Eindhoven University of Technology

MASTER

Offset-free MPC for a ball-balancing Stewart platform

Kempers, Boudewijn

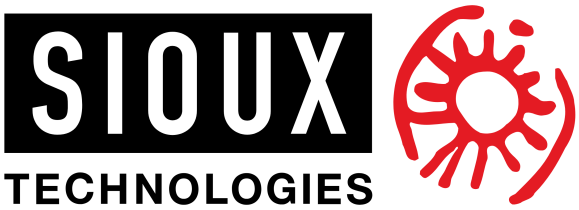
Award date:
2023

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain



DEPARTMENT OF MECHANICAL ENGINEERING
CONTROL SYSTEMS TECHNOLOGY

Offset-free MPC for a ball-balancing Stewart platform

B. Kempers (1398164)
Master thesis CST2023.021

TU/e Supervisors:

prof.dr.ir. W.P.M.H. Heemels
dr.ir. D. Krishnamoorthy
ir. T.J. Meijer

Sioux Technologies Supervisor:

dr.ir. C. Gruijthuijsen

This report was made in accordance with the TU/e Code of Scientific Conduct for the Master thesis
Eindhoven, April 30, 2023

Offset-free MPC for a ball-balancing Stewart platform

Boudewijn Kempers

***Abstract*—Model predictive control (MPC) has yet to gain popularity in industry, in particular in the application for motion systems. The high computational burden of solving MPC online, and a motion system's demand for high sampling rates made it impractical in the past. However, with the increasing availability of computing power and advancements in optimization software efficiency, MPC has now become a viable solution for motion control applications. This field of application is currently dominated by classical control methods such as PID. In this work, we examine the potential of MPC for motion systems by comparing it on a demonstrator system with PID. To compensate for the unmeasured disturbances that affect this system, the standard MPC is extended by an offset-free MPC scheme. Finally, the advantages of (offset-free) MPC for motion systems are demonstrated for several use-cases by means of experiments on the system.**

I. Introduction

THE general purpose of a controller is to optimally achieve a certain target, considering potential system bounds. Therefore, it is natural to formulate a controller as a constrained optimization problem. From this insight, it can be stated that model predictive control (MPC) is a very suitable way to control a real system: it approaches the control problem as a constrained optimization problem and considers the system's state-space model, and its physical limitations through constraints in the optimization problem. Moreover, unlike most frequency-domain classical control methods, MPC is able to effectively handle non-linear and multiple-input-multiple-output (MIMO) systems. Additionally, since MPC uses a model to predict future states and generate control input, it can easily adapt to changes in the system model, which will prove to be instrumental for our application.

However, despite these advantages, MPC has yet to gain significant popularity in industry, in particular for applications with fast dynamics, e.g. motion control systems. The main reasons for this slow adoption are the long computation times, which limits the sampling rate of the control loop, the dependency on an accurate model, and the stability and robustness guarantees, which are difficult to prove but important for expensive (industrial) systems. However, over the past 15 years, accurate modeling techniques, optimization libraries, and robustness and stability theory have matured. We also experienced a growth in available computing power. These developments have opened up a new application domain for MPC, namely the domain of systems with fast dynamics.

Although adoption has been slow, some work of MPC for systems with fast dynamics has already been done. Industries such as the automotive-, aerospace- and automation industry have shown successful implementation [1], [2]. Furthermore, in [3], MPC is applied on a motion platform with time delay. This work shows that MPC can outperform PID control for non-smooth trajectories. Moreover, in [4], explicit MPC outperformed PID and LQR controllers based on the settling time of balancing a ball on a ball-and-plate system.

Explicit MPC performs offline optimization to construct a function, which is generally represented as a look-up table. This function, with state and parameter input, is used to formulate a control sequence online. The size and required memory of this function increases rapidly with the complexity of the model and the number of constraints. Moreover, the function is only optimal for a certain set of operating conditions. When the optimization problem is significantly changed by means of unmeasured disturbances, the pre-computed and fixed analytical solution, i.e. the function, is no longer optimal. In summary, explicit MPC allows for faster computation times for low-order systems when compared to conventional MPC, but is not flexible when subjected to unmeasured disturbances.

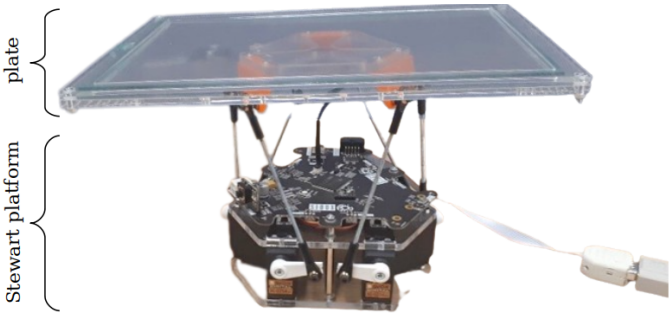


Fig. 1: The demonstrator system featuring a Stewart platform and plate.

Previous work on MPC for systems with fast dynamics denote challenges in the integration of MPC with respect to solve times and challenges in achieving zero-offset steady-state behavior. These challenges must be overcome in order to accelerate the acceptance and adoption of MPC in motion systems. In order to identify and propose solutions to these challenges, a low-budget ball-and-plate demonstrator system has been built, see Figure 1. This system features a Stewart platform with a flat plate, on which a ball is balanced and controlled to a target position. The system's inherent instability, multiple inputs and outputs, its kinematic limits and the fast dynamics require a (constrained) stabilizing controller with a fast sampling rate. These properties make the ball-

balancing Stewart platform challenging but also very suitable for examining the potential of MPC.

In a recent pilot study, a first version of an MPC scheme was implemented for this platform [5]. The pilot only laid the foundations for examining the full potential of MPC. However, the attained control performance was poor due to issues with inconsistent time-delays and unmeasured disturbances, which led to oscillations and residual steady-state errors respectively. Also, no comparison with other controllers was conducted, which prohibited the demonstration of any potential of MPC.

This work aims to solve the issues with (inconsistent) time delays, high MPC solve times and residual steady-state errors. Clearing these issues provides a pathway to the main contribution of this work, namely that of examining the potential of MPC for a ball-balancing Stewart platform.

The paper is structured as follows. In Section II, the problem and approach are stated. Section III describes the characteristics, kinematics and dynamics of the demonstrator system, and introduces the PID baseline controller to which the MPC is compared. Next, Section IV provides a general and an offset-free formulation of MPC, together with observers for state and disturbance estimation. The results with the simulated and real setup are presented in Sections V and VI, respectively. Finally, Section VII presents the conclusions and leads for future work. Furthermore, derivations of Euler rotations, state equations and kinematics are provided in Appendices A, B and C, respectively.

Notation. Scalars, vectors and matrices are denoted by, respectively, lowercase letters (a), lowercase bold letters ($\boldsymbol{a}$) and uppercase bold letters ($\boldsymbol{M}$). Next, coordinate frames are denoted by $\mathcal{O}$. This is used to express a vector, identified by a subscript, in a reference frame, denoted by a superscript, e.g. by $\boldsymbol{a}_{id}^{ref}$.

II. Problem statement

In this work, the following three problems are considered.

1. The demonstrator system is limited to being modeled using first principle methods, leading to unmodeled friction terms, kinematic uncertainties and motor dynamics. Moreover, the assumption that the system is mounted on a level surface, which is often invalid in practice, also results in modeling errors. The modeling errors translate to unmeasured disturbances that cause a steady-state error for an MPC scheme without an integrator, as we will discuss in more detail later on. A similar problem was addressed in [6] by using an offset-free MPC scheme of [7]. This scheme facilitated the estimation and mitigation of the disturbance caused by unmodeled friction terms. Motivated by their successful implementation, we seek to answer the following question: *How effective is the offset-free MPC scheme in improving the robustness of the ball-balancing Stewart platform against unmeasured disturbances?*

2. The accuracy of a model, i.e. absence of modeling errors, is directly associated with the model complexity. A complex physical system such as the Stewart platform can be approximated more accurately using a high- rather than a low order model. Thus, considering the fact that unmodeled dynamics were the cause of the previous problem, a high-order model is desired. However, the drawback of using a high-order model in MPC is the large amount of time required to solve MPC's optimization problem. The online solve time introduces a delay, and sets an upper limit to the rate at which a closed-loop control loop can be controlled. On the other hand, the control rate cannot be chosen too low either due to the fast dynamics of the inherently unstable ball-and-plate system. With this in mind, the second research question is stated as follows: *To what extend can the MPC model be simplified, minimizing the effect of disturbances by unmodeled dynamics, while achieving a control rate of* 200Hz*?*

3. The adoption of MPC in motion systems has been slow, despite the advantages of MPC over other control methods. However, solving the previous two problems allows us to demonstrate the potential of MPC in motion systems, e.g. using the demonstrator setup. This could accelerate the adoption of MPC in motion systems. A demonstration of this kind is only quantitative when compared to another controller, preferably one that is already commonly applied on motion systems. This raises the following question: *How does MPC perform on a motion system, according to the performance criteria introduced in Section V, when compared to PID in a series of use-cases which include setpoint-tracking, reference-tracking and obstacle avoidance.*

III. System description

In this section, specifications on the system and its control software are provided. Next, the kinematics and dynamics of the Stewart platform and plate are introduced. And finally, a PID controller used to set a performance baseline for MPC, is presented.

A. Specifications

System hardware.
The hardware under study is a single-stage Stewart platform with a plate attached to its end effector, see Figure 1. The Stewart platform comprises six legs that connect *base* and *plate* through joints.

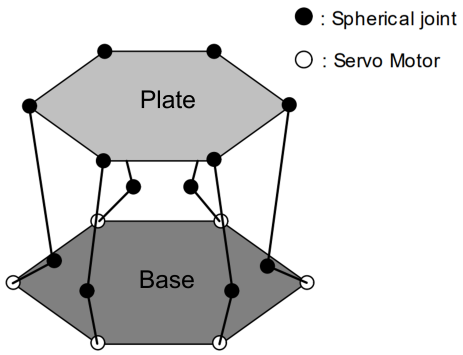


Fig. 2: Concept of the Stewart platform with rotary actuators in a 6-6 arrangement, adapted from [8].

These legs are positioned in a 6-6 arrangement, meaning that their joints are positioned distinctly and arbitrarily positioned

on the plate and base, see Figure 2. Most Stewart platforms are equipped with prismatic actuators to manipulate the plate. To reduce cost and space, the platform considered here uses servo's with a crank and lever mechanism. This mechanism converts the rotational movement of the servo's into a translational movement. The *base* plane, which is defined to intersect the servo's axes of rotation, remains fixed in place, and the *plate* can be manipulated in the 6 DoF space. Moreover, the plate is equipped with a pressure-sensitive resistive film layer, which provides 2D position measurements of objects that exert sufficient pressure on its surface. The hardware is provided by *Sioux Technologies* and will be used to demonstrate the capabilities of (offset-free) MPC.

In a control context, the hardware is defined as the plant, see also Fig. 3. The input of the plant, i.e. the servo angles denoted by q_i^b with $i \in \mathbb{N}^*_{\leqslant 6}$, is used to manipulate the translation and rotation (i.e. the pose, denoted by $\mathbf{T}$) of the plate. This is combined with position feedback $\mathbf{y} \in \mathbb{R}^2$ to formulate the following closed-loop control objective: *Balance a (heavy) ball, rolling on top of the plate, to a specific position on the plate*. In order to achieve this objective with MPC as a

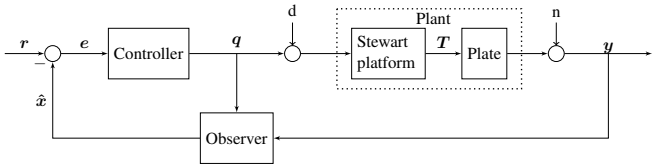


Fig. 3: Control diagram with error $\boldsymbol{e}$, reference $\boldsymbol{r}$, servo angles $\boldsymbol{q}$ as control input, the measured output $\boldsymbol{y}$ and estimated state $\hat{\boldsymbol{x}}$. The plant consists of a Stewart platform that manipulates the pose $\mathbf{T}$ of the plate, and is disturbed by process and measurement noise ($\boldsymbol{d}$ and $\boldsymbol{n}$).

controller, a dynamic model of the plant needs to be integrated in the MPC scheme. However, due to the strict requirement to solve MPC online, we must be aware of the complexity of this model. The authors of [9] and [10] show that the dynamic model of a parallel manipulator in general, and a Stewart platform (SP) in particular, is very complex and highly nonlinear due to its closed-loop kinematic characteristics. Moreover, the dynamics of the SP and the dynamics of the plate develop in significantly different time scales, such that in practice, it suffices to model the Stewart platform with a static (kinematic) mapping. For these reasons, the MPC scheme only considers the dynamic model of the plate, hereinafter referred to as the system. Before providing the derivation of the kinematic mapping of the Stewart platform, and dynamics of the system, a system simplification is made. The derivation of the kinematic mapping of the Stewart platform, and dynamics of the system are provided in the following subsections.

Control software

The firmware of the demonstrator system allows for external control of the servo actuators, and for access to measurements of the ball position. The processing of the measurements and formulation of a control input is done on an external PC running Linux. Software related to communication, kinematic calculation and reference generation is written in the C++ language.

For the design (tuning) and implementation of a controller on a physical setup it is valuable to have a simulated equivalent of the closed-loop system. In a simulated setup, the physical plant is replaced by an approximated model. When accurately approximated, a simulation can provide valuable insight in the behavior of the control system without the need and care for a physical system. The simulation in this work is performed using the Robot Operating System (ROS) library [11]. This library captures the components of the control diagram as computation blocks called *nodes*. The nodes are connected and communicate by sending and receiving data. As a basis, ROS-nodes for the reference, the controller, the simulated plant and the observer are implemented. The nodes and their interconnections, called a framework, are initialized using a parameter file. Additionally, the set of active nodes is selected using a configuration file. This modularity allows us to easily setup and implement various experiments.

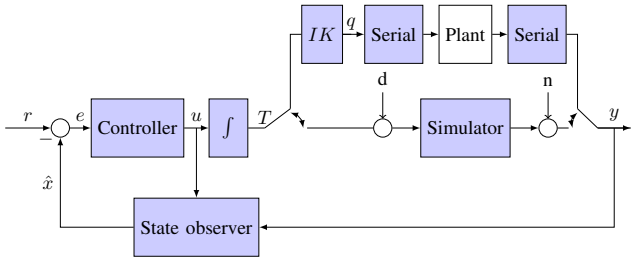


Fig. 4: Control diagram with ROS nodes in blue, configured with different implementations for controller and state observers, functionality for switching between simulation and real experiments.

The full framework, used for performing both simulation and real (hardware in the loop) experiments, is presented as a control diagram in Figure 4. The technical aspects of the diagram and implementation details of the ROS-nodes, depicted in blue, will be discussed in the following sections. However, let us first introduce its components: The *controller* node in the framework can be configured with a PID and an MPC implementation, which allows for a performance comparison between controllers. Next, the *state observer* node can be configured with a Kalman Filter (KF) and a Moving Horizon Estimator (MHE). The MHE is, like MPC, formulated as an optimization problem and will be implemented to examine the impact of optimization-based observers on the overall computation time. The MPC scheme and MHE observer are both implemented using the Acados library [12]. This library contains a collection of solvers and tools for fast embedded optimization. It also interfaces with high-level languages and is written on top of BLASFEO [13], a high-performance linear algebra library. This basis, together with automatic C-code generation, makes Acados suitable for the time-sensitive execution of MPC on a highly dynamic system like the demonstrator system. In addition to solving optimization problems, Acados also contains a module for simulation of plant models. Which leads to the next node in the control diagram: the *simulator* node. This node is configured with a four-stage explicit Runge-Kutta integrator, which integrates the nonlinear

dynamics of the plate system. As mentioned previously, the framework can be configured for using the simulator or the demonstrator setup, here denoted as the plant. However, in order to include the plant in the loop, two additional nodes are required. First, the *serial* node for communicating with the plant using the RS232 serial interface. This node encodes outgoing messages and decodes incoming messages according to a SLIP protocol. The outgoing messages contain firmware configuration settings, and setpoints for the servo actuators of the Stewart platform. The incoming messages contain measurements of the position of the ball. An iteration of a closed-loop simulation or real experiment gets initiated by an incoming measurement. Once a new measurement is received, an estimate of the full state is made. With this information, a control input gets formulated and is then sent to the (simulated) plant. This way, the plant determines the control rate. The second node required for performing experiments with the plant in the loop is the node for the inverse kinematic mapping (*IK*) of the Stewart platform. This node maps the pose of the plate $\boldsymbol{T}$ to the servo angles of the Stewart platform $\boldsymbol{q}$. Section III-C explains this mapping in more detail. However, in order to better understand the plant, the forward kinematic mapping is first provided in the following section.

B. Forward kinematics of the plate system

The process of determining the position and orientation of the end effector, i.e. the plate, given the joint angles of the system is called forward kinematics. This process is crucial for the derivation of the system's dynamics in Section III-D, since the motion, forces and torques of system are affected by the plate's position and orientation.

The forward kinematic problem of the *Stewart platform* can be interpreted as the mapping of a set of servo joint angles, denoted by $\boldsymbol{q}$, to the plate's position and orientation, i.e. the pose, denoted by $\boldsymbol{T}_p^b$. In contrast to serial structures, the forward kinematics of parallel manipulators is very complicated [14]. It involves a set of highly coupled nonlinear equations, that in general does not admit a closed form and unique solution [15]. In fact, the authors of [16] prove the existence of up to 40 different plate poses for a given set of servo angles. Since solving the forward kinematics of the Stewart system is not a prerequisite for the derivation of the dynamic model of the plate system, it is decided not to include it in this work. The rest of this section focuses on the forward kinematics of the *plate system*, see Figure 5.

The forward kinematics of the plate system expresses the mapping of the joint space Euler angle vector in the plate frame, i.e. the plate angles denoted by $\boldsymbol{\theta}_p = [\alpha, \beta, \gamma]_p^T$, to the plate's pose in the base frame, denoted by $\boldsymbol{T}_p^b$. This mapping is expressed by

$$\boldsymbol{\theta}_p \xrightarrow[\text{kinematics}]{\text{forward}} \boldsymbol{T}_p^b \tag{1}$$

With the plate's pose $\boldsymbol{T}_p^b$ comprised of the translation vector, denoted by $\boldsymbol{t}_p^b$, and the rotation matrix, denoted by $\boldsymbol{R}_p^b$. Firstly, the translation between the plate frame w.r.t. the base frame is given by

$$\boldsymbol{t}_p^b = \begin{bmatrix} x^b & y^b & z^b \end{bmatrix}_p^T. \tag{2}$$

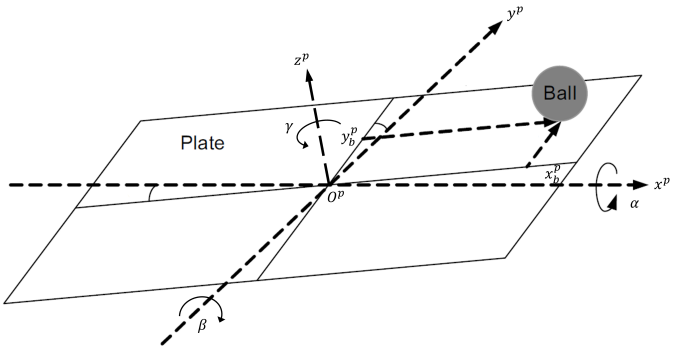


Fig. 5: Ball-balancing plate system

And lastly, the rotation of the plate frame w.r.t. the base frame is given by

$$\boldsymbol{R}_p^b = \begin{bmatrix} c\gamma c\beta & c\gamma s\beta s\alpha - s\gamma c\alpha & c\gamma s\beta c\alpha + s\gamma s\alpha \\ s\gamma c\beta & s\gamma s\beta s\alpha + c\gamma c\alpha & s\gamma s\beta c\alpha - c\gamma s\alpha \\ -s\beta & c\beta s\alpha & c\beta c\alpha \end{bmatrix}_p \tag{3}$$

with $c\theta = \cos(\theta)$ and $s\theta = \sin(\theta)$. The complete derivation of the rotation matrix of Eq. (3) can be found in Appendix A.

Thus, combining (2) and (3) allows to express the pose as

$$\boldsymbol{T}_p^b = \left[\begin{array}{c|c} \boldsymbol{R}_p^b & \boldsymbol{t}_p^b \\ \hline \boldsymbol{0} & 1 \end{array}\right]. \tag{4}$$

Note that the pose defined in (4) is in fact a transformation matrix, which is used to transform position coordinates in the plate frame, denoted by $\boldsymbol{p}^p = [x^p, y^p, z^p, 1]^T$, to position coordinates in the base frame by $\boldsymbol{p}^b = [x^b, y^b, z^b, 1]^T = \boldsymbol{T}_p^b \boldsymbol{p}^p$. Also, note that the rotation matrix of (3) is a function of the angles of the plate, i.e. of $\boldsymbol{\theta}_p$, and that, in practice, the translation vector $\boldsymbol{t}_p^b$ will be set stationary. Therefore, the pose $\boldsymbol{T}_p^b$ is only a function of the variable vector $\boldsymbol{\theta}_p$. This concludes the mapping shown in (1).

C. Inverse kinematics of the Stewart platform

As previously mentioned, the control objective is to balance the ball on the plate by manipulating the pose of the plate, i.e. using control input $\boldsymbol{T}$. However, this input cannot be controlled directly: it requires the inverse kinematic mapping that describes the transformation of the plate pose, $\mathbf{T}_p^b$, to a set of servo joint angles, q_i with $i \in \mathbb{N}_{\leqslant 6}^*$ of the Stewart platform. This mapping is denoted by

$$\boldsymbol{q} \xleftarrow[\text{kinematics}]{\text{inverse}} \boldsymbol{T}_p^b. \tag{5}$$

The inverse kinematics of the Stewart platform are inspired by [8], which considers a simplified modeling approximation of the Stewart platform by assuming small operating angles over α and β, no rotation over z_p ($\gamma = 0$) and by neglecting centrifugal forces. As the full derivation is quite cumbersome and platform-specific, it is provided in Appendix C.

The derivation of the mapping in (5) is dependent on kinematic properties, such as the mounting position of the servo actuators, the length of the servo arms, and the length of the legs connecting the base and the plate. Due to the absence of

an accurate 3D model, these properties are measured manually. The measurement errors, and consequential disturbance due to an inaccurate kinematic mapping that this method imposes are assumed to be insignificant.

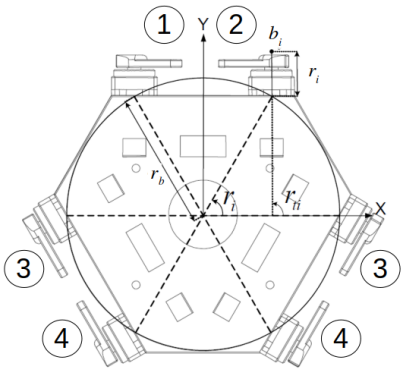


Fig. 6: Stewart platform base and numbered servo layout

The Stewart platform has, by design, a lot of symmetries, as depicted in Figure 6. Since the servo-arms, and their respective operating range are mirrored for even- and odd-numbered servos, the derivation of the servo angles leads to different equations for $\boldsymbol{q}$:

$$q^{odd} = arcsin\left(\frac{c_i}{\sqrt{a_i^2 + b_i^2}}\right) - arctan2(b_i, a_i) \tag{6a}$$

$$q^{even} = \pi - arcsin\left(\frac{c_i}{\sqrt{a_i^2 + b_i^2}}\right) - arctan2(b_i, a_i) \tag{6b}$$

with

$$\begin{aligned} a_i &= -l_z, \\ b_i &= l_x \sin r_{ti} - l_y \cos r_{ti}, \\ c_i &= \frac{\|d\|^2 - \|\boldsymbol{l}\|^2 - r_m^2}{2r_m}, \end{aligned}$$

where r_{ti} is the angle at which a servo motor (and arm) is mounted on the base, d is the length of a leg connecting the servo arm and plate, and r_m is the length of a servo arm, and $\boldsymbol{l} = [l_x, l_y, l_z]^T$ is the distance between the rotation axis of the servo ($\boldsymbol{b}_i$) and the attachment joint on the plate ($\boldsymbol{p}_i$), i.e. the virtual actuator length, see also Figures 6 and 7. The virtual length is expressed by

$$\|\boldsymbol{l}\| = (\boldsymbol{p}^b - \boldsymbol{b}^b)^T(\boldsymbol{p}^b - \boldsymbol{b}^b).$$

Note that the position of the attachment joint in $\mathcal{O}_b$ is determined by the pose of the plate. This concludes the inverse kinematic mapping of the pose of the plate $\boldsymbol{T}_p^b$, in joint space, to the attachment joint $\boldsymbol{p}^b$, to a servo angle q_i with $i \in \mathbb{N}^*_{\leqslant 6}$ (6) in Cartesian space.

D. Dynamics

A dynamic model allows to predict and control the behavior of a system over time, which is a key component of model predictive control. A dynamic model contains the equations of motion of a manipulator in the joint space. Full dynamic modeling of the Stewart platform would provide detailed insight. However, as previously mentioned, the complexity of the dynamic model of a parallel manipulator makes this inappropriate for use in the real-time application of model predictive control. For this reason, it is decided to model the dynamics of the ball-and-plate system only. Since this work is a continuation and expansion of [5], it is convenient to use similar notation for the system dynamics. The dynamics are therefore expressed using the *Lagrange formulation*, starting with its general form:

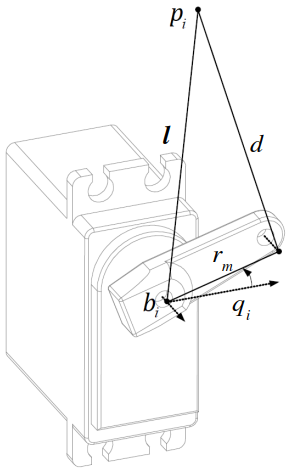


Fig. 7: Servo crank and lever mechanism with virtual actuator length $\boldsymbol{l}$

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{q}_i^p} - \frac{\partial T}{\partial q_i^p} + \frac{\partial V}{\partial q_i^p} = Q_i \tag{7}$$

Where q_i^p is the i^{th} direction coordinate in $\mathcal{O}_p$, with

$$\boldsymbol{q}^p = [\underbrace{\alpha\ \beta\ \gamma}_{\boldsymbol{\theta}_{plate}^p}\ \underbrace{x_b\ y_b\ z_b}_{\boldsymbol{p}_{ball}^p}]^p. \tag{8}$$

The kinetic and potential energy are denoted by T and V, and the external forces by Q_i. The following general assumptions are made in order to reduce the complexity of the derivation and of the product of the dynamic model:

Assumption 1: There are no external forces acting on the ball-and-plate system, i.e. $Q_i = 0$.
In the static position and trajectory tracking test cases, that are being considered in this project (see Section V), no external forces are being applied on the setup. It is therefore reasonable to assume that no forces will be applied on the setup.

Assumption 2: The ball remains in contact with the plate at all times, i.e. $z_b(t) = r_b$ for all $t \in \mathbb{R}_{\geqslant 0}$.
A situation in which the ball loses contact with the plate, would result in a temporary loss of the ball's position state. This is undesired. Therefore, the accelerations of the plate will be constrained in order to maintain contact with the ball.

Assumption 3: The position of the plate coordinate system $\mathcal{O}_p$ with respect to the base coordinate system $\mathcal{O}_b$ is stationary, i.e. $\dot{p}_{plate}^p = \dot{p}_{plate}^b = 0$.
The plate translations and corresponding linear velocities w.r.t. the base have little influence on the ball-balancing performance, whereas they greatly increase the complexity of the dynamic model.

Assumption 4: The rotation of the plate over z^p is stationary, i.e. $\gamma = 0$.

A rotation of the plate over z^p would induce little to no displacement of the ball, due to the unconstrained rolling properties of the ball on the plate's surface. This degree of freedom, denoted by γ, does not contribute in the objective of controlling the ball towards a specific position setpoint, and is therefore fixed to zero.

Considering these assumptions, the energy equations can be derived. But first, since the energy equations are functions of the angular velocity $\boldsymbol{\omega}$, instead of the Euler angle rate $\dot{\boldsymbol{\theta}}$, a conversion must be made. The body-fixed Euler angle rates are related to the inertial angular velocity vector by matrix $\boldsymbol{B}(\boldsymbol{\theta})$. For instance, the angular velocity of the plate along the principal axes of a frame is given by

$$\boldsymbol{\omega}^p = \boldsymbol{B}^{-1}\dot{\boldsymbol{\theta}} \tag{9}$$

With $\boldsymbol{\theta} = [\alpha, \beta, \gamma]^T$ and $\dot{\boldsymbol{\theta}} = [\dot{\alpha}, \dot{\beta}, \dot{\gamma}]^T$ denoting the Euler angles and angular rates. Then, considering an intrinsic 1-2-3 Euler angle set, matrix $\boldsymbol{B}^{-1}$ obtained from [17] is given by

$$\boldsymbol{B}^{-1} = \begin{bmatrix} c\beta c\gamma & s\gamma & 0 \\ -c\beta s\gamma & c\gamma & 0 \\ s\beta & 0 & 1 \end{bmatrix} \tag{10}$$

Energy equations

The equation for the kinetic energy can be written as the sum of the angular kinetic energy and the translational kinetic energy, with their general forms $T_r = \frac{1}{2}\boldsymbol{\omega}^T\boldsymbol{I}\boldsymbol{\omega}$ and $T_t = \frac{1}{2}m\dot{\boldsymbol{p}}^T\dot{\boldsymbol{p}}$. The kinetic energy of the ball is written as

$$\begin{aligned} &T_b = T_{br} + T_{bt} = \\ &\frac{1}{2}\boldsymbol{\omega}_{ball}^{pT}\boldsymbol{I}_{ball}^{p}\boldsymbol{\omega}_{ball}^{p} + \frac{1}{2}\boldsymbol{\omega}_{plate}^{pT}\boldsymbol{I}_{ball}^{p}\boldsymbol{\omega}_{plate}^{p} + \frac{1}{2}m_b\dot{\boldsymbol{p}}_{ball}^{pT}\dot{\boldsymbol{p}}_{ball}^{p} \end{aligned} \tag{11}$$

With $\dot{\boldsymbol{p}}_{ball}^{p} = \boldsymbol{\omega}_{ball}^{p} \times \boldsymbol{r}_{ball}$ denoting the linear velocity of the ball, where $\boldsymbol{r}_{ball} = [0,\ 0\ , r_b]$. And, with $\boldsymbol{\omega}_{plate}^{p} = \boldsymbol{B}^{-1}\dot{\boldsymbol{\theta}}_{plate}$ denoting the angular velocity of the plate using the Euler transformation in (10) and $\boldsymbol{I}_{ball}^{p} = \frac{2}{5}m_b r_b$ the inertia matrix of the ball in the plate coordinate system.

Subsequently, the kinetic energy of the plate can be described as

$$T_p = T_{pr} = \frac{1}{2}\boldsymbol{\omega}_{plate}^{pT}(\boldsymbol{I}_{plate}^{p} + \boldsymbol{I}_{im}^{p})\boldsymbol{\omega}_{plate}^{p} \tag{12}$$

With $\boldsymbol{I}_{im}^{p} = -m_b[\boldsymbol{p}_{ball}^{p}]_{\times}^2$ being the contribution of the ball to the plate's inertia (creating an imbalanced plate) where $[\boldsymbol{p}_{ball}^{p}]_{\times}$ denotes the skew-symmetric matrix of the ball position w.r.t. O^p.

Finally, the potential energy of the ball is defined as:

$$V = m_b\boldsymbol{g}^{bT}\boldsymbol{p}_{ball}^{b} \tag{13}$$

With $\boldsymbol{p}_{ball}^{b} = \boldsymbol{R}_{p}^{b}\boldsymbol{p}_{ball}^{p} + \boldsymbol{t}_{p}^{b}$ denoting the position vector of the ball in the base coordinate system, where the position vector of the ball in O^p is given by $\boldsymbol{p}_{ball}^{p} = [x_b, y_b, z_b]^T$ and the gravitation vector by $\boldsymbol{g} = [0, 0, -g]$.

State equations

The system has been expressed in terms of kinetic and potential energy. On this basis, the state equations can be derived by formulating the energy equations as denoted in (7). The complete derivation, including the vector derivatives of the energy equations, is provided in Appendix B. Using equation (7) and $L = V - T$ as the expression of the Lagrangian, the equations of acceleration of the ball on the plate are expressed as follows.

$$\ddot{x}_b = \frac{5}{7}\left(\frac{2}{5}\ddot{\beta}r + \dot{\beta}^2 x_b - \dot{\alpha}\dot{\beta}y_b - g\sin\beta\right) \tag{14a}$$

$$\ddot{y}_b = \frac{5}{7}\left(\frac{2}{5}\ddot{\alpha}r + \dot{\alpha}^2 y_b - \dot{\alpha}\dot{\beta}x_b + g\cos\beta\sin\alpha\right) \tag{14b}$$

Equation (14) shows that, under assumption 4, the mass of the ball has no impact on the acceleration of the ball on the plate. Therefore, the choice of using a heavy ball for reliable position measurements on the pressure-sensitive plate should not influence the ball's acceleration. The dynamics of the ball-and-plate system can be linearized by the following assumptions as in [8], [18], [19]:

Assumption 5: The plate rotations α and β that will be used to balance the ball on the plate will be sufficiently small ($\theta = \pm 0.2\text{rad}$) to be linearized on $[\alpha, \beta] = [0, 0]$. This implies that $\cos\theta = 1$ and $\sin\theta = \theta$.

Assumption 6: The rate of change around the linearization point is small, thus $\dot{\alpha}^2 \approx 0$, $\dot{\beta}^2 \approx 0$ and $\dot{\alpha}\dot{\beta} \approx 0$.

Assumption 7: The contribution of the angular acceleration of the plate to the acceleration of the ball is much smaller than the contribution of the gravitational component, therefore, can be ignored, thus $\ddot{\alpha} \approx 0$ and $\ddot{\beta} \approx 0$.

The end effector of the Stewart platform, i.e. the plate, cannot attain angles larger than 0.2rad due to it kinematic restrictions. Moreover, the angular velocity and acceleration of the plate are limited by that of the servo actuators. It can therefore be assumed that the contribution of the plate's angular velocity and acceleration to the acceleration of the ball is less significant than that of the angle of the plate. Applying these assumptions applied to (14a) and (14b) leads to the simplified equations of motion.

$$\ddot{x}_b = -\frac{5}{7}\beta g \tag{15a}$$

$$\ddot{y}_b = \frac{5}{7}\alpha g \tag{15b}$$

This can be written in continuous time state-space form as:

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}\boldsymbol{x}(t) + \begin{bmatrix} 0 & 0 \\ 0 & -\frac{5}{7}g \\ 0 & 0 \\ \frac{5}{7}g & 0 \end{bmatrix}\boldsymbol{u}(t) + \boldsymbol{w}(t) \tag{16a}$$

$$\boldsymbol{y}(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\boldsymbol{x}(t) + \boldsymbol{v}(t) \tag{16b}$$

with the notions of the continuous time state vector $\boldsymbol{x}(t) = [x_b(t), \dot{x}_b(t), y_b(t), \dot{y}_b(t)]^T$, input vector $\boldsymbol{u}(t) = [\beta(t), \alpha(t)]^T$ and measurement vector $\boldsymbol{y}(t) = [x_b(t), y_b(t)]^T$. Vectors $\boldsymbol{w}(t) \in \mathbb{R}^4$ and $\boldsymbol{v}(t) \in \mathbb{R}^2$ are introduced to account for process disturbance and measurement noise respectively.

E. Baseline PID controller

A PID controller is implemented as a baseline/reference controller to which the performance of the (offset-free) MPC is compared with. In general, tuning a PID controller for a for MIMO systems such as the ball-and-plate system is a difficult task. This is due to the coupling between input- and output channels. However, the system dynamics in (16) show that this system is fully decoupled under assumptions $5-7$. This way, the two separate SISO systems can be controlled using two PID controllers. Each controller is responsible for controlling either one of the variables $\ddot{x}_b$ and $\ddot{y}_b$ by respectively manipulating β or α. The discrete time PID controller for β is defined as

$$u(k) = K_p \cdot e(k) + K_i \cdot \sum_{k-i}^{k} e(k) + K_d \cdot e_v(k) \tag{17}$$

with $i \in \mathbb{R}_{\geqslant 0}$ and where the input $u(k) = \beta(k)$ is comprised of the sum of the proportional K_p, integral K_i, and derivative K_d gains, that act on the position error

$$e(k) = x_b(k) - x_{b,ref}(k),$$

and the velocity error

$$e_v(k) = \frac{(x_b(k) - x_b(k-1)) - (x_{b,ref}(k) - x_{b,ref}(k-1))}{T_s} \tag{18}$$

The PID controller for α has a similar notation. In order to close the control loop, feedback of the ball position is provided. Since PID control also requires access to the ball velocity, an approximation is made, see (18). This approximation is based on the difference in position over one step in time (T_s) and is only accurate for constant velocities. However, the acceleration and deceleration of the ball on the plate will render the approximation inaccurate. In this case, a better approximation of the velocity can be made using a state observer, see IV-B.

IV. MPC DESIGN

In this section, the design of the model predictive controller is presented. Starting with a discrete-time MPC formulation, an adaptation to the dynamic model and elaboration of the design choices made for a computation efficient implementation. Additionally, a Kalman filter state observer is introduced to provide the MPC with complete access to the system's state. Furthermore, a moving horizon estimator state observer is presented to examine the impact of optimization-based observers on the overall computation time. These observers can also be implemented to estimate unknown disturbances. Finally, the offset-free MPC scheme, which combines the disturbance observer with MPC, is discussed in this section.

A. MPC

Model predictive control (MPC) is an advanced control method that uses a plant model to predict future states and inputs by solving a constrained optimization problem. Because the plant will be sampled at regular time intervals, it makes sense to discretise the continuous time state space model. However, the main reason of using a discretised model is because it allows the optimization problem to be described as a finite-, rather than an infinite dimensional problem. The discretisation of the linear time-invariant model in (16) is done by assuming zero-order hold (ZOH) for the input u, and rewriting it as

$$\begin{aligned} x_{k+1} &= A_d x_k + B_d u_k \\ y_k &= C_d x_k + D_d u_k \end{aligned} \tag{19}$$

with

$$\begin{aligned} A_d &= e^{AT_s} \\ B_d &= \left(\int_{\tau=0}^{T_s} e^{A\tau} d\tau \right) B \\ C_d &= C \\ D_d &= D \end{aligned}$$

where $T_s \in \mathbb{R}_{>0}$ denotes the sampling time, and $x_k \in \mathbb{R}^n$ and $u_k \in \mathbb{R}^m$ denote the state and control input at time $k \in \mathbb{N}$, respectively.

In linear constrained MPC, given the current state $x_{0|k} = x_k$, the following constrained finite-horizon quadratic optimal control problem (QP) is solved at every sampling instant k [20]:

$$\begin{aligned} \min_{\boldsymbol{x}_k,\, \boldsymbol{u}_k} \quad & x_{N|k}^T Q_N x_{N|k} + \Sigma_{i=0}^{N-1} x_{i|k}^T Q x_{i|k} + u_{i|k}^T R u_{i|k}, \\ \text{s.t.} \quad & x_{i+1|k} = A x_{i|k} + B u_{i|k}, \quad i \in \mathbb{N}_{\leq N-1}, \\ & M x_{i|k} + J u_{i|k} \leq c, \quad i \in \mathbb{N}_{\leq N-1}, \\ & M_N x_{N|k} \leq c_N, \end{aligned} \tag{20}$$

where Q, $Q_N \in \mathbb{S}_{\succeq 0}^n$, $R \in \mathbb{S}_{\succ 0}^m$ are the weighting matrices used in the MPC cost function and $N \in \mathbb{N}_{>0}$ denotes its prediction horizon. Moreover, $M \in \mathbb{R}^{p \times n}$, $J \in \mathbb{R}^{p \times m}$ and $c \in \mathbb{R}^p$ capture the polyhedral state and input constraints whereas $M_N \in \mathbb{R}^{q \times n}$ and $c_N \in \mathbb{R}^q$ characterize the terminal set constraints. The notation $x_{i|k}$ is used to denote the predicted state at future time instant $k+i$ in the optimal control problem (OCP) solved at sampling instant k, and a similar notation is adopted for the future inputs $u_{i|k}$. Finally, the sequences of predicted states and future inputs are defined as $\boldsymbol{x}_k := \{x_{1|k}, x_{2|k}, ..., x_{N|k}\}$, and $\boldsymbol{u}_k := \{u_{0|k}, u_{1|k}, ..., u_{N-1|k}\}$. After the optimal, i.e. cost minimizing, input sequence $\boldsymbol{u}_k^* := \{u_{0|k}^*, u_{1|k}^*, ..., u_{N-1|k}^*\}$ is determined, the first control action is implemented, i.e. $u_k = u_{0|k}^*$, and the OCP of (20) is solved again at time $k+1$ for $x_{0|k+1} = x_{k+1}$ leading to the receding horizon principle.

The OCP in (20), formulated as a linear least-squares problem, is implemented using the Acados library. As previously mentioned, this library interfaces with libraries for fast linear algebra and with several QP solvers, which is beneficial for the solve times. The Acados OCP is provided with a dynamic model in its linear and discretized form. This avoids Acados from performing additional linearization and discretization steps. Furthermore, to solve the OCP, a HPIPM solver is

selected which applies (partial) condensing techniques to exploit the structure, and the interior-point method to solve the quadratic problem (QP). However, the parameter that impacts the solve time most, besides the order of the linear model, is the length of the prediction horizon N. This parameter should be chosen sufficiently large, such that the transient response of the system is captured. While a large horizon generally results in better performance, compared with a short horizon, it also brings computational complexity, as the required number of operations scales cubic with N in cases where the QP structure is not exploited.

At this point, using the theory of this section and the Acados library a MPC scheme can be realised. However, this scheme only allows for the control of simulated plants, as these provide access to the full and true state and operate at any desired sample rate. However, real motion systems require high and consistent control rates to be stabilized. Morover, they often provide noisy and corrupted measurements. Although state observers are useful in filtering noise and completing the state by an estimate (see next subsection), their capacity is limited by the quality of the input they get. These, and other challenges regarding the implementation of MPC on a real system will be addressed Section VI-A.

Constrained optimization

The satisfaction of constraints in process control is particularly important, since efficiency often demands operating points on, or close to, the boundary of the set of admissible states and controls. These bounds are defined by the measurement and actuation limitations of the system. MPC's general formulation (20) imposes these bounds as inequality constraints, and the system dynamics as equality constraints. On the basis of these constraints, a feasible set can be constructed. The optimal solution to the optimization problem must be found within this set. For the ball-and-plate system, the state bounds (constraints) are defined by the area in which the position of the ball can be measured, i.e. the dimensions of the plate, and by the angles that the plate can attain. The bounds are symmetric in the upper and lower values. The upper state bound is defined as $u_{b,x} = [x_b, \dot{x}_b, y_b, \dot{y}_b, \beta, \alpha]^T = [0.09, 5, 0.11, 5, 0.2, 0.2]^T$. The input bounds are defined by converting the maximum angular velocity of the servo actuators, ω_{servo}, to the angular velocity of the plate ω_p. The enforced symmetric input bounds are chosen somewhat conservatively at $u_{b,u} = [\dot{\beta}, \dot{\alpha}]^T = [5, 5]^T$. These bounds relate to the final MPC configuration, which uses the angular velocity of the plate as input for the MPC, instead of the angle of the plate. This allows the rate of change of the angle to be penalized. A penalization of this kind acts as a low-pass filter on the angle To enable this change, the model in (16) has been adjusted to (21).

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{5}{7}g & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{5}{7}g \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{u}(t) \tag{21a}$$

$$\boldsymbol{y}(t) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{x}(t) \tag{21b}$$

With input $u(t) = [\dot{\beta}(t), \dot{\alpha}(t)]$ and state $x(t) = [x_b(t), \dot{x}_b(t), y_b(t), \dot{y}_b(t), \beta(t), \alpha(t)]^T$. This CT model is discretised according to the principles of (19). Now, using the assumption that the angular velocity is constant over a sampling instance, the plant input (angle) is calculated by $\theta(k) = \theta(k-1) + \dot{\theta}(k) \cdot Ts$.

The previously mentioned constraints are related to physical limitations. However, to better demonstrate the capabilities of MPC, it is be interesting to implement obstacle avoidance on the ball-and-plate system. For example, by using nonlinear constraints, a circular or square region could be defined as an obstacle, and the system could be prevented from entering this region.

Where the input and state bounds are generally defined as hard and inviolable rules, the additional constraints are defined as soft or relaxed states. This is done to preserve feasibility of the optimization problem while strongly discouraging the MPC from violating the constraints. Constraint relaxation is done by adding slack variables, denoted by s. The slack variable transforms the inequality into an equality constraint and ads a non-negativity constraint on the slack variable. This is based on the observation that $h(x,u) \leq \bar{h}$ if and only if there is an $s \geq 0$ that satisfies $h(x,u) + s = \bar{h}$, [21]. When (x,s) is feasible for the relaxed optimization problem, then x is a feasible for the original problem. Hence the optimization problems are equivalent. In practice, it requires careful tuning of the slack variables: Acados requires to set upper and lower bounds for the slack variable, with s_u and s_l respectively. The lower bound must be fixed to a sufficiently low but non-negative value, here chosen as 0. The upper bound must be chosen sufficiently high, such that a constraint violation has a significant impact on the total cost. However, choosing this value too high can cause numerical issues, or causes the problem to be ill-conditioned. In the experiments where relaxed constraints are applied (presented in the following sections), choosing a value for s_u which is slightly above the maximum cost-to-go has proven to give the desired result.

B. Observer

In most practical problems, states of the system are not directly measured and must be estimated/observed. The quality of state estimates has important bearings on the overall performance of a model predictive controller. Two state observers are examined in this work: a linear Kalman filter (KF) and a moving horizon estimator (MHE). The KF will serve as the primary observer in the (offset-free) MPC scheme, while

the MHE will be included to assess its feasibility for implementation on a fast-sampling control system by analyzing computation time. For a linear unconstrained system, disturbed by Gaussian white noise, the authors of [22] show that MHE is equivalent to the KF for certain choices of weighting matrices but regardless of horizon size. However, when applied on a nonlinear system, the MHE (with $N > 1$) retains all recent information and is more efficient at summarizing past information when compared with a nonlinear (extended) KF.

Although estimation performance of MHE and KF on a linear system might be similar, the ability to add constraints has several advantages: Firstly, the system bounds, which are generally physically and/or electrically determined limits, can be enforced on the estimation. This forces the estimated state to represent only feasible states. This is a requirement for the MPC scheme, as this operates on these same bounds. Where the 'out-of-bounds' estimated state of the KF has to be truncated, it is not even considered being a feasible estimate by the MHE. Secondly, the application of constraints can improve robustness against faulty measurements, i.e. outliers. As the KF is designed based on constant covariances, sudden changes due to faulty measurements can lead to unrealistic state estimates. The application of limits on the unmeasured disturbances, measurement noise and states can improve the reliability of the estimates in this instance. Both state observers are introduced next.

1) Kalman filter: The Kalman filter (KF) is the dual of the Linear Quadratic Regulator, where both have a cost function, dependent on matrices Q, R and P_0, which is minimized in order to estimate a state from a noisy measurement. Whereas cost matrices in LQR relate to the state and input, the cost matrices in KF relate noise covariances. The KF algorithm uses input data u_k and output measurements y_k to construct an estimate of the state and of the modelled disturbances, denoted by $\hat{x}_k$ and $\hat{d}_k$.

The KF algorithm comprises two steps: the prediction, or the a-priori estimate, and the update, or the a-posteriori estimate. Starting with the initial guess of the state $\hat{\boldsymbol{x}}_0$, its covariance matrix $\boldsymbol{P}_0$ and the covariance matrices of the input and output noise $\boldsymbol{Q}$ and $\boldsymbol{R}$.

Step 1: Prediction

- Perform open-loop estimation
  $\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{A}\hat{\boldsymbol{x}}_{k-1|k-1} + \boldsymbol{B}\boldsymbol{u}_k$
- Calculate the open-loop innovation covariance
  $\boldsymbol{P}_{k|k-1} = \boldsymbol{A}\boldsymbol{P}_{k-1|k-1}\boldsymbol{A}^T + \boldsymbol{Q}^T$

Step 2: Update

- Calculate the innovation covariance
  $\boldsymbol{S}_k = \boldsymbol{C}\boldsymbol{P}_{k|k-1}\boldsymbol{C}^T + \boldsymbol{R}$
- Calculate the Kalman gain
  $\boldsymbol{L}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{C}^T\boldsymbol{S}_k^{-1}$
- Perform the closed-loop estimation
  $\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{L}_k(\boldsymbol{y}_k - \boldsymbol{C}\hat{\boldsymbol{x}}_{k|k-1})$
- Calculate the closed-loop innovation covariance
  $\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{L}_k\boldsymbol{C})\boldsymbol{P}_{k|k-1}$

The KF only provides a non-diverging (stable) state estimate when $(\boldsymbol{A}, \boldsymbol{B})$ is stabilizable, i.e. all uncontrollable states are stable, and $(\boldsymbol{A}, \boldsymbol{C})$ is detectable, i.e. the unobservable States remain stable. In addition, it can be shown that the effect of the initial estimate $\hat{\boldsymbol{x}}_0$, and covariance matrix $\boldsymbol{P}_0$ on the estimate of the state $\hat{\boldsymbol{x}}_{k|k}$ is dampened as k is increased. The Q and R weighting matrices, with $\sigma^2_{w,i}$ and $\sigma^2_{v,i}$ on their diagonals, are tuned using the whiteness of the prediction error as a design criterion. Where the prediction error is defined as $\boldsymbol{e}_k = \boldsymbol{y}_k - \boldsymbol{C}\hat{\boldsymbol{x}}_{k|k-1}$. The whiteness is verified by analyzing the cumulative periodogram. In the case of excess high-frequency components, the weight for $\sigma^2_{w,i}$ must be lowered, and vise versa for an excess of low-frequency components. Note that, in this case, only the ratio of the weights is relevant.

2) Moving horizon estimator: The MHE estimates the state of a system by minimizing the difference between the predicted and the measured output over a finite time horizon. This method is the dual of MPC but uses a preceding rather than a receding horizon. Moreover, while MPC stability is affected by the terminal cost, the stability of MHE is equally affected by the arrival cost.

The positive definite weighting matrices R, Q and Q_0 are quantitative measures of the confidence in the output model, the dynamical system model, and the initial estimate, respectively. The diagonal values of these matrices are often chosen such that their diagonal values are inversely proportional to the squared magnitude of the corresponding variable.

The MHE can be formulated as a linear least squares QP. The MHE scheme builds on the following discrete-time model:

$$\begin{aligned} x_{k+1} &= F(x_k, u_k) + w_k, \quad x_0 = x(t_0), \\ 0 &= g(x_k, u_k), \\ y_k &= h(x_k) + v_k \end{aligned}$$

v_k and w_k denote the measurement and the process noise respectively. The following constrained finite-horizon optimal estimation problem is solved at every sampling instance k:

$$\begin{aligned} \min_{\boldsymbol{x}_j, \boldsymbol{u}_j \boldsymbol{w}_j} \; & \left\|x_L - \bar{x}_L\right\|^2_{Q_L} + \sum_{j=L}^{k} \|\tilde{y}_j \quad -h(x_j)\|^2_R + \sum_{j=L}^{k-1} \|w_j\|^2_Q \\ \text{s.t.} \; & x_{j+1} = F(x_j, u_j) + w_j, && j = L, ..., k-1, \\ & 0 = g(x_j, u_j), && j = L, ..., k \\ & x_{j,min} \le x_j \le x_{j,max}, && j = L, ..., k \end{aligned} \tag{22}$$

When using the MHE in closed loop, it takes L measurements to fill the horizon. To speed up the convergence to the actual state, a horizon initialisation process is examined. The initialization process fills the horizon with states generated by the Acados simulator that considers a nonlinear plant model. The simulator is initialized with a noisy initial state (based on the measurement at $k = 0$) and perturbed by a zero-input. However, as this initialization process with Acados simulator takes up to 50ms, it is not practical for use on the demonstrator system with a sampling rate of 200Hz. Another method uses a flexible QP formulation where the length of the horizon grows until L amount of measurements have been received. Unfortunately, Acados does not allow this kind of flexibility in the QP formulation. Therefore, in the final MHE setup, the initialization is done by simply filling the measured states, i.e. the ball position, with the values received at $k = 0$.

C. Offset-free MPC

In a setpoint tracking context, *offset* generally refers to a steady-state deviation between a reference state, i.e. the setpoint, and the actual state of the system that is being controlled. The occurrence of an offset can have many different origins. However, in the context of the Stewart platform and ball-and-plate, the offset is assumed to be induced by a disturbance. More specifically, by a time-invariant modeling disturbance originating from the assumption that the base of the platform is completely level, i.e. orthogonal to the gravitational vector. In practice, this assumption is often invalid as the platform will be placed/mounted on surfaces that are slightly tilted. The tilted base causes an offset in the plate's pose through the kinematic chain of the Stewart. However, since plate translation and rotation in γ are fixed, the disturbance causing an offset in the ball position is expected to be only in the plate angles, α and β.

Classical control methods such as PID control achieve zero offset for a constant reference by integrating the tracking error. Although it is not perfect due to the integral windup phenomenon, it is an easy-to-implement and effective solution for an steady-state offset. In MPC, the tracking error is not integrated. Instead, the model error is integrated. This method, which compensates for unmeasured disturbances, is called *offset-free* MPC [23].

The method for offset-free MPC comprises of three parts:

1) Modeling of the disturbance(s),
2) use the measurements and an augmented model of the system to estimate the disturbance, and
3) solve the modified QP to find the inputs that minimize the effect of the disturbance(s) on the controlled variables,

and considers the linear, time-invariant discrete system (19) in which $x_k \in \mathbb{R}^n$, $u_k \in \mathbb{R}^m$ and $y_k \in \mathbb{R}^p$, with the assumption that (A, B) is stabilizable and (C, A) is detectable. If the number of measurements is less than or equal to the number of manipulated variables, i.e. $p \leq m$, all measured variables can be controlled without offset [23].

Offset-free MPC is only effective to remove asymptotically constant nonzero disturbances. It achieves this by augmenting the original system with a replicate of the constant, nonzero disturbance model, and adjusting the QP accordingly. Thus, the states of the original system are moved to cancel the effect of the disturbance on the controlled variables. The general augmented model is given by

$$\begin{aligned} \xi_{k+1} &= A_a \xi_k + B_a u_k + w_k \\ y_k &= C_a \xi_k + v_k \end{aligned} \tag{23}$$

with the state vector denoted by $\xi_k = \begin{bmatrix} x_k \\ d_k \end{bmatrix}$, and the augmented model matrices by

$$A_a = \begin{bmatrix} A & B_{di} \\ 0 & I \end{bmatrix} \quad B_a = \begin{bmatrix} B \\ 0 \end{bmatrix} \quad C_a = [C\, C_{di}]$$

where $d_k \in \mathbb{R}^{n_d}$, $B_d \in \mathbb{R}^{n \times n_d}$ and $C_d \in \mathbb{R}^{p \times n_d}$. The vectors $w_k \in \mathbb{R}^{n+n_d}$ and $v_k \in \mathbb{R}^p$ are zero-mean white-noise disturbances. The pair of disturbance matrices (B_{di}, C_{di}) must be chosen such that (23) is detectable. This limits the dimension of disturbance vector d_k to be chosen less or equal to the number of measurements, i.e. $n_d \leq p$ [7].

As the tilted base disturbs the system in β and α, it is valid to add disturbance terms that describe this offset. Therefore, the model in (21) is augmented disturbance states $d_k = [d_\beta, d_\alpha]^T$. The disturbance matrices are chosen to be $B_{di} = \begin{bmatrix} 0 & 0 \\ -\frac{5}{7}g & 0 \\ 0 & 0 \\ 0 & \frac{5}{7}g \end{bmatrix}$, and $C_{di} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$. This augmented model is used by the KF/MHE to estimate the disturbance states. The estimate of the disturbance terms is then combined with the optimal MPC input u^* to attenuate the offset. Once the estimated disturbance converges to the actual disturbance, i.e. $\hat{d}_k = d_k$, zero-offset tracking is achieved. Note that this is only valid for the modeled disturbance, while the unmodeled disturbances caused by inaccuracies in the kinematic and dynamic model also contribute to an offset. In practice, all sources that cause a disturbance of the ball dynamics will be explained through these disturbance terms. This means that the disturbance observer tries to compensate for a part of the unmodeled disturbances as well.

The weights of the disturbance estimator are chosen such that a good balance between converge time and parameter stability is achieved. This is proven to be quite challenging, as high weight on the measurements causes quick convergence but a noisy estimate, and a high weight on the process results in slow convergence to a steady estimate. In order to maintain stabilizing capability of the disturbed ball-and-plate system, priority is given to fast convergence. The speed at which disturbance estimation can be done is dependent on the levels of measurement and process noise, and on the contribution of other unmodeled disturbances.

V. Simulation Results

A software-in-the-loop (SIL) framework has been created using ROS. This framework accurately simulates the ball & plate system using a non-linear model and by introducing process and measurement noise. The measurement noise level is set to $\sigma_v = [2.5, 2.5] \cdot 10e^{-4}$, and is approximated based on the squared resolution of the touch screen. Next, the process noise level is approximated at $\sigma_w = [0.01, 0.2, 0.01, 0.2]$, based on the servo resolution multiplied by the maximum ratio of servo angle to plate angle. The values for MPC that will be referred to as standard are defined as follows $Q = [5, 2, 0, 5, 2, 0]^T$, $R = [0.2, 0.2]^T$ according to the state and input vectors of (21).
Furthermore, SIL experiments have been conducted to provide insight in the behavior of the controllers and observers for different test cases. The difference in behavior is expressed using the following control performance metrics:

Settling time (ST) The time required for the response to reach a steady state, here defined with 2% tolerance bands.

Overshoot (PO) The maximum amount a system exceeds its final steady state value divided by its final value, here defined as a percentage.

Rise time (TR) The time for the response to rise from 10% to 90% of its final value.

Normalized mean square error (NMSE) The mean of the squared difference between a system's output and the reference value over a predefined period of time, normalized to unit for comparison.

A. MPC vs PID

The following paragraphs contain multiple test cases by which the performance differences of PID (III-E) and MPC (IV-A) is evaluated. To single out controller performance, other sources that may influence the performance metrics, such as artificial plant-model mismatches, noise, and observer effects, have been turned off. Also, for setpoint tracking, it is assumed that no future reference information is available, thus MPC preview is disabled.

1) *Stabilization: Control the ball to move from a random position on the plate to a stable equilibrium, here defined as the center of the plate, i.e.* $x_N = 0.0\text{m}$. Note that this experiment is designed in 1D (only in x_b), which is valid since the dynamic model (linearized using assumptions 5-7) exhibits symmetric and uncoupled behavior and results in an equal response for y_b.
In this experiment, a PD controller (zero integrator PID) will be compared with a default MPC (balanced penalty on position and velocity errors), and an aggressively tuned MPC (heavy penalty on the position error).
The experimental results in Figure 8 and Table I reveal that an aggressively tuned MPC regulator can compete with PID in terms of rise time (RT), but performs less in all other metrics. However, the balanced MPC has an advantage over PID when it comes to percentage overshoot (PO).

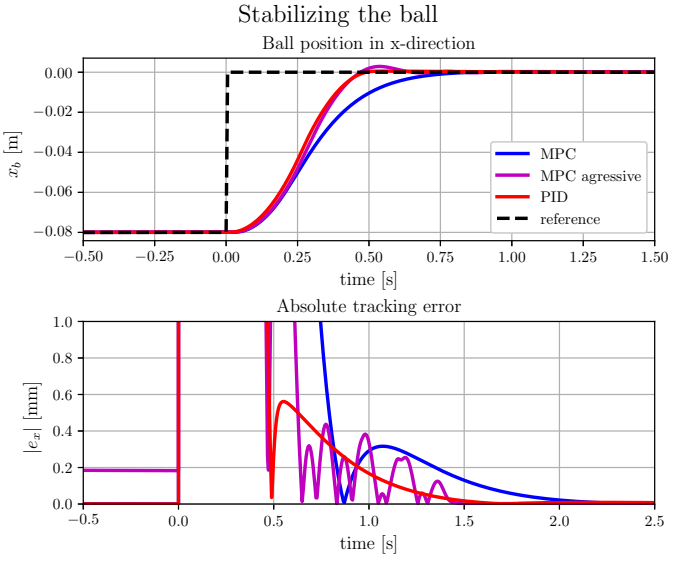


Fig. 8: Simulation results of balancing the ball on the plate controlled by MPC (blue), aggressively tuned MPC (magenta) and PID (red). Top figure: ball position, bottom figure: absolute tracking error (zoomed in).

TABLE I: Stabilizing the ball: performance metrics

	PID (SIL)	MPC (SIL)	aMPC (SIL)	PID (HIL)	MPC (HIL)
PO	0.70%	0.40%	3.63%	-	-
RT	0.28s	0.39s	0.27s	0.45s	0.38s
ST	0.46s	0.70s	0.59s	2.2s	1.8s
NMSE	0.527	0.596	1.000	-	-

2) *Setpoint tracking with unmeasured disturbances: The setpoint* x_N *must be maintained for* . In this experiment, the position of the ball is disturbed by a sudden angle offset in the first 0.1s of the experiment. In Figure 9 the aggressively tuned MPC is shown to overcompensate for the disturbance that has been applied. Furthermore, the PID is the fastest to reject the disturbance and to attenuate the error that occurs. The results of this case are consistent with previous results of setpoint tracking, where a disturbance is implicitly applied by a step in reference.

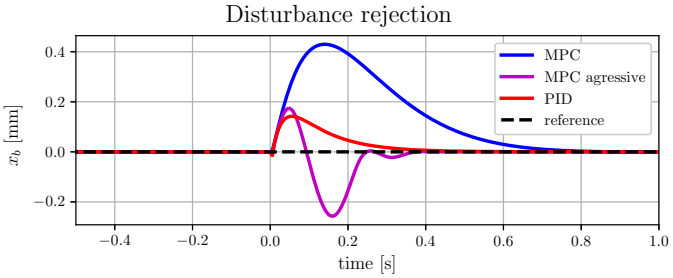


Fig. 9: Disturbance rejection simulation ($[\beta, \alpha] = 0.1\text{rad}$ for 0.1s) results of the ball & plate system controlled by MPC (blue), aggressively tuned MPC (magenta) and PID (red).

3) *Setpoint tracking with additional constraints: A setpoint must be reached while adhering to additional state constraints.*
In the default formulation, input and state constraints cannot be enforced using a PID controller. For this reason, the experiment was conducted using only MPC. The state constraints in this experiment are defined by $\underline{h} \leq x_b^2 + y_b^2 + s_{l,h}$, and $x_b^2 + y_b^2 + s_{u,h} \leq \bar{h}$, where $s_{l,h}$ and $s_{u,h}$ are slack variables corresponding to the lower and upper bound, respectively. The slack variables are introduced to soften the constraints (IV-A). This is necessary to maintain the feasibility of the optimization problem, even in case of a constraint violation. Moreover, the lower limit, denoted by $\underline{h}$, represents the squared radius of the circle, and is set to 0.05^2 (m). The upper limit is set to an arbitrary high value such that the upper bound does not cause any violations.
Figure 10 (right) shows that MPC is successful in controlling the ball to avoid the inadmissible states (i.e. the obstacle) while tracking a setpoint: the ball does not violate the position constraint during its trajectory. The figure also shows that the velocity is relatively low in the first part of the trajectory. The cautious behavior of the MPC regulator is a direct result of the high penalty on violating the constraint (high value for the slack

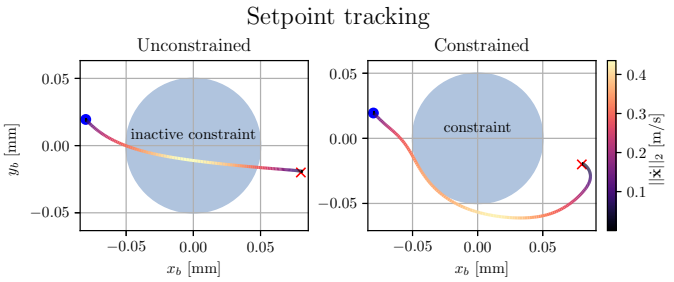


Fig. 10: Setpoint tracking with MPC in simulation: plot of x/y-plane with velocity in color gradient from x_0 (●) to a setpoint (×) under unconstrained and constrained conditions.

variable) and a short prediction horizon (it takes a while before the MPC can 'look' past the circular constraint). In the other case, when the constraints are removed, it is expected that the ball will take a straight path to the setpoint. Interestingly, as shown in Figure 10 (left), this is not the case. The start of the trajectory is perfectly diagonal but deviates from the shortest path between x_0 and the setpoint. This can be explained as follows. Any MPC action corresponds to an optimal solution, i.e. lowest cost, of the OCP. The cost is defined by the weighted sum of errors between the desired state of a node (reference) on the horizon and its actual state. When position errors dominate the total cost, and the cost contribution of the error in y_b (denoted by J_y) is approximately equal to that in x_b (J_x), the most optimal solution would lead to an equally fast attenuation of these errors. Hence, leading to a diagonal trajectory. Then, from $[x_b, y_b] = [-0.05, 0.0]$ onward, the contribution of J_y is less significant compared to J_x. The resulting optimal solution therefore mainly focuses on attenuating the error in x_b rather than the remaining error in y_b. The distribution of the error-penalizing weights contributes to the final shape of the MPC trajectory. Other parameters that influence the shape are the amount of nodes in the horizon, the length of the horizon in time and the time spacing between these nodes.

4) *Reference tracking: Application where $N_{horizon} = 20$ samples of the reference are available at every time instance, which is equal to a* 0.1s *preview.* The reference is based on a fifth-order polynomial [24]. This allows the first few temporal derivatives to be continuous (smooth). The smoothness of higher-order derivatives is desired to avoid exciting potential high-frequency modes of the Stewart platform. A different method uses a trapezoid velocity profile. This has the advantage of spending more time at a maximum velocity, thus decreasing the time to reach a setpoint. However, due to the discontinuity of the trajectory, this method has been rejected.
Both the MPC and the PID have access to the same position and velocity reference, denoted by $s_{ref,k}$ and $v_{ref,k}$. The MPC regulator utilizes the information of the reference of $N_{horizon} + 1$ samples, i.e. a reference preview from $k = 0$ to $k = N_{horizon}$. This information is used as a reference for each node in the cost function. The PID controller only acts on the current error (proportional term) and the previous error(s) (derivative and integral terms). Therefore, the PID controller as designed in III-E (without feedforward) uses only the trajectory information at $k = 0$.

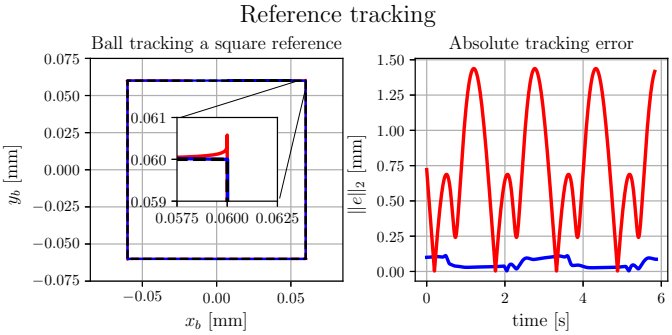


Fig. 11: Tracking a square-shaped reference (black dashed line) in simulation with MPC (blue line: –) and PID (red line: –).

The tracking performance of a square-shaped reference trajectory is depicted in Figure 11. The results reveal that MPC performs significantly better than PID in reference tracking: $\max(\|e\|_2) = 0.11$mm and 1.44mm, $\mu(\|e\|_2) = 0.06$mm and 0.73mm. A large contrast in tracking behavior can also be seen in the corners of the square, where MPC uses its preview to anticipate on the upcoming corner, the PID does not. This results in sharp cornering by the MPC and in an overshoot by the PID. This example showcases the advantage of MPC: using knowledge of the plant to optimally anticipate upcoming changes in the trajectory.

B. Offset-free MPC vs PID

In the following experiments, a plant model mismatch has deliberately been created by adding a static offset to the angles of the plate; see IV-C. The offset must be within the bounds of the input space, i.e. $d_\theta < \theta$, as the system would otherwise be uncontrollable. Furthermore, during the experiment the process and measurement noise have been activated. In the absence of process and measurement noise, the only disturbance imposed on the plant model would be the angle offset. A well-tuned observer can estimate the disturbance in a very short period of time. This would not represent the performance in a real-world application well, as measurement and process noise could not be neglected.

1) *Stabilize the ball to the center of the plate, subject to a static disturbance of $d_\beta = 0.1$ and $d_\alpha = -0.1$*rad. In this experiment, the offset-free MPC scheme uses a KF as a disturbance observer and state estimator. The PID controller is only provided with the KF state estimate. The results in Figure 12 show that both controllers are able to balance the ball to the center of the plate. The PID controller uses the integral term to attenuate the steady-state offset, caused by the static disturbance, and the

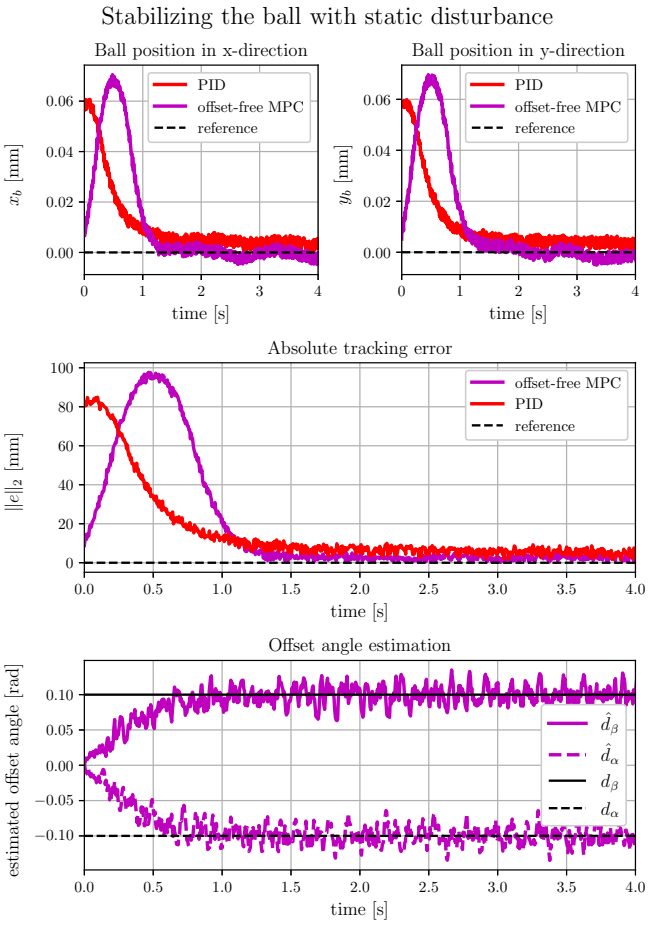


Fig. 12: Stabilizing the ball on the plate subject to a static disturbance of $d_\beta = 0.1$ and $d_\alpha = -0.1$rad. With a KF as disturbance observer in the offset-free MPC scheme and integrator action in the PID controller.

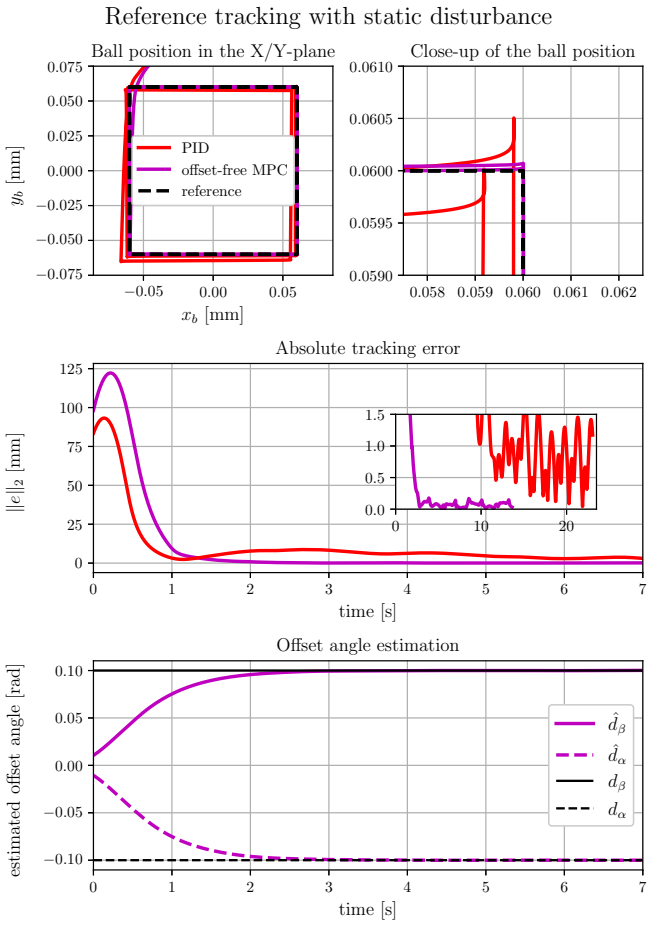


Fig. 13: Tracking a square-shaped reference (black dashed line) in simulation with offset-free MPC (magenta line: –) and PID (red line: –) with a static disturbance of $d_\beta = 0.1$ and $d_\alpha = -0.1$rad estimated by a Kalman filter.

offset-free MPC scheme does this by correcting for the estimated angle offset $\hat{d}_\theta$. The KF disturbance observer is able to correctly estimate the angle offset in ≈ 1.7s while being subject to process and measurement noise. In this particular setup, with PID values of $P = 15.2, I = 1, D = 4.2$, standard MPC values (=weights) and KF values of $R = [5e^{-6}, 5e^{-6}]$, $Q = [1e^{-7}, 1e^{-3}, 1e^{-7}, 1e^{-3}]$ for the internal state and $Q_d = [3e^{-4}, 3e^{-4}]$ for the disturbance, the PID is slower to converge to an offset-free state, i.e. to $\|e\|_2 = 0$.

2) *Reference tracking subject to a static disturbance of* $d_\beta = 0.1$ *and* $d_\alpha = -0.1$rad. The results of this experiment are depicted in Figure 13. In order to compare this experiment with that of Figure 11, no noise has been added to the closed-loop system. The results are, as expected, similar to the previous experiments since both PID and offset-free MPC manage to attenuate the steady state-offset. While it takes the PID several laps around the square to converge to the reference, the offset-free MPC scheme is quicker to converge, see upper right graph in Figure 13. Furthermore, after convergence, the absolute error profile of the offset-free MPC scheme is equal in magnitude to that of the MPC scheme without added disturbance, as depicted in magenta –, in Fig. 13) and in blue – in Fig. 11 respectively. And, although it takes some time to appear, the absolute error profile of the PID controllers are also equal in magnitude.

VI. Experimental results

Several challenges have been encountered during the process of performing experiments with hardware in the loop (HIL). These challenges did not occur during simulation and are therefore assigned to the sim-to-real gap. For this reason, this section starts with a summary of the measures taken to close this gap, after which the experimental results are presented.

A. Sim to real

One of the major differences of SIL vs HIL is that SIL can run slower than real-time: the plant can be simulated for any desired sample rate. However, the sample rate of HIL experiments is fixed at 5ms, see section III-A. This rate is

only feasible when the time between receiving a measurement and sending a control command is less than 5ms. From simulations, it is found that the software overhead (SO), i.e. message decoding/encoding, inverse kinematic calculations and internal data transfer, takes a maximum of 1ms. This leaves $\leq 4\text{ms}$ for the QP solver(s) to initialize the QP, converge to an optimum and output a solution.

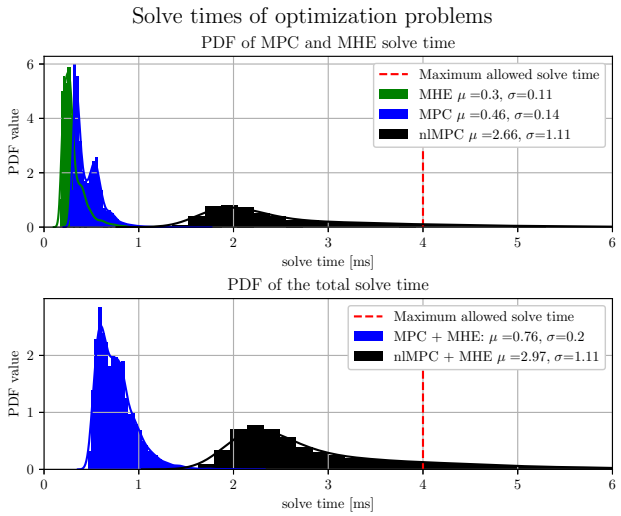


Fig. 14: Probability density function (PDF) values of the solve times of MPC, MHE and nonlinear MPC optimization (nlMPC) problems, based on 30s of data.

Figure 14 depicts the time it takes for a solver (HPIPM) to initialize and solve a QP, while the MPC is configured with $n = 6$, $m = 2$, $N = 20$ and the MHE with $n = 6$, $m = 4$, $L = 10$. In this experiment, the MPC and MHE are operating in closed-loop and are tasked with tracking a square reference. In the worst case scenario (maximum value of blue distribution), where two computational heavy components as a MPC and MHE are combined, a maximum total solve time of 2.2ms is measured. Since $|t_{SO}|_\infty + |t_{solve}|_\infty < 5\text{ms}$, a control rate of 200Hz can be maintained. Hence, the real-time requirement is met. However, when a nonlinear constraint is added to the MPC optimization problem, the QP solve times increase significantly, see solve time distribution marked in black. In this case, 15% of the iterations require $> 4\text{ms}$ to converge an optimum. This scenario is avoided by limiting the maximum amount of iterations and using the most optimal but feasible solution the solver has found until then. Furthermore, since the solve time varies, a varying delay is introduced in the control loop, i.e. the control action that belongs to state k is applied at time $k+\delta$, with $\delta \in (0, 4]\text{ms}$. This varying delay can lead to unpredictable behavior and is therefore undesired. To solve this, the $k+1$ input of the MPC sequence is applied, i.e. is sent to the Stewart platform, as soon as a new measurement is received. This way, a constant delay of 5ms is introduced, but accounted for by applying the input that is associated with the predicted state.

Next, during HIL experiments it became apparent that the measurements of the ball position contained faulty values. These values fitted within the boundaries of the plate, but did not match the actual position of the ball. The faulty measurements occurred more frequently during experiments with a high-gain PID controller than with the (cautious) MPC scheme. Based on this information, the source of the faulty measurements was determined to be in the decrease in pressure between the ball and plate during fast angular accelerations of the plate. Since it was not possible to bound the angular acceleration for every controller, it was chosen to reduce the impact of the faulty measurements on the state estimation. First, outlier measurements were filtered. This was done by marking measurements that are not within a multiple of the measurement variance ($2\sigma_y$), which is calculated over a window of 15 measurements. When an outlier measurement was marked, the state estimate is only conducted by the KF only performs the prediction step instead of also an update step. A similar technique is used for MHE by giving zero weight to that particular measurement in the horizon, thus replicating a prediction-only estimate.

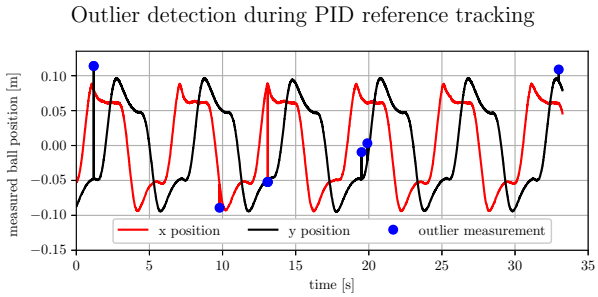


Fig. 15: Outlier detection during HIL reference tracking, with the PID controller operating on the KF-based estimated state.

The effect of detecting and rejecting outlier measurements is depicted in Figure 15. Since the ball trajectory remains its shape, it is shown that the faulty measurements do not significantly affect the estimated state, and therefore do not alter (worsen) the PID control commands.

B. MPC vs offset-free MPC

Stabilization: Control the ball to move to the center of the plate while being subject to a constant disturbance. In this experiment, the system is mounted on a seemingly level surface. However, in practice, only a slightly tilted surface is required for a ball to start rolling. For the ball-balancing Stewart platform, a surface that is tilted by $\approx 0.01\text{rad}$ in α and $\approx 0.007\text{rad}$ in β, results in an absolute steady state error of 10mm, see Figure 16. However, when the offset-free MPC scheme is implemented, the offset, caused by the tilted surface the system is mounted on, is mitigated. The graph for the absolute tracking error indicates that a steady state error of only 0.3mm is reached by the offset-free MPC scheme. This indicates the limitation of the standard MPC scheme, where the model error is not integrated, and the non-zero offset persists.

C. Offset-free MPC vs PID

In the following HIL experiments, the offset-free MPC scheme is compared with the PID controller.

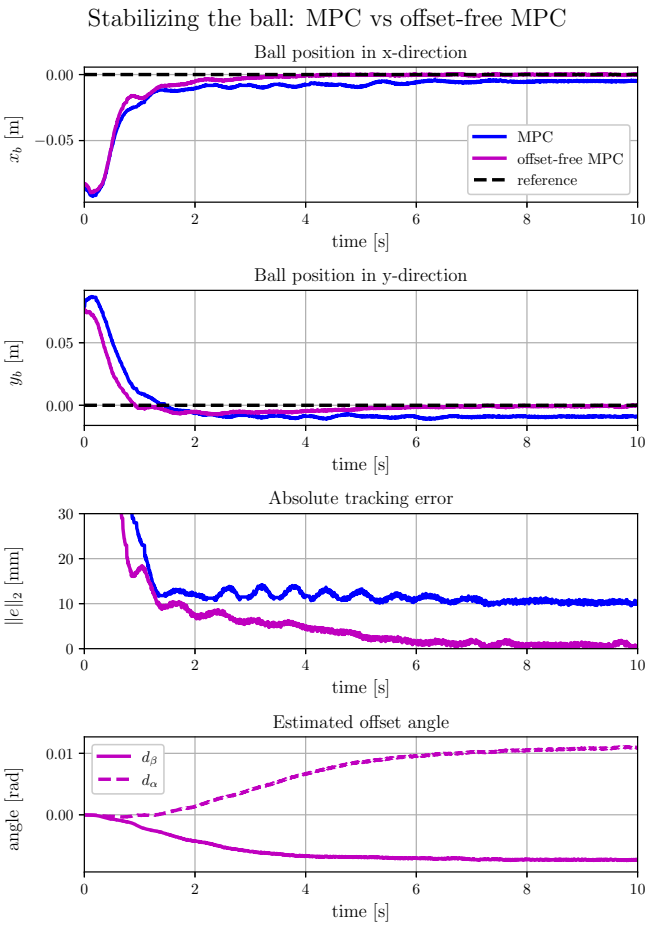


Fig. 16: Comparison of the offset-free MPC scheme (magenta line: –) with a KF as disturbance observer, and a standard MPC scheme (blue line: –) in stabilizing the ball on the plate while being subject to a minor static disturbance.

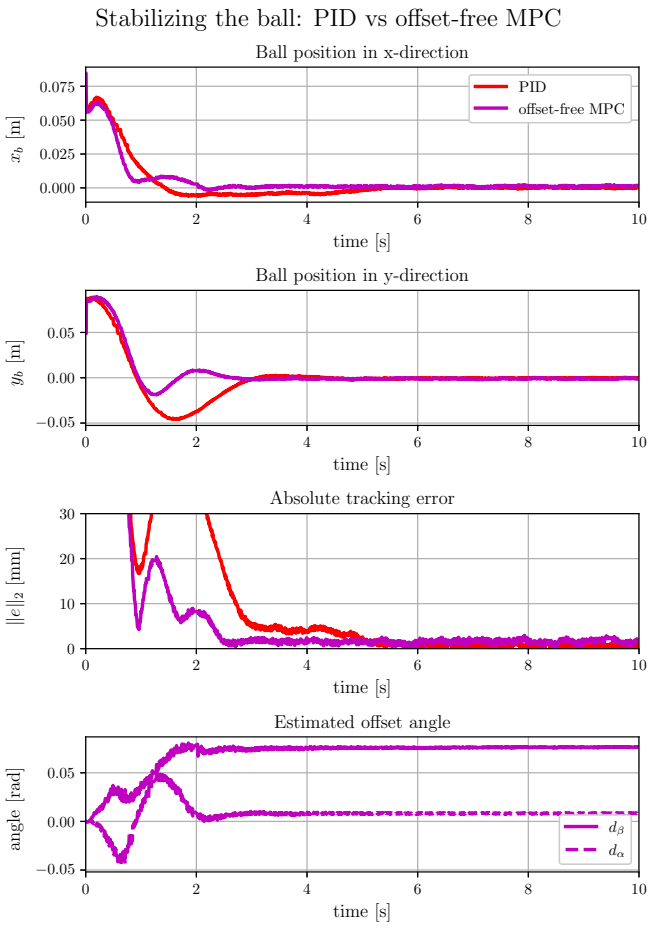


Fig. 17: Comparison of the offset-free MPC scheme (magenta line: –) with a KF as disturbance observer, and a PID controller (red line: –) in stabilizing the ball on the plate while being subject to a static disturbance.

1) *Stabilization: Control the ball to move to the center of the plate while being subject to a constant disturbance.* The previous HIL experiment indicated the issue a standard MPC scheme has with modeling errors, how this results in a steady-state offset and how this is solved using the offset-free scheme. The first experiment of this subsection focuses on the performance of the offset-free MPC scheme in comparison with the PID controller. Figure 17 shows that both the offset-free MPC scheme and the PID controller are able to mitigate the error. In the offset-free MPC scheme, it is notable that the KF disturbance is very fast in estimating the offset angle ($\approx$ 2s), which is displayed in the bottom graph. The MPC uses these values to update the QP and mitigate the error appropriately, resulting in a settling time of 2.5s. The PID controller is also able to bring the ball close to the center of the plate fast (3s-mark). However, the integrator term is slow to mitigate the remaining error in the x_b-direction, resulting in a settling time of $\approx$ 5s. In addition to mitigating the offset error, like PID does as well, the offset-free MPC scheme also quantifies the disturbance that induced the offset, which might be beneficial in some cases. Upon closer inspection of the absolute tracking error, the PID manages to obtain a lower error when compared to the offset-free MPC scheme, with a mean absolute error of 0.7mm and 1.5mm respectively. Furthermore, to decrease the variance of the disturbance estimate, while allowing for a fast initial convergence, the weight on the estimate is increased at $t = 1.5$ and $t = 3\text{s}$. With this method, the disturbance observer is required to have converged to a neighborhood of the real value at the first time step ($t = 1.5$) and to sufficiently accurately approximate the real value at the second time step. However, when the estimate is disturbed just before the second time step, for instance by the effect of unmodeled dynamics, a significant estimation error can occur. And, as the model considered in the offset-free MPC scheme does not take the dynamics of the Stewart platform and friction terms of the ball in account, this effect is likely to have occurred, causing the higher steady state offset of 1.5mm. In conclusion, the weights of the

disturbance observer can be chosen to benefit either the rate of convergence or the accuracy of the estimate. As the stabilization time had a higher priority in this experiment, the weights of the disturbance observer are tuned accordingly.

2) *Reference tracking subject to a static disturbance.* This experiment is similar to that depicted in Figures 11 and 13 where PID, MPC and offset-free MPC are compared. In terms of overall tracking offset, all controllers manage

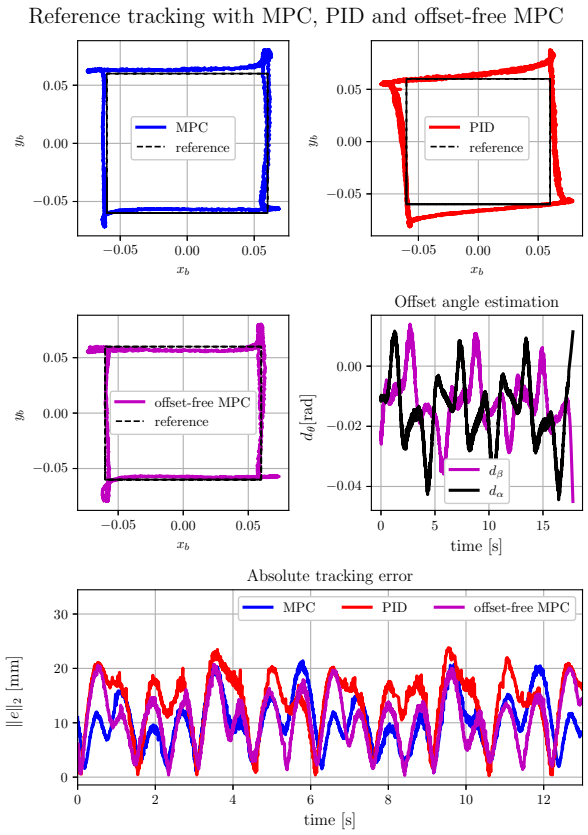


Fig. 18: Tracking a square-shaped reference counter-clockwise (black dashed line) in HIL, with MPC (blue line: –), PID (red line: –) and offset-free MPC (magenta line: –), and the disturbance estimated by a Kalman filter.

to track a square that is centered around the reference square, see Figure 18. This indicates that the demonstrator setup is positioned on a surface that is approximately level. Furthermore, the ball trajectory of the MPC and offset-free MPC is quite similar. However, the ball trajectory of the PID controller seems to be tilted. This likely to caused by the integrator term. Which builds up to compensate for the constant offset during the straight lines. And consequently causing an offset when the reference trajectory changes direction at a corner. Moreover, the poor tracking performance of PID is substantiated by its high mean absolute tracking error of 13.8mm, when compared to 9.9mm, and 10.5mm of MPC and offset-free MPC scheme consecutively.

The ability to anticipate on upcoming changes in reference gave MPC a tracking advantage over PID in simulation. This allowed the overshoot to be minimal, see Figure 11. In the HIL experiment however, the MPC has a significant overshoot in the corners. This is due to a plant-model mismatch through which a discrepancy in the MPC predictions and the actual behavior of the demonstrator setup is present. Moreover, the disturbance observer of the offset-free MPC scheme was not able to estimate the dynamic disturbances fast enough and compensate for this model discrepancy.

3) *Setpoint tracking with additional constraints, i.e. obstacle avoidance.* In this experiment, which is similar to that Figure 10, the MPC is tasked to track setpoint that are randomly generated in the admissible position state space (i.e. outside of the obstacle). This experiment fits in the offset-free MPC -category, as the disturbance observer has been used to estimate the disturbance (angle offset) beforehand. This disturbance is then added as a constant value to the standard MPC scheme. This way, the disturbance observer does not interfere with the MPC control actions, since this does not always benefit the performance as shown in the previous experiment.

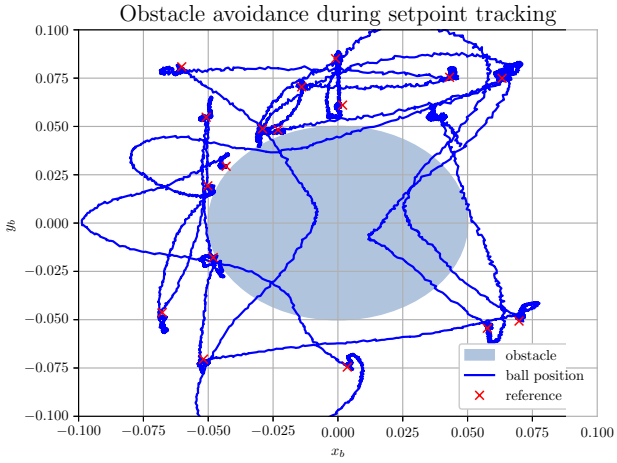


Fig. 19: Obstacle avoidance while moving between setpoints (red crosses: ×) using MPC.

The results of Figure 19 show that the position trajectory of the ball violates the constraint, i.e. the obstacle, in the instances where consecutive reference setpoints are positioned on opposite sides of the obstacle. This is explained by the same plant-model mismatch that affected the previous HIL experiments: The MPC model does not accurately represent the real system, as friction terms and servo dynamics remain unmodeled and only a part of the ball-and-plate dynamics are considered. This discrepancy causes the predictions of the MPC to differ from the actual behavior. In addition, the MPC only predicts 0.1s ahead in time. During this period of time, the initial MPC control action has caused the ball to move towards a position where it violates, or cannot steer away from violating the constraint.

VII. Conclusions and future work

In this work, we examined the potential of MPC for motion systems, using a ball-balancing Stewart platform as a demonstrator setup. A modular software framework is implemented to facilitate the comparison between MPC and PID on this setup. The framework interfaces with efficient optimization software that, in conjunction with a linearized and simplified MPC model, allows to achieve and consistently maintain a sampling rate of $200\mathrm{Hz}$, even in the case of MPC with a nonlinear constraint, combined with an MHE state observer. In SIL experiments, MPC outperformed PID for nonsmooth trajectories by more than a factor 10 decrease in both mean and maximum tracking error. The potential of MPC is also demonstrated by its ability to include constraints in its formulation, thereby avoiding obstacles in both SIL and HIL experiments. Furthermore, the steady-state offset of only $0.3\mathrm{mm}$ in a HIL stabilization experiment indicates that the offset-free MPC scheme is effective in estimating and compensating for the unmeasured static disturbances that have been applied during the experiment. Moreover, it is found that the speed of convergence of the disturbance observer, and therefore its ability to compensate for these disturbances, depends on the tuning of the weights and the level of non-white process noise, e.g. the dynamic disturbances originating from unmodeled dynamics. The offset-free MPC scheme was unable to compensate for these dynamic disturbances. This leaves room for quantifying the effect of model simplifications on dynamic disturbances while considering MPC solve times. In addition, a different system might be more optimal for examining the ability of MPC to handle MIMO systems, as the ball-and-plate dynamics are fully decoupled after linearization.

References

[1] A. Bemporad, D. Bernardini, R. Long, and J. Verdejo, "Model predictive control of turbocharged gasoline engines for mass production," 04 2018.

[2] S. D. Cairano, "An industry perspective on mpc in large volumes applications: Potential benefits and open challenges," *IFAC Proceedings Volumes*, vol. 45, no. 17, pp. 52–59, 2012. 4th IFAC Conference on Nonlinear Model Predictive Control.

[3] J. Folker, "Comparison of model predictive and classical control for a benchmark motion system with time delay." unpublished, 2022.

[4] L. M. Zarzycki K., "Fast real-time model predictive control for a ball-on-plate process," *Sensors*, vol. 21, 2021.

[5] F. D. Cero, "Ball and plate mpc control of a 6 dof stewart platform," 2022.

[6] M. Bianchi, A. van der Maas, E. Maljaars, and W. Heemels, "Offset-free mpc for resource sharing on a nonlinear scara robot∗∗this work is part of the research programme chameleon "hybrid solutions for cost-aware high-performance motion control" with project number 13896, which is (partly) financed by the netherlands organisation for scientific research (nwo).," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 265–272, 2018. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.

[7] G. Pannocchia, M. Gabiccini, and A. Artoni, "Offset-free mpc explained: novelties, subtleties, and applications," *IFAC-PapersOnLine*, vol. 48, no. 23, pp. 342–351, 2015. 5th IFAC Conference on Nonlinear Model Predictive Control NMPC 2015.

[8] H. Bang and Y. Lee, "Implementation of a ball and plate control system using sliding mode control," *IEEE Access*, vol. PP, pp. 1–1, 05 2018.

[9] A. M. Lopes, "Fast dynamic model of a moving-base 6-dof parallel manipulator," 2010.

[10] Z. Bingul and O. Karahan, *Dynamic Modeling and Simulation of Stewart Platform*. 03 2012.

[11] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system."

[12] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, Oct 2021.

[13] G. Frison, D. Kouzoupis, T. Sartor, A. Zanelli, and M. Diehl, "Blasfeo: Basic linear algebra subroutines for embedded optimization," *ACM Trans. Math. Softw.*, vol. 44, jul 2018.

[14] X. Huang, Q. Liao, S. Wei, X. Qiang, and S. Huang, "Forward kinematics of the 6-6 stewart platform with planar base and platform using algebraic elimination," pp. 2655 – 2659, 09 2007.

[15] H. Sadjadian and H. Taghirad, "Comparison of different methods for computing the forward kinematics of a redundant parallel manipulator," *Journal of Intelligent and Robotic Systems*, vol. 44, pp. 225–246, 11 2005.

[16] M. Raghavan, "The Stewart Platform of General Geometry Has 40 Configurations," *Journal of Mechanical Design*, vol. 115, pp. 277–282, 06 1993.

[17] H. Schaub and J. L. Junkins, *Analytical Mechanics of Space Systems - Appendix B: Various Euler Angle Transformations*. American Institute of Aeronautics and Astronautics, 2018.

[18] L. Spacek, V. Bobál, and J. Vojtesek, "Digital control of ball & plate model using lq controller," *2017 21st International Conference on Process Control (PC)*, pp. 36–41, 2017.

[19] Z. Fei, Q. Xiaolong, L. Xiaoli, and W. Shangjun, "Modeling and pid neural network research for the ball and plate system," in *2011 International Conference on Electronics, Communications and Control (ICECC)*, pp. 331–334, 2011.

[20] J. Yang, T. Meijer, V. Dolk, B. d. Jager, and W. Heemels, "A system-theoretic approach to construct a banded null basis to efficiently solve mpc-based qp problems," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 1410–1415, 2019.

[21] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[22] D. G. Robertson, J. H. Lee, and J. B. Rawlings, "A moving horizon-based approach for least-squares estimation," *AIChE Journal*, vol. 42, no. 8, pp. 2209–2224, 1996.

[23] G. Pannocchia and J. B. Rawlings, "Disturbance models for offset-free model-predictive control," *AIChE Journal*, vol. 49, no. 2, pp. 426–437, 2003.

[24] P. Corke, *Robotics, Vision and Control - Fundamental Algorithms in MATLAB®*, vol. 118 of *Springer Tracts in Advanced Robotics*, pp. 70–73. Springer, 2011.

[25] P. Singla, D. Mortari, and J. Junkins, "How to avoid singularity when using euler angles?," *Advances in the Astronautical Sciences*, vol. 119, pp. 1409–1426, 01 2005.

APPENDIX

A. Derivation of the rotation matrix using Euler angles

The plate's rotations over z^p, y^p and x^p is described by an intrinsic 1-2-3 Euler angle set. This set is denoted by the Euler/Tait-Bryan angles α, β and γ, respectively. The Tait-Bryan angles form a minimum parametrization of a rotation. This introduces geometric/mathematical singularities which implies that two axes of the $\mathbb{R}^3$ frames are aligned, i.e. $\mathcal{O}_{b,i} = \mathcal{O}_{p,i}$, to create a $\mathbb{R}^2$ frame that is incapable of describing rotations in $\mathbb{R}^3$. Note that this does *not* reflect any physical limitations of system. Asymmetric Euler angle sets, such as the 1-2-3 set, introduce geometric/mathematical singularities at $\beta = \pm\frac{\pi}{2}$ [25]. This angle cannot physically be attained by the system, which renders the 1-2-3 Euler angle set a safe choice.

The rotation matrix is defined to describe an intrinsic rotation, i.e. rotations about the axes of the body-fixed axes. In this case, intrinsic to O^p: the coordinate system of the plate. The rotation matrix $\boldsymbol{R}_p^b$, previously given in (3), is obtained by multiplying the right hand frame rotation matrices for roll, pitch and yaw:

$$\boldsymbol{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\theta & -s\theta \\ 0 & s\theta & c\theta \end{bmatrix}$$

$$\boldsymbol{R}_y(\theta) = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix}$$

$$\boldsymbol{R}_z(\theta) = \begin{bmatrix} c\theta & -s\theta & 0 \\ s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and according to the 1-2-3 Euler angle set:

$$\boldsymbol{R}_p^b = R_z(\gamma)R_y(\beta)R_x(\alpha) \tag{24}$$

B. Derivation of the state equations from the Lagrange-energy equations

The ball-and-plate system has been expressed in terms of kinetic and potential energy, see equations (11), (12) and (13) of section III-D. On this basis, the state equations can be derived by formulating the energy equations as denoted in (7).

Starting with the partial derivative of the kinetic energy w.r.t. the Euler angle rate of the plate:

$$\frac{\partial T}{\partial \dot{\boldsymbol{\theta}}^p_{plate}} = \frac{\partial (T_p + T_b)}{\partial \dot{\boldsymbol{\theta}}^p_{plate}} = \left[\frac{\partial T}{\partial \dot{\alpha}^p_{plate}}, \frac{\partial T}{\partial \dot{\beta}^p_{plate}}, \frac{\partial T}{\partial \dot{\gamma}^p_{plate}} \right] \tag{25}$$

Note that the dimensions of T and $\dot{\boldsymbol{\theta}}$ determine the dimension of the resulting vector.
The same can be done when taking the partial derivative of the kinetic energy w.r.t. the velocity of the ball in the plate coordinate system:

$$\frac{\partial T}{\partial \dot{\boldsymbol{p}}^p_{ball}} = \frac{\partial (T_p + T_b)}{\partial \dot{\boldsymbol{p}}^p_{ball}} = \left[\frac{\partial T}{\partial \dot{x}^p_b}, \frac{\partial T}{\partial \dot{y}^p_b}, \frac{\partial T}{\partial \dot{z}^p_b} \right] \tag{26}$$

Next, the time derivative of (25) and (26) is denoted by

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{\boldsymbol{q}}} = \left[\frac{d}{dt}\frac{\partial T}{\partial \dot{\boldsymbol{\theta}}^p_{ball}}, \frac{d}{dt}\frac{\partial T}{\partial \dot{\boldsymbol{p}}^p_{ball}} \right] \tag{27}$$

Furthermore, the *Lagrangian* is defined as $L = V - T$, this results in

$$\frac{\partial L}{\partial \boldsymbol{\theta}^p_{plate}} = \left[\frac{\partial L}{\partial \alpha}, \frac{\partial L}{\partial \beta}, \frac{\partial L}{\partial \gamma} \right] \tag{28}$$

The objective is to control the position of the ball through actuation of the plate angle, rather than with force actuation. The plate angles are controlled with servo motors, which have their own internal setpoint controller. For this reason, it is unnecessary to include the plate dynamics of (28) in the model of the plant.

The partial derivatives of the Lagrangian w.r.t. the position vector of the ball is

$$\frac{\partial L}{\partial x_b} = m_b(g\sin\beta - \dot{\beta}^2 x_b - \dot{\alpha}\dot{\beta}y_b)$$
$$\frac{\partial L}{\partial y_b} = -m_b(g\cos\beta\sin\alpha - \dot{\alpha}\dot{\beta} + \dot{\alpha}^2 y_b).$$

Next, (7) is used with $L = V - T$ to express the state functions of the acceleration of the ball explicitly:

$$\ddot{x}_b = \frac{5}{7}\left(\frac{2}{5}\ddot{\beta}r + \dot{\beta}^2 x_b - \dot{\alpha}\dot{\beta}y_b - g\sin\beta\right)$$
$$\ddot{y}_b = \frac{5}{7}\left(\frac{2}{5}\ddot{\alpha}r + \dot{\alpha}^2 y_b - \dot{\alpha}\dot{\beta}x_b + g\cos\beta\sin\alpha\right)$$

C. Inverse kinematics of the Stewart platform

The inverse kinematic mapping of the Stewart platform is denoted by the mapping from the pose of the plate in $\mathcal{O}_b$, to the servo angles, i.e. by.

$$\boldsymbol{q} \xleftarrow[\text{kinematics}]{\text{inverse}} \boldsymbol{T}^b_p. \tag{31}$$

The following mapping is inspired by [8], which considers a simplified modeling approximation of the Stewart platform by assuming small operating angles over α and β, no rotation over z_p ($\gamma = 0$) and by neglecting centrifugal forces. Whereas, the position vector of the joints attached to the plate w.r.t. $\mathcal{O}_b$ can be expressed as

$$\boldsymbol{p}^b_i = \boldsymbol{T}^b_p \boldsymbol{p}^p_i \tag{32}$$

and the length of the virtual legs, denoted by $\boldsymbol{l}_i$, see also Fig. 20b, can be expressed as

$$\boldsymbol{l}^b_{p(i)} = \boldsymbol{p}^b_i - \boldsymbol{b}^b_i \tag{33}$$

where $i \in \mathbb{N}^*_{\leqslant 6}$ denotes the index of a part in Eqs. (32, 33). Here,

- $\boldsymbol{b}^b_i = [r_b\cos(r_i), r_b\sin(r_i), 0, 1]^T$ the position vector of the rotation center of each servo motor, with r_b the radius of the circle shown in Fig. 20c, r_i is the angle between the x-axis of $\mathcal{O}_b$ and the line that crosses the center line of the servo.

- $\boldsymbol{p}_i^p = [r_p \cos(r_i^p), r_p \sin(r_i^p), 0, 1]^T$ the position vector of the joints attached to the plate w.r.t. $\mathcal{O}_p$, with r_p the distance from a joint to the origin of the plate and r_i^p the angle of the joints w.r.t. the x-axis of $\mathcal{O}_p$, see Fig. 20d.

The angles used in this project are shown in Table II.

TABLE II: Actuator and joint mounting orientation angles in degrees.

Servo	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$
r_i	120	60	0	300	240	180
r_i^p	135	45	15	285	255	165
r_{ti}	90	90	330	330	210	210

Next, in order to derive the servo angle q_i for a certain plate pose the geometric representation in Figure 21 is used.

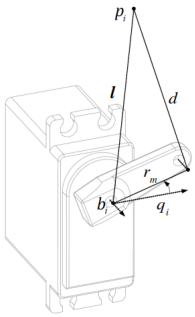


Fig. 21: Representation of the virtual leg length $|\boldsymbol{l}_i|$ to servo angle q_i mapping problem.

The position vector of a joint attached to the servo arm is $\boldsymbol{m}_i(q_i) = \boldsymbol{b}_i + \boldsymbol{R}_z(r_t)\boldsymbol{R}_y(-q_i) \cdot [r_m, 0, 0]^T$, where r_m is the length of the servo arm, $\boldsymbol{R}_y$ and $\boldsymbol{R}_z$ are the right-hand frame rotation matrices for y and z. Then, using the explicit expressions of the rotation matrices, $\boldsymbol{m}_i$ can be written as:

$$\boldsymbol{m}_i(q_i) = \begin{bmatrix} -r_m \cos q_i \sin r_{ti} \\ r_m \cos q_i \cos r_{ti} \\ r_m \sin q_i \end{bmatrix} + \boldsymbol{b}_i^b \tag{34}$$

The illustration in Fig. 21 helps to derive the following relations

$$\begin{aligned} \boldsymbol{r}_m^2 &= (\boldsymbol{m}_i(q_i) - \boldsymbol{b}_i)^T(\boldsymbol{m}_i(q_i) - \boldsymbol{b}_i), \\ d^2 &= (\boldsymbol{p}_i - \boldsymbol{m}_i(q_i))^T(\boldsymbol{p}_i - \boldsymbol{m}_i(q_i)), \\ |\boldsymbol{l}_i|^2 &= (\boldsymbol{p}_i - \boldsymbol{b}_i)^T(\boldsymbol{p}_i - \boldsymbol{b}_i). \end{aligned} \tag{35}$$

Where d is the (non-virtual) length of the leg. Then, by substituting (35) into (34), an implicit expression of q_i can be reached. This can be rewritten using the following trigonometric identity:

$$c_i = a_i \sin q_i + b_i \cos q_i \tag{36}$$

Where

$$\begin{aligned} a_i &= -l_z, \\ b_i &= l_x \sin r_{ti} - l_y \cos r_{ti}, \\ c_i &= \frac{\|d\|^2 - \|\boldsymbol{l}\|^2 - r_m^2}{2r_m}, \end{aligned}$$

with r_{ti} being the angle at which a servo motor (and arm) is mounted on the base, d the length of a leg connecting the servo arm and plate, r_m the length of a servo arm, $\boldsymbol{l} = [l_x, l_y, l_z]^T$ the distance between the rotation axis of the servo ($\boldsymbol{b}_i$) and the attachment joint on the plate ($\boldsymbol{p}_i$), i.e. the virtual actuator length.

Finally, the explicit expression of the servo angle for each of the servo motors is obtained by rewriting the implicit expression of (36). Since the servo-arms, and their respective operating range are mirrored for even- and odd-numbered servos (see Fig. 20b), different equations for the servo angle $\boldsymbol{q}$ are derived.

$$q^{odd} = arcsin\left(\frac{c_i}{\sqrt{a_i^2 + b_i^2}}\right) - arctan2(b_i, a_i) \tag{37a}$$

$$q^{even} = \pi - arcsin\left(\frac{c_i}{\sqrt{a_i^2 + b_i^2}}\right) - arctan2(b_i, a_i) \tag{37b}$$

Note that the 2-argument arc-tangent function uses the signs of a_i and b_i to place the servo joint in the right quadrant of the Euclidean space.

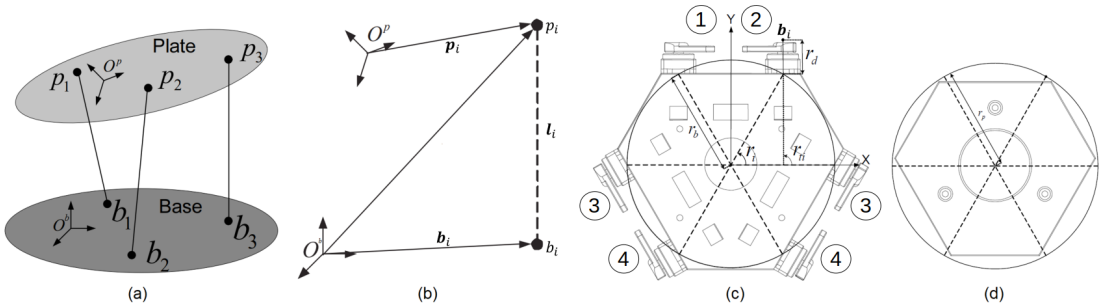


Fig. 20: Schematic illustration of a Stewart platform. (a) shows the different coordinate frames of the plate and the base. (b) shows the relationship of the vectors, (c) shows a top view of the base with vector $\boldsymbol{b}_i$ and the corresponding angles, (d) shows a top view of the plate with the joint connections $\boldsymbol{p}_i$, adapted from [8].