Table Tennis Playing Robots!?

*Final project report for 16745: Optimal Control and Reinforcement Learning

Mohit Vikas Javale
Dept. of Mechanical Engineering
Carnegie Mellon University

Atharva Sunder Ramdas
Dept. of Mechanical Engineering
Carnegie Mellon University

***Abstract*—This work presents an optimal control formulation for robotic table tennis, focusing on returning a ball with given initial conditions to a desired location on the table within a specified time. The system models the hybrid dynamics of the ball, including aerodynamic forces, table contact dynamics, and impact with a racket mounted on a 6-DOF robotic manipulator. A custom simulator was used to capture these interactions and validate the physical realism of the resulting trajectories. The control strategy separates the ball and manipulator trajectory optimization. Results demonstrate that the robot adapts its racket orientation in response to varying spin conditions, mimicking real-world playing techniques. This framework paves the way for future developments in robotic table tennis, such as fully autonomous rallies and real-time planning. Code and video demo for this work can be found in this repository.**

***Index Terms*—Trajectory Optimization, Hybrid System**

I. Introduction

A. Motivation

Table tennis is a fast-paced, high-dexterity task that is easy to learn but difficult to master—even for humans. A major challenge arises from the ball's lightweight and smooth design, which makes it highly susceptible to aerodynamic forces. These effects, primarily accentuated by heavy spin, significantly influence the ball's motion and make accurate placement of the ball difficult. Motivated by this challenge, this work explores what an optimal table tennis stroke looks like, with the possibility of improving human technique while playing the game.

B. Related work

Despite being a relatively niche task, modeling and control for table tennis has long attracted substantial interest due to its real-time decision-making demands and the added complexity of extremely fast, contact-rich dynamics, making it a valuable precursor for many other problems in robotics. Foundational work to accurately simulating and planning over a table tennis setup involves figuring out appropriate models for the ball dynamics and it's various interactions with the table and the racket. Work by Matsumoto et al. in [5] goes into a comprehensive derivation of a rebound dynamics model for a rigid ball, while also capturing the frictional and elastic phenomena during impacts. Their formulation is critical for simulating realistic ball trajectories in robotic table tennis and serves as the basis for many further works.

For control and motion planning in robotic table tennis, the work by Müller et al. titled "Optimal Trajectory Generation and Learning Control for Robot Table Tennis" [4] formulates table tennis as an optimal control problem, and suggests two strategies: 'focused' player and 'defensive' player as two tractable ways to solve the problem. A seminal dissertation by the same author with the same title dives further into more details regarding real-time deployment of hardware, and also looks at learning based approaches.

More recently, with the rise of deep reinforcement learning for robotics control tasks, there are several works from Google Research such as [1], [3], [2], that study the use and effectiveness of such methods for playing table tennis using a robot. These works showcase systems that learn highly dynamic behaviors directly from interaction, often using model-free or model-based RL combined with sim-to-real transfer. These approaches also highlight the potential of RL to learn control strategies in high-dimensional, discontinuous environments, in comparison to the usual control theoretic approaches traditionally used.

C. Approach

The work done in [4] approached the problem by discretizing a predicted ball trajectory based on the initial conditions of the ball, and then solving for the desired racket state at each of these discrete points. While this can be done in a fast manner using gradient-based solvers and warm starting, our approach prefers not to perform such repetitive optimization over a single trajectory, but rather adds the trajectory prediction problem as part of the optimization problem itself.

The contributions of this work include the following.

- Creation of a custom simulator capable of emulating complex table-tennis interactions.
- Formulating the problem of playing table tennis as a single trajectory prediction-optimization problem in a hybrid-system free-time setup to return the ball to a desired location at a desired time.

The remainder of this paper is organized as follows. Section II talks about dynamic models for the entire setup, Section III details the optimization problem formulations, Section IV consists of a qualitative comparison of obtained dynamic behaviors with expected behaviors and a discussion on different challenges faced during implementation, and Section V concludes the paper with comments on possible future work.

II. Dynamic Models

A. Coordinate Frame Setup

The problem is set up using two coordinate frames, one stationary world frame at the center of the table, and the other attached to the racket, which translates and rotates with the racket, as shown in Figure 1(the net is removed in this figure for ease of visualization). The racket coordinate frame's y-axis points towards the manipulator arm. The manipulator is placed on the negative x-axis.

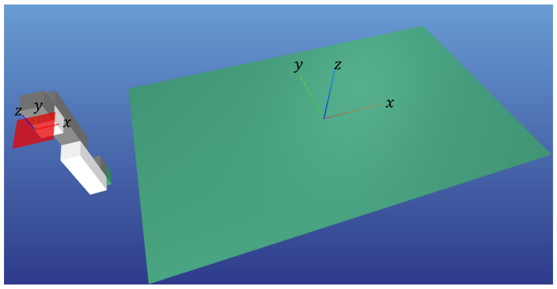

Fig. 1. Coordinate frame setup

B. Ball Flight Dynamics

A table tennis ball weighs 2.7 grams and can reach speeds of 25 m/s while spinning at over 1000 rpm during a game. The light weight and high speed of table tennis balls make non-linear aerodynamic effects on them significant. The dynamics of the ball in flight phase are modeled as,

$$\ddot{\mathbf{b}} = \mathbf{g} - C_D v_b \dot{\mathbf{b}} + C_L \boldsymbol{\omega}_\mathbf{b} \times \dot{\mathbf{b}} \tag{1}$$

where $\dot{\mathbf{b}}$ (written later as $\mathbf{v}_b$) is the ball's linear velocity, $v_b = \|\dot{\mathbf{b}}\|_2$ is the ball's speed, and $\boldsymbol{\omega}_b$ is the ball's angular velocity. The model accounts for gravity ($\mathbf{g}$), air drag (parameterized by the Drag Coefficient C_D) that opposes linear velocity, and the Magnus Effect (parameterized by the Lift Coefficient C_L) which generates a force on the ball due to the difference in airflow velocities across its surface when it is spinning. The ball's angular velocity is assumed to stay constant when in flight, and it is assumed that the ball's states are estimated accurately.

C. Table Contact Reset Map

Contact with the table affects the ball's linear and angular velocity due to friction. Since a table tennis table's surface is pretty smooth just like the ball, contact with the table is a sliding contact, i.e. the point of contact between the ball and the table has a non-zero velocity at the instant of impact. In addition, the rebound of the ball is not perfectly elastic. Accounting for these desired behaviors, the table contact reset map is chosen as,

$$\mathbf{v}'_\mathbf{b} = \mathbf{A_v v_b} + \mathbf{B_v}\boldsymbol{\omega}_\mathbf{b} \tag{2}$$

$$\boldsymbol{\omega}'_\mathbf{b} = \mathbf{A}_\omega \mathbf{v_b} + \mathbf{B}_\omega \boldsymbol{\omega}_\mathbf{b} \tag{3}$$

for a ball with incoming velocities $\mathbf{v_b}$, $\boldsymbol{\omega}_\mathbf{b}$ and outgoing velocities $\mathbf{v}'_\mathbf{b}$ and $\boldsymbol{\omega}'_\mathbf{b}$. Matrices $\mathbf{A_v}$, $\mathbf{B_v}$, $\mathbf{A}_\omega$ and $\mathbf{B}_\omega$ represent the directions of the impulse provided to the ball by the table, with friction affecting velocities in the x-y plane and a normal force pushing the ball in the z-direction with coefficient of restitution e_t. The term α is a measure of change in velocity due to frictional impulses. Mathematically, these terms are given by,

$$\mathbf{A_v} := \begin{bmatrix} 1-\alpha & 0 & 0 \\ 0 & 1-\alpha & 0 \\ 0 & 0 & -e_t \end{bmatrix}, \tag{4}$$

$$\mathbf{B_v} := \begin{bmatrix} 0 & \alpha r & 0 \\ -\alpha r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5}$$

$$\mathbf{A}_\omega := \begin{bmatrix} 0 & -\frac{3\alpha}{2r} & 0 \\ \frac{3\alpha}{2r} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{6}$$

$$\mathbf{B}_\omega := \begin{bmatrix} 1-\frac{3\alpha}{2} & 0 & 0 \\ 0 & 1-\frac{3\alpha}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

$$\alpha := \mu(1+e_t)\frac{|\mathbf{v}_{b,z}|}{\|\mathbf{v}_{b,T}\|} \tag{8}$$

It can be observed from these equations that friction between the ball and the table causes generation of a spin on the ball after bouncing even if the ball initially has no spin.

D. Racket Contact Reset Map

The racket contact reset map is modeled in a similar way as the table-contact reset map, just transformed to be expressed in the racket's moving frame.

Let the Rotation Matrix from the world frame to the racket frame be $\mathbf{R}$. Converting the input velocities to the racket frame,

$$\mathbf{v}'_\mathbf{b} = \mathbf{A_v R}^\top(\mathbf{v_b} - \mathbf{v_r}) + \mathbf{R}^\top \mathbf{B_v}\boldsymbol{\omega}_\mathbf{b} \tag{9}$$

$$\boldsymbol{\omega}'_\mathbf{b} = \mathbf{A}_\omega \mathbf{R}^\top(\mathbf{v_b} - \mathbf{v_r}) + \mathbf{B}_\omega \mathbf{R}^\top \boldsymbol{\omega}_\mathbf{b} \tag{10}$$

Similarly, converting the output velocities back to the world frame,

$$\mathbf{v}'_\mathbf{b} = \mathbf{R}\left[\mathbf{A_v R}^\top(\mathbf{v_b} - \mathbf{v_r}) + \mathbf{R}^\top \mathbf{B_v}\boldsymbol{\omega}_\mathbf{b}\right] + \mathbf{v_r} \tag{11}$$

$$\boldsymbol{\omega}'_\mathbf{b} = \mathbf{R}\left[\mathbf{A}_\omega \mathbf{R}^\top(\mathbf{v_b} - \mathbf{v_r}) + \mathbf{B}_\omega \mathbf{R}^\top \boldsymbol{\omega}_\mathbf{b}\right] + \mathbf{v_r} \tag{12}$$

where,

$$\mathbf{A_v} := \begin{bmatrix} 1-k_{pv} & 0 & 0 \\ 0 & 1-k_{pv} & 0 \\ 0 & 0 & -e_r \end{bmatrix}, \tag{13}$$

$$\mathbf{B_v} := k_{pv} \begin{bmatrix} 0 & r & 0 \\ -r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{14}$$

$$\mathbf{A}_{\boldsymbol{\omega}} := k_{p\omega} 1 \begin{bmatrix} 0 & -r & 0 \\ r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{15}$$

$$\mathbf{B}_{\boldsymbol{\omega}} := \begin{bmatrix} 1-k_{p\omega}r^2 & 0 & 0 \\ 0 & 1-k_{p\omega}r^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{16}$$

$$k_{pv} := \frac{k_p}{m}, \quad k_{p\omega} := \frac{k_p}{I} \tag{17}$$

In the above equations, k_p is a physical parameter that related the impulse along the racket plane based on the ball's tangential velocity. The radius, mass, moment of inertia of the ball are represented by r, m, I respectively.

III. Implementation Details

A. Architecture

Since the ball is much lighter than the manipulator, the velocity imparted to the manipulator links and the racket (which is the manipulator end effector here) due to momentum transfer from the ball is negligible. Therefore, the ball and manipulator trajectory are solved separately, and this still creates a realistic simulation of the impact between the ball and the manipulator.

The ball trajectory optimization solves for the trajectory of the ball in a hybrid fashion with a pre-defined contact mode sequence: a flight phase (until n_1 timestep), contact with the table (at n_1 timestep), another flight phase (n_1 to n_2 timestep), contact with the racket (at n_2 timestep), a flight phase (n_2 to n_3 timestep), and a final contact with table (at n_3 timestep). The contact phases create a rebounding effect on the ball's trajectory, essentially making the ball appear to bounce on a table or be hit by a racket, as shown in Figure 2. The optimizer also solves for the position, orientation and velocity of the racket at impact.

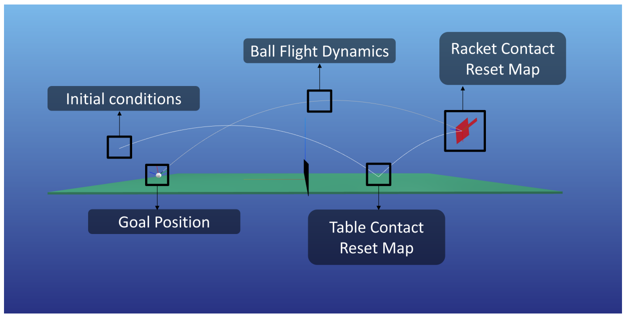

Fig. 2. Contact modes (racket rendered at desired orientation)

The manipulator trajectory optimization plans a smooth trajectory to reach the desired racket state (position, orientation and velocity) in the same time the ball takes to reach that state, as shown in Figure 3. The net effect of this architecture is that the ball is hit by the manipulator arm and is returned to the other side of the table while the optimizations are solved separately.

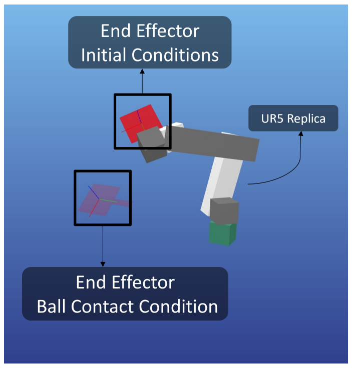
End Effector Initial Conditions
UR5 Replica
End Effector Ball Contact Condition

Fig. 3. Racket goal states obtained from ball trajectory optimization

B. Ball Trajectory Optimization

$$\begin{aligned} \min_{\mathbf{X_b},\mathbf{h},\mathbf{x_r}} \quad & 10||\mathbf{r_b^N} - \mathbf{r_g}||^2 + 10(\sum_{i=n_2}^{n_3} h^i - t_g)^2 + \mathbf{x_r}^T\mathbf{x_r} + \mathbf{h}^T\mathbf{h} \\ \text{s.t.} \quad & \mathbf{x_b^0} - \mathbf{x_{IC}} = \mathbf{0} \\ & \mathbf{x_b^{k+1}} = f_k(\mathbf{x_b^k}, \mathbf{h^k}) \\ & ||\mathbf{r_b^{n_2}} - \mathbf{r_m}||_2 \leq 0.6 \\ & \mathbf{h} \geq \mathbf{0} \end{aligned}$$

1) *Free-Time Formulation:* The problem is formulated as a free-time problem, allowing the solver to solve for step lengths as well, in turn allowing the solver to adjust the time of contact. For hybrid transitions, constraints are enforced to ensure corresponding guard sets are reached at the same timesteps as when reset maps are triggered to ensure the free time formation accommodates natural ball dynamics appropriately.

2) *Decision Variables:* The decision variables that are solved for are the ball states ($\mathbf{X_m} \in \mathbb{R}^{12\times N}$) and step lengths ($\mathbf{h} \in \mathbb{R}^N$) at ever time step, and the racket state ($\mathbf{x_r} \in \mathbb{R}^{12}$) at the strike time. The ball and racket states are a 12 dimension vector, containing the corresponding position (r), orientation (as euler angles), linear velocities and angular velocities.

3) *Objective Function:* The objective function consists of 4 distinct terms. The first two promote the solver to achieve desired goal states: reach the desired location and at the desired time, which the remaining two are regularization terms promoting lower racket velocities and smaller and equally distributed step lengths. Note that adding the desired goal state as an objective and not as a constraints was an active design decision since in extreme cases, the desired goal conditions might not be physically feasible, but the solver shall still converge to the closest feasible solution.

4) Constraints: The constraints applied can be broadly classified into four classes.The first simply enforces the initial conditions of the ball. The second class enforce dynamics, which include natural ball flight dynamics over most indices, and reset maps at specific indices of hybrid transitions. It is important to also enforce the ball entering the corresponding transition guard set under ball flight dynamics at the specified timestep (as shown in Figure 4) to ensure the trajectories are continuous and feasible. The third class of constraints enforces the need for the contact point to be within the manipulators workspace, while the last class enforces step length positivity.

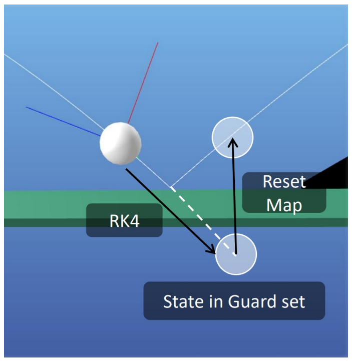

Fig. 4. Ensuring guard set is hit at transition timestep

C. Manipulator Trajectory Optimization

$$\begin{aligned}
\min_{\mathbf{X_m}} \quad & ||\mathbf{r}_\mathbf{m}^N - \mathbf{r_g}||_2^2 + ||\mathbf{v}_\mathbf{m}^N - \mathbf{v_g}||_2^2 + 10^{-6} \sum_{i=1}^{N} ||\mathbf{a}_\mathbf{m}^i||_2^2 \\
\text{s.t.} \quad & \mathbf{x}_\mathbf{m}^0 - \mathbf{x_{IC}} = 0 \\
& \phi(\mathbf{x}_\mathbf{m}^{n_2}) = \lambda(\mathbf{x_r}) \\
& \mathbf{r}_\mathbf{m}^{k+1} = \mathbf{r}_\mathbf{m}^k + \mathbf{h}^k \mathbf{v}_\mathbf{m}^k \\
& \mathbf{v}_\mathbf{m}^{k+1} = \mathbf{v}_\mathbf{m}^k + \mathbf{h}^k \mathbf{a}_\mathbf{m}^k
\end{aligned}$$

1) Kinematic Formulation: The manipulator trajectory optimization is formulated as kinematically feasible trajectories, since optimizing the trajectories with the 6DOF manipulator dynamics took a significant amount of time. The vector of time steps, $\mathbf{h}$, and the racket states, $\mathbf{x_r}$ are taken as inputs from the ball trajectory optimization. These timesteps ensure that there is a time synchronization between the manipulator's and ball's motion. The racket states are used as the contact constraint for the manipulator.

2) Decision Variables: The manipulator states that are optimized over are the six joint positions ($\mathbf{r_m} \in \mathbb{R}^6$), velocities ($\mathbf{v_m} \in \mathbb{R}^6$) and accelerations ($\mathbf{a_m} \in \mathbb{R}^6$) for each timestep, combined to form matrix $\mathbf{X_m} \in \mathbb{R}^{18\times N}$.

3) Objective Function: The objective function contains an acceleration regularization term, which helps enforce smooth trajectories by preventing jumps in accelerations, and terms to help return the manipulator to the home position ($\mathbf{r_g}, \mathbf{v_g}$) after hitting the ball.

4) Constraints: Constraints mandate the end effector transformation matrix and end effector velocities (captured through forward kinematics function ϕ) at timestep n_2 to match the desired racket transformation matrix and racket velocity (obtained by converting racket states to a transformation matrix, represented by λ). Additionally, kinematic derivative constraints are placed on the manipulator joint angles to ensure that trajectories generated are kinematically feasible.

D. Software and Solvers

Due to the large-scale, nonlinear nature of the problem, IPOPT was selected as the solver. Initial experiments with Python's CyIpopt were slow (5–10 minutes per solve), primarily due to the lack of sparsity exploitation and absence of analytical jacobians. A transition to Julia with fmincon improved performance at lower resolutions (20–30 seconds) but remained slow (3–4 minutes) at higher resolutions. The final implementation used CasADi in Python, which internally exploits sparsity and computes jacobians analytically, achieving solve times of 3–4 seconds at high resolution. For visualization and rendering, MuJoCo was initially used but later replaced with MeshCat due to the convenience in replaying and slow-motion visualizing capabilities without additional manual GUI or bookkeeping overhead.

IV. Results and Discussion

A. Verifying ball dynamics

Table tennis balls experience aerodynamic forces based on the direction of their spins: Balls with topspin are pushed downwards, with backspin are pushed upwards, and balls with sidespin curve sideways in the air. The ball flight dynamics model is verified by testing these effects with different angular rates as initial conditions. One such test is shown in Figure 5 where the ball is released with an initial velocity along the negative x- axis and provided a high angular velocity about the positive z-direction. It is evident that the ball curves to the left (negative y-direction) as a result of this spin.

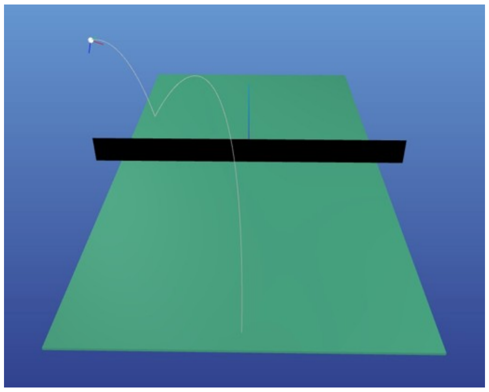

Fig. 5. Left side curve due to spin about positive z-axis

When a spinning ball contacts the table, the normal force from the table reverses the direction of the z-component of the ball's velocity, and friction changes the ball's velocity in the x-y plane. The Table Contact Reset Map is verified in a similar manner as the aerial dynamics; by observing how the ball interacts with the table and comparing it with an expected realistic scenario. One such example is shown in Figure 6 where the ball is lobbed with a high backspin, and after contacting the table, the ball's backspin reverses its direction of motion. The bounce height decreases as a result of the rebound not being perfectly elastic.

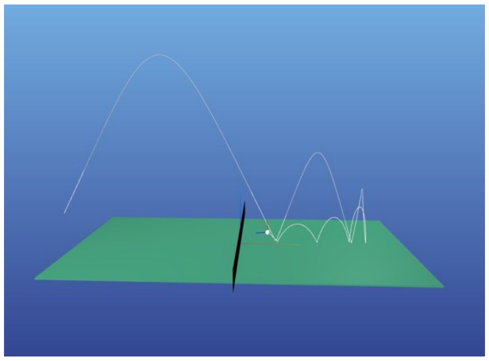
Fig. 6. Ball returns towards initial position due to backspin

B. Verifying Racket Angles

In a table tennis game, if a top-spinning ball hits a racket that is held vertically, it will shoot upwards (positive z-direction) on rebounding off the racket, making the ball take longer durations to reach the other side of the table. Similarly, backspinning balls after contacting a vertical racket tend to shoot downwards (negative z-direction), making it hit the table faster. These tendencies are shown in Figure 7.

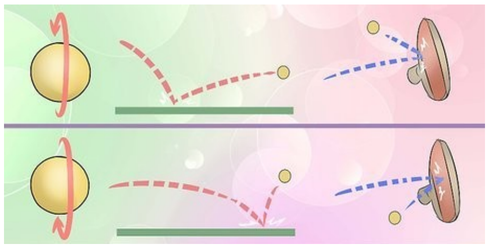
Fig. 7. Top: top-spinning ball, Bottom: Back-spinning ball

In order to return the top spinning ball faster, or to ensure back spinning balls reach the other side of the table, players hold their rackets at open or closed angles respectively, effectively countering tendencies to shoot upwards or downwards. This is shown in Figure 8.

Our setup verifies this requirement by tossing the ball at the manipulator at the same initial velocity with top spins and back spins, and asking the manipulator to hit the ball to identical locations on the table within the same time duration. The orientation of the racket relative to the vertical is observed while returning the ball. In accordance with the desired behavior, it is seen that an incoming back spin ball is sliced back by the manipulator, using an open racket angle, as shown in Figure 9, and an incoming top spinning ball is returned with a closed racket angle, as shown in Figure 10.

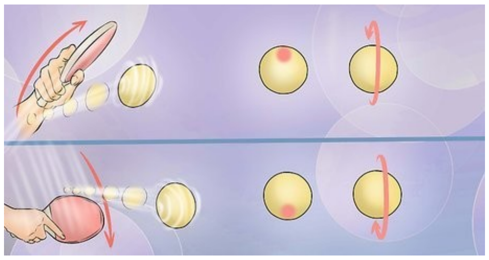
Fig. 8. Top: returning top spin, Bottom: Returning back spin

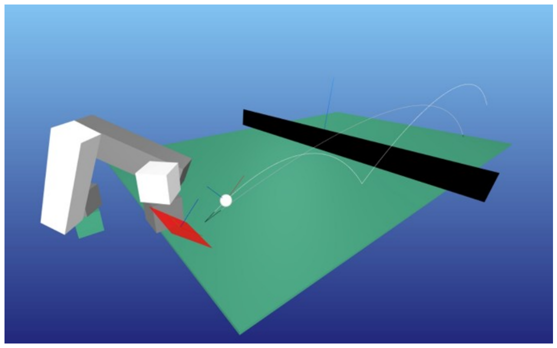
Fig. 9. Racket with open angle pushes ball upwards

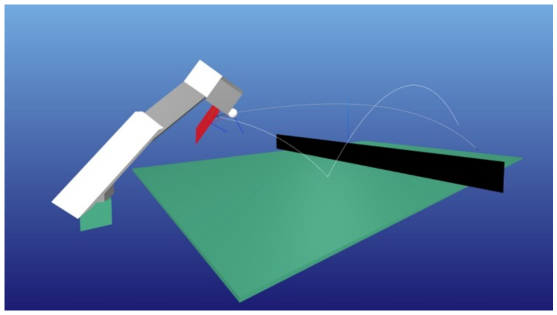
Fig. 10. Racket with closed angle pushes ball downwards

C. Solver Times

With CasADi, leveraging matrix sparsity and precomputing Jacobians, the ball trajectory optimization converged within 3–4 seconds, while the manipulator trajectory solved in 0.2–0.8 seconds. Interestingly, solver performance did not always correlate straightforwardly with problem size; in some cases, introducing additional constraints or shifting between constraints and objective functions improved solve times, likely due richer gradient information and a more restricted search space outweighing the overhead of computing their jacobians.

D. Rendering Challenges

A notable challenge encountered during rendering involved the visualization of high-spin ball trajectories. When the ball's spin frequency approached the simulation rendering frequency, visual artifacts emerged—for instance, if the spin frequency matched the rendering rate, the ball appeared stationary in orientation despite rotating. Slight mismatches is spin can result in an apparent reversal of spin direction as well! This issue was exacerbated when rendering trajectories with variable free-time step lengths, leading to the false perception of spin changes mid-flight. Importantly, this is purely a rendering artifact; the underlying dynamics remain accurate. Hence, it is an vital lesson: the simulation/rendering frequency must significantly exceed the frequency of the any other event's/phenomenon's that we are interested in for the phenomenon to be appropriately 'observable'

E. Manipulator Dynamics?

Initially, the full manipulator dynamics were modeled, and the optimization was performed directly over joint torques to generate the desired manipulator trajectory. However, due to the highly nonlinear nature of the manipulator dynamics, the solver consistently failed to converge—even for a single dynamics timestep! To address this, the approach was shifted to solve for only kinematic feasibility, optimizing directly over joint positions, velocities, and accelerations to obtain smooth trajectories. These trajectories can then be tracked using a PID controller, or alternatively, used to compute the corresponding joint torques through inverse dynamics.

F. Euler Angles

In the manipulator trajectory optimization, the end effector transformation matrix is constrained to match the racket transformation matrix at the timestep n_2, i.e. the instant of impact. An important step while working with such constraints is wrapping the racket's Euler angles and the manipulator joint angles between $-\pi$ and π. This is because transformation matrices contain sine and cosine terms, which take the same values for larger angles, and on solving the optimization without wrapping angles, the manipulator could (and did) follow trajectories where joints rotated multiple times at very high rates before reaching a goal state. An example illustrating this is, if the optimizer solves for a goal angle of some joint to be $\frac{5\pi}{2}$, the manipulator would follow a trajectory from joint angle 0 to $\frac{5\pi}{2}$ when it could have just gone from 0 to $\frac{\pi}{2}$. Therefore, wrapping angles enforces a minimalist trajectory and serves as a way to constrain accelerations too.

V. Concluding Remarks

The constructed optimal control formulation has been successful in making the robotic arm return an incoming table tennis ball to a target location on the table within a desired amount of time. This is rendered in a custom Meshcat simulation environment. Tests were conducted with different initial conditions and target hitting locations, and the observed behavior of the manipulator aligns well with real life techniques. Some extensions to the proposed methods that can lead to interesting future work are the following,

- Implementing a "defensive player" approach, where given some initial condition, the control algorithm itself figures out an optimal location and time to return the ball, such that table tennis rules are followed over the trajectory, effectively optimizing for hitting the ball back onto the table irrespective of how slow/fast the ball is hit.
- Adding a second manipulator and optimizing ball and manipulator trajectories for both manipulators, creating a rally.
- In the current implementation, not much emphasis was given to the speed at which code runs. A useful extension of this work would be to optimize the solver for near real-time deployment.

References

[1] David B. D'Ambrosio, Jon Abelian, Saminda Abeyruwan, Michael Ahn, Alex Bewley, Justin Boyd, Krzysztof Choromanski, Erwin Johan Coumans, Tianli Ding, Omar Escareno, Wenbo Gao, Laura Graesser, Atil Iscen, Navdeep Jaitly, Deepali Jain, Juhana Kangaspunta, Satoshi Kataoka, Gus Kouretas, Yuheng Kuang, Nevena Lazic, Corey Lynch, Reza Mahjourian, Sherry Moore, Thinh Nguyen, Ken Oslund, Barney J. Reed, Krista Reymann, Pannag Sanketi, Anish Shankar, Pierre Sermanet, Vikas Sindhwani, Avi Singh, Vincent Vanhoucke, Grace Vesom, and Peng Xu. Robotic table tennis: A case study into a high speed learning system. 2023.

[2] David B. D'Ambrosio, Saminda Abeyruwan, Laura Graesser, Atil Iscen, Heni Ben Amor, Alex Bewley, Barney J. Reed, Krista Reymann, Leila Takayama, Yuval Tassa, Krzysztof Choromanski, Erwin Coumans, Deepali Jain, Navdeep Jaitly, Natasha Jaques, Satoshi Kataoka, Yuheng Kuang, Nevena Lazic, Reza Mahjourian, Sherry Moore, Kenneth Oslund, Anish Shankar, Vikas Sindhwani, Vincent Vanhoucke, Grace Vesom, Peng Xu, and Pannag R. Sanketi. Achieving human level competitive robot table tennis, 2025.

[3] Tianli Ding, Laura Graesser, Saminda Abeyruwan, David B. D'Ambrosio, Anish Shankar, Pierre Sermanet, Pannag R. Sanketi, and Corey Lynch. Goalseye: Learning high speed precision table tennis on a physical robot, 2022.

[4] Okan Koç, Guilherme Maeda, and Jan Peters. Online optimal trajectory generation for robot table tennis. *Robotics and Autonomous Systems*, 105:121–137, 2018.

[5] Akira Nakashima, Yuki Ogawa, Yosuke Kobayashi, and Yoshikazu Hayakawa. Modeling of rebound phenomenon of a rigid ball with friction and elastic effects. In *Proceedings of the 2010 American Control Conference*, pages 1410–1415, 2010.