Nonlinear Model Predictive Control : An Implementation using CasADi

Santosh Rajkumar

The Ohio State University

Abstract

This project delves into the implementation of Nonlinear Model Predictive Control (NMPC) using CasADi within the MATLAB environment, leveraging its capabilities in numerical optimization and automatic differentiation. Theoretical foundations covering MPC, non-linear programming, and transcription methods are discussed to provide a comprehensive understanding. A demonstration of the NMPC scheme is conducted utilizing a non-linear system featuring an inverted pendulum with a spring-damper setup. The NMPC implementation for the pendulum system is showcased across various initial conditions. Additionally, the project investigates the robustness of MPC in handling parameter uncertainties and noise within the pendulum system.

Contents

1 Introduction **1**

1.1 Introduction to Model Predictive Control 1

1.2 Nonlinear Programming 4

1.3 Transcription Methods 5

1.4 CasADi 7

2 The System Under Consideration **9**

2.1 Modeling of the Pendulum System 10

3 Direct Multiple Shooting Method **13**

3.1 Implementation of Direct Multiple Shooting Method 13

3.2 Solving the NLP and Implementing NMPC 17

4 Results **19**

4.1 Discussion 22

4.2 MATLAB Code 24

5 Conclusions **25**

List of Figures

1 A block diagram of MPC implementation 2

2 Reference tracking using MPC (adapted from [2]) 3

3 CasADi facilitating solution to OCPs 8

4 The nonlinear system under consideration 9

5 Free body diagram of the pendulum system 10

6 The phase portrait of the pendulum system (autonomous case) 11

7 Pictorial description of continuity constraints in multiple shooting method 15

8 Stabilization of the pendulum with initial conditions $\theta = 35, \dot{\theta} = -5.73^o/sec$ 19

9 Stabilization of the pendulum with initial conditions $\theta = -20^o, \dot{\theta} = -5.73^o/sec$ 20

10 Stabilization of the pendulum with parameter mismatch considering conditions $\theta = 35^o, \dot{\theta} = -5.73^o/sec$ 21

11 the stabilization of the pendulum in the presence of both parameter mismatch and noisy measurements with initial conditions $\theta = 35^o, \dot{\theta} = -5.73^o/sec$ 21

List of Tables

1 Physical parameters of the pendulum system 10

2 The average computational time taken for solving the NLP problem in NMPC 23

1 Introduction

In recent years, Model Predictive Control (MPC) has gained popularity among researchers. One of the primary reason being the fact that MPC enables the incorporation of constraints into the control problem making it easier to account for real-world limitations. Further, MPC involves low tuning effort because it integrates a model directly into the optimization process, which takes care of the tuning effort to a great extent. Moreover, MPC inherently offers robust control, as it recalculates an optimal solution for each time step, accommodating disturbances as they occur. Comparative studies presented in [1] have highlighted the strengths and weaknesses of various control methods including MPC. For instance, while PID control requires significant tuning effort and does not consider actuator limits, robust control balances performance and robustness, albeit faces diffculties with actuator constraints. Input-output linearization offers easy analysis of controller performance and stability but may not be suitable for scenarios involving nonlinear dynamics and unstable equilibriums. However, like other methods, MPC finds drawback in its computational complexity, which arises from considering complex system dynamics and solving optimization problems over an entire time horizon in a repeated manner. Nevertheless, with the computing hardware advancement, real-time implementation of MPC has become reality.

1.1 Introduction to Model Predictive Control

Model Predictive Control (MPC) involves optimizing a cost function to track a reference over a time horizon (t_N). Using current measurements and predicted future outputs, MPC calculates optimal control inputs for the horizon by minimizing the cost function based on a system model. These optimal inputs are then applied at each time step, and the process repeats for the next step, with the horizon shifting along with the current time step, creating a receding horizon. MPC's predictive capability enables it to consider future time

steps in its calculations. Since MPC involves an optimization to compute control input at each time instant, it produces discrete control outputs that we can feed to practical systems (continuous time) using a zero-order hold (ZOH). Figure 1 shows a block diagram for MPC implementation where $\mathbf{x}(t)$ represents the system states (considered available through measurement), $\mathbf{x}_k$ is the states at measurement instant t, and $\mathbf{u}_k^*$ is the optimal control for the instant t which is fed to the plant as $\mathbf{u}^*(t)$ through the ZOH circuit. Figure 2 shows MPC tracking a reference using predicted optimal control actions and states over the receding time horizon (t_N) divided into N control intervals (usually equally spaced with an interval of $dT = t_N/N$ sec). This optimal control sequence is determined by solving an optimization problem that seeks to minimize a cost function while satisfying system dynamics and constraints over the prediction horizon. The reference trajectory represents the desired behavior of the system, which could be a setpoint trajectory, a reference signal, or a trajectory generated by another controller. MPC tries to adjust the control actions based on the predicted future states to ensure that the actual system behavior closely follows the reference trajectory.

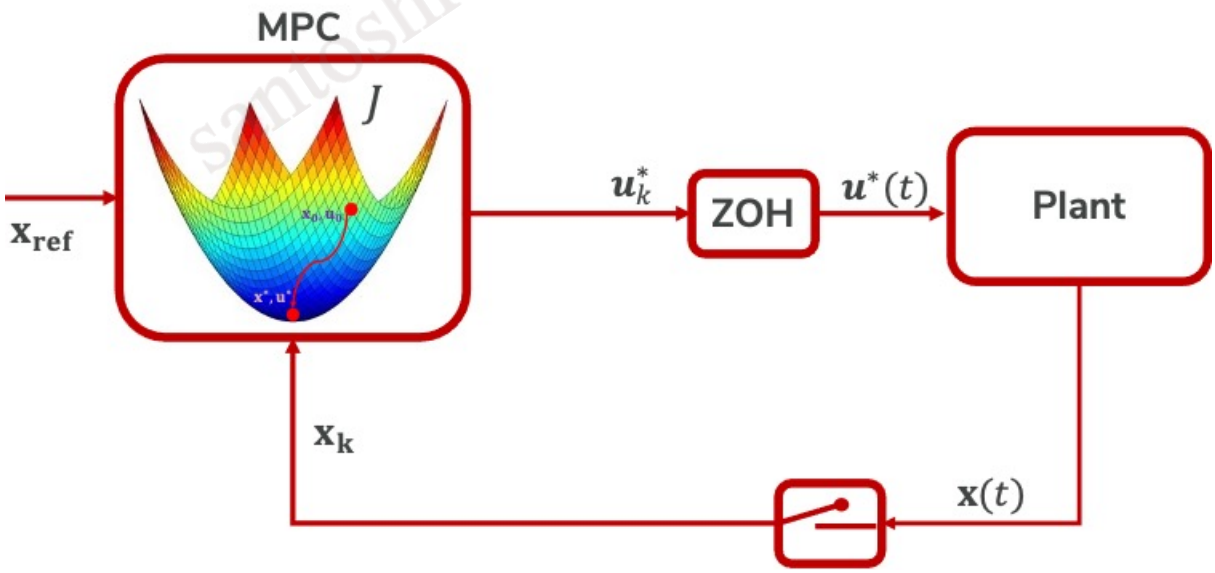

Figure 1: A block diagram of MPC implementation

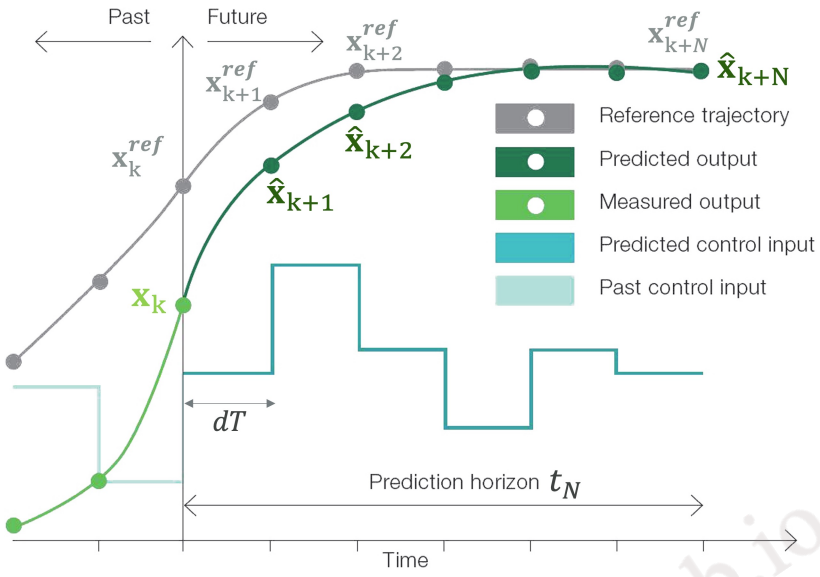

Figure 2: Reference tracking using MPC (adapted from [2])

Consider the dynamics of a system being controlled is,

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(\mathbf{t}), \mathbf{u}(\mathbf{t})) \tag{1}$$

with the sampled measured states (assuming all the states are available for measurement) at any time instant t_k is $\mathbf{x}_k$. Then, the finite horizon optimal control problem (OCP) at each instant is given by,

$$\min\ J(x,u) = \int_0^{t_N} ||\mathbf{x} - \mathbf{x}_{ref}||^2_{\mathbf{Q}} + ||\mathbf{u} - \mathbf{u}_{ref}||^2_{\mathbf{R}}$$

subject to :

$$\begin{aligned} &\mathbf{x} \in \boldsymbol{\mathcal{X}}, \mathbf{u} \in \boldsymbol{\mathcal{U}} \\ &\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(\mathbf{t}), \mathbf{u}(\mathbf{t})) \\ &\mathbf{x}(0) = \mathbf{x}_k \end{aligned} \tag{2}$$

where, $\boldsymbol{\mathcal{X}}$ is the set of admissible states, $\boldsymbol{\mathcal{U}}$ is the set of admissible control, $\mathbf{x}_{ref}$ is the vector of desired states, $\mathbf{u}_{ref}$ is the vector of desired control (we want to keep it as minimum

as possible. Keeping $\mathbf{u}_{ref} = \mathbf{0}$ will give $\mathbf{u}^T\mathbf{R}\mathbf{u}$), $\mathbf{Q}$ and $\mathbf{R}$ represents the weighting on tracking the reference states and penalising the input respectively as we do in standard LQR techniques. $\mathbf{Q}$ and $\mathbf{R}$ are usually diagonal square matrices.

At each sampled time instant t_k, the MPC schemes solves the OCP in (2) and generates the optimal controls and the corresponding states based on the modeled dynamics for the time horizon t_H into the future, and applies the optimal control generated for the current time instant t_k into the system. If $\mathbf{f}(.)$ in (2) is non-linear w.r.t $\mathbf{x}$ and $\mathbf{u}$, we regard the MPC scheme as **non-linear MPC (NMPC)**. Similarly, if $\mathbf{f}(.)$ in (2) is linear w.r.t $\mathbf{x}$ and $\mathbf{u}$, we regard the MPC scheme as **linear MPC (LMPC)**. An MPC control approach can be likened to driving a car. Similar to a driver following a trajectory within a lane, MPC considers a reference path over a set horizon. Drawing from past driving experiences, the driver possesses a mental model of the car's behavior in response to various inputs. While the control action is executed at each time step, it's also pre-planned, like anticipating braking before a turn.

The common methods widely used for handling Nonlinear Model Predictive Control (NMPC) problem are Sequential Quadratic Programming (SQP) and the Interior-Point Method (IPM). For the available solvers (SQP/IPM) to solve the OCP in (2), the OCP needs to be casted as a Non-linear Programming problem (NLP) or Quadratic Programming (QP) problem [3]. NLP can be thought of a superset of QP. In this report, we are considering NLP with IPM for NMPC implementation in CaSADi framework.

1.2 Nonlinear Programming

Nonlinear programming involves the optimization of a nonlinear objective function while considering various types of constraints, such as bound constraints, linear constraints, or nonlinear constraints [3]. These constraints may take the form of inequalities or equalities. The optimization problem described in (2) must be reformulated into a specific formulation

suitable for a dedicated Nonlinear Programming (NLP) solver. A transcription method is employed to convert the optimization problem into an NLP. Consider a general form of an NLP,

$$\begin{aligned} &\min_{\mathbf{w}} \ \Phi(\mathbf{w}) \\ &\text{subject to: } g_i(\mathbf{w}) \leq 0 \text{ for each } i \in \{1, 2, \ldots, m\} \\ &\qquad\qquad h_j(\mathbf{w}) = 0 \text{ for each } j \in \{1, 2, \ldots, n\} \end{aligned} \tag{3}$$

Therefore, we need to represent the decision variables $\mathbf{w}$ as discrete set of states and control inputs for which the NLP optimization problem needs to be solved provided the constraints are satisfied. The objective function $\Phi(\mathbf{w})$ depicts the mapping from the decision vector to the cost function J in (2). Once we formulate our OCP in (2) as an NLP, we can use dedicated solver. Increased attention towards effective optimization techniques has spurred the advancement of interior-point or barrier methods (IPMs) tailored for large-scale nonlinear programming [4]. These approaches offer a compelling alternative to active set strategies when addressing problems characterized by numerous inequality constraints. IPMs implemented within both trust region and line-search frameworks, have been devised to facilitate convergence even from unfavorable initial conditions. These methods utilize exact penalty merit functions to compel advancement towards the solution. A commonly used IPM based open-source solver is Ipopt (Interior Point OPTimizer) [Github Page] available across different platforms and operating systems.

1.3 Transcription Methods

Transcription methods are the ways in which we convert optimal control problems (OCPs) into parameter optimization problems[2]. For OCPs, we can broadly categorize transcription methods into two types [2], [5]:

- **Direct methods:** Direct methods transform the Optimal Control Problem (OCP) into a Nonlinear Programming (NLP) problem by discretizing the independent vari-

able and employing a numerical integration technique to discretize the continuous-time dynamics. Subsequently, the NLP is solved using an NLP solver.

- **Indirect methods:** Indirect methods rely on Pontryagin's Maximum Principle (PMP), utilizing necessary optimality conditions to derive a boundary value problem. This problem needs to be solved in order to determine the optimal control.

Indirect transcription methods, as they are based on PMP, are disadvantageous at times because it is hard to find an initial guess for the so called adjoint variable. Therefore, they are ill-suited for inequality constraints. The main advantage of using direct methods are in handling the inequality constraints reasonably well. Direct transcription methods can be further classified into the following types:

- **Direct single shooting:** In direct single shooting, only the discrertized control trajectories are taken as decision variables for the NLP. The continuous-time dynamics are discretized by numerically integrating the state trajectory. This integration occurs in a single sweep from the initial time instant to the final time.

- **Direct multiple shooting:** In direct multiple shooting, the discretized control trajectories and state trajectories are taken as the decision variables for the NLP. The time horizon is converted into a finite number of intervals and the continuous-time dynamics are discretized by numerically integrating the state trajectory for each interval taking the control signal piecewise constant for each time-interval.

- **Direct collocation:** This method is similar to multiple shooting with the time horizon is converted into a finite number of points called collocation points. The state and control variables are approximated by piecewise polynomial functions, such as splines, between the collocation points.

Direct single shooting method performs poorly when the underlying system dynamics is unstable [5]. In contrast to single shooting methods, multiple shooting methods gener-

ally exhibit improved convergence of Nonlinear Programming (NLP) problems and greater numerical stability. This means they are less susceptible to errors stemming from factors like poorly chosen initial guesses. Direct collocation offers advantages in handling system dynamics that are not explicitly known or that may have complex functional forms, as it directly incorporates these dynamics into the optimization problem. Implementing direct multiple shooting is often simpler compared to direct collocation and multiple shooting can handle large volume of constraints compared to direct collocation method. In this report we will implement multiple shooting method and discuss in detail.

1.4 CasADi

CasADi is an open-source software designed for nonlinear optimization and algorithmic differentiation [6]. It enables quick and efficient implementation of various techniques for numerical optimal control, suitable for both offline applications and nonlinear model predictive control (NMPC). CasADi relies on a symbolic framework that incorporates both forward and reverse modes of Algorithmic Differentiation (AD) on expression graphs. This framework facilitates the construction of gradients, large-and-sparse Jacobians, and Hessians. The expression graphs, encapsulated within Function objects, can be assessed within a virtual machine or exported as standalone C code. Aside from directly formulating problems within CasADi, it offers the capability to import Nonlinear Programming Problems (NLPs) formulated in the algebraic modeling language AMPL. AMPL comprises a closed-source parser and an open-source runtime called the AMPL Solver Library (ASL). Through the use of the .nl file format, CasADi can import NLPs formulated in AMPL. This format's symbolic nature enables CasADi to construct a symbolic representation of the NLPs, which can then be manipulated and solved using NLP solvers within or interfaced with CasADi. Additionally, .nl files can be generated using the open-source modeling environment Pyomo. Furthermore, CasADi supports model import from the physical modeling language

Modelica via the open-source compiler JModelica.org, utilizing an extension of the standard Functional Mock-up Interface (FMI) standard. This import capability allows CasADi to establish a symbolic representation of the Optimal Control Problems (OCPs), enabling various symbolic manipulations such as scaling, variable and equation sorting, and reformulation of Differential Algebraic Equations (DAEs) from fully-implicit to semi-explicit form. For the scope of this project, CasADi acts as a facilitator in converting our OCP like in (2) to an NLP and solving it using a standard NLP solver (e.g., IPOPT) as depicted in Figure 3.

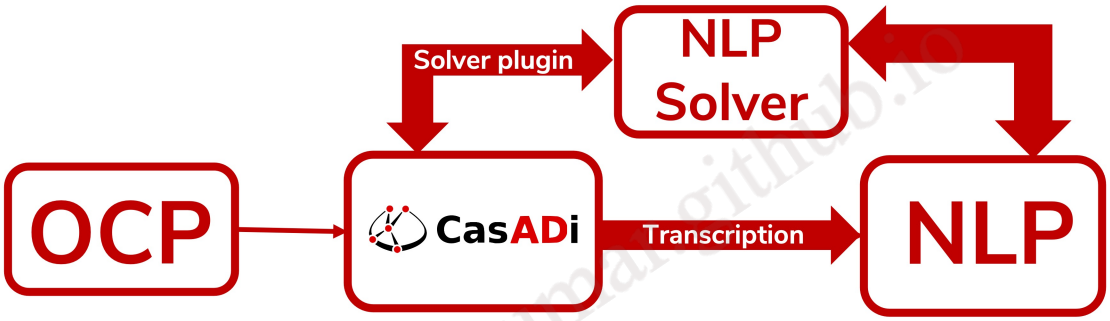

Figure 3: CasADi facilitating solution to OCPs

2 The System Under Consideration

In this report, we focus on studying a nonlinear system consisting of an inverted pendulum coupled with a spring-damper arrangement as shown in Figure 4. The pendulum consists of a mass-less rod of length l with a mass m attached to the tip of the rod. The spring-damper arrangement is attached to the pendulum at a length $d < l$. The pendulum's angle (θ) is measured from the vertical with the positive direction as shown in Figure 4. At the base of the pendulum (O) we have an arrangement that provides moment or torque (M) input to the pendulum. Here, c is the damping co-efficient and k is the spring constant. We consider that due to the spring-damper arrangement, the pendulum's motion is restricted as $-40^o \leq \theta \leq 40^o$. Our objective is to balance the pendulum around $\theta = 0^o$.

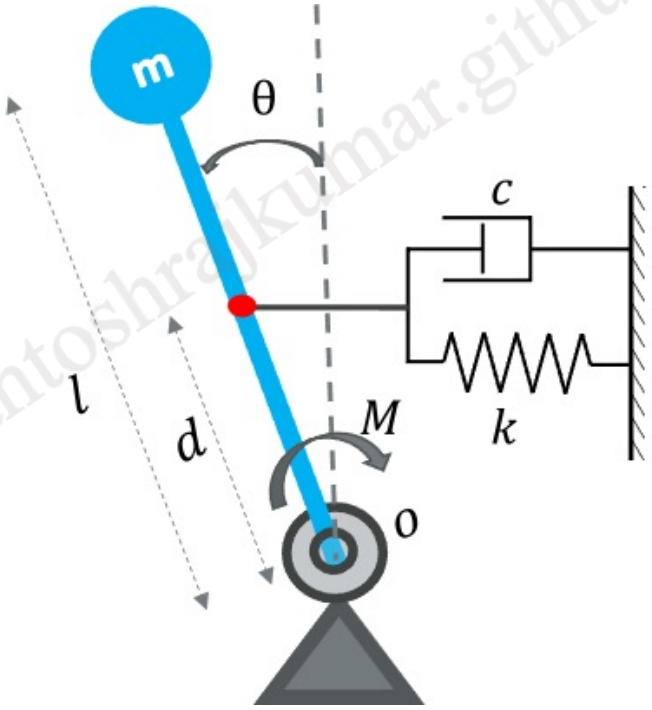

Figure 4: The nonlinear system under consideration

The physical parameters of the pendulum system are given in Table 1.

Table 1: Physical parameters of the pendulum system

Parameter	Value	Unit
m	0.15	kg
l	4	m
d	3.5	m
k	0.3	N/m
c	0.05	Ns/m
g	9.81	m/s^2

2.1 Modeling of the Pendulum System

We model the pendulum system in Figure 4 using the laws of physics. The free body diagram of the pendulum system is shown in Figure 5.

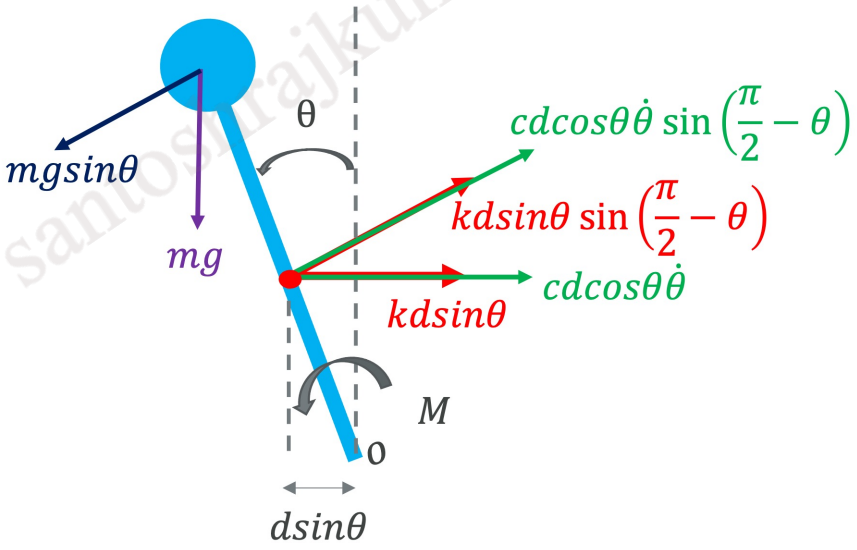

Figure 5: Free body diagram of the pendulum system

The moment of inertia of the pendulum $\mathcal{J}$ is ml^2. Using the free body diagram, we write the equation for the balance of moments at the base O as,

$$\begin{aligned} \circlearrowleft + \sum \mathcal{M} &= \mathcal{J}\ddot{\theta} \\ \implies M + mglsin\theta - kd^2sin\theta cos\theta - cd^2cos^2\theta\dot{\theta} &= ml^2\ddot{\theta} \end{aligned} \tag{4}$$

Therefore the equation of motion of the pendulum system is given as,

$$ml^2\ddot{\theta} + kd^2sin\theta cos\theta + cd^2cos^2\theta\dot{\theta} - mglsin\theta = M \tag{5}$$

Considering $x_1 = \theta$, $x_2 = \dot{\theta}$, and $u = M$, we can write (5) in state-space form as,

$$\begin{aligned} \dot{x_1} &= x_2 \\ \dot{x_2} &= -\frac{kd^2}{ml^2}sinx_1sinx_2 - \frac{cd^2}{ml^2}cos^2x_1x_2 + \frac{g}{l}sinx_1 + \frac{u}{ml^2} \end{aligned} \tag{6}$$

The phase portrait of the pendulum system without external control u can be seen in Figure 6. We see that it is an inherently unstable system around $(x_1, x_2) = (0, 0)$ equilibrium point.

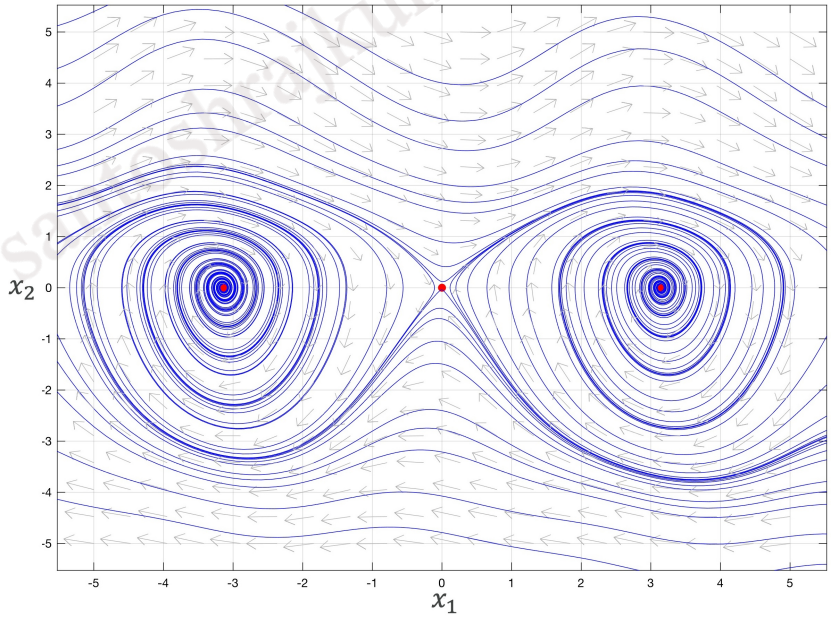

Figure 6: The phase portrait of the pendulum system (autonomous case)

Considering $\mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$, we can write (6) in compact form as,

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), u(t)) \tag{7}$$

where,

$$\mathbf{f} = \begin{bmatrix} x_2 \\ -\frac{kd^2}{ml^2} sinx_1 sinx_2 - \frac{cd^2}{ml^2} cos^2 x_1 x_2 + \frac{g}{l} sinx_1 + \frac{u}{ml^2} \end{bmatrix}$$

3 Direct Multiple Shooting Method

As previously mentioned, the primary objective of multiple shooting method is to convert the Optimal Control Problem (OCP) into a Nonlinear Programming (NLP) problem. This conversion process involves discretizing the independent variable, typically time, into a finite number of intervals. Within each interval, the control signal is parameterized, often as a constant, resulting in a piecewise-constant control signal across the entire time span. Next, the continuous-time dynamics are discretized separately for each interval using a numerical integration routine, involves integrating the system's dynamics individually for each segment. During integration, the system dynamics receives the constant control signal value specified for that segment. Since the system dynamics are integrated separately for each time interval, the overall state trajectory may not be continuous. To address this discontinuity, a constraint is introduced in the NLP formulation to bind the trajectories together. This constraint ensures continuity in the state trajectory across the entire time span. Through the discretization of time, parameterization of control signals, and integration of dynamics within each segment, the original OCP is transformed into an NLP, allowing for the application of NLP solvers to find the optimal control solution.

3.1 Implementation of Direct Multiple Shooting Method

Consider the following OCP for our pendulum problem,

$$\min J = \int_0^{t_N} \left(||\mathbf{x} - \mathbf{x}_{ref}||^2_{\mathbf{Q}} + ||u||^2_{\mathbf{R}} \right) dt$$

subject to :

$$\begin{aligned} &-40^o \leq x_1 \leq 40^o, -2 \leq u \leq 2 \\ &\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), u(t)) \\ &\mathbf{x}(0) = \mathbf{x}(t_i) \end{aligned} \tag{8}$$

where, $\mathbf{x}(t_i)$ is the measured state vector at the sampling instant t_i. We imposed the constraint for θ/x_1 and the input, leaving $x_2/\dot{\theta}$ unconstrained. The steps for implementing multiple shooting method are delineated below:

- The first step of multiple shooting is to discretize the time horizon $[0, t_N]$ into N equally spaced intervals of length $dT = t_N/N$. Therefore we have discrete time instants of $t_1, t_2, \ldots, t_k$ spanning the time horizon $[0, t_N]$ $(t_1 = 0, t_N = t_k)$. We parameterize the control for each subinterval $[t_k, t_{k+1}]$ as a piecewise constant value,

$$u(t) = \tilde{u}_k \text{ for } t \in [t_k, t_{k+1}] \ , \ k = 1, \ldots, N-1 \tag{9}$$

- Then we parameterize the initial state of each subinterval as,

$$\mathbf{x}(t_k) = \tilde{\mathbf{x}}_k \ , \ k = 1, \ldots, N \tag{10}$$

  We impose the following equality constraint for $\tilde{\mathbf{x}}_1$ such that it is equal to the measured system response $\mathbf{x}(t_i)$ at the sampling instant of the continuous-time process or the pendulum system.

$$\tilde{\mathbf{x}}_1 - \mathbf{x}(t_i) = 0 \tag{11}$$

- The state trajectories are evaluated in each subinterval $[t_k, t_{k+1}]$ using numerical integration methods as show in (12). For this project we have used the 4th order Runge-Kutta discrete integration.

$$\mathbf{x}_{k+1} \approx \int_{t_k}^{t_{k+1}} \mathbf{f}(\mathbf{x}(t), u(t))dt \ , \ k = 1, \ldots, N-1 \tag{12}$$

- We force the state trajectories to be continuous by imposing the following equality constraints,

$$\mathbf{x}_{k+1} - \tilde{\mathbf{x}}_k = 0 \ , \ k = 1, \ldots, N-1 \tag{13}$$

  A graphical illustration of the continuity constraints is shown in Figure 7.

- For each sub interval k, we compute the objective function in similar manner as that of the state trajectories using numerical integration,

$$J_k \approx \int_{t_k}^{t_{k+1}} J(\mathbf{x}(t), \tilde{u}_k)dt \approx \Big(\sum_{i=k}^{k+1} ||\tilde{\mathbf{x}}_i - \mathbf{x}_{ref}||^2_{\mathbf{Q}}\Big) + ||\tilde{u}_i||^2_{\mathbf{R}} \tag{14}$$

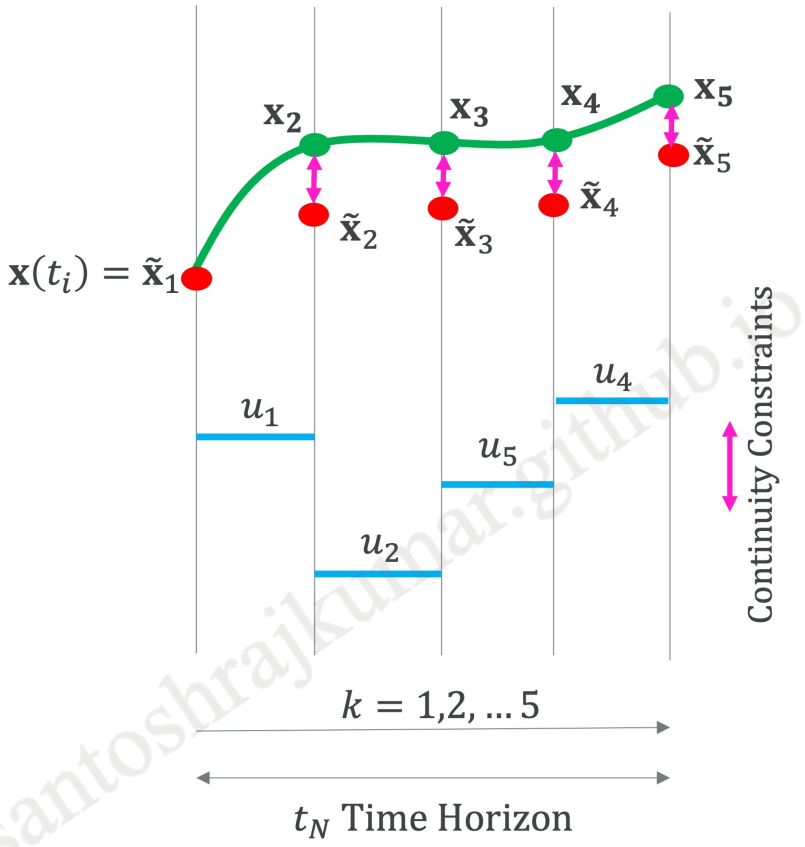

Figure 7: Pictorial description of continuity constraints in multiple shooting method

- Therefore the objective function for the entire time horizon (as in (8)) becomes sum of the objective function for each time interval, equivalently represented as,

$$J \approx \sum_{k=1}^{N} J_k \approx \sum_{k=1}^{N-1} \int_{t_k}^{t_{k+1}} J(\mathbf{x}(t), \tilde{u}_k)dt \approx \sum_{i=1}^{N-1} ||\tilde{\mathbf{x}}_i - \mathbf{x}_{ref}||^2_{\mathbf{Q}} + ||\tilde{u}_i||^2_{\mathbf{R}} \tag{15}$$

- Considering the parameter vector matrix as, $\mathbf{w} = \begin{bmatrix} \tilde{\mathbf{x}}_1(:) & \tilde{\mathbf{x}}_2(:) & \ldots & \tilde{\mathbf{x}}_N(:) & \tilde{u}_1, & \tilde{u}_2 & \ldots \tilde{u}_{N-1} \end{bmatrix}^T$, we convert the OCP objective function to a parameterized NLP objective function as,

$$\Phi(\mathbf{w}) = \Phi(\tilde{\mathbf{x}}_i, \tilde{u}_i) = \sum_{i=1}^{N-1} ||\tilde{\mathbf{x}}_i - \mathbf{x}_{ref}||^2_{\mathbf{Q}} + ||\tilde{u}_i||^2_{\mathbf{R}} \tag{16}$$

- We arrive at the following NLP problem from the OCP problem for our pendulum problem in (8) as,

$$\begin{aligned} &\arg\min_{\mathbf{w}} \ \Phi(\mathbf{w}) \\ &\text{subject to :} \\ &\mathbf{H} = 0 \\ &\mathbf{G1} \leq 0 \\ &\mathbf{G2} \leq 0 \end{aligned} \tag{17}$$

where,

$$\mathbf{H} = \begin{bmatrix} \tilde{\mathbf{x}}_1 - \mathbf{x}(t_i) \\ \mathbf{x}_2 - \tilde{\mathbf{x}}_1 \\ \mathbf{x}_3 - \tilde{\mathbf{x}}_2 \\ \vdots \\ \mathbf{x}_N - \tilde{\mathbf{x}}_{N-1} \end{bmatrix}$$

$$\mathbf{G1} = \begin{bmatrix} (\tilde{x}_1)_1 - 40^o \\ (\tilde{x}_1)_2 - 40^o \\ \vdots \\ (\tilde{x}_1)_N - 40^o \\ \tilde{u}_1 - 2 \\ \tilde{u}_2 - 2 \\ \vdots \\ \tilde{u}_{N-1} - 2 \end{bmatrix}$$

$$\mathbf{G1} = \begin{bmatrix} (\tilde{x}_1)_1 + 40^o \\ (\tilde{x}_1)_2 + 40^o \\ \vdots \\ (\tilde{x}_1)_N + 40^o \\ \tilde{u}_1 + 2 \\ \tilde{u}_2 + 2 \\ \vdots \\ \tilde{u}_{N-1} + 2 \end{bmatrix}$$

3.2 Solving the NLP and Implementing NMPC

We have converted the OCP problem in (8) to an NLP in (17) for the nonlinear pendulum system under consideration. In MATLAB, we can conveniently do the conversion from OCP to NLP using CasADi. To solve the NLP derived from multiple shooting in CasADi within MATLAB using the IPOPT solver, CasADi offers a convenient plugin specifically designed for interfacing with IPOPT. This plugin seamlessly integrates CasADi's symbolic framework and optimization capabilities with MATLAB, allowing users to leverage CasADi's powerful features directly within the MATLAB environment. By utilizing the CasADi-IPOPT plugin, MATLAB users can easily set up and solve complex NLPs obtained from multiple shooting, benefiting from IPOPT's efficiency and robustness in handling large-scale nonlinear optimization problems.

Now we delineate the NMPC scheme with the NLP problem defined and loaded. Consider, we want to stabilize the inverted pendulum system around $\theta = 0^o$ in a time span $[0, t_f]$. Consider that we have system states available at any instant $t_i \in [0, t_f]$ with a sampling interval of dt. The NMPC scheme is implemented in the following ways:

- We obtain the initial states $\mathbf{x}_0$ at $t_0 = 0$. Then we initialize the decision variables for

the NLP problem as,

$$\mathbf{w}_{t_0} = \begin{bmatrix} \tilde{\mathbf{x}}_1(:) = \mathbf{x}_0(:) & \dots \tilde{\mathbf{x}}_N(:) = \mathbf{x}_0(:) & \tilde{u}_1 = 0 & \dots & \tilde{u}_{N-1} = 0 \end{bmatrix}^T$$

- We solve the NLP with $\mathbf{w}_{t_0}$ and $\mathbf{x}_0$ to obtain,

$$\mathbf{w}^*_{t_0} = \begin{bmatrix} \tilde{\mathbf{x}}^*_1(:) & \dots \tilde{\mathbf{x}}^*_N(:) & \tilde{u}^*_1 & \dots & \tilde{u}^*_{N-1} \end{bmatrix}^T$$

  The optimal control input $u^*(t_0) = \tilde{u}^*_1$ is applied to the pendulum system with zero-order hold (ZOH) to obtain the measured output at the instant t_1 as $\mathbf{x}(t_1)$ and shift $\mathbf{w}^*_{t_0}$ one step in time as,

$$\hat{\mathbf{w}}^*_{t_0} = \begin{bmatrix} \tilde{\mathbf{x}}^*_2(:) & \dots \tilde{\mathbf{x}}^*_N(:) & \tilde{\mathbf{x}}^*_N(:) & \tilde{u}^*_2 & \dots & \tilde{u}^*_{N-1} & \tilde{u}^*_{N-1} \end{bmatrix}^T$$

- At the sampling instant t_1, we initialize the decision variables as $\hat{\mathbf{w}}^*_{t_0}$ and use $\mathbf{x}(t_1)$ (the measured output from the system) to solve the NLP problem again to obtain $\mathbf{w}^*_{t_1}$ and apply $u^*(t_1) = \tilde{u}^*_1$ to the pendulum system.

- Continue solving one such NLP problem at each sampling instant till we reach final time t_f.

4 Results

We tested the delineated NMPC scheme for balancing the inverted pendulum system in simulation to stabilize it aroud $\theta = 0^0$ in MATLAB (version 2023a) with CasADi (version 3.6.5). The simulations are carried out on an Apple MacBook Pro with 2.6 GHz 6-core Intel Core i7 processor. The sampling interval dt is chosen as 0.05 sec (at 20 Hz). For the MPC scheme, the time horizon t_N is selected as 0.5 sec equally spaced at an interval of $dT = 0.1$ sec. The $\mathbf{Q}$ and $\mathbf{R}$ for the objective function of the OCP/NLP are chosen as, $\mathbf{Q} = 1000 * \mathbf{I}_2$ and $\mathbf{R} = 0.1$. The pendulum system is implemented using *ode*45 solver to symbolize a continuously running plant. For discrete integration in the OCP problem, we have used 4th-order Runge-Kutta method. The NMPC problem is constained for θ and u as discussed in the previous section.

To demonstrate the stabilization of the pendulum system, we considered two sets of initial conditions away from the unstable zero equilibrium. Figure 8 shows stabilization of the pendulum with initial conditions $\theta = 35^o, \dot{\theta} = -5.73^o/sec$ and Figure 9 shows the same for initial conditions $\theta = -20^o, \dot{\theta} = -5.73^o/sec$.

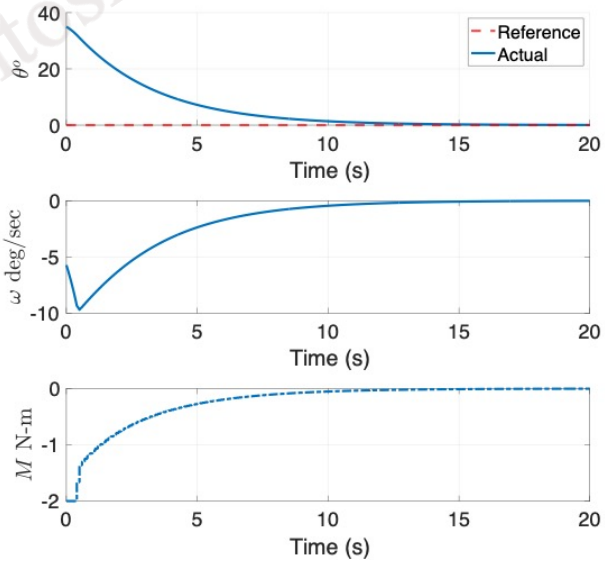

Figure 8: Stabilization of the pendulum with initial conditions $\theta = 35, \dot{\theta} = -5.73^o/sec$

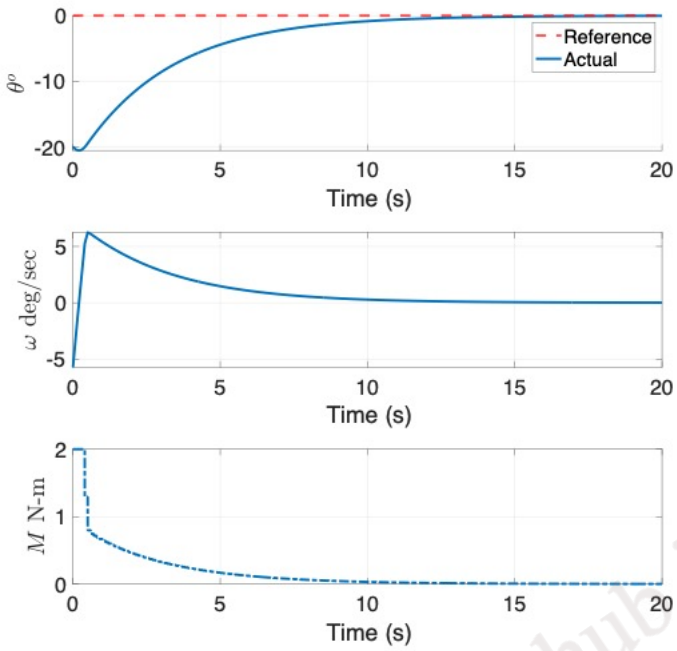

Figure 9: Stabilization of the pendulum with initial conditions $\theta = -20^o, \dot{\theta} = -5.73^o/sec$

To illustrate the effects of parameter uncertainty or mismatch, we introduced variations in the system parameters (mass m, damping coefficient c, and spring stiffness k) by increasing them by 20% of their actual values in the model utilized for Model Predictive Control (MPC). Figure 10 showcases the stabilization of the pendulum under these conditions, considering initial conditions $\theta = 35^o, \dot{\theta} = -5.73^o/sec$. Additionally, we examined a scenario involving noisy measurements with the parameter-mismatched model. To simulate this scenario, we corrupted the measured states with Gaussian noise, with a mean of 3. Figure 11 illustrates the stabilization of the pendulum in the presence of both parameter mismatch and noisy measurements, using the same initial conditions as before, i.e., $\theta = 35^o, \dot{\theta} = -5.73^o/sec$.

The NMPC scheme applied to the demonstrated scenarios is found to stabilize the non-linear inverted pendulum system as expected.

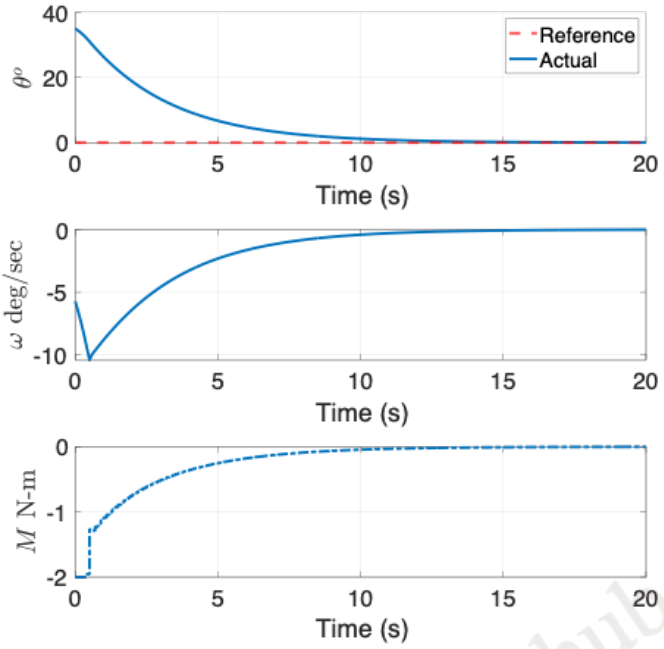

Figure 10: Stabilization of the pendulum with parameter mismatch considering conditions $\theta = 35^o, \dot{\theta} = -5.73^o/sec$

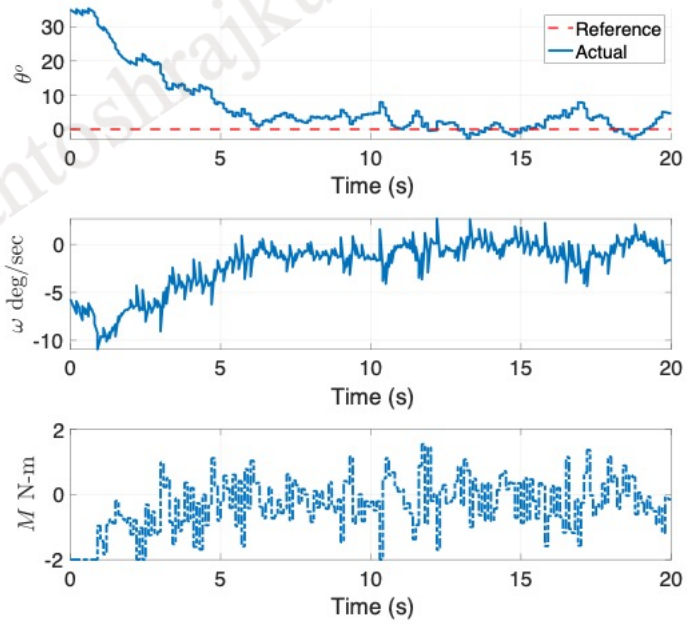

Figure 11: the stabilization of the pendulum in the presence of both parameter mismatch and noisy measurements with initial conditions $\theta = 35^o, \dot{\theta} = -5.73^o/sec$

4.1 Discussion

The NMPC scheme applied to the demonstrated scenarios effectively stabilized the nonlinear inverted pendulum system, achieving the expected outcome. In all considered scenarios, NMPC successfully stabilized the system around the zero equilibrium position, even when starting from initial conditions significantly distant from the equilibrium. Moreover, the constraints imposed on the control input and the pendulum angle (θ) were consistently satisfied across all scenarios. This demonstrates the robustness and effectiveness of the NMPC approach in controlling the inverted pendulum system under various conditions, including parameter uncertainty and noisy measurements. In the presence of a 20% variation in the system parameters, the Nonlinear Model Predictive Control (NMPC) scheme demonstrated its ability to effectively stabilize the system while adhering to both control and state constraints. Despite the significant perturbation introduced by the parameter mismatch, the NMPC controller successfully maintained stability without exhibiting any undesired transient behavior such as overshoot or undershoot. This indicates the robustness of the NMPC approach in handling parameter uncertainties and its capability to adaptively adjust the control inputs to compensate for variations in the system parameters. Furthermore, the ability of the NMPC scheme to maintain constraint satisfaction underscores its effectiveness in ensuring the safe and reliable operation of the inverted pendulum system under challenging conditions. Overall, these results highlight the efficacy and reliability of NMPC in controlling nonlinear systems subjected to parameter uncertainties. Further, in the presence of both a 20% variation in the system parameters and Gaussian noise in the measurements, we observed small oscillations in the steady-state motion around the zero equilibrium position. This behavior was anticipated due to the corrupt measurements introduced by the Gaussian noise, which inherently introduced uncertainty into the system feedback. Despite these oscillations, the Nonlinear Model Predictive Control (NMPC) scheme successfully maintained stability and satisfied all imposed constraints. The fact that

the system remained within acceptable bounds despite the presence of measurement noise highlights the robustness and effectiveness of the NMPC controller. While small oscillations were observed, the overall control performance was reasonable, given the challenging conditions imposed by parameter variations and measurement uncertainties. Thus, despite the imperfect measurement conditions, the NMPC scheme demonstrated its capability to provide reliable and effective control of the inverted pendulum system. These results highlight how effective and useful NMPC can be in real-world scenarios where uncertainties in the system are unavoidable. It demonstrates NMPC's potential to tackle intricate control problems across various domains, showcasing its versatility and applicability.

One drawback often associated with MPC is its computational expense. This arises from the need to solve optimization problems repeatedly within a short time frame to generate control actions. To study the computational time taken by the NMPC scheme for the said scenarios above, we summarize the average computational time taken for solving the NLP problem in Table 2. We observe that at each time step, the NLP was solved,on average, in a time less than the sampling interval $dt = 0.05sec$. Therefore, this NMPC scheme can be implemented in a real-time setting for the said system with the sampling interval $dt = 0.05sec$ and with a time horizon of $0.5sec$. How the system behavior changes with coarser division of the MPC horizon and/or shorter MPC horizon, can be an interesting future scope of this project.

Table 2: The average computational time taken for solving the NLP problem in NMPC

Scenario	Avg. Computational Time
Figure 8	0.0364 sec
Figure 9	0.0371 sec
Figure 10	0.0414 sec
Figure 11	0.0436 sec

4.2 MATLAB Code

** The MATLAB code for the implementation can be found Here.

5 Conclusions

In conclusion, this project has explored the implementation of Nonlinear Model Predictive Control (NMPC) using CasADi within the MATLAB environment. The project provided a comprehensive understanding of NMPC, covering theoretical foundations such as MPC, nonlinear programming, and transcription methods. Through a demonstration using a nonlinear system featuring an inverted pendulum with a spring-damper setup, the effectiveness of NMPC across various initial conditions was showcased. Additionally, the project investigated the robustness of MPC in handling parameter uncertainties and noise within the pendulum system.

References

[1] Boliang Yi. *Integrated Planning and Control for Collision Avoidance Systems*, volume 39. KIT Scientific Publishing, 2018.

[2] G Campagne et al. A nonlinear model predictive control based evasive manoeuvre assist function. *TU Delft masters student theses repository*, 2019.

[3] Giovanni Lavezzi, Kidus Guye, Venanzio Cichella, and Marco Ciarcià. Comparative analysis of nonlinear programming solvers: Performance evaluation, benchmarking, and multi-uav optimal path planning. *Drones*, 7(8):487, 2023.

[4] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 2006.

[5] Viktor Leek. An optimal control toolbox for matlab based on casadi, 2016.

[6] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.