

Design and Control of a Three-Link Serial Manipulator for Lessons in Particle Dynamics

Mark A. Minor, Assistant Professor 2nd LT. Kent Jensen, US Air Force Youngshik Kim, Graduate Student
Department of Mechanical Engineering
University of Utah, Salt Lake City, UT 84106
minor@mech.utah.edu

Abstract

Design, control, and performance of a ball-throwing robot are examined in this paper. The objective of this project is to provide an interactive ball-throwing robotic arm for illustrating the roles of engineers and computer scientists in the design and usage of such a system to high school or pre-engineering students. Activities in particle dynamics and trajectory calculation will provide basic hands on engineering experience and the opportunity to interact with the robot. In order to effectively provide this activity, the robot must consistently throw the ball from a known point with a desired velocity. This requires a minimum of a two-link manipulator with control strategies sufficient to converge two joint positions and velocities simultaneously. Due to limited micro-controller computational resources, feed-forward torques are calculated offline based on 3rd order cubic spline trajectories. Feedback compensation for position and velocity error is then examined and compared for ball throwing accuracy and precision to a technique supplementing the previous controller with acceleration error compensation. Experimental results are presented that illustrate the improved accuracy and reduced repeatability the later technique. Gripper design providing consistent hold on the ball and rapid release is also examined.

1. Introduction

Robotics research is consistently gaining attention amongst students entering engineering curriculum. To help pre-engineering and high school students gain a better understanding of engineering disciplines and make an educated decision about their career choice, the University of Utah is developing an interactive ball throwing robotic arm, Figure 1. The robot is designed to illustrate the roles of the mechanical engineering, electrical engineering, and computer science disciplines within a robotic system, as well as provide interactive engineering activities based upon the particle dynamics of a ball in motion. To be precise, the students will compete in teams to design a trajectory for the ball, program the initial velocity of the trajectory into the robot, and then the robot will throw the ball.

In order to simplify the trajectory calculations, the robot is designed to release the ball at a predefined point, indicated by the position of the robot in Figure 1. The students will measure the distance from the target to that point and calculate a feasible initial velocity. The main objective of this research is thus to develop a robotic arm capable of achieving a wide range of initial velocities while releasing from that same initial point.

Key to achieving this goal is the kinematic structure of the robot, appropriate selection of actuators and sensors, motion planning techniques, and control algorithms. The kinematic structure must provide sufficient Degrees of Freedom (DOF) such that initial velocity and position requirements can be achieved. Yet, excessive DOF will result in an arm that is too heavy, too complex, and not capable of achieving the desired range of velocities.

As Figure 1 indicates, a two-link arm is selected with dimensions comparable to an adult human arm. Given these dimensions, the actuators must be capable of

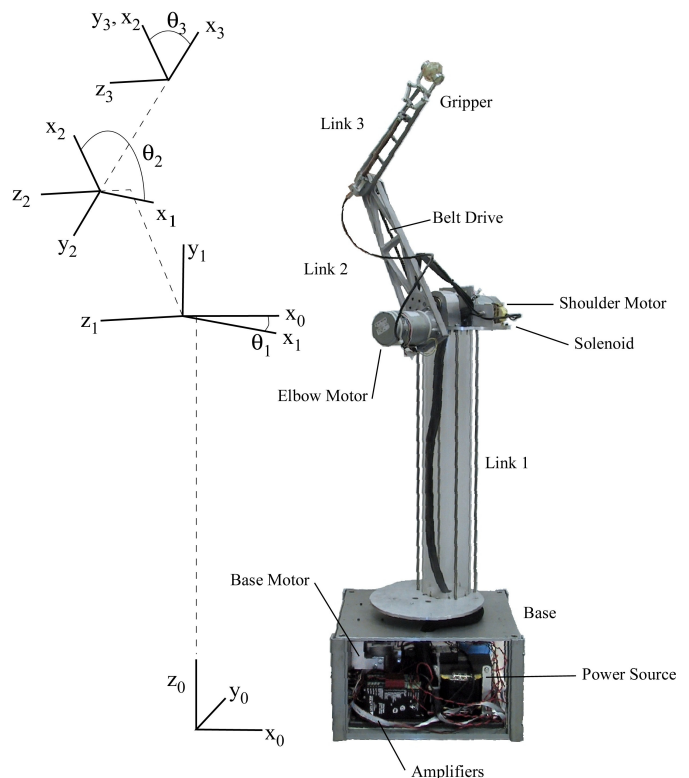


Figure 1. Ball throwing robot and kinematic diagram.

generating the necessary dynamic loads to accommodate the requirements of the motion planner and control algorithm for the desired pitch velocity.

Planning of the robot motion must consider the distinct operating regimes of the system. The robot must first pickup a ball, windup and throw the ball, and then decelerate. A 3rd order cubic spline is thus used to approximate the angle, velocity, and acceleration components of the trajectory. Computational resources are limited, and these trajectories are calculated offline prior to throwing the ball. Limited computational resources also hinder the control algorithm. Hence, two techniques based upon applying a feed forward torque for counteracting non-linearities are examined. The feed-forward component uses offline calculations based on the system model and planned motion to predict the necessary joint torques. Feedback control techniques based upon position and velocity error compensation, as well as these supplemented with acceleration error compensation, are then compared for minimizing the ball throwing error.

In Section 2, we first examine existing throwing robots and compare our kinematic structure and control system to those found in the literature. In Section 3, we present the robot itself and describe its functionality. Section 4 focuses on the planning and control algorithms that we examined, and Section 5 presents testing and performance results. Conclusions are stated in Section 6.

2. Background

A survey of similar projects uncovers many common threads, but different objectives. There are many robots in the literature that throw objects using various throwing movements. They span a broad range of type and purpose. The simplest are single DOF throwing robots built both commercially and for research purposes. The next common category is the two-link category, which includes traditional rigid planar pairs and flexible member manipulators. Many robots have more DOF and have been built to catch as well as throw balls and even juggle multiple balls. The juggling robots encompass their own group, varying broadly in approach and form. The ideas encompassed in these differing projects do encounter similar challenges in implementation.

Northrop [1] uses a one-link robot to feed parts in an assembly line by throwing them and manipulating their landing position and orientation. Once the part is identified and the trajectory planned, then the motor must execute the command precisely. Aboaf [2] explored the advantage of task-level control when throwing balls with a one-link robot. The system uses a vision system to measure the ball's actual landing position, which can then

be compared against the desired landing position. They explored two methods of using this error to improve performance. Their Fixed-Model method learns by applying an inverse model to the result. Their Refined Model procedure manipulates both the model and the command signal to get the desired results. Kato [3] explores the control of a two-link planar pair robot using an adaptive non-linear controller that modifies the release time. He shows improved results through release time manipulation. Other common techniques include sliding mode control [4] and feedback linearization [5]. Due to cost and computational limitations, though, these types of control strategies are not feasible. Primarily for this reason, more cost effective feed-forward linearization with feedback compensation is examined.

While the Ball-Throwing Robot shares objectives and requirements with the preceding projects, one feature not shared is that the Ball-Throwing Robot has a goal of converging two positions and two velocities at one fixed point in space and time. This is important for allowing the students to always calculate their velocity trajectories from a common release point. In contrast, the other systems are flexible in their release and catching points, and they typically have the benefit of more computational power.

3. Mechanical Structure

Derivation of the robot design was driven by several factors. As an interactive demonstration robot, it was necessary that the robot be mobile, lightweight, and capable of accurate and repeatable motion. Towards this end, the design indicated in Figure 1 was created. The robot is entirely self-contained, lightweight and portable, fail-safe, and capable of throwing accurately. These features are feasible because of the designs for the base, arm linkage, and gripper.

Base

The base of the robot actually serves many purposes. Probably most importantly, it is required to serve as a stable platform to support the robot. Since dimensions of the base are limited by the need to also be easily transported, the robot's center of gravity must be as low as possible. Towards this end the majority of the robot's mass is designed into the base. Thus, the power supplies, amplifiers, one joint motor, and all support electronics are contained therein.

The joint motor housed in the base controls rotation of the Link 1, described by the variable θ_1 , as indicated in Figure 1. Link 1 elevates the shoulder of the robot sufficiently to prevent the gripper of the robot from colliding with the base and also stores balls that will be thrown by the robot.

The transparent tube that stores balls and supports the shoulder is actually offset from the center of rotation. This allows the gripper to move in a plane containing the axis of Link 1 rotation. The angle θ_l determines the angle of this plane and the direction that the ball will be thrown. DH parameters for Link 1 are shown in Table 1. Inertial parameters are neglected since this link remains stationary during the throwing motion.

Arm Linkage

Mounted at the end of Link 1 is a platform that supports the arm. The arm consists of a two link serial chain with a gripper mounted at the distal end. The parameters for the linkage are shown in Table 1, where Link 2 is the upper arm of the robot and Link 3 is the forearm. The Shoulder Joint supports Link 2, and the Elbow Joint supports Link 3. Link lengths have been selected to be similar to that of an adult human arm while allowing complete rotation of the links without collision or interference. Link masses have been decreased substantially via the gusseted link shapes, which reduce the weight of the robot by both decreasing link volume and motor size.

The motor driving Link 2 is mounted on the platform at the end of Link 1 and is directly coupled to the shoulder joint. The motor driving Link 3 is offset from the Elbow joint and is coupled via a timing belt. The elbow motor is mounted on Link 2 near the shoulder joint, but on the opposite side of the link. This mounting location allows Link 2 to rotate fully and decrease the inertia of the arm to allow faster throwing velocities with smaller and lighter motors. Link offsets a_2 and a_3 in Table 1 are designed to allow the gripper to travel in a plane containing the Link 1 axis of rotation.

Gripper

The primary requirements for the gripper include sufficient strength to hold the ball given dynamic forces, fast opening dynamics to minimize residual effects on the ball’s trajectory, and light weight to minimize forearm inertia. A gripper design consisting of two opposing four-bar linkages was selected for this purpose. Using the slider-rocker linkage configuration, Figure 2, both jaws can be actuated simultaneously by the pull cable indicated. The cable allows the actuator to be mounted off the arm to reduce inertia and improve dynamic performance. A spring is then used to pre-load the linkage and produce a gripping force. The magnitude of the gripping force is a function of the cable force and linkage position.

The gripper linkage is actuated by a pull-type solenoid. Figure 3 indicates the solenoid pull force as a function of displacement at its rated voltage of

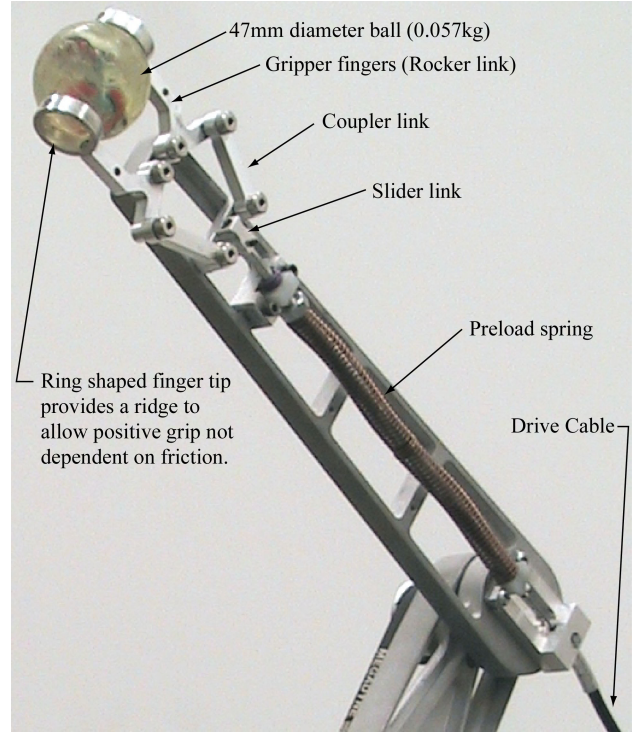


Figure 2. Gripper system.

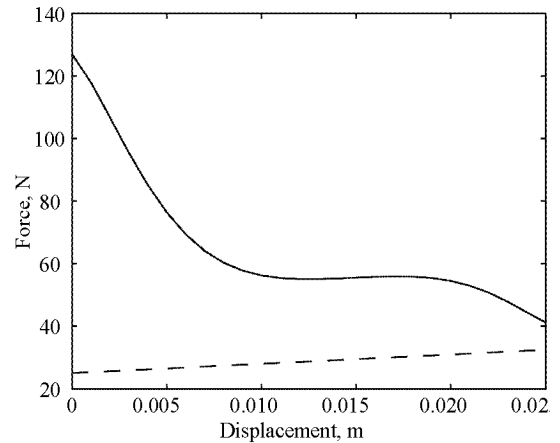


Figure 3. Solenoid force (solid) and spring preload force (dashed) versus displacement.

120 VAC. As the figure illustrates, the solenoid produces much larger forces when displacement is small and hence the device is very well suited to accelerating the gripper quickly.

Table 1. Robot D-H and inertial parameters.

Link (i)	a_i meters	$D_i (l_i)$ meters	α_i deg	I_i kg-m ²	Mass, m_i , kg	Cog, r_i meters
1	0.000	0.908	90°	-	-	-
2	0.022	0.305	0°	0.0171	1.760	-0.0142
3	0.038	0.294	0°	0.0028	0.213	0.1098

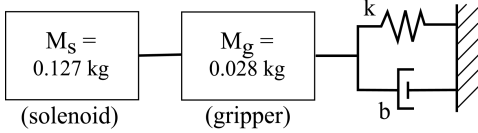


Figure 4. Spring Mass Damper Model of Hand.

In order to select the appropriate spring for the gripper, a simple mass-spring model, Figure 4, is used to approximate the gripper dynamics and the solenoid force is modeled as a step input. The gripper mass is treated as an effective mass, m_g , connected directly in series with the mass of the solenoid core, m_s . The gripper pre-load spring, k , is then connected between the masses and ground. Assuming damping is negligible, gripper response is predicted via the differential equation,

$$(m_s + m_g) \ddot{x} + kx = f(t) \quad (1)$$

where the natural frequency, $\omega_n = \sqrt{k/(m_s + m_g)}$, determines the system response. With the desired outcome of releasing the ball in approximately 50 ms, the spring constant was chosen to produce a natural frequency near 10Hz. Given the masses m_g and m_s , a desired spring constant was calculated to be less than 612 N/m. A spring with a constant of 297 N/m was selected, which ultimately produces an estimated 36 ms release time.

Preload of the spring is reduced by the ring shaped finger design. This allows the ball to nest in the gripper finger slightly and eliminates dependence on friction for restraining the ball during the throw. Based on a dynamic analysis of the ball nested in the finger, neglecting friction, a gripper force of 24N is sufficient for a 47mm diameter ball with mass of 0.057 kg.

When the gripper opens, the spring force increases linearly. The linkage design is such that approximately 2.5cm of solenoid travel will correspond to each jaw opening 45° . This is sufficient to allow the ball to release from the gripper without alteration of the ball's trajectory. As Figure 3 indicates, the solenoid provides sufficient force to open the gripper to this position.

4. Control

Trajectory Planning

The robot was designed not to impact itself in any joint configuration, therefore all joint configurations are attainable and end-effector positions are only critical for the start, release and end points. Likewise, no obstructions exist within the workspace and intermediary positions are inconsequential. The only limiting factor that must be considered is the wiring to Link 3's actuator, which prevents the robot from rotating the shoulder more

than 360° in either direction. Such a configuration allows for the planning of joint trajectories independently, and the calculation of inverse kinematics unnecessary.

Third order cubic splines describe the joint trajectories. One set of splines interpolates between the start and release states, and then another set of splines interpolates between the release and the stop states. A brief constant angular velocity trajectory is placed between the acceleration and deceleration splines to reduce sensitivity to release time. Splines can account for all combinations of initial and final joint states, which allows a simple program to perform the calculations off line. Such a program is flexible enough to work with any feasible input the students provide. Since cubic splines are used to approximate position trajectories, the acceleration trajectories appear as linear functions. These linear trajectories are more easily tracked than the higher order acceleration trajectories that end-effector path planning would produce.

One disadvantage of the joint trajectory spline technique is the fact that the velocity vector is continually rotating throughout the entire throwing motion. The ball is therefore always being accelerated and consistent throwing is more challenging. This emphasizes the need for the repeatable and accurate execution of the trajectories and the ball release.

Feed-Forward Linearization

Due to cost considerations, a microprocessor with limited capability is used to control the robot. In order to compensate for limited computational capabilities, offline feed-forward torque predictions are calculated and streamed into the controller to linearize the manipulator dynamics. Based on the Newton-Euler recursive method [6] applied to the last two planar links, Links 2 and 3, the model based linearizing torques are determined from the system of dynamic equations,

$$\mathbf{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{V}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \mathbf{G}(\boldsymbol{\theta}) + \mathbf{F}(\dot{\boldsymbol{\theta}}) = \boldsymbol{\tau} \quad (2)$$

where $\mathbf{M}(\boldsymbol{\theta})$ is the inertia matrix, $\mathbf{V}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ is the centrifugal and Coriolis terms, and $\mathbf{G}(\boldsymbol{\theta})$ represents the gravity terms. These are determined by,

$$\mathbf{M}(\boldsymbol{\theta}) = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \quad (3)$$

$$M_{11} = I_2 + I_3 + m_2 r_2^2 + m_3 (r_3^2 + l_2^2 + 2r_3 l_2 \cos \theta_3) \quad (4)$$

$$M_{12} = I_3 + m_3 r_3 (r_3 + l_2 \cos \theta_3) \quad (5)$$

$$M_{21} = I_3 + m_3 r_3 (r_3 + l_2 \cos \theta_3) \quad (6)$$

$$M_{22} = I_3 + m_3 r_3^2 \quad (7)$$

$$\mathbf{V}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \begin{bmatrix} -m_3 r_3 l_2 \sin \theta_3 (2\dot{\theta}_2 \dot{\theta}_3 + \dot{\theta}_3^2) \\ (m_3 r_3 l_2 \sin \theta_3) \dot{\theta}_2^2 \end{bmatrix} \quad (8)$$

$$\mathbf{G}(\boldsymbol{\theta}) = \begin{bmatrix} m_3 r_3 g \sin(\theta_2 + \theta_3) + (m_2 r_2 + m_3 l_2) g \sin \theta_2 \\ m_3 r_3 g \sin(\theta_2 + \theta_3) \end{bmatrix} \quad (9)$$

where these terms are based upon the parameters shown in Table 1. Additionally coulomb friction estimates were measured and are incorporated into the model as $\mathbf{F}(\boldsymbol{\theta})$. Hence, the linearizing torques, are calculated by evaluating Eq.(2) at the desired joint positions, velocities, and accelerations, $\boldsymbol{\theta}_d$, $\dot{\boldsymbol{\theta}}_d$, and $\ddot{\boldsymbol{\theta}}_d$, respectively:

$$\boldsymbol{\tau}_{lin} = \begin{bmatrix} T_{2-linearizing} \\ T_{3-linearizing} \end{bmatrix} = \boldsymbol{\tau} \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_d, \dot{\boldsymbol{\theta}}=\dot{\boldsymbol{\theta}}_d} \quad (10)$$

Error Feedback Controllers

Given the linearizing torques, two different feedback controllers were compared for minimizing error. Both techniques are derived from model-based control [7], Figure 5, where the model based torques are calculated offline outside of the servo loop due to computational limitations. In the tradition of model-based control, however, feed-back gains are applied to both position and velocity signals, and model based feedback terms are used to compensate for the back-EMF of the motor. The applied torque, $\boldsymbol{\tau}$, is then determined by,

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{lin} + \boldsymbol{\tau}_e \quad (11)$$

where $\boldsymbol{\tau}_e$ is the error based torque calculated by the feedback terms. The conventional Model-Based (MB) torque is then determined by,

$$\boldsymbol{\tau}_e = \mathbf{K}_v \dot{\mathbf{e}} + \mathbf{K}_p \mathbf{e} \quad (12)$$

where error is defined as $\mathbf{e} = \boldsymbol{\theta} - \boldsymbol{\theta}_d$. For small error it is assumed that the system matrices are approximately equal and the error equation becomes,

$$\ddot{\mathbf{e}} + \mathbf{M}^{-1}(\boldsymbol{\theta}) \mathbf{K}_v \dot{\mathbf{e}} + \mathbf{M}^{-1}(\boldsymbol{\theta}) \mathbf{K}_p \mathbf{e} = \mathbf{0} \quad (13)$$

Evaluating the inertia matrix at a particular angle, $\boldsymbol{\theta} = \boldsymbol{\theta}^*$, then diagonalizes the inertia matrix and allows the gain matrices \mathbf{K}_v and \mathbf{K}_p to be easily selected to decouple the error equations ,

$$\mathbf{K}_v = \mathbf{M}(\boldsymbol{\theta}^*) \hat{\mathbf{K}}_v \quad \text{and} \quad \mathbf{K}_p = \mathbf{M}(\boldsymbol{\theta}^*) \hat{\mathbf{K}}_p \quad (14)$$

where

$$\hat{\mathbf{K}}_v = \begin{bmatrix} 2\xi_2 \omega_2 & 0 \\ 0 & 2\xi_3 \omega_3 \end{bmatrix} \quad \text{and} \quad \hat{\mathbf{K}}_p = \begin{bmatrix} \omega_2^2 & 0 \\ 0 & \omega_3^2 \end{bmatrix} \quad (15)$$

are the diagonal matrices whose parameters are the natural frequencies and damping ratios of the decoupled error equations at $\boldsymbol{\theta} = \boldsymbol{\theta}^*$. Hence, the controller gains \mathbf{K}_v and \mathbf{K}_p are also diagonal matrices that allow decoupled servo calculations as shown in Figure 5 for reduced computational load. The challenge is that the Eigenvalues of the closed loop error equation (13) are still dependent on $\boldsymbol{\theta}$ and vary throughout the throwing motion.

Selection of decoupled damping ratios and natural frequencies, Eq (15), was thus approached as an optimization problem. The objective was then to vary these parameters so as to place the resulting poles as far to the left as possible, while not exceeding the microcontroller capacity to implement the equivalent of a

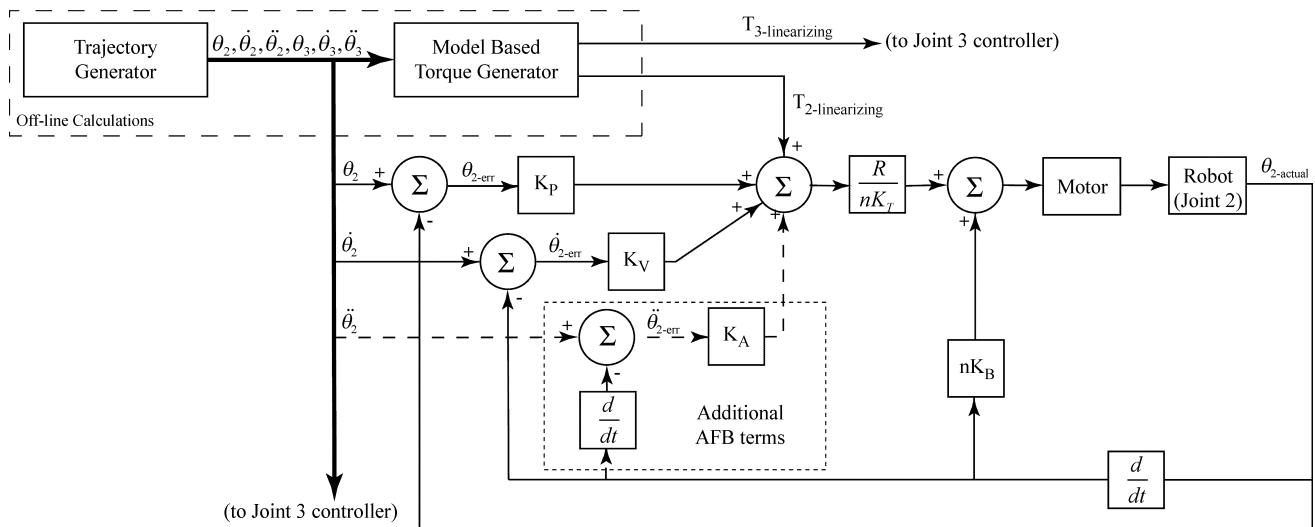


Figure 5. Joint controller using a model-based (MB) technique [6], with the optional acceleration error compensator (AFB) terms highlighted.

continuous state controller. It was thus required that the resulting system bandwidth was substantially lower than the sampling rate, $40\omega_{BW} \leq 2\pi f_s$. Initial conditions for the optimization were determined empirically, and varied to help avoid local minimum. The resulting optimized gains were thus determined to be,

$$K_v = \begin{bmatrix} 1.21 & 0 \\ 0 & 0.37 \end{bmatrix} \text{ and } K_p = \begin{bmatrix} 9.07 & 0 \\ 0 & 9.49 \end{bmatrix} \quad (16)$$

The second technique applies supplemental Acceleration Feed-Back (AFB) control, Figure 5. The resulting error torque is determined by,

$$\tau_e = K_a \ddot{e} + K_v \dot{e} + K_p e \quad (17)$$

An unexpected advantage to this controller is very specific to the context of this robot. When the stream of feed-forward torque predictions terminates, the manipulator should always be at its initial position. If it is displaced while the power is off and the power is reapplied, this linear controller slowly corrects itself without violent jerks that the model-based controller exhibits under similar circumstances. The acceleration reference of zero in addition to the velocity reference of zero further damps out the response commanded by the position step input. This characteristic is ideal for use in an interactive setting, where safety must be a top priority.

Implementation of the desired torques requires an accurate model of the Matsushita GMX-7MC01 9B DC motor used for the joints. These motors were modeled with two states, ignoring inductance and filtering in the amplifier, since these states are several orders of magnitude faster than those we intended to control. The torque estimates described earlier are then converted into appropriate motor voltages based on the motor torque constant, K_T , armature resistance, R , gear ratio, n , and motor speed. This is derived from the classical permanent magnet DC motor equations where the armature current, i , is calculated assuming that the armature inductance effects have settled out. This gives the applied motor voltage,

$$v_m = T \frac{R}{K_T n} + n K_B \dot{\theta} \quad (18)$$

5. Testing and Performance

The capability of the robot to throw a 47 mm ball with a mass of 0.057 kg was examined using both controllers. Several initial velocity trajectories were examined to evaluate the performance of the system while throwing the ball increasing distances. At each trajectory, eight to twelve trials were conducted to obtain an estimate of consistency and error for the control techniques.

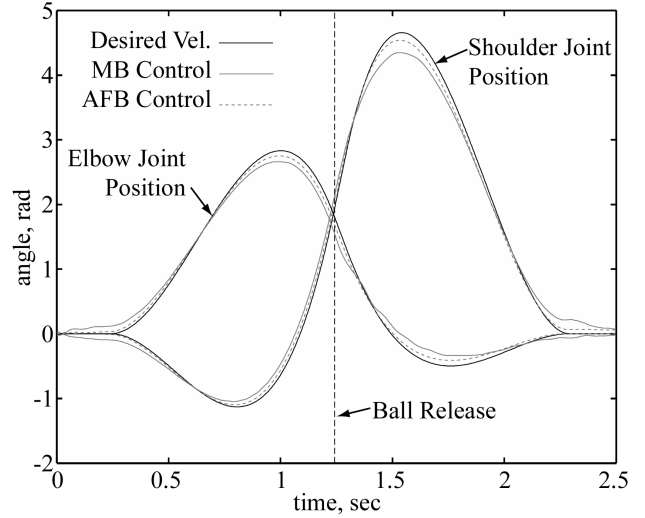


Figure 6. Joint position during throwing motion comparing Model Based (MB) and Acceleration Feed Back (AF) controls.

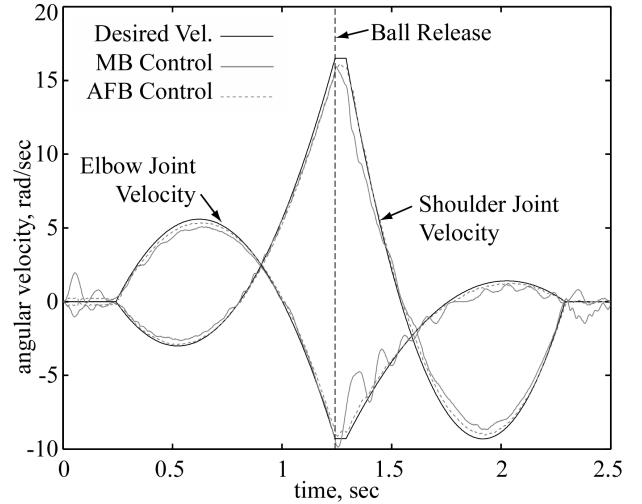


Figure 7. Joint velocity during throwing motion.

Typical joint position and velocity responses are indicated in Figure 6 and Figure 7, respectively, while attempting to throw a ball 5.7m. As the figures indicate, the AFB controller exhibited better tracking and reduced velocity error throughout the entire motion. The MB controller exhibited noticeable damped oscillations in elbow joint velocity, Figure 7, but these oscillations settled out prior to releasing the ball.

The resulting throwing accuracy of the MB and AFB control techniques are illustrated in, Figure 8 and Figure 9, respectively. Given a desired throwing distance, these plots indicate the result produced by each controller. Overall, the repeatability of the MB controller was better than the AFB controller. As the 95% confidence intervals indicate, the MB controller exhibited a maximum

confidence interval of $\pm 0.36\text{m}$, compared to the $\pm 0.56\text{m}$ interval of the AFB controller. Both data sets are approximated well by linear regression, resulting in slope variations of approximately 5% from ideal ($y=x$) for each controller, but the MB controller exhibited slightly more linear throwing accuracy, as $R^2 = 0.98$ indicates. As the regression fit ($y=1.05x+0.01$) indicates, however, the resulting throwing distance were larger than desired. In contrast, the AFB throwing response was nominally closer to the ideal, but the variations were much larger. Thus, for purposes of consistency and reduced computational load, the traditional Model Based controller is preferred.

6. Conclusion

Design, implementation, and performance of a three link serial manipulator have been presented. The purpose of the system is to illustrate to potential engineers the typical roles of mechanical engineers, electrical engineers, and computer scientists in such a project. The system will provide an interactive engineering activity, calculation of and implementation of throwing trajectories, which will allow the students to program the robot and compete for accuracy. Two separate control strategies are examined and compared for ball throwing accuracy and repeatability. Moderately accurate results have been obtained using a feed forward linearizing controller with position, velocity, and acceleration feedback. Further work will be completed to improve accuracy and implement the control interface on a limited and cost effective micro-controller.

Acknowledgement

This work was completed with the support of Robert Roemer, department chair of Mechanical Engineering at the University of Utah.

References

1. Northrop, M.J., "Parts Feeding with a Throwing Robot," Masters Thesis, Northwestern University, Department of Mechanical Engineering, 1999.
2. Aboaf, E.W., C.G. Atkeson, and D.J. Reinkensmeyer, "Task-Level Robot Learning," 1988 IEEE International Conference on Robotics and Automation, **2**, p1309-1310, 1988.
3. Kato, N., K. Matsuda, and T. Nakamura, "Adaptive Control for a Throwing Motion of a 2 DOF Robot," 1996 4th International Workshop on Advanced Motion Control, **1**, p203-207, 1996.
4. Park, C., Kim, J.; Kwon, C., Park, M., "Tracking control of a robot manipulator using sliding mode controller with fast and accurate performance," 1999

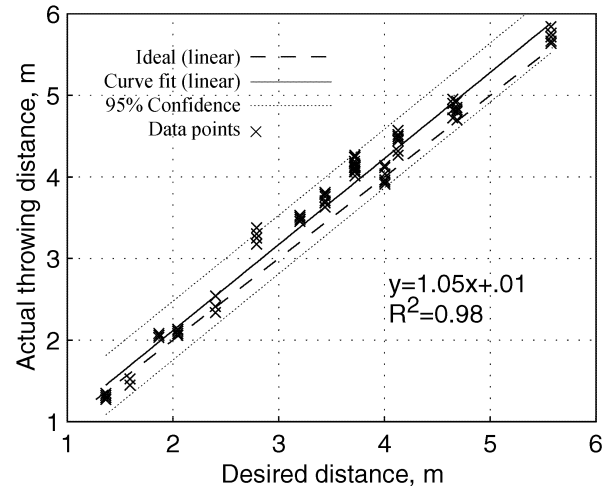


Figure 8. MB control throwing accuracy.

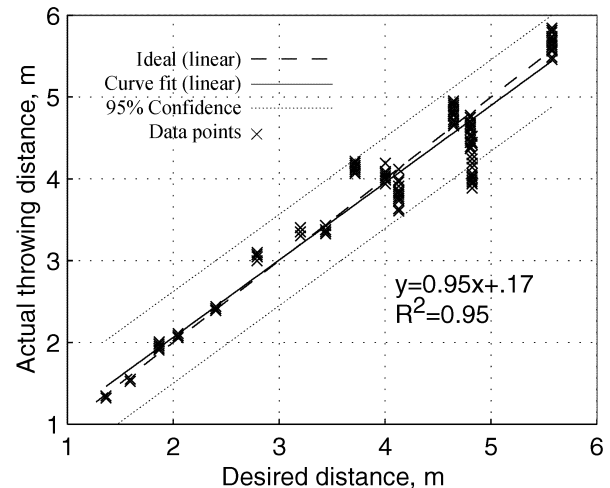


Figure 9. AFB control throwing accuracy.

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'99), Oct 17-Oct 21 1999, Kyongju, South Korea, p305-310, 1999.

5. Yurkovich, S., Garcia-Benitez, E., and Watkins, J., "Feedback linearization with acceleration feedback for a two-link flexible manipulator," Proceedings of the 1991 American Control Conference, Jun 26-28 1991, Boston, MA, p1360-1365, 1991.
6. Fu, K.S., Gonzalez, R.C., and Lee, C.S.G., Robotics Control, Sensing, Vision, and Intelligence, McGraw-Hill, p98-102, 1987.
7. Craig, J.J., Introduction to Robotics Mechanics and Control, 2nd ed., Addison-Wesley Longman Inc, p310-321, 1989.