Lecture 2: Preliminary

Changliu Liu
Preliminary

• Basic robotics
• Basic closed-loop control
• Basic system analysis
• Basic optimization
• Basic machine learning
Robotics

• Types of robots

• Key components

• Mathematical models
Types of Robots

- Fixed-base robot
  - Sequential manipulator
  - Parallel manipulator
- Mobile robot
  - Legged robot
  - Wheeled robot
  - Mobile manipulator
- Humanoid
Key Components

- Power conversion unit
- Linkage / Base
- Actuator
- Controller
- Sensor (vision, force, proximity, tactile, etc)
- User Interface
Mathematical Models

- Cartesian space
- Configuration space
- State space

Example: SCARA robot

\[ x = [\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2] \]
Mathematical Models

From configuration space to a point on the robot
(Forward kinematics)

\[
[ l_1 \cos(\theta_1) + l_2 \cos(\theta_1+\theta_2),
  l_1 \sin(\theta_1) + l_2 \sin(\theta_1+\theta_2) ]
\]

This mapping is neither injective, nor surjective!
(Backward kinematics are not unique)
Mathematical Models

From configuration space to the area occupied by the robot

This mapping is injective, but still not surjective!
Mathematical Models

• A general way to encode the mapping from the configuration space to the Cartesian space: **D-H Parameters**

• For each joint, we assign a reference frame:
  
  • **z-axis:** in the direction of the joint axis
  
  • **x-axis:** parallel to the common normal \( x_n = z_n \times z_{n-1} \)
  
  • **y-axis:** follow from right handed coordinate system
D-H Parameters

- Transformation parameters from one joint to another:
  - $d$: offset along previous z-axis to the common normal
  - $\theta$: angle about previous z-axis from old x to new x
  - $r$: length of common normal
  - $\alpha$: angle about common normal from old z-axis to new z-axis
Example: 6 DOF Industrial Robot Arm

<table>
<thead>
<tr>
<th>θ</th>
<th>d</th>
<th>r</th>
<th>α</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.05</td>
<td>-π</td>
</tr>
<tr>
<td>-π</td>
<td>0</td>
<td>0.44</td>
<td>2π</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0.035</td>
<td>-π</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>π</td>
</tr>
<tr>
<td>0</td>
<td>-0.42</td>
<td>0</td>
<td>π</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-π</td>
</tr>
<tr>
<td>0</td>
<td>-0.08</td>
<td>0</td>
<td>2π</td>
</tr>
</tbody>
</table>

These are configuration space variables.
Example: 6 DOF Industrial Robot Arm

https://github.com/changliuliu/RobotKinematics

This script is to illustrate robot kinematics.

Step 1: Choose a robot

```matlab
robot = robotproperty(M16iB)
theta = robot.DH(:,1);
```

Step 2: Choose joint angles (offset in rad)

```matlab
robot.DH(:, 1) = theta + [
    -0.05
    -0.06
    0.43
    -0.02
    -0.21
    -0.28
];
```

Step 3: Now let's draw the robot!

```matlab
Draw(robot, CAD);
```
Robot Dynamic Model

- Different levels of abstraction

- First-order model

\[ x = \theta, u = \dot{\theta} \Rightarrow \dot{x} = u \]

- Second-order model

\[ x = [\theta, \dot{\theta}], u = \ddot{\theta} \Rightarrow \dot{x} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ I \end{bmatrix} u \]

- Torque model

\[ x = [\theta, \dot{\theta}], u = \tau \Rightarrow \dot{x} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ M(\theta)^{-1} \end{bmatrix} (u - C(x)) \]
Useful Resources

- VideoLecture by Oussama Khatib: http://academicearth.org/courses/introduction-to-robotics http://www.virtualprofessors.com/introduction-to-robotics-standford-cs223a-khatib
  (focus on kinematics, dynamics, control)

- Oliver Brock’s lecture http://courses.robotics.tu-berlin.de/mediawiki/index.php/Robotics:_Schedule_WT09

- Stefan Schaal’s lecture Introduction to Robotics: http://www-clmc.usc.edu/Teaching/TeachingIntroductionToRoboticsSyllabus
  (focus on control, useful: Basic Linear Control Theory (analytic solution to simple dynamic model → PID), chapter on dynamics)


- CMU lecture “introduction to robotics” http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/16311/www/current/syllabus.html
  (useful: PID control, simple BUGs algorithms for motion planning, non-holonomic constraints)

- Handbook of Robotics (partially online at Googlebooks) http://tiny.cc/u6tz

- LaValle’s Planning Algorithms http://planning.cs.uiuc.edu/
Control

• Feedback control system

• Control Methods

• Feedforward control system
Feedback Control System

- A basic feedback control system architecture
Control Theory

• Classic control theory
  • Frequency domain analysis using Laplace transform.

• Modern control theory
  • Time domain state space analysis
  • Linear control / Nonlinear control
  • Optimal control / Model predictive control
  • Adaptive control
  • Robust control
  • Hybrid control
The internal **state variables** $x$ are the smallest possible subset of system variables that can represent the entire state of the system at any given time, such that the future evolvement of the system only depends on the current state, not the past.

<table>
<thead>
<tr>
<th></th>
<th>Deterministic Model</th>
<th>Stochastic Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Continuous time</strong></td>
<td>$\dot{x} = f(x,u)$</td>
<td>$\dot{x} = f(x,u,w)$</td>
</tr>
<tr>
<td></td>
<td>$y = h(x,u)$</td>
<td>$y = h(x,u,v)$</td>
</tr>
<tr>
<td><strong>Discrete time</strong></td>
<td>$x_{k+1} = f(x_k,u_k)$</td>
<td>$x_{k+1} = f(x_k,u_k,w_k)$</td>
</tr>
<tr>
<td></td>
<td>$y_k = h(x_k,u_k)$</td>
<td>$y_k = h(x_k,u_k,v_k)$</td>
</tr>
</tbody>
</table>
State Space Model

\[
x_{k+1} = f(x_k, u_k, w_k)
\]
\[
y_k = h(x_k, u_k, v_k)
\]

We need to design this mapping to achieve certain control objectives
Control Objectives

• Stabilization at certain state (Regulation)
  • e.g., inverted pendulum

• Trajectory tracking
  • e.g., industrial robot

• Multi-objective control
  • e.g., autonomous driving
# Linear and Nonlinear Control

<table>
<thead>
<tr>
<th>Linear System</th>
<th>Nonlinear System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Control Law</td>
<td>Very mature</td>
</tr>
<tr>
<td>Nonlinear Control Law</td>
<td>Well studied</td>
</tr>
</tbody>
</table>
Optimal Control

• Basic form:

\[
\min_u J(x, u) = \int_0^T l(x(t), u(t)) dt + r(x(T), u(T)) \\
\text{s.t.} \quad \dot{x} = f(x, u) \\
\quad \phi(x, u) \leq 0
\]

• Similarity between Markov Decision Process

• Two ways to analytically obtain \( u = g(x, t) \):
  
  • Calculus of variation
  
  • Dynamic Programming

• Special case: Linear Quadratic Regulator (LQR)
Model Predictive Control

• Solve the optimal control problem in a open-loop fashion online

• Also known as receding horizon control

\[
\min_u J(x, u) = \int_0^T l(x(t), u(t))dt + r(x(T), u(T)) \\
\text{s.t. } \dot{x} = f(x, u) \\
\phi(x, u) \leq 0
\]
Adaptive Control

• A control method that adapts to the parameters of the controlled system

• Can be combined with other control approaches, e.g., adaptive optimal control, adaptive MPC, adaptive robust control, etc.

• Indirect adaptive control: first identify the parameters of the controlled system, then adapt the parameters of the controller

• Direct adaptive control: directly adapt the parameters in the controller
Indirect adaptive control is similar to model-based reinforcement learning.
System Identification

System ID is similar to imitation learning
Direct Adaptive Control

Direct adaptive control is similar to model-free reinforcement learning.
Feedforward Control

Typical feedforward control methods: input shaping, iterative learning control (ILC)
* Note the similarity to reward shaping in reinforcement learning
Useful References

Linear Quadratic Optimal Control

- Bryson and Ho, Applied Optimal Control: Optimization, Estimation, and Control, Wiley

Stochastic Control Theory and Optimal Filtering

- Lewis and Xie and Popa, Optimal and Robust Estimation, Second Edition CRC
- Grewal and Andrews, Kalman Filter, Theory and Practice, Prentice Hall
- Astrom, Introduction to Stochastic Control Theory, Dover Books on Engineering (paperback), New York, 2006

Adaptive Control

- Astrom and Wittenmark, Adaptive Control, Addison Wesley, 2nd Ed., 1995
- Goodwin and Sin, Adaptive Filtering Prediction and Control, Prentice Hall, 1984
- Krstic, Kanellakopoulos, and Kokotovic, Nonlinear and Adaptive Control Design, Willey
System Analysis

- Controllability and Observability
- Stability and Region of Attraction
- Reachability
Controllability and Observability

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

• Controllability: the ability of the control input \( u \) to move the internal state \( x \) of a system from any initial state to any other final state in a finite time interval.

• Observability: for any possible sequence of state \( x \) and control \( u \), the current state \( x_0 \) can be determined in finite time using only the outputs \( y \).

• Controllability and observability are dual aspects of the same problem.

• Internal dynamics: dynamics that are neither controllable nor observable.

• Controllability and observability are independent from stochasticity and other uncertainties.
Examples

Controllable, Observable

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

Controllable, Unobservable

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

Uncontrollable, Observable

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

Uncontrollable, Unobservable

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

* The first state’s dynamics are internal
Open-Ended Questions

• Any example of real-world uncontrollable / unobservable systems? Is human-robot system controllable / observable from robot’s perspective?

• For uncontrollable / unobservable systems, how to ensure safety? How to determine responsibility?

• How to modify the system to improve controllability / observability?
Closed-Loop Analysis

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

\[
\dot{x} = f(x, g(x)) = f^*(x)
\]

1. Whether the system will rest in the target state?
   - Equilibrium
   - Stability

2. From where the target state can be reached?
   - Region of attraction

3. Whether unsafe region will be reached?
   - Reachability
Closed-Loop Analysis

\[ \dot{x} = f^*(x) \] defines a vector field

**Equilibrium:**
A state that satisfies that
\[ \dot{x}^* = f^*(x^*) = 0 \]

**Stable equilibrium**
**Unstable equilibrium**

**Real-world equilibria:**
- trajectory tracking error to 0
- system resting in static
Stability and Region of Attraction (ROA)

- (Local) stability of an equilibrium
  - The system will eventually go to the equilibrium if it is close to the equilibrium.
  - Stability can be defined in other ways.
- Region of attraction of an equilibrium
  - The set of initial states under which the system will eventually reach the equilibrium.
Reachability

• Reachability from a set of initial states
  • The set of states that can be reaches along the system trajectories from the initial states.
  • The system is **safe** if the reachable set is separated from the unsafe set.
Optimization

• Characterizations of optimization problems involved in this course

• Optimization algorithms

• Optimization solvers
Optimization Problems

• Control:
  • Optimal control / MPC

• Learning:
  • System identification / Function approximation

• System Analysis:
  • Reachability
Characteristics of Optimization Problems

• Smoothness (differentiability)

• Linearity and convexity

• Real variables or integer variables

• Dimensionality
Optimization Algorithms

\[
\min J(x)
\]

- Gradient descent / Stochastic gradient descent

\[
x' \leftarrow x - \alpha \nabla J(x)
\]

- Newton’s method

\[
x' \leftarrow x - \alpha [\nabla^2 J(x)]^{-1} \nabla J(x)
\]

- Sequential convexification

- For non-convex \( J(x) \), first approximate it as a convex function \( \tilde{J}(x) \), then use methods for convex optimization
Characterizations of Solutions

- Optimality
  - KKT condition (First order necessary condition)
  - Local optima v.s. global optima

- Feasibility
  - Satisfying hard constraints
Optimization Solvers

- GUROBI: General optimization solver. https://www.gurobi.com
- CERES: Nonlinear least square solver. ceres-solver.org
- QPOASES: Quadratic programming solver. https://projects.coin-or.orgqpOASES
- OOQP: Quadratic programming solver. pages.cs.wisc.edu/~swright/ooqp/
- ACADO: Toolkit for automatic control and dynamic optimization. acado.github.io
Machine Learning

• Types of learning:
  • Supervised Learning
  • Unsupervised Learning
  • Reinforcement Learning

• Components in a learning algorithm:
  • Function representation
  • Function optimization
  • Function evaluation
Machine Learning

Conventional Approach

Machine Learning
Supervised Learning

• Given samples in input x and output y, learn a mapping f from x to y

• Example: system identification, imitation learning, etc.

• Minimize a loss function $f = \arg\min l(y, f(x) \mid x^{(i)}, y^{(i)})$

  $l(y, f(x) \mid x^{(i)}, y^{(i)}) = \sum_{i} \| y^{(i)} - f(x^{(i)}) \|^2$

• Least square problem:

• Adversarial learning: $l(y, f(x) \mid x^{(i)}, y^{(i)}) = \max_{d} d(y, f(x) \mid x^{(i)}, y^{(i)})$
Unsupervised Learning

• Given samples in input $x$, and a metric defining similarity among the samples (usually Riemannian metric), learn a mapping $f$ such that $f(x)$ almost preserves the metric.

• Examples:
  
  • Principle component analysis (PCA)
  
  • Clustering
Reinforcement Learning

• Given samples of input $x$, output $y$, next input $x'$, and the corresponding reward $r(x, y, x')$, learn a function $f$ that maximizes the reward

$$f = \arg \max Q(x, f(x))$$

$$Q(x, y) = \sum_{x'} [r(x, y, x') + \delta \sum_{y'} Q(x', y')]$$

• Note the similarity between adaptive control
Function Representation

- Numerical functions
  - Linear regression
  - Neural networks
  - Support vector machines
- Symbolic functions
  - Decision trees
  - Rules in propositional logic
  - Rules in first-order predicate logic
- Instance-based functions
  - Nearest-neighbor
  - Case-based
- Probabilistic graphical models
  - Naive Bayes
  - Bayesian networks
  - Hidden-Markov Models (HMM)

Slide credit: Ray Mooney
Algorithms for Function Search/Optimization

- Gradient descent
  - Backpropagation
- Dynamic programming
  - HMM learning
- Divide and Conquer
  - Decision tree induction
  - Rule learning
- Evolutionary Computation
  - Genetic Algorithms (GAs)
  - Genetic Programming (GP)
  - Neuro-evolution
Function Evaluation

- Accuracy
- Cost / Utility
- Robustness
- Satisfiability
Deep Learning

• A branch of machine learning where the function $f$ is parameterized by deep neural networks

• Universal approximation theorem

• (Verifiable) Design of the network structure