Trajectory Tracking Nonlinear Model Predictive Control for Stair Climbing of a Planar Five Link Walker

Grant A. Gibson and M. Eva Mungai

I. INTRODUCTION

Our lab, the Biped Robotics Laboratory, has primarily focused on dynamic, periodic walking for the past several years. Recently, however, in order to improve the robustness of our walking controllers in response to unknown and rough terrain, we have integrated vision into our system (LiDARS and RGB Cameras). The main advantage of perception is that it allows us to preview the terrain thus making autonomous walking possible, especially in adverse environments such as stair climbing.

The next challenge is then to develop a robust controller that can dynamically adapt to various stair heights. We decided to use Nonlinear Model Predictive Control (NMPC). This choice was based on the fact that the dynamics of our system are highly nonlinear. Furthermore, rather than use Cassie, which has 16 degrees of freedom (DoF), we decide to use the planar five link walker (7 DoF) to simplify the problem. This approach has been shown to scale up to more complex robots [1]. As a design choice, we decide to use a trajectory tracking NMPC (TT-NMPC) formulation because it allows us to impose state constraints, and incorporate new information into the controller in real time. We simplify this formulation to also include results for regulator NMPC (R-NMPC). The trajectories are generated using the Fast Robotics Optimization and Simulation Toolbox (FROST) [2]. The minimization of the TT-NMPC nonlinear program is solved using the open-source packages, CasADi and IPOPT [3] [4].

To validate our controller, we tested our NMPC formulation on the well known cart-pole system. We then show our TT-NMPC and R-NMPC results for the five link walker. In addition we analyze the stability and robustness of our controller to varied initial conditions and uncertainties. Lastly, we re-formulate the TT-NMPC problem to include disturbance preview for change in stair height, mid step.

II. NMPC Trajectory Tracking and Regulator Stabilization Formulation

A. Trajectory Tracking

We first describe the nonlinear program used to compute the TT-NMPC controller. The general nonlinear program is shown in Equation (1).

$$\min_{\mathbf{Z}} J_N = \sum_{k=0}^{N-1} [(x_k - \hat{x}_k)^T Q (x_k - \hat{x}_k) + (u_k - \hat{u}_k)^T R (u_k - \hat{u}_k)] + (x_N - \hat{x}_N)^T Q_{term} (x_N - \hat{x}_N)$$

$$\begin{aligned} \textbf{subject to} \quad & x_{k+1} = x_k + \Delta T f(x_k, u_k), \quad (k = 0, \cdots, N-1) \\ & x_0 = \text{current state} \in \mathbf{X} \\ & u_k \in \mathbf{U}, \quad (k = 0, \cdots, N-1) \\ & x_k \in \mathbf{X}, \quad (k = 0, \cdots, N-1) \\ & x_N \in \mathbf{X}_f \subset \mathbf{X} \end{aligned} \tag{1}$$

Where $\mathbf{Z} = \{x_0, \cdots, x_N, u_0, \cdots, u_{N-1}\}$ is the minimized decision variable set. x_i are the states and u_i are the control inputs over the entire prediction horizon, N. $Q \succcurlyeq 0$ is the running stage penalty, $R \succ 0$ is the control input penalty, and $Q_{term} \succcurlyeq 0$. $\hat{\mathbf{X}} = \{\hat{x}_0, \cdots, \hat{x}_N, \hat{u}_0, \cdots, \hat{u}_{N-1}\}$ is the reference trajectory set defined prior to optimization.

We use a multi-shooting approach to increase sparsity and improve efficiency of the optimization solver. Next, we reformulate the problem into a form suitable for numerical minimization solvers (in our case we used IPOPT) [4]. Here we group all of the dynamics equations into a single equality constraint ($G = 0$) and we define a convex polyhedron to constrain the decision variables. The constraints from Equation (1) are now updated using the same cost (2).

$$\min_{Z} J_N = \sum_{k=0}^{N-1} [(x_k - \hat{x}_k)^T Q (x_k - \hat{x}_k) + (u_k - \hat{u}_k)^T R (u_k - \hat{u}_k)] + (x_N - \hat{x}_N)^T Q_{term} (x_N - \hat{x}_N)$$

$$\begin{aligned} \textbf{subject to} \quad & G = \begin{bmatrix} x_0 - \hat{x_0} \\ x_1 - (x_0 + \Delta T f(x_0, u_0)) \\ \vdots \\ x_N - (x_{N-1} + \Delta T f(x_{N-1}, u_{N-1})) \end{bmatrix} = \mathbf{0} \\ & \begin{bmatrix} I \\ -I \end{bmatrix} Z \leq \begin{bmatrix} Z_{max} \\ -Z_{min} \end{bmatrix} \end{aligned} \tag{2}$$

where $Z_{max} = [x_0^{max}, \cdots, x_N^{max}, u_0^{max}, \cdots, u_{N-1}^{max}]^T$ and $Z_{min} = [x_0^{min}, \cdots, x_N^{min}, u_0^{min}, \cdots, u_{N-1}^{min}]^T$.

Notice that the problem is very similar to the constrained Linear Quadratic Model Predictive Control (LQ-MPC) formulation [5]. The only difference is that the nonlinear dy-

namics are included in the equality constraint. Discretization of the system is achieved using the forward Euler method with a step size of ΔT. These similarities are used later in the paper to simplify the stability analysis of the five-link NMPC controller.

B. Regulator Stabilization

When formulating the nonlinear program for computing the R-NMPC controller we only need to make a small modification to the formulation. The reference trajectory set is changed to only include the initial condition and the desired set-point states ($\hat{x}^s$) and control inputs ($\hat{u}^s$) (more explicitly, $\hat{\mathbf{X}} = \{\hat{x}_0, \hat{x}^s, \hat{u}^s\}$). The program is described below in Equation (3).

$$\min_{\mathbf{Z}} J_N = \sum_{k=0}^{N-1} [(x_k - \hat{x}^s)^T Q (x_k - \hat{x}^s) + (u_k - \hat{u}^s)^T R (u_k - \hat{u}^s)] + (x_N - \hat{x}^s)^T Q_{term} (x_N - \hat{x}^s)$$

$$\textbf{subject to} \qquad G = 0$$

$$\begin{bmatrix} I \\ -I \end{bmatrix} Z \leq \begin{bmatrix} Z_{max} \\ -Z_{min} \end{bmatrix} \tag{3}$$

III. Cart-Pole Preliminary

Prior to analyzing the five link walker we tested our regulator and trajectory tracking formulations on a simpler, cart-pole model (Figure 1). The basic results are presented and we did not investigate the system further since it is outside the scope of this project.

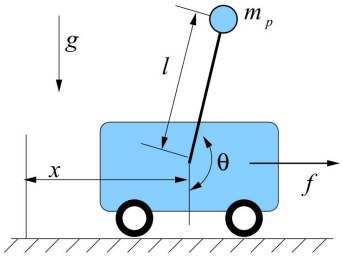

Fig. 1: Cart-Pole System. [6]

A. Dynamics

We consider the Euler-Lagrange formulation for representing the nonlinear dynamics of this system. Using standard-form generalized coordinates we define $q_{cp} = [x, \theta]^T, u_{cp} = f$, and state, $x_{cp} = [q_{cp}, \dot{q}_{cp}]^T$. The equations of motion for the system are shown below in Equation (4).

$$D_{cp}(q_{cp})\ddot{q_{cp}} + C_{cp}(q_{cp}, \dot{q}_{cp})\dot{q}_{cp} + G_{cp}(q_{cp}) = B_{cp} u_{cp} \tag{4}$$

where,

$$D_{cp} = \begin{bmatrix} m_c + m_p & m_p l \cos(\theta) \\ m_p l \cos(\theta) & m_p l^2 \end{bmatrix}, \quad C_{cp} = \begin{bmatrix} 0 & -m_p l \dot{\theta} \sin(\theta) \end{bmatrix}$$

$$G_{cp} = \begin{bmatrix} 0 \\ m_p g l \sin(\theta) \end{bmatrix}, \quad B_{cp} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Here, D_{cp}, C_{cp}, G_{cp}, and B_{cp} represent the mass-inertia, Coriolis, gravity, and input matrices, respectively. We leave out the arguments of these matrices from now on for simplicity. From this, we arrive at the nonlinear state-space form to be used in the Euler discretization of the NMPC formulations described in Section II.

$$\dot{x}_{cp} = f(x_{cp}, u_{cp}) = \begin{bmatrix} \dot{q}_{cp} \\ D_{cp}^{-1}[B_{cp} u_{cp} - C_{cp}\dot{q}_{cp} - G_{cp}] \end{bmatrix} \tag{5}$$

B. Regulator Results

Table I contains the model and optimization parameters used for computing the trajectories of the R-NMPC and TT-NMPC cartpole examples.

TABLE I: System and NMPC Parameters

$\mathbf{m_c}$	$\mathbf{m_p}$	$\mathbf{l}$	$\mathbf{g}$
1 kg	1 kg	0.5 m	9.81 m/s
$\mathbf{\Delta T}$	$\mathbf{N}$	$\mathbf{Q = Q_{term}}$	$\mathbf{R}$
0.005 sec	200	$\begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{bmatrix}$	0.1

Two example R-NMPC trajectories are shown in Figure 2. The top example uses an initial condition $x_0 = [0, \pi, 0, 0]^T$ with a set point condition of $([\hat{x}^s, \hat{u}^s) = ([2, \pi, 0, 0]^T, 0)$, while the bottom example has $x_0 = [0.5, \pi + 0.4, 0.2, -0.1]^T$ and $\hat{x}^s = [0, pi, 0, 0]^T$. The R-NMPC controller successfully stabilizes the states and input about the desired set-points for both examples. We set $Z_{max} = [x_{max}, u_{max}] = [\infty, \frac{5\pi}{4}, 2, 0.5, 10]^T$ and $Z_{min} = [x_{min}, u_{min}] = [-\infty, \frac{3\pi}{4}, -2, -0.5, -10]^T$. We confirm that the constraints are satisfied by analyzing the plots for saturation (which happens for $\dot{x}, \dot{\theta}$ and u).

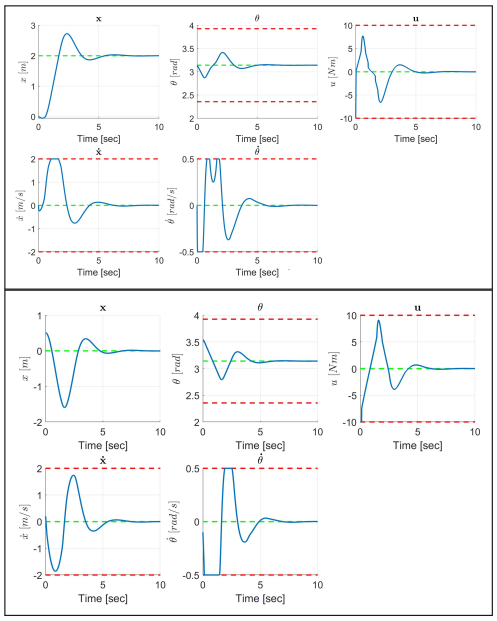

Fig. 2: Regulator NMPC for cart-pole system. The solution trajectories (blue) versus time stabilize to the desired trajectories (green). The constraints for each state and control input are also shown (red).

C. Trajectory Tracking Results

The two solution trajectories from the previous subsection were used as reference trajectories for verifying the TT-NMPC formulation. The same parameters in Table I were used as well. For each trajectory the initial condition was perturbed and the constraints were held the same. Result trajectories are shown in Figure 3.

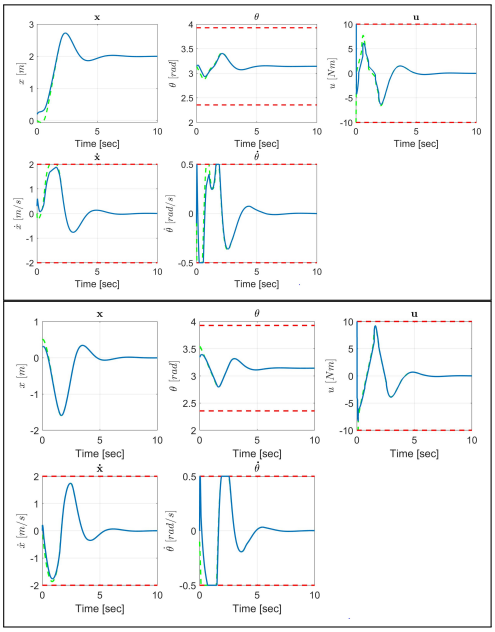

Fig. 3: Trajectory Tracking NMPC for cart-pole. The reference trajectories (green) are tracked by the system (blue) while satisfying constraints (red).

IV. Five Link Walker

A. Dynamics

After implementing NMPC on the cart-pole, and analyzing the results, the next step was to implement NMPC on the five link walker. The model, depicted in Figure 4, is planar and consists of five links and four actuated joints.

1) Right Hip : q_{1R}
2) Left Hip : q_{1L}
3) Right Knee : q_{2R}
4) Left Knee : q_{2L}

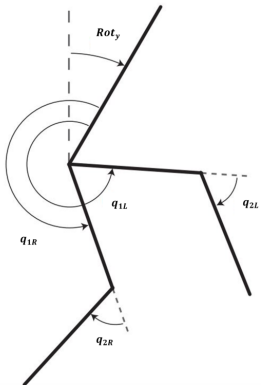

Fig. 4: Five Link Walker Model. [7]

The dynamics model of the robot is generated using the floating base Euler-Lagrange formulation. This floating base formulation is similar to that of the cart-pole model, except that the ground reaction forces (GRF) appear explicitly in the formulation. The dynamics equation for the five link walker are shown below

$$D(q)\ddot{q}+C(q,\dot{q})\dot{q}+G(q)=Bu+J^T\lambda \tag{6}$$

where

$$q=\begin{bmatrix} torsoPosX\ (x) \\ torsoPosZ\ (z) \\ torsoYaw\ (Rot_y) \\ q_{1R} \\ q_{2R} \\ q_{1L} \\ q_{2L} \end{bmatrix} \tag{7}$$

is the generalized state vector, D is the inertia matrix, C is the Coriolis matrix, G is the gravity vector,u is the vector of the joint torques, λ is the GRF vector, and B and J^T are the jacobian that map the u and λ to the generalized coordinates. Due to the fact that the equation of motion is under-determined, it is necessary to define an additional equation. Since we assume the stance foot is stationary during each step, the additional equation constrains the acceleration of the stance foot to equal zero. The stance foot acceleration constraint is shown in Equation (8). The equations of motion for our system during the continuous domain then consist of Equation (6) and (8).

$$J(q))\ddot{q}+\dot{J}(q,\dot{q})\dot{q}=0 \tag{8}$$

B. Nonlinear Trajectory Generation

The reference trajectory was generated using the Fast Robotics Optimization and Simulation Toolbox (FROST) which uses direct collocation and IPOPT [2]. As the five link walker is planar, there is no difference between right stance and left stance, therefore, the hybrid system model used in the trajectory optimization only consists of the single stance domain. An instantaneous discrete double support transition takes the system back to the single stance domain after impact. The transition from the single stance domain to the discrete domain is triggered when the height of the swing foot is equal to the stair height ($h^z_{sw}=stairheight$). A reset map, RM, is then used to redefine the previous swing foot as the stance foot and the previous stance foot as the swing foot. As a design choice, our hybrid system model starts with the right stance, therefore the RM defined below redefines the left stance as the right stance after impact.

$$RM=\begin{bmatrix} 1&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0 \\ 0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&1 \\ 0&0&0&1&0&0&0 \\ 0&0&0&0&1&0&0 \end{bmatrix} \tag{9}$$

The dynamics for the continuous domain are described using the floating base Euler-Lagrange and stance foot acceleration constraint equations. Since during impact these dynamics are no longer valid, Equations (10) and (11) are the equations of motions used to determine the robot's states after impact.

$$D(q^+)\dot{q}^+ - D(q^{-1})\dot{q}^- = J_{sw}^T(q^-)\lambda_{sw} \tag{10}$$

$$J_{sw}^T(q^+)\dot{q}^+ = 0 \tag{11}$$

where q^+ and q^- are the states of the robot after and before impact, λ_{sw} are the forces of the swing foot during impact, and J_{sw}^T is the jacobian that maps, λ_{sw} to the generalized coordinates.

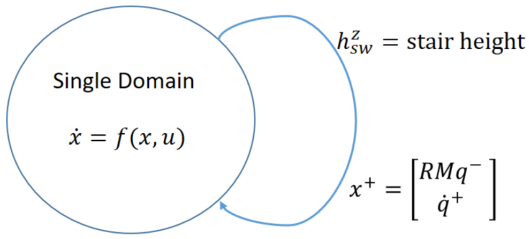

Fig. 5: Five Link Walker Hybrid System.

The cost function for the trajectory optimization of each time discretization (node) is defined as the norm of the joint torques. The constraint function consists of four virtual constraints, swing foot clearance, output constraints for the actuated joints($\begin{bmatrix} q_{1R} & q_{2R} & q_{1L} & q_{2L} \end{bmatrix}^T$), forward average velocity, and the height of the swing foot when impact occurs. There are four virtual constraints because the holonomic constraint, which ensures zero acceleration of stance foot, restricts 3 of the five link walker's 7 DoF. These virtual constraints, defined in Equation (12), capture the difference between the desired ($y_d(\alpha,\tau)$) and actual values ($y_a(q)$) of the actuated joints. The desired parameters of the virtual constraints are defined using the bezier polynomials and depend on bezier coefficients (α) and a monotonic phase variable (τ).

$$y = y_a(q) - y_d(\alpha,\tau) \tag{12}$$

The objective of the trajectory optimization is then to find the optimal gait ($x = \begin{bmatrix} q & \dot{q} \end{bmatrix}^T$ and u) and bezier coefficients that satisfy the equations of motions such that the cost function is minimized and the constraints are respected for each domain.

C. Consideration of the Hybrid System for MPC

To simplify the problem, only the continuous domain of the hybrid system described in Section IV-B is utilized for our MPC problem formulation. However, we suggest the same formulation utilized for disturbance preview in Section VIII with slight modifications can be used to implement the hybrid system model in MPC.

V. Implementation & Results

We implemented and tested our TT-NMPC controller of the five link walker in MATLAB for both ascending and descending stair height trajectories. CasADi was used to structure the nonlinear program and compute symbolic derivatives for our forward Euler integration scheme. The interior point based optimization method, IPOPT, was then used to solve the optimization problem at each time step ($\Delta T = 0.005s$) and set our NMPC feedback controller equal to the first control input in the decision variable solution set (Equation (13)).

$$u_{NMPC} = u_0^* \in Z^* \tag{13}$$

where u_0^* is the first control input belonging to Z^*, the optimal decision variable set obtained from minimization of the nonlinear program.

A. Shrinking Prediction Horizon

An additional, shrinking prediction horizon, modification was made because we use finite-time trajectories. Since our system is hybrid, an impact is expected at the end of each step and would require an alteration to the dynamics. For this report we consider only the continuous time domain of the hybrid system and thus, require the prediction horizon to decrease as it approaches the end of each trajectory (see Figure 6

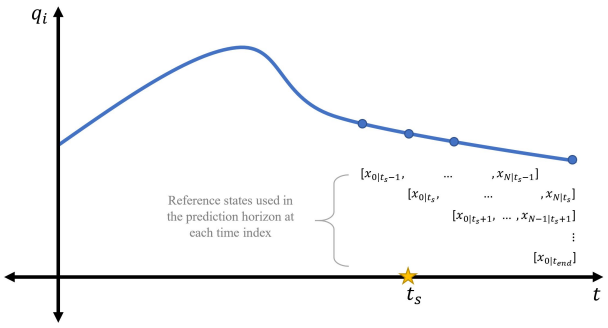

Fig. 6: Shrinking Horizon Schematic. t_s represents the shrinking horizon time. All subsequent prediction horizons will be shortened to make sure that the last state in the prediction horizon is the final reference state.

B. Trajectory Tracking Results

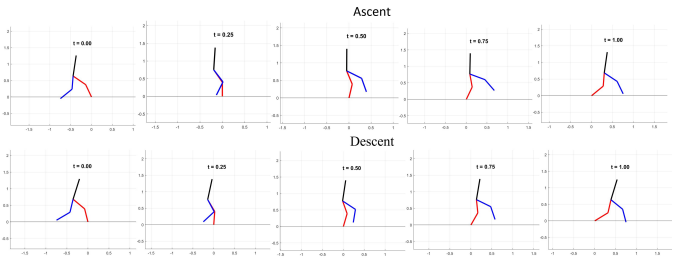

Fig. 7: Example stair climbing trajectories for $0.05m$ ascent (top) and descent (bottom). The stance foot is colored in red and remains on the ground for the entire step.

For brevity, we plot the TT-NMPC simulation results for a single trajectory (Figure 8). Each TT-NMPC controller

was calculated using the same objective function and constraint parameters (Equation (14)). The penalty matrices were chosen as a design choice to equally penalize the tracking error of both the states and controls. We tried various combinations of penalty matrices and the ones presented gave were satisfactory in comparison. Figure 7 illustrates how the five link walker is configured throughout time for two separate reference trajectories.

$$
\begin{aligned}
Q &= Q_{term} = \mathbf{I}_{14\times 14}, \qquad R = \mathbf{I}_{4\times 4}, \\
x_k^{max} &= \begin{bmatrix} 10 & 2 & 1.3 & 5.3 & 2.3 & 5.3 & 2.3 & 20 * ones(1,7) \end{bmatrix}^T, \\
u_k^{max} &= \begin{bmatrix} 10 & 10 & 10 & 10 \end{bmatrix}^T, \\
x_k^{min} &= -x_k^{max}, \qquad u_k^{min} = -u_k^{max}
\end{aligned} \tag{14}
$$

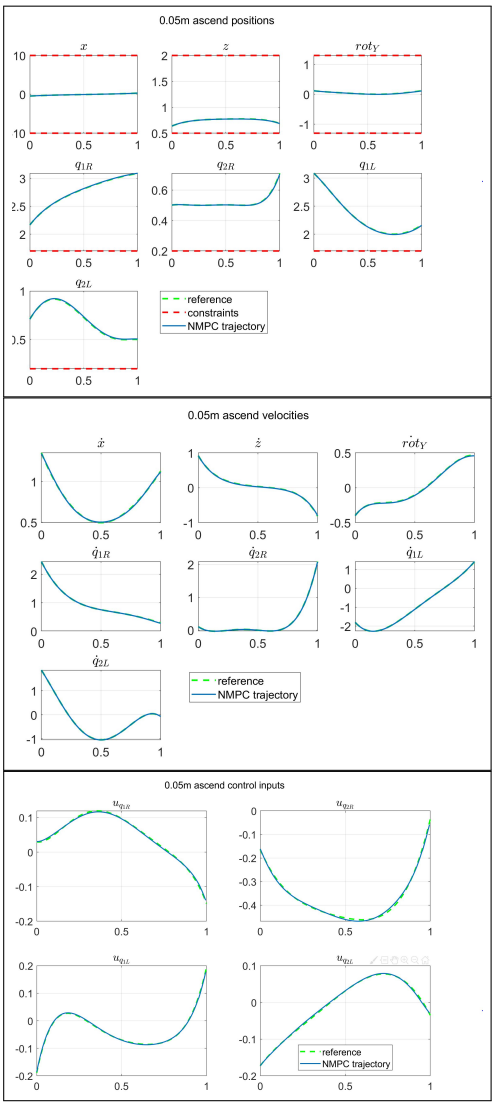

Fig. 8: Position, velocity, and control input trajectories for the TT-NMPC 0.05m ascent controller.

Lastly, we include a comprehensive list of our results for all tested trajectories (Table II).

While we only used a prediction horizon of 10 for the results presented in this paper, we briefly explored how increasing this horizon could improve results. We include the 2-norm position and velocity error in addition to the end swing foot position of a single TT-NMPC trajectory. Based on this data, we hypothesize that increasing the prediction horizon and modifying the cost function could reduce tracking error further.

TABLE II: TT-NMPC Position and Velocity Error for Ascending/Descending Stairs.

Stair Height	Ascend (↑) Descend (↓)	Position Error ($\|\cdot\|_2$)	Velocity Error ($\|\cdot\|_2$)	Swing Foot Position (m)
0.04	↑	0.0125	0.157	0.0449
	↓	0.0113	0.218	-0.0318
0.05	↑	0.0125	0.159	0.0544
	↓	0.0113	0.248	-0.0403
0.06	↑	0.0128	0.162	0.0641
	↓	0.0111	0.2584	-0.0507
0.075	↑	0.0131	0.167	0.0787
	↓	0.0115	0.308	-0.0625
0.1	↑	0.0134	0.171	0.104
	↓	0.0119	0.301	-0.0888
0.15	↑	0.0152	0.205	0.1512
	↓	0.0137	0.377	-0.136
0.2	↑	0.0179	0.230	0.202
	↓	0.0159	0.412	-0.182

TABLE III: Position and velocity tracking error, and ending swing foot position for various prediction horizons using the 0.05m stair height reference trajectory

N	Position Error ($\|\cdot\|_2$)	Velocity Error ($\|\cdot\|_2$)	End Swing Foot Position (m)
10	0.0125	0.159	0.0544
30	0.0127	0.151	0.0541
50	0.0127	0.148	0.0541
100	0.0129	0.144	0.0538

C. Regulator Results

Using Equation (2), we reformulate the five link trajectory tracking problem into a regulator problem. The desired state is chosen by picking a point in the reference trajectory derived from FROST and setting the velocity to zero. The results, however, are undesirable as the R-NMPC isn't able to reach and maintain the desired state. An explanation for this instability result is discussed in the next section.r This phenomenon might be attributed to poor cost gains, and underactuation. The results can be found in Figure 9.

D. Stability Analysis

We utilize stability analysis techniques shown and proven in [8] in order to justify the stability analysis of our TT-NMPC and R-NMPC controllers. Based on the fact that f and the running and terminal cost are continuous, $\mathbf{X}$ and $\mathbf{X_f}$ closed where $\mathbf{X_f} \subset \mathbf{X}$, and $\mathbf{U}$ compact and uniformly bounded, Assumptions 2.25 and 2.26 are satisfied. Additionally, since we use a convex and quadratic cost function with positive definite penalty matrices, $J_N(\mathbf{0}) = 0$, and $f(0,0) = 0$, Assumptions 2.33 and 2.37 are met.

However, since we do not have a terminal cost function that ensures the trajectory doesn't leave the terminal state, we cannot guarantee stability. This observation, offers another explanation for the unstable R-NMPC results in Section V-C. As for the TT-NMPC controller, it makes sense to define

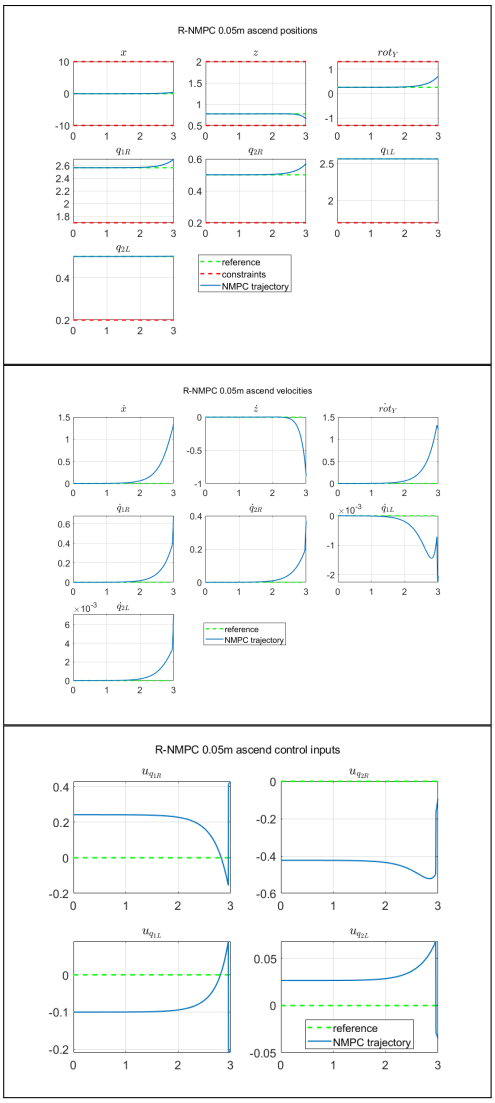

Fig. 9: R-NMPC results for five link walker.

the terminal set as a periodic trajectory. Even though the reference trajectory used is periodic, since our TT-NMPC formulation doesn't capture the hybrid system, it's not possible to talk about periodic stability. To guarantee global stability, we have to define a terminal constraint function as well as a terminal cost function for the states [9][10].

In more simplified terms, we require a terminal controller where $f(x_N, u_N^{NMPC}) = x_N$ at the end of each prediction horizon. For under-actuated systems it is predicted that increased prediction horizons could lead to improved stability results.

VI. Perturbation Analysis

In real world experiments it is impossible to have knowledge of the exact pose and governing dynamics of a robot. These machines are highly nonlinear and the hardware and sensors are susceptible to many types of inaccuracies (i.e. frictional, mechanical, and electrical losses). The simulations shown in Section V assume that the initial state of our robot and desired trajectory are the same, and we propagate these trajectories with known dynamics and control inputs. In this section we analyze the performance of the TT-NMPC controller in response to pose, model, and hardware uncertainty.

A. Perturbed Initial Condition

We first consider the case when the initial state of the walker is inaccurate. Due to the constrained nature of the dynamics (Section IV), we must be careful in how we perturb this condition. By arbitrarily choosing an initial condition we might not only violate the given constraints of the TT-NMPC formulation, but also violate the stricter constraints imposed by zero stance foot acceleration (Equation (8)). For this reason we make the initial condition equal to that of a different, known, trajectory. Figure 10 illustrates how we used initial conditions of the $0.04m$ and $0.06m$ ascent trajectories to simulate a desired $0.05m$ ascent step height. From inspection, the error, introduced by using a different initial conditions, seems to remain bounded throughout the trajectory. Although this error does not converge to that of the original TT-NMPC controller we believe that appropriately tuned penalty matrices could improve the results.

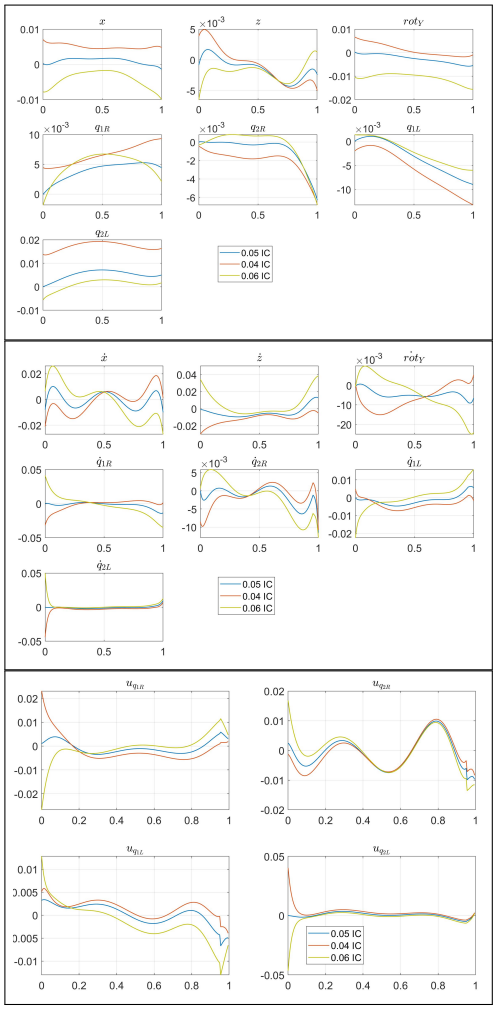

Fig. 10: Initial Condition Perturbation Error for Nominal $0.05m$ Ascent Trajectory.

B. Model Mismatch

Next we analyze the error trajectories for TT-NMPC controllers that simulate the dynamics with an incorrect mass-inertia matrix, D. Figure 11 shows the error between the simulated and reference trajectories for $D = D, 0.9$, and $1.1D$. We achieve similar results when modifying the input matrix, B, which represents an inaccuracy in the modeling of the motor transmission of the system.

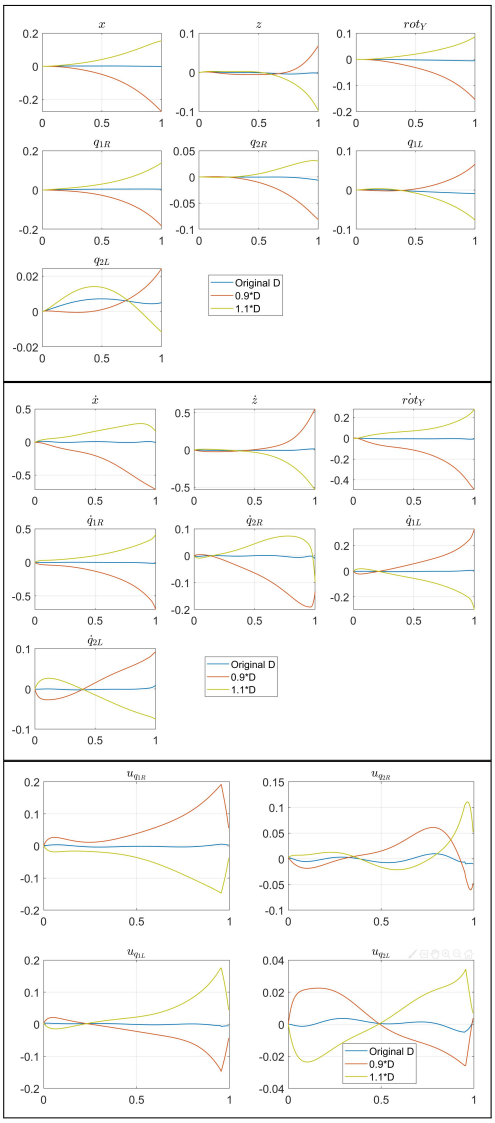

Fig. 11: Comparing error trajectories for $0.05m$ ascent while modifying the mass-inertia matrix.

C. Input Torque Noise

When commanding motor torques, the desired torque is inherently noisy to do mechanical and electrical uncertainties. We test the robustness of our controller in response to normally distributed torque inputs with standard deviations $\sigma = 0.1, 0.5\ N-m$. Additional improvements could made using a low pass filter with ammeter measurement readings for torque inputs large standard deviations.

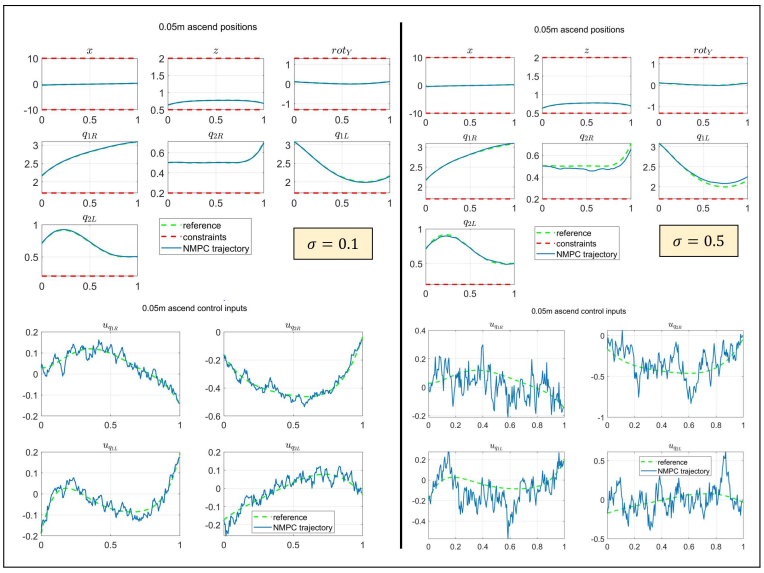

Fig. 12: Control Input Torque Noise. Results are shown for motor torques with white noise and a standard deviation of 0.1 (left) and 0.5 (right).

VII. Comparison to Input-Output Feedback Linearization Controller

To see how our TT-NMPC controller compares to the input-output feedback linearization controller (I/O controller), we run both controllers while saturating the control inputs at various values. The motivation for saturating the inputs comes from a situation one might counter with hardware, where the motors are not able to provide the expected torque. Instead of tracking the states and torques of the reference trajectory, the I/O controller tracks the desired virtual constraints from the reference trajectory. For more information on how the I/O controller was derived see [11] These virtual constraints are the same ones defined in the trajectory optimization. Therefore, when looking at the performance of the I/O controller, we will only look at the tracking error of the actuated states.

At first glance it may appear as if the I/O controller performs better than the TT-NMPC controller for most of the actuated joints with a torque saturation of $\pm 0.45Nm$. However, since the tracking error for both controllers is relatively small(10^{-3}), the errors of the two controllers are comparable, except for q_{2R} where the TT-MPC performs better than the I/O. The I/O controller trajectory terminates before the TT-MPC trajectory due to the detection of impact. These results can be observed in Figure 13. Further saturating the torque to $\pm 0.35Nm$ results in the TT-NMPC not finding a solution that respects the state constraints. This lack of solution could be addressed by using slack variables. As there is no way to impose state constraints in the I/O controller, the I/O controller finds a solution that violates the state bounds. This violation of state constraints by the I/O controller can be seen in Figure 14.

VIII. Disturbance Preview

As the motivation for this project is to have the five link walker autonomously climb up stairs using information from

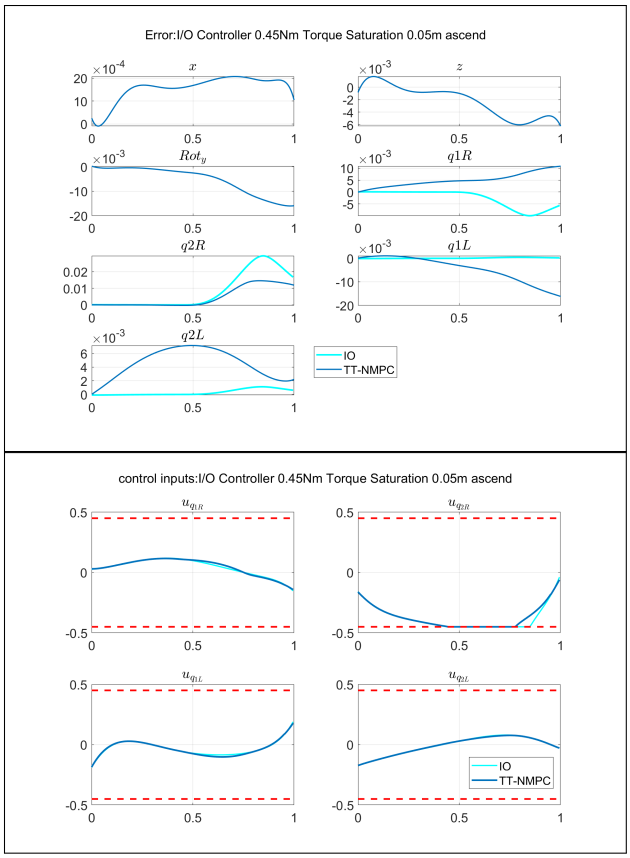

Fig. 13: I/O and TT-MPC comparison plots for a torque saturation of $\pm 0.45Nm$

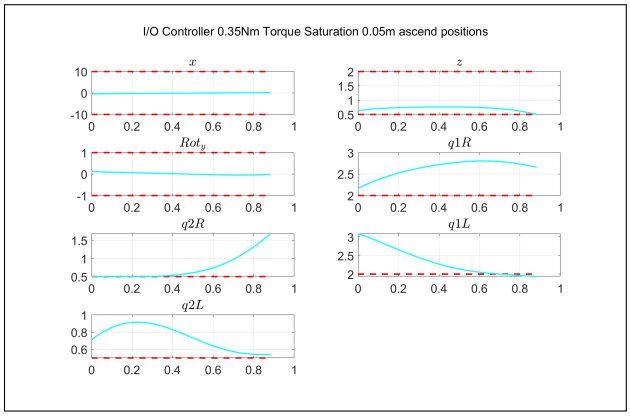

Fig. 14: state trajectories of the I/O controller for a torque saturation of $\pm 0.35Nm$

a vision system with MPC as the controller, it is necessary to find a way to incorporate new information about the environment such as the stair height and length into the MPC controller. Therefore, we look into how our MPC problem can be reformulated to capture a change in stair height mid step. In other words, our goal is to get our NMPC tracking controller to converge to the new stair height by tracking the correct reference trajectory. This turns out to be a disturbance preview problem, with the disturbance defined as

$$w_{t_d} = \tilde{x}_{t_d} - \hat{x}_{t_d} \tag{15}$$

where $\hat{x}_{t_d}$ and $\tilde{x}_{t_d}$ are the states of old and correct reference trajectory. It is important to note, that even though the formulation developed below is for a stair height change, with slight modifications this formulation can be utilized to capture other disturbances such as impact.

In order to address the stair height change disturbance, the first step is to build a gait library with multiple stair height trajectories, which one can then index into. These trajectories are generated using FROST and the formulation described in Section IV-B. To simplify the problem, we assume that we know, a priori, the time the new stair height will be conveyed from the vision system (t_d), which will determine when the controller needs to start tracking the correct reference trajectory, and the new stair height. With these assumptions, we can then define a new reference trajectory (disturbance trajectory) for our NMPC controller to track, composed of the old and correct trajectories from the gait library. By defining the disturbance trajectory as the reference, we ensure that the NMPC controller gets a preview of the stair height change as depicted in Figure 15. As a design choice, the new reference trajectory is right continuous at the switching time t_d. Therefore,

$$\tilde{x}_{t_d} = \hat{x}_{t_d-1} + \Delta T f(\hat{x}_{t_d-1}) + w_{t_d} \tag{16}$$

where $f : X \to Y \ \forall x_i \in X, y_i \in Y$where $f(x_i) = y_i$ is the dynamic equation. Note that $\hat{x}, \tilde{x} \in X$. The disturbance preview

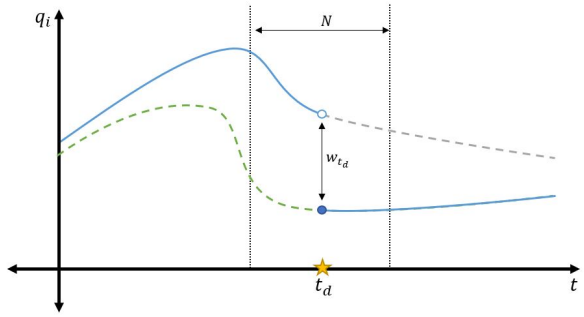

Fig. 15: New Reference Trajectory for Incorporating Disturbance Preview

problem can then be formulated with slight modifications to Equation 2 as follows:

$$\min_{Z} \ J_N$$

$$\textbf{subject to} \quad G = \begin{bmatrix} x_0 - \hat{x}_0 \\ x_1 - (x_0 + \Delta T f(x_0, u_0)) \\ \vdots \\ x_{k_{t_d}} - (x_{k_{t_d}-1} + \Delta T f(x_{k_{t_d}-1}, u_{k_{t_d}-1})) \\ \vdots \\ x_N - (x_{N-1} + \Delta T f(x_{N-1}, u_{N-1})) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ w_{t_d} \\ \vdots \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} I \\ -I \end{bmatrix} Z \leq \begin{bmatrix} Z_{max} \\ -Z_{min} \end{bmatrix} \tag{17}$$

A. Results

With the formulation above,where $Q_{term} = Q$, $t_d = 0.055$, and the previously defined cost matrices, The controller, however, is not able to converge to the new stair height. We hypothesize, that with better tuned cost matrices, a cost function on the swing foot's position, and possibly a longer horizon, the controller will eventually converge to the new stair height over several steps. The results with a disturbance reference trajectory with an initial desired stair height of 0.04 and an ending stair height of 0.06 are shown in Figure 16, whereas Table IV contains the tracking results of various disturbance reference trajectories.

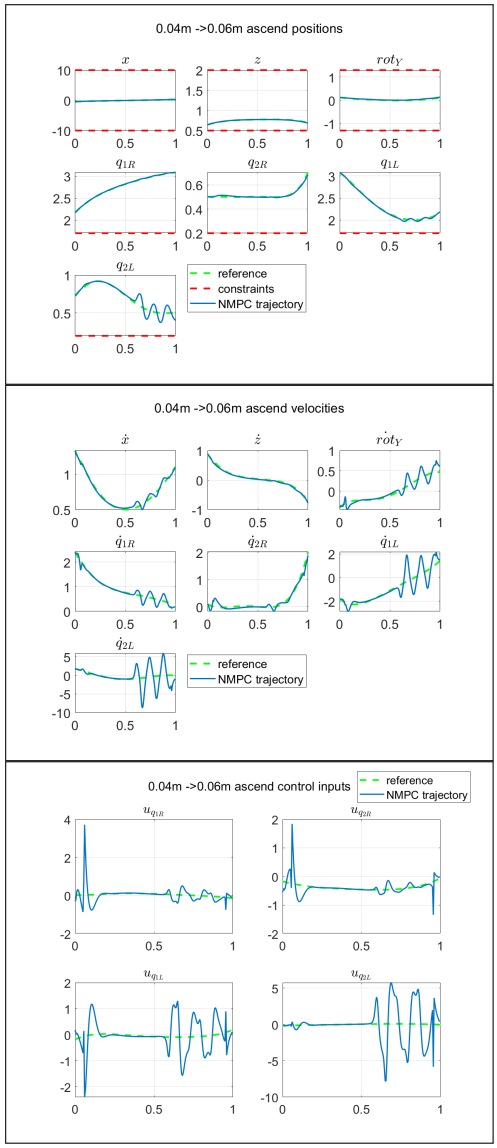

Fig. 16: Disturbance Preview Trajectory Tracking NMPC for a trajectory that has an initial stair height of 0.04 and an ending stair height of 0.06. The disturbance reference trajectories (green) are tracked by the system (blue) while satisfying constraints (red).

IX. Discussion

Due to time constraints, we did not formulate the LQ-MPC for trajectory tracking of the five link walker. We hypothesize that using the linearized dynamics might increase computational efficiency but at the loss of accuracy.

TABLE IV: System and NMPC Parameters

Stair Height: Old Traj (m)	Stair Height: Correct Traj (m)	End Swing Foot Position (m)	Norm Position Error	Norm Velocity Error
0.06	0.04	0.057	0.128	0.485
0.2	0.04	0.1364	0.018	3.376
0.04	0.06	0.0518	0.012	0.456
0.04	0.2	0.105	0.012	3.435

The goal of disturbance preview is to use updated vision information and gait library trajectories in order to achieve a different step height. Therefore we suggest modifying the cost function to directly penalize the output position of the swing foot. Additional output constraints could also be added to the minimization problem.

In this report, we presented our TT-NMPC and R-NMPC results for a cart-pole and five link walker. We proved local asymptotic stability of our controllers and provided some robustness results. Lastly, we developed a disturbance preview which can be modified for impact. This will allow us to extend our results to fully the hybrid system for periodic stair climbing.

Appendix I
Team Contributions

Grant A. Gibson took the lead in the cart-pole analysis, while M. Eva Mungai took the lead in trajectory generation using FROST. Both team members contributed equally to the formulation and analysis of the five link walker TT-NMPC and R-NMPC.

Appendix II
Appendix: Code

A github repository of our code can be found at https://github.com/Emungai/mpc_fiveLink . The repository contains all of our FROST and CasADi code for nonlinear trajectory generation and NMPC controller computations.

References

[1] X. Da, O. Harib, R. Hartley, B. Griffin, and J. W. Grizzle, "From 2d design of underactuated bipedal gaits to 3d implementation: Walking with speed tracking," *IEEE Access*, 2016.

[2] A. Hereid and A. D. Ames, "Frost*: Fast robot optimization and simulation toolkit," *IROS*, 2017.

[3] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, In Press, 2018.

[4] A. Wachter and L. T. Biegler, "On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming," *Mathematical Programming*, 2006.

[5] I. Kolmanovsky, "Introduction to lq model predictive control," *Lecture Notes: AEROSP 740*, Winter 2019.

[6] R. Tedrake, *The Acrobot and Cart-Pole*. MIT OpenCourseWare, 2009, ch. 3.

[7] Q. Nguyen and K. Sreenath, "Optimal robust safety-critical control for dynamic robotics," *American Control Conference*, 2016.

[8] M. M. D. James B. Rawlings, David Q.Mayne, *Model Predictive Control: Theory, Computation, and Design 2nd Edition*. Nob Hill Publishing, 2017, ch. 2.

[9] C. R. D.Q. Mayne, J.B. Rawlings and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, 2000.

[10] K. A. Mina Kamel, Thomas Stastny and R. Siegwart, "Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system," *Robot Operating System(ROS)*, 2017.

[11] C. C. J. H. C. Eric R. Westervelt, Jessy W. Grizzle and B. Morris, *Feedback Control of Dynamic Bipedal Robot Locomotion*. CRC Press, 2009, ch. 5.

[12] M. W. Mehrez, "Mpc-and-mhe-implementation-in-matlab-using-casadi," 2019. [Online]. Available: https://github.com/MMehrez/MPC-and-MHE-implementation-in-MATLAB-using-Casadi/tree/master/workshop_github