Flexible Link Manipulators: Modelling and Control

Caglar Uyulan

https://orcid.org/0000-0002-6423-6720

Part III

Control Strategies (Introduction, Basic Definitions, Concepts, and Example)

In this part, we focus on the intricacies of designing control systems for flexible-link manipulators, which are a subclass of underactuated systems characterized by having fewer control inputs than degrees of freedom.

An underactuated system is one where the number of actuators is less than the number of degrees of freedom to be controlled.

Such systems present unique challenges for control design, primarily because not all the system's states can be directly influenced by control inputs.

A flexible-link manipulator exemplifies this, as its inherent flexibility introduces more degrees of freedom than there are actuators available for direct control.

A significant property of underactuated systems, as highlighted by Spong, is the collocated partial feedback linearization property.

This property emerges from the positive definiteness of the inertia matrix of the system. The inertia matrix $\mathrm{M}(\mathrm{q},\delta)$ for a flexible-link manipulator can be represented as:

$$M(q,\delta) = \begin{bmatrix} M_{11}(q,\delta) & M_{12}(q,\delta) \\ M_{21}(q,\delta) & M_{22}(q,\delta) \end{bmatrix}$$

Where q represents the joint positions and δ signifies the flexural displacements.

The positive definiteness of this matrix is a foundational aspect that allows for the implementation of ***feedback linearization techniques*** in a way that is tailored to the specific dynamics of underactuated systems.

While collocated control, which involves placing sensors and actuators at the same physical locations (typically at the joints), simplifies the design by leveraging the collocated partial feedback linearization property, it has limitations. Specifically, it may lead to a reduction in system bandwidth and suboptimal performance in tracking tasks, especially when the system engages in complex motions or when the end-effector is required to follow a precise trajectory.

To circumvent these limitations, non-collocated control strategies have been proposed. These approaches involve strategic placement of sensors and actuators away from the joints to enhance system bandwidth and improve overall performance, albeit at the cost of increased complexity in the control design.

For a rigid system, the dynamic equations simplify considerably. In the absence of flexure (δ=0), the dynamics of a flexible-link manipulator reduce to:

$$M_{11}(q,0)\ddot{q} + f_1(q,\dot{q}) + h_1(q,0) = u$$

This equation encapsulates the classic dynamics of rigid manipulators, which can be effectively controlled using the computed torque method. The control law for this approach is given by:

$$u = M_{11}(q,0)\left(\ddot{q}_r + K_d(\dot{q}_r - \dot{q}) + K_p(q_r - q)\right) + f_1(q,\dot{q}) + h_1(q,0)$$

where $q_r, \dot{q}_r, \ddot{q}_r$ represents the desired (reference) joint trajectory for the joint positions, velocities, and accelerations, respectively. K_d and K_p are the derivative and proportional control gains. This results in a decoupled system with the closed-loop behavior described by:

$$\ddot{e} + K_d\dot{e} + K_p e = 0$$
$$e = q_r - q$$

where e represents the tracking error. This formulation highlights the robustness of computed-torque control in handling uncertainties in system dynamics for rigid manipulators.

When flexural effects are significant, applying the same control strategy without modification can lead to problems. Specifically, the sensor readings at the joints may include components attributable to structural flexibility, complicating the control task. This necessitates a refined control strategy that can accommodate the flexural dynamics without compromising system stability and performance.

The exploration of control strategies for flexible-link manipulators presents a multifaceted challenge, particularly due to phenomena such as observation and control spillover, which can significantly hinder the system's performance and stability. This complexity arises from the interaction between the manipulator's rigid-body dynamics and its inherent flexible modes.

Observation spillover occurs when sensor readings include effects not directly related to the rigid-body motion, primarily due to the system's flexible dynamics. This phenomenon can lead to instability because of the unintended coupling between the observed states and the control inputs. In essence, the control system, designed primarily around the manipulator's rigid dynamics, inadvertently excites the flexible modes, leading to unpredictable or undesired system behavior.

Control spillover refers to the unintended excitation of flexural modes by a control strategy that is solely based on the system's rigid-body dynamics. This excitation can degrade system performance, even if it does not directly lead to instability. The dynamic equations reveal that certain system modes, due to their positioning on the imaginary axis (jω-axis), are marginally stable and susceptible to being pushed into the unstable region of the s-plane through improper feedback.

A pivotal aspect of controlling flexible-link manipulators is their ***non-minimum phase*** nature, particularly challenging when the objective is to move the end-effector along a desired trajectory. Non-minimum phase systems present a critical hurdle for control design, as attempts to enforce exact trajectory tracking can result in unbounded state trajectories and instability.

This issue underscores the need for control strategies that account for both the rigid and flexible dynamics of the system.

To achieve robust and satisfactory performance, it is imperative to include flexural modes in both sensing and actuation strategies. This necessitates a control scheme that incorporates the dynamic effects of flexibility, ensuring that the equations governing the control law reflect the system's actual behavior, including its flexible motions.

While much of the research has focused on single flexible arms, extending these findings to multi-link manipulators introduces additional complexities. The mass matrix in multi-link systems depends on joint position variables, introducing significant nonlinearities. This contrasts with the single-link case, where the mass matrix primarily depends on deflection variables, allowing for a closer approximation to linear system behavior.

Early experimental efforts aimed at endpoint regulation have led to innovative strategies, such as output redefinition, to mitigate tracking errors. For instance, the concept of the reflected-tip position as a new output has been explored to improve control outcomes. However, achieving stable and accurate control in multi-link systems remains a daunting task, compounded by the non-minimum phase property when focusing on end-effector positions.

The pursuit of effective control strategies has also ventured into **nonlinear control methods** and the application of intelligent control techniques, such as fuzzy logic and neural networks. These approaches offer promising avenues for managing the complex dynamics of flexible-link manipulators, although comprehensive and unbiased comparisons with classical control strategies are still needed to determine their relative efficacy.

For example, the stability of an aircraft in an open-loop configuration necessitates the center of mass being positioned ahead of the center of pressure. This fundamental principle of aircraft design holds a valuable lesson for the design of flexible structure robots. By analogy, ensuring that a robot is designed to inherently minimize control problems suggests a proactive approach to system stability and performance, akin to the preemptive stability considerations in aircraft design.

The inherent flexibility of robots, while contributing to their versatility and efficiency, poses significant control challenges. A pivotal question arises: Is it possible to design the robot such that these control challenges are not merely mitigated but also result in a system that requires simpler control schemes? This design philosophy does not suggest eliminating flexibility—since it serves critical functional requirements—but rather optimizing it to enhance control effectiveness without increasing the total inertia of the robot.

Improving the system behavior through mechanical design involves optimizing the manipulator's link shapes to achieve desirable features, such as low mass, reduced moments of inertia, and higher natural frequencies. This approach is grounded in the intuition that increasing the first structural natural frequency, while maintaining constant inertia, improves system response. This is because, in a rigid system, the first natural frequency is inherently high. Advancements in materials and the integration of distributed actuators and sensors offer substantial benefits.

In addressing the quintessential aspects of control engineering, our discourse focuses on two fundamental paradigms: regulation, also known as stabilization, and tracking, oftentimes referred to as servoing.

These paradigms underscore the foundational objectives in the realm of control systems engineering, aiming to architect strategies that either stabilize a system at a desired equilibrium or enable it to follow a predetermined, time-variant trajectory.

Regulation Problem

The essence of the regulation problem lies in formulating a control law that ensures the system's states converge to and remain stable around a designated equilibrium point. This involves manipulating the system's input in such a manner that, irrespective of its initial condition within a specified domain, the state vector asymptotically approaches a state of equilibrium as time progresses indefinitely.

Mathematically, the regulation problem for a nonlinear dynamic system is articulated through the differential equation:

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

Here, $x \in \mathrm{R}n$ denotes the state vector, $u \in \mathrm{R}m$ represents the control input vector, and t signifies the temporal dimension.

The objective is to discover a control law u that propels the state vector x towards zero as $t \to \infty$, symbolizing the attainment of the desired equilibrium.

Tracking Problem

Tracking Problem focuses on designing a controller that ensures the system's output mimics a given time-varying trajectory. This is more complex, requiring the system's output to follow a predefined path while keeping the entire system's state within acceptable bounds. The tracking problem is defined by the system's dynamics: $\dot{x}$=f(x,u,t) and the output equation y=h(x), alongside a desired output trajectory yd(t). The challenge is to determine a control input u that minimizes the tracking error y(t)−yd(t) to zero, ensuring the state vector x remains bounded.

The control input u can be static or dynamic. A static input depends directly on real-time measurements of the system's signals, while a dynamic input is derived from a set of differential equations that integrate these measurements.

Tracking problems are inherently more complex than regulation problems due to the need to follow a dynamic trajectory while maintaining system stability. Regulation problems are essentially a subset of tracking problems with a constant desired trajectory. In the context of flexible-link manipulators, the tip-position tracking problem is particularly challenging, requiring an understanding of internal dynamics, zero-dynamics, and non-minimum phase characteristics within a nonlinear framework. We consider square nonlinear systems linear in the input (affine systems), described by $\dot{x}$=f(x)+g(x)u and y=h(x).

This setup allows for a comprehensive examination of control system dynamics, strategies, and goals in both regulation and tracking.

Particularly, when discussing output tracking and system stability, it's crucial to focus on the mathematical formalisms that guide our understanding and design of control systems.

We investigate the intricate process of input-output linearization and its implications for control system design, especially focusing on systems with well-defined vector relative degrees.

System Description

Consider a dynamic system where x∈Rn represents the state vector, u∈Rp the control input vector, and y∈Rp the output vector.

The vector fields)f(x) and g(x), along with the output function h(x), are assumed to be smooth over an open subset of Rn. Our objective is to control the system in a way that the output vector y tracks a desired trajectory while ensuring all other states remain within reasonable bounds.

Vector Relative Degree

A key concept in this discussion is the system's vector relative degree, denoted as $r = [r_1, r_2, \dots, r_p]$, which essentially quantifies the direct influence of the input vector on the rate of change of the output vector.

This notion is vital because it allows us to understand at what derivative level the input u begins to explicitly influence each output component yi.

Input-Output Linearization

By applying state-dependent coordinate and input transformations, we can reframe our system into a form that is linear concerning the input-output relationship.

This process, known as input-output linearization, yields a transformed system characterized by:

A set of p integrator chains, each corresponding to an output component and its derivatives up to the (ri−1)th order.

A remaining set of dynamics, referred to as internal dynamics, which are unaffected by these transformations.

Mathematically, the transformed system is represented as follows:

$$\begin{aligned}
\dot{\xi}_i^j &= \xi_{i+1}^j, && \text{for } i \in \{1,\dots,p\}, j \in \{1,\dots,r_i - 1\} \\
\dot{\xi}_{r_i}^i &= u_i, && \text{for } i \in \{1,\dots,p\} \\
\dot{\eta} &= \alpha(\xi,\eta) + \beta(\xi,\eta)u && \\
y_i &= \xi_1^i, && \text{for } i \in \{1,\dots,p\}
\end{aligned}$$

where

$$\xi := \psi(x) = \begin{bmatrix} \xi_1^1 \\ \xi_2^1 \\ . \\ . \\ . \\ \xi_{r_1}^1 \\ \xi_1^2 \\ . \\ . \\ . \\ \xi_{r_2}^2 \\ \xi_1^p \\ . \\ . \\ . \\ \xi_{r_p}^p \end{bmatrix} = \begin{bmatrix} y_1 \\ \dot{y}_1 \\ . \\ . \\ . \\ y_1^{(r_1-1)} \\ y_2 \\ . \\ . \\ . \\ y_2^{(r_2-1)} \\ y_p \\ . \\ . \\ . \\ y_p^{(r_p-1)} \end{bmatrix}$$

Here, ξ represents the transformed state vector related to the output and its derivatives, and η represents the internal dynamics of the system.

Zero-Dynamics

The concept of zero dynamics plays a crucial role in the analysis of control systems, especially in the context of stability. Zero dynamics are defined by the behavior of the internal dynamics when the output of the system is forcibly held at zero. The stability of these zero-dynamics is a critical determinant of the overall system's behavior; if the zero-dynamics are stable, the system is considered minimum-phase, which is a desirable property for control purposes. Conversely, if the zero-dynamics are unstable, the system is non-minimum-phase, posing additional challenges for control design.

Implications for Control System Design

Understanding and applying the concept of **vector relative degree** and **input-output linearization** provides a powerful toolset for designing controllers, especially for complex nonlinear systems.

By transforming the system into a structure where the input directly influences the output linearly, we can more easily design control laws that achieve desired tracking performance while ensuring stability.

Moreover, the analysis of **zero dynamics** offers insight into the inherent limitations and challenges posed by the system's internal dynamics, guiding us in the design of robust controllers that can ensure stability and performance even in the presence of non-minimum-phase behaviors.

Fig. 1 represents the external and internal dynamics of the system.

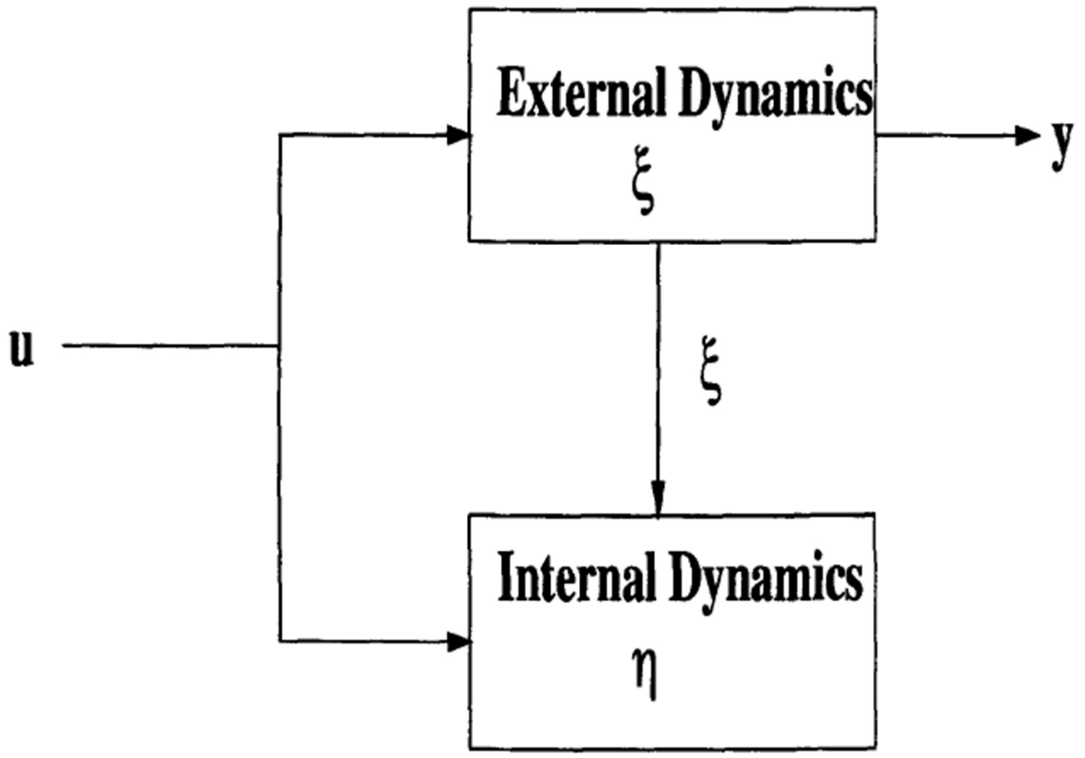


Figure 1: Representation of the internal and external dynamics of the system.

In the context of input-output linearized systems, the notion of internal dynamics (η) emerges as a critical aspect. These dynamics are described by functions $\phi i(x)$ for $i=1,\dots,n-p$, which are smooth functions of x ensuring that the state transformation $(\xi,\eta)=\psi(x)$ forms a local diffeomorphism at the origin.

This transformation isolates the dynamics that are not directly influenced by the control input in the linearized input-output representation.

The Zero dynamics, a concept borrowed from linear system theory, pertain to the behavior of these internal dynamics when the system's output—and consequently the transformed output ξ—is maintained at zero. Mathematically, this condition is expressed as $\eta'=\alpha(0,\eta)$.

The stability of these zero dynamics is pivotal; if they are asymptotically stable, the system is deemed minimum-phase, indicating a favorable condition for control. Conversely, instability in zero-dynamics categorizes the system as non-minimum-phase, presenting significant challenges for controller design.

Minimum-Phase and Non-Minimum-Phase Systems

The delineation between minimum-phase and non-minimum-phase systems is a fundamental concept in control theory. For linear systems, this characteristic is often analyzed through the system's transfer function H(s), particularly examining the placement of zeros in the complex plane.

Non-minimum phase systems, characterized by zeros in the right half of the complex plane, exhibit a phase behavior that complicates control design, as these systems can undergo undesirable phase shifts that are not present in their minimum-phase counterparts.

Transmission Zeros and System Stability

Transmission zeros play a crucial role in understanding the behavior of linear systems and, by extension, offer insights into the control of nonlinear systems. For an nth-order, m input, m output linear system represented by

$$\dot{x} = Ax + Bu$$
$$y = Cx$$

The condition for transmission zeros is given by the rank deficiency of the matrix

$$\begin{bmatrix} A - \lambda I_n & B \\ C & 0 \end{bmatrix}$$

being less than n+m. Furthermore, if the product CB is invertible, transmission zeros are identified as the finite eigenvalues of A+gBC as $g \to \infty$. This interpretation reveals an intriguing dynamic: employing static output feedback tends to drive the closed-loop system's poles towards these transmission zeros as feedback gain g is increased.

Hence, a system with right-half plane transmission zeros (indicative of a non-minimum phase behavior) may become unstable under certain conditions of static output feedback.

Understanding the distinction between minimum-phase and non-minimum-phase systems, and the role of transmission zeros, is paramount in control system design. For non-minimum-phase systems, care must be taken in designing control laws to ensure stability, as traditional approaches that work for minimum-phase systems may lead to instability. This necessitates a more nuanced approach, often involving advanced control strategies that can accommodate the peculiarities of non-minimum phase dynamics.

Examples (Non-minimum phase systems)

We delve into the non-minimum phase behavior in underactuated mechanical systems, drawing parallels to the underactuated flexible-link system.

The Acrobot System

A prime example of such a system is the Acrobot, a simplified representation of a human gymnast performing on a parallel bar.

This system, consisting of a two-link manipulator operating in a vertical plane, serves as a foundational model for understanding the dynamics of underactuated systems like unicycles and walking machines.

The Acrobot, an abbreviation for acrobatic robot, mirrors the actions of a gymnast utilizing hip rotation to achieve an inverted position.

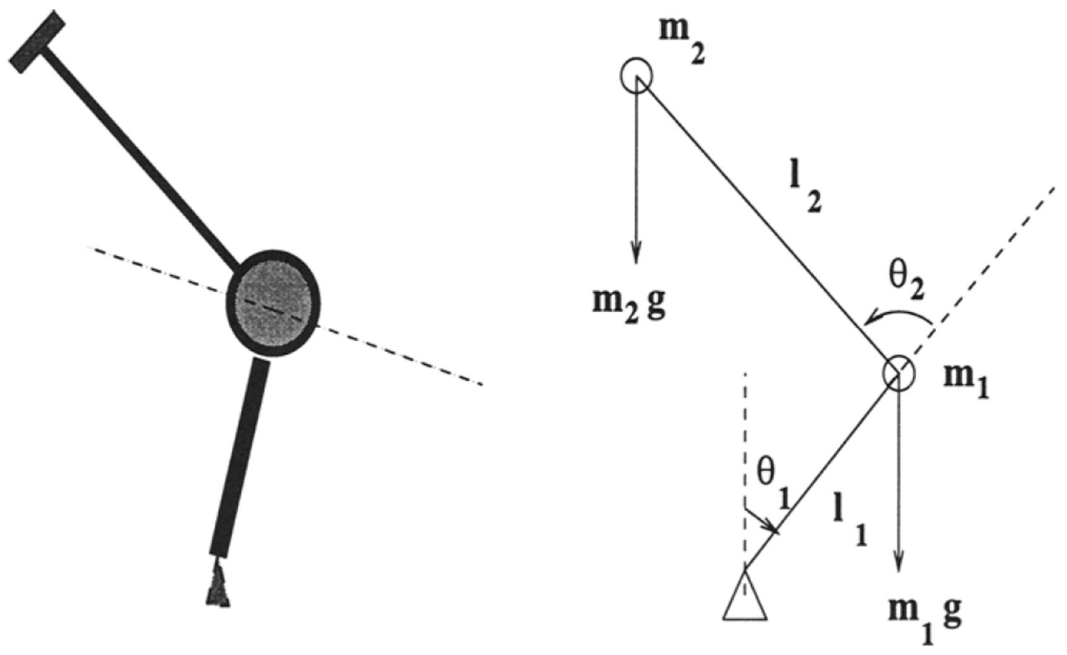


Figure 2: Acrobot: an acrobatic robot.

The system comprises two joints: the first joint is analogous to a gymnast's hand moving freely on the bar, while the second joint, equipped with a motor, simulates the gymnast's hip torque generation. The dynamics of this system are elegantly captured using the Lagrangian formulation, leading to the equation:

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta}) + G(\theta) = \begin{pmatrix} 0 \\ \tau \end{pmatrix}$$

where:

- $\theta = (\theta_1, \theta_2)$ represents the joint angles,
- $M(\theta)$ is the positive-definite inertia matrix,
- $C(\theta, \dot{\theta})$ encompasses the Coriolis and centrifugal forces,
- $G(\theta)$ denotes the gravitational effects,
- τ is the applied torque between the links.

The Lagrangian ($\mathcal{L}$) of a system is defined as the difference between the kinetic energy (T) and potential energy (V) of the system:

$$\mathcal{L} = T - V$$

For the Acrobot, the kinetic and potential energies are functions of the joint angles θ and their time derivatives. The kinetic energy T generally involves terms for both linear and angular motion, while the potential energy V is influenced by the gravitational potential at the heights of the masses involved.

Kinetic Energy T

The kinetic energy of the Acrobot can be calculated considering the rotational kinetic energy of both links. Given the masses m_1, m_2 and lengths l_1, l_2, and assuming point masses at the distal end of each link for simplicity, the kinetic energy is:

$$T = \frac{1}{2}m_1 l_1^2\dot{\theta}_1^2 + \frac{1}{2}m_2\left[l_1^2\dot{\theta}_1^2 + l_2^2\dot{\theta}_2^2 + 2l_1l_2\dot{\theta}_1\dot{\theta}_2\cos(\theta_2)\right]$$

Potential Energy V

The potential energy, focusing on gravitational potential, is influenced by the height of each mass:

$$V = -m_1 g l_1 \cos(\theta_1) - m_2 g[l_1 \cos(\theta_1) + \quad l_2 \cos(\theta_1 + \theta_2)]$$

Lagrangian Formulation

By substituting T and V into the Lagrangian $\mathcal{L} = T - V$, and applying the Euler-Lagrange equation for each coordinate θ_i:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_i}\right) - \frac{\partial \mathcal{L}}{\partial \theta_i} = \tau_i$$

where τ_i is the external torque applied to the i-th joint, we derive the equations of motion for the system.

The inertia matrix $M(\theta)$, Coriolis and centrifugal matrix $C(\theta, \dot{\theta})$, and gravitational vector $G(\theta)$ can be directly extracted from the Lagrangian's second-order differential equations. This extraction involves computing partial derivatives of $\mathcal{L}$ with respect to $\theta, \dot{\theta}$, and differentiating these with respect to time.

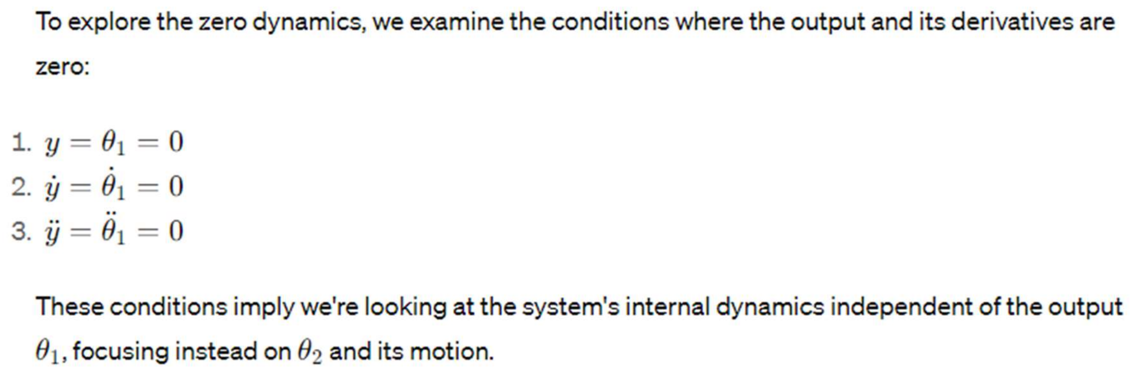
For instance, the components of the inertia matrix $M(\theta)$ are derived from the coefficients of $\ddot{\theta}$ in the kinetic energy expression. Similarly, the Coriolis and centrifugal terms arise from products of velocities ($\dot{\theta}$) and their derivatives, and the gravitational vector is derived from the gradient of the potential energy V with respect to θ.

Given the system dynamics:

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta}) + G(\theta) = \begin{pmatrix} 0 \\ \tau \end{pmatrix}$$

where $\theta = (\theta_1, \theta_2)$ are the joint angles, and the output y is defined as $y = \theta_1$, our goal is to analyze the behavior when $y = 0$, which implies that θ_1 and its derivatives $\dot{\theta}_1, \ddot{\theta}_1$ are zero.

Setting Up for Zero-Dynamics

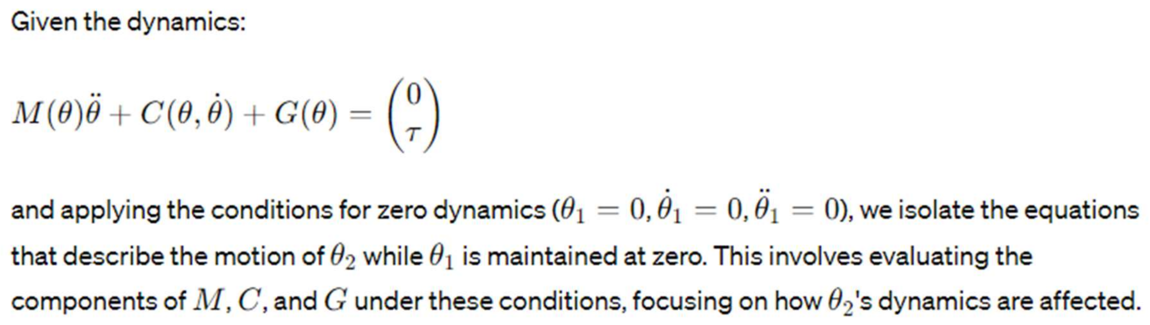


To explore the zero dynamics, we examine the conditions where the output and its derivatives are zero:

1. $y = \theta_1 = 0$
2. $\dot{y} = \dot{\theta}_1 = 0$
3. $\ddot{y} = \ddot{\theta}_1 = 0$

These conditions imply we're looking at the system's internal dynamics independent of the output θ_1, focusing instead on θ_2 and its motion.

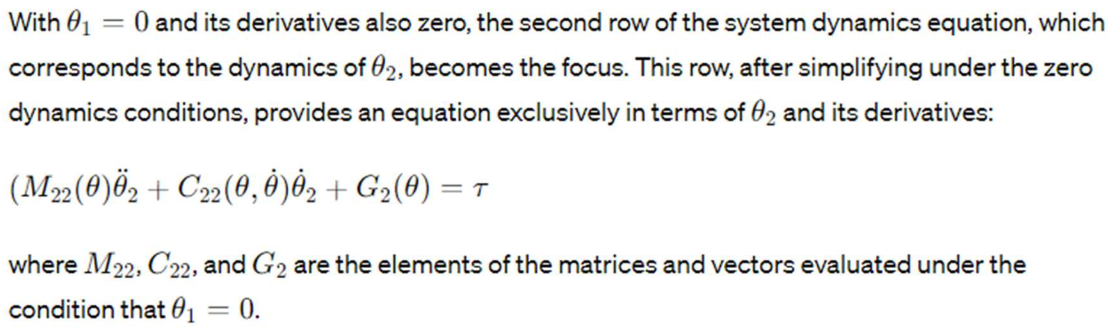
Isolating Internal Dynamics



Given the dynamics:

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta}) + G(\theta) = \begin{pmatrix} 0 \\ \tau \end{pmatrix}$$

and applying the conditions for zero dynamics ($\theta_1 = 0, \dot{\theta}_1 = 0, \ddot{\theta}_1 = 0$), we isolate the equations that describe the motion of θ_2 while θ_1 is maintained at zero. This involves evaluating the components of M, C, and G under these conditions, focusing on how θ_2's dynamics are affected.



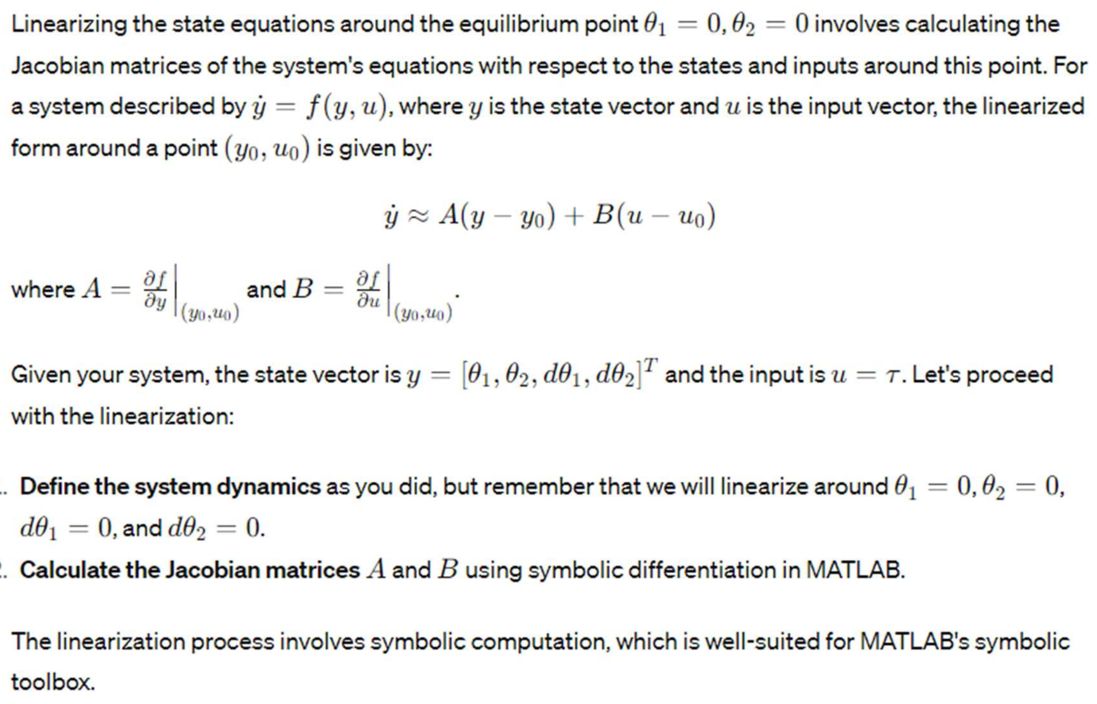
With $\theta_1 = 0$ and its derivatives also zero, the second row of the system dynamics equation, which corresponds to the dynamics of θ_2, becomes the focus. This row, after simplifying under the zero dynamics conditions, provides an equation exclusively in terms of θ_2 and its derivatives:

$$(M_{22}(\theta)\ddot{\theta}_2 + C_{22}(\theta, \dot{\theta})\dot{\theta}_2 + G_2(\theta) = \tau$$

where M_{22}, C_{22}, and G_2 are the elements of the matrices and vectors evaluated under the condition that $\theta_1 = 0$.

Considering the equilibrium point $\theta_2 = 0$ and linearizing around this point simplifies the dynamics to a form where the nonlinear terms are approximated based on their behavior close to the equilibrium. This often results in a linear differential equation representing the zero dynamics around this point.

The zero dynamics, thus derived, represent the internal behavior of the system when the output is forcibly kept at zero. This analysis is crucial for understanding how the system behaves independently of the control input designed to track or stabilize the output. For systems like the Acrobot, it highlights the inherent challenges in control design due to the presence of unstable internal dynamics, as revealed through the analysis of zero dynamics.


Linearizing the state equations around the equilibrium point $\theta_1 = 0, \theta_2 = 0$ involves calculating the Jacobian matrices of the system's equations with respect to the states and inputs around this point. For a system described by $\dot{y} = f(y, u)$, where y is the state vector and u is the input vector, the linearized form around a point (y_0, u_0) is given by:

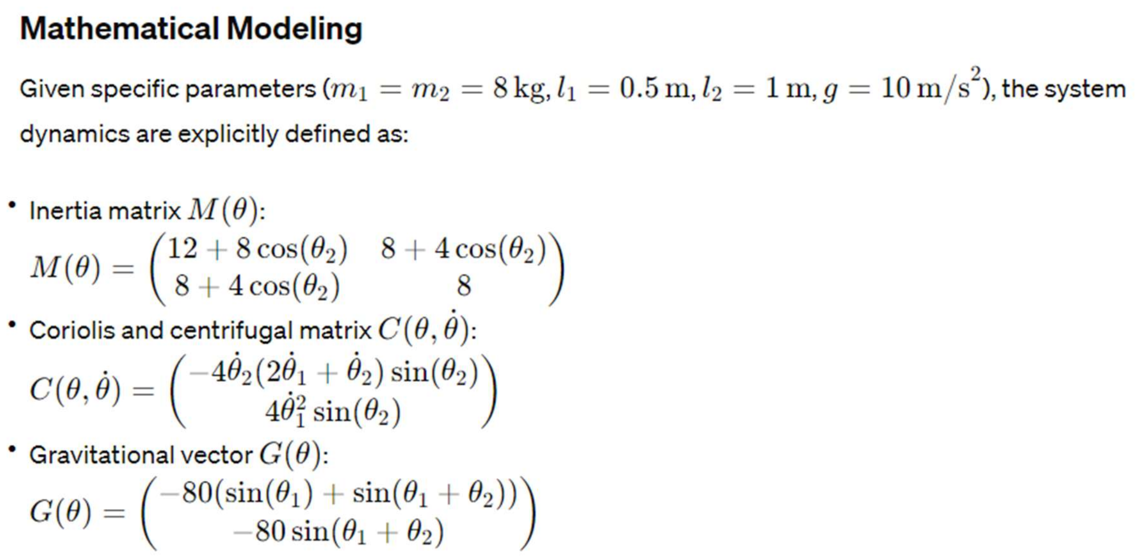
$$\dot{y} \approx A(y - y_0) + B(u - u_0)$$

where $A = \frac{\partial f}{\partial y}\Big|_{(y_0, u_0)}$ and $B = \frac{\partial f}{\partial u}\Big|_{(y_0, u_0)}$.

Given your system, the state vector is $y = [\theta_1, \theta_2, d\theta_1, d\theta_2]^T$ and the input is $u = \tau$. Let's proceed with the linearization:

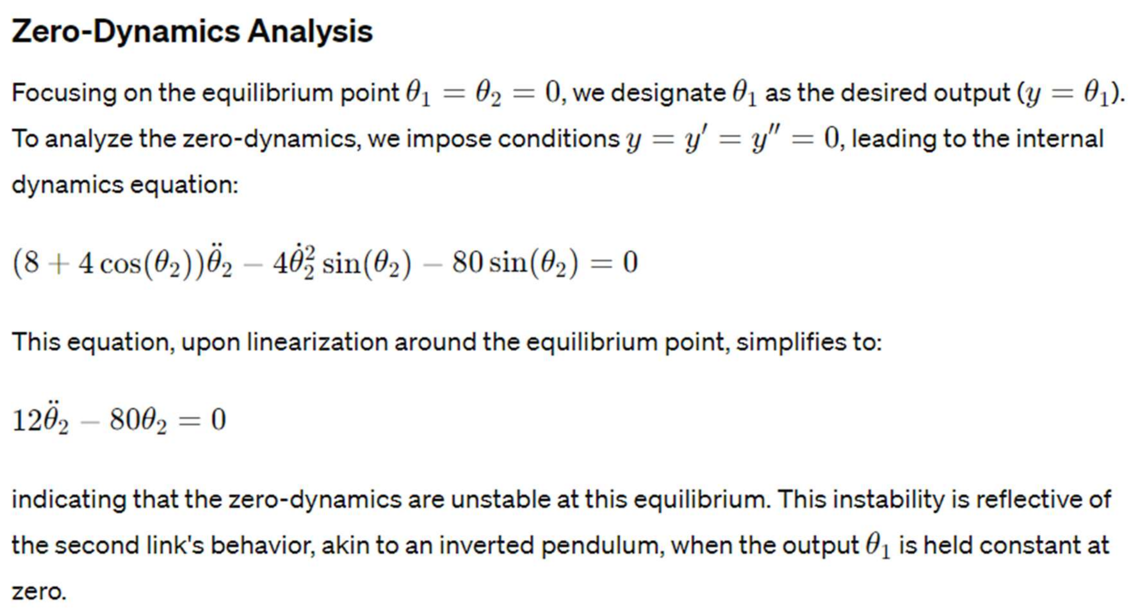
1. **Define the system dynamics** as you did, but remember that we will linearize around $\theta_1 = 0, \theta_2 = 0$, $d\theta_1 = 0$, and $d\theta_2 = 0$.
2. **Calculate the Jacobian matrices** A and B using symbolic differentiation in MATLAB.

The linearization process involves symbolic computation, which is well-suited for MATLAB's symbolic toolbox.


Mathematical Modeling

Given specific parameters ($m_1 = m_2 = 8\,\mathrm{kg}, l_1 = 0.5\,\mathrm{m}, l_2 = 1\,\mathrm{m}, g = 10\,\mathrm{m/s}^2$), the system dynamics are explicitly defined as:

- Inertia matrix $M(\theta)$:
$$M(\theta) = \begin{pmatrix} 12 + 8\cos(\theta_2) & 8 + 4\cos(\theta_2) \\ 8 + 4\cos(\theta_2) & 8 \end{pmatrix}$$
- Coriolis and centrifugal matrix $C(\theta, \dot{\theta})$:
$$C(\theta, \dot{\theta}) = \begin{pmatrix} -4\dot{\theta}_2(2\dot{\theta}_1 + \dot{\theta}_2)\sin(\theta_2) \\ 4\dot{\theta}_1^2 \sin(\theta_2) \end{pmatrix}$$
- Gravitational vector $G(\theta)$:
$$G(\theta) = \begin{pmatrix} -80(\sin(\theta_1) + \sin(\theta_1 + \theta_2)) \\ -80\sin(\theta_1 + \theta_2) \end{pmatrix}$$



Zero-Dynamics Analysis

Focusing on the equilibrium point $\theta_1 = \theta_2 = 0$, we designate θ_1 as the desired output ($y = \theta_1$). To analyze the zero-dynamics, we impose conditions $y = y' = y'' = 0$, leading to the internal dynamics equation:

$$(8 + 4\cos(\theta_2))\ddot{\theta}_2 - 4\dot{\theta}_2^2 \sin(\theta_2) - 80\sin(\theta_2) = 0$$

This equation, upon linearization around the equilibrium point, simplifies to:

$$12\ddot{\theta}_2 - 80\theta_2 = 0$$

indicating that the zero-dynamics are unstable at this equilibrium. This instability is reflective of the second link's behavior, akin to an inverted pendulum, when the output θ_1 is held constant at zero.

Backstepping Controller Design

Designing a backstepping controller for a system like the Acrobot, where the objective is to make θ1 and θ2 converge to zero in the presence of disturbances, involves several steps. Backstepping is a recursive design technique used for nonlinear systems, allowing for the construction of a stabilizing controller by progressively designing feedback laws for a sequence of expanding subsystems.

Given the complexity of the Acrobot's dynamics and the requirement for a step-by-step derivation, we'll outline a simplified approach to design a backstepping controller for a generic two-link system like the Acrobot.

Note that a detailed design would require a comprehensive analysis involving the system's full dynamics, which are significantly complex. The explanation here is conceptual and meant to guide the development of a more specific solution.

Step 1: Define the Control Objectives and Errors

Let's define θ_1 and θ_2 as the positions of the two links, with the control objective to make $\theta_1 \to 0$ and $\theta_2 \to 0$.

Define the errors as:

- $e_1 = \theta_1$ and $e_2 = \dot{\theta}_1$,
- $e_3 = \theta_2$ and $e_4 = \dot{\theta}_2$.

Step 2: Backstepping Design for Subsystem 1 (Link 1)

2.1 Virtual Control Law for e_1:

Start with the first subsystem (link 1) and design a virtual control law for e_1:

$\alpha_1 = \quad k_1 e_1$

where $k_1 > 0$ is a design parameter.

2.2 Define Error for e_2:

$e_2 = \dot{e}_1 + \alpha_1$

2.3 Virtual Control for e_2:

$\alpha_2 = \quad k_2 e_2 + \frac{d\alpha_1}{dt}$

where $k_2 > 0$.

Step 3: Backstepping Design for Subsystem 2 (Link 2)

3.1 Define Errors for Second Link:

Given the definitions of e_1 and e_2 for the first link, we proceed to define the errors for the second link, θ_2. The objective is to stabilize θ_2 to a desired value, which we'll assume to be zero for simplicity. The errors for the second link are defined as follows:

- $e_3 = \theta_2$ represents the position error of the second link.
- $e_4 = \dot{\theta}_2 - \alpha_3$ represents the velocity error of the second link, where α_3 is a virtual control to be designed.

The design of α_3 aims to make $e_3 \to 0$, and subsequently, α_3 will help in defining the control law that makes $e_4 \to 0$.

3.2 Virtual Control Law for e_3:

To design α_3, we first need a control objective for e_3. The virtual control law for e_3 is designed to stabilize e_3 by counteracting the dynamics of θ_2. A simple choice for α_3 could be:

$$\alpha_3 = -k_3 e_3$$

where $k_3 > 0$ is a constant gain. This choice is analogous to a proportional controller aiming to reduce the error e_3 by applying a control action proportional to the error itself.

3.3 Define Error for e_4:

With α_3 designed, we now define e_4 as:

$$e_4 = \dot{e}_3 + \alpha_3 = \dot{\theta}_2 + k_3 e_3$$

This equation represents the deviation of the actual velocity of the second link from the desired velocity trajectory defined by α_3.

3.4 Actual Control Law for e_4:

The final step in the backstepping design for the second link involves designing an actual control law that directly uses the system's input, τ, to stabilize e_4. The control law must consider the dynamics of the entire system, including the influence of the first link's motion on the second link. A possible form of the control law to achieve this could be:

$\tau = \quad k_4 e_4 + \text{dynamics compensation terms}$

where $k_4 > 0$ is a constant gain, and the "dynamics compensation terms" are terms derived from the system's dynamics equations that account for the interconnectedness of the two links, including Coriolis, centrifugal, and gravitational effects. These terms are crucial for ensuring that the control law accurately counteracts the system's natural dynamics to achieve stabilization.

Implementation Considerations:

- The choice of gains k_3 and k_4 and the exact form of the dynamics compensation terms are critical for the success of the controller. These should be derived based on a detailed analysis of the system's dynamics.
- The dynamics compensation terms would typically involve nonlinear functions of $\theta_1, \theta_2, \dot{\theta}_1$, and $\dot{\theta}_2$, and possibly estimates or measurements of external disturbances if they are significant.
- This simplified outline omits the intricate details of deriving exact expressions for the dynamics compensation terms, which require a thorough nonlinear analysis of the Acrobot's equations of motion.

Step 4: Control Law Synthesis

4.1 Final Control Input τ:

The final step involves synthesizing the control input τ that considers the dynamics of both links, the virtual controls, and the designed errors. The actual control law might look something like:
$\tau = f(e_1, e_2, e_3, e_4, \dot{e}_3, \dot{e}_4, parameters)$

This control law aims to stabilize both links by considering the interconnected dynamics and the desired error dynamics.

Step 5: Stability and Lyapunov Analysis

Conduct a Lyapunov analysis to ensure the stability of the closed-loop system. Choose a Lyapunov function candidate that reflects the total energy of the system and show that its derivative along the trajectories of the closed-loop system is negative semidefinite.

Implementation and Tuning

Implement the derived control law in a simulation environment like MATLAB/Simulink. Adjust the design parameters $(k_1, k_2, ...)$ through simulation and experimentation to achieve the desired control objectives.

Full Control Action Formulation

Combining these elements, the symbolic form of the control action τ, which is a simplification and not directly applicable without further detailed derivation, could be expressed as:

$$\tau = \quad k_4(\dot{\theta}_2 + k_3\theta_2) + f(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2, \ddot{\theta}_1, \ddot{\theta}_2)$$

Note on Practical Implementation

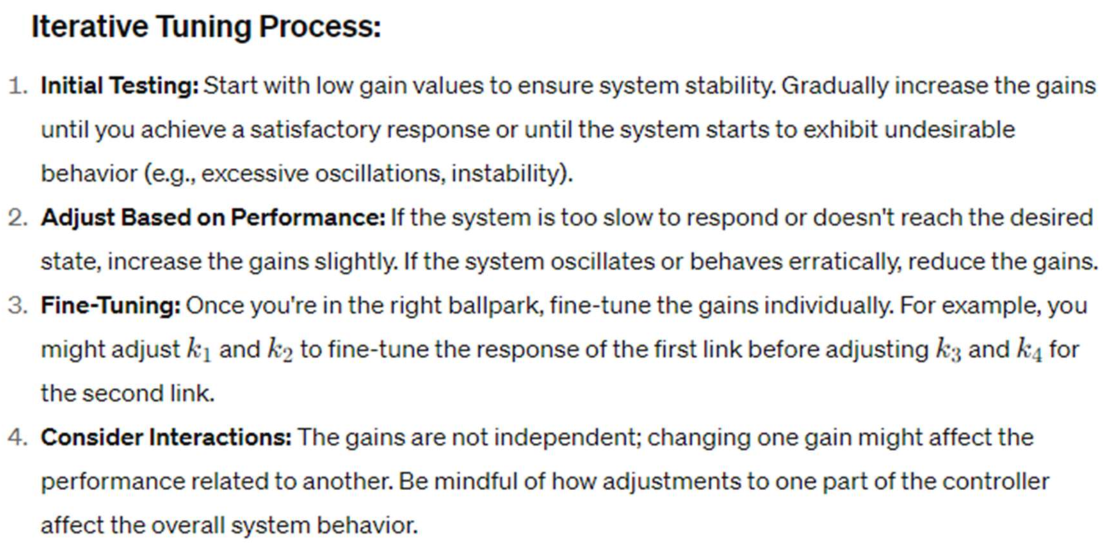
- The symbolic form here is conceptual and serves to illustrate the methodology of backstepping control design. The actual implementation requires deriving f explicitly based on the Acrobot's dynamics, which involves solving the system's nonlinear differential equations.
- Designing f involves calculating the partial derivatives of the system's Lagrangian with respect to the system's states and then solving for $\ddot{\theta}_1$ and $\ddot{\theta}_2$, which are not directly controlled due to the underactuated nature of the system.
- The gains k_3 and k_4, and any additional parameters in f, must be carefully tuned based on the system's physical parameters and desired performance metrics, often involving simulation-based optimization or experimental tuning.

In summary, while the symbolic representation provides a framework, actual control design for nonlinear, underactuated systems like the Acrobot requires in-depth analysis, detailed modeling, and careful consideration of the system's dynamics and interactions.

Selecting gain values ($k_1, k_2, k_3, k_4, \ldots$) for a backstepping controller, especially for complex and highly nonlinear systems like the Acrobot, requires a careful balance between system responsiveness, stability, and robustness to disturbances. Without specific details about the system parameters (masses, lengths, moments of inertia, etc.) and desired performance criteria (settling time, overshoot, stability margins), providing exact gain values can be challenging. However, I can offer some general guidelines and a starting point for tuning.

Guidelines for Selecting Gain Values:

1. **Stability and Responsiveness:** Higher gain values can improve system responsiveness and reduce settling time but might lead to increased oscillations or even instability if too high. Conversely, too low gains might make the system sluggish or unable to counteract disturbances effectively.
2. **Robustness:** Consider the system's robustness to model uncertainties and external disturbances. Robust control design often involves selecting gains that provide a good trade-off between performance and sensitivity to parameter variations.
3. **Simulation and Experimentation:** Use simulation tools (e.g., MATLAB/Simulink) to model your system and test different gain values. Experimentation with a physical system, if available, can also provide valuable insights.


Iterative Tuning Process:

1. **Initial Testing:** Start with low gain values to ensure system stability. Gradually increase the gains until you achieve a satisfactory response or until the system starts to exhibit undesirable behavior (e.g., excessive oscillations, instability).
2. **Adjust Based on Performance:** If the system is too slow to respond or doesn't reach the desired state, increase the gains slightly. If the system oscillates or behaves erratically, reduce the gains.
3. **Fine-Tuning:** Once you're in the right ballpark, fine-tune the gains individually. For example, you might adjust k_1 and k_2 to fine-tune the response of the first link before adjusting k_3 and k_4 for the second link.
4. **Consider Interactions:** The gains are not independent; changing one gain might affect the performance related to another. Be mindful of how adjustments to one part of the controller affect the overall system behavior.

The dynamic compensation term in a control law, particularly for complex systems like the Acrobot, is crafted to counteract the system's intrinsic dynamics that are not directly influenced by the control input—often referred to as the zero dynamics. These dynamics represent the internal behavior of the system when the output is constrained to be zero.

For the Acrobot, a two-link underactuated robotic system, the dynamic compensation term must address the interactions between the links, including gravitational, Coriolis, and centrifugal effects, as well as the influence of the un-actuated dynamics. Since the Acrobot is underactuated (only one control input for two degrees of freedom), direct control over both links is not possible. Instead, the control strategy must cleverly exploit the system's dynamics to achieve the desired motion.

Formulating the Dynamic Compensation Term

The dynamic compensation term typically includes elements to counteract the gravitational forces, Coriolis and centrifugal forces, and possibly other nonlinear dynamics. For the Acrobot, let's denote:

- $M(\theta)$ as the mass matrix,
- $C(\theta, \dot{\theta})$ as the Coriolis and centrifugal matrix,
- $G(\theta)$ as the gravity vector,
- B as the input matrix,
- τ as the control input (torque),
- $\theta = [\theta_1, \theta_2]^T$ as the vector of joint angles,
- $\dot{\theta} = [\dot{\theta}_1, \dot{\theta}_2]^T$ as the vector of joint angular velocities.

The equations of motion for the Acrobot can be written as:
$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + G(\theta) = B\tau$

The dynamic compensation term, let's call it $\mathrm{Comp}(\theta, \dot{\theta})$, would ideally cancel out the terms that are not directly influenced by the control input. This term could be structured as follows, assuming $B = [0, 1]^T$ indicating control applied to the second joint:

$$\mathrm{Comp}(\theta, \dot{\theta}) = -M(\theta)^{-1}(C(\theta, \dot{\theta})\dot{\theta} + G(\theta))$$

However, this direct inversion of the mass matrix $M(\theta)$ is not straightforward in practice due to its dependence on the system states and potential for singularity. Instead, the control law, including the dynamic compensation, often requires a more nuanced approach, leveraging Lyapunov-based control design or other techniques to ensure stability and robustness.

Implementing Zero Dynamics Control

To effectively implement a control that leverages the zero dynamics, one might design a feedback linearization control where the dynamic compensation term is part of the control input τ, such as:

$$\tau = \mathrm{Comp}(\theta, \dot{\theta}) + \mathrm{ControlLaw}(\theta, \dot{\theta}, \text{desired states})$$

The $\mathrm{ControlLaw}$ term here represents the additional control actions needed to drive the system towards the desired states or trajectories, possibly designed through backstepping or another nonlinear control technique.

Caveats

Model Precision: The effectiveness of the dynamic compensation relies heavily on the precision of the system model. Any discrepancies between the model and the real system can degrade control performance.

Robustness: It's crucial to consider robustness to model uncertainties and external disturbances. Adaptive control strategies or robust control techniques may be necessary for practical applications.

Computational Complexity: Real-time computation of the dynamic compensation term requires efficient numerical methods, especially for systems with fast dynamics like the Acrobot.

Reference Book:

Moallem, M., Patel, R. V., & Khorasani, K. (2000). Flexible link robot manipulators: Control techniques and Structural Design. Springer.

Backstepping control of Nonlinear Dynamical Systems. (2020). . ELSEVIER ACADEMIC Press. https://www.amazon.com/Backstepping-Nonlinear-Dynamical-Advances-Dynamics/dp/0128175826

Slotine, J.-J. E., & Li, W. (2005). Applied Nonlinear Control. Pearson Education Taiwan.