

Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar
Nevét Ide Tanszék



TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

CSILLAPÍTOTT NEWTON-MÓDSZER, MINT DEADBEAT SZABÁLYOZÓ: MEGVALÓSÍTHATÓSÁGI TANULMÁNY

Szerző(k): **Gadelmoly Mostafa Gamal Barkat**
KANDÓ KÁLMÁN
VILLAMOSMÉRNÖKI KAR, MSc II.
évfolyam

Konzulens(ek): **Borsos Döníz**
tanársegéd

Budapest, 2024.

CSILLAPÍTOTT NEWTON-MÓDSZER, MINT DEADBEAT SZABÁLYOZÓ: MEGVALÓSÍTHATÓSÁGI TANULMÁNY

Gadelmoly Mostafa Gamal Barkat

Óbudai Egyetem

KANDÓ KÁLMÁN VILLAMOSMÉRNÖKI KAR, MSc II. évfolyam

Konzulens: Borsos Döníz, tanársegéd

This work proposes a new method of designing a deadbeat controller with the help of damped Newton's method. This method can be applied to the linear approximation of discrete-time systems. Traditional deadbeat controllers, while effective in many cases, suffer from issues such as overshooting and aggressiveness. Using Newton's method as a deadbeat controller enhances robustness while maintaining the core performance attributes and fundamental characteristics of traditional deadbeat control. This work analyzes and compares the suggested method to the commonly used pole placement in terms of convergence speed, overshooting, and overall stability. The goal of this work is not to prove that this is the best method but to showcase the possibilities that come from using it.

The main hypothesis is that utilizing Newton's method as a deadbeat controller will result in the system reaching the steady state (zero) in finite time while requiring less aggressive control effort and providing improved stability.

Contents

1	Introduction	3
1.1	Background	3
1.2	Thesis Structure	4
2	Literature Review	5
2.1	System formulation	5
2.2	Deadbeat Control and Pole Placement	5
2.3	Newton's Method	6
2.3.1	State of Art and Research Gap	8
3	Methodology	9
3.1	System Design	9
3.2	Control Design	11
3.2.1	Deadbeat Approach	11
3.2.2	Newton's Method Approach	11
3.2.3	Optimization Problem	12
3.2.4	Backtracking Line Search	13
3.2.5	Iteration Process (summarized)	14
4	Execution and Results	16
4.1	Simulation Parameters	16
4.1.1	Newton's Method Simulation Parameters	16
4.2	Pole Placement Results	17
4.3	Newton's Method Results	17
4.3.1	Effect of Different Parameters	21
5	Conclusion	27

Chapter 1

Introduction

1.1 Background

The primary aim of a deadbeat controller is to drive the state of a linear discrete-time system to zero within a finite number of steps, typically the minimal possible. This is commonly achieved by positioning the poles of the system at or near zero [1]. With that being said, most of the dynamical systems in the real world are nonlinear systems. In most cases dealing with nonlinear requires linearization of the system as working with the linear approximation of the system allow us to utilize the benefits and simplicity of linear systems. The concept of deadbeat controller first introduced in the 1980s by J. Ackermann, whose name is associated with one of the principal techniques, Ackermann's method [2]. Over time, various approaches have been proposed to enhance the benefits of this method while minimizing associated errors, the gain matrix and its computation often being the central focus [3].

Challenges with the original methods can be assessed from two perspectives. Practically, they demand significant control efforts and may result in high overshooting. To address these issues, some studies have proposed the use of deadbeat-like PID controllers [4], while others recommend larger time steps to reduce overshooting [5]. Additionally, Phillips and Nagle (1995) introduced a weighted pole placement method as an alternative solution.

This work adopts a different approach, building on the foundational concept of deadbeat control while dynamically adjusting the control gain in response to the state and errors of the system. To ensure system convergence, a damping factor and gradient descent are incorporated, provided an appropriate initial guess is available.

1.2 Thesis Structure

The thesis is organized as follows. Chapter 2 provides a review of the relevant literature. Chapter 3 presents the mathematical model and the description of the system, along with an outline of the proposed technique. Chapter 4 discusses the results and analysis, while Chapter 5 offers the conclusions of the study. The thesis concludes with the Appendix and References.

Chapter 2

Literature Review

2.1 System formulation

Understanding the formulation of a dynamical system is a crucial step in designing and implementing a controller, as controllers inherently function as dynamical systems with an input and at least one output MISO.

2.2 Deadbeat Control and Pole Placement

In digital control, a deadbeat controller is a discrete-time control system designed to drive the system to a desired state, typically the zero state, in the minimal number of sampling steps. This method is particularly useful in scenarios requiring rapid stabilization [6].

Consider the following state-space equations representing a discrete-time system:

$$y(k+1) = Hy(k) + Gu(k) \quad (2.1)$$

Here, $y(k)$, $y \in R^n$, represents the state vector, H is the state matrix, while the input matrix is denoted as G , and $u(k)$, $u \in R^m$ denotes the control input. The aim of the deadbeat controller is to design the system such that $y(k)$ reaches the target value, often zero, within a finite number of time steps. Ideally, this is accomplished in n steps, where n is the dimension of the state vector. This rapid convergence is achieved by placing the poles of the closed-loop system at the origin, thereby ensuring the fastest possible response. Provided the system is controllable, the approach guarantees optimal performance [7].

The controllability subspaces of the system is defined as:

$$\mathcal{C} = [G, HG, H^2G, \dots, H^{n-1}G] \quad (2.2)$$

Thus $\mathcal{C}$ is the set of all of the system's states which can be driven to the origin in a defined number of steps using an appropriate sequence of control inputs $u_0, u_1, \dots, u_{n-1}$. Furthermore, if $\mathcal{C}$ is full rank and H is not a singular matrix, the system is so-called controllable and a gain matrix K with the following property can be determined.

$$(H - GK)C_n \subset C_{n-1} \quad (2.3)$$

The selection of K is guided by the desired placement of the poles.

To achieve this, a state-feedback method is employed, where the control input is defined as:

$$u(k) = -Ky(k) \quad (2.4)$$

The state feedback gain matrix K is chosen to ensure that all of the closed-loop system states

$$y_{k+1} = (H - GK)y_k \quad (2.5)$$

belonging to the control subspace are brought to the origin in n steps. [8].

Ackermann's formula provides a systematic solution for computing the state feedback gain matrix [9] [10]:

$$K = [0, \dots, 0, 1]\mathcal{C}^{-1}\Delta(H) \quad (2.6)$$

Here, $\Delta(H)$ represents the desired characteristic polynomial, applied to the state matrix H , while $\mathcal{C}^{-1}$ is the inverse of the controllability matrix. This formula simplifies the process of pole placement, enabling precise control design.

2.3 Newton's Method

Newton's method is a numerical methodology that uses iterative methods to identify a function's approximate roots [11]. It is used as one of the optimal control algorithms. [12] [13]

With a suitable initial guess for y_0 , the tangent line search method can be applied iteratively to find the best approximation of the roots.

$$f(y) = 0 \quad (2.7)$$

Where the tangent at the point n will be y_n , and it can be expressed as

$$y = f(y_n) + f'(y_n)(y - y_n) \quad (2.8)$$

Setting this equation equal to zero allows it to be expressed as

$$0 = f(y_n) + f'(y_n)(y_{n+1} - y_n) \quad (2.9)$$

Rearranging the equation and solving for the next approximation y_{n+1} yields the iterative approximation:

$$y_{n+1} = y_n - \frac{f(y_n)}{f'(y_n)} \quad (2.10)$$

One reason Newton's method is efficient, especially when close to the solution, is the fact that it has quadratic convergence. This means that if the current y_n is close enough to the desired root y^* , the error of each step will be squared:

$$|y_{n+1} - y^*| \approx C|y_n - y^*|^2 \quad (2.11)$$

Challenges of Newton's Method in Control Applications

Cost: Despite the positives of using Newton's method, the main challenge is its computational complexity, especially for high-dimensional systems. However, for a relatively simple system, this cost can be negligible compared to the benefits.

Sensitivity to Initial Guess: A reasonably accurate first guess y_0 is required for convergence to occur. If the initial guess is too far from the actual roots, the iteration may diverge or converge to a local minimum. This issue can be avoided by using a damped variation of Newton's method. A damping parameter $t \in (0, 1]$ can be introduced:

$$y_{n+1} = y_n - t \frac{f(y_n)}{f'(y_n)} \quad (2.12)$$

More controlled convergence is ensured by this setting. In this paper, this approach was taken.

2.3.1 State of Art and Research Gap

Despite many studies on the use of Newton's method as a tool for various control and optimization techniques, no previous work has suggested using it as part of direct deadbeat control. This part of the work explored the state-of-the-art of using this method in control engineering.

- Newton's method for continuous heterogeneous systems: Starting with (Veliov and Vladimir 2003) [14] the work addresses the use of Newton's method for optimizing the control problems with a focus on continuous system heterogeneous systems, where a set-valued version of the method was used to solve variation inequalities. This was done using a second-order Runge-Kutta scheme to allow accurate discretization of the system.
- Model Predictive Control (MPC): A system of parallel Newton methods was proposed for the non-linear model predictive control algorithm. The work of (Haoyang and Toshiyuki 2018) resulted in the ability to effectively parallelize the NMPC problem by using an approximately decoupled single-step subproblem. Using a quad-rotor control in a numerical simulation, the result of this method proved to be faster when it comes to convergence than the sequential quadratic programming (SQP) and interior-point (IP) methods [15]. The method was further developed to ensure even higher accuracy with less iteration [16]
- Hierarchical Newton's Method for Numerically Stable Prioritized Dynamic Control: The latest work using the Newton method was applied in a hierarchical setting of the method. The prioritization of tasks The new method maintains using a specialized formulation of the Hessian matrix. This method was used in velocity-based control; the method allows for fast response, with the ability to balance high ad low priority control tasks [17]

Chapter 3

Methodology

3.1 System Design

For this work, a simple case of the nonlinear dynamics of a simple pendulum is used. The general dynamics can be described as the following second-order differential equation

$$L\ddot{y}(t) + g \sin(y(t)) = u(t) \quad (3.1)$$

where:

- L : Length of the pendulum [m].
- g : Acceleration (in m/s^2).
- $y(t)$: Angular displacement [radians].
- $u(t)$: External input torque applied to the pendulum [in Nm].

Starting from this second-order differential equation, the system dynamics can be expressed in state-space representation as follows. First, define the state variables:

$$y_1(t) = y(t) \quad (3.2)$$

$$y_2(t) = \dot{y}(t) \quad (3.3)$$

Next, the state-space representation of the nonlinear system:

$$\dot{y}_1(t) = y_2(t) \quad (3.4)$$

$$\dot{y}_2(t) = -\frac{g}{L} \sin(y_1(t)) + \frac{1}{L}u(t) \quad (3.5)$$

Implementing the controller requires a linear representation of the system dynamics, which is achieved by applying the small-angle condition ($y \approx 0$), where $\sin(y) \approx y$. The resulting system is represented as follows:

$$\dot{y}(t) = Hy(t) + Gu(t) \quad (3.6)$$

where:

$$H = \begin{bmatrix} 0 & 1 \\ -\frac{g}{L} & 0 \end{bmatrix}, \quad G = \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} \quad (3.7)$$

Discretization As a digital controller is implemented, the continuous-time system will have to be discretized. This can be done using the sampling time T_s and converting from the continuous-time state-space model into a discrete-time model using the Zero-Order Hold (ZOH) method. The discrete state-space equations can be written as:

$$y[k+1] = H_d y[k] + G_d u[k] \quad (3.8)$$

where:

- H_d and G_d : Discrete-time system matrices.
- k : Discrete time step.

The discrete-time system matrices are computed from the original continuous-time H and G as follows:

$$H_d = e^{HT_s} \quad (3.9)$$

$$G_d = \int_0^{T_s} e^{H\tau} G d\tau \quad (3.10)$$

3.2 Control Design

3.2.1 Deadbeat Approach

In comparison to other optimal control approaches where Newton's method is mainly used for discretizing system dynamics. [18]. In other cases it also used as Sequential Quadratic Programming (SQP) [19] or as an interior-point method [20]. In this work, the aim is to drive the state vector $y(k)$ to zero, or close to zero, in a finite number of steps. This can be done by placing the closed-loop system poles at the origin of the z -plane. This guarantees the quickest possible convergence of the system. It is important to choose the state feedback matrix so that all of the eigenvalues of the closed-loop system are zero:

$$u(k) = -Ky(k) \quad (3.11)$$

3.2.2 Newton's Method Approach

The following is a representation of the closed-loop system with state feedback :[21]

$$y[k+1] = (H_d - G_d K)y[k] \quad (3.12)$$

An error function $\mathcal{E}(K)$ is defined to describe the difference between the current characteristic polynomial equation and the desired characteristic polynomial:

$$\mathcal{E}(K) = P_{\text{current}}(z) - P_{\text{desired}}(z) \quad (3.13)$$

$P_{\text{current}}(z)$: The closed-loop system matrix $H_d - G_d K$ characteristic polynomial. It is described as:

$$P_{\text{current}}(z) = \det(zI - (H_d - G_d K)) \quad (3.14)$$

Where the root of this polynomial represents the eigenvalues of the system and they determine the system response and stability. Similarly, the desired characteristic polynomial, which contains the designed eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$ needed to drive the system to the desired state, is:

$$P_{\text{desired}}(z) = (z - \lambda_1)(z - \lambda_2) \cdots (z - \lambda_n) \quad (3.15)$$

As this work is focusing on deadbeat beat control for the system

$$P_{\text{desired}}(z) = z^n \quad (3.16)$$

Now, the problem can be described as a minimization problem, where the objective is to adjust the K gain matrix such that

$$P_{\text{current}}(z) \rightarrow P_{\text{desired}}(z)$$

The error function will be written as

$$\mathcal{E}(K) = (\det(zI - (H_d - G_d K))) - z^n \quad (3.17)$$

3.2.3 Optimization Problem

As the goal is to minimize $\mathcal{E}(K)$. The optimization problem can be written as

$$\min_K \mathcal{E}(K)$$

The second-order Taylor expansion can then be derived as a function approximation.

$$\mathcal{E}(K) \approx \mathcal{E}(K_i) + \nabla \mathcal{E}(K_i)^T (K - K_i) + \frac{1}{2} (K - K_i)^T \nabla^2 \mathcal{E}(K_i) (K - K_i) \quad (3.18)$$

where:

- K_i : The gain matrix at the i -th iteration.
- $\nabla \mathcal{E}(K_i)$: The error function Gradient with respect to K .
- $\nabla^2 \mathcal{E}(K_i)$: Hessian matrix of the error function at iteration i .

By taking the derivative with respect to K and setting it equal to zero:

$$\nabla \mathcal{E}(K) \approx \nabla \mathcal{E}(K_i) + \nabla^2 \mathcal{E}(K_i) (K - K_i) = 0 \quad (3.19)$$

Rearranging the system for K :

$$K - K_i = - (\nabla^2 \mathcal{E}(K_i))^{-1} \nabla \mathcal{E}(K_i) \quad (3.20)$$

Thus, the Newton step update rule is

$$K_{i+1} = K_i - (\nabla^2 \mathcal{E}(K_i))^{-1} \nabla \mathcal{E}(K_i) \quad (3.21)$$

By adding a step size t_i to control the update and ensure convergence, the new step update equation is:

$$K_{i+1} = K_i - t_i (\nabla^2 \mathcal{E}(K_i))^{-1} \nabla \mathcal{E}(K_i) \quad (3.22)$$

K is updated with each iteration, and the gradient and Hessian are computed. The Hessian gives information about the curvature, while the gradient gives the error's sensitivity to variations in the gain matrix. [22]

3.2.4 Backtracking Line Search

Since the system is unconstrained, a backtracking line search method will be employed to ensure the convergence of the approach [11]. This technique is utilized to determine the appropriate step size along the search direction. The implementation of the backtracking line search for the error function is described as follows:

The error function at the updated point derived from the step update equation is:

$$\mathcal{E}(K_i - t \nabla^2 \mathcal{E}(K_i)^{-1} \nabla \mathcal{E}(K_i)) \quad (3.23)$$

Using the second-order Taylor expansion around K_i :

$$\mathcal{E}(K_i - t \nabla^2 \mathcal{E}(K_i)^{-1} \nabla \mathcal{E}(K_i)) \approx \mathcal{E}(K_i) - t \nabla \mathcal{E}(K_i)^T \nabla^2 \mathcal{E}(K_i)^{-1} \nabla \mathcal{E}(K_i) \quad (3.24)$$

A sufficient decrease condition is imposed, which requires that the new value of $\mathcal{E}(K)$ be less than the current value $\mathcal{E}(K_i)$ by at least a fraction α of the anticipated decrease in $\mathcal{E}(K)$, which is proportional to $t \|\nabla \mathcal{E}(K_i)\|^2$.

$$\mathcal{E}(K_i - t \nabla^2 \mathcal{E}(K_i)^{-1} \nabla \mathcal{E}(K_i)) \leq \mathcal{E}(K_i) + \alpha t \nabla \mathcal{E}(K_i)^T \nabla \mathcal{E}(K_i) \quad (3.25)$$

The left-hand side, $\mathcal{E}(K_i - t \nabla^2 \mathcal{E}(K_i)^{-1} \nabla \mathcal{E}(K_i))$, represents the actual value of the error function after the step. The right-hand side, $\mathcal{E}(K_i) + \alpha t \nabla \mathcal{E}(K_i)^T \nabla \mathcal{E}(K_i)$, represents the required reduction in $\mathcal{E}(K)$, scaled by α and the step size t .

- $\mathcal{E}(K_i)$: The error function current value at iteration i .
- $\mathcal{E}(K_i - t \nabla^2 \mathcal{E}(K_i)^{-1} \nabla \mathcal{E}(K_i))$: Value of the error function after taking a Newton step with step size t .

- $\alpha \in (0, 1)$: A small constant (e.g., $\alpha = 0.1$ or $\alpha = 0.01$) that controls the required decrease in $\mathcal{E}(K)$.
- $t\nabla\mathcal{E}(K_i)^T\nabla\mathcal{E}(K_i)$: A proportional term to the squared norm of the gradient, $\|\nabla\mathcal{E}(K_i)\|^2$.

3.2.5 Iteration Process (summarized)

This section will explain the steps of using the purposed method in this work.

Step 1: Initialization

- Choose an initial guess for K_0 .
- Compute the **current characteristic polynomial**:

$$P_{\text{current}}(z) = \det(zI - (H_d - G_d K_0))$$

- Define the **desired characteristic polynomial**:

$$P_{\text{desired}}(z) = \prod_{i=1}^n (z - \lambda_i)$$

- Compute the **error function**:

$$\mathcal{E}(K_0) = P_{\text{current}}(z) - P_{\text{desired}}(z)$$

- Set initial parameters:
 - Step size $t = 1$
 - Constants $\alpha \in (0, 1)$ (e.g., $\alpha = 0.1$) and $\beta \in (0, 1)$ (e.g., $\beta = 0.5$)
 - Convergence tolerance $\varepsilon > 0$

Step 2: Compute Gradient and Hessian

At each iteration i , calculate:

- The **gradient** of the error function:

$$\nabla\mathcal{E}(K_i)$$

which measures the sensitivity of $\mathcal{E}(K)$ with respect to changes

- The **Hessian** matrix:

$$\nabla^2 \mathcal{E}(K_i)$$

Step 3: Determine the Newton Step

Using the gradient and Hessian, compute the Newton direction:

$$K_{i+1} = K_i - t_i (\nabla^2 \mathcal{E}(K_i))^{-1} \nabla \mathcal{E}(K_i)$$

Step 4: Perform Backtracking Line Search

$$\mathcal{E}(K_i - t \nabla^2 \mathcal{E}(K_i)^{-1} \nabla \mathcal{E}(K_i)) \leq \mathcal{E}(K_i) + \alpha t \nabla \mathcal{E}(K_i)^T \nabla \mathcal{E}(K_i)$$

Update the step size till the inequality is satisfied.

Step 5: Recalculation and repetition

Once the step size t satisfies the sufficient decrease condition, update K . Then recompute the current characteristic polynomial and update the new error function

$$\mathcal{E}(K_{i+1})$$

Repeat the steps until the gradient norm satisfies:

$$\|\nabla \mathcal{E}(K_i)\| \leq \varepsilon,$$

Chapter 4

Execution and Results

4.1 Simulation Parameters

A MATLAB code was created to simulate a generic second-order discrete-time linear system. For the normal deadbeat controller, a K matrix is calculated using MATLAB's Ackermann function, which provides the K matrix directly. Several parameters significantly impact the simulation result, and the following values were selected to imitate a real control problem. These variables are shared between both control approaches:

- L : Length of the pendulum, assumed to be 1.0 [m].
- g : Gravitational acceleration, 9.81 [m/s²].
- T_S : Sampling time, 0.1 [s].
- T_{tot} : Total simulation time, 10 [s].
- $X_1(0)$ and $X_2(0)$: Initial conditions, 0.1 (small angle) and 0, respectively.

4.1.1 Newton's Method Simulation Parameters

For Newton's method simulation, various parameters were chosen to ensure optimal results. These are detailed below:

1. **Maximum Number of Iterations:** The maximum number of iterations allowed. If the algorithm fails to converge before reaching this threshold, the simulation halts to prevent endless execution.

2. **Convergence Tolerance:** Defines the threshold for stopping the iteration. The error norm is compared to this value, and if smaller, the algorithm stops.
3. **Initial Step Size:** Determines the step size in the gradient direction. Larger steps may accelerate convergence but risk overshooting.
4. **Initial Guess for the Gain:** This was previously addressed.
5. **Damping Factor:** Functions in conjunction with the step size. If the step size leads to an error increase, the damping factor decreases the step size, promoting a more cautious update.
6. **Maximum Number of Damping Steps:** Sets a cap on the number of times the system can reduce the step size, preventing it from becoming too small.

A detailed list of all parameter values used in the simulations can be found in Appendix A.

4.2 Pole Placement Results

After running the simulation, the following results were obtained:

In Figure 4.1, the results of the pole placement method on the state trajectories of the closed-loop system are shown. The effect of the controller on x_1 (position) and x_2 (velocity) is illustrated over the simulation time. The system reaches the steady state (zero) for both states in approximately 1.21 seconds, meaning it took 12 steps to reach zero. A small oscillation can be observed, caused by both the initial conditions and the controller's effort to push the system to the steady state.

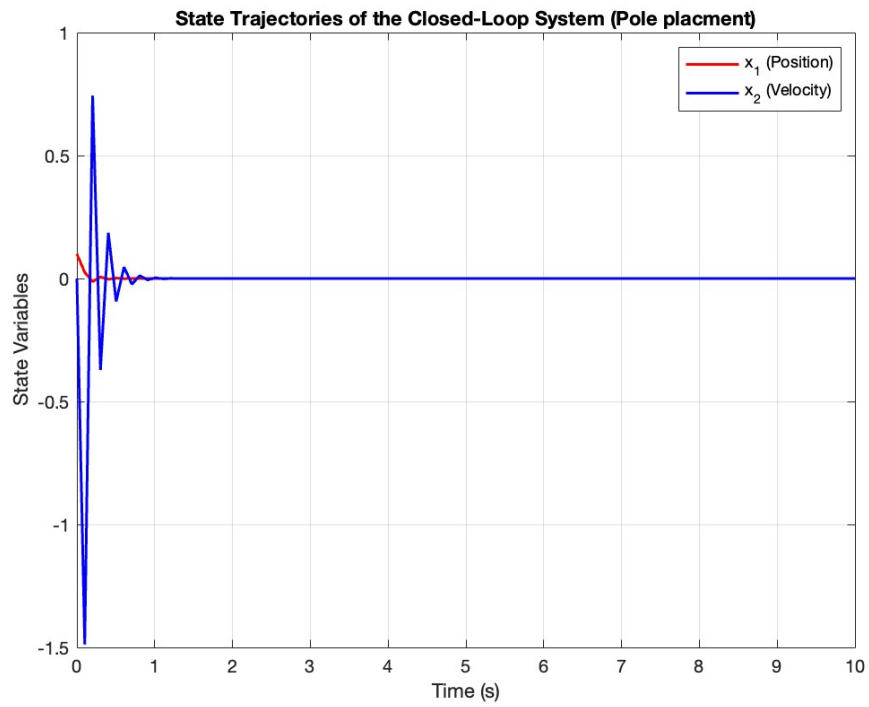
In the second part of Figure 4.1, the control effort over time is shown. The maximum control effort is 22.19.

4.3 Newton's Method Results

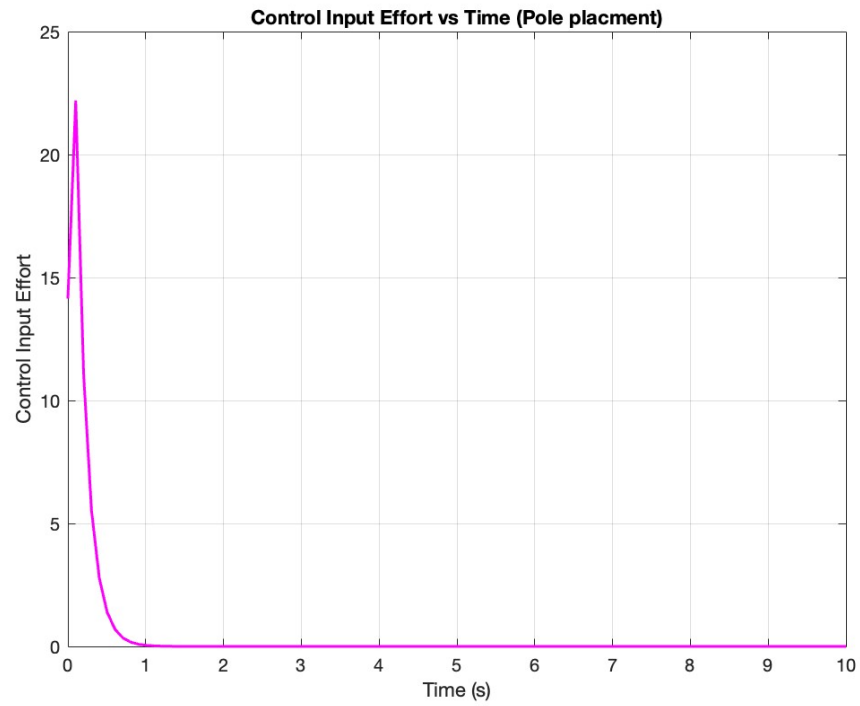
This section analyzes the simulation results in terms of state trajectories, control input effort, and performance, similar to the previous section.

For the trajectories in 4.2a of position and velocity, the system reaches the steady state (zero) in 5.6 seconds. The graph shows a relatively smooth trajectory with some oscillation in the velocity, which is well damped over time.

Regarding the control input effort 4.2b, it starts relatively high (around 1.2) but gradually decreases, reaching zero in the same time span.



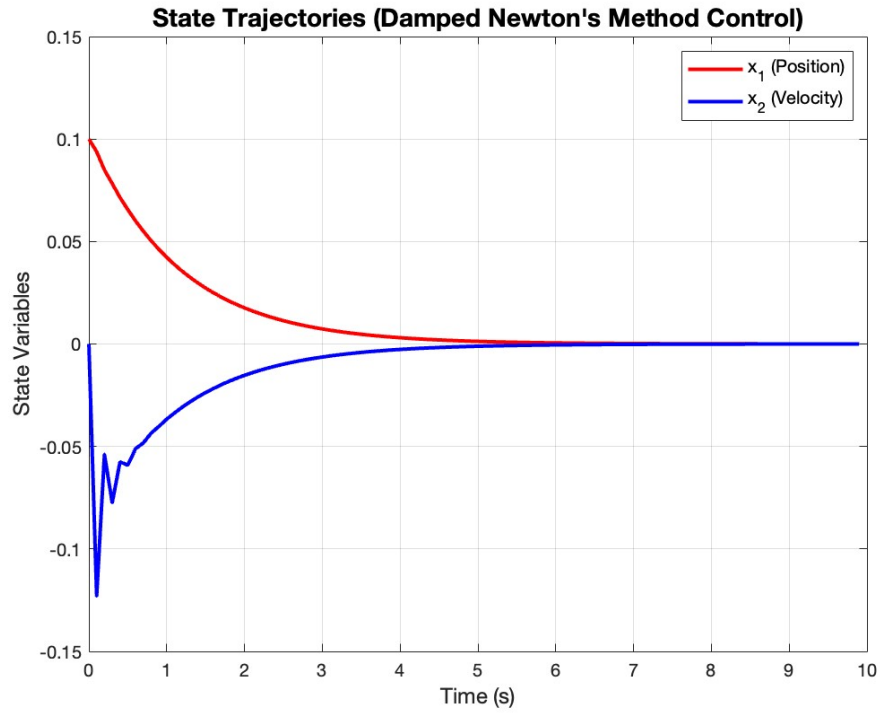
(a) State Trajectories of the Closed-Loop System (Pole Placement)



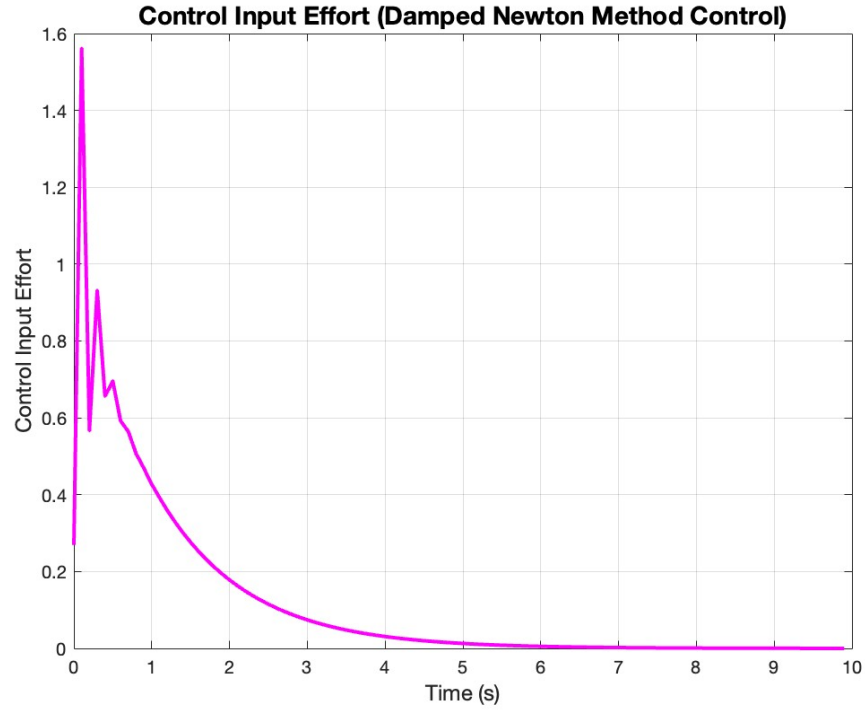
(b) Control Input Effort vs Time (Pole Placement)

Figure 4.1: Comparison of State Trajectories and Control Input Effort for Pole Placement.

While the system takes notably longer to reach the zero state than with the pole placement method, the results show that Newton's method effectively guides the system to the zero state without requiring sudden or excessive control inputs after the initial adjustments, aligning with the objectives of this work.



(a) State Trajectories of the Closed-Loop System (Newton's Method)



(b) Control Input Effort vs Time (Newton's Method)

Figure 4.2: Comparison of State Trajectories and Control Input Effort for Newton's Method.

4.3.1 Effect of Different Parameters

This section focuses on running simulations to test the impact of changing and applying modifications to the main parameter of the system. The results will show how changes in this parameter affect system behavior, and the control outcomes. When one parameter is altered, all other parameters will remain constant.

Initial guess

Figure 4.3 illustrates the impact of different initial guesses on the convergence rate and control input effort, both of which directly influence the system's behavior.

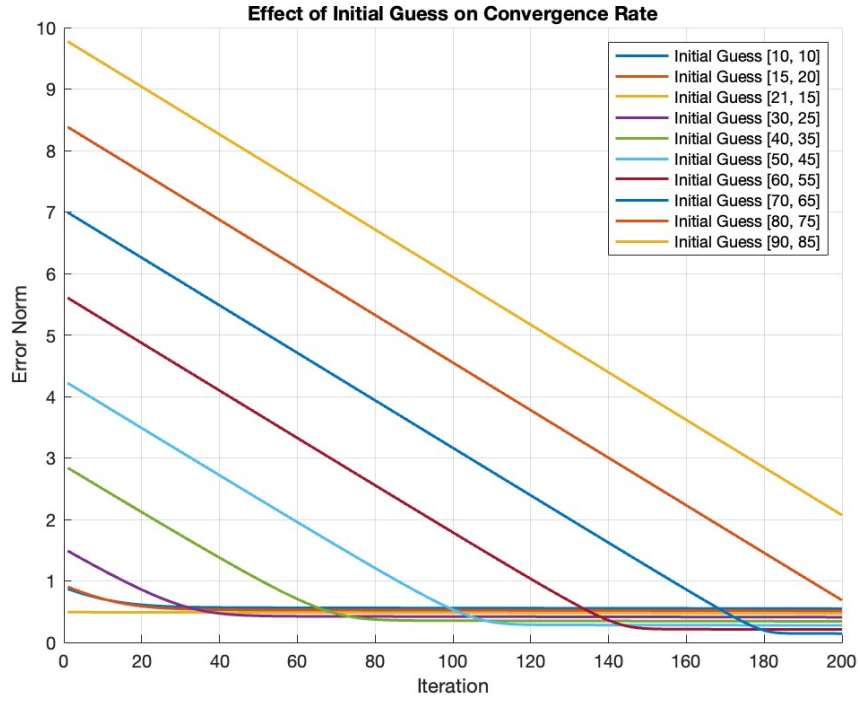
In the first part of the graph, each line represents a the convergence of error norm over the number of iterations. The error norm here is Euclidean distance between the current characteristic polynomial and the desired characteristic polynomial. The error equation that was previously driven at 3.17 can be further defined as

$$||E|| = \sqrt{\sum_{i=1}^n (P_{\text{current},i} - P_{\text{desired},i})^2} \quad (4.1)$$

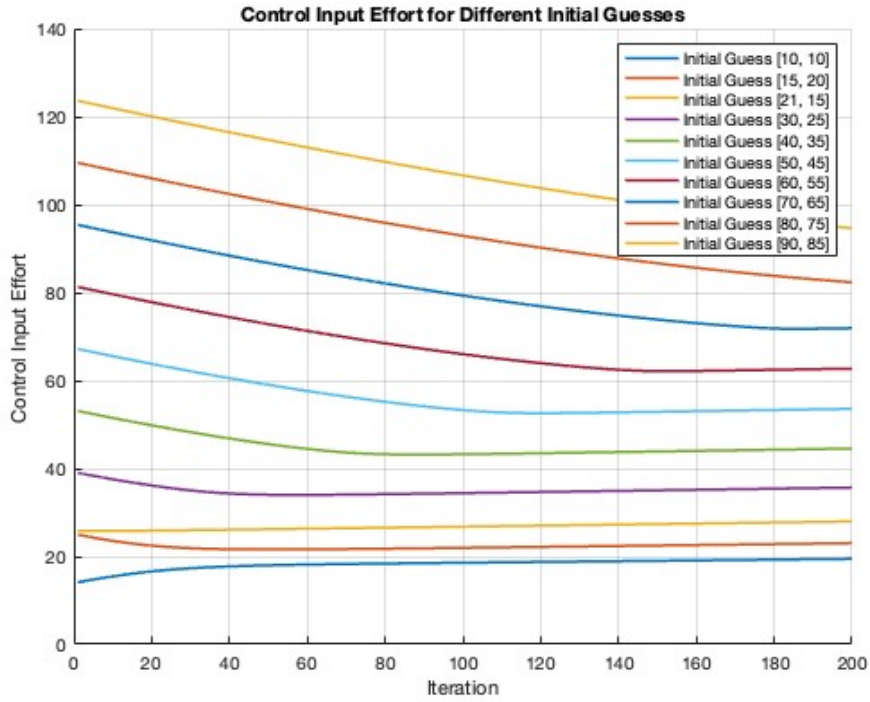
Thus the rate of which the error norm decreases is an indication of how quickly the system reaches the desired state. The convergence rate is highly impacted by how far is the initial guess from the optimal point. As previously indicated a better initial guess will result in a smaller error at the beginning.

$$E_0 = A_d - B_d K_0 - P_{\text{desired}} \quad (4.2)$$

Meaning when the error is smaller a less number of iteration is needed to reach the desired state, which also will reduce the computations and the time needed to reach that state. With that being said, from second part of the graph 4.3b it shows that there is a trade-off in initial control input effort and convergence speed. A higher initial guess will result on faster and more aggressive controller which will require a higher input value.



(a) Impact of different initial guess on the convergence rate



(b) Impact of different initial guess on the control effort

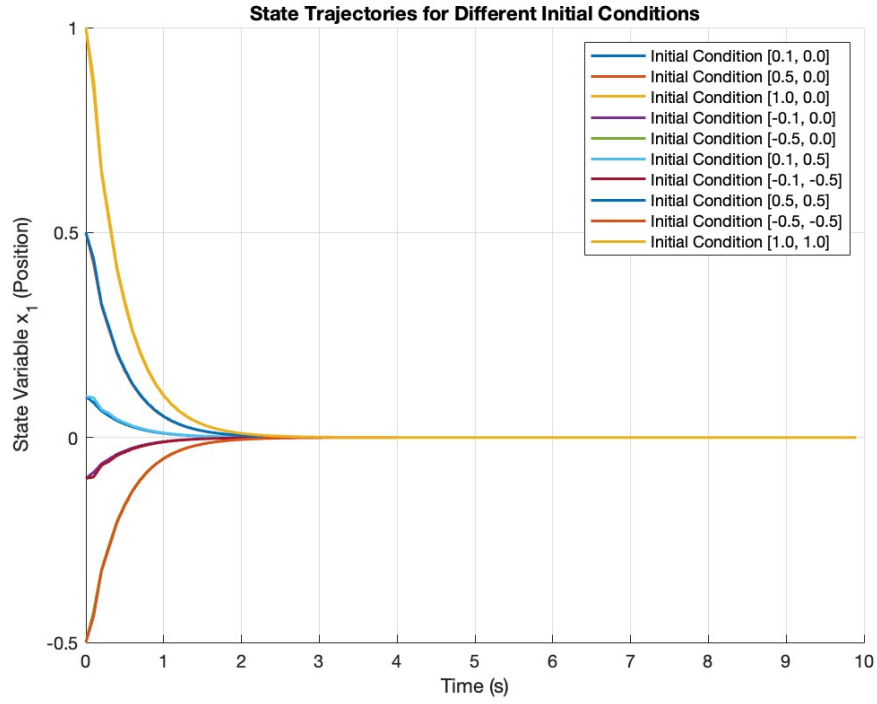
Figure 4.3: Comparison of State Trajectories and Control Input Effort for different initial guess.

Initial conditions

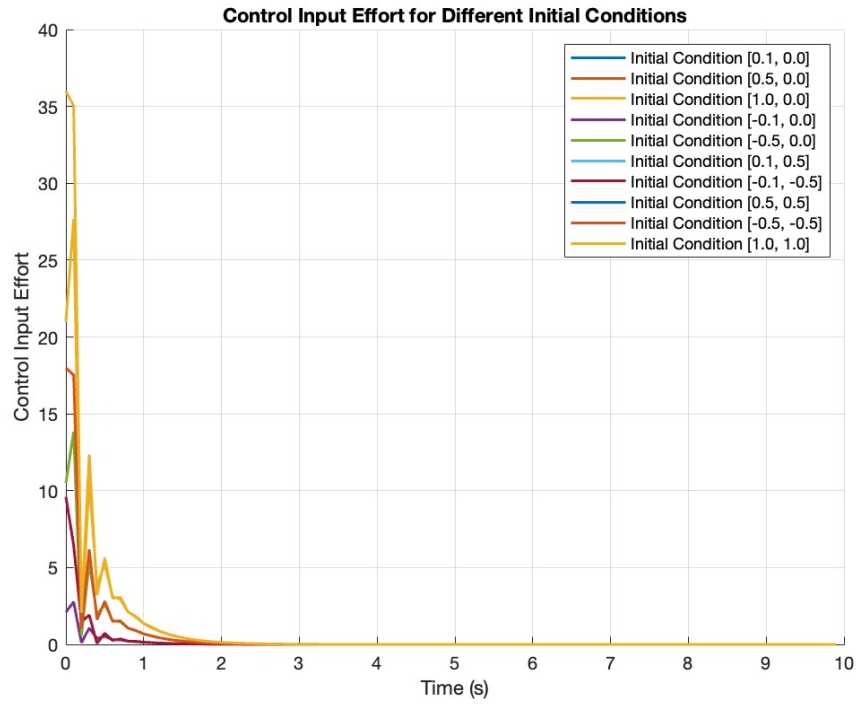
For this part of the simulation testing a number of initial conditions were given. The result is showing in the graph [4.4](#). Where the system proves to make all trajectories converge to zero. In another words regardless of the initial conditions, the system will reach the desired state as far as good initial guess were made. In addition as expected, the larger is the deviations from the equilibrium the oscillatory behavior will occur, meaning longer convergence times, and higher initial control efforts.

Sample time

A series of sampling time were used to test the impact of on the system behaviour. [4.6](#) Starting with $T_s = 0.05s$ and gradually increasing to $T_s = 0.2s$, the system initially exhibits well-controlled behavior with minimal oscillation at smaller sample times. As the sample time increases, higher oscillations, a sharper dip in control effort, and a shorter settling time become evident. Last case where the $T_s = 1.0s$ shows a sever oscillations that drive the system to higher instability by time. This oscillations grows in magnitude, and the system will not reach the desired state. This is the result of not updating the controller fast enough. In the control effort chart [4.6d](#) the system starts applying control after a high oscillations already occurred. A conclusion of this that another trade-off is taking place. Smaller sampling time will result in less energy consumption, more settling time and better stability.

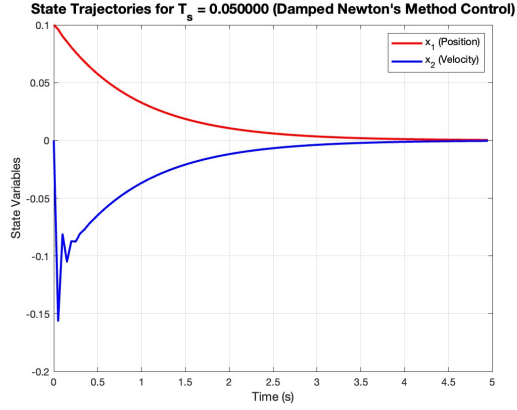


(a) Impact of different initial conditions on the convergence rate

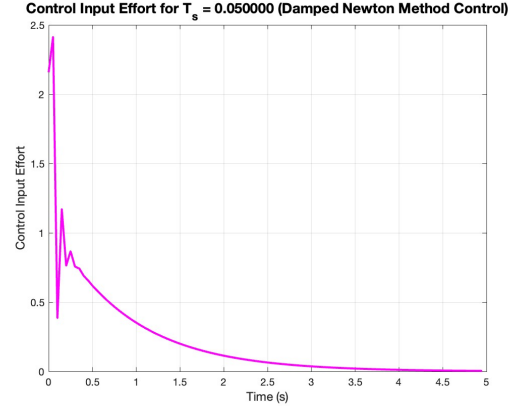


(b) Impact of different initial conditions on the control effort

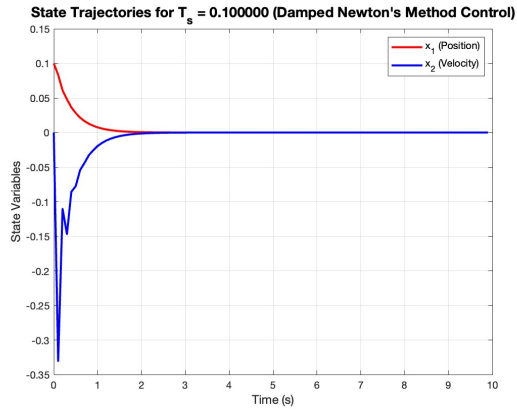
Figure 4.4: Comparison of State Trajectories and Control Input Effort for different initial conditions



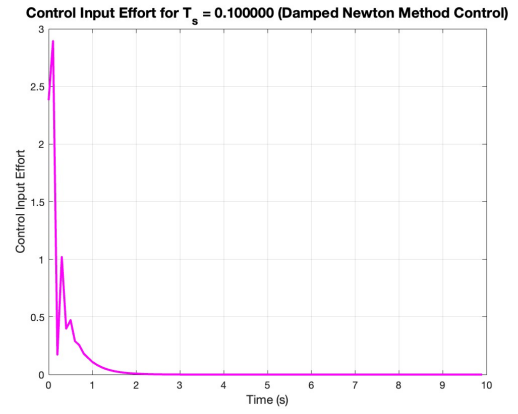
(a) State Trajectories for $T_s = 0.05s$



(b) Control Input Effort for $T_s = 0.05s$

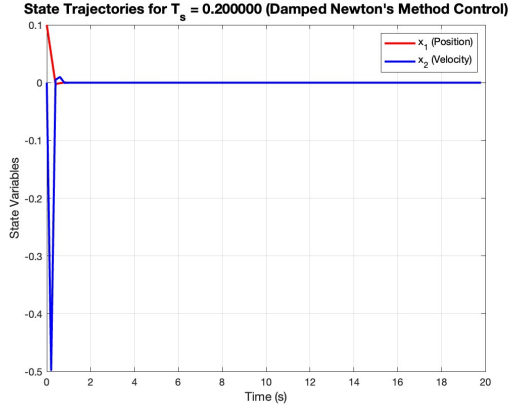


(c) State Trajectories for $T_s = 0.1s$

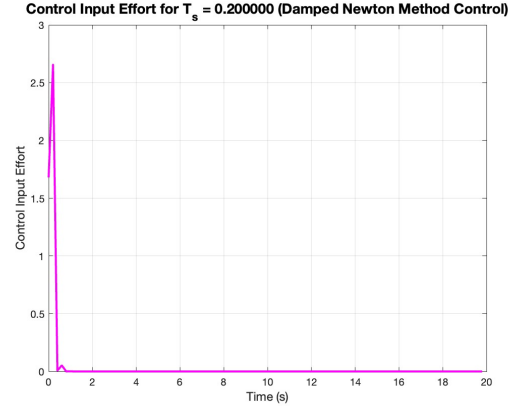


(d) Control Input Effort for $T_s = 0.1s$

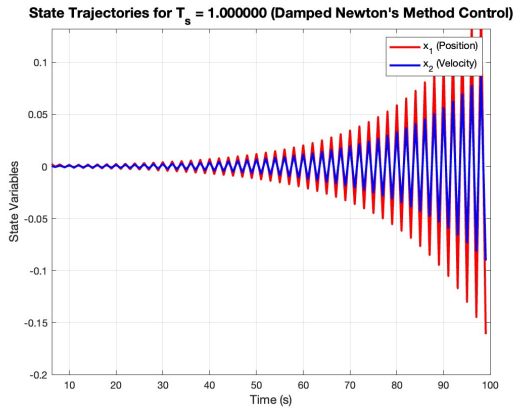
Figure 4.5: State trajectories and control input efforts for different sampling times (T_s).



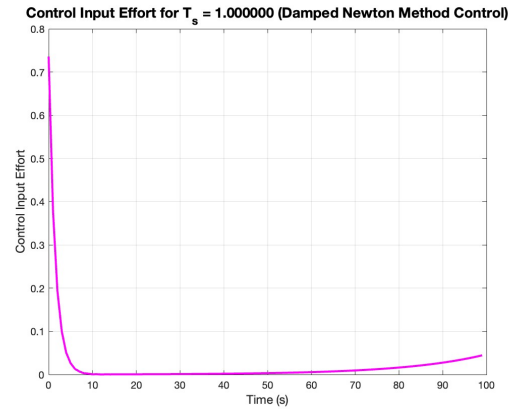
(a) State Trajectories for $T_s = 0.2s$



(b) Control Input Effort for $T_s = 0.2s$



(c) State Trajectories for $T_s = 1.0s$



(d) Control Input Effort for $T_s = 1.0s$

Figure 4.6: State trajectories and control input efforts for different sampling times (T_s).

Chapter 5

Conclusion

While pole placement resulted in faster convergence, it required significantly higher control effort. For limited energy systems or systems that cannot handle such high input or strong overshoot, this will not be suitable. On the other hand, Newton's method was shown to be effective as a deadbeat control. The results show a less disturbed trajectory while requiring more settling time.

Newton's method could be used as an alternative to traditional deadbeat controllers, particularly in systems where reaching the steady state in minimal time with smooth transitions is critical. A usage example will be in the case of servo motors control as the system provides minimal overshoot that will allow precise and fast positioning, a main application of that will be in robotics systems and CNC machines. In more advanced usage can be for high-speed permanent magnet synchronous motor (PMSM) drivers mainly in the context of current control.

Future work could investigate advanced damping strategies to enhance this method, or more adaptive control mechanisms to improve both the convergence rate and maintain control effort at the same time.

Bibliography

- [1] J. Ackermann. *Sampled-Data Control Systems*. Springer Verlag, Berlin, Heidelberg, 1989.
- [2] James Crowe, GR Chen, R Ferdous, DR Greenwood, MJ Grimble, HP Huang, JC Jeng, Michael A Johnson, MR Katebi, S Kwong, et al. *PID control: new identification and design methods*. Springer, 2005.
- [3] M.R. Marrari, A. Emami-Naeni, and G.F. Franklin. Output deadbeat control discrete-time multivariate systems. *IEEE Transactions on Automatic Control*, 34(6):644–648, 1989.
- [4] Chi-Tsong Chen. *Linear system theory and design*. Saunders college publishing, 1984.
- [5] K. Ogata. *Discrete-time Control Systems*. Prentice-Hall, Englewood Cliffs, 1987.
- [6] B Porter and A Bradshaw. Design of dead-beat controllers and full-order observers for linear multivariable discrete-time plants. *International Journal of Control*, 22(2):149–155, 1975.
- [7] Rudolf Emil Kalman, Peter L Falb, and Michael A Arbib. *Topics in mathematical system theory*, volume 1. McGraw-Hill New York, 1969.
- [8] V Kucera and Michael Sebek. On deadbeat controllers. *IEEE transactions on automatic control*, 29(8):719–722, 1984.
- [9] JE Ackermann. Pole placement control. *Control System, Robotics and Automation*, 8(2011):74–101, 2009.
- [10] Taha HS Abdelaziz and M Valášek. A direct algorithm for pole placement by state-derivative feedback for single-input linear systems. *Acta Polytechnica*, 43(6), 2003.
- [11] Daniel Ralph. Global convergence of damped newton’s method for nonsmooth equations via the path search. *Mathematics of operations research*, 19(2):352–389, 1994.

- [12] Daniel M Murray and Sidney J Yakowitz. Differential dynamic programming and newton's method for discrete optimal control problems. *Journal of Optimization Theory and Applications*, 43(3):395–414, 1984.
- [13] Markus Giftthaler, Michael Neunert, Markus Stäuble, Jonas Buchli, and Moritz Diehl. A family of iterative gauss-newton shooting methods for nonlinear optimal control. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.
- [14] Vladimir M Veliov. Newton's method for problems of optimal control of heterogeneous systems. *Optimization Methods and Software*, 18(6):689–703, 2003.
- [15] Haoyang Deng and Toshiyuki Ohtsuka. A highly parallelizable newton-type method for nonlinear model predictive control**this work was partly supported by jsps kakenhi grant number 15h02257. *IFAC-PapersOnLine*, 51(20):349–355, 2018. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.
- [16] Haoyang Deng and Toshiyuki Ohtsuka. A parallel newton-type method for nonlinear model predictive control. *Automatica*, 109:108560, 2019.
- [17] Kai Pfeiffer, Adrien Escande, Pierre Gergondet, and Abderrahmane Kheddar. The hierarchical newton's method for numerically stable prioritized dynamic control. *IEEE Transactions on Control Systems Technology*, 31(4):1622–1635, 2023.
- [18] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 4. Athena scientific, 2012.
- [19] Immanuel M Bomze, Vladimir F Demyanov, Roger Fletcher, Tamás Terlaky, and Roger Fletcher. The sequential quadratic programming method. *Nonlinear Optimization: Lectures given at the CIME Summer School held in Cetraro, Italy, July 1-7, 2007*, pages 165–214, 2010.
- [20] AS El-Bakry, Richard A Tapia, T Tsuchiya, and Yin Zhang. On the formulation and theory of the newton interior-point method for nonlinear programming. *Journal of Optimization theory and Applications*, 89(3):507–541, 1996.
- [21] STANISLAW H ŻAKJ and Carl A Maccarley. State-feedback control of non-linear systems. *International Journal of Control*, 43(5):1497–1514, 1986.

- [22] Larry Armijo. Minimization of functions having lipschitz continuous first partial derivatives.
Pacific Journal of mathematics, 16(1):1–3, 1966.

Appendix A

Newton's Method Simulation Parameters

Parameter	Value
System Parameters	
Length of the pendulum (m)	1.0
Acceleration due to gravity (m/s^2)	9.81
Sampling time (s)	0.1
Desired Characteristic Polynomial	
Desired characteristic polynomial	$[0, 0, 0]$
Damped Newton's Method Parameters	
Initial guess for feedback gain	$[0.2, 1.5]$
Initial step size	1
Damping factor	0.5
Perturbation for gradient calculation	1×10^{-5}
Maximum number of damping steps	10
Maximum number of iterations	200
Convergence tolerance	1×10^{-3}
Steady State Detection and Simulation Parameters	
Steady state tolerance	1×10^{-3}
Number of simulation steps	100
Initial state of the system (position, velocity)	$[0.1, 0]$

Table A.1: Newton's Method Simulation Parameters