Ball Throwing Acrobot: A Trajectory Optimization Approach

Samir Wadhwa
Mechanical Science and Engineering
University of Illinois at Urbana-Champaign
samirw2@illinois.edu

***Abstract*—A ball throwing acrobot capable of throwing balls with different masses to different locations is created in this report. Trajectory optimization is used to determine admissible trajectories for the two-link acrobot and an infinte horizon LQR controller is used for the system linearized along the trajectory. The simulations are performed on a system which includes actuator dynamics and motor constraints.**
It is shown that this acrobot utilizes its dynamics to throw balls of different masses without a significant increase in energy consumed. Also, the acrobot follows a natural motion and the trajectory determined by optimization is similar to the way a human would perform the task.

***Index Terms*—trajectory optimization, nonlinear LQR control, underactuated robots, acrobot**

I. Introduction

Underactuated robots have the capability to exploit the dynamics of a system to achieve performance and robustness. Traditional robotics focused on cancelling the dynamics of the system to make it behave desirably. However, it is seen that these robots have unnatural movements and require excessive energy to run. On the other hand, living beings have evolved to have dynamics that can be exploited to achieve the desired motion with minimal energy. As examples, the shape of a fish aids it in swimming upstream and the ability to stand with legs vertical (a singularity normally avoided in robotics) reduces the energy needed by muscles to stand.

Trajectory optimization is an indispensable tool in underactuated robotics as underactuated systems cannot follow arbitrary trajectories. It is used to find a trajectory that meets the desired specifications and then control theory is used to make the robot follow that trajectory. Trajectory optimization has been used in [1] for online autonomous vehicle motion planning. They use differential dynamic programming to smooth the trajectory while accounting for non-holonomic constraints of the vehicle. In [2], trajectory optimization of the climb segment of a hypersonic vehicle is performed and a hybrid algorithm of particle swarm optimization and sequential quadratic programming is proposed. Motions for a quadruped bot such as walking, trotting and bouncing are created in [3] using a trajectory optimization formulation.

In this project a two-link acrobot is designed that can throw a ball to a bucket using only one actuator. This system is underactuated as the robot has two degrees of freedom and the ball cannot be controlled in flight. The bot must move its lower link to pump sufficient energy into the system and also leave the ball at an appropriate state to make it reach the bucket. The concepts of trajectory optimization and LQR control are used to first find a feasible trajectory from the starting position and then make the system follow this trajectory while accounting for actuator dynamics and adhering to motor constraints.

The report is structured as follows. In Section II, the concepts used in this report are introduced. In Section III, the methodology used for formulating this problem is described. In Section IV the results of the experiments performed using simulations are provided. The implications of these results are discussed in Section V. Finally, the report is concluded in Section VI. Some derivations and table of parameters are mentioned in Section VII.

II. Background

Acrobots have been studied before in [4], [5], [6]. As seen in this report, utilizing this underactuated system leads to a natural, energy efficient motion of the robot. Traditionally, the problem of the acrobot swing up to it's unstable equilibrium (upright) from the stable equilibrium is solved using energy shaping or trajectory optimization to reach close to the upright state and LQR control is used to stabilize the bot at this equilbirium.

This project considers the added complexity that the final state is not an equilibrium of the system and the real robot must be as close to this state as possible as the relation between the ball end state and the location of where it drops is nonlinear and small errors in the final state can lead to large deviations, thus making the ball miss the bucket.

The two major tools used in this report, trajectory optimization and LQR control are introduced now.

A. Trajectory Optimization

The term trajectory optimization refers to a set of methods that are used to find the best choice of trajectory, typically by selecting the inputs to the system, known as controls, as functions of time [7]. This is seen as an optimization routine where the states of the system and control inputs are the optimization variables and the system dynamics, actuator dynamics, control limitations are constraints. Thus, optimization routines are used to find a trajectory that can be satisfied by the system dynamics and leads to the desired motion. For brevity, only the case without motor constraints in described in this section.

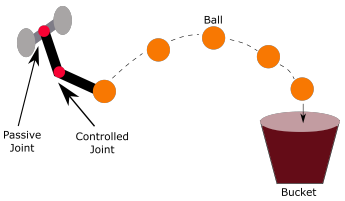

Fig. 1. Schematic diagram of the problem

First, the system dynamics must be discretized to have a finite number of optimization variables. Let the dynamics of a system be,

$$\dot{x} = f(x, u) \quad x(t_0) = x_0$$

Then, the discretized dynamics are written as,

$$x_{k+1} - x_k = \int_{t_k}^{t_{k+1}} f(x_k, u_k) dt$$

Where $x_k \approx x(t_k)$, $t_k = t_0 + (k-1)\Delta t$ and a numerical approximation of the integral is used. As Δt approaches zero, the dynamics of this discrete system get closer to the dynamics of the actual system. The method used to represent these constraints are called collocation constraints.

The cost function commonly used is the integral of control effort square. This is chosen as it is easy to compute and leads to smoother trajectories. Thus, the trajectory optimization problem becomes,

$$\min_{x[k], u[k]} \int_{t_0}^{t_f} u^2(\tau) d\tau$$

$$\text{s.t.} \quad x_{k+1} - x_k = \int_{t_k}^{t_{k+1}} f(x_k, u_k) dt$$
$$x_1 = x(0)$$

B. LQR Control

A Linear-Quadratic-Regulator (LQR) controller is used to find an optimal control policy for a linear system with quadratic constraints. An infinite horizon LQR controller, where the cost-function contains states as $t \to \infty$ has a direct solution. The problem inmplicitly enforces that the state x must go to zero as $t \to \infty$. This is formally defined below,

$$\min_u \int_{t_0}^{\infty} x(t)^T Q x(t) + u(t)^T R u(t) dt$$

$$\text{s.t.} \quad \dot{x}(t) = Ax(t) + Bu(t)$$

Here, Q represents the cost of having the state away from zero, R represents the cost to control effort. The solution is simply a state feedback of the form $u(t) = -Kx(t)$ and can be found using the MATLAB command 'lqr'.

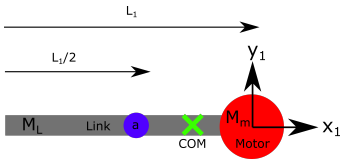

Fig. 2. Figure used for calculation of inertia of link 1

III. Problem Formulation and Solution Methods

The acrobot is a two-link manipulator that is attached to the pole with a revolute passive joint and has an actuated revolute joint (the hip) to move the second link as shown in Figure 1. The acrobot holds the ball at the end of the second link (foot) and releases the ball at a desired state. The ball is assumed to be a point mass that has the linear velocity of the foot when it is released and follows a projectile motion thereafter.

A. Dynamics of acrobot

The links are assumed to have a uniform distribution of mass and the motor is attached at the end of the first link and the ball is held at the end of the second link. The calculation for center of mass and moment of inertia of link 1 is shown here. The diagram used for this calculation is shown in Figure 2.

$$x_{com} = \frac{-M_L(L_1)/2}{M_L + M_m} \tag{1}$$

Where L_1 is the length of link 1, M_L is the mass of the link and M_m is the mass of the motor. Thus, the moment of inertia about point a is,

$$I_a = \frac{M_L L_1^2}{12} + M_m \frac{L_1^2}{4} \tag{2}$$

The moment of inertia about the center of mass is determined using the parallel axis theorem as shown in the equation below,

$$I_{com} = I_a - (M_L + M_m)\left(x_{com} - (\frac{-L_1}{2})\right) \tag{3}$$

The dynamics of the robot are then determined using standard Lagrane-Euler equations by writing the energy and potential energy of the system(see VII-A). These are succinctly written as,

$$D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = B_e u \tag{4}$$

Where u is the torque input by the motor at the hip joint, $q = \begin{bmatrix}\theta_1 & \theta_2\end{bmatrix}^T$. $D(q)$ is the inertia matrix, $C(q, \dot{q})$ is the coriolis matrix and $G(q)$ is the gravity vector. $B_e = \begin{bmatrix}0 & 1\end{bmatrix}^T$.

The actuator dynamics lead to an augmented form of (4) shown below,

$$\tilde{D}(q)\ddot{q} + \tilde{C}(q, \dot{q})\dot{q} + G(q) = B_e u \tag{5}$$

$$\tilde{D}(q) = D(q) + J_{rotor} \quad \tilde{C}(q,\dot{q}) = C(q,\dot{q}) + B_{damp}$$
$$J_{rotor} = diag(0, I_{rotor}N_H^2) \quad B_{damp} = diag(0, \frac{k_v k_T}{R_W}N_i^2)$$

These are written in the state space form with $x = \begin{bmatrix} q & \dot{q} \end{bmatrix}$,

$$\dot{x} = f(x,u) = \begin{bmatrix} \dot{q} \\ D^{-1}\left(B_e u - C(q,\dot{q})\dot{q} - G(q)\right) \end{bmatrix} \tag{6}$$

The dynamics are calculated using the symbolic toolbox in MATLAB and then converted to independent functions.

B. Trajectory Optimization Setup

Trajectory optimization is used to find an admissible trajectory for this underactuated system that starts from rest at the stable equilibrium and ends at a point that leads to the ball falling into the bucket. Trapezoidal collocation with direct transcription is used in this report where both states and control inputs are optimization variables. This is preferred due to the simplicity in writing constraints and better numerical conditioning. For other transcription and collocation methods the reader is referred to [7], [8].

The constraints used in this optimization are the system dynamics, motor constraints and final state constraints (to ensure that ball reaches the bucket). The system is discretized into N time segments leading to $5(N+1)$ optimization variables (4 states, 1 control). Trapezoidal collocation is used to determine the system constraints as shown below,

$$x_{k+1} - x_k = \frac{1}{2}\Delta t\left(f(x_k,u_k) + f(x_{k+1},u_{k+1})\right) \quad \forall k = [1,N] \tag{7}$$

The motor constraints are linear inequality constraints shown in (8)-(10).

$$u_k \frac{1}{N_H} \leq \tau_{Imax} \qquad -u_k \frac{1}{N_H} \leq \tau_{Imax} \tag{8}$$

$$u_k \frac{1}{N_H} + \dot{q}_{2k}\frac{N_H \tau_{stall}}{\omega_{NL}} \leq \tau_{stall} \tag{9}$$

$$-u_k \frac{1}{N_H} - \dot{q}_{2k}\frac{N_H \tau_{stall}}{\omega_{NL}} \leq \tau_{stall} \quad \forall k = [1,N+1] \tag{10}$$

The constraints for the final state are calculated using the equations of projectile motion. Let x_0, y_0, v_{x0}, v_{y0} be the position of the ball when the acrobot releases the ball and x_b, y_b be the location of the bucket. In the simulation, x_0, y_0 are determined using forward kinematics and v_{x0}, v_{y0} are determined using the velocity jacobian at state x_{N+1}. The constraints are then calculated as shown below.

Equation of motion for the y-coordinate is (with initial time as zero),

$$y(t) = v_{y0}t - \frac{gt^2}{2} + y_0 \tag{11}$$

Time taken for the ball to reach x-coordinate x_b is,

$$t_e = \frac{x_b - x_0}{v_{x0}} \tag{12}$$

Thus at time t_e the y-coordinate should be y_b, i.e the constraints become,

$$y_b - v_{y0}t_e + \frac{gt_e^2}{2} - y_0 = 0, \quad t_e = \frac{x_b - x_0}{v_{x0}} \tag{13}$$

$$t_e \geq 0 \tag{14}$$

where (13) ensures that the ball falls into the bucket and (14) ensures that the ball falls into the bucket in forward time. It can be seen that there are infinite solutions to final states that satisfy this constraint and the optimization routine must determine the one that can be achieved by the acrobot. This optimization was performed using 'fmincon' in MATLAB.

After the solution is found at discrete points, the optimal trajectory and control input are found using second order interpolation for states and first order interpolation for control inputs as shown in (15), (16).

$$x_{opt}(t) = x_k + \xi f_k + \frac{\xi^2}{2\Delta t}(f_{k+1} - f_k) \tag{15}$$

$$u_{opt}(t) = u_k + \frac{\xi}{\Delta t}(u_{k+1} - u_k) \tag{16}$$

$$f_k = f(x_k,u_k),\ t \in [t_k, t_{k+1}],\ \xi = t - t_k$$

C. Initial condition for optimization

As the system is nonlinear and the added constraints to make the ball fall into the bucket are nonlinear the optimization routine is not guaranteed to return a globally optimal solution and the result is sensitive to the initial guess given. A bad initial guess may lead to the optimization being unable to find an admissible solution and a good guess can significantly speedup the process.

A common practice is to use linear interpolation between the initial state and final state for all variables as an initial guess. As the final state in this problem is not fixed, the initial guess is broken down into two parts.

1) A final state which leads to the ball going into the bucket is used to form the interpolation (which may not be achievable by the acrobot) for the initial guess and a solution is found without adding motor constraints(see VII-B).
2) The solution obtained above is used as an initial guess to the solve the full problem with motor constraints

Optimization without these two steps does not return an admissible solution and this idea speeds up the process.

D. Tracking of the trajectory

Once the trajectory is determined using the optimization routine, the real system must be made to follow this trajectory. An infinte-horizon LQR controller is used to achieve this. Denoting the deviation of the states and control inputs of the system as $\delta x = x - x_{opt}, \delta u = u - u_{opt}$, the system is linearized along the trajectory at every time as,

$$\delta\dot{x} + \dot{x}_{opt} \approx f(x_{opt}, u_{opt}) + \frac{\partial f(x,u)}{\partial x}\Big|_{(x_{opt},u_{opt})} \delta x + \frac{\partial f(x,u)}{\partial u}\Big|_{(x_{opt},u_{opt})} \delta u \quad (17)$$

Since the optimial trajectory approximately follows the system dynamics, $\dot{x}_{opt} \approx f(x_{opt}, u_{opt})$. The linearized dynamics can then be written as,

$$\delta\dot{x} = A_{lin}(t)\delta x + B_{lin}(t)\delta u \quad (18)$$

This represents the linearized system at every time step. At every time step the control law is determined by setting $R = 1, Q = I_4$. A function defining A_{lin}, B_{lin} with arguments x_{opt}, u_{opt} is determined while the dynamics of the system are generated. This control law is used to drive the dynamics of δx to zero as shown in (19).

$$u(t) = u_{opt} - K(x - x_{opt}), \quad K = lqr(A_{lin}, B_{lin}, Q, R) \quad (19)$$

To ensure that the real system satisfies motor constraints, it was first seen that the control signal does not saturate at τ_{Imax} and only the motor stall constraints are binding. Decreasing the torque to satisfy constraints online would not lead to the ball reaching the bucket as the final state must be achieved accurately. For this reason a value of $0.9\tau_{Imax}$ is used during the optimiation process. With this value, the motor constraints are not violated on the actual robot despite augmenting the control signal with a LQR control law.

IV. Results

The purpose of this study is to exploit underactuated systems to achieve better performance in terms of minimizing the trajectory time and energy required by the acrobot to achieve the final state. This is done using a series of experiments shown in this section.

A. Numerical results of a nominal system

For the optimization routine, time is discretized into 25 segments and the bucket location is $x_b = 2.0m$ and $y_b = -1.0m$. The optimization procedure requires 207 iterations to generate a solution for a good guess and 298 iterations to generate the solution for the actual system. The constraint tolerance is set to 10^{-6} and optimality tolerance is 10^{-2}. The constraint tolerance is chosen as a low value to ensure an admissable solution is generated which leads to the ball falling in the bucket. Time for the trajectory is chosen as $2sec$ and mass of the ball is $100g$ which is close to the mass of a tennis ball.

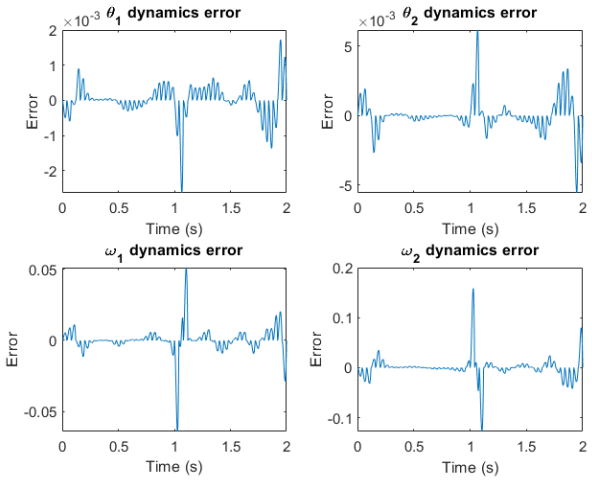

Fig. 3. Error in dynamic equations, $x - x_k - \int_{t_k}^{t} f(x,u)dt,\ t \in [t_k, t_{k+1}]$, where the integral is calculated using the trapezoid rule. This error is plotted against time, to determine the value at interpolation points

This code was written from scratch to allow for analysis and changes as required.

As seen in Figure 3, the error between the actual dynamics and the results of the optimization with interpolation is zero at the knot points and reasonably small between them. The error is 10^{-3} for the angles and 10^{-1} for velocities which is expected as the errors in velocity dynamics are high at fast portions in the trajectory.

The results of this optimization are used to determine the control law (19) in the real system and drive the dynamics to the required state. As seen in Figure 4, the actuator used on the real time does not violate the constraints and is not close to saturation as mentioned earlier. The plots of the state trajectory taken by the system and the control input are generated along with the optimal trajectory to compare the results. It can be seen in Figure 5 that the real system tracks the trajectory well implying that a valid admissible trajectory was generated during the optimization routine.

The full simulation is structured as follows, since the ode45 solver is MATLAB does not allow discontinuous differential equations, first the simulation of the robot with the ball is performed using the system dynamics calculated earlier. Then, the projectile motion of the ball is calculated using initial conditions and velocity as the position and velocity of the foot in the task space and it is checked if the ball falls into the bucket. This simulation is depicted in Figure 6.

B. Handling balls with different mass

The simulation procedure above was used to make the same robot throw different balls to bucket at the same location. During the simulation the total energy used by the actuator is determined using (20).

$$E = \int_0^{tf} V_H I_H dt \quad (20)$$

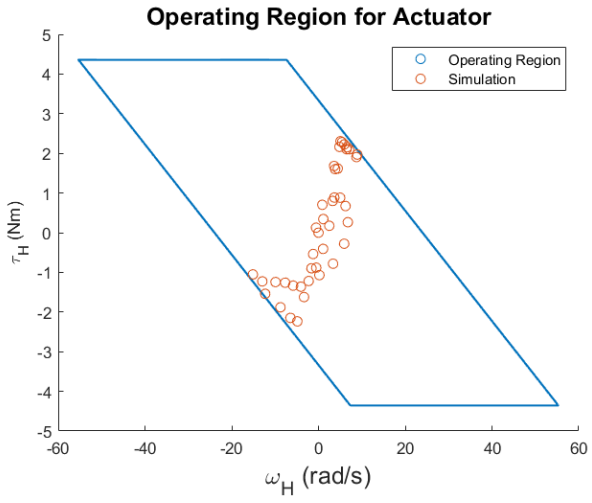

Fig. 4. The motor operating region along with the trajectory used by the control law, it can be seen that the motor only operates within the operating region on the real system

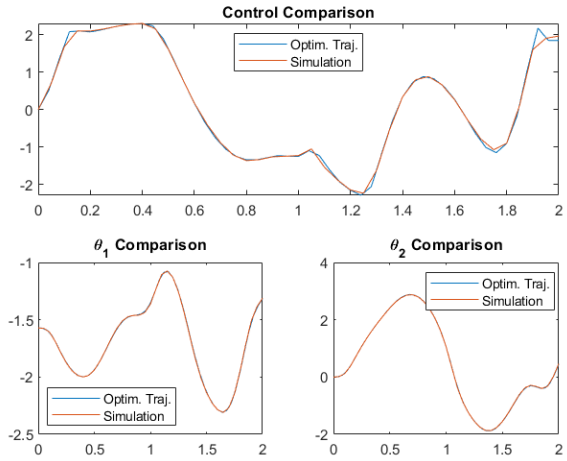

Fig. 5. The trajectory determined using optimization and the actual trajectory are plotted to confirm that an admissible solution is found

$$I_H = \frac{u}{N_H K_T} \quad V_H = R_W I_H + K_v N_H \omega_2$$

Average power consumed is determined as $P_{avg} = \frac{E}{t_f}$. The results for balls with different masses are shown in Table I. The trajectory time and bucket location are fixed to $2s$ and $x_b = 2.0m, y_b = -1m$ and the mass of the ball is varied. Iter1 is the number of iterations required to generate a good initial guess and Iter2 is the number of iterations for the final solution. For mass 0.4kg the solver terminated prematurely and hence the solution is not optimal. Nonetheless, it can still be seen that the energy consumed by the robot does not increase significantly with a significant increase in ball mass. A basketball has a mass of 0.623 kg and during the simulation the exact radii of a basketball and basket were used, this is shown in the last row

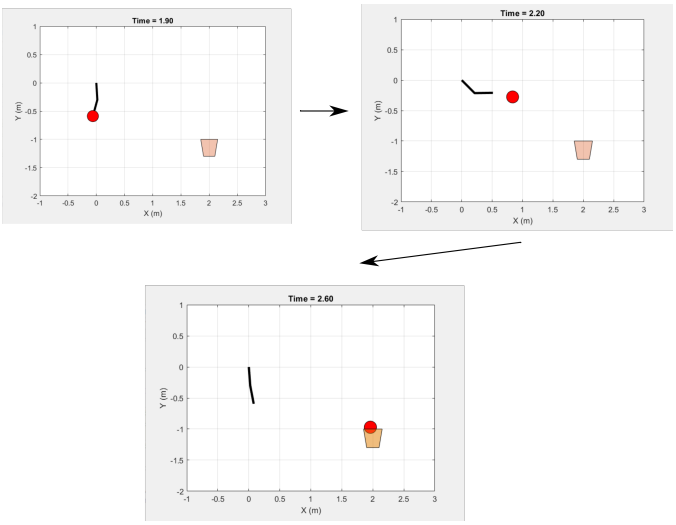

Fig. 6. Simulation of the real system to make the ball reach the bucket

M (kg)	E (J)	P_{avg} (W)	Contr. Tol.	Opt. Tol.	Iter1	Iter2
0.1	60.87	30.43	10^{-6}	10^{-2}	207	298
0.2	62.23	31.12	10^{-6}	10^{-2}	181	245
0.3	64.52	32.26	10^{-6}	10^{-2}	188	183
0.4	94.04	47.02	9×10^{-10}	0.2	199	384
0.5	73.46	36.73	10^{-6}	10^{-1}	192	383
0.623	75.19	37.59	10^{-6}	10^{-1}	191	172

TABLE I
RESULTS OF EXPERIMENTS WITH VARYING MASSES

of Table I. This confirms that the acrobot is energy efficient and exploits its dynamics to throw the ball.

C. Experiments with different trajectory times

In this experiment the mass of the ball is fixed to $0.1g$ and the bucket location is $x_b = 2.0m, y_b = 1.0m$. The time given to the optimization routine is changed to determine how low can be time be to ensure that the ball reaches the bucket and the time at which minimum energy is consumed by the acrbot. The constraint and optimality tolerance for these experiments were 10^{-6} and 10^{-2} respectively. The results are shown in Table II. As seen here, the power required is high for $t_{traj} = 1s$, it reduces till $t_{traj} = 2.5s$ and increases for $t_{traj} = 3s$. From the values of energy it can be seen that the optimum value of time is somewhere around $2.5s$ and $2.75s$.

D. Experiments with changing basket location

Here, the second goal of the project is studied, i.e. the optimization and controller should be able to handle different bucket locations. Here, the ball mass and trajectory time

Time (s)	E (J)	P_{avg} (W)	Iter1	Iter2
1	82.37	82.37	41	386
2	60.87	30.43	207	298
2.5	59.46	23.68	151	93
2.75	59.67	21.69	232	176
3	127.1	42.37	222	174

TABLE II
RESULTS OF EXPERIMENTS WITH VARYING TRAJECTORY TIME

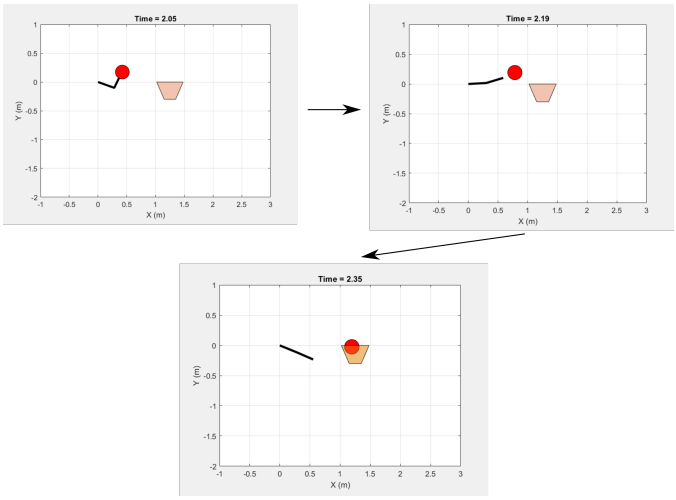

Fig. 7. Acrobot throwing the ball when bucket location is at the height of the first joint

Sr. No.	x_b	y_b	E(J)	P_{avg} (W)	Achieved?
1	2	-1	60.87	30.44	Yes
2	2	0	186.54	93.27	No
3	1.5	0	102.48	51.24	Yes
4	1.25	0.25	120.41	60.20	Yes

TABLE III
EXPERIMENTS WITH VARYING BUCKET LOCATION

are fixed to be $0.1kg$ and $2s$ and the bucket locations are varied. The same idea is used to solve the problem, first an initial guess is determined using optimization without motor constraints followed by the solution for the full system. This is then used in the LQR based controller to make the ball fall into the bucket. The results for these experiments are shown in Table III.

It is seen that for $x_b = 2.0m, y_b = 0m$ an admissible solution is not found which is expected as the system cannot throw the ball at arbitrary locations and has limitations due to the time allowed, motor constraints and dynamics of the system. However, on decreasing the distance between the bucket and manipulator to $x_b = 1.5m$ the manipulator is able to throw the ball.

Also, the acrobot follows an interesting trajectory to throw the ball when the bucket is above the first joint. The numerical results are shown in row 4 of Table III. The manipulator follows a trajectory similar to shooting a basketball towards the end to throw the ball upwards. This is shown in Figure 7.

V. DISCUSSION

It is seen from the results and animations performed that this underactuated system performs natural and smooth motions which is unlike the results of traditional control laws used for fully actuated systems.

Also, the optimization routine performs well for different ball masses, bucket location and trajectory times. The system is energy efficient as the energy consumed for ball with different masses does not change significantly even when the mass of the ball is increased by a factor of 6. The optimal time to release the ball was also determined using experiments and it was seen that very short and long trajectories consume high energies and an intermediate value is optimal.

Lastly, by varying bucket locations the limitations of the underactuated system are seen where it cannot throw the ball to arbitrary locations. Moreover, it was seen that the acrobot realizes ball throwing motions used by humans while playing basketball which further corroborates the fact that underactuated system have more natural motions.

This project utilized the concepts of forward kinematics and lagrange-euler dynamics to define the system dynamics and constraints. Symbolic toolbox in MATLAB was used to generate the dynamics and the dynamics of the linearized system (18). Actuator dynamics and motor constraints were included to increase the applicability of this report to a real system.

Trajectory optimization was used to determine an admissable trajectory that achieves the desired objective. The optimization routine was built from scartch only using 'fmincon' in MATLAB to make the system robust to different ball masses, bucket locations and trajectory times. The nonlinear system was linearized along the trajectory determined by the optimization routine and infinite horizon LQR controller was used to achieve tracking.

The next steps in continuation of the same project would be to explore different methods of giving an initial condition which could lead to more energy efficient solutions than the ones obtained here as 'fmincon' only converges to a local minima. The mechanism needed to release the ball from the manipulator can be explored and it's dynamics can be accounted for to further ensure that the methods derived here can be used on a real robot.

Also, different optimization techniques such as reinforcment learning could be explored to compare results or reach an online solution.

VI. CONCLUSION

An underactuated system, the acrobot, was used to throw a ball to a desired bucket location. Admissible trajectories are determined using trajectory optimization and the real system is controlled using linearization along this trajectory at every point with LQR control. The system was simulated while accounting for actuator dynamics and motor constraints to make the study more practically applicable.

The algorithm developed was shown to be robust to changes in ball mass, bucket location and trajectory time and the results pertaining to changing each of these parameters were discussed. It is also seen that the acrobot exploits it's dynamics to throw the ball in an energy efficient manner.

Further work will be to make the nonlinear optimization process better by exploring different initial guesses for solvers. Also, the mechanism used to release the ball can be accounted for to make the simulation even more realistic. Moreover, the problem can be approached using different optimization

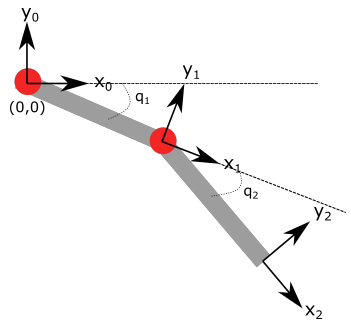

Fig. 8. Diagram of the acrobot with coordinate frames and angles

Sr. No.	Label	Description	Value
1	M_1	Mass of link 1 without motor	1kg
2	M_2	Mass of link 2 without ball	1kg
3	L_1	Length of link 1	0.3m
4	L_2	Length of link 2	0.3m
5	M_m	Mass of motor	0.5kg
6	N_H	Motor Gear Ratio	26.9
7	K_v	Motor speed constant	0.0186 V/rad s
8	K_T	Motor torque constant	0.0135 Nm/A
9	I_{rotor}	Moment of inertia of rotor	$7 \times 10^{-6} kgm^2$
10	τ_{stall}	Motor stall torque	0.124 Nm
11	ω_{NL}	Motor no load speed	645.2 rad/s
12	τ_{Imax}	Maximum motor torque	0.162 Nm

TABLE IV
TABLE OF PARAMETERS USED IN THE SIMULATION

routines such as reinforcement learning and the results of the two methods can be compared.

VII. Appendix

A. Calculation of System Dynamics

Dynamics of the acrobot are calculated using Lagrange-Euler equations. The kinetic energy of each link is written as a function of the state $x = \begin{bmatrix} q & \dot{q} \end{bmatrix}^T$. First, the forward dynamics of the system are determined using Figure 8. Let the transformation matrices be T_i^j, that represents the transformation matrix from frame i to j. The rotation matrix R_i^j is defined similarly. Using these, the location of center of mass of the two links $r_i(q)$ is determined and the velocity jacaboian is determined as,

$$J_{vi} = \frac{\partial r_i(q)}{\partial q} \tag{21}$$

The angular velocity jacobian of the two links is,

$$J_{\omega 1} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix} \quad J_{\omega_2} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$$

Then, the kinetic energy of the two links are written as,

$$K_i = \frac{1}{2}(J_{vi}\dot{q})^T M_i (J_{vi}\dot{q}) + \frac{1}{2}(J_{\omega i}\dot{q})^T R_i^0 I_i R_0^i J_{\omega i}\dot{q} \tag{22}$$

Here M_i is the total mass of the link including the motor or ball. The potential energy of the system is,

$$P = \sum_{i=1}^{2} M_i \begin{bmatrix} 0 & g & 0 \end{bmatrix} r_i \tag{23}$$

Using these, the inertia matrix of the system $D(q)$ is determined using the energy of the system, coriolis matrix $C(q, \dot{q})$ is determined using Christoffel symbols and the gravity vector $G(q)$ is determined as $\frac{\partial P}{\partial q}$. The dynamics of the two-link system can then be written in the form of (4). These are then augmented to account for actuator dynamics as shown in III-A.

B. Initial guess for optimization

For the optimization problem used to generate an initial guess, a good initial guess must be given that aids the 'fmincon' in converging to the solution without motor constraints.

For this initial guess, a linear interpolation between the initial state of the system, $x_0 = [\frac{-\pi}{2}, 0, 0, 0]$ and a final state x_f is used which is one valid state that will lead to the ball falling into the bucket. This state may not necessarily be achievable in the time/motor constraints but still works as a good starting point for the solver.

The calculation of this x_f for any bucket location is shown here. Reasonable joint angles at the final state are chosen as,

$$q_f = \begin{bmatrix} -\frac{\pi}{4} & \frac{\pi}{6} \end{bmatrix} \tag{24}$$

This allows the position of the ball at release, x_{b0}, y_{b0} to be determined using forward kinematics. Now, the time of flight of the ball is taken to be $t_e = 1.5s$ and the initial and final velocity are determined using (11) and (12) as,

$$v_{x0} = \frac{x_b - x_{b0}}{t_e} \tag{25}$$

$$v_{y0} = \frac{1}{t_e}(y_b + \frac{g t_e^2}{2} - y_{b0}) \tag{26}$$

Thus, the angular velocities at the release state are found using the velocity jacobian as,

$$\dot{q} = J_{foot}^{-1} \begin{bmatrix} v_{x0} \\ v_{y0} \end{bmatrix} \tag{27}$$

This guess leads to convergence of the optimization problem for generating an initial guess for the full problem.

References

[1] W. Huang, X. Wu, Q. Zhang, N. Wu, and Z. Song, "Trajectory optimization of autonomous driving by differential dynamic programming," in *2014 13th International Conference on Control Automation Robotics Vision (ICARCV)*, 2014, pp. 1758–1763.

[2] L. Feng, L. Liu, and Y. Wang, "Trajectories optimization of hypersonic vehicle based on a hybrid optimization algorithm of pso and sqp," in *The 27th Chinese Control and Decision Conference (2015 CCDC)*, 2015, pp. 4518–4522.

[3] A. W. Winkler, F. Farshidian, D. Pardo, M. Neunert, and J. Buchli, "Fast trajectory optimization for legged robots using vertex-based zmp constraints," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2201–2208, 2017.

[4] R. Jafari, F. B. Mathis, and R. Mukherjee, "Swing-up control of the acrobot: An impulse-momentum approach," in *Proceedings of the 2011 American Control Conference*, 2011, pp. 262–267.
[5] F. Xue, Z. Hou, and H. Deng, "Balance control for an acrobot," in *2011 Chinese Control and Decision Conference (CCDC)*, 2011, pp. 3426–3429.
[6] K. Kawada, S. Fujisawa, M. Obika, and T. Yamamoto, "Creating swing-up patterns of an acrobot using evolutionary computation," in *2005 International Symposium on Computational Intelligence in Robotics and Automation*, 2005, pp. 261–266.
[7] M. Kelly, "An introduction to trajectory optimization: How to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.
[8] R. Tedrake, "Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation (course notes for mit 6.832)," *Downloaded in Fall*, 2014.