UNIVERSITY OF CALIFORNIA, SAN DIEGO

Simulation and Optimization of a Two-Wheeled, Ball-Flinging Robot

A thesis submitted in partial satisfaction of the requirements for the degree

Master of Science

in

Engineering Sciences (Aerospace Engineering)

by

Po-Ting Chen

Committee in charge:

Professor Thomas Bewley, Chair
Professor Frank Talke
Professor Mauricio de Oliveira

2010

The Thesis of Po-Ting Chen is approved and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

Chair

University of California, San Diego

2010

DEDICATION

This thesis is dedicated to my parents, who have been with me through the good and the difficult times. Thank you for your support, guidance, and unconditional love.

Thank you Ching-Hung and Chin-Yi

Table of Contents

Chapter 3

Chapter 4

Chapter 5

Chapter 6

List of Figures

Acknowledgements

I would like to thank Professor Thomas Bewley, my adviser, for his support and guidance.

I would also like to thank Rob Krohn for helping me understand the necessary mathematical background and the optimization strategy. Without his help, I would not be able to finish this work.

My sincere thanks goes to Andrew Cavender and Chris Schmidt-Wetekam for helping me with the modeling and simulation.

My special thanks goes to my research partner, Benjamin Sams, for the successful and ingenious prototype.

ABSTRACT OF THE THESIS

Simulation and Optimization of a Two-Wheeled, Ball-Flinging Robot

by

Po-Ting Chen

Master of Science in Engineering Sciences (Aerospace Engineering)

University of California, San Diego, 2010

Professor Thomas Bewley, Chair

This paper presents a method for optimizing the throwing distance of a two-wheeled, self-balancing, remote controlled robot. The robot is maneuverable and comparable in size to a remote controlled toy car, but it moves around in an upright configuration using feedback control. In addition, it is capable of automatically picking up and throwing ping-pong balls. When throwing a ball, the body attached throwing arm is allowed to rotate quickly from a lay-down position to an upright position utilizing the principle of conservation of angular momentum.

The equations of motions of a representative model of the dynamic system are derived with Lagrange equation and Rayleigh dissipation functions. The optimization consists of two parts. The shape optimization is based on the ball's exit condition from the arm, which wraps around an adjoint based motor input optimization. In simulation, this optimization scheme results in a significant increase in throwing distance while keeping the magnitude of the motor input within comparable range.

Chapter 1

Introduction

Figure 1: Picture of the current iFling prototype

1.1 Background

The UCSD robotics laboratory started iFling robot program several years ago to develop an agile remote controlled toy car that is capable of throwing ping-pong balls. There has been several design iterations, and it is now in the third generation. The second-generation prototype performed very successful in the upright configuration. It was able to move around nimbly and brought much fun to the user. However, it was incapable of retrieving ping-pong balls efficiently and was only capable of making a catapult style lobbing, which resulted in a short throwing distance. The third design iteration focuses on creating an efficient ball pickup and ball-tossing robot.

1.2 Motivation

iFling is a novel radio controlled robot intended for the toy industry. The core focus is to deliver an affordable toy package that has the agility of a remote controlled car and the excitement of playing catch and throw. The design has the potential of becoming everyday toys for the children, group competitions, and possibly serving balls for athletes by improving the level of automation and by scaling up the system to the appropriate size.

1.3 Outline

In this paper, I will present the overall design of the third generation iFling, so the people can better understand the physical system. Second, I will focus on the theory of modeling the ball throwing motion, which includes the combined dynamic system and the trajectory of the ball in the air. From there, I will explain the optimization approach used to obtain a solution for making a longer ball toss. Finally, simulation results and comparison of the optimal solution with the initial guess will be presented and discussed. In addition, I will explore a few possibilities for future research directions to bring the project to the next level.

Chapter 2

Prototype Design

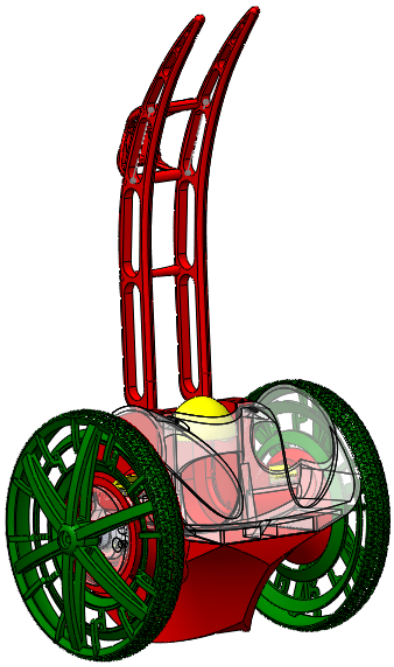
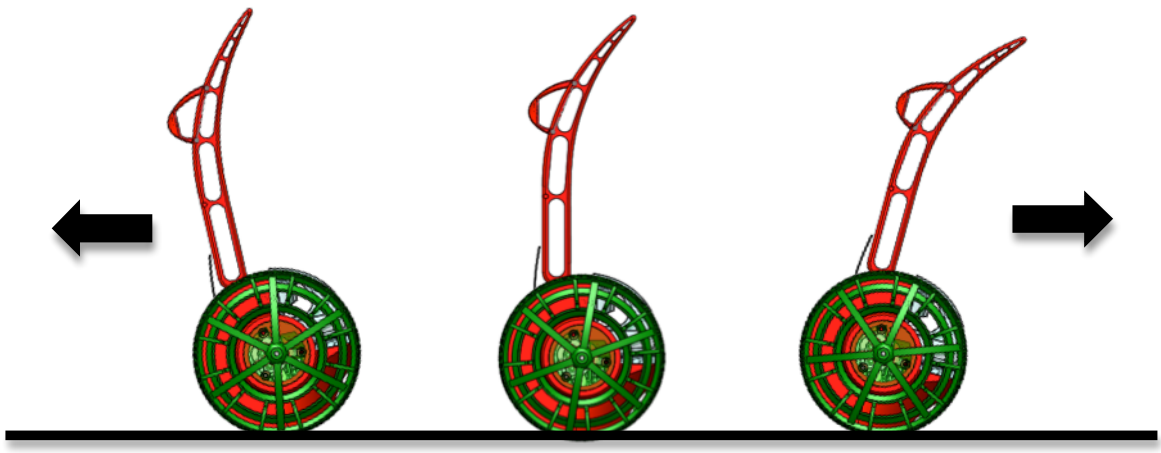


Figure 2: 3D computer model of iFling

iFling is an unique remote controlled robot. It has many features and detail designs that make this robot one of a kind. However, the purpose of this paper is not to focus on the design process or the manufacturing steps of the robot. In this chapter, I will briefly explain its features and designs relevant to the main topic of this paper, which is the ball throwing optimization.

2.1 How iFling Works?

2.1.1 Upright/Pickup Mode

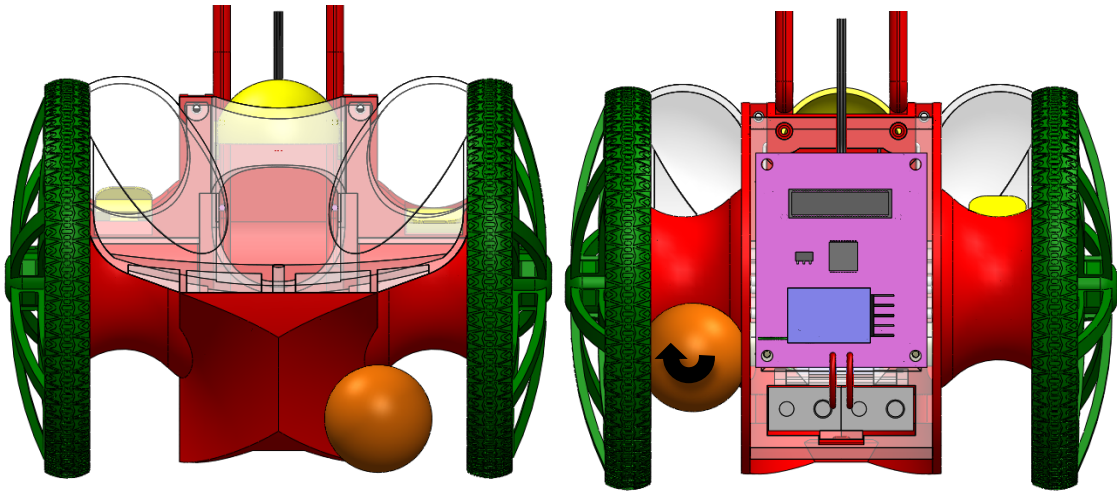


(a) moving backward (b) stationary (c) moving forward

Figure 3: Illustration of the upright mode

iFling's operation can be separated into two modes. The first mode is the upright mode. In this mode, the robot self-balances with a PID feedback control, which also acts as a drive mechanism to move the robot forward and backward. See figure 3. This mode is also the ball pickup mode. The body and the wheels are spaced apart with a gap of size slightly less than the diameter of a ping-pong ball, which is about 4cm. This spacing is designed to provide automatic ping-pong ball pickup. The curvature of the body and the shape of the wheels are especially design to make sure the ball is caught between the wheels and the body. See figure 4.a. The rotation of the wheels will smoothly bring the ball onboard and store them inside a basket. See figure 4.b. In order to prevent the balls from coming out of the basket during violent maneuver, there are two spring-loaded one-

way gates located at each entrance. This design makes ball pick up very simple for the users because all they have to do is to drive the robot towards the ball with a slight bias either to the left or to the right of the target ball for pickup.



(a) (b)

Figure 4: Illustration of ball pickup. (a) The body and the wheel are used to catch the ping-pong ball. (b) The ball is carried to the basket by the wheel.

2.1.2 Flinging Mode

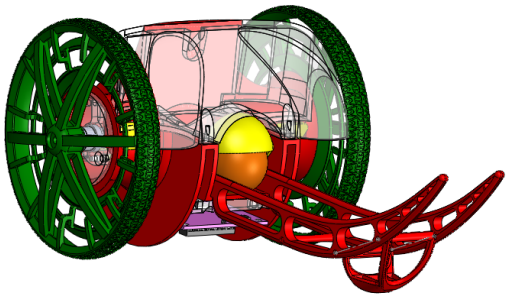


Figure 5: Flinging mode

Once the balls are stored inside the basket, it lays down to load a single ball onto the throwing arm. See figure 5. The ball is prevented from rolling down the track with a ball release mechanism that is actuated with a servomotor located at the bottom of the robot. In this configuration, the ball is ready to be launched. Once the release mechanism opens, the ball will roll down the track, and the robot will perform a quick jerking motion to rotate the body attached throwing arm to fling the ball. See figure 6. This rotation is possible because of the large difference between the rotational inertia of the wheels and the body. This optimization of the jerking motion and the shape of the arm are the main focus in chapter 4.

2.2 Throwing Arm

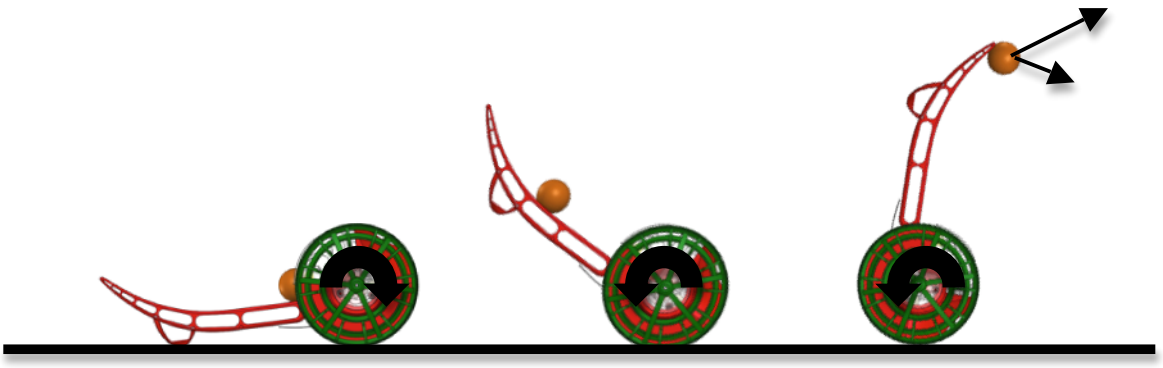


Figure 6: iFling throwing motion with velocity components

Originally, two types of throwing arm were considered. The first is the catapult style, and the second is the Jai Lai style. Their first main difference is the launch velocity. The initial velocity of the ball impart from a catapult style arm is only the tangential velocity. On the other hand, the Jai Lai style throwing arm is capable of giving the ball not only a tangential component but also a radial component. Refer to figure 6.

In short, the launch velocity of a ball coming out of the Jai Lai style arm will be higher than that of the catapult style arm, which directly affects the distance of the throw.

The second difference is the control input. In order to prevent the arm from simply dragging the ball from the start of the throwing sequence to the end, the control input must reverse at some point in the time window. This kind of inherent behavior will certainly limit the launch velocity since part of the launching time window is dedicated for slowing down the arm rather than accelerating the ball.

2.3 Actuator

iFling uses two Solarbotics RM2 RC motors see figure 7 and is powered by two 9V batteries. The motors are placed in the base of the robot. Each motor is coupled with a 40:1 gearbox located on the right and the left side of the robot. The typical stall torque for this type of motor is around 0.00373 *Nm*. Adding the gear ratio and multiply it by two (iFling has two motors) gives the total torque of around 0.3 *Nm.* This maximum torque is the reference used in finding the optimal control solution.

Figure 7: Solarbotics RM2 DC motors

Chapter 3

System Modeling

Developing a model is always the first step in optimizing a system. The optimization result is only effective, when the model of the system is representative. For iFling, the system modeling can be separated into two parts. The first part is the dynamics of the iFling and the second is the projectile motion of the ping-pong ball. In order to simplify the model complexity and optimization strategy, a few assumptions are made.

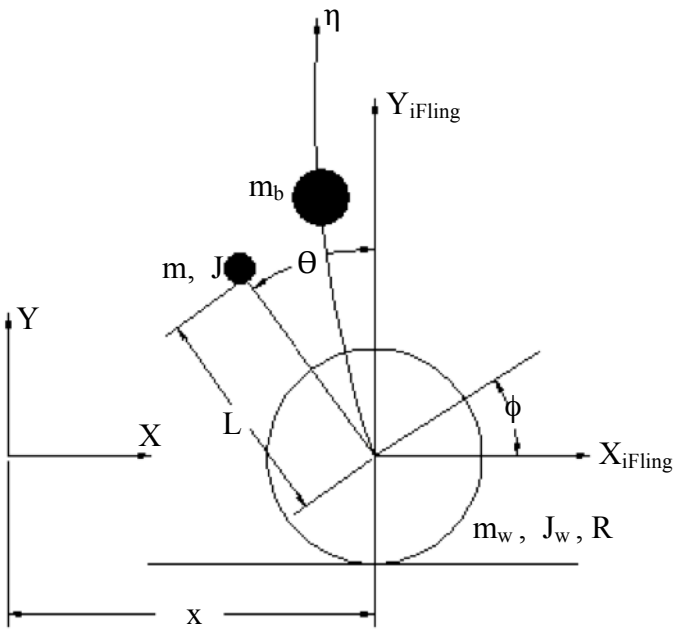


Figure 8: Model of iFling

3.1 Dynamics of iFling

iFling's dynamics is similar to an inverted pendulum on a cart except the motor input can directly control the position of the body. The center of mass of the body is above the pivot point, which makes the system unstable. Nevertheless, it can be stabilized with feedback control by linearizing the system about $\theta=0$. The following is a list of assumptions for the model of iFling

1. The ping-pong ball is assumed to stay attached to the throwing arm until it reaches the tip of the arm.
2. No-slip constraint is assumed for the rotation of the wheel.
3. The ping-pong ball is modeled as a point particle that moves on the throwing arm.
4. When $\theta=0$, the center of mass of the body is assumed to be directly above the rotational axis of the wheel.
5. Rolling friction is assumed to be negligible.

Figure 8 shows the model of iFling. ϕ is the angle describing the wheel rotation measured with respect to the x-axis, θ is the angle of the body measured with respect to the y-axis, and η is the parametric variable describing the curve of the arm. The throwing arm is attached to the body, so its angular position is also measured by θ. Thus, the generalized coordinates, q_i, are $[x\ \phi\ \theta\ \eta]$. R and c are the radius of the wheel and the ball. m, m_b, and m_w are the mass of the body, the ping-pong ball and the wheel respectively. Location of m is the center of gravity of the body, which has length, L, from the wheel axle. J, and J_w are the rotational inertia of the body, and the wheel. In order to simplify

the optimization procedure, a cubic parametric function is chosen to describe the shape of the throwing arm.

$$x_a = a_3\eta \tag{1}$$

$$y_a = b_1\eta^3 + b_2\eta^2 + b_3\eta + b_4, \tag{2}$$

where η is a scalar from 0 to p, and p is the parametric value corresponding to the arc length of the throwing arm.

In order to make sure the throwing arm does not go through the ground during simulation, torsion spring and damper are used to simulate the effect of a ground. The spring and the dampers are only activated, when the arm or the ball is below certain equilibrium. The ping-pong ball is prevented from penetrating iFling's base by using a similar setup as well. From figure 8, the position and velocity vectors of m_w, m and m_b are

$$r_w = x\hat{i}, \tag{3}$$

$$\dot{r}_w = \dot{x}, \tag{4}$$

$$r = \left[x - L\sin\theta\right]\hat{i} + \left[L\cos\theta\right]\hat{j}, \tag{5}$$

$$\dot{r} = \left[\dot{x} - L\dot{\theta}\cos\theta\right]\hat{i} + \left[-L\dot{\theta}\sin\theta\right]\hat{j}, \tag{6}$$

and

$$\begin{aligned} r_b = &\left[a_3\eta\cos\theta - \left(b_1\eta^3 + b_2\eta^2 + b_3\eta + b_4\right)\sin\theta + x\right]\hat{i} \\ &+ \left[a_3\eta\sin\theta + \left(b_1\eta^3 + b_2\eta^2 + b_3\eta + b_4\right)\cos\theta\right]\hat{j}, \end{aligned} \tag{7}$$

$$\begin{aligned} \dot{r}_b = &\left[a_3\dot{\eta}\cos\theta - a_3\eta\dot{\theta}\sin\theta - \left(3b_1\eta^2 + 2b_2\eta + b_3\right)\dot{\eta}\sin\theta - \left(b_1\eta^3 + b_2\eta^2 + b_3\eta + b_4\right)\dot{\theta}\cos\theta + \dot{x}\right]\hat{i} \\ &+ \left[a_3\dot{\eta}\sin\theta + a_3\eta\dot{\theta}\cos\theta + \left(3b_1\eta^2 + 2b_2\eta + b_3\right)\dot{\eta}\cos\theta - \left(b_1\eta^3 + b_2\eta^2 + b_3\eta + b_4\right)\dot{\theta}\sin\theta\right]\hat{j}, \end{aligned} \tag{8}$$

For a given dynamic system, the Lagrangian can be used to derive the equation of motions. The Lagrangian is defined as $\mathbf{L} = T - U$, where T is the kinetic energy and U is the potential energy of the system. For this particular case, the T and the U are

$$T = \frac{1}{2}\left(m\dot{r}^2 + J\dot{\theta}^2\right) + \frac{1}{2}\left(m\dot{r}_b^2\right) + \frac{1}{2}\left(m\dot{r}_w^2 + J_w\dot{\phi}^2\right) \tag{9}$$

$$\begin{aligned} U = &mgL\cos\theta + \frac{1}{2}k_a\left(\theta - \theta_0\right)^2 \\ &+ m_b g\left[\left(b_1\eta^3 + b_2\eta^2 + b_3\eta + b_4\right)\cos\theta + a_3\eta\sin\theta\right] + \frac{1}{2}k\left(\eta - \eta_0\right)^2 \end{aligned} \tag{10}$$

where k and k_a are spring constants. θ_0 and η_0 are the equilibrium positions of the springs and dampers. The Lagrange equation for generalized coordinate q_i is

$$\frac{\partial}{\partial t}\left(\frac{\partial \mathbf{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathbf{L}}{\partial q_i} = Q_i, \qquad i = 1,2,...,n, \tag{11}$$

where Q_i are the generalized inputs and n is the number of generalized coordinates. For this case the control input, u, is Q_i, which acts between the body and the wheel. Rayleigh dissipation function is used to model the dampers, whose general form is

$$D = \frac{1}{2}\sum_{l}^{n}\sum_{k}^{n} b_{lk}\dot{q}_l\dot{q}_k, \tag{12}$$

and for this case it is

$$D = \frac{1}{2} b \dot{\eta}^2 + \frac{1}{2} b_a \dot{\theta}^2, \tag{13}$$

where b and b_a are damping coefficients.

Adding D and the constraint, the modified Lagrange equation becomes

$$\frac{\partial}{\partial t}\left(\frac{\partial \mathbf{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathbf{L}}{\partial q_i} + \frac{\partial D}{\partial \dot{q}_i} - \sum_j^w \lambda_j a_{ij} = Q_i, \qquad i = 1,2,...,n, \tag{14}$$

where w is the number of constraints. There is only one constraint, which is the no-slip condition between the wheel and the ground. The constraint written in mathematical form is

$$R d\phi + dx = 0, \tag{15}$$

From the two above equations, it is clear that $a_{1\phi} = R$ and $a_{1x} = 1$. Then, solving λ_1 from the ϕ equation to obtain

$$\lambda_1 = \frac{u}{R} + \frac{J_w \ddot{\phi}}{R}. \tag{16}$$

No slip condition was also considered for the ball and the throwing arm. It was not used because η is not a physical length along the throwing arm. Define γ as the angular rotation of the ball with respect to the throwing arm, then the distance travel by the ball is $c\gamma$. The relationship between $c\gamma$ and η is the arc length formula, which is

$$c\gamma = \int_0^{\eta} \sqrt{\left(\frac{dx_a}{d\xi}\right)^2 + \left(\frac{dy_a}{d\xi}\right)^2}\, d\xi, \tag{17}$$

where ξ is the integration variable. Due to the square root, there is no explicit solution to the integral when using the cubic parametric equation. As a result, the γ equation and the η equation cannot be combined, and that leads to assumption number 3.

Thus, the generalized coordinates reduce from [x ϕ θ η] to [ϕ θ η]. The states of the system are $x = \begin{bmatrix} \phi & \dot{\phi} & \theta & \dot{\theta} & \eta & \dot{\eta} \end{bmatrix}^T$. The set of equations describing the motion of the ball and the iFling is in the form,

$$\begin{aligned} M(x)\frac{dx}{dt} &= f(u, x) \quad \text{on } 0 < t < t_f, \\ x &= x_0 \qquad \text{at } t = 0 \end{aligned} \tag{18}$$

where M is a symmetric matrix that varies with the states and the coefficient of the throwing arm, u is the control input, and t_f is the terminal time. The set of differential equations is nonlinear and does not have an analytical solution. The coefficients a_3, b_1, b_2, and b_3 describe the curvature of the arm shape. See appendix A.1 for the details of the equations of motion. Given an initial condition, the above equation can be used to simulation the system from $t = 0$ to $t = t_f$.

3.2 Projectile Motion

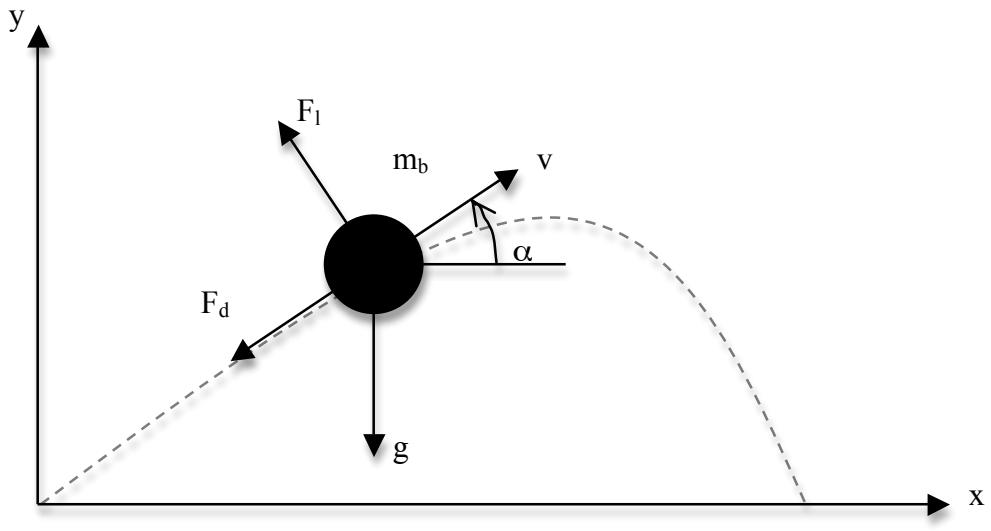


Figure 9: Projectile Motion

The derivation in this section focuses on the motion of the ball once it leaves the throwing arm. When the ball leaves the arm, it has not only velocity in x and y directions but also a backspin. This spin will generate lift force known as the Magnus force. Because of the spin, its drag is also higher than that of a non-rotating sphere. Modeling such force is complicated and requires wind tunnel data. Therefore, simplified models are used for the simulation and optimization. The following is a list of assumptions for the model.

6. When simulating the projectile motion, the ping-pong ball is assumed to be non-rotating; therefore, no lift is generated in flight.
7. Furthermore, the model used in optimization computation has no drag at all.

Figure 9 shows the free body diagram of the ball traveling in the air. The sum of forces in x and y direction are

$$m_b \frac{d^2x}{dt^2} = -F_d \cos\alpha - F_l \sin\alpha, \tag{19}$$

$$m_b \frac{d^2y}{dt^2} = -F_d \sin\alpha + F_l \cos\alpha - m_b g. \tag{20}$$

Since the ball is assumed to generate no lift, F_l can be neglected from the above equations. The drag of the ball is found using

$$F_d = \frac{1}{2}\rho A v^2 C_d, \tag{21}$$

where ρ is the density of the air, A is the projected area of the ball, v is the velocity of the ball, and C_d is the drag coefficient of the ball. Using $\frac{v_x}{v} = \cos\alpha$ and $\frac{v_y}{v} = \sin\alpha$, the governing equations can be simplified to

$$\frac{d^2x}{dt^2} = -\frac{\rho A v C_d v_x}{2m_b} \tag{22}$$

$$\frac{d^2y}{dt^2} = -\frac{\rho A v C_d v_y}{2m_b} - g, \tag{23}$$

where $v_x = \dot{x},\ v_y = \dot{y},\ and\ v = \sqrt{v_x^2 + v_y^2}$. Equations 22 and 23 are used in system simulation.

In the most ideal case where drag is neglected, there is an analytical solution to the projectile motion problem, which is $x = x_0 + v_x t$ and $y = y_0 + v_y t - \frac{1}{2} g t^2$. Using the y-equation to solve for *t* and substituting the result into the x-equation, the formula for the range of the throw is

$$x = v_x \left(\frac{v_y + \sqrt{v_y^2 + 2 y_0 g}}{g} \right) + x_0 . \tag{24}$$

Rename the variable *x* with *d*, and it is of the form $d = f\left(x_0, y_0, v_x, v_y\right)$. Equation 24 is used in the optimization calculation. x_0, y_0, v_x, *and* v_y are obtained using equations 7 and 8 by substituting the state values at the final time.

Chapter 4

Optimization

This chapter focuses on the main problem of this paper, which is to compute an optimal input sequence for the motors, and to determine the optimal throwing arm shape to be used on iFling to achieve an increase in range. The optimization presented in this paper is an iterative scheme. Using this scheme the final solution is obtained slowly from the initial guess. The shape optimization is based on the ball's exit condition from the arm, which wraps around an adjoint based input optimization.

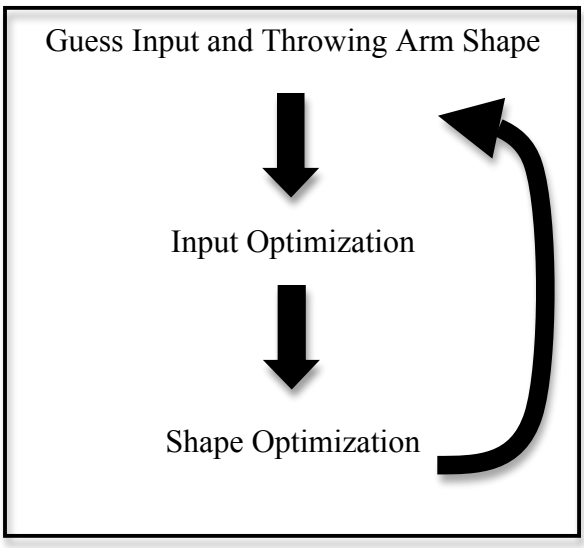


Figure 10: Overall optimization scheme

4.1 Input Optimization

The iFling system is governed by a set of continuous-time state equations given in chapter 3. The cost function J is defined to measure the trajectory of the throw,

$$J = \frac{1}{2}\int_0^{t_f} x^T Q_x x + u Q_u u \, dt + \frac{1}{2} x^T\left(t_f\right) M^T Q_T M x\left(t_f\right) - \frac{1}{2} d Q_d d \tag{25}$$

where $Q_x \geq 0, Q_u > 0, Q_T \geq 0,$ and $Q_d \geq 0$. The choices of Q_x, Q_u, Q_T, and Q_d represents mathematically the objective one wants to achieve. The first two terms in the integral influence the state and the input during the time window from 0 to t_f. The third term influences the terminal condition of the iFling system, and the fourth term influences the distance of the throw, where d is the range of the throw. In short, what we want is to minimize J with respect to the control sequence, u. Note by adding a negative sign in front of the fourth term is equivalent of maximizing the distance of the throw.

Second consider how the system change if there is a small perturbation in the control input, u'. This small perturbation in input at every single time step causes small perturbation x' to the state x. Such perturbations are governed by the perturbation equation,

$$\begin{aligned} \mathcal{L}x' &= Bu' \quad \text{on } 0 < t < t_f, \\ x' &= 0 \qquad\quad \text{at } t = 0, \end{aligned} \tag{26}$$

where operator $\mathcal{L} = \left(M\frac{d}{dt} - A\right)$ and matrix $A(t)$ and B are obtained from linearization of equation 18 about the trajectory $x\left(t, u, a_3, b_1, b_2, b_3\right)$, which is obtained with a forward

march of the state equations The perturbation *x'* and *u'* also cause perturbations in the cost function, which is

$$J' = \int_0^{t_f} x^T Q_x x' + u Q_u u' \, dt + x^T\left(t_f\right) M^T Q_T M x'\left(t_f\right) - d Q_d \left(\frac{\partial d}{\partial x_i}\right) x'\left(t_f\right), \quad i = 1,2,..n \tag{27}$$

Using the perturbed cost function, *J'*, one can derive the gradient $\frac{dJ}{du}$ explicitly. First, define the weighted inner product $\langle\langle a,b \rangle\rangle = \int_0^{t_f} a^T b \, dt$. Using the weighted inner product, the adjoint identity

$$\left\langle\left\langle r, \mathcal{L} x' \right\rangle\right\rangle = \left\langle\left\langle \mathcal{L}^* r, x' \right\rangle\right\rangle + b \tag{28}$$

can be derived using integration by parts, which gives $\mathcal{L}^* = -\left(M^T \frac{d}{dt} + A^T\right)$ and $b = \left[r^T M x'\right]_0^{t_f}$. The detail of the derivation is given in appendix A.2.

Now we define the adjoint function, driven by the system states, such that

$$\begin{aligned} \mathcal{L}^* r &= Q_x x && \text{on } 0 < t < t_f, \\ r &= Q_T M x - \left(M^{-1}\right)^T \left(\frac{\partial d}{dx_i}\right)^T Q_d d && \text{at } t = t_f. \end{aligned} \tag{29}$$

Substitute equation 29 into the adjoint identity to obtain

$$\int_0^{t_f} r^T B u' \, dt = \int_0^{t_f} x^T Q_x x' \, dt + x^T\left(t_f\right) M^T Q_T M x'\left(t_f\right) - d Q_d \frac{\partial d}{\partial x_i} x'\left(t_f\right) \tag{30}$$

Then combine equation 30 and equation 27 to get

$$J' = \int_0^{t_f} \left(B^T r + Q_u u\right)^T u' \, dt = \left\langle \left\langle \frac{dJ}{du}, u' \right\rangle \right\rangle. \tag{31}$$

The gradient of J with respect to u is given by

$$\frac{dJ}{du} = B^T r + Q_u u. \tag{32}$$

This gradient can be used to update the entire input sequence at each input optimization iteration using steepest descent or conjugate gradient. The size of the step for each iteration is determined using Brent's method.

4.2 Shape Optimization

The objective of the shape optimization is to determine the coefficients, a_3, b_1, b_2, and b_3, of the parametric equation to achieve the longest throw. a_3 for the initial guess is assumed to be positive because most athletic throwing arms curve forward in the direction of the throw. Examples are Lacrosse stick and Jai Lai basket. The basic shape optimization scheme is similar to the input optimization, where a gradient is needed to maximize the distance of the throw. Since equation 24 gives an explicit relationship between the range of the throw and the slope of the tip of the throwing arm, the gradient can be analytically determined by taking the partial derivatives of d with respect to the coefficients. The gradient is,

$$\nabla d = \left[\frac{\partial d}{\partial a_3} \quad \frac{\partial d}{\partial b_1} \quad \frac{\partial d}{\partial b_2} \quad \frac{\partial d}{\partial b_3} \right]. \tag{33}$$

In short, the shape optimization tries to find out a new throwing arm shape for each iteration, so the distance of the throw is longer than that of the previous iteration.

4.3 Algorithm

The input optimization will find the best input for a given shape with respect to the cost function. Then the shape optimization will find a better shape with respect to the distance of the throw. The following is an outline of the algorithm for optimizing the control input and the throwing arm shape,

1. Give an initial condition
2. Guess a control input and throw arm shape
3. March the state equations forward
4. March the adjoint equations backward
5. Compute the gradient given in equation 32
6. Find the step size using Brent's method
7. Update control input, u
8. Repeat step 5 to 7 until a minimum is reached
9. Compute the gradient given in equation 33
10. Find the step size using Brent's method
11. Update throwing arm shape
12. Repeat step 3 to step 11 until a minimum is reached

The result of the optimization is presented in chapter 5.

Chapter 5

Results and Discussion

In this chapter, I will present the result of the optimization along with the mechanical properties and initial conditions used. A simulation of the initial guess and the optimal solution is shown and discussed in details.

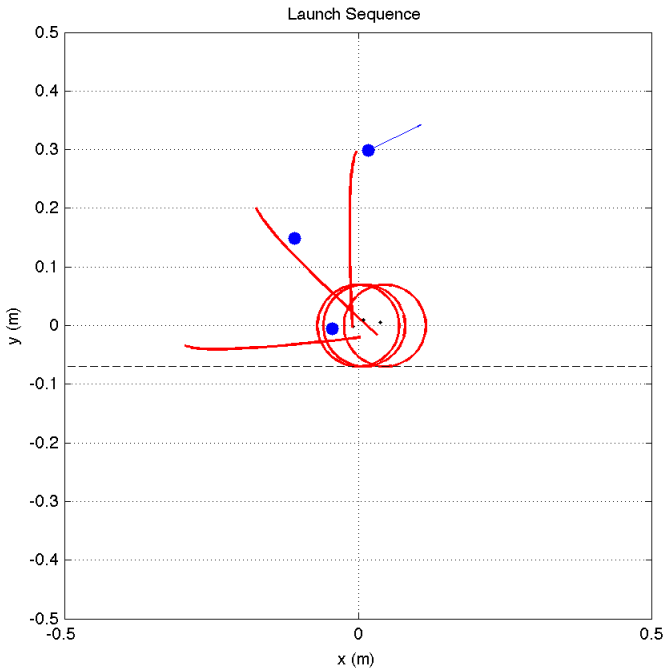


Figure 11: Ball launching sequence

5.1 Mechanical Properties and Initial Conditions

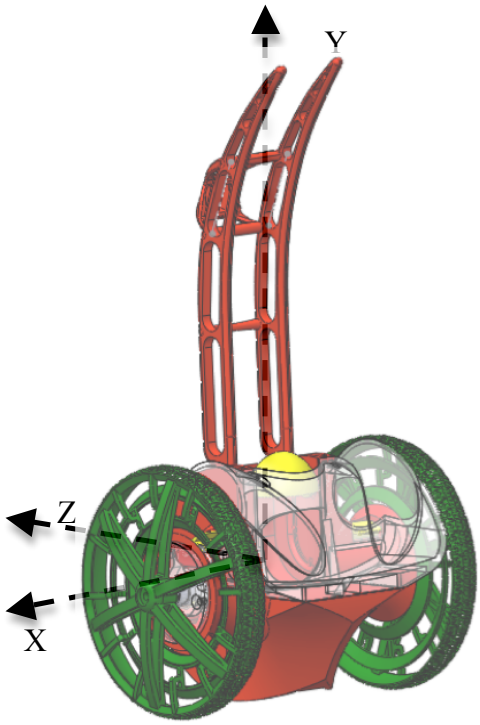


Figure 12: Coordinate system used for finding mass properties

5.1.1 Mechanical Properties

In order to obtain accurate mass properties, a 3D model is drawn in Solidworks. Components with uniform density are calculated with their corresponding material properties. Most of the parts are made of ABS plastics with a few acrylic parts. However, certain components such as the motors, batteries, circuit board, radio receivers and servomotors do not have uniform density and are difficult to model. Therefore, their weights are measured separately and manually entered into the 3D model. The center of gravity and the weight of the body are obtained using the built in Mass Properties Calculator. The coordinate system is arranged so the x-axis is parallel to the wheel's axis

of rotation, the y-axis points up vertically, and the z-axis follows the right hand rule. See figure 12.

The iFling uses standard size ping-pong balls. It has a radius of $0.02\ m$, weights $0.0027\ kg$. The radius of the wheel is $0.07\ m$, weights $0.0606\ kg$ and the rotational inertia is $1.9176 \times 10^{-4}\ kg\,m^4$. The weight of the body of the robot is $0.5338\ kg$, and its rotational inertia about the center of mass is $1.976 \times 10^{-3}\ kgm^2$. When $\theta=0$, the center of mass of the body is located at $0.0094\ m$ above the rotational axis of the wheel.

5.1.2 Initial Conditions

The robot starts at rest in the lay-down position, and the ball is positioned $0.045\ m$ down the throwing arm measuring from the rotational axis of the body.

5.1.3 Optimization Constraints

With this input optimization scheme, one can set many different constraints, such as the position or velocity of the arm or the position of the ball at the terminal time and many others. In order to simplify the optimization procedure, I used the following constraints to obtain a reasonable solution.

1. The time window is chosen to be $0.5\ s$.
2. The arc length of the throwing arm is $0.3\ m$.
3. At $t=t_f$ the ball reaches the end of the throwing arm.
4. The applied torque cannot exceed $\pm 0.3\ Nm$.

The terminal condition for the position of the arm was not set because it is not possible to know before hand which position will result in the longest throw. The following section shows the result of the optimization and simulation.

5.2 Input Sequence

Figure 13 shows the initial guess and the optimal input solution. The two inputs are very different, but they are both in between the assumed ± 0.3 *Nm* motor limit. The desire input magnitude can be adjusted by tuning the Q_u parameter. In order to throw the ball, the control input must move the robot forward to allow the ball to roll down the arm then backward to swing the arm. Therefore, an obvious solution to the throwing problem would be an input that is linear with respect to time and with a negative slope. Using this intuition, the initial guess for the input starts at 0.23 *Nm* and decreases linearly with time to -0.25 *Nm*. It does not result in a launch condition that is close to the optimal solution. This guess is tuned to satisfy the first optimization constraint for the purpose of comparison with the optimal solution.

On the other hand, the optimal solution is not a straight line, and has much more features. Section A of the input is similar to the first portion of the initial guess, which indicates the robot is moving forward to allow the ball to roll towards the end of the arm. Section B is to slow down the robot to manipulate the ball to a good location before initializing the fast rotation of the arm. Section C is the arm swing up and consists of a drop from almost zero torque to high torque. The manipulating of the ball's position before launch and the high torque to rotate the body is the essence of input optimization. The optimization takes into account the above two factors to compute the input sequence

to produce an optimal launch condition. If the ball's position before the throw is not optimized, then it can result in a short throw. In general, a larger torque is more helpful towards the distance of the throw, which is apparent in figure 13. The magnitude of section C of the optimal solution is larger than the throwing portion of the initial guess. However, a high motor torque at the end can also make the ball to overshoot before the terminal time. It is a balance between throwing distance and the terminal condition. Figure 11 illustrates the launch sequence.

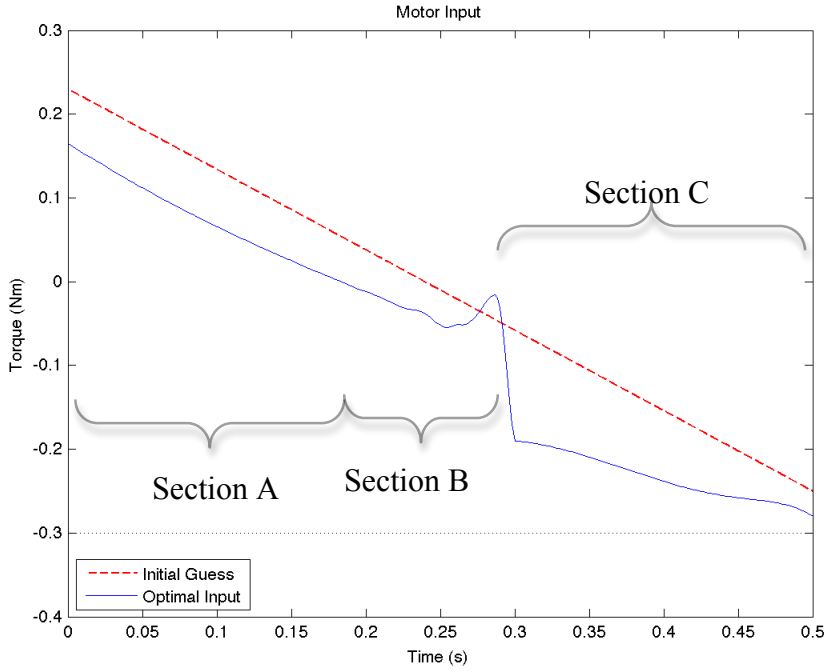


Figure 13: Result of the input optimization

5.3 Throwing Arm Shape

The constraints for the throwing arm are the cubic function constraint and the arc length. The arc length is set to be 0.3 *m*. The parametric value, *p*, varies to satisfy this

constraint. The initial guess for the coefficients, a_3, b_1, b_2, and b_3, were 0.01, 0.002, -0.024, and 0.15. These numbers were chosen arbitrarily. Using the shape optimization scheme, the refined coefficients were computed to be 0.0088, 0.0014, -0.0247, and 0.01493. Both the initial and the refined arm shape are shown in figure 14. Even though all four coefficients changed, the new shape is only slightly different from the original guess. The only noticeable change is the shape at the tip. The overall shape seems to be more bent, which allows the ball to leave the throwing arm at a lower launch angle. After trying many different initial guesses, the computed shape never changes drastically from the original guess. Therefore, I believe this method is only capable of finding a refined shape for a particular input sequence, not a complete redesign of the throwing arm.

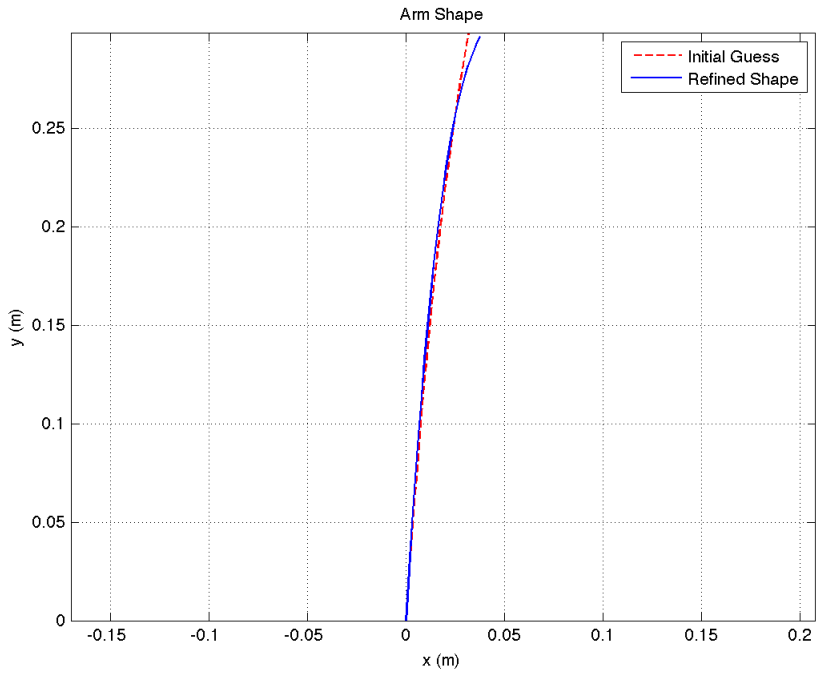


Figure 14: Result of shape optimization

5.5 Simulation

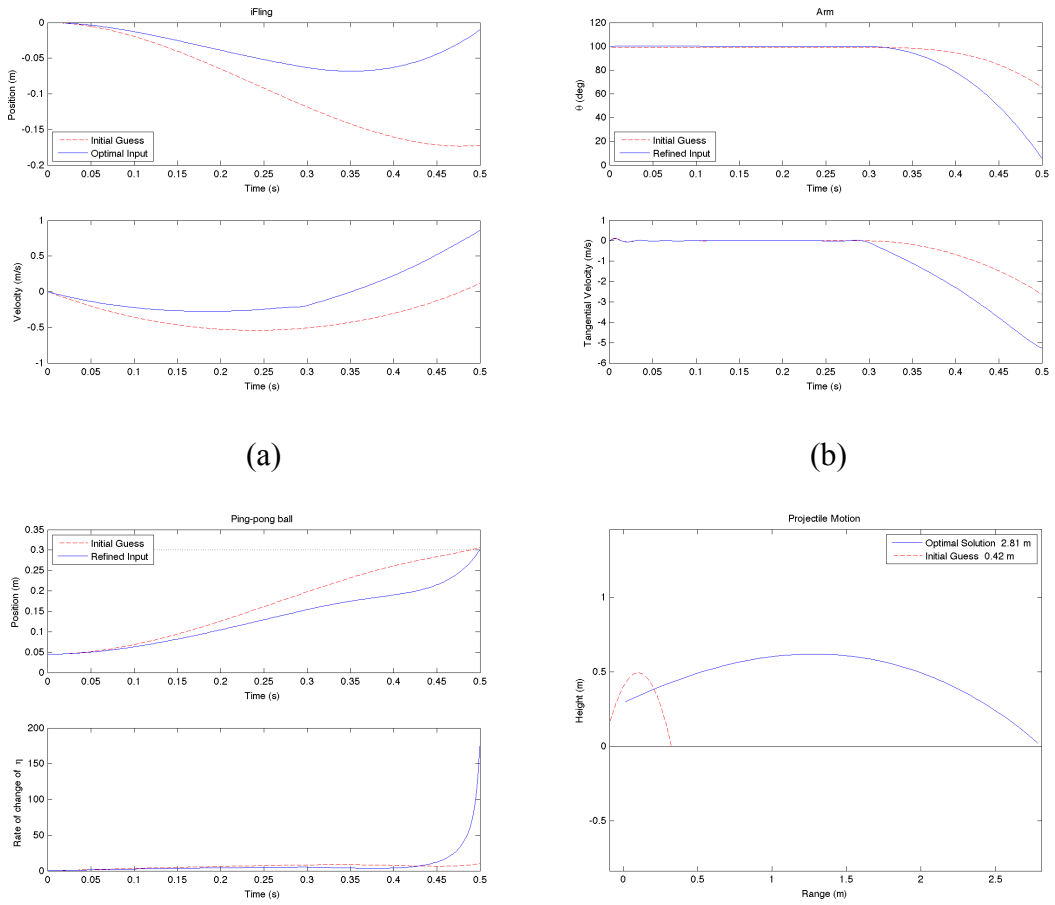


Figure 15: iFling simulation using initial guess and optimal solution

Figure 15 presents the simulation of the system from $t = 0$ to $t = 0.5\ s$. Figure 15.a shows the movement and the velocity of iFling in time. The solid line is the optimal solution and the dash line is the initial guess. Note since ϕ is positive in the counterclockwise direction, the forward motion corresponds to negative values in the figure. Using the optimal solution, the iFling only travels about 0.01 m forward, which is almost stationary, whereas the guess results in a longer displacement of $0.18m$. For both

cases the velocities behave similarly. The robot is moving backward at the terminal time for both cases. An interesting point is that optimal solution arrives at a much higher backward velocity, and that indicates a more powerful throw.

Figure 15.b shows the motion of the throwing arm. The positive direction for angle θ is the same as ϕ. Therefore, when the arm swings forward, the plot will show the angle changes from positive to negative. The transient oscillatory behavior is due to the torsion spring and damper that are used to simulate the ground. Focusing on the optimal solution, the arm starts at around 100 degrees and remains at rest, until $t \cong 0.29\ s$ then it starts to rotate forward to fling the ball. The terminal angle is about 6 degrees for the optimal solution. The terminal tangential velocity at the tip for the optimal solution is about two times more than the initial guess, which contributes to the longer throwing distance.

Figure 15.c describes the movement of the ping-pong ball on the throwing arm and the time rate of change of η. Both simulations show the ball reaches the end of the arm at $t=t_f$. Looking at the solid line, the ball slowly rolls down the track until $t \cong 0.43\ s$. At that time, it moves rapidly towards the end of the arm and reaches the tip at $t=t_f$. Even though the rate of change of η is not the physical velocity of the ball traveling down the arm, it gives a general trend to the actual speed. It is clear from figure 15.c that the rate of change of η at terminal time is a lot higher than the initial guess. I believe this distinction is due to the higher motor torque in section C of figure 13. Together with the increase in tangential velocity, and the release position, they are the main factors in the improvement of the throwing distance.

Figure 15.d displays the motion of the ping-pong ball after $t > 0.5\ s$. The projectile motion presented in this figure includes the pressure drag of a smooth sphere. The effect of the drag is apparent because the trajectory is asymmetric, whereas an ideal projectile motion has a symmetric parabolic trajectory. The launch velocity and angle for the guess and optimal solution are 2.75 *m/s* and 73.9 degrees, and 6.25 *m/s* and 25.5 degrees respectively. The resulting ranges are 0.42 *m* and 2.81 *m*. The optimal solution gives a significant improvement over the initial guess. This increases is mainly because of the much higher launch speed and lower launch angle.

5.5 Discussion

I have tried different initial conditions to see if they will converge to the same input solution. If the adjustment is small, then I get the same solution, but if the change is relatively larger, then the solution is different. Therefore, I believe this method is only capable of finding a local minimum. It is expected because the derivation does not insure a global minimum solution. Despite this disadvantage, this method is still very powerful simply because it gives a reasonable solution to a complex problem. Even though the results are different when the initial conditions are changed, the solutions for the input all have similar shape and features. They all start from a positive value and decease with time, then a non-smooth region follows next. After that, the large motor torque is used to perform the swing up motion.

On the other hand, the shape variation from one to another is small. The reasons that the change is small are because the shape is limited to a cubic shape and the arc length is limited. Despite this limitation, the improvement in throwing distance due to

the throwing arm change is comparable to the improvement due to new input. Therefore, the overall improvement in throwing distance dependents on both the motor inputs and the shape of the arm.

Because of the assumptions and simplifications made in the derivation, the models used in optimization and simulations are ideal, so the distance of the throw shown in simulation may seem too much for a small robot. In reality, the actual distance will be shorter due to more drag, rolling friction, friction of the arm, motor damping, and many other factors.

Chapter 6

Conclusion

iFling has gone through several design iterations, the current design iteration focuses on the ball pick up and ball throwing capabilities. This paper gave an introduction of the iFling program, explained the features of the robot design, derived the equations of motion for the system, and formulated a method for finding the control input for the motors and the shape for the throwing arm. Results of the optimization and simulation shows a significant increase in distance and indicates the input and the shape both played significant parts in the throwing distance. The adjoint-based calculation is capable of finding a local optimal solution to the problem. On the other hand the shape optimization is capable of finding a refined answer to the initial guess, not a redesign. Nevertheless, the input optimization method is powerful because it is not restricted to only this problem. It can be used to optimize many different systems provided the state equations and cost functions are known.

Even though this paper provides a theoretical solution to the problem, there are still a lot of possibilities for improving this mathematical solution. First the iFling dynamic model can be more refined by adding friction, and air resistance. The projectile model can be more realistic by adding pressure drag and the effect of the rotation of the ball. The function describing the curve can be revised by using a Legendre polynomial to generate a more general curve, which will lead to a better optimal solution. The

optimization strategy can also be improved by combining the shape and input calculations into one single optimization loop, which may reduce the time, computing resources, and provide a true optimal solution.

Implementation of the optimized solution is an obvious next step in this project. In order to implement the solution, one would need to model the dynamics of the motor and approximate the control input with piecewise mathematical functions to save the memory space of the onboard electronics. There are two ways in implementing the solution. First option is open loop control. However, due to the un-modeled dynamics, the open loop method required more fine-tuning to make it to work. A second way would be to implement a feedback control on top of the open loop control to force iFling to follow the computed trajectory, which is a lot more difficult. That being said I believe a more feasible and easier way would be to redesign the system to make it lighter, so the weight to torque ratio would be higher. Then use on/off control and manually adjust the run time for each segment to achieve the throw.

Appendix

A.1

Equations of Motion for iFling

$$M(x)\frac{dx}{dt} = f(u, x) \quad \text{on } 0 < t < t_f,$$
$$x = x_0 \qquad \text{at } t = 0$$

$$x = \begin{bmatrix} \phi & \dot{\phi} & \theta & \dot{\theta} & \eta & \dot{\eta} \end{bmatrix}^T$$

M(x)=[1, 0, 0, 0, 0, 0;
0, M(2,2), 0, M(2,4), 0, M(2,6);
0, 0, 1, 0, 0, 0;
0, M(4,2), 0, M(4,4), 0, M(4,6);
0, 0, 0, 0, 1, 0;
0, M(6,2), 0, M(6,4), 0, M(6,6)];

M(2,2)=Jw + R^2*m + R^2*mb + R^2*mw;

M(2,4)=R*(b4*mb*cos(X(3)) + L*m*cos(X(3)) + b3*X(5)*mb*cos(X(3)) +
a3*X(5)*mb*sin(X(3)) + b1*X(5)^3*mb*cos(X(3)) + b2*X(5)^2*mb*cos(X(3)));

M(2,6)=R*mb*(3*b1*sin(X(3))*X(5)^2 + 2*b2*sin(X(3))*X(5) - a3*cos(X(3)) + b3*sin(X(3)));

M(4,2)=R*(b4*mb*cos(X(3)) + L*m*cos(X(3)) + b3*X(5)*mb*cos(X(3)) + a3*X(5)*mb*sin(X(3)) + b1*X(5)^3*mb*cos(X(3)) + b2*X(5)^2*mb*cos(X(3)));

M(4,4)=m*L^2 + mb*a3^2*X(5)^2 + mb*b1^2*X(5)^6 + 2*mb*b1*b2*X(5)^5 + 2*mb*b1*b3*X(5)^4 + 2*mb*b1*b4*X(5)^3 + mb*b2^2*X(5)^4 + 2*mb*b2*b3*X(5)^3 + 2*mb*b2*b4*X(5)^2 + mb*b3^2*X(5)^2 + 2*mb*b3*b4*X(5) + mb*b4^2 + J;

M(4,6)=mb*(- a3*b4 + 2*a3*b1*X(5)^3 + a3*b2*X(5)^2);

M(6,2)=R*mb*(3*b1*sin(X(3))*X(5)^2 + 2*b2*sin(X(3))*X(5) - a3*cos(X(3)) + b3*sin(X(3)));

M(6,4)=mb*(- a3*b4+ 2*a3*b1*X(5)^3 + a3*b2*X(5)^2);

M(6,6)=mb*(a3^2 + 9*b1^2*X(5)^4 + 12*b1*b2*X(5)^3 + 6*b1*b3*X(5)^2 + 4*b2^2*X(5)^2 + 4*b2*b3*X(5) + b3^2);

f(1,1)=X(2);

```
f(2,1)=L*R*X(4)^2*m*sin(X(3)) - u(i) - 2*R*b2*X(6)^2*mb*sin(X(3)) +
R*b4*X(4)^2*mb*sin(X(3)) - 2*R*b3*X(6)*X(4)*mb*cos(X(3)) -
2*R*a3*X(6)*X(4)*mb*sin(X(3)) - R*a3*X(4)^2*X(5)*mb*cos(X(3)) -
6*R*b1*X(6)^2*X(5)*mb*sin(X(3)) + R*b3*X(4)^2*X(5)*mb*sin(X(3)) +
R*b1*X(4)^2*X(5)^3*mb*sin(X(3)) + R*b2*X(4)^2*X(5)^2*mb*sin(X(3)) -
4*R*b2*X(6)*X(4)*X(5)*mb*cos(X(3)) - 6*R*b1*X(6)*X(4)*X(5)^2*mb*cos(X(3));

f(3,1)=X(4);

f(4,1)=- 2*X(4)*mb*a3^2*X(6)*X(5) - 6*mb*a3*b1*X(6)^2*X(5)^2 -
2*mb*a3*b2*X(6)^2*X(5) - g*mb*cos(X(3))*a3*X(5) -
6*X(4)*mb*b1^2*X(6)*X(5)^5 - 10*X(4)*mb*b1*b2*X(6)*X(5)^4 -
8*X(4)*mb*b1*b3*X(6)*X(5)^3 - 6*b4*X(4)*mb*b1*X(6)*X(5)^2 +
g*mb*sin(X(3))*b1*X(5)^3 - 4*X(4)*mb*b2^2*X(6)*X(5)^3 -
6*X(4)*mb*b2*b3*X(6)*X(5)^2 - 4*b4*X(4)*mb*b2*X(6)*X(5) +
g*mb*sin(X(3))*b2*X(5)^2 - 2*X(4)*mb*b3^2*X(6)*X(5) - 2*b4*X(4)*mb*b3*X(6) +
g*mb*sin(X(3))*b3*X(5) + u(i) - ba*X(4) - ka*X(3) + ka*th0 + L*g*m*sin(X(3)) +
b4*g*mb*sin(X(3));

f(5,1)=X(6);

f(6,1)=mb*a3^2*X(4)^2*X(5) - g*mb*sin(X(3))*a3 - 18*mb*b1^2*X(6)^2*X(5)^3 +
3*mb*b1^2*X(4)^2*X(5)^5 - 18*mb*b1*b2*X(6)^2*X(5)^2 +
```

```
5*mb*b1*b2*X(4)^2*X(5)^4 - 6*mb*b1*b3*X(6)^2*X(5) +
4*mb*b1*b3*X(4)^2*X(5)^3 + 3*b4*mb*b1*X(4)^2*X(5)^2 -
3*g*mb*cos(X(3))*b1*X(5)^2 - 4*mb*b2^2*X(6)^2*X(5) +
2*mb*b2^2*X(4)^2*X(5)^3 - 2*mb*b2*b3*X(6)^2 + 3*mb*b2*b3*X(4)^2*X(5)^2 +
2*b4*mb*b2*X(4)^2*X(5) - 2*g*mb*cos(X(3))*b2*X(5) + mb*b3^2*X(4)^2*X(5) +
b4*mb*b3*X(4)^2 - g*mb*cos(X(3))*b3 - b*X(6) - k*X(5) + k*et0;
```

A.2

$$\left\langle \left\langle r, \mathcal{L}x' \right\rangle \right\rangle = \left\langle \left\langle \mathcal{L}^* r, x' \right\rangle \right\rangle + b$$

$$\left\langle \left\langle r, \mathcal{L}x' \right\rangle \right\rangle = \int_0^{t_f} r^T \left(M \frac{dx'}{dt} - Ax' \right) dt \qquad \text{integrating by parts}$$

$$= \int_0^{t_f} r^T M \frac{dx'}{dt} - r^T Ax'\, dt \qquad u = r^T \quad dv = M \frac{dx'}{dt}$$

$$= \int_0^{t_f} -\frac{dr^T}{dt} Mx' - r^T Ax'\, dt + r^T Mx' \Big|_0^{t_f} \qquad du = \frac{dr^T}{dt} \quad v = Mx'$$

$$= \int_0^{t_f} \left(-M^T \frac{dr}{dt} - A^T r \right)^T x'\, dt + r^T Mx' \Big|_0^{t_f}$$

$$= \left\langle \left\langle \mathcal{L}^* r, x' \right\rangle \right\rangle + b,$$

where

$$\mathcal{L}^* = -\left(M^T \frac{d}{dt} + A^T \right)$$

$$b = r^T Mx' \Big|_0^{t_f}$$

Bibliography

[1] Bewley, T.R., Numerical Renaissance: Simulation, Optimization, and Control. Renaissance Press, San Diego, CA, 2009 http://numerical-renaissance.com/Numerical_Renaissance.html

[2] Brody, Howard, Rodney Cross, and Crawford Lindsey. *The Physics and Technology of Tennis*. Solana Beach, Calif.: Racquet Tech Pub., 2002.

[3] Bryson, Arthur E., and Yu-Chi Ho. *Applied Optimal Control: Optimization, Estimation, and Control*. Levittown, PA. Taylor & Francis, 1975

[4] Cavender, Andrew. "iFling: Introduction, Equations of Motion, and Assorted Stuff", University of California, San Diego 2010

[5] Greiner, Walter. *Classical Mechanics: Systems of Particles and Hamiltonian Dynamics*. New York: Springer, 2003

[6] How, Deyst, Aerospace Dynamics, Spring 2003. (Massachusetts Institute of Technology: MIT OpenCouseWare), http://ocw.mit.edu (Accessed May 1, 2010). License: Creative Commons BY-NC-SA

[7] Landau, L. D., and E. M. Lifshitz. *Mechanics*. Oxford: Butterworth-Heinemann, 1999.