_c (y_k - H \bar{x}_k) \]

Time update:

\[ \bar{x}_{k+1} = \Phi \hat{x}_k + \Gamma u_k \quad (128) \]

These are the equations for a discrete Kalman Filter (in steady state).

Estimator block diagram

Substitute \( \hat{x} \) into measurement update equation:

\[ \Rightarrow \bar{x}_{k+1} = \Phi \bar{x}_k + L_c (y - H \bar{x}) + \Gamma u_k \]

\[ = \Phi \bar{x}_k + \Phi L_c (y - H \bar{x}_k) + \Gamma u \]
Comparing with predictive estimator

\[ L_p = \Phi L_c \]
\[ \ddot{x}_{k+1} = (\Phi - L_c H\Phi) \ddot{x}_k \]

Now

\[ \mathcal{O}[\Phi, H\Phi] \]

should be full rank for this estimator.

\[ \mathcal{O}_c = \begin{pmatrix} H\Phi \\ H\Phi^2 \\ \vdots \\ H\Phi^n \end{pmatrix} = \mathcal{O}_p \Phi \]

\[ \det \mathcal{O}_c = \det \mathcal{O}_p \det \Phi \]

⇒ Cannot have poles at origin of open loop system.

\[ L_c = \Phi^{-1} L_p \]

4.4 Miscellaneous Topics

4.4.1 Reduced order estimator

- If one of the states \( x_i = y \), the output, then there is no need to estimate that state.
- Results in more complicated equations.

4.4.2 Integral control

\[ \dot{x} = F x + G u \]
\[ y = H x \]
\[ \dot{x}_i = -y + r = -H x + r \]
\[
\begin{pmatrix}
\dot{x}_i \\
\dot{x}_2
\end{pmatrix} =
\begin{pmatrix}
0 & -H \\
0 & F
\end{pmatrix}
\begin{pmatrix}
x_i \\
x
\end{pmatrix} +
\begin{pmatrix}
0 \\
G
\end{pmatrix} u
\]

\[u = \begin{pmatrix}
-K_i \\
K
\end{pmatrix}
\begin{pmatrix}
x_i \\
x
\end{pmatrix}\]

Gains can be calculated by place command.

\[\begin{pmatrix}
-K_i & K
\end{pmatrix} = \text{place}\left(\begin{pmatrix}
0 & -H \\
0 & F
\end{pmatrix},\begin{pmatrix}
0 \\
G
\end{pmatrix},P\right)\]

- \(n+1\) poles can be placed.
- Controllable if \(F,G\) are controllable and \(F,G,H\) has no zeros at \(s = 0\).

### 4.4.3 Internal model principle

The output \(y(t)\) is made to track arbitrary reference \(r(t)\) (e.g., sinusoidal tracking for a spinning disk-drive).

- Augment state with \(r\) dynamics.

Ex 1:

\[
\begin{align*}
\dot{x} &= Fx + Gu \\
y &= Hx \\
\dot{r} &= 0 \rightarrow \text{constant step}
\end{align*}
\]

Or

\[
\ddot{r} - w_0^2 r = 0 \rightarrow \text{Sinusoid}
\]

Define

\[
e = y - r
\]
as new state which is driven to zero.

### 4.4.4 Polynomial methods

- Direct transformation between input/output transfer function.
- Often advantageous for adaptive control.
5 Kalman

(Lecture)
6 Appendix 1 - State-Space Control

Why a Space? The definition of a vector or linear space is learned in the first weeks of a linear algebra class and then conveniently forgotten. For your amusement (i.e. you-will-not-be-held-responsible-for-this-on-any-homework-midterm-or-final) I will refresh your memory.

A linear vector space is a set \( V \), together with two operations:

+ , called \textit{addition}, such that for any two vectors \( x,y \) in \( V \) the sum \( x+y \) is also a vector in \( V \)

\( \cdot \), called \textit{scalar multiplication}, such that for a scalar \( c \in \mathbb{R} \) and a vector \( x \in V \) the product \( c \cdot V \) is in \( V \).

Also, the following axioms must hold:

\( (A1) \quad x + y = y + x, \forall x,y \in V \) (commutativity of addition)

\( (A2) \quad x + (y + z) = (x + y) + z, \forall x,y,z \in V \) (associativity of addition)

\( (A3) \quad \text{There is an element in } V, \text{ denoted by } 0_v \text{ (or } 0 \text{ if clear from the context), such that } (x + 0_v = 0_v + x = x), \forall x \in V \) (existence of additive identity)

\( (A4) \quad \text{For each } x \in V, \text{ there exists an element, denoted by } -x, \text{ such that } x + (-x) = 0_v \) (existence of additive inverse)

\( (A5) \quad \text{For each } r_1, r_2 \text{ in } \mathbb{R} \text{ and each } x \in V, \)
\[ r_1 \cdot (r_2 \cdot x) = (r_1 r_2) \cdot x \]

\( (A6) \quad \text{For each } r \text{ in } \mathbb{R} \text{ and each } x,y \in V, \)
\[ r \cdot (x + y) = r \cdot x + r \cdot y \]

\( (A7) \quad \text{For each } r_1, r_2 \text{ in } \mathbb{R} \text{ and each } x \in V, \)
\[ (r_1 + r_2) \cdot x = r_1 \cdot x + r_2 \cdot x \]

\( (A8) \quad \text{For each } x \in V, \)
\[ 1 \cdot x = x \]

Example 1: \( n \)-tuples of scalars. (We will not prove any of the examples.)

\[ \mathbb{R}^n \triangleq \left\{ \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_n \end{bmatrix} \mid x_i \in \mathbb{R} \right\} \]

Example 2: The set of all continuous functions on a finite interval

\[ C[a,b] \triangleq \{ f : [a,b] \rightarrow \mathbb{R} \mid f \text{ is continuous} \} \]

\(^2\mathbb{R} \text{ is used to denote the field of real numbers. A vector space may also be defined over an arbitrary field.} \)
A *subspace* of a vector space is a subset which is itself a vector space.

**Example 1**

\[ \mathcal{N}(A) \triangleq \{ x \in \mathbb{R}^n \mid Ax = 0, A \in \mathbb{R}^{n \times q} \} \]

is a subspace of \( \mathbb{R}^n \) called the *nullspace* or \( A \).

**Example 2**

\[ \mathcal{R}(A) \triangleq \{ x \in \mathbb{R}^n \mid \exists u \in \mathbb{R}^q \text{ such that } Au = x, A \in \mathbb{R}^{n \times q} \} \]

is a subspace of \( \mathbb{R}^n \) called the *range* of \( A \).

And finally a very important example for this class.

**Example 3**

\[ \{ x \in C^0[\mathbb{R}_+] \mid \dot{x} = Ax \text{ for } t \geq 0 \} \]

is a subspace of \( C^0[\mathbb{R}_+] \) (continuous functions: \( \mathbb{R}_+ \to \mathbb{R}^n \)). This is the set of all possible trajectories that a linear system can follow.
Controller

\[ A_c = \begin{bmatrix} -a_1 & -a_2 & -a_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \]

\[ B_c = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \]

\[ C_c = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix} \]

\( A_c \) is top companion
Observer

\[ A_o = \begin{bmatrix} -a_1 & 1 & 0 \\ -a_2 & 0 & 1 \\ -a_3 & 0 & 0 \end{bmatrix} \]

\[ B_o = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \]

\[ C_o = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \]

\( A_o \) is left companion
Controllability

\[ A_{\text{co}} = \begin{bmatrix} 0 & 0 & -a_3 \\ 1 & 0 & -a_2 \\ 0 & 1 & -a_1 \end{bmatrix} \]

\[ B_{\text{co}} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \]

\[ C_{\text{co}} = \begin{bmatrix} \beta_1 & \beta_2 & \beta_3 \end{bmatrix} \]

\[ A_{\text{co}} \] is right companion
Observability

\[ A_{ob} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -a_3 & -a_2 & -a_1 \end{bmatrix} \]

\[ B_{ob} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} \]

\[ C_{ob} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \]

\( A_{ob} \) is bottom companion

\[ G(s) = \frac{b_1 s^2 + b_2 s + b_3}{s^3 + a_1 s^2 + a_2 s + a_3} = \sum_{i=1}^{\infty} \beta_i s^i \]
Part VII
Advanced Topics in Control

1 Calculus of Variations - Optimal Control

1.1 Basic Optimization

1.1.1 Optimization without constraints

\[ \min_u L(u) \text{ without constraints.} \]

Necessary condition

\[ \frac{\partial L}{\partial u} = 0 \]

\[ \frac{\partial^2 L}{\partial u^2} \geq 0 \]

1.1.2 Optimization with equality constraints

\[ \min_u L(x, u) \text{ subject to } f(x, u) = 0 \]

Define

\[ H = L(x, u) + \lambda^T f(x, u) \]

“adjoin” constraints

Since \( u \) determines \( x \)

\[ H = L \]

Differential changes on \( H \) due to \( x \) and \( u \)

\[ dH = \frac{\partial H}{\partial x} dx + \frac{\partial H}{\partial u} du \]

We want this to be 0, so start by selecting \( \lambda \) such that

\[ \frac{\partial H}{\partial x} = 0 \]

\[ \frac{\partial H}{\partial x} = \frac{\partial L}{\partial x} + \lambda^T \frac{\partial f}{\partial x} = 0 \quad (129) \]

\[ \Rightarrow \lambda^T = -\frac{\partial L}{\partial x} \left( \frac{\partial f}{\partial x} \right)^{-1} \]

\[ \Rightarrow dL = dH = \frac{\partial H}{\partial u} du \]
For a minimum we need
\[ \frac{dL}{du} = 0 \]
\( \Rightarrow \)
\[ \frac{\partial H}{\partial u} = 0 = \frac{\partial L}{\partial u} + \lambda^T \frac{\partial f}{\partial u} = 0 \]  \hspace{1cm} (130)

**Summary**

\[ \min_u L(x, u) \] such that \( f(x, u) = 0 \)

\[ \frac{\partial H}{\partial x} = 0, \frac{\partial H}{\partial u} = 0 \]

where
\[ H = L(x, u) + \lambda^T f(x, u) \]

**Example**

\[ \min_u L(u) = \frac{1}{2} \left( \frac{x^2}{a^2} + \frac{u^2}{b^2} \right) \]

such that
\[ f(x, u) = x + mu - c = 0 \]

\[ H = \frac{1}{2} \left( \frac{x^2}{a^2} + \frac{u^2}{b^2} \right) + \lambda(x + mu - c) \]

\[ \frac{\partial H}{\partial x} = \frac{x}{a^2} + \lambda = 0 \]
\[ \frac{\partial H}{\partial u} = \frac{u}{b^2} + \lambda m = 0 \]
\[ x + mu - c = 0 \]

There are 3 equations and three unknowns \( x, u, \lambda \)

\[ \Rightarrow x = \frac{a^2 c}{a^2 + m^2 b^2}, u = \frac{b^2 mc}{a^2 + m^2 b^2}, \lambda = \frac{c}{a^2 + m^2 b^2} \]
1.1.3 Numerical Optimization

Non-linear case

\[ \min_u L(x, u) + \lambda^T f(x, u) = \min_u H \]
\[ \text{where } f(x, u) = 0 \]

Setting \( \frac{\partial H}{\partial x} = 0 \) and \( \frac{\partial H}{\partial u} = 0 \) leads to an algorithm for computing \( u \):

1. Guess \( u \), for \( f(x, u) = 0 \) solve for \( x \).
2. \( \lambda^T = -\frac{\partial L}{\partial x} \left( \frac{\partial f}{\partial x} \right)^{-1} \).
3. \( \frac{\partial H}{\partial u} = \frac{\partial L}{\partial u} + \lambda^T \frac{\partial f}{\partial u} \), test for \( \frac{\partial H}{\partial u} = 0 \).
4. \( u \leftarrow u - \eta \frac{\partial H}{\partial u} \)

Therefore one does a gradient search.

Related optimization

- Inequality constraints \( L(x, u) \) subject to \( f(x, u) \leq 0 \), requires Kuhn-Tucker conditions.
- If \( L \) and \( f \) linear in \( x, u \) then use Linear Programming.

Constrained optimization + calculus of variations \( \rightarrow \) Euler-Lagrange Equations.

1.2 Euler-Lagrange and Optimal control

1.2.1 Optimization over time

Constraint

\[ \dot{x} = f(x, u, t), \ x(t_0) \]

with cost

\[ \min_{u[t]} J = \varphi(x(t_f), t_f) + \int_{t_0}^{t_f} L(x(t), u(t), t) \, dt \]

Adjoin the constraints

\[ J = \varphi(x(t_f), t_f) + \int_{t_0}^{t_f} L(x(t), u(t), t) + \lambda^T(t) \{ f(x, u, t) - \dot{x} \} \, dt \]

where the Hamiltonian is defined as

\[ H = L(x(t), u(t), t) + \lambda^T(t) f(x, u, t). \]

Using integration by parts on the \( \lambda^T \dot{x} \) term

\[ J = \varphi(x(t_f), t_f) - \lambda^T(t_f)x(t_f) + \lambda^T(t_0)x(t_0) + \int_{t_0}^{t_f} H + \dot{x} \lambda^T x \, dt. \]
To minimize variation in $\tilde{J}$ due to variations in the control vector $u(t)$ for fixed times $t_0$ and $t_f$, minimize

$$\delta \tilde{J} = \left[ \frac{\partial \varphi}{\partial x} - \lambda^T \right] \delta x \big|_{t=t_f} + \left[ \lambda^T \delta x \right] \big|_{t=t_0} + \int_{t_0}^{t_f} \left[ \frac{\partial H(x,u,t)}{\partial x} + \lambda^T \delta x + \frac{\partial H(x,u,t)}{\partial u} \delta u \right] dt$$

Choose $\lambda(t)$ such that the $\delta x$ coefficients vanish

$$\lambda^T = -\frac{\partial H}{\partial x} = -\frac{\partial L}{\partial x} - \lambda^T \frac{\partial f}{\partial x}$$

$$\lambda^T(t_f) = \frac{\partial \varphi}{\partial x(t_f)}$$

Then

$$\delta \tilde{J} = \lambda^T(t_0) \delta x(t_0) + \int_{t_0}^{t_f} \frac{\partial H}{\partial u} \delta u \, dt$$

But we still want to minimize $\delta \tilde{J}$ for variations $\delta u$ at the optimal $u$ yielding the optimal $J$ (i.e., we need $\frac{\partial \delta J}{\partial u} = 0$). Then

$$\frac{\partial H}{\partial u} = 0 = \frac{\partial L}{\partial u} + \lambda^T \frac{\partial f}{\partial u} \hspace{1cm} (133)$$

$\lambda$ is called an ‘influence function’ on $J$.

Equations 131, 132 and 133 define the Euler Lagrange Equations for a two-point boundary problem.

Solution of these equations leads to Pontryagin’s minimum principle (w/o proof)

$$H(x^*,u^*,\lambda^*,t) \leq H(x^*,u,\lambda^*,t)$$

**Summary:** given the cost

$$\min_{u(t)} J = \varphi(x(t_f),t_f) + \int_{t_0}^{t_f} L(x(t),u(t),t) \, dt,$$

and constraint

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

$$\dot{\lambda} = -\left( \frac{\partial f}{\partial x} \right)^T \lambda - \left( \frac{\partial L}{\partial x} \right)^T = -\frac{\partial H}{\partial x}$$

where $u$ is determined by

$$\frac{\partial H}{\partial u} = 0 \text{ or } \left( \frac{\partial f}{\partial u} \right)^T \lambda + \left( \frac{\partial L}{\partial u} \right)^T = 0.$$  

$$x(t_0)$$

is given;

$$\lambda(t_f) = \left( \frac{\partial \varphi}{\partial x} \right)^T_{t_f}$$

is the boundary condition. In general this leads to some optimal solution for $u(t)$, which is determined by initial value $x(t_0)$ and final constraint $\varphi(x(t_f),t_f)$, resulting in open-loop control since the control is fixed without feedback.
1.2.2 Miscellaneous items

- The family of optimal paths for all initial conditions \(x(t_0), 0 < t_0 < t_f\) is called a ‘field of extremals’.

- If we solve for all states such that \(u = \text{function}(x, t)\) then this is the ‘optimal feedback control law’.

- \(J^* = J(x^*, t)\) gives the optimal return function.

1.3 Linear system / ARE

Consider the time-varying linear system

\[
\dot{x} = F(t)x + G(t)u, \ x(t_f) \rightarrow 0
\]

with cost function

\[
J = \frac{1}{2}(x^T S_f x)_{t=t_f} + \frac{1}{2} \int_{0}^{t_f} x^T Q x + u^T R u \ dt
\]

with \(Q\) and \(R\) positive definite.

Then using the Euler-Lagrange equations from Equation 1.33

\[
\frac{\partial H}{\partial u} = 0 = Ru + G^T \lambda
\]

so that the control law is

\[
u = -R^{-1}G^T \lambda
\]

and from Equation 1.31

\[
\dot{\lambda} = -Q x - F^T \lambda
\]

with boundary condition

\[\lambda(t_f) = S_f x(t_f)\]

from Equation 1.32. Assimilating into a linear differential equation

\[
\begin{bmatrix}
\dot{x} \\
\dot{\lambda}
\end{bmatrix} =
\begin{bmatrix}
F & -GR^{-1}G^T \\
-Q & -F^T
\end{bmatrix}
\begin{bmatrix}
x \\
\lambda
\end{bmatrix},
\begin{bmatrix}
x(0) \\
\lambda(t_f)
\end{bmatrix} =
\begin{bmatrix}
x(t_0) \\
S_f x(t_f)
\end{bmatrix}
\]

Try a solution that will convert the two-point boundary problem to a one-point problem. As a possible solution try \(\lambda(t) = S(t)x(t)\)

\[
\dot{S} x + S \dot{x} = -Q x - F^T S x
\]

so that

\[
(\dot{S} + SF + F^T S - SGR^{-1}G^T S + Q)x = 0
\]

resulting in the matrix Ricatti equation

\[
\dot{S} = -SF - F^T S + SGR^{-1}G^T S - Q, \ \text{with} \ S(t_f) = S_f
\]

Note that

\[
u = -R^{-1}G^T S(t)x(t) = K(t)x(t)
\]
where $K(t)$ is a time-varying feedback.

$\dot{S} = 0$ gives the steady state solution of the Algebraic Ricatti Equation (A.R.E) where $t_f \to \infty$. Then $K(t) = K$ ie state feedback.

The A.R.E may be solved

1. by integrating the differential Ricatti equation to a fixed point
2. symbolically
3. by eigen value decomposition
4. with the schur method
5. with a generalized eigen formulation.
6. MATLAB

1.3.1 Another Example

\[
\dot{x} = ax + u \\
J = \int_{0}^{\infty} a^2 + \rho^2 x^2
\]

with open loop pole at $s = a$. Steady state A.R.E gives

\[
sF + F^T s - sGR^{-1}G^T s + Q = 0 \\
as + as - s.1.1.1.s + \rho^2 = 0
\]

Therefore

\[
s^2 - 2as - \rho^2 = 0
\]

and

\[
s = \frac{2a \pm \sqrt{4a^2 + 4\rho^2}}{2} = a \pm \sqrt{a^2 + \rho^2}
\]

\[
K = R^{-1}G^T s = s = a + \sqrt{a^2 + \rho^2}
\]

with the closed loop poles at $s = a - k$. See figure 17.

1.4 Symmetric root locus

Take the Laplace transform of the Euler Lagrange equations for the linear time invariant case of example 1.3.

\[
sx(s) = Fx(s) + Gu(s) \\
s\lambda(s) = -Qx(s) - F^T \lambda(s) \\
u(s) = -R^{-1}G^T \lambda(s) \\
Q = H^T H \rho^2 \text{ and } R = 1.
\]
Then
\[ u(s) = G^T(sI + F^T)^{-1}H^T H \rho^2(sI - F)^{-1} G \hat{u}(s) \]
where
\[ -G^T(-s) = G^T(sI + F^T)^{-1} H^T \]
and
\[ G(s) = H(sI - F)^{-1} G, \]
the open loop transfer function. Then
\[ \left[ 1 + \rho^2 G^T(-s) G(s) \right] u(s) = 0 \]
giving the symmetric root locus (SRL).

- Has 2N roots for the coupled system of x, λ of order N.
- As \( t \to \infty \), \( u(t) = -K_x \lambda(t) \) it becomes an Nth order system with N stable poles.

1.4.1 Discrete case

Given the discrete plant
\[ x(k + 1) = \Phi x(k) + \Gamma u(k), \]
\[ u = -K x \]
the task is to minimize the cost function
\[ J = \frac{1}{2} \sum_{k=0}^{N} \left[ x^T(k)Qx(k) + u^T(k)Ru(k) \right]. \]

Using a Lagrange multiplier
\[ J = \frac{1}{2} \sum_{k=0}^{N} \left[ x^T(k)Qx(k) + u^T(k)Ru(k) + \lambda^T(k + 1)(-x(k + 1) + \Phi x(k) + \Gamma u(k)) \right] \]
leads to the Euler Lagrange equations
\[ \frac{\partial H}{\partial u(k)} = u^T(k)R + \lambda^T(k + 1) \Gamma = 0 \]
(134)
\[ \frac{\partial H}{\partial \lambda(k + 1)} = -x(k + 1) + \Phi x(k) + \Gamma u(k) = 0 \]
(135)
\[ \frac{\partial H}{\partial x(k)} = \lambda^T(k + 1) \Phi + x^T(k)Q - \lambda^T(k) = 0 \]
(136)
(137)
From the cost function clearly $u(N) = 0$ and from Equation 134 $\lambda(N + 1) = 0$ so that from Equation 136

$$\lambda(N) = Qx(N).$$

Again choosing

$$\lambda(k) = S(k)x(k)$$

it follows that

$$u(k) = -R^{-1}\Gamma^T\lambda(k + 1)$$

with the Discrete Ricatti Equation (non-linear in $S$)

$$S(k) = \Phi^T\left[S(k + 1) - S(k + 1) \Gamma N^{-1}\Gamma^T S(k + 1)\right] \Phi + Q,$$

with the end condition

$$S(N) = Q$$

and where

$$N = R + \Gamma^T S(k + 1)\Gamma.$$

The feedback gain then is

$$K(k) = \left[R + \Gamma^T S(k + 1)\Gamma\right]^{-1}\Gamma^T S(k + 1)\Phi.$$

Note that the minimum cost is

$$J = \frac{1}{2}x^T(0)S(0)x(0).$$

### 1.4.2 Predictive control

Here the cost is taken as

$$J(x, u_{t\rightarrow t+N}) = \sum_{k=t}^{t+N-1} L(K, x, \dot{u})$$

giving $u^*(t), \ldots u^*(t + N)$. Only $u(t) = K(t)x(t)$ is used, and the RHC problem re-solved for time $t + 1$ to get $u(t + 1)$. $u(t)$ provides a stable control if $N \geq$ the order of the plant, and the plant is reachable and observable.

- Guaranteed stabilizing if prediction horizon length exceeds order of plant.
- (SIORHC) = stabilizing input-output receding horizon control.
- This is popular for slow linear and non-linear systems such as chemical batch processes.

See ‘Optimal predictive and adaptive control’ – Mosca.
1.4.3 Other applied problems

Terminal constraint of $x$

Here $x(t_f) = x_f$ or

$$\Psi(x(t_f)) = 0.$$ \hfill (1)

Adjoin $v^T \Psi(x(t_f))$ to $J$. Then

$$\lambda^T(t_f) = \left( \frac{\partial \varphi}{\partial x} + v^T \frac{\partial \Psi}{\partial x} \right)_{t=t_f}. \hfill (2)$$

Examples include 1) robotics for a fixed endpoint at fixed time, 2) transfer of orbit radius in given time.

Unspecified time

Want to move an arm from A to B, but do not know how long it will take. A 4'th condition is required together with the prior Euler Lagrange equations to minimize time as well

$$\left( \frac{\partial \varphi}{\partial t} + \lambda^T f + L \right)_{t=t_f} = 0. \hfill (3)$$

Some minimum time problems.

◊ Missile trajectory.

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

$$J(u) = \int_{t_0}^{t_f} 1 \, dt = t_f - t_0 \Rightarrow L = 1$$

such that $|u(t)| \leq 1$ is a finite control. Solving,

$$H = L + \lambda^T [f(t) + g(t)u], \quad L = 1$$
Clearly to minimize $H$ the resulting optimal $u$ is ‘bang-bang control’

$$u = \begin{cases} 
1 & \lambda^T g < 0 \\
-1 & \lambda^T g > 0 
\end{cases}$$

To solve for $\lambda$

$$-\frac{\partial H}{\partial x} = \dot{\lambda}^T = -\frac{\partial L}{\partial x} - \lambda^T \frac{\partial f}{\partial x}$$

so that $\lambda$ is obtained as the solution to

$$\dot{\lambda}^T = -\lambda^T f(t)$$

and

$$\lambda^T [fx + gu] = -1.$$
2 Dynamic Programming (DP) - Optimal Control

2.1 Bellman

Dynamic Programming represents an alternative approach for optimization.

- Suppose that we need to determine an optimal cost to go from point a to point c.

- If the optimal path from a to c includes the path from a to b, then the path from b to c is the optimal path from b to c.

\[ J_{ac}^* = J_{ab}^* + J_{bc}^* \]

Bellman: An optimal policy has the property that whatever the initial state and the initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.
2.2 Examples:

2.2.1 Routing

- Start out with paths $bc$ or $bd$ or $be$.
- The optimal path from $b$ to $f$ is hence the minimum of $J_{bc} + J^*_{cf}$, $J_{bd} + J^*_{df}$, $J_{be} + J^*_{ef}$
- $J^*_{cf}$, $J^*_{df}$ and $J^*_{ef}$ are determined by back-tracking.
- The viterbi algorithm for processing of speech is based on dynamic programming.

2.2.2 City Walk

- Want 'walk' from $c$ to $h$. Which is the optimal path?
- We have two choices; proceed along path from either $c$ to $d$ or $c$ to $f$.

So, $C^*_{cdh} = J_{cd} + J^*_{dh} = 15$

$C^*_{cfh} = J_{cf} + J^*_{fh} = 8$

$J^*_{ch} = \min\{C^*_{cdh}, C^*_{cfh}\}$

$= \min\{15, 8\} = 8$

- As with the routing example, we need to know $J^*_{dh}$ and $J^*_{fh}$ initially, which implies a backtracking problem.

2.3 General Formulation

$$J^*_{k:N}(x(k)) = \min_{U(k)}[J_{k,k+1}(X(k), U(k)) + J^*_{k+1:N}(X(k + 1))]$$
\[ J_{N-k,N}^*(X(N-k)) = \min_{U(N-k)} \left[ J_{N-k,k-1}(X(N-k), U(N-k)) + J_{N-k-1,k}(X(N-k+1)) \right] \]

\[ \Rightarrow J_{N-(k-1)} \Rightarrow J_{N-k,N}^* \]

- This implies a backward recurrence.
- Start at the last time step and define \( J_{N-1,N}^* \triangleq J_{N,N}^* \).
- Then solve for \( J_{N-1,N}^* \) and work backwards.

**Curse of Dimensionality:** For all possible \( X \), suppose that \( X \in \mathbb{R}^3 \) and \( X_i \) can take-on 100 values. Thus we need to have already found and stored \( 10^2 \times 10^2 \times 10^3 = 10^9 \) values for \( J_{N-1,k,0}^* \) to solve for \( J_{N-1,k,0}^* \).
- Dynamic programming is _not practical_ unless we can intelligently restrict the search space.

### 2.4 LQ Cost

Given

\[ X_{n+1} = AX_n + bU_n \]
\[ Y_n = HX_n \]

we have the following:

\[ J_{0,N} = \frac{1}{2} \sum_{k=0}^{N-1} \left( X^T(k)QX(k) + U^T(k)RU(k) \right) + \frac{1}{2} X^T(N)QX(N) \]

Need to minimize \( J_{0,N} \) over time. We saw before that the solution to the above involves a _Riccati equation_. By definition,

\[ J_{0,N}^*(X_0) = \min_{U_0, U_1, \ldots, U_N} J_{0,N}(X_0) \]

Let’s write this as follows:

\[ J_{0,N} = \min_{U \in \mathbb{R}} \left\{ \frac{1}{2} X^T(0)Q(0)X(0) + U^T(0)RU(0) + \frac{1}{2} \sum_{k=1}^{N-1} \left( X^T(k)QX(k) + U^T(k)RU(k) \right) + X^T(N)QX(N) \right\} \]

Next, express the cost at the final state as the following.

\[ J_{N,N}^* = \frac{1}{2} X^T(N)QX(N) \]

Let \( S(N) = Q \),

\[ J_{N-1,N} = \frac{1}{2} X^T(N-1)QX(N-1) + \frac{1}{2} U^T(N-1)RU(N-1) + \frac{1}{2} X^T(N)S(N)X(N) \quad (138) \]
where, \( X(N) = (AX(N-1) + BU(N-1)) \)

\[
J_{N-1,N}^* \triangleq \min_{u(N-1)} J_{N-1,N}
\]

Take \( \frac{\partial J_{N-1,N}}{\partial U(N-1)} = 0 = RU(N-1) + B^T S(N)[AX(N-1) + BU(N-1)] \)

(Exercise: Show that \( \frac{\partial^2 J}{\partial U^2} \) is P.D.)

\[
\Rightarrow U^*(N-1) = -\left[R + B^T S(N)B \right]^{-1} S(N) A X(N-1)
\]

1. Substitute \( U^*(N-1) \) back in equation 138. \( \Rightarrow J_{N-1,N}^* = \frac{1}{2} X^T(N-1) S(N-1) X(N-1) \)

where \( S(N-1) \) is the same as we derived using the Euler-Lagrange approach.

2. \( S(N-1) \) is given by a Ricatti Equation.

3. Continue by induction.

Comments

- \( J^* \) is always quadratic in \( X \). This makes it easy to solve.

- As \( N \to \infty \), this becomes the LQR. \( S(k) = S \), and \( J_k^* = X_k^T S X_k \)

2.5 Hamilton-Jacobi-Bellman (HJB) Equations

This is the generalization of DP to continuous systems.

\[
J = \varphi(X(t_f), t_f) + \int_t^{t_f} L(X(\tau), U(\tau), \tau)\,d\tau
\]

such that, \( F(X, U, t) = \dot{X} \). Need to minimize \( J \) using DP.

\[
J^*(X(t), t) = \min_{U(\tau), t < \tau < t + \Delta t} \left[ \int_t^{t+\Delta t} L d\tau + J^*(X(t + \Delta t), t + \Delta t) \right]
\]

by the principle of optimality.

Expand \( J^*(X(t + \Delta t, t + \Delta t)) \) using Taylor’s series about \( (X(t), t) \). So,

\[
J^*(X(t)) = \min_{u(\tau) \in \mathcal{U}} \left[ \int_t^{t+\Delta t} L(X, U, \tau)\,d\tau + J^*(X(t), t) + \frac{\partial J^*}{\partial t} (X(t), t) \Delta t \right.
\]

\[
\left. + \left[ \frac{\partial J^*}{\partial X} (X(t), t) \right]^T \begin{bmatrix} F(X, U, t) \Delta t + \text{higher order terms} \end{bmatrix} - X(t) \right]
\]

\[
\min_{U(t)} [L(X, U, t) \Delta t + J^*(X(t), t) + \frac{\partial J^*}{\partial t} (X(t), t) \Delta t + \frac{\partial J^*}{\partial X} (X(t), t) F(X, U, t)] \Delta t
\]

does not depend on \( U \)

170
Divide by $\Delta t$ and take limit as $\Delta t \to 0$

$$0 = \frac{\partial J^*}{\partial t}(X(t), t) = \min_u [L(X, U, t) + \frac{\partial J^*}{\partial X}(X(t), t)F(X, U, t)]$$

Boundary Condition: $J^*(X(t_f, t_f) = \varphi(X(t_f), t_f)$

Define Hamiltonian as:

$$H(X, U, \frac{\partial J^*}{\partial X}, t) \triangleq L(X, U, t) + \frac{\partial J^*}{\partial X}(X(t), t)[F(X, U, t)]$$

(recall $H$ from Euler-Lagrange equation \(\Rightarrow \lambda^T(t) = \frac{\partial J^*}{\partial X}(X(t), t) \leftarrow \text{influence function}\))

The optimal Hamiltonian is given by:

$$H^*(X, U^*, \frac{\partial J^*}{\partial X}, t) = \min_{U(t)} H(X(t), U(t), \frac{\partial J^*}{\partial X}, t)$$

Thus, $0 = \frac{\partial J^*}{\partial t} + H^*(X, U^*, \frac{\partial J^*}{\partial X}, t)$ (139)

Equations 139 and 140 are the HJB equations.

Exercise: Derive the E-L equations from the H-J-B Equation.

2.5.1 Example: LQR

$$\dot{X} = A(t)X + B(t)U$$

$$J = \frac{1}{2} X^T(t_f)S_f X(t_f) + \frac{1}{2} \int_{t_0}^{t_f} X^T Q X + U^T R U dt$$

$$H = \frac{1}{2} X^T Q X + \frac{1}{2} U^T R U + \frac{\partial J^*}{\partial X}[AX + BU]$$

So, to minimize $H$,

$$\frac{\partial H}{\partial U} = RU + B^T \frac{\partial J}{\partial X} = 0$$

(note that $\frac{\partial^2 H}{\partial U^2} = R$ is P.D. $\rightarrow$ global min.)

$$\Rightarrow U^* = -R^{-1} B^T \frac{\partial J}{\partial X}$$

Thus,

$$H^* = \frac{1}{2} X^T Q X + \frac{1}{2} \frac{\partial J^T}{\partial X} B R^{-1} B^T \frac{\partial J}{\partial X} + \frac{\partial J^T}{\partial X} A X - \frac{\partial J^T}{\partial X} B R^{-1} B^T \frac{\partial J}{\partial X}$$

Thus, the H-J-B equation is:

$$0 = \frac{\partial J^*}{\partial t} + \frac{1}{2} X^T Q X - \frac{1}{2} \frac{\partial J^T}{\partial X} B R^{-1} B^T \frac{\partial J}{\partial X} + \frac{\partial J^T}{\partial X} A X$$
With boundary conditions,
\[
\frac{\partial J^*}{\partial X}(X(t_f), t_f) = \frac{1}{2}X^T(t_f)S_f X(t_f)
\]
Let’s guess \( J^* = \frac{1}{2}X^T S(t) X(t) \)

Then we get,
\[
0 = \frac{1}{2}X^T \dot{S} X + \frac{1}{2}X^T Q X - \frac{1}{2}X^T S B R^{-1} B^T S X + X^T S A X
\]

let \( SA = \frac{1}{2}[SA + (SA)^T] + \frac{1}{2}[SA - (SA)^T] \)

symmetric un-symmetric

Also, \( X^T(\frac{1}{2}[SA - (SA)^T])X = 0 \)

Thus,
\[
0 = \frac{1}{2}X^T \dot{S} X + \frac{1}{2}X^T Q X - \frac{1}{2}X^T S B R^{-1} B S X + \frac{1}{2}X^T S A X + \frac{1}{2}X^T A^T S X
\]

Now, this must hold for all \( X \)

\[
0 = \dot{S} + Q - S B R^{-1} B S + S A + A^T S
\]

which is the Ricatti equation we saw before. With boundary conditions,

\( S(t_f) = S_f \)

2.5.2 Comments

- Equation 140 is another form of Pontryagin’s minimal principle.
- Involves solving partial differential equations which may not lead to closed form solution.
- For global optimality, we need:

\[
\frac{\partial^2 H}{\partial U^2} \geq 0 \Rightarrow \text{Legendre-Clebsch condition}
\]

- \( J^* \) is a necessary condition to satisfy the H-J-B equations. It can also be proved that \( J^* \) is a sufficient condition.
- Solution often requires numerical techniques or the method of characteristics.
- The H-J-B equations relate dynamic programming to variational methods.
3 Introduction to Stability Theory

3.1 Comments on Non-Linear Control

3.1.1 Design Methods

- Linearization of the plant followed by techniques like adaptive control.
- Lots of "rules of thumb", which are not well-defined.
- Not as easy as Bode, root-locus (i.e., classical linear control).
- Methods include: DP, E-L, neural and fuzzy.

3.1.2 Analysis Methods

- Many analysis methods for non-linear systems.
- Equivalent concepts of controllability and observability for non-linear systems, but very hard to prove.
- Some common ones in control: describing functions, circle theorems, Lyapunov methods, small-gain theorem, passivity (the latter two are very advanced methods).
- Most of the analysis methods are concerned with stability issues of a given system $\dot{X} = F(X,U)$.

3.2 Describing Functions

This is an old method which is not always rigorous.

- Consider systems of the following configuration.

\[ \text{NL} \rightarrow \text{G(Z)} \]

where, \( NL \) stands for simple non-linearities like any of the following:
Consider the following:

\[ \text{Let input} = A \sin \omega t. \text{ Then, output} = A_0 + \sum_{n=1}^{\infty} Y_n \sin(n \omega t + \phi_n) \text{ (sum of harmonics).} \]

Describing function \( D(A) = \frac{Y_1}{A} \frac{\phi_1}{\omega} \) fundamental of output \( \frac{\text{output}}{\text{input}} \)

**Example:** Consider \( X \) to be a saturation function of the following form:

\[
X \triangleq \begin{cases} 
+m & X > 1 \\
-kX & |X| < 1 \\
-m & X < 1 
\end{cases}
\]

So,

\[
D(A) = \begin{cases} 
\frac{2k}{\pi} \arcsin \frac{m}{Ak} + \frac{2m}{Ak} \sqrt{1 - \frac{m^2}{Ak^2}} & \frac{Ak}{m} > 1 \\
\frac{k}{\pi} & \frac{Ak}{m} \leq 1 
\end{cases}
\]
Note:
- For certain regions below saturation, there is no distortion.
- As the gain is increased, $Y_1$ decreases.

- Describing functions give us information regarding stability and limit cycles.

Steps:
1. Plot the Nyquist of $G(Z)$.
2. Plot $-\frac{1}{D(A)}$ on the same graph.

If the 2 plots intersect, then there may exist instabilities.

### 3.3 Equivalent gains and the Circle Theorem

This is a rigorous way of formalizing the describing functions $D(A)$ (developed in the '60s).
Consider the following non-linearity:

\[ \phi(x) \]

The N-L is bounded with two lines of slope \( k_1 \) and \( k_2 \), such that:

\[ k_1 \leq \frac{\phi(X)}{X} \leq k_2 \]

Hence,

\[ \phi(X) \in \text{sector}[k_1, k_2] \]

Example: quantizer

Azermans Conjecture, 1949
Consider the following system:
If this system is stable for \( k_1 \leq k \leq k_2 \), then the following system is also stable.

where, \( \phi(.,t) \in \text{sector}[k_1, k_2] \). This is actually not true.

- **Circle Theorem:** (Sandberg, Zames 1965)

  The system is stable if the Nyquist plot of \( G(z) \) does not intersect the disk at \( \frac{1}{k_1} \) and \( \frac{1}{k_2} \), for \( k_1 > 0 \) and \( k_2 < \infty \), as in the following figure.
Now, if \( k_1 = 0 \Rightarrow \frac{1}{k_1} = \infty \), then the following case would result.

A related theorem is \textbf{Popov's theorem}, for time-independent \( \phi(t) \).

### 3.4 Lyapunov's Direct Method

If \( \dot{X} = F(X,t) \)

where, \( F(0,t) = 0 \), then, is \( X = 0 \) stable at \( 0 \)?

We proceed to define stability in a \textit{Lyapunov sense}. 

178
• Find $V(X)$ such that:
  
  – $V(X)$ has continuous derivatives.
  – $V(X) \geq 0$ (P.S.D. near 0)
    
        ($V(X) \geq \alpha(\|X\|)$ for global asymptotic stability).
  
  $V(X)$ is the Lyapunov Function.

• Then, if

    $$\dot{V}(t, X) \leq 0 \; \forall t \geq t_0 \; \forall X \in B_r$$

   where, $B_r$ is a ball of radius $r$, then,

   $$\dot{X} = F(X, t)$$ is stable at 0

• The system is stable if

   $$|X_0| < \delta \Rightarrow |X| < \epsilon$$

• The system is asymptotically stable if

   $$|X_0| < \epsilon \Rightarrow X \to 0 \; \text{as} \; \epsilon \to \infty$$

• The system is globally stable if the above ”balls” go to 0.

Think of $V$ as an energy function. So, $\dot{V}$ represents the loss of energy.

Thus, if

\[
\dot{V}(t, X) < 0 \rightarrow \text{asymptotically stable} \\
\leq 0 \rightarrow \text{stable}
\]

Need La Salle's Theorems to prove asymptotic stability for the $\leq$ case.

• A common Lyapunov function is the sum of the kinetic and the potential energies as in the case of a simple pendulum.
3.4.1 Example

Let

\[ \dot{X} = -\sin X \]

Here, \( X = 0 \) is an equilibrium point. So, let

\[ V(X) = X^2 \]

\[ \Rightarrow \dot{V} = 2X \dot{X} = -2X \sin X \]

Note that the above \( V(X) \) is not a valid Lyapunov function for unlimited \( X \). So, we need to find a "ball" such that it is a Lyapunov function. But, \( \sin X \geq \frac{1}{2}X \) for \( 0 < X < C \) \( (C \) is a constant) \n\nThus,

\[ \dot{V} \leq -X^2 \text{ for } |X| \leq C \]

So,

\[ \dot{V} \leq 0 \text{ within the "ball"} \]

\[ \Rightarrow 0 \text{ is an asymptotically stable equilibrium point.} \]
3.4.2 Lyapunovs Direct Method for a Linear System

Let

$$\dot{X} = AX$$

Therefore, postulate

$$V(X) = X^T Px$$

where, $$P = P^T \geq 0$$

$$J(X_0) = \int_0^\infty X^T Q X \, dt$$

$$= X_0^T \left[ \int_0^\infty e^{AT} Q e^{At} \, dt \right] X_0$$

$$= V(X_0)$$

$$\Rightarrow \dot{V} = -X^T Q X < 0$$

where, $$Q = -A^T P + PA \Rightarrow$$ Lyapunov Equation

- If A is asymptotically stable, then for $$Q = Q^T \geq 0$$, there is a unique $$P = P^T \geq 0$$, such that,

$$A^T P + PA + Q = 0$$

- **Exercise**: Plug-in $$u = -kX$$ and show we get the algebraic Ricatti equation.

- This approach is much harder for time-varying systems.

3.4.3 Comments

- Lyapunovs theorem(s) provide lots of variations as far as some of the following parameters are concerned.
  - Stability
  - Global stability
  - Asymptotic global stability, etc.

- What about adaptive systems? An adaptive system can be written as a non-linear system of higher order.

So,

$$W_{k+1} = W_k + F(U, Y, C)$$

$$\Rightarrow$$ use Lyapunov function to determine stability.

$$Y = WU$$ or $$Y = F(X, U, W)$$

So,

\[ \dot{W} = -\mu e V = -\mu (d - F(X, U, W))X \]

which is a non-linear difference equation that is updated. The Lyapunov function gives conditions of convergence for various learning rates, \( \mu \).

- MATLAB \texttt{lyap} solves the Lyapunov equation and \texttt{dlyap} gives the solution for the discrete case.

### 3.5 Lyapunovs Indirect Method

\( \dot{X} = F(X) \) is asymptotically stable near equilibrium point \( X_e \) if the linearized system \( \dot{X}_L = DF(X_E)X_L = AX_L \) is asymptotically stable (other conditions).

#### 3.5.1 Example: Pendulum

\[ \ddot{\theta} = \tau + \sin \theta \]

The suitable lead controller would be:

\[ \tau(s) = \frac{-4s + 4}{s + 3} \theta(s) \]

In state form,

\[
\begin{align*}
X_1 & = \theta \\
X_2 & = \dot{\theta} \\
X_3 & = \text{state of the controller}
\end{align*}
\]

So,

\[
\begin{bmatrix}
\dot{X}_1 \\
\dot{X}_2 \\
\dot{X}_3
\end{bmatrix} = \begin{bmatrix}
X_2 \\
\sin X_1 - 4X_1 - 4X_3 \\
-3X_3 - 2X_1
\end{bmatrix}
\]
Linearize this at X=0.

\[ A = \begin{bmatrix} 0 & 1 & 0 \\ -3 & 0 & -4 \\ -2 & 0 & -3 \end{bmatrix} \]

So,

\[ \text{eig } A = -1, -1, -1 \Rightarrow \text{stable} \]

Hence, we have designed a lead-lag circuit which will give asymptotic stability for some region around zero. *The non-linear structure is stable as long as we start near enough \( \theta \approx 0 \).*

### 3.6 Other concepts worth mentioning

- **Small gain theorem.**

\[
\begin{align*}
H_1 &+ H_2 + U = Y_1 \\
Y_1 &- H_1 + e_1 \rightarrow Y_1 \\
Y_2 &- H_2 + e_2 \rightarrow U_2
\end{align*}
\]

A system with inputs \( U_1 \) and \( U_2 \) and outputs \( Y_1 \) and \( Y_2 \) is BIBO stable, provided \( H_1 \) and \( H_2 \) are BIBO stable and the product of the gains of \( H_1 \) and \( H_2 \) is less than unity. This is often useful for proving stability in adaptive control.

- **Passivity.**

These are more advanced and abstract topics used for proving other stability theorems. They are also useful in adaptive systems.

### 4 Stable Adaptive Control

#### 4.1 Direct Versus Indirect Control

- **Direct control**
  - Adjust control parameters directly.
  - Model reference adaptive control is a direct control method.

- **Indirect control**
  - Plant parameters are estimated on-line and control parameters are adjusted.
- Gain scheduling is an open loop indirect control method where predetermined control is switched based on current plant location.

4.2 Self-Tuning Regulators

A Control Structure
- System identification is carried out by RLS, LMS etc.
- Issues: The inputs should persistently excite the plant so that system identification is properly done.
- Control Design
  - Time varying parameters
  - PID
  - LQR
  - Kalman (If the system is linearized at each time point then it is called Extended Kalman)
  - Pole-placement
  - Predictive control
  - The design is often done directly in polynomial form rather than state space. This is because system ID (e.g. using RLS) provides transfer functions directly.
- Issues:
  - Stability
  - Robustness
  - Nonlinear issues

4.3 Model Reference Adaptive Control (MRAC)

\[
\begin{align*}
\dot{y}_p(t) & = -a_p \ y_p(t) + K_p u(t) \\
\dot{y}_m(t) & = -a_m \ y_m(t) + K_m r(t)
\end{align*}
\]
Let
\[ u(t) = c_o(t) r(t) + d_o(t) y_p(t) \]

Substitute this \( u(t) \).

Note that if
\[ c_o^* = \frac{K_m}{K_p} \quad d_o^* = \frac{a_p - a_m}{K_p} \]

Then
\[ \dot{y}_p(t) = -a_m \ y_p(t) + K_m r(t) \]

Consider Gradient descent
\[
\frac{\partial \theta}{\partial t} = -\mu \frac{\partial e_o^2(\theta)}{\partial \theta} = -2\mu c_o \frac{\partial e_o(\theta)}{\partial \theta} = -2\mu c_o \frac{\partial y_p}{\partial \theta}
\]

\[ y_p = \left( \frac{K_p}{s + a_p} \right) (d_o y_p + r c_o) \]

\[ \frac{\partial y_p}{\partial c_o} = P \ r \]
\[ \frac{\partial y_p}{\partial d_o} = P \ y_p \]

where
\[ P = \left( \frac{K_p}{s + a_p} \right) \]

\[ \text{• If we don’t know plant.} \]

- Just use proper sign of \( K_p \) and ignore \( P \) i.e. set \( P = 1 \) or
- Use current estimates of \( K_p \) and \( a_p \) (MIT Rule)

With \( P = 1 \)
\[ \Rightarrow \dot{c} = -\mu \ c \ r \]
\[ \dot{d} = -\mu \ e \ y_p \]
Is the system stable with this adaptation?

Define

\[ \dot{\phi}_r = e_o - e_o^* \]
\[ \dot{\phi}_y = d_o - d_o^* \]

Then \( e_o = y_p - y_m \)

After some algebraic manipulations

\[ \dot{e}_o = -a_m e_o + K_p (\phi_r r + \phi_y e_o + \phi_y y_m) \]
\[ \dot{\phi}_r = -\mu e_o r \]
\[ \dot{\phi}_y = -\mu e_o y_p = \mu e_o^2 - \mu e_o y_m \]

Define Lyapunov function

\[ V(e_o, \phi_r, \phi_y) = \frac{e_o^2}{2} + \frac{K_p}{2\mu} (\phi_r^2 + \phi_y^2) \]

\[ \dot{V} = -a_m e_o^2 + K_p \phi_r e_o r + K_p \phi_y e_o^2 + K_p \phi_o e_o y_m - K_p \phi_r e_o r - K_r \phi_y e_o^2 - K_p \phi_o e_o y_m \]
\[ = -a_m e_o^2 \leq 0 \]

\( \Rightarrow \) This is a stable system since \( V \) is decreasing monotonically and bounded below.

\[ \lim_{t \to \infty} V(t) \text{ as } t \to \infty \text{ exist. Thus } e_o \to 0 \text{ as } t \to \infty. \]

- Indirect method can also be used
  - Estimate \( \hat{K}_p, \hat{a}_p \) and design \( e_o \) and \( d_o \).

4.3.1 A General MRAC

Plant

\[ \hat{p}(s) = K_p \frac{\hat{n}_p(s)}{d_p(s)} = \frac{y_p(s)}{u(s)} \]

Model

\[ M(s) = K_m \frac{n_m(s)}{d_m(s)} = \frac{\hat{y}_m(s)}{\hat{r}(s)} \]
Model Reference Adaptive Control

• Controller Structure

A Control Structure

• Existence of solution

• Adaptation Rules
  
  – Many variations
    “Heuristic” Gradient descent
    Try to prove stability and convergence
  
  – Start with Lyapunov function
    Find adaptation rule.

• Issues
  
  – Relative order of systems
  
  – Partial knowledge of some parameters.
5 Robust Control

- Suppose plant $P$ belongs to some family $P$, Robust Control provides stability for every plant in $P$. (Worst case design)

**Nominal performance specifications**

\[
g(s) = \frac{g(s)k(s)}{1 + g(s)k(s)}r(s) + \frac{1}{1 + g(s)k(s)}d(s) - \frac{g(s)k(s)}{1 + g(s)k(s)}n(s)
\]

- $L = gk$ = open loop gain
- $J = 1 + gk$ = return difference
- $S = \frac{1}{1 + gk}$ = sensitivity transfer function
- $T = \frac{gk}{1 + gk}$ = complementary transfer function
- $S + T = 1$

- Command tracking
  - Want $gk$ large
  - $S$ has to be small (low frequency)

- Disturbance rejection
  - $S$ has to be small (low frequency)

- Noise suppression
  - $T$ has to be small (high frequency)
5.1 MIMO systems

\[ A = GK(I + GK)^{-1} \]

where A is Transfer-Function Matrix
\[ \sigma_i(A) = (\lambda_i(A^*A))^{\frac{1}{2}} \rightarrow \text{Singular values} \]

\[ S = (I + GK)^{-1} \quad T = GK(I + GK)^{-1} \]

- Singular plots
  - Plots of \( \lambda_i(A) \) Vs frequency, which are generalization of Bode plots.

<table>
<thead>
<tr>
<th>Loop Transfer Function Properties</th>
<th>Low Frequency</th>
<th>High Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Command Performance (r)</strong></td>
<td>(</td>
<td>gkl</td>
</tr>
<tr>
<td><strong>MIMO Case</strong></td>
<td>( \sigma(GK) &gt; 1 ) or ( \bar{\sigma}(S) &lt; 1 )</td>
<td>( \sigma(GK) &gt;&gt; 1 ) or ( \bar{\sigma}(S) &lt;&lt; 1 )</td>
</tr>
<tr>
<td><strong>Disturbance Rejection (d)</strong></td>
<td>(</td>
<td>gkl</td>
</tr>
<tr>
<td><strong>MIMO Case</strong></td>
<td>( \sigma(GK) &gt;&gt; 1 ) or ( \bar{\sigma}(S) &lt;&lt; 1 )</td>
<td>( \sigma(GK) &gt;&gt; 1 ) or ( \bar{\sigma}(S) &lt;&lt; 1 )</td>
</tr>
<tr>
<td><strong>Noise Suppression (n)</strong></td>
<td>(</td>
<td>gkl</td>
</tr>
<tr>
<td><strong>MIMO Case</strong></td>
<td>( \sigma(GK) &lt;&lt; 1 ) or ( \bar{\sigma}(T) &lt;&lt; 1 )</td>
<td>( \sigma(GK) &lt;&lt; 1 ) or ( \bar{\sigma}(T) &lt;&lt; 1 )</td>
</tr>
</tbody>
</table>

5.2 Robust Stability

- Suppose we design for \( G(s) \)
- The design is robust if the closed loop system remains stable for true plant \( \tilde{G}(s) \)

5.2.1 Small gain theorem (linear version)

\[ |G(s)K(s)| < 1 \Rightarrow \text{stability} \]

\[ |G(s)K(s)| \leq |G(s)||K(s)| \]

\( \Rightarrow \) Stability guaranteed if \( |G(s)||K(s)| < 1 \)

Instability according to Nyquist
5.2.2 Modeling Uncertainty

We use $G(s)$ for design whereas the actual plant will be $\hat{G}(s)$.

\[
\hat{G}(s) = G(s) + \delta_a(s) \rightarrow \text{Additive}
\]
\[
\hat{G}(s) = (1 + \delta_m(s))G(s) \rightarrow \text{Multiplicative}
\]

\[
G(s) = \frac{10(s+1)}{s^2(s+5)}
\]

The flexible mode is given by

\[
\text{flexible mode} = \frac{s^2 + 2(0.1)(12)s + 12^2}{s^2 + 2(0.05)(10)s + 10^2} \cdot \frac{10^2}{12^2}
\]

The true plant model, $\tilde{G}(s)$, must also include the flexible mode

\[
\tilde{G}(s) = \frac{10(s+1)}{s^2(s+5)} \cdot \frac{s^2 + 2(0.1)(12)s + 12^2}{s^2 + 2(0.05)(10)s + 10^2} \cdot \frac{10^2}{12^2}
\]

Modeling the flexible mode as additive uncertainty, we get

\[
\Delta_a(s) = \tilde{G}(s) - G(s) = \frac{-3.05(s+1)(s-2.18)}{s(s+5)(s^2+s+1)}
\]

Using the multiplicative model, we obtain

\[
\Delta_m(s) = \frac{\tilde{G}(s) - G(s)}{G(s)} = \frac{-3.05(s-2.18)s}{s^2+s+1}
\]
5.2.3 Multiplicative Error

\[ M(s) = \frac{-G(s)K(s)}{1 + G(s)(K(s))} \]

Closed system is stable if

\[ |\delta_m| < \frac{1}{|GK(1 + GK)^{-1}|} \]

\[ \Rightarrow |\delta_m| < \frac{1}{|T|} \]

- **Bound the uncertainty**

Suppose \(|\delta_m| < \gamma\)

This is stable if

\[ |T| < \frac{1}{\gamma} \]

- **Smallest destabilizing uncertainty**

  (Multiplicative Stability Margin)
\[ \text{MSM} = \frac{1}{M_r} \]

\[ M_r = \sup_w |T(jw)| = ||T||_\infty \rightarrow \text{infinity norm} \]

This represents distance to -1 point on Nyquist plot.

For MIMO

\[ \delta = \sup_w \sigma(T(jw)) = ||T||_\infty \]

For additive uncertainty

\[ |\delta_w| < \frac{1}{|KS|} \]

Additive Stability Margin is given by

\[ \text{ASM} = \frac{1}{||KS||_\infty} \]

Recall \( |T + S| = 1 \)

If we want MSM large then \( T \) has to be small. This is good for noise suppression but conflicts with tracking and disturbance.

Trade off

\[ ||\delta T||_\infty < 1 \]
\[ ||\delta KS||_\infty \leq 1 \]
\[ T + S = 1 \]
5.3 2-port formulation for control problems

\[ y_p = Gd + Gn \]
\[ u = u \]
\[ y = -Gd - n - Gu \]

Now let

\[ w = \begin{bmatrix} d \\ n \end{bmatrix}, \quad z = \begin{bmatrix} y_p \\ u \end{bmatrix} \quad \text{and} \quad P = \begin{bmatrix} P_{zw} & P_{zu} \\ P_{yw} & P_{yu} \end{bmatrix} \]

we get

\[ P_{zw} = \begin{bmatrix} G & 0 \\ 0 & 0 \end{bmatrix}, \quad P_{zu} = \begin{bmatrix} G \\ 1 \end{bmatrix}, \quad P_{yw} = \begin{bmatrix} -G & -1 \end{bmatrix}, \quad \text{and} \quad P_{yu} = -G \]
closed loop transfer function from $w$ to $z$ is:

$$T_{zw} = \begin{bmatrix} SG & -T \\ -T & -T/G \end{bmatrix}$$

where $S = \frac{GK}{1+GK}$ and $T = 1 - S$

5.4 $H_\infty$ control

- Standard problem:
  Find $K(s)$ stabilizing so that $\|T_{zw}\|_\infty \leq \gamma$

- Optimal control
  Find $K(s)$ stabilizing to minimize $\|T_{zw}\|_\infty$
  Typically solve standard problem for $\gamma$, then reduce $\gamma$ until no solution possible to get the optimal control.
  (LQG results from $\min \|T_{zw}\|_2$)

- Many assumptions like stabilizable, detectable for solution to exist.

Two port block diagram

Transfer function:

$$z = P_{zw} w + P_{zu} u$$
$$y = P_{yw} w + P_{yu} u$$
$$u = Ky$$

State-space:

$$\dot{x} = A x + B_1 w + B_2 u$$
$$z = C_1 x + D_{11} w + D_{12} u$$
$$y = C_2 x + D_{21} w + D_{22} u$$

Packed-matrix notation
\[ P(s) = \begin{bmatrix}
A & B_1 & B_2 \\
C_1 & D_{11} & D_{12} \\
C_2 & D_{21} & D_{22}
\end{bmatrix} \]

\[ H_\infty \text{ Controller} \]

\[ u = -K_c \hat{x} \]

\[ \dot{\hat{x}} = A \hat{x} + B_2 u + B_1 \hat{w} + Z_\infty K_c (y - \hat{y}) \]

where \( \hat{w} = \gamma^{-2} B_1' X_\infty \hat{x} \) and \( \hat{y} = C_2 \hat{x} + \gamma^{-2} D_{21} B_1' X_\infty \hat{x} \)

\[ K(s) = \begin{bmatrix}
A - B_2 K_c - Z_\infty K_c C_2 + \gamma^{-2} (B_1 B_1' - Z_\infty K_c D_{21} B_1') X_\infty \\
-\gamma^{-2} B_1 B_1' - B_2 \tilde{D}_{12} B_2' \\
-\gamma^{-2} C_2 C_1 - C_2 \tilde{D}_{21} C_2
\end{bmatrix}
\]

\[ Z_\infty = (I - \gamma^{-2} Y_\infty X_\infty)^{-1} \]

\[ K_c = \tilde{D}_{12} (B_2' X_\infty + D_{12}' C_1) \quad \text{where} \quad \tilde{D}_{12} = (D_{12}' D_{12})^{-1} \]

\[ K_c = (Y_\infty C_2 + B_1 D_{21}') \tilde{D}_{21} \quad \text{where} \quad \tilde{D}_{21} = (D_{21}' D_{21})^{-1} \]

\[ H_\infty \text{ constants:} \]

\[ X_\infty = \text{Ric} \left[ \begin{bmatrix} A - B_2 \tilde{D}_{12} D_{12}' C_1 & \gamma^{-2} B_1 B_1' - B_2 \tilde{D}_{12} B_2' \\ -\tilde{C}_1 \tilde{C}_1 & -(A - B_2 \tilde{D}_{12} D_{12}' C_1)' \end{bmatrix} \right] \]

\[ Y_\infty = \text{Ric} \left[ \begin{bmatrix} (A - B_1 D_{21}') \tilde{D}_{21} C_2' & \gamma^{-2} C_2 C_1 - C_2 \tilde{D}_{21} C_2 \\ -\tilde{B}_1 \tilde{B}_1' & -(A - B_1 D_{21}') \tilde{D}_{21} C_2 \end{bmatrix} \right] \]

\[ X = \text{Ric}(H), \quad H = \begin{bmatrix} A & -R \\ -Q & -A' \end{bmatrix} \quad \text{and} \quad (A - RX) \text{ is stable} \]

\[ A'X +XA -XRX +Q = 0 \]
Closed loop system:

\[
\begin{bmatrix}
\dot{x} \\
\dot{\hat{x}} \\
z
\end{bmatrix}
= \begin{bmatrix}
A & -B_2 K_r \\
Z_w K_r C_2 & A - B_2 K_r + \gamma^2 B_1 B'_1 X_w - Z_w K_r (C_2 + \gamma^2 D_{21} B'_1 X_w) \\
C_1 & -D_{12} K_r \\
C_2 & 0
\end{bmatrix}
\begin{bmatrix}
x \\
\dot{x} \\
z
\end{bmatrix}
+ \begin{bmatrix}
B_1 \\
0 \\
D_{21}
\end{bmatrix} w
\]

Necessary conditions for the above solutions are existence of Ricatti equations, and,

\[\rho(X_w Y_w) < \gamma^2\]

\(H_\infty\) Controller
5.5 Other Terminology

- Generalization to finite horizon problems

\[ \| T_{zw} \|_\infty \rightarrow \| T_{zw} \|_{[O,T]} \]

where \([O,T]\) is induced norm i.e. \(L_2[O,T]\)

This is a much harder problem to solve as it is a 2-point boundary value problem

- Loop Shaping: Design for characteristics of \(S\) and \(T\).

  - Adjust open loop gains for desired characteristics (possibly by some pre-compensators)

  - Determine \(\gamma_{\text{opt}}\) using \(H_\infty\) design and repeat if necessary.

  - May iterate (for example to reduce model order).
Part VIII
Neural Control

Neural control often provides an effective solution to hard control problems. This does not imply that more traditional methods are necessarily incapable of providing as good or even better solutions. The popularity of neural control can in part be attributed to the reaction by practitioners to the perceived excessive mathematical intensity and formalism of recent development in modern control theory. Although neural control techniques have been inappropriately used in the past, the field of neural control is now reaching a level of maturity where it is becoming more widely accepted in the controls community.

Certain aspects of neural control are especially attractive, e.g., the ability to systematically deal with non-linear systems, and simple generalizations to multi-input-multi-output problems. However, we must caution students that this field is still at its infancy. Little progress has been made to understand such issues as as controllability, observability, stability, and robustness.

1 Heuristic Neural Control

We start with some “heuristic” methods of control which utilize neural networks in conjunction with classical controllers. 1) Control by emulation, in which we simply attempt to learn an existing controller. 2) Inverse and open-loop control. 3) Feedback control, where the feedback is ignored during designing the controller. These methods, while rather ad-hoc, can be quite effective and are common in industrial applications today.

1.1 Expert emulator

If a controller exists we can try to emulate it as illustrated below

![Diagram](image)

There is no feedback and we train with standard BP. This is useful if the controller is a human and we want to automate the process.

An example of a network trained in this manner is ALVINN at CMU. The input to the controller was a 30 by 32 video input retina and the output was the steering direction. The neural controller was trained by “watching” a human drive. Fifteen shifted images per actual input were used during training to avoid learning to just drive straight. Two hundred road scenes from the past
are stored and trained on to avoid "forgetting". Fifteen of the two hundred are replaced for each new input image. Two networks were trained, one for single lane roads, and another for double lane roads. Each network was also trained as an auto associator. The network with the best L.S. reconstruction of the input image gave a confidence level to determine which network got to drive (*SHOW VIDEO).

1.2 Open loop / inverse control

![Diagram of open loop control system]

Early systems were designed open loop making the neural controller approximately the inverse of the plant: \( C \approx P^{-1} \).

Not all systems have inverses, such as the robotic arm illustrated below. (To resolve this problem, it is necessary to incorporate trajectory constraints.)

- **Inverse Kinematics**

![Diagram of inverse kinematics]

**Direct inverse modeling:** as illustrated in the figure below, one could reverse the order of plant and controller and use standard BP to train the net before copying the weights back to the actual controller. This is a standard approach for linear systems. However this is a bit dubious for non-linear systems since the plant and controller do not necessarily commute.
Indirect learning: It often makes more sense to train the direct system. Consider the uncommuted plant and controller in the previous figure. Taking the gradient of the error with respect to the weights in the neural controller:

\[
\frac{\partial e^2}{\partial w} = 2e^T \frac{\partial y}{\partial w} = 2e^T \frac{\partial y}{\partial u} \frac{\partial u}{\partial w} = 2e^T G_p G_w
\]

where \(G_p\) and \(G_w\) are the jacobians.

- We could calculate \(G_p\) directly if we had the system equations.
- We can also calculate the Jacobian by a perturbation method \(\frac{\Delta Y}{\Delta U}\). If \(U\) is dimension \(N\), Step 0: forward pass nominal \(U\). Step 1 through \(N\): forward pass \([U_1, U_2, \ldots, U_{i+\Delta}, \ldots, U_N]\).
- If \(P\) is a neural net, \(e^T G_p\) is found directly by backpropagation. To see this, we observe that

\[
\frac{\partial e^T e}{\partial x} = 2e^T \frac{\partial y}{\partial x} = 2e^T G_{xy} = \delta^0
\]

where \(\delta^0\) are found by backpropagating the error directly through the network, as can be argued from the following figure:
Thus we see backpropagation gives the product of the Jacobian and the error directly. The complexity: \( O(N \times H_1 + H_1 \times H_2 + H_2 \times M) \) is less than calculating the terms separately and then multiplying. Note, we could get the full Jacobian estimate by successively backpropagating basis vectors through model.

The conclusion is that it is useful to build a plant emulator – system ID.

We then train the controller as illustrated below:

This model is appropriate when we have a state representation of our plant. The state vector \( x \) with the reference are the inputs to a static network. If we do not have full state information,
then we might use a non-linear ARMA model (NARMA),

\[ y_k = \hat{P}[u(k), u(k-1), \ldots, u(k-m_1), y(k-1), \ldots, y(k-m_2)] \]

This may also be viewed as a simple generalization of linear zero/pole systems. With the recursion, gradient descent training is more complicated. We will explore adaptation methods later (Elman Nets just ignore the consequence of feedback).

- FIR/TDNN Net, ARMA Models etc...

We would then need to “backprop” through a dynamic system illustrated below. We will develop methods to do this later.

Often, it makes more sense to train to some “model reference” rather than a direct inverse:
1.3 Feedback control - ignoring the feedback

These methods use neural nets heuristically to improve dynamics in conjunction with “classical” methods. See Figure below. We will illustrate the approach through several examples.

1.4 Examples

1.4.1 Electric Arc furnace control

Figures 20 to 22 show the plant. Figure 23 shows flicker removal and the need for predictive control. Figure 25 shows the regulator emulator, furnace emulator and combined furnace/regulator. Figure 24 shows the plant savings. (SHOW VIDEO)

- North Star Steel Iowa
- 30 MVA transformer (50,000 amps)

Figure 20: Electric arc furnace
Figure 21: Furnace system

Figure 22: Three phase system
Figure 23: Flicker remover and predictive control

<table>
<thead>
<tr>
<th>ITEM</th>
<th>PERCENT IMPROVEMENT</th>
<th>SAVINGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electrode Savings</td>
<td>22.5%</td>
<td>$279,700</td>
</tr>
<tr>
<td>Power Savings</td>
<td>6.5%</td>
<td>$234,000</td>
</tr>
<tr>
<td>Productivity Gains</td>
<td>11.00%</td>
<td>$1,650,000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$2,163,700</td>
</tr>
</tbody>
</table>

Figure 24: Plant saving
Figure 25: Regulator emulator, furnace emulator and combined furnace/regulator
1.4.2 Steel rolling mill

The steel rolling mill is characterized by the following parameters (see Figure 26).

![Steel rolling mill diagram]

Figure 26: Steel rolling mill

- $f_w$: rolling force
- $h_e$: strip thickness at input
- $h_a$: strip thickness at output
- $l_e$, $l_a$: sensor locations
- $s_r$: roll dimension
- $v_1$: input velocity
- $v_w$: output velocity

$h_e$, $h_a$, $v_1$, $v_w$, $f_w$ are measured values. Goal: keep $h_a$ at $h_{a,ref}$ (step command).

$$ J = \int_{t=0}^{t_{max}} (h_{a,ref} - h_a(t))^2 \, dt $$

A linear PI controller is shown in Figure 27 and the system's closed-loop response in Figure 28.

Figure 29 shows the open-loop controller with a neural-net model of the plant's system inverse. The neural net implements a RBF (radial basis function) network

$$ y = \sum_{i=1}^{N} w_i G(x, \bar{x}_i, \Delta). $$

The inverse plant model is reasonable since it is a smooth static SISO mapping. Figure 30 shows the system response with this neural inverse open-loop controller. Note the poor steady-state error.

Figure ?? shows the closed-loop PI controller with a neural-net model of the plant's system inverse. The PI controller sees an approximately linear system. Figure 31 shows the system response with
Figure 27: Feedforward and feedback linear controller

Figure 28: Closed-loop response of feedforward and feedback linear PI controller

Figure 29: Open-loop controller with neural model of the plant’s system inverse
Figure 30: Response of the system with neural inverse as open-loop controller

Figure 31: Closed-loop PI controller with neural model of the plant’s system inverse

Figure 32: Response of the system with PI controller and inverse neural net model
this neural inverse closed-loop PI controller. Note the improved steady-state error.

**Internal model control (IMC):** A control structure, popular for non-linear control. See Figure 33.

- Feeding back error rather than output.
- Given plant and controller which are input/output stable, and perfect model of plant then closed loop is stable.
- Advantage: uncertainty of model is fed back to system.

\[
G(s) = \frac{G_1G_p}{1 + G_1G_p - G_1G_m}
\]

if \( G_1G_m = 1 \Rightarrow G(s) = 1 \), and zero s.s. for non-linear system. Conclusion: need inverse of model to be perfect (not of plant). (Filtering may be done additionally for high frequency removal.) Figure 34 shows the response of the system with IMC. Since \( G_1 \neq 1/G_m \) exactly there is some s.s. error.

![Neural network model of the plant](image1)

![Internal model control structure](image2)

*Figure 33: Internal model control structure*
2 Euler-Lagrange Formulation of Backpropagation

It is possible to derive backpropagation under the framework of optimal control using the Euler-Lagrange method. Referring back to the feed-forward neural network model as presented in class, we have the following formulation for the activation values

$$A(l) = f(w(l)A(l-1))$$

where $f$ is the sigmoid function and $w(l)$ is the weight matrix for layer $l$. We define $A(0)$ to be the vector of inputs, $x$, and $A(L)$ to be the vector of outputs, $Y$. (Think of the weights $w$ as the control $u$, the activation $A$ as the state, and the layer $l$ as the time index $k$.)

Recall that we desire

$$\min_w J = \sum_{k=1}^{n} \|D_k - A_k(L)\|^2$$

subject to certain constraints which in this case are given in equation 143. So adjoining equation 143 to equation 144 results in the following energy function

$$J = \sum_{k=1}^{n} \left\{ \|D_k - A_k(L)\|^2 + \sum_{l=1}^{L} \lambda_k(l)^T(A_k(l) - f(w(l)A_k(l-1))) \right\}$$

What is desired then is

$$\frac{\partial J}{\partial A_k(l)} = 0 \quad \text{and} \quad \frac{\partial J}{\partial w(l)} = 0$$

Note that $w$ is not a function of $k$. We will consider the first condition in two cases

$$\frac{\partial J}{\partial A_k(L)} \Rightarrow \lambda_k(L) = 2(D_k - A_k(L))$$

$$\frac{\partial J}{\partial A_k(l)} \Rightarrow \lambda_k(l) = w^T(l+1)\nabla f(w(l+1)A_k(l))\lambda_k(l+1)$$

and we will make the definition

$$\delta_k(l) \equiv \nabla f(w(l)A_k(l-1))\lambda_k(l) = f'(l)\lambda_k(l)$$

213
So we have

\[ \delta_k(L) = 2f_k'(L)(D_k - A_k(L)) \quad \text{and} \]
\[ \delta_k(l) = f_k'(l)w_T(l + 1)\delta_k(l + 1) \]

which we recognize from before as backpropagation. Now we need an update rule for the weights

\[ \frac{\partial J}{\partial w(l)} = 0 = \sum_{k=1}^{n} f_k'(l)\lambda_k(l)A_k^T(l - 1) \]
\[ = \sum_{k=1}^{n} \delta_k(l)A_k^T(l - 1) \]

which can be minimized using stochastic gradient descent. The update rule is then

\[ w_k(l) \leftarrow w_k(l) - \mu\delta_k(l)A_k^T(l - 1) \]

### 3 Neural Feedback Control

Incorporating feedback, neural system can be trained with similar objectives and cost functions (LQR, minimum-time, model-reference, etc.) as classical control but solutions are iterative and approximate. “Optimal control” techniques can be applied which are constrained to be approximate by nature of training and network architectural constraints.

A typical controller implementation for a model reference approach is shown below.
In order to understand how to adapt such structures, we must first review some algorithms for training networks with feedback.

### 3.1 Recurrent neural networks

We provide some review of the more relevant recurrent structures and algorithms for control.

The above figure represents a generic network with feedback expressed as:

\[ x(k) = N(x(k-1), r(k-1), W) \]

The network may be a layered structure in which case we have output feedback. If there is only one layer in the network then this structure is equivalent to a fully recurrent network.

If there is a desired response at each time step, then the easiest way to train this structure is to feedback the desired response \( d \) instead of \( x \) at each iteration. This effectively removes the feedback and standard BP can be used. Such an approach is referred to as “teacher forcing” in the neural network literature or “equation error” adaptation in adaptive signal processing. While simple, this may lead to a solution which is biased if there is noise in the output. Furthermore, in controls, often a feedback connection represents some internal state which we may not have a desired response for. In such case, we must use more elaborate training methods:

#### 3.1.1 Real Time Recurrent Learning (RTRL)

RTRL was first presented by Williams and Zipser (1989) in the context of fully recurrent networks. It is actually a simple extension of an adaptive algorithm for linear IIR filters developed by White in 1975. We present a derivation here for the general architecture shown above.

We wish to minimize the cost function:

\[ J = \frac{1}{2} E \left[ \sum_{k=1}^{K} e^T(k)e(k) \right] \]

Where the expectation is over sequences of the form

\[
\begin{align*}
& (x(0), d(0)), \ (x(1), d(1)), \ \ldots, \ (x(k), d(k)) \\
& (x_2(0), d_2(0)), \ (x_2(1), d_2(1)), \ \ldots, \ (x_2(k), d_2(k)) \\
& (x_3(0), d_3(0)), \ \ldots \ \ (x_3(k), d_3(k)) \\
& \vdots \ \ \ \ \ \ \ \ \vdots \\
& (x_p(0), d_p(0)), \ \ldots \ \ (x_p(k), d_p(k))
\end{align*}
\]
Define
\[ e(k) = \begin{cases} 
  d(k) - x(k) & \text{if } \exists d(k) \\
  0 & \text{else}
\end{cases} \]
(some outputs may have desired responses; others governed by internal states.) The update rule is
\[ \Delta W = -\mu \frac{d e^T(k)e(k)}{dW} = 2\mu e^T(k) \frac{dx(k)}{dW} \]
Then
\[ \frac{dx(k)}{dW} = \frac{\partial N(k)}{\partial x(k-1)} \frac{dx(k-1)}{dW} + \frac{\partial N(k)}{\partial r(k-1)} \frac{dr(k-1)}{dW} + \frac{\partial N(k)}{\partial W} \frac{dW}{\Lambda(k)} \]
so that
\[ \Lambda^{rec}(k) = G_x(k)\Lambda^{rec}(k-1) + \Lambda(k) \]
with
\[ G_x = \frac{\partial N(k)}{\partial x(k-1)} \in \mathcal{R}^{N_x \times N_x} \]
and
\[ \Lambda = \frac{\partial N(k)}{\partial W} \in \mathcal{R}^{N_x \times N_W} \]
Then the update rule becomes
\[ \Delta W = 2\mu e^T(k)\Lambda^{rec}(k). \]
Note that this is \( O(N^2M) \) for \( M \) weights and \( N \) outputs. For a fully recurrent network this is \( O(N^4) \), which is not very efficient.

To calculate the necessary Jacobian terms, we observe that,
\[ \frac{dJ(k)}{dW} = \frac{\partial J^T(k) dN(k)}{\partial N(k)} \frac{dN(k)}{dW} = -e^T \frac{dN(k)}{dW} \]
\[ \frac{dJ(k)}{dx} = \frac{\partial J^T(k) dN(k)}{\partial N(k)} \frac{dN(k)}{dx} = -e^T \frac{dN(k)}{dx} \]
These are found by a single backpropagation of the error. To get \( \frac{dN(k)}{dW} \) and \( \frac{dN(k)}{dx(k)} \), backprop unit vectors \( e = [0\ldots010\ldots0]^T \).

### 3.1.2 Dynamic BP

This method by Narendra is an extension of RTTL.

![Dynamic BP Diagram](figure.png)
where $W(z)$ is a Linear Operator (we should really use the notation $W(q)$).

Example 1: Full Feedback:

$$X(k - 1) = W(z)X(k)$$ \hspace{1cm} (145)

$$\Rightarrow \begin{bmatrix} (z^{-1}) & 0 & 0 & 0 \\ 0 & (z^{-1}) & 0 & 0 \\ 0 & 0 & (z^{-1}) & 0 \\ 0 & 0 & 0 & (z^{-1}) \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix}$$

Example 2: Tap delay line on $X_1$.

$$\Rightarrow \begin{bmatrix} (z^{-1}) + (z^{-2}) + (z^{-3}) + (z^{-4}) & 0 & \ldots & \ldots \\ 0 & 0 & \ldots & \ldots \\ 0 & 0 & \ldots & \ldots \\ 0 & 0 & \ldots & \ldots \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix}$$

The equation for DBP is the following modification of RTRL:

$$\Lambda^{tec} = \frac{dx(k+1)}{dW}$$ \hspace{1cm} (146)

$$\Lambda^{tec}(k) = G_x(k)W(z)\Lambda^{tec}(k) + \Lambda(k)$$ \hspace{1cm} (147)

(Note the mix of time-domain and “frequency” domain operators)

### 3.2 BPTT - Back Propagation through Time

(Werbos, Rumelhart et al, Jordan, Nguyen,...)

- Just unravel in time to get one large network.
- Backprop through the entire structure.
• Error may be introduced at final time step or at intermediate time steps
• Store $\Delta w$ for each stage $\rightarrow$ average all of them together to get a single $\Delta w$ for the entire training trajectory.

3.2.1 Derivation by Lagrange Multipliers

We present a derivation of BPTT by Lagrange Multipliers, which is a common tool for solving N-stage optimal control problems.

We want to

$$
\min_W \frac{1}{2} \sum_{k=1}^{N} (d_k - X_k)^T (d_k - X_k)
$$

Subject to

$$
X(k+1) = N(X(k), r(k), W)
$$

Note that $W$ acts like the control input. Next, simply adjoin the constraint to get the modified cost function:

$$
J = \sum_{k=1}^{N} \frac{1}{2} (d_k - X_k)^T (d_k - X_k) + \sum_{k=0}^{N-1} \lambda_{k+1}^T [N(k) - X_{k+1}]
$$

$$
= \sum_{k=1}^{N-1} \left( \frac{1}{2} |d_k - X_k|^2 + \lambda_{k+1} N(k) - \lambda_k X_k \right) + \frac{1}{2} |d_N - X_N|^2 + \lambda_N^T N(0) + \lambda_N^T X_N
$$

The first constraint is that

$$
\frac{\partial J}{\partial X(k)} = 0
$$

For $k = N$, we get

$$
\lambda_N = -(d_N - X_N)^T \quad \text{Terminal Condition,}
$$

For $k \neq N$, we get

$$
\lambda_k = -(d_k - X_k)^T + \lambda_{k+1}^T \frac{\partial N(k)}{\partial X(k)}
$$

which is the BPTT equation that specifies to combine the current error with the error propagated through the network at time $k+1$.

The second optimality condition:

$$
\frac{\partial J}{\partial W} = \sum_{k=0}^{N-1} \lambda_{k+1}^T \frac{\partial N(k)}{\partial W} = 0,
$$

which gives the weight update formula, and explicitly says to change $\Delta W$ by averaging over all time as we argued before.

Variations

• N-Stage BPTT. Just backprop a fixed $N$ stages back. This allows for an on-line algorithm. (If $N$ is large enough, then gradient estimate will be sufficient for convergence).
• 1-Stage BPTT. In other words, just ignore the feedback as we saw last lecture.
3.2.2 Diagrammatic Derivation of Gradient Algorithms:

The adjoint system resulting from the Lagrange approach specifies an algorithm for gradient calculations. Both the adjoint system and algorithm can be described using flow diagrams. It is possible to circumvent all tedious algebra and go directly from a diagram of the neural network of interest to the adjoint system. This method provides a simple framework for deriving BP, BPTT, and a variety of other algorithms. Reference: Wan and Beaufays. (SHOW SLIDES)

3.2.3 How do RTRL & BPTT Relate

- Both minimize same cost function.
- They differ by when weights are actually updated.
- We can use flow graph methods to relate the algorithms directly and argue that they are simply “transposes” of each other. Reference: Beaufays & Wan, 1994 - Neural Computation. Other examples like this include emitting and receiving antennas, controller and observer canonical forms, decimation in time/decimation in frequency for the FFT.

Misc. RTRL and BPTT are gradient based methods. Many second-order methods also exist which try to estimate the Hessian in some way. Most ignore the feedback when actually estimating the Hessian. A popular method with controls used the extended decoupled Kalman filter (EDKF) (Feldkamp).

4 Training networks for Feedback Control

Returning to feedback control, we can now apply RTRL (or DBP), or BPTT to train the control system. The indirect nature of the problem (where a plant model is required) is illustrated in the figures.

For example, let's again assume availability of state information $x$:

If we assume we know the Jacobian for the plant or have trained a neural network emulator, the system may be thought of as one big recurrent network which may be trained using either RTRL or BPTT. For BPTT we visualize this as unfolding the network in time.
we simply treat every block as a Neural Network and backprop all the errors. If we have a model with the output as a function of the states, \( y = h(X) \):

For more complicated structures such NARMA models we still apply either BPTT or RTRL. For example, Feldkamp’s control structure uses a feedforward network with a tap delay line feed back from the output along with locally recurrent connections at each layer.

### 4.1 Summary of methods for desired response

Regulators (steady-state controllers):

- Maintain system about reference using acceptable control
- Control task has no final time-step
- Focus of theoretical work in neural networks (Narendra ’90)
1. Model Reference (as seen before)

\[ J = \min \sum ||d_k - X_k||^2 \]

Where \( d_k \) come from some plausible model that we would like the system to emulate (e.g., a 2nd order linear model).

2. “Linear” Quadratic Regulator

\[ J = \min \sum \left( X_k^T Q_k X_k + U_k^T R_k U_k \right) \]

Effectively \( d_k = 0 \) or some constant, and we weight in control effort. To modify either RTRL or BPTT, simply view \( U \) and \( X \) as outputs:

\[ \frac{\partial J}{\partial U_k} = 2(U_k)^T R \rightarrow (error \ for \ U) \]

\[ \frac{\partial J}{\partial X_k} = 2(X_k)^T Q \rightarrow (error \ for \ X) \]
Final Time Problems (Terminal Control)

1. Constraint is only at final time. Examples: Robotic arm, Truck backing.

\[ J = \min \| d_N - X_N \|^2 \]

although one could add additional trajectory constraints (e.g. smoothness).

4.1.1 Video demos of Broom balancing and Truck backing

Comments:

<table>
<thead>
<tr>
<th></th>
<th>Network Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single broom</td>
<td>4-1</td>
</tr>
<tr>
<td>Double broom</td>
<td>6-10-1</td>
</tr>
<tr>
<td>Single Trailer</td>
<td>4-25-1</td>
</tr>
<tr>
<td>Double Trailer</td>
<td>5-25-1</td>
</tr>
</tbody>
</table>
Example: Truck Backing

Problem Formulation

- Trailer-truck with multiple trailers must be backed to loading dock

Truck Backing

Controller Training

- Train to back from progressively more difficult configurations
- Stop forward run where truck crosses dock
Double-Broom

\[ (\theta_1, \dot{\theta}_1) \]

force

\[ (x, \dot{x}) \]

Double-Trailer Truck

first trailer angle

second trailer angle

cab angle

DOCK
Example: Navigation and Obstacle Avoidance

- Control steering angle of the truck
- Back to goal around field of known obstacles

Navigation and Control Structure

- Processor array computes a navigational field using an iterative relaxation process
- Feedforward state-feedback controller uses field to control steering
Truck-Backing Results

Truck and Feedforward Controller

\[ \Delta S_j = s_i - s_c \]

Feedback control

Local state information
Sample Obstacle Fields

Barrier Potential Field

Navigational Field

Example: Automotive Suspension Control

Problem Formulation  (Ford Motor Company)

GOAL
small RMS body deflection (smooth ride)
small tire deflection  (good handling)

PARAMETERS
body weight
wheel weight
spring constant
tire spring constant
spring damping
tire damping
time step
gravity

INPUTS
road disturbance
actuator force

STATES
body momentum
spring deflection
wheel momentum
tire deflection
Automotive Suspension Control

Control Structure

Automotive Suspension Control

Sample Results

- Actuator Force (lb)
- Road Input (in/s)
- Body Acceleration (in/s^2)
- Tire Deflection (in)

Number of Time Steps (5 ms/step)
4.1.2 Other Misc. Topics

- Obstacle avoidance and navigation. See slides and video.
- TOBPTT (time optimal): In above problems, the final time \( N \) is unknown (or hand picked). We can also try to optimize over \( N \).

\[
J = \min_{W, N(X_0)} E[\phi(X, N) + \sum_{k=0}^{N(X_0)} L(X(k), U(k))]
\]

where \( \phi(X, N) \) is the cost of final time. The basic idea is to iteratively forward sweep, followed by a backward error sweep, up to time a time \( N \), which minimizes above cost for current (Plumer, 1993).

- Model Predictive control with neural networks also possible. (Pavillion uses this for process control, also Mozer’s energy control.)

4.1.3 Direct Stable Adaptive Control

Uses Radial Basis functions and Lyapunov method to adapt controller network without need for plant model (Sanner and Slotine, 1992). This is very good work, and should be expanded on in future notes.

4.1.4 Comments on Controllability / Observability / Stability

Controllability refers to the ability to drive the states of the systems from arbitrary initial conditions to arbitrary final states using the control input. In linear systems, this implies state feedback is capable of arbitrarily modifying the dynamics.

Observability refers to the ability to determine the internal states of the system given only measurements of the control signal and the measures plant output. For a linear system, this implies we can estimate the states necessary for feedback control.

For a nonlinear systems:

- The System \( X(k+1) = f(X(k), U(k)) \) is controllable in some region of an equilibrium point, if the linearized system is controllable.

- The same holds for observability.

Global controllability and observability for nonlinear systems are extremely difficult prove. Proofs do not currently exist that neural networks are capable of learning an optimal control scheme, assuming controllability, or are capable of implementing global observers.

General proofs of stability, robustness, etc. are still lacking for neural control schemes.
5 Reinforcement Learning

The name “Reinforcement learning” comes from animal learning theory and refers to “occurrence of an event in proper relation to a response, that tends to increase the probability that the response will occur again in the same situation”.

We will take two different cracks at presenting this material. The first takes an historical approach and then arrives at the dynamic programming perspective. Second, we will come back for a more detailed coverage taking “approximate” dynamic programming as a starting place.

Learning can be classified as:

- **Supervised Learning**:
  - output evaluation provided
  - desired response encoded in cost function.

- **Unsupervised Learning**:
  - no desired response given
  - no evaluation of output provided.

- **Reinforcement Learning**:
  - no explicit desired response is given
  - output coarsely evaluated (reward / punish).

Examples requiring reinforcement:

- Playing checkers (win/lose)
- Broom balancing (crashed/balanced)
- Vehicle Navigation (ok/collision/docked).

Note we already showed how to solve some of these problems using BPTT based on the Euler-Lagrange approach. Reinforcement Learning is just another approach (based on DP).

Exploration and Reinforcement

- In reinforcement learning, system must discover which outputs are best by trying different actions and observing resulting sequence of rewards.

- Exploration vs Exploitation dilemma:
  - **Exploitation** use current knowledge about environment and perform best action currently known
  - **Exploration** explore possible actions hoping to find a new, more optimal strategy.

- Exploration requires randomness in chosen action.
Adding randomness to a system (see Figure 35 and 36). For those figures: $Pr\{y_i = 1|w^i, x^i\} = P_i$.

Method 3 - Ising model

\[
\begin{align*}
y_i &= \pm 1, \\
s_i &= \sum w_{ij} x_j, \\
Pr\{y_i = 1\} &= \frac{1}{1 + \exp(-2\beta s_i)}, \\
E[y_i] &= \tanh(\beta s_i),
\end{align*}
\]

where $\beta$ is the temperature parameter.

We will see that the current view of reinforcement relates to approximate dynamic programming. While true DP explicitly explores all possible states of the system, RL must use randomness to “explore” some region around current location in state space. Sometimes strict adhesion to this principle makes the job more difficult than it really is.
Basic reinforcement structure

Classes of Reinforcement Problems:

1. Unique credit assignment
   - one action output
   - reinforcement given at every step
   - reinforcement independent of earlier actions and states
   - state independent of earlier states and actions

2. Structural credit assignment
   - like (1) but involves multiple action units
   - how to distribute credit between multiple actions

3. Temporal credit assignment (game playing, control)
   - reinforcement not given on every step
   - state dependent on earlier states and actions
   - how to distribute credit across sequence of actions

Class (I), (II) are easy. Class (III) is more challenging, and is what current research deals with.

Class(I) - Simple credit assignment Example:

1. Example: Adaptive Equalizers (see Figure below)

- “reward” every decision

\[
\Delta w = \mu \delta_k x(k-1),
\]

and \( \delta_k = \text{sign}(s_k) - s_k \)
• "reinforces" response for next time
• if we start out fair response, this method continues to get better.

Reward-Penalty:

\[
\text{"psuedo" desired } d_i(t) = \begin{cases} 
  y_i(t) & \text{if } r(t) = +1 \\
  -y_i(t) & \text{if } r(t) = -1 
\end{cases}
\]

\[
\text{define error } = y_i(t) - <y_i(t)>
\]

Here \(<y_i(t)\) is the expectation for stochastic units and \(P_i\) for Bernoulli units. This says to make the current output more like the average output. Then we have

\[
\Delta w = \mu r(y_i - <y_i(t)>)X
\]


• Temporal problem but temporal credit problem ignored
• Also called "selective boot strap"
• Particular encoding makes optimal strategy linearly separable
• Punish or reward all decisions equally
• Weak punish / strong reward -> better convergence
• Works because wins tend to result from many good decisions.

Class(II) - Structural Credit Assignment.
• Train a model to predict $E[r]$ for given $(x,a)$
• Backprop through environment model to get action errors
• Like plant model in control

Class(III) - Temporal Credit Assignment.

Adaptive Heuristic Critic

Critic synthesizes past reinforcements to provide current heuristic reinforcement signal to actor.

Samuel's checker programs:

• Used a method to improve a heuristic evaluation function of the game of checkers during play
• Best evaluation for a given move $\rightarrow$ win/lose sequence used to determine next move
• Updated board evaluations by comparing an evaluation of the current board position with an evaluation of a board position likely to arise later in the game (make current evaluation look like final outcome - basic temporal credit assignment)
• Improved to evaluate long term consequences of moves (tries to become optimal evaluation function)
• In one version, evaluation was a weighted sum of numerical features, update based on error between current evaluation and predicted board position.

5.1 Temporal Credit Assignment

5.1.1 Adaptive Critic (Actor - Critic)

• Adaptive critic seeks to estimate a measure of discounted, cumulative future reinforcement.
• Many problems only involve reinforcement after many decisions are made.
• Critic provides a step by step heuristic reinforcement signal to action network at each time step.
• The actor is just the RL terminology for the controller.
5.1.2 TD algorithm

Re-popularized reinforcement learning. (1983 - Sutton, Barto, Anderson)

ACE (Adaptive Critic Element) or called ASE (Associative Search Element), which is the controller.

\[ V(t_0) = E \left\{ \sum_{t=0}^{\infty} \gamma^t r(t) \mid w, x(t) \right\} \]

This gives the overall utility of state \( x(t) \) given current actor \( W \) with action \( a = N(x, W) \), where \( N \) may be a Neural Network actor). (Notation: \( V = J \) in Chapter on DP)

Critic

1. \( V(t) \) is unknown. The critic builds an estimate \( \hat{V}(t) \) by interaction with the environment.

2. The critic also provide an immediate heuristic reinforcement to the actor.

\[ \rho(t + 1) = r(t + 1) + \gamma \hat{V}(t + 1) - \hat{V}(t) \]

This is immediate reinforcement with temporal credit assignment. Discounted immediate return by taking action \( a \).

- if our action takes us to a better utility then reward.
- otherwise punish.

(Along optimal \( \rho(t + 1) = 0 \), assuming no randomness).

Actor Weight Update (controller)

Define a “weight eligibility” trace.
\( e(t) \) (not error) - analogous to \((d - y)x\) in LMS (\( y \) is the actor \( a \)).
But don’t have \( d \), use error as before.

\[ e(t) = (a(t) - < a(t) >)x \]

Averaged eligibility trace

\[ \tau(t) = \beta \tau(t) + (1 - \beta) e(t) \]

\[ \Delta W_a(t + 1) = \mu_a \rho(t + 1) \tau_{ij}(t) \]

(Similar form to immediate reinforcement seen before).
Comment
If using Net, we need to Backprop terms through network to maintain eligibility traces for each weight vector.

Where to get \( \hat{V} \) - train critic

TD method

\[
\hat{V} = ACE(x(t), W_c(t))
\]

where the ACE may be a Neural Net or table lookup (box-representation). Tries to predict \( V^* \) based on only the current state.

Consider finding \( W_c \) to minimize \( (V - \hat{V})^2 \)

\[
\Delta W_c = \mu_c (V - \hat{V}) \sum_w \hat{V} \quad \text{BP term}
\]

where \( V \) is the actual future utility.
But we still need the true \( V \).

Total weight update is

\[
\Delta W = \sum_{i=1}^{m} \Delta W_i = \sum_{i=1}^{m} \mu_i (V - \hat{V}_i) \nabla_w \hat{V}_i
\]

But,

\[
V - \hat{V}_i = \sum_{k=i}^{m} (\hat{V}_{k+1} - \hat{V}_k)
\]

Assuming \( \hat{V}_{m+1} = V \)

(sum of individual differences).

\[
\Delta W = \sum_{i} \mu (\sum_{k} \hat{V}_{k+1} - \hat{V}_k) \nabla_w \hat{V}_i
\]

\[
= \sum_{k=1}^{m} \mu \sum_{i=1}^{m} (\hat{V}_{k+1} - \hat{V}_k) \nabla_w \hat{V}_i
\]

Rearranging terms and changing indices \( (k \leftrightarrow t) \),

\[
\Delta W = \sum_{i=0}^{m} \mu (\hat{V}_{i+1} - \hat{V}_i) \sum_{k=1}^{i} \nabla_w \hat{V}_k
\]

sum of all previous gradient vectors.
But this can be done incrementally.

\[ \Delta W_t = \mu (\hat{V}_{t+1} - \hat{V}_t) \sum_{k=1}^{t} \nabla_w \hat{V}_k \]

Variations -

- Discount previous gradient estimates (like RLS).

\[
\sum \nabla_w \hat{V}_k \rightarrow \sum_{k=1}^{t} \lambda^{t-k} \nabla_w \hat{V}_k
\]

implemented as

\[
\varepsilon_t = \nabla_w \hat{V}_t + \lambda \varepsilon_{t-1} \quad \varepsilon_0 = 0
\]

(TD(\lambda))

Here's a second (easier) argument for TD(\lambda):

- Assume discounted future sum

\[
V \rightarrow V(t) = \sum_{k=0}^{\infty} \gamma^t r(t + k + 1)
\]

\[
V(t) - \hat{V}(t) = r_{t+1} + \gamma V(t + 1) - \hat{V}(t)
\]

Assume \( V = \hat{V}(t + 1) \)

\[
= r_{t+1} + \gamma \hat{V}(t + 1) - \hat{V}(t) = \rho(t + 1)
\]

Summarizing, the TD algorithm is expressed as:

\[
\Delta W_t \text{ critic } = \mu_c \rho(t + 1) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w \hat{V}_k \rightarrow ACE
\]

\[
\Delta W_t \text{ actor } = \mu_a \rho(t + 1) \hat{\tau}(t) \rightarrow ASE
\]

Alternative argument: by Bellman's optimality condition,

\[
V^*(t) = \max_{a(t)} \{ r(t + 1) + \gamma V^*(t + 1) \}
\]

Let \( V^* \rightarrow \hat{V} \) and ignore probability.

Error for critic with output \( \hat{V} \) is

\[
r(t + 1) + \gamma \hat{V}(t + 1) - \hat{V}(t) = \rho(t + 1)
\]
5.2 Broom Balancing using Adaptive Critic

1983 Barto, Sutton, Anderson.


Using an ML net to produce non-linear critic.

Quantizing input space and - use 0/1 representation for each bin. Binary inputs combined using single layer.
\[
\text{action } a = \begin{cases} 
+1 & : \text{left 10 Newton} \\
-1 & : \text{right 10 Newton}
\end{cases}
\]

\[
\text{reward } r = \begin{cases} 
-1 & : \text{punish if } |\theta| > 12 \text{ or } |\dot{h}| > 2.4m \\
-1 & : \text{nothing otherwise}
\end{cases}
\]

- output neurons are Bernoulli initialized to \( P_i = 0 \) (W=0)
- "Backprop" used to update hidden units. Uses \( \delta \) computed at output based on pseudo-targets. 1983 version used single layer (1 neuron).
- trained on 10,000 balancing cycles.
- instead of always starting from \( \theta = 0, \dot{h} = 0 \), each run was initialized to random position. This ensures rich set of training positions.

\[\text{Fig. 8. Output of action network.}\]
5.3 Dynamic Programming Perspective

\[ V(t) = E \left\{ \sum_{t=0}^{\infty} \gamma^t r(x(t), a(t)) \mid x_0 = x(t), W \right\} \]

where \( W \) represents the current actor parameters.

Bellman’s Optimality Equation from Dynamic Programming.

\[ V^*(x) = \max_{a(t)} \left[ r(x(t), a(t)) + \gamma \sum_{x(t+1)} \text{Prob}(x(t+1) \mid x(t), a(t)) V^*(x(t+1)) \right] \]

Note

\[ V^*(x) = \max_{W} V(x, W) \]

only if the model we choose to parameterize \( V \) by is powerful enough.

Aside:

\( V^* \rightarrow \hat{V} \), ignore probability, assume actor chooses a “good” action -

\[ r(t+1) + \gamma \hat{V}(t+1) - \hat{V}(t) = \rho(t+1) \]

which is the immediate reinforcement signal for the controller, Alternatively, we can solve directly for the control action by:

\[ a^* = \arg \max_{a(t)} \left[ r(x_t, a_t) + \gamma \sum P(x_{t+1} \mid x_t, a_t) V^*(x_{t+1}) \right] \]

Difficulty is that \( P \) as well as \( V^* \) must be known (need some model to do this correctly).

We thus see that we don’t really need a network for the controller. However, the role of actor to do exploration. Also, it may be advantageous during training for helping “smooth” the search,
and for providing “reasonable” control constrained by the parameterization of the controller.

_Policy Iteration_ - a parameterized controller is maintained.
_Value Iteration_ - Only the “value” function (critic) is learned, and the control is inferred from it.

### 5.3.1 Heuristic DP (HDP)

(Werbos and others)

Note that we only use critic to get $\frac{d\hat{V}}{dx_k}$ to actually update networks.

_Dual Heuristic Programming (DHP)_ -

Updates an estimate

$$\hat{\lambda}(x, w) = \frac{d\hat{V}}{dx_k}$$

directly.
5.4 Q-Learning

(Watkins)

Notation: $V^W$ or $Q^W \rightarrow$ where $W$ is the current critics parameters.

We showed,

$$V^* = \max_{a_t} \left[ r(x_t, a_t) + \gamma \sum_{x_{t+1}} \Pr(x(t+1) | x_t, a_t)V^*(x(t+1)) \right]$$

Define

$$Q^W(x_t, a_t) = r(x_t, a_t) + \gamma \sum_{x_{t+1}} P(x_{t+1} | x_t, a_t)V^W(x_{t+1})$$

$$V^* = \max_a [Q^*(x, a)]$$

Bellman’s Equation in $Q^*$ alone

$$Q^*(x_t, a_t) = r(x_t, a_t) + \gamma \sum_{a_{t+1}} P(x(t+1) | x_t, a_t) \left( \max_{a_{t+1}} Q^*(x_{t+1}, a_{t+1}) \right)$$

Note - max is only on the last term, thus given

$$a^* = \arg \max_a [Q^*(x, a)]$$

without the need for a model!

Q-Learning Algorithm:

$$\hat{Q}(x_t, a_t) \leftarrow \hat{Q}(x_t, a_t) (1 - \beta_t) + \beta_t \left[ r_t + \gamma \max_{a_{t+1}} \hat{Q}(x_{t+1}, a_{t+1}) \right]$$

where

$$0 < \beta_t < 1, \quad \sum_{t=1}^{\infty} \beta_t = \infty, \quad \beta_t^2 < \infty$$

$\hat{Q} \rightarrow Q$ with probability 1 if all pairs are tried infinitely often.

Look up table - grows exponentially in size.

1. choose action $a$ (with some exploration) and $x_{t+1}$ is resulting state.
2. update $\hat{Q}$
3. repeat

final controller action is $\max_a \hat{Q}(x, a)$ or an approximation.

A Neural Network may also be used to approximate the controller:

```
\begin{tikzpicture}
    \node (a) {a};
    \node (x) {x} [above=of a];
    \node (NN) at (0,0) {NN};
    \node (Q) at (1,0) {$\hat{Q}$};
    \draw [->] (a) -- (NN);
    \draw [->] (x) -- (NN);
    \draw [->] (NN) -- (Q);
\end{tikzpicture}
```

242
\[ \dot{Q}_{dc} = r_t + \gamma \max_{a_{t+1}} \dot{Q}(x_{t+1}, a_{t+1}) \]

No proof for convergence.

A related algorithm is Action Dependent HDP (ADHDP) - Werbos. Uses action network output as best \(a_{t+1}\) - just like HDP but critic also has \(a_t\) as input.

### 5.5 Model Dependant Variations

TD, Q-Learning are model free, i.e. they don’t use \(Prob(x_{t+1} \mid x_t, a_t)\).

Several methods exist which explicitly use knowledge of a model. These are referred to as “Real Time DP”.

Example: Backgammon - Sutton’s “Dyna” architecture (could also apply it to Broom Balancing).

Methods are based on approximation:

\[ V^*_t(x_{t+1}) = \max_a \left[ r(x_t, a_t) + \gamma \sum_{x_{t+1}} Prob(x(t + 1) \mid x_t, a_t) V^*_t(x_{t+1}) \right] \]

(with some reasonable restrictions based on current states).

\(V^*_t\) is iteratively updated in this manner.

(TD and Q-Learning can be shown to be approximations to the RTDP methods).

### 5.6 Relating BPTT and Reinforcement Learning

We have seen two approaches to neural control problems which optimize a cost function over time, BPTT and RL. How are these related? BPTT is based on the calculus of variations or Euler-Lagrange approach, while RL is based on DP. EL and DP were our two main frameworks for solving optimal control problems. Recall also the relation-ship derived using the HJB equations. The influence function \(\lambda(k) = \partial J^*/\partial u(k)\), or equivalently \(\delta(k) = \partial V^*/\partial u(k)\). In words, the delta terms found with backpropagation-through-time can be equated to the partial derivative of the value function.

### 5.7 LQR and Reinforcement Learning

RL has the reputation of often being excruciatingly slow. However, for some problems we can help the approach out by knowing something about the structure of the value function \(V\). For example, in LQR we have shown a number of times that the optimal LQ cost is \textit{quadratic} in the states. In
fact the V function is simply the solution of a Ricatti equation. However, rather than algebraically solving for the Ricatti equation we can use RL approaches to learn the quadratic function. A nice illustration of this is given by Stefan Schaal at Georgia Tech., where learning pendulum balancing is achieved after a single demonstration (SHOW TAPE on Anthropomorphic Robot Experiments).

6 Reinforcement Learning II

(Alex T. Nelson)

6.1 Review

Immediate RL: $r(x_t, a_t)$ is an evaluation (or reinforcement) of action $a_t$ and state $x_t$, and is available at time $t$.

Delayed RL: a single value of $r$ is available for an entire sequence of actions and states $\{(x_t, a_t)\}$

Typical Delayed RL methods maintain an approximation, $\hat{V}$ of the optimal value function $V^*$, and use Immediate RL methods as follows: If $a : x \mapsto y$, then the difference, $\hat{V}(y) - \hat{V}(x)$ provides an immediate evaluation of the pair $(x,a)$.

6.2 Immediate RL

We want to find parameters $w$ such that:

$$r(x, \hat{\pi}(x; w)) = r^*(x) \quad \forall x \in \mathcal{X},$$

where $r^*(x) = \max_a r(x, a)$ is the reinforcement for the optimal action at state $x$, and where $\hat{\pi}(x; w)$ represents the action policy parameterized by $w$. In other words, we want to find a policy which will maximize the reinforcement.

Unlike supervised learning, we don’t have directed error information:

$$\eta = \hat{\pi}(x; w) - \pi^*(x)$$

So …

1. Have to obtain weight change some other way.

2. Also, we don’t know when we’ve converged to the correct mapping.

6.2.1 Finite Action Set

$\mathcal{A} = \{a_1, a_2, a_3, \ldots, a_m\}$

Devise two functions, $g(\cdot) : \mathcal{X} \mapsto \mathbb{R}^m$, and $M(\cdot) : \mathbb{R}^m \mapsto \mathcal{A}$.

- $g(\cdot)$ produces a merit vector for the possible actions.
- $M(\cdot)$ selects the action with the highest merit.
- $\hat{\pi}(x; w) = M(g(x; w))$
- Since $M(\cdot)$ is predefined, we only need to learn $g(\cdot)$
Approximate the optimal reinforcement with \( \hat{r}(x; v) \approx r^*(x) \), where we write the training rule for the parameters \( v \) in shorthand:

\[
\hat{r}(x; v) := \hat{r}(x; v) + \beta (r(x, a) - \hat{r}(x; v))
\]

Notice that \( \hat{r}(x) \) will approximate \( E[r(x, a)|x] \), but that this value will converge to \( r^*(x) \) if \( a \) is chosen with the max selector, \( M(.) \).

The update for the merit function (\( k^{th} \) action) is:

\[
g_k(x; w) := g_k(x; w) + \alpha (r(x, a_k) - \hat{r}(x; v)).
\]

- Problems:
  1. at each \( x \), need to evaluate all members of \( \mathcal{A} \)
  2. maximum selector \( M(.) \) doesn’t allow for exploration

- So:
  1. use stochastic action policy
  2. decrease stochasticity with time (simulated annealing)

- Also:
  1. decrease \( \beta \) as \( \hat{r} \to r^* \)
  2. randomize initial conditions over many trials.

### 6.2.2 Continuous Actions

\( \mathcal{A} \subseteq \mathbb{R} \)

Merit vectors are no longer suitable, so we need to adapt the action policy directly. There are two basic approaches.

**Stochastic:** A perturbation \( \eta \) is added for exploration.

\[
\hat{\pi}(\cdot) = h(\cdot; w) + \eta,
\]

where \( \eta \sim N(0, \sigma^2) \) The learning rule for the function approximator is:

\[
h(x; w) := h(x; w) + \alpha [r(x, a) - \hat{r}(x; v)] \eta
\]

Which increases the probability of choosing actions with optimal reinforcement.

**Deterministic:** If \( r \) is differentiable,

\[
\hat{\pi}(x; w) := \hat{\pi}(x; w) + \alpha \frac{\partial r(x, a)}{a}
\]

If \( r \) is unknown, use \( \hat{r} \).

The stochastic method does not suffer from local maxima problems, whereas the deterministic method is expected to be much faster because it searches systematically.
6.3 Delayed RL

6.3.1 Finite States and Action Sets

Assume Markovian and stationary stochastic dynamic systems; i.e.,

\[ P(x_{T+1} = y | \{ (x_t, a_t) \}^T_0) = P(x_{T+1} = y | x_T, a_T), \]

which is just \( P_{xy}(a) \), the probability of moving to state \( y \) from state \( x \) via action \( a \).

**Value Functions:** We begin by defining the value of a policy \( \pi \) at state \( x \):

\[
V^\pi(x) = E \left[ \sum_{t=0}^{\infty} \gamma^t r(x_t, \pi(x_t)) \left| x_0 = x \right. \right],
\]

which can be interpreted as:

\[
V^\pi(x) = \lim_{N \to \infty} E \left[ \sum_{t=0}^{N-1} \gamma^t r(x_t, \pi(x_t)) \left| x_0 = x \right. \right].
\]

\( \gamma \in [0, 1) \) is a discount factor on future rewards. We can now define the optimal value function in terms of \( V^\pi(x) \):

\[
V^*(x) = \max_{\pi} V^\pi(x) \quad \forall x
\]

**Optimal Policy:** Define optimal policy \( \pi^* \) as:

\[
\pi^*(x) = \arg \max_{\pi} V^\pi(x) \quad \forall x \in \mathcal{X},
\]

which is to say,

\[
V^*(x) = V^{\pi^*}(x) \quad \forall x \in \mathcal{X}.
\]

Using Bellman’s optimality equation,

\[
V^*(x) = \max_{a \in \mathcal{A}(x)} \left[ r(x, a) + \gamma \sum_{y \in \mathcal{X}} P_{xy}(a) V^*(y) \right],
\]

and provided that \( V^* \) is known, we have a mechanism for finding the optimal policy:

\[
\pi^*(x) = \arg \max_{a \in \mathcal{A}(x)} \left[ r(x, a) + \gamma \sum_{y \in \mathcal{X}} P_{xy}(a) V^*(y) \right]
\]

Unfortunately, this requires the system model, \( P_{xy}(a) \) in addition to our approximation of \( V^\pi \).

**Q Functions:** The idea is to incorporate \( P_{xy}(a) \) into the approximation. Define

\[
Q^\pi(x, a) = r(x, a) + \gamma \sum_{y \in \mathcal{X}} P_{xy}(a) V^*(y)
\]

so \( Q^\pi = Q^{\pi^*} \Rightarrow V^*(x) = \max_a [Q^*(x, a)] \), and

\[
\pi^*(x) = \arg \max_a [Q^*(x, a)]
\]

246
6.4 Methods of Estimating $V^\pi$ (and $Q^\pi$)

6.4.1 n-Step Truncated Return

This is a simple approximation of the value function

$$V^{[n]}(x) = \sum_{\tau=0}^{n-1} \gamma^\tau r_\tau \quad \hat{V}(x; v) = E[V^{[n]}(x)]$$

$\hat{V}(x; v) \to V(x)$ uniformly as $n \to \infty$, but note that the variance of $V^{[n]}(x)$ increases with $n$.

- **Problem:** computing $E[.]$.
- **Solutions:**
  1. use $\hat{V}(x; v) = V^{[n]}(x)$ (high variance for high $n$)
  2. use $\hat{V}(x; v) := \hat{V}(x; v) + \beta[\hat{V}^{[n]}(x) - \hat{V}(x; v)]$ and iterate over many trials.

6.4.2 Corrected n-Step Truncated Return

We can take advantage of having an approximation of $V$ on hand by adding a second term to the n-step truncated return:

$$V^{[n]}(x) = \sum_{\tau=0}^{n-1} \gamma^\tau r_\tau + \gamma^n \hat{V}(x; v_{old}) \quad \hat{V}(x; v_{new}) = E[V^{[n]}(x)]$$

- **Approximating $E[.]$:**
  1. use $\hat{V}(x; v) := \hat{V}(x; v) + \beta[V^{[n]}(x) - \hat{V}(x; v)]$

6.4.3 Which is Better: Corrected or Uncorrected?

- **Known Model Case, Using Expectation**
  1. Uncorrected convergence:
     $$\max |\hat{V}(x; v) - V(x)| \leq \frac{\gamma^n r_{\max}}{1 - \gamma}$$
  2. Corrected convergence:
     $$\max |\hat{V}(x; v_{new}) - V(x)| \leq \gamma^n \max |\hat{V}(x; v_{old}) - V(x)|$$

     This is a tighter bound.

- **Model Unavailable, or Expectation Too Expensive**
  1. If $\hat{V}$ is good, use Corrected and small $n$. Using Uncorrected would require large $n$, resulting in higher variance.
  2. If $\hat{V}$ is poor, both require large $n$, and the difference is negligible.

- $V^{[n]}$ is better

- We need a way of choosing truncation length $n$ dynamically, depending on the goodness of $\hat{V}$. 
6.4.4 Temporal Difference Methods

- Consider a geometric average of $V^{(n)}$'s, over all $n$.

\[
V^\lambda(x) = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} V^{(n)}(x)
= (1 - \lambda)(V^{(1)} + \lambda V^{(2)} + \lambda^2 V^{(3)} + \cdots)
\]

Using our definition of $V^{(n)}$ and expanding gives:

\[
V^\lambda(x) = r_0 + \gamma (1 - \lambda)\hat{V}(x_1; v) + \gamma \lambda [r_1 + \gamma (1 - \lambda)\hat{V}(x_2; v) + \gamma \lambda [r_2 + \gamma (1 - \lambda)\hat{V}(x_3; v) + \cdots
\]

which (using $r_0 = r(x, \pi(x))$ ) can be computed recursively as

\[
\left\{
\begin{array}{l}
V^\lambda(x_t) = r(x_t, \pi(x_t)) + \gamma (1 - \lambda)\hat{V}(x_{t+1}; v) + \gamma \lambda V^\lambda(x_{t+1}) \\
x_{t+1} \coloneqq x_t
\end{array}
\right.
\]

- Note that:

\[
\lambda = 0 \implies V^0(x) = r_0 + \hat{V}(x; v) = V^{(1)}(x)
\]

\[
\lambda = 1 \implies V^1(x) = r_0 + \gamma V^1(x_1)
\]

\[
\implies V^1(x) = r_0 + \gamma (r_1 + V^1(x_2)) = V^{(\infty)}(x)
\]

\[
\implies V^1(x) = V(x)
\]

- Idea: vary approximate truncation length, $n$, from 0 to $\infty$ by starting with $\lambda = 1$ and letting $\lambda \to 0$ as $\hat{V}$ improves.

- Problem: $V^\lambda$ takes forever to compute because of the infinite sum over $n$.

- Solution: approximate $V^\lambda$ on-line. Replace the corrected n-step estimate with:

\[
\hat{V}(x; v) := \hat{V}(x; v) + \beta [V^\lambda(x) - \hat{V}(x; v)] .
\]

Now define the temporal difference as the difference between the new prediction of the value and the old prediction:

\[
\Delta(x) = r(x; \pi(x)) + \gamma \hat{V}(x_1; v) - \hat{V}(x; v) .
\]

The second term in equation 148 now can be written as:

\[
[V^\lambda(x) - \hat{V}(x; v)] = \Delta(x) + (\gamma \lambda) \Delta(x_1) + (\gamma \lambda)^2 \Delta(x_2) + \cdots
\]

and we can approximate equation 148 with the online update:

\[
\hat{V}(x; v) := \hat{V}(x; v) + \beta (\gamma \lambda)^r \Delta(x_r) \quad (149)
\]

if $\beta$ is small.
Note that equation 148 and 149 update $\hat{V}$ for only a single state. We can update several states by introducing the notion of eligibility.

$$e(x, t) = \begin{cases} 
0 & \text{if } x \text{ never visited} \\
\gamma \lambda e(x, t - 1) & \text{if } x \neq x_t \text{ (not currently visited)} \\
1 + \gamma \lambda e(x, t - 1) & \text{if } x = x_t \text{ (currently visited)} 
\end{cases}$$

This results in an update equation for all states:

$$\hat{V}(x; v) := \hat{V}(x; v) + \beta e(x, t) \Delta(x_t) \quad \forall x \in \mathcal{X}$$

(150)

Notes:
1. Reset $\epsilon$’s before each trial.
2. Decrease $\beta$ during training
3. Basically the same for Q function

6.5 Relating Delayed-RL to DP Methods

<table>
<thead>
<tr>
<th>Delayed Rein. Learning</th>
<th>Dynamic Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>works on-line</td>
<td>works off-line</td>
</tr>
<tr>
<td>uses subset of $\mathcal{X}$</td>
<td>uses entire state-space $\mathcal{X}$</td>
</tr>
<tr>
<td>can be model-free</td>
<td>requires a model</td>
</tr>
<tr>
<td>time-varying</td>
<td>stationary</td>
</tr>
</tbody>
</table>

There are two popular DP methods: value iteration and policy iteration. Both of these have spawned various dRL counterparts. There are two categories of delayed RL methods: model-based and model-free. Model-based methods have direct links to DP methods. Model-free methods are essentially modifications of the model-based methods.

6.6 Value Iteration Methods

Original DP Method: The basic idea is to compute the optimal value function as the limit of finite-horizon optimal value functions:

$$V^*(x) = \lim_{n \to \infty} V_n^*(x).$$

To this end, we can use the recursion

$$V_{n+1}^*(x) = \max_{a \in \mathcal{A}} \left[ r(x, a) + \gamma \sum_y P_{xy}(a) V_n^*(y) \right] \quad \forall x$$

to compute $V_N^*$ for a large number $N$. The policy associated with any given value function (say, $V_n^*$ after an arbitrary number of iterations) is:

$$\pi(x) = \arg \max_{a \in \mathcal{A}} \left[ r(x, a) + \gamma \sum_y P_{xy}(a) V_n^*(y) \right] \quad \forall x$$

Problem: $\mathcal{X}$ can be very large, making value iteration prohibitively expensive.
• Solution: We can update \( V^*_n(x) \) on a subset \( B_n \subset \mathcal{X} \) of the state-space.

**Real Time Dynamic Programming**: RTDP is a model-based dRL method that does this on-line (\( n \) is system time) and \( B_n \) includes the temporal neighbors of \( x_n \).

**Q-Learning**: this is a model-free dRL approach to value-iteration. As described in section 3, the idea is to incorporate the system model into the approximation. Again the update rule for the \( Q \)-function is:

\[
\hat{Q}(x, a; v) := r(x, a) + \gamma \sum_{y \in A(y)} P_{xy} \max_{b \in A(y)} \hat{Q}(y, b; v).
\]

(151)

However, this update relies on a system model. We can avoid this by looking only at the \( Q \)-function of the state that the system actually transitions to (say, \( x_1 \)):

\[
\hat{Q}(x, a; v) := r(x, a) + \gamma \max_{b \in A(x_1)} \hat{Q}(x_1, b; v).
\]

(152)

To approximate the averaging effect found in equation 151, we can use an averaging update rule:

\[
\hat{Q}(x, a; v) := (1 - \beta)\hat{Q}(x, a; v) + \beta \left[ r(x, a) + \gamma \max_{b \in A(x_1)} \hat{Q}(x_1, b; v) \right].
\]

(153)

The Q-learning algorithm consists of repeatedly (1) choosing an action \( a \in A \), (2) applying equation 153, and (3) updating the state \( x := x_1 \).

The choice of action in step (1) can be greedy with respect to \( \hat{Q} \), or can be made according to some stochastic policy influenced by \( \hat{Q} \), to allow for better exploration.

### 6.6.1 Policy Iteration Methods

Whereas value iteration methods maintain an explicit representation of the value function, and implements a policy only in association with the value function, policy iteration methods maintain an explicit representation of the policy.

**Original DP Method**: The idea is that if the \( Q \)-function for a policy \( \pi \) is known, then we can improve on it in the following way:

\[
\mu(x) = \arg \max_{a \in A(x)} Q^\pi(x, a) \quad \forall x,
\]

(154)

so that the new policy \( \mu \) is at least as good as the old policy \( \pi \). This is because the \( Q^\pi(x, a) \) gives the value of first choosing action \( a \), and then following policy \( \pi \). The steps of policy iteration are:

1. Start with a random policy \( \pi \) and compute \( V^\pi \).
2. Compute \( Q^\pi \).
3. Find \( \mu \) using equation 154.
4. Compute \( V^\mu \).
5. Set \( \pi := \mu \) and \( V^\pi := V^\mu \).
6. Go to step 2 until \( V^\mu = V^\pi \) in step 4.

**Problem**: It is very expensive to recompute \( V^\pi \) at each iteration. The following method gives an inexpensive approximation.
**Actor-Critic:** This is a model-free dRL method (there isn’t a model-based dRL counterpart to policy iteration). The “actor” is a stochastic policy generator, and the “critic” is a value-function estimator. The algorithm is as follows:

1. Select action $a$ stochastically according to the current policy, $a = \pi(x)$.
2. $\Delta(x) = r(x, a) + \gamma \hat{V}(x_1, v) - \hat{V}(x; v)$ serves as the immediate reinforcement.
3. Update the value estimator, $\hat{V}(x; v) := \hat{V}(x; v) + \beta \Delta(x)$.
4. Update the deterministic part of the policy generator, $g_k(x; w) := g_k(x; w) + \alpha \Delta(x)$ where $k \ni a = a_k$ is the index of the action selected.
5. Repeat.

### 6.6.2 Continuous Action Spaces

So far the assumption has been that there are only a finite number of actions for the policy generator to choose from. If we now consider the case of continuous action spaces, the problem becomes much more difficult. For example, even value-iteration methods must maintain a policy generator because the max operation becomes nontrivial. The general approach is to update the policy generator by taking the partial derivative of the Q-function (or $\hat{Q}$) with respect to action $a$. This is a gradient ascent approach, which adjusts the policy in the direction of increasing utility; it is comparable to the immediate RL methods for continuous action spaces.

### 7 Selected References on Neural Control

**(Books)**


**(Articles)**

2. S. Keerthi, B. Ravindran. A tutorial Survey of Reinforcement Learning. Available by request to ravi@hanakya.csa.iiisc.ernet.in.


8 Videos

*(List of Videos shown in class)*

1. AIVINN - Dean Pomerleau, CMU
2. Intelligent Arc Furnace - Neural Applications corp,
3. Trucks and Brooms - Bernard Widrow, Stanford University,
4. Artificial Fishes - Demitri Terzopolous, Toronto.
5. Anthropomorphic Robot Experiments - Stefan Schaal, Georgia Tech.
Part IX
Fuzzy Logic & Control

1 Fuzzy Systems - Overview

- Fuzzy Methods are firmly grounded in the mathematics of Fuzzy Logic developed by Lofti Zadeh (1965).
- Very popular in Japan (great buzz-word). Used in many products as decision making elements and controllers.
- Allows engineer to quickly develop a smooth controller from an intuitive understanding of problem and engineering heuristics.
- Are different from and complementary to neural networks.
- Possible over-zealous application to unsuitable problems.
- Typically the intuitive understanding is in the form of inference rules based on linguistic variables.
- Rule based
- An example might be a temperature controller: If the break temperature is HOT and the speed is NOT VERY FAST, then break SLIGHTLY DECREASED.
- A mathematical way to represent vagueness.
- Different from Probability Theory.
- An easy method to design adequate control system.
- When to use Fuzzy Logic:
  - When one or more of the control variables are continuous.
  - When an adequate mathematical model of the system does not exist or is too expensive to use.
  - When the goal cannot be expressed easily in terms of equations.
  - When a good, but not necessarily optimal, solution is needed.
  - When the public is still fascinated with it!

2 Sample Commercial Applications

- Fujitec / Toshiba - Elevator: Evaluates passenger traffic, reduces wait time.
- Maruman Golf - Diagnoses golf swing to choose best clubs.
- Sanyo Fisher / Canon - Camcorder focus control: Chooses best lighting and focus with many subjects.

• Mitsubishi - Air conditioner: Determines optimum operating level to avoid on/off cycles.


• Subaru - Auto Transmission: Senses driving style and engine load to select best gear ratio. (Continuum of possible gears!)

• Saturn - Fuzzy Transmission.

• Yamaichi Securities - Stocks: Manage stock portfolios (manages $350 million).

3 Regular Set Theory

• We define a ‘set’ as a collection of elements. Example:

\[ A = \{x_1, x_2, x_3\} \]
\[ B = \{x_3, x_4, x_5\} \]

• We can thus say that a certain thing is an element of a set, or is a member of a set.

\[ x_1 \in A; \quad x_3 \in B \]

• We can also speak of ‘intersection’ and ‘union’ of sets.

\[ A \cup B = \{x_1, x_2, x_3, x_4, x_5\} \]
\[ A \cap B = \{x_3\} \]

• “Fuzzy people” call the set of all possible “things” (all X) the universe of discourse.

• The complement is \( \bar{A} = \{x_i \mid x_i \notin A\} \)

• “Membership” is a binary operation, either \( x \in A \) or \( x \notin A \).
4 Fuzzy Logic

- In Fuzzy Logic we allow membership to occur in varying degrees.
- Instead of listing elements, we must also define their degree of membership.
- A membership function is specified for each set.

\[ f_A(x) : X \rightarrow [0, 1] \leftarrow \text{degree of membership} \]

- Example:

\[ f_A : X \rightarrow [0, 1] \]

- \( X \rightarrow \) range of possible members or input values, typically subset of \( \mathbb{R} \).
- If \( f_A(x) = \{0, 1\} \) then we have regular set theory.
- Typically we describe a set simply by drawing the activation function (membership function).

\[ f_A(1.0) = 0.4; \; f_B(1.0) = 0.1 \]

1.0 ‘belongs’ to sets A and B in differing amounts.

4.1 Definitions

- Fuzzy Logic developed by Zadeh shows that all the normal set operations can be given precise definitions such that normal properties of set theory still hold.

- Empty set

\[ A = \emptyset \text{ if } f_A(x) = 0 \; \forall x \in X \]

- Equality

\[ A = B \text{ if } f_A(x) = f_B(x) \; \forall x \in X \]
• **Complement**

\[ A^c \text{ has } f^c_A = 1 - f_A(x) \]

• **Containment or Subset**

\[ A \subset B \text{ if } f_A(x) \leq f_B(x) \forall x \in X \]

• **Union**

\[ f_{A\cup B}(x) = \max [f_A(x), f_B(x)] \forall x \in X \]

\[ = f_A \cup f_B \]

This is equivalent to the ‘Smallest set containing both A and B’.

• **Intersection**

\[ f_{A\cap B}(x) = \min [f_A(x), f_B(x)] \forall \epsilon X \]

\[ = f_A \cap f_B \]

This is the ‘Largest set contained in both A and B’.

4.2 **Properties**

• A number of properties (set theoretical and algebraic) have been shown for Fuzzy Sets. A few are:

• **Associative**

\[ (A \cap B) \cap C = A \cap (B \cap C) \]

\[ (A \cup B) \cup C = A \cup (B \cup C) \]

• **Distributive**

\[ C \cap (A \cup B) = (C \cap A) \cup (C \cap B) \]

\[ C \cup (A \cap B) = (C \cup A) \cap (C \cup B) \]

• **De Morgan**

\[ (A \cup B)^c = A^c \cap B^c \]

\[ (A \cap B)^c = A^c \cup B^c \]
4.3 Fuzzy Relations

- We can also have fuzzy relations of multiple variables.
- In regular set theory, a ‘relation’ is a set of ordered pairs i.e.,
  \[ A = x \leq y = \{(5, 4), (2, 1), (3, 0), \ldots \} \]
- We can have fuzzy relations by assigning a ‘degree of correctness’ to the logical statement and define membership functions in terms of two variables.
  \[ f_A(10, 9) = 0; \quad f_A(100, 10) = 0.7; \quad f_A(100, 1) = 1.0 \]

4.4 Boolean Functions

- In a real problem, \( x \) would represent a real variable (temperature) and each set a linguistic premise (room is cold).
- Example, let \( t \) and \( P \) be variables representing actual temperature and pressure.
  - Let \( A \) be the premise “temperature is hot”
  - Let \( B \) be “pressure is normal”.
  - Assume the following membership functions.

  [Diagram of membership functions for A and B]

  - “temperature hot AND pressure normal” is a fuzzy relation given by the set
    \[ \{ A \text{ AND } B \} = C_1 = \{ A \cap B \} \]
    \[ f_{C_1}(t, P) = \min [f_A(t), f_B(P)] \forall t, P \]
  - For OR,
    \[ \{ A \text{ OR } B \} = C_2 = \{ A \cup B \} \]
    \[ f_{C_2}(t, P) = \max [f_A(t), f_B(P)] \forall t, P \]
  - For NOT,
    \[ \{ \text{NOT } A \} = C_3 = A^c \]
    \[ f_{C_3}(t, P) = 1 - f_A(t) \forall t \]
  - Note, the resulting fuzzy relation is also a fuzzy set defined by a membership function.
4.5 Other Definitions

- Support and cross over.

\[ U_A \leftarrow U^\infty_A[X] \quad n > 1 \]

- Singleton

4.6 Hedges

- Modifiers: not, very, extremely, about, generally, somewhat...
5 Fuzzy Systems

- Fuzzifier
  - Each Sensor input is compared to a set of possible linguistic variables to determine membership.
  - Provides “Crisp” values

- Process Rules (Fuzzy Logic)
  - Each rule has an antecedent statement with an associated fuzzy relation. Degree of membership for antecedent is computed. - Fuzzy Logic.
  - IF (ANTECEDENT) THEN (CONSEQUENCE).
Defuzzifier

- Each rule has an associated action (consequence) which is also a fuzzy set or linguistic variable such as 'turn right slightly'.
- Defuzzification creates a combined action fuzzy set which is the intersection of output sets for each rule, weighted by the degree of membership for antecedent of the rule.
- Converts output fuzzy set to a single control value.
- This defuzzification is not part of the 'mathematical fuzzy logic' and various strategies are possible.
- We introduce several Defuzzification possibilities later.

Design Process - Control designer must do three things:

1. Decide on appropriate linguistic quantities for inputs and outputs.
2. Define a membership function (fuzzy sets) for each linguistic quantity.
3. Define the inference rules.

5.1 A simple example of process control.

- We must first decide on linguistic variables and membership functions for fuzzy sets.

Next, we choose some rules of inference

if \( P = H \) and \( t = C \) then \( \Delta V = G \) (1)
if \( P = H \) and \( t = W \) then \( \Delta V = L \) (2)

if \( P = N \) then \( \Delta V = Z \) (3).

This is the same as
if \( P = N \) and "1" then \( \Delta V = Z \)

“Karnaugh Map”

- Fuzzify - Infer - Defuzzify
5.2 Some Strategies for Defuzzification

- Center of Mass

If $B^*$ is the composite output Fuzzy set,

$$y_0 = \frac{\int y f_{B^*}(y) \, dy}{\int f_{B^*}(y) \, dy}$$

This is probably the best solution, and best motivated from fuzzy logic, but is hard to compute.

- Peak Value

$$y_0 = \arg \max_y f_{B^*}(y)$$

This may not be unique.

- Weighted Maximums

For each rule there is a $B_i$ and an activation degree for antecedent $w_i$.

$$y_0 = \frac{\sum w_i \max_y f_{B_i}(y)}{\sum w_i}$$

- functions of inputs.

Can also have outputs be different functions of the original inputs.

$$y_0 = \frac{\sum w_i \Phi_i(\chi_1 \cdots \chi_n)}{\sum w_i}$$

5.3 Variations on Inference

- Inference
  - Mamdani - minimum
  - Larson - product

- Composition
  - Maximum (traditional method)
  - Sum (weighted and/or normalized)
Fig. 12. Diagrammatic representation of fuzzy reasoning 1.

Fig. 13. Diagrammatic representation of fuzzy reasoning 2.
5.4 7 Rules for the Broom Balancer

5.5 Summary Comments on Basic Fuzzy Systems

- Easy to implement (if number of inputs are small).
- Based on rules which may relate directly to understanding of problem.
- Produces smooth mapping automatically.
- May not be optimal.
- Based on designer’s heuristic understanding of the solution.
- Designer may have no idea what inference rules may be.
- Extremely tedious for large inputs.
- Often a conventional controller will work better.
- Cautions: There is a lot of over-zealousness about Fuzzy Logic (buzzword hype). There is nothing magical about these systems. If conventional controller works well, don’t toss it out.
6 Adaptive Fuzzy Systems

Adaptive fuzzy systems use various ad hoc techniques which attempt to:

- Modify shape and location of fuzzy sets.
- Combine or delete unnecessary sets and rules.
- Use unsupervised clustering techniques.
- Use “neural” techniques to adapt various parameters of the fuzzy system.

6.1 Fuzzy Clustering

6.1.1 Fuzzy logic viewed as a mapping

Example: If X is NS then Y is PS.
6.1.2 Adaptive Product Space Clustering

- Vector quantization - Unsupervised Learning.
- Widths defined by covariance.
- Gaussian Membership Functions ≡ Radial Basis Function Network.
- If $X_1$ is $A_{1i}$ and $X_2$ is $A_{2i}$ then $Y$ is $B_i$ (adds 3rd dimension).
- Kosko - optimal rules cover extremes

6.2 Additive Model with weights (Kosko)

- Can also set weights $w_i$ to equal $w_i = \frac{k_i}{k}$, where $k_i$ is the cluster count in cell $i$, and $k$ is the total count.
If we threshold this to \( w_i = 0/1 \) which implies adding or removing rule.

- Additive Fuzzy Models are “universal approximators”.

### 6.3 Example Ad-Hoc Adaptive Fuzzy

- **Adaptive Weights**

  Weights initially set to 1 and then adapted. Truth of premise (0 to 1) reduced as \( w \to 0 \).

  Adaptation can be supervised (like BP) or some reinforcement method (punish/reward weights based on frequency of rules that fire).

- **Adaptive Membership function support**

  Example:

  If \( e = y - d \) is positive the the system is over-reacting thus narrow regions of rules that fired.

  If \( e \) is negative \( \rightarrow \) Widen regions.

  (Restrict amount of overlap to avoid instabilities)

### 6.4 Neuro-Fuzzy Systems

Make Fuzzy systems look like Neural Net and then apply any of the learning algorithms, (BP, RTBP, etc).
6.4.1 Example: Sugene Fuzzy Model

if $X$ is $A$ and $Y$ is $B$ then $Z = f(x, y)$.

- **Layer 1 - MFs**

  $$MF_{A_i}(x) = \frac{1}{1 + \left(\frac{x-a_i}{b_i}\right)^2}$$

  $\Rightarrow$ bell shape function parameterized by $a_i, b_i, c_i$.

- **Layer 2 - AND - product or soft min**

  $$soft\text{-}min(x, y) = \frac{xe^{-kx} + ye^{-ky}}{e^{-kx} + e^{-ky}}$$

- **Layer 3 - Weight Normalization**

- **Layer 4 = $\overline{w_i}(p_i x + q_i y + r_i)$**

- **Layer 5 = $\sum w_i f_i$**

  Network is parameterized by $a_i, b_i, c_i, p_i, q_i, r_i$. Train using gradient descent.

- In general replace hard max/min by soft max/min and use BP/RTRL, BPTT, etc.
6.4.2 Another Simple Example

- Membership functions:

\[
U_{ij} = \begin{cases} 
\frac{x_i - a_{1(i-1)}}{a_{1i} - a_{1(i-1)}} & a_{1(i-1)} < x_i < a_{1i} \\
\frac{a_{1(i+1)} - x_i}{a_{1(i+1)} - a_{1i}} & a_{1i} < x_i \leq a_{1(i+1)} \\
0 & \text{otherwise}
\end{cases}
\] (155)

if \(U_{1i} \text{ AND } U_{2j}\) then ....

- Constraint \(a_{11} < a_{12} < a_{1m}\)
- AND - Product Rule
- Defuzzification - Weighted Average

\[
Y = \sum_{i,j} (U_{1i}U_{2j})w_{ij}
\]

- Gradient descent on error:

\[
\text{Error} = \frac{1}{2} \Sigma (Y_k - d_k)^2
\]

\[
\frac{\partial E}{\partial w_{ij}} = \sum_k (Y_k - d_k)(U_{1i}U_{2j})
\]

\[
\frac{\partial E}{\partial a_{1i}} = \sum_j \sum_k (Y_k - d_k)(\sum_{i-1}^{i+1} U_{1i}w_{ij})
\]

7 Neural & Fuzzy

In 1991 Matsushita Electronics claimed to have 14 consumer products using both neural networks and fuzzy logic.
7.1 NN as design Tool

- Neural Net outputs center locations and width of membership functions.

7.2 NN for pre-processing

- Output of Neural Net is "Predicted Mean Vote" - PMV, and is meant to be an index of comfort.

- This output is fed as control input to a fuzzy controller.

- Example: Input to NNet is image. Output of net in classification of stop sign / no stop sign which goes as input to Fuzzy logic.
7.3 NN corrective type

- Example: Hitachi Washing Machine, Sanyo Microwave.
- Fuzzy designed on original inputs.
- NN added later to handle additional inputs and avoid redesigning with too many inputs and membership functions.

7.4 NN for post-processing

- Fuzzy for higher level logic and NN for low level control.
- Example: Fuzzy determines flight high-level commands, neural implements them.
- Example: Sanyo fan - Points to holder of controller. Fuzzy - 3 infrared sensors used to compute distance. Neural - given distance & ratio of sensor outputs are used to command direction.

7.5 Fuzzy Control and NN system ID

- Use NNet for system ID which provides error (through BP) to adapt Fuzzy controller.
8 Fuzzy Control Examples

8.1 Intelligent Cruise Control with Fuzzy Logic

R. Muller, G. Nocker, Daimler-Benz AG.

Figure 3: Membership functions for controller inputs and output

Figure 4: Rule base for fuzzy controller
8.2 Fuzzy Logic Anti-Lock Brake System for a Limited Range Coefficient of Friction Surface

D. Madau, F. Yuan, L. Davis, L. Feldkamp, Research Laboratory, Ford Motor Company.

Figure 6: Fuzzy Logic ABS: High $\mu$ Results

Figure 7: Conventional ABS: High $\mu$ Results
The following five rules were used to drive the fuzzy logic kernel:

1. If $\lambda$ is Pos. Small then $\delta u$ is Pos. Small
2. If $\alpha_{\text{whl}}$ is Neg. Large then $\delta u$ is Neg. Med.
3. If $\alpha_{\text{whl}}$ is Pos. Large then $\delta u$ is Pos. Large
4. If $\lambda$ is Pos. Large then $\delta u$ is Neg. Large
5. If $\lambda$ is Pos. Medium then $\delta u$ is Neg. Small

Figure 9: Final Rule Set for ABS
9 Appendix - Fuzziness vs Randomness

(E. Plumer)

Fuzzy logic seems to have the same “flavor” as probability theory. Both attempt to describe uncertainty in certain events. Fundamentally, however, they are very different. For an excellent discussion on the differences and implications thereof, read (B. Kosko, Neural networks and fuzzy systems: A dynamical systems approach to machine learning, Prentice-Hall, 1992, chapter 7). It would seem that we can make the following identifications and use probability theory to describe fuzzy logic:

<table>
<thead>
<tr>
<th>Fuzzy Logic</th>
<th>Probability Theory</th>
</tr>
</thead>
<tbody>
<tr>
<td>fuzzy set</td>
<td>random variable</td>
</tr>
<tr>
<td>membership function, $f_A(x)$</td>
<td>probability density, $p_A(x)$</td>
</tr>
<tr>
<td>element of fuzzy set</td>
<td>value of random variable</td>
</tr>
</tbody>
</table>

As we shall see, this won’t work. Before waxing philosophical on the differences, we note a few problems with using probability theory. Recalling basics of probability theory (see M. Gray and L. Davisson, Random processes: A mathematical approach for engineers, Prentice-Hall, 1986) we define a probability space $(\Omega, F, P)$ as having three parts:

1. Sample Space, $\Omega$ — collection of points or members. Also called “elementary outcomes.”

2. Event Space, $F$ — Nonempty collection of subsets of $\Omega$. Also called a sigma algebra of $\Omega$. This set must obey the following properties: (1) if $F \in F$, then $F^c \in F$, where $F^c = \{ \omega : \omega \in \Omega, \omega \not\in F \}$ is the compliment of $F$; (2) for finite (or countably infinite) number of sets $F_i$, we have $\bigcup_i F_i \in F$.

3. Probability Measure, $P$ — assignment of a real number, $P(F)$ to every member of the sigma algebra such that $P$ obeys the following axioms of probability: (1) for all $F \in F$, $P(F) \geq 0$; (2) for disjoint sets $F_i$ (i.e. $i \neq j$ implies $F_i \cap F_j = \emptyset$), we have $P(\bigcup_i F_i) = \sum_i P(F_i)$.

A random variable is simply a mapping from the event space to the real numbers $X : F \to \mathbb{R}$. These properties must hold in order to have a probability space and to be able to apply probability theory.

Thus, a probability space has at its root a set which obeys regular set theory. Built upon this set is the probability measure describing the “randomness.” On the other hand, in fuzzy logic, the basic notion of a set and of all set operations and set properties is redefined. Furthermore, fuzzy sets and membership functions do not satisfy the axioms of probability theory. As an example, recall that for a probability density function $p_X(x)$, we must have $0 \leq p_X(x) \leq 1$ and $\int_\mathbb{R} p_X(x) \, dx = 1$. However, for membership functions, although $0 \leq f_A(x) \leq 1$, the value of $\int_\mathbb{R} f_A(x) \, dx$ can be anything! Also, consider this very profound statement: $A \cap A^c = \emptyset$ which states that a set and its compliment are mutually exclusive. This certainly holds true for the event space underlying a probability space. However, in fuzzy logic this is no longer true! $A \cap A^c$ has membership function $f_{A \cap A^c}(x) = \min [f_A(x), 1 - f_A(x)]$ which is clearly not identically zero. Moral: probability theory is not adequate to describe fuzzy logic.

We can also provide a few intuitive reasons why “randomness” and “fuzziness” are not the same thing. To quote Kosko:
Fuzziness describes \textit{event ambiguity}. It measures the degree to which an event occurs, not whether it occurs. Randomness describes the uncertainty of \textit{event occurrence}. An event occurs or not, ... The issue concerns the occurring event: Is it uncertain in any way? Can we unambiguously distinguish the event from its opposite? [pp. 265].

Basically probability theory accounts for an event occurring or not occurring but doesn’t permit an event to partially occur.

Underlying probability theory is the abstract idea of some type of “coin toss experiment” which chooses one of the elements of \( F \) as the event. The measure \( P \) describes the likelihood of any particular event occurring and the random variable \( X \) maps the event to some output number \( x \). In fuzzy logic, there is no randomness, no notion of an underlying experimental coin toss, and no uncertainty of the outcome.

A further contrasting can be made as follows: In fuzzy logic, each real value (such as \( T = 50^\circ C \)) is associated, in varying degrees, with one or more possible fuzzy sets or linguistic variables (such as “\( T \) is hot” or “\( T \) is very hot”). The membership function measures the degree to which each fuzzy set is an “attribute” of the real value. In probability theory, on the other hand, the roles are reversed. A real value does not take on a random variable, rather, a random variable takes on one of a number of possible real values depending on the outcome of the underlying experiment.

Beyond simply describing and approach to designing control systems, “fuzziness” proposes a completely different way of explaining uncertainty in the world. As Kosko points out in his book, for example:

Does quantum mechanics deal with the probability that an unambiguous electron occupies spacetime points? Or does it deal with the degree to which an electron, or an electron smear, occurs at spacetime points? [pp. 266].

More fun philosophical examples are given in the rest of chapter 7 of Kosko’s book.
Part X
Exercises and Simulator

- Home work 1 - Continuous Classical Control
- Home work 2 - Digital Classical Control
- Home work 3 - State-Space
- Home work 4 - Double Pendulum Simulator - Linear SS Control
- Midterm
- Double Pendulum Simulator - Neural Control/BPTT
- Double Pendulum Simulator Documents
0.1 System description for double pendulum on a cart

System parameters

\[
\begin{align*}
    l_1 &= \text{length of first stick} \\
    l_2 &= \text{length of second stick} \\
    m_1 &= \text{mass of first stick} \\
    m_2 &= \text{mass of second stick} \\
    M &= \text{mass cart}
\end{align*}
\]  

States

\[
\begin{align*}
    x &= \text{position of the cart} \\
    \dot{x} &= \text{velocity of cart} \\
    \theta_1 &= \text{angle of first stick} \\
    \dot{\theta}_1 &= \text{angular velocity of first stick} \\
    \theta_2 &= \text{angle of second stick} \\
    \dot{\theta}_2 &= \text{angular velocity of second stick}
\end{align*}
\]

Control force

\[
u = \text{force}
\]
Dynamics

Using a Lagrange approach, the dynamic equations can be written as:

\[
(M + m_1 + m_2) \ddot{x} - (m_1 + 2m_2)l_1 \ddot{\theta}_1 \cos \theta_1 - m_2 l_2 \ddot{\theta}_2 \cos \theta_2 = u + (m_1 + 2m_2) l_1 \dot{\theta}_1^2 \sin \theta_1 + m_2 l_2 \dot{\theta}_2^2 \sin \theta_2 \tag{168}
\]

\[
-(m_1 + 2m_2) l_1 \dot{x} \cos \theta_1 + 4 \left(\frac{m_1}{3} + m_2\right) l_1 \ddot{\theta}_1 + 2m_2 l_1 l_2 \ddot{\theta}_2 \cos (\theta_2 - \theta_1) = (m_1 + 2m_2) gl_1 \sin \theta_1 + 2m_2 l_1 l_2 \dot{\theta}_2^2 \sin (\theta_2 - \theta_1) \tag{169}
\]

\[
-m_2 \ddot{\theta}_2 + 2m_2 l_1 l_2 \ddot{\theta}_1 \cos (\theta_2 - \theta_1) + \frac{4}{3} m_2 l_2 \dot{\theta}_2^2 = m_2 gl_2 \sin \theta_2 - 2m_2 l_1 \dot{\theta}_1^2 \sin (\theta_2 - \theta_1) \tag{170}
\]

The above equations can be solved for \( \ddot{x}, \ddot{\theta}_1 \) and \( \ddot{\theta}_2 \), to form a state-space representation.
0.2 MATLAB files

MATLAB scripts for the simulation of the double inverted pendulum

dbroom.m:

Main scripts that designs and graphically implements linear control of the double pendulum. Just type "dbroom" from MATLAB to run it. The script can be edited to change initial conditions and model specifications. The script designs the control law for the current model specifications. Functions from the controls toolbox are necessary.

Functions called by dbroom.m

dbroom_figs.m - Plots responses of simulation
dbroom_init.m - Initializes graphics windows
dbroom_lin.m - Linearizes the nonlinear model for given parameters
dbroom_plot.m - Graphic plot of system for current state
dbroom_sim.m - Nonlinear state-space simulation of double pendulum
dbroom_view.m - Graphic prompt for viewing figures

broom.ps: system definitions and dynamics.

Eric A. Wan
ericwan@ece.ogi
Oregon Graduate Institute

Snapshot of MATLAB window:
% Double Inverted Pendulum with Linear Controller

% Eric Wan, Mahesh Kumashikar, Ravi Sharma
% (ericwan@eap.ogi.edu)

% PENDULUM MODEL SPECIFICATIONS

Length_bottom_pendulum  = .5;
Length_top_pendulum     = 0.75;
Mass_of_cart            = 1.5;
Mass_of_bottom_pendulum = 0.5;
Mass__top_pendulum      = 0.75;

pmodel = [Length_bottom_pendulum Length_top_pendulum Mass_of_cart
Mass_of_bottom_pendulum Mass__top_pendulum];

% INITIAL STATE

Cart_position = 0;
Cart_velocity = 0;
Bottom_pendulum_angle   = 20;  % degrees  % +- 9 max
Bottom_pendulum_velocity = 0;
Top_pendulum_angle      = 20;  % degrees
Top_pendulum_velocity   = 0;

% Random starting angles for fun
%Bottom_pendulum_angle   = 15*(2*(rand(1)-.5));
%Top_pendulum_angle      = 15*(2*(rand(1)-.5));

u = 0;  % initial control force

deg = pi/180;
x0 = [Cart_position Cart_velocity Bottom_pendulum_angle*deg
Bottom_pendulum_velocity*deg Top_pendulum_angle*deg Top_pendulum_velocity*deg];

% TIME SPECIFICATIONS

final_time = 10;  % seconds
dt= 0.01;  % Time for simulation step
DSample = 2;  % update control every Sample*dt seconds

T = dt:dt:final_time;
steps = length(T);
plotstep = 4;  % update graphics every plotstep time steps

%To record positions and control force.
X = zeros(6,steps);
U = zeros(1,steps);

% setup beginning of demo
dbroom_init

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DESIGN LINEAR CONTROLLER
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Obtain a linear state model
[A, B, C, D] = dbroom_lin(pmodel, 0.001);

% form discrete model
[Phi, Gam] = c2d(A, B, DSample*dt);

% Design LQR

Q = diag([50 1 100 20 120 20]);
K = dlqr(Phi, Gam, Q, 1);

% Closed loop poles
% p = eig(Phi - Gam*K)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Simulate controlled pendulum
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Plots initial pendulum state
fall = dbroom_plot(x0, u, 0, f1, 1, pmodel);
drawnow;

fprintf('\n Hit Return to begin simulation \n')
pause

x = x0'; i = 1;

while (i<steps) & (fall ~= 1),
    i = i+1;

    % Update Control force
    if rem(i, DSample) <= eps, % Apply control once in DSample time steps.
        u = -K*x; % Linear control law
    end

    % UPDATE STATE
    % state simulation based on Lagrange method
    x = dbroom_sim(x, u, pmodel, dt);

    % PLOT EVERY plotstep TIMESTEPS
    if rem(i, plotstep) <= eps
        fall = dbroom_plot(x, u, i*dt, f1, 0, pmodel);
    end
% SAVE VALUES
X(:,i) = x;
U(:,i) = u;

end

% graph responses
dbroom_view