Lecture #5: *Linear Dynamical Systems*

Scribes: *Stefan Niculae, Pradeep Kadubandi, Mahalakshmi Subramanian, Daiming Yang*

Contents

1 Recap: Bayesian Networks **2**
1.1 Serial Connection . . . 2
1.2 Diverging Connection . . . 3
1.3 Converging Connection . . . 3
1.4 Example: Multiple Connections . . . 4

2 Recap: The Markov Assumption **5**
2.1 Markov Chains . . . 5
2.2 Second Order Markov Chains . . . 5
2.3 Markov Chains in Robotics . . . 6

3 Linear Dynamical Systems **8**
3.1 Continuous Time . . . 8
3.2 Discrete Time . . . 8
3.3 Example: Pendulum System . . . 9

4 Control Strategies **11**
4.1 Open-Loop Control . . . 11
4.2 Closed-Loop Control . . . 12
4.3 Feedforward and Feedback Control . . . 12
4.4 PID Control . . . 13
4.5 Example: Simple Pendulum Control . . . 14

1 Recap: Bayesian Networks

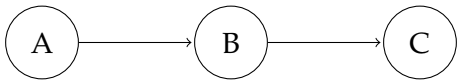
This section reviews conditional independence properties in Bayesian Networks considering different types of causal relationships:

1. Serial
2. Diverging
3. Converging

and a composite example of two kinds of connections.

1.1 Serial Connection

Sequential nodes, where each node depends on the previous one:



State Description:

A: battery dead

B: robot won't start

C: no sound from the motor

In the above serial connection, A causes B which in turn causes C.

If we observe that battery is dead (A), we are going to infer with more confidence that robot has not started (B) and thus we will also assign higher probability for hearing no sound from motor (C). The opposite is also true: When we observe no sound from motor (C), we are going to believe that robot has not started (B) and we might infer that battery could be dead (A).

However, if we observed that robot did not start (B), knowing that battery is dead (A) is not going to effect the probability of not hearing sound from the motors (C) because we already accounted for the cause (B) in our belief.

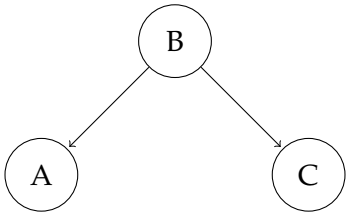
That is, given information about B, knowing more about A won't give any new information about C, and vice versa.

$\Rightarrow$ Given B, A and C are conditionally independent. Putting it mathematically,

$$P(A, C|B) = P(A|B)P(C|B)$$

1.2 Diverging Connection

Two child (A, C) nodes depending on a parent node (B):



State Description:

A: light on/off

B: battery low/high

C: robot moving/stopped

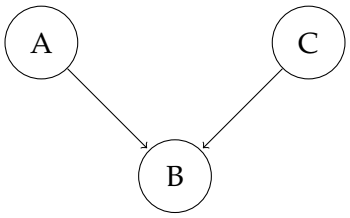
Let us assume that we do not know the level of Battery. Then receiving information about whether the light is on/off (A) will influence our belief about Battery(B). The updated belief about the state of Battery(B) will in turn make us update our belief about the state of Robot if it is moving or stopped(C). The opposite case (i.e., knowing if Robot is moving or stopped) will lead to a similar conclusion.

However, if we know that the battery is low (B), checking if the light is off (A) doesn't give any more information about the robot's motion (C)

$\Rightarrow$ Given B, A and C are conditionally independent.

1.3 Converging Connection

A child (B) node being dependent on two parent nodes (A, C):



State Description:

A: battery dead

B: robot doesn't move

C: wheels damaged

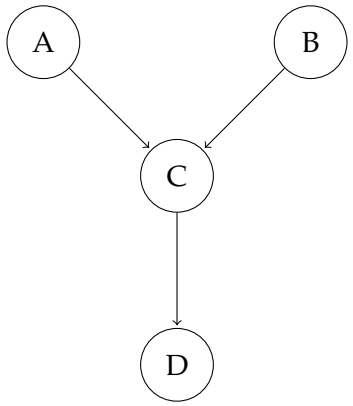
In general, the states of robot's battery (A) and wheels (C) are independent. However, if we observe that the robot doesn't move (B), checking if the battery is dead (A) affects the probability of wheels being damaged (C).

$\Rightarrow$ Given B, A and C are conditionally dependent. Mathematically,

$$P(A, C) = P(A)P(C), \text{ however, } P(A, C|B) \neq P(A|B)P(C|B)$$

1.4 Example: Multiple Connections

A 4-node network with converging and serial connections:



$$\begin{aligned} P(A, B, C, D) &= P(A = true, B = true, C = true, D = true) \\ &= P(D \mid A, B, C)P(A, B, C) \\ &= P(D \mid C)P(C \mid A, B)P(A, B) \\ &= P(D \mid C)P(C \mid A, B)P(A)P(B) \end{aligned}$$

Rule (in Bayesian Network): Each node is conditionally independent of its non-descendants, given its parents. In the above simplification, D is conditionally independent of A and B given C.

Note: $X \rightarrow Y$ means that the X (parent) is cause and Y (child) is effect.

2 Recap: The Markov Assumption

This section reviews the Markov property and Markov chains, frequently used in Bayesian Network-based Robotics.

2.1 Markov Chains

The probability of moving to next state depends only on the current state i.e., it depends on only one previous state. This is called first order Markov chain.

Figure 1: First order Markov chain

Based on first order Markov assumption, to compute the probability of next state given its history,

$$P(x_t|x_0 : t-1) = P(x_t|x_{t-1})$$

The probability of the next state depends only on the current state and not on any other histories. This is the transition model that specifies how the state changes next based on the current one.

2.2 Second Order Markov Chains

If the assumption is made such that the next state depends on the current state and one state before the current state, the equation of joint probability changes as below.

$$P(x_t|x_0 : t-1) = P(x_t|x_{t-1}, x_{t-2})$$

This is a second order Markov chain. The below diagram shows the Bayesian network representation for second order Markov chain.

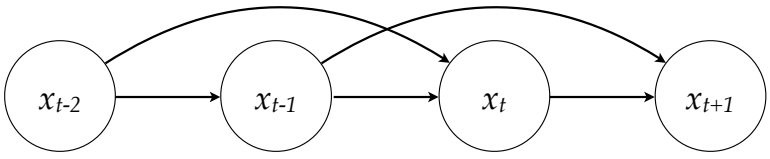


Figure 2: Second order Markov chain

To convert second order Markov chain to first order, create a new composite state that contains x at two time steps (refer the diagram below).

$$P(x_t, x_{t-1}|x_{t-1}, x_{t-2}) = p(x_t|x_{t-1}, x_{t-2})$$

Since x_{t-1} is assumed to be given, it can be removed from the inference.

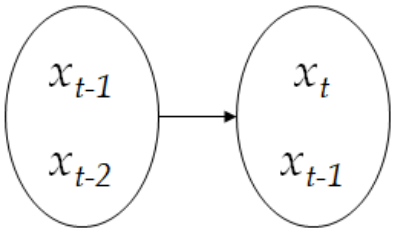


Figure 3: Markov chain with two-timestep dependencies

2.3 Markov Chains in Robotics

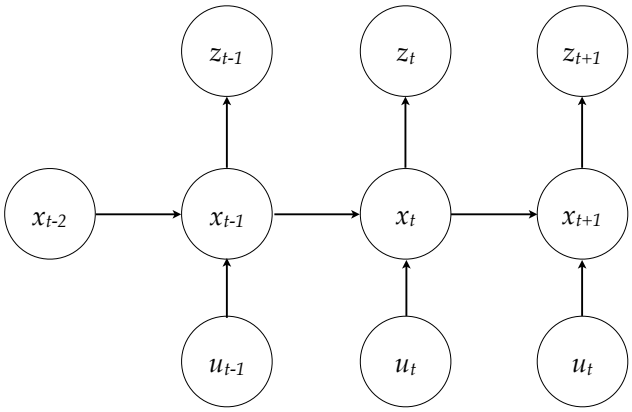
When we represent the environment / robot states using first order Markov chain, we usually assume that the rules based on which the robot moves does not change over time. This is called the **stationary assumption**. This means that $P(x_{t+1}|x_t)$ is going to stay the same as time progresses.

In a robot, this might not be true for longer timescales because as the time goes the tyres of the robot can get worn out or motors can get damaged. However, the stationary assumption is reasonable over short periods of time and practical for most applications.

If there are control inputs u_t and consider the next state of the system depends not only on the current state but also on the control input applied at that time t, the probability equation of 1st order Markov assumption is modified as $P(x_t|x_{t-1}, u_t)$

Also, the observation depends on the current state and not on the control inputs or any previous states. Given the current state x_t, the observation of the current state z_t is conditionally independent of the previous states and control inputs $P(z_t|x_t)$.

The new Bayesian Network including the control inputs and the observations looks like below:



[1]

Figure 4: Bayesian Network with control inputs and state-observation dependencies

Additional material: Bayesian Networks by MIT

3 Linear Dynamical Systems

This section presents Linear Dynamical Systems using continuous and discrete time and an example using a simple pendulum.

Note: in this course, we will use mostly Discrete time. Continuous time is presented just for the sake of completion.

3.1 Continuous Time

Figure 5 shows the block diagram of the Continuous Dynamical System. The inputs are fed to the Robot Dynamics that calculates the change of state given the input (derivative $\dot{x}$). The actual state is computed by integrating these change of states.

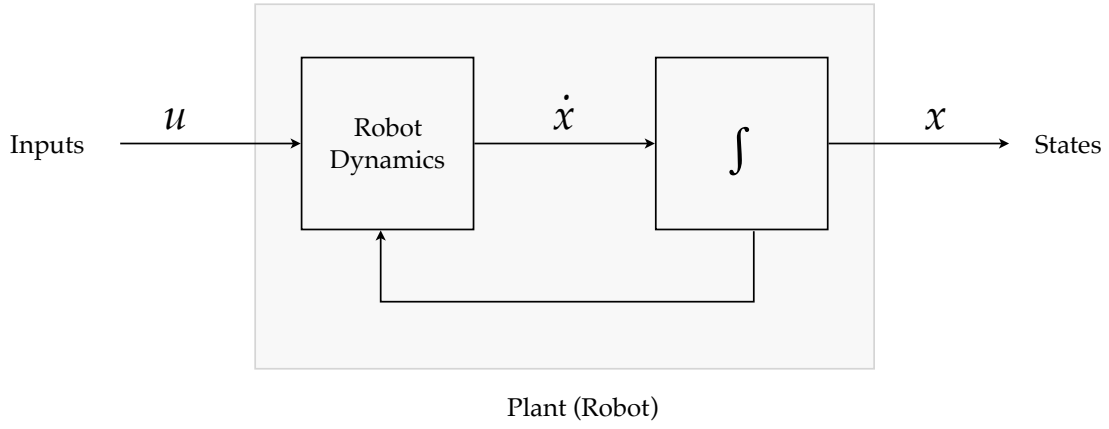


Figure 5: Block diagram of the continuous dynamical system

$\dot{x} = f(x, u, t) \leftarrow$ equations of dynamics
$y = g(x, u, t) \leftarrow$ output of the system (or observation)

3.2 Discrete Time

Figure 6 shows the block diagram of the Discrete Dynamical System. In Discrete Dynamical System, given the inputs U, the Robot Dynamics that computes change of state of the robot and directly outputs the state of next time step.

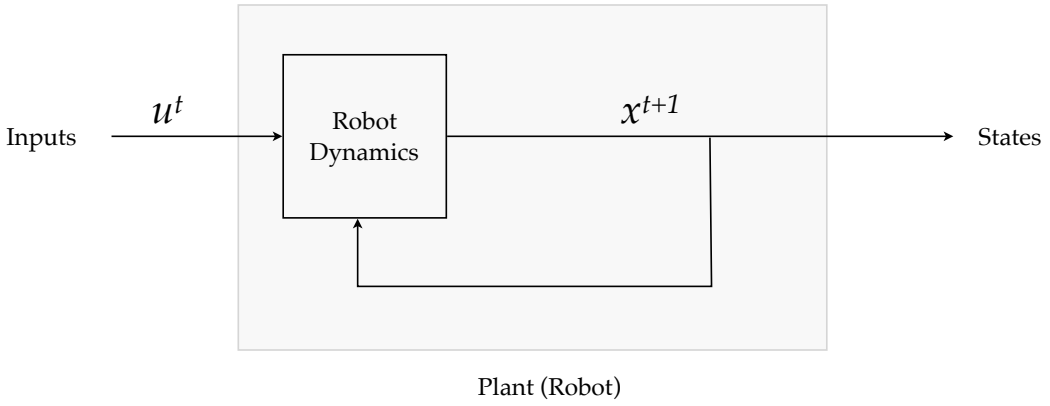


Figure 6: Block diagram of the discrete dynamical system

$x^{t+1} = f(x^t, u^t, t)$ ← equations of dynamics
$y^t = g(x^t, u^t, t)$ ← output of the system (or observation)

Note that the third argument, time t, is added in general, because behaviors of the system may change with respect to time. When the system is assumed to be stationary (or time-invariant), the third argument t can be omitted.

3.3 Example: Pendulum System

This simple example illustrates the continuous time dynamics.

Shown in the Figure 7 is the free body diagram of a simple pendulum. The point mass at the end of the rod connects to the fixed point. A motor applies a torque which induces a change in the rod's angle. Related parameters include:

- l: the length of the rod,
- m: the mass on the endpoint of the rod,
- θ: the angle of the rod (shown in the positive direction),
- τ: a torque applied by a motor to the rod,
- I: the moment of inertia of the mass with respect to the rotating center, ($I = ml^2$)
- g: the gravitational acceleration, with assumed uniform gravitational field.

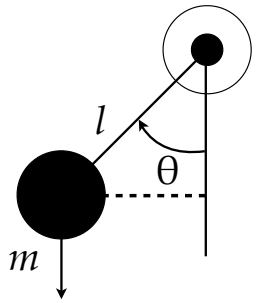


Figure 7: Free body diagram of a simple pendulum

From the Newton's second law of motion, the equation of dynamics is:

$$I\ddot{\theta} = -mgl\sin\theta + \tau \tag{1}$$

which can be substituted and simplified into:

$$\ddot{\theta} = -\frac{g}{l}\sin\theta + \frac{\tau}{ml^2} \tag{2}$$

The inertial torque $I\ddot{\theta}$ represents how much an object resists when pushed. The larger the torque, the more it resists. The reason it contains a minus sign is because when the pendulum tries to move up, gravity applies downward force.

Note that, the gravitational force moves the mass counterclockwise, when the positive direction is clockwise, thus the minus sign.

4 Control Strategies

Figure 8 shows the block diagram of a general control strategy for robots. The general equation for control signal is $u = \Pi(x, \alpha, t)$, where

- $\Pi()$ is the policy,
- x is the state,
- α is the parameter set,
- t is the time.

The desired behavior is a part of the parameter set (α).

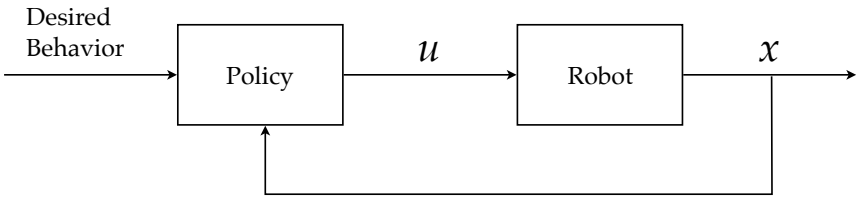


Figure 8: Block Diagram of a General Control Strategy

For the pendulum example, state (x) is the angle (θ) and the angular velocity ($\dot{\theta}$). The desired behavior can be the goal angle (simplest case), a trajectory to follow, or even a reward (most complicated).

4.1 Open-Loop Control

Open-loop control does not check for feedback from the world when executing a plan.

Figure 9 is the high-level block diagram of a open-loop control system. By this configuration, the control signal u doesn't depend on the current state x, so $u = \Pi(\alpha, t)$.

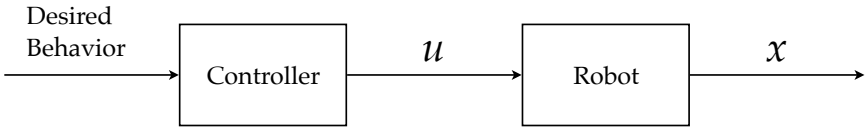


Figure 9: Block Diagram of a Open-Loop System

Open-loop systems are clearly much faster and computationally cheaper than closed-loop systems (section 4.2). For example, to flicker the lights, we just need to turn the switch on and off, knowing that the lights will react accordingly. It's excessive to check whether the lights are on or off after each switch. In the field of robotics, the designer may count the total steps first, and just let the robot move the certain number of steps without sensing.

Another example of an open-loop control: a person wants to reach the window in a room, starting from the door. They estimate the distance, approximate the number of steps needed, close their eyes and start walking. The person only checks where they ended up walking the planned number of steps.

Due to faster computation, open-loop systems are useful for reactive behaviors. For example, humans can instinctively crawl down when scared to protect vital organs.

4.2 Closed-Loop Control

Closed-loop control continuously check for feedback from the world while executing a plan, and, potentially adjust the plan accordingly mid-execution.

Figure 10 is the high-level block diagram of a closed-loop control system. This is the same configuration as in the general control shown before. Most controllers need to constantly check the current state (x) and adjust the control signal (u).

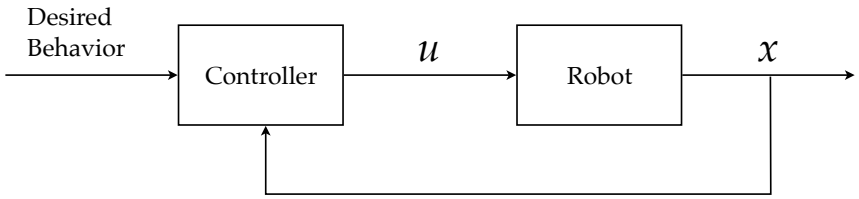


Figure 10: Block Diagram of a Closed-Loop System

4.3 Feedforward and Feedback Control

There are two variations of closed loop systems : feed-forward controller and feedback controllers, they can be usually combined in a single system as well.

Figure 11 shows how to combine a feedforward controller with a feedback controller. Both controllers can function simultaneously, and the combined control signal ($u_f + u_b$) will transmit into the robot.

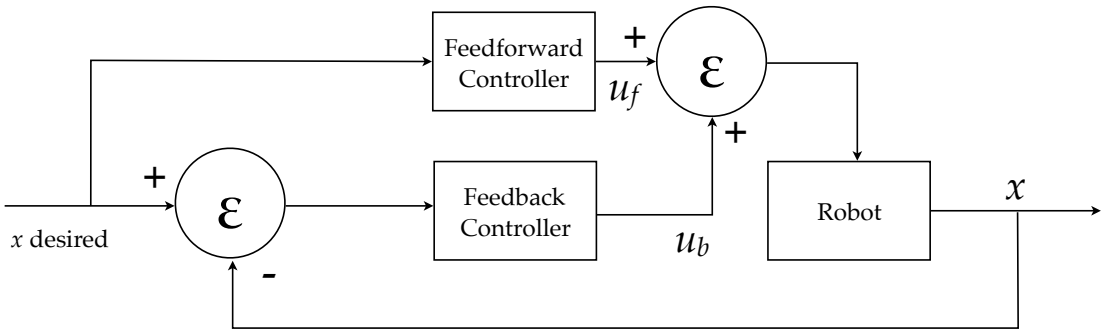


Figure 11: Block Diagram of a Combined System

A real-world example, starting a car:

- when starting the car, I just press the gas pedal (feedforward);
- after a couple of seconds, I compare the current speed with the desired one, and press the pedal accordingly (feedback).

4.4 PID Control

PID controllers are very effective and popular in the field of dynamic control. The algorithm focuses on the error of the current state $(x_{des}(t) - x(t))$ and tries to reduce the error to zero.

1. Proportional term: $u_P = K_P(x_{des}(t) - x(t))$

   The P controller is the most direct controller. When the error gets larger, the control signal also becomes larger to correct the error. As shown, the basic version of P controller constitutes a linear relationship between the error $(x_{des}(t) - x(t))$ and the control signal (u_P). However, the relationship can even be quadratic or piecewise to better control the system.

   The problems of P controller alone are that (1) when external forces exist, the error cannot be eliminated, and (2) due to inertia or discrete time, the system may oscillate and diverge if K_P is too large. Most noticeably, all systems in this course are discrete, so we have to choose K_P wisely.

2. Derivative term: $u_D = K_D(\dot{x}_{des}(t) - \dot{x}(t))$

   The derivative term is also called the damping term. The derivative term tries to reduce oscillations (or damping) by predicting the next position and preventing overshoot.

3. Integral term: $u_I = K_I \int_{\tau=0}^{\tau=t} (x_{des}(t) - x(t))dt$

   The integral term can account for a sustained error, like the torque introduced by gravity. When the state is steady and close to the target position, both P and D controllers will tend to become zero, while the I controller can counter the sustained error.

Note that, when the state (x) is a scalar, all three gains (K_P, K_I, K_D) are also scalars. If the state is a vector ($\mathbf{x}$), then all three gains become matrices, usually diagonal matrices.

4.5 Example: Simple Pendulum Control

We apply a controller to the simple pendulum presented previously. Figure 12 shows the diagram of the same pendulum.

We would like to find out the value of torque (τ) that will bring the mass to the desired angle (x_d) and stop it from moving ($\dot{\theta} = 0$).

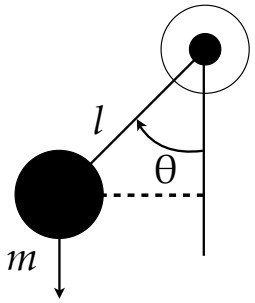


Figure 12: Free body diagram of a simple pendulum

States (angle and velocity):

$$x_1 = \dot{\theta} \qquad\qquad x_2 = \theta$$

Desired values (given angle and zero velocity):

$$x_{1d} = 0 \qquad\qquad x_{2d} = x_d$$

Equations of dynamics (PD Controller):

$$\dot{x_1} = -\frac{g}{l}\sin(x_2) + \frac{\tau}{ml^2} = -\frac{g}{l}\sin(x_2) + \frac{K_p(x_d - x_2) + K_d(0 - x_1)}{ml^2}$$
$$\dot{x_2} = x_1$$

The controller features a proportional and a derivative term and specifies how much torque τ should be applied to bring the pendulum from the current position and velocity to the desired ones.
At equilibrium:

$$\dot{x}_2 = 0 \Rightarrow x_1 = 0$$
$$\Rightarrow \dot{x}_1 = 0 = -\frac{g}{l}\sin(x_2) + \frac{K_p(x_d - x_2)}{ml^2}$$

Because $\sin(x_2)$ is non-linear, it's hard to solve for x_2. Therefore, we also assume that the angle x_2 is small. Under the small-angle assumption, $\sin(x_2) = x_2$, as shown in the Figure 13.

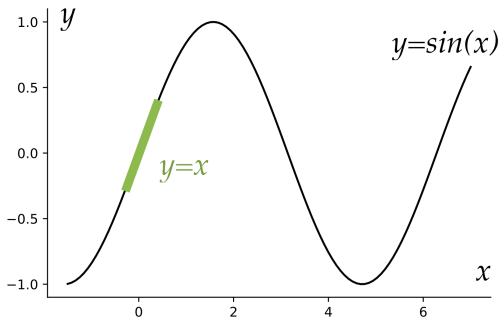


Figure 13: Small-Angle Assumption for the Sine Function

Then, the equation becomes:

$$0 = -\frac{g}{l}x_2 + \frac{K_p(x_d - x_2)}{ml^2}$$
$$\Rightarrow x_2 = \frac{K_p x_d}{K_p + mlg}$$

Noticeably, all m, l, and g are non-zero, otherwise the question becomes meaningless. Then, after the system runs, x_2 will always be less than x_d. Theoretically, K_P can be arbitrarily large, so x_2 can approach x_d. However, the real-world control are discrete, and the power is limited, so the system will be unstable or impractical for a large K_P.

To counter the constant gravity, the I controller is introduced. We assume the derivative of the integral of angles is the difference between x_d and x_2. So

$$\dot{x}_3 = x_d - x_2$$

The linear equation of dynamics now becomes:

$$\begin{pmatrix} \dot{x_1} \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 & -g/l & 0 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + \begin{pmatrix} \tau/ml^2 \\ 0 \\ x_d \end{pmatrix}$$

where

$$\tau = K_p(x_d - x_2) + K_D(-x) + K_I x_3$$

Note in the above equation that at equilibrium (i.e., when system is in a stable state), $\dot{x_3} = 0$, and the accumulated sum of error x_3 is non-zero, so there is a non-zero torque to keep the system in stable state.

Additional material: Stefan Schaal's USC Robotics lectures from previous years.