Optimal motion generation for hydraulic robots

Murali Krishna

Ph.D. Thesis Proposal Document

Last Modified: March 5, 1998

An approach to optimal motion generation for hydraulic robots

Murali Krishna

1. Introduction

Since the late 50's hydraulics have been the systems of choice where high force-to-weight ratios are required. Thanks in part to a flurry of activity by the aircraft industry in the 1940's and 50's, hydraulic systems moved from an experimental concept to a practical and usable technology. Today hydraulic machines are widely used in a variety of applications ranging from aircraft actuator systems to construction, mining, excavation, and forestry equipment.

A hydraulic excavator (HEX for short) shown in Fig 1, is a typical excavating machine which uses hydraulic actuators.

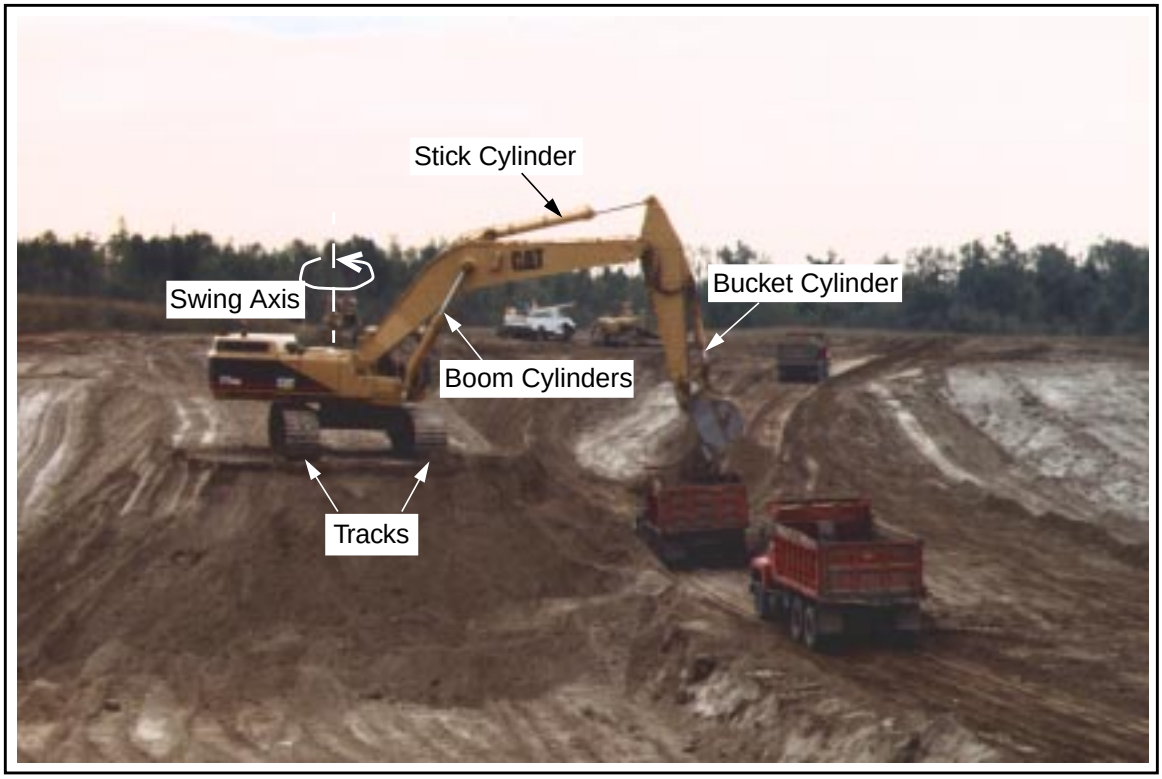


FIGURE 1. A hydraulic excavator (HEX)

It has six actuated joints -

- two tracks which are independently controllable,
- a swing joint with a vertical axis of rotation, and

- boom, stick and bucket joints.

The boom, stick and bucket links are planar with axes of rotation normal to the plane of the links. All three joints are actuated by hydraulic cylinders visible in Fig 1. The swing joint uses a rotary hydraulic actuator (a hydraulic motor) not visible in the figure.

The HEX is powered by a single diesel engine which drives two hydraulic pumps. The hydraulic pumps take low-pressure hydraulic oil and output it at high pressure which is then used to drive the hydraulic actuators.

The above described HEX belongs to a class of machinery with the following characteristics:

- They use hydraulic actuators to drive the different joints
- They are powered by a single engine (or other power source) mounted on the machine
- They are under-powered even during normal operation resulting in dynamic power redistribution, i.e. the actuators (and other systems powered by the engine) routinely request more power than the engine can supply. This causes the available power to be non-uniformly distributed amongst the different actuators.

Through the rest of this document, the term “hydraulic machine” or “hydraulic robot” will refer to a typical hydraulic machine or robot with the above characteristics. The testbed used for all the work in this document is a CAT 325 HEX similar to that in Fig 1.

2. Motivation

Roboticists are now beginning to examine problems related to automating a few tasks in the areas of construction, mining, excavation and forestry. These are operations such as mass excavation and continuous mining where a digging machine fills a bucket with material from a pile or a rock face, transports the bucket load to a waiting truck or conveyer belt, and dumps the load in the truckbed/belt[1]. These tasks are ideal candidates for automation since they are repetitive and there exists room for enhancing productivity while decreasing production costs.

Automation can be a practical reality only if the robotic machinery is more productive (i.e higher output in tons/hour) than a manually operated one, while offering lower production cost (i.e. cost/ton of material excavated). An end-user, such as a mine operator, does not invest money in new technology for the sake of technology alone. The user is primarily interested in higher productivity and lower costs, while maintaining safe operation.

1. This load-transport-dump operation is repeated over and over again during a typical excavation or mining operation, and is called a “loading cycle”.

Automated machines automatically yield some productivity increase when operator fatigue, and hence the associated productivity loss, is eliminated by the robot. Changing the design of the robotic machine may also yield further gains in productivity, besides more efficient operation, since manually operated machines are designed for operator comfort and ergonomics. Removal of the operator comfort restriction will allow the designer to make the robotic machine more efficient/productive than the maual machine. However, if robotic machines are to be phased in gradually the interim machines must allow a dual mode of operation. Thus major changes in machine design may not be a feasible option, atleast in the near-term future.

The most significant gains can be realized by observing that most human operators use the machines sub-optimally[2] all the time, and the few expert operators who do operate optimally cannot sustain it for very long. Herein lies the greatest opportunity for productivity improvement. Through consistent optimal performance a robotic machine can yield significant gains over a manually operated machine. The optimal operation can also be chosen to not merely optimize productivity but rather optimize a combination of productivity (tons/hr) and cost of production ($/ton).

The optimal motions computed can also be used to train operators to perform more efficiently. Thus, optimal motion computation is not limited in its application to robotic machinery alone. In addition to the above two uses, optimal motion computation also has use in improving machine design to improve efficiency of operation. Today, there is no easy means of gauging how a particular design change affects (say) a truck loading operation, which consists of digging from a pile of soil and dumping in a waiting truck. The effect of the design change is measured through simple single joint tests and qualitative comments from expert operators, all of which does not quite provide a direct mapping from design change to change in performance. However, an optimal motion computation system can provide just such an answer.

This document proposes a strategy for addressing the problem of optimal motion computation for hydraulic robots. Although the approach is tailored to address the highly non-linear characteristics of hydraulic robots, it is not limited to hydraulic robots alone. In fact it is a general optimization methodology that can be used for any robotic system.

The proposed approach utilizes dynamic models of the robot to compute the optimal motions. The models of the robot can never be perfect - there will always be some discrepancy between the model and the real machine. Also, each individual instance of the robotic machine will have different characteristics due to factors including component wear, joint friction, and manufacturing tolerances.

Parameterized scripts used by Rowe et al. [24] offer a practical approach to performing excavation tasks with on-line local optimization of the operation. The local optimization uses robot feedback and therefore the “real”

2. The term “optimal operation” refers to machine operation while optimizing a combination of productivity and operating cost.

machine model. However, the approach requires the apriori construction of the scripts. A script is a sequence of machine motions required to perform a task (For more details refer [24]). The scripts are currently constructed using an expert human's knowledge of the task. The optimal motion computation system can aid in script construction by playing the role of human expert. The script thus constructed provides a starting point for the on-line local optimization.

The next section, Sec 3, states the problem addressed by the proposed work. That is followed (Sec 4) by a brief overview of existing literature on this subject, and some related areas. Sec 5 looks at the various caveats in optimal motion computation for hydraulic robots. The proposed approach to solving the problem is described in Sec 7, followed by some results from preliminary testing, in Sec 8. Sec 33 summarizes the proposed approach and Sec 10 lists the contributions of the proposed work. Sec 10 gives a detailed description of the schedule of work from the time of this proposal to completion of the research in Spring 1999.

3. Problem Statement

The problem can be described as: "Given a power limited hydraulically actuated robot plant with a controller, how to compute the sequence of inputs which will perform a specified task subject to all specified constraints, while optimizing an objective function".

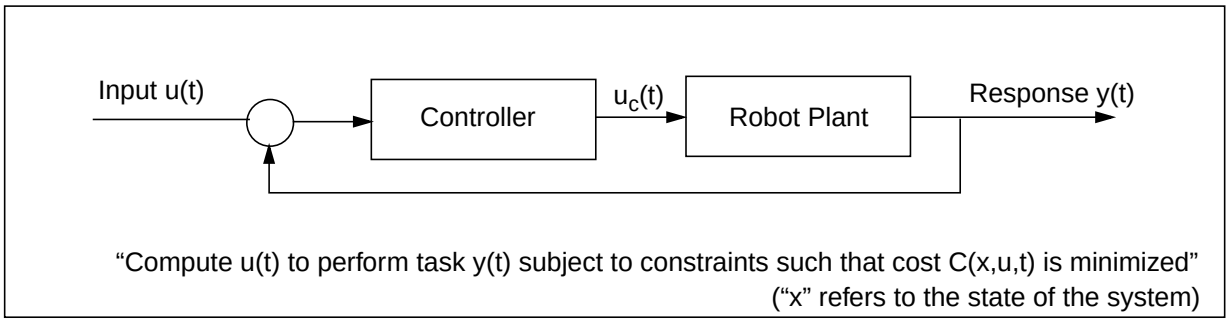


FIGURE 2. Problem description

In the figure above, the "Robot Plant" refers to all the hardware of the robot, i.e. the engine, hydraulic pumps, valves, and cylinders/motors which move the robot links, and the robot links themselves. The controller typically controls the robot plant through electro-hydraulic valves. Thus, the signal $u_c(t)$ is a control signal for the control valves, while the command u(t) is, in general, a trajectory, i.e. a sequence of desired robot tip positions or velocities versus time.

The objective function is composed of a combination of cost factors such as fuel consumption and execution time. The optimization is not merely with respect to linkage dynamics but with respect to the combination of actuator and linkage effects. This is especially important for hydraulic robots where the actuator effects are very significant.

The term “actuator dynamics” requires a little clarification vis-a-vis hydraulic robots. Used in the context of traditional serial chain electric drive robots it refers to the dynamics of all the elements responsible for moving the robot links, i.e. the motors and the gears that move the robot’s links. In a similar vein the actuator dynamics of a hydraulic robot refers to the dynamics of the entire hydraulic system responsible for moving the joints. Typically a number of components are involved here - the engine that supplies the shaft power, the hydraulic pumps that use the shaft power to transform low pressure hydraulic oil into high pressure hydraulic oil, the valves used to regulate the flow of the high-pressure oil to the cylinder/motor, any gears that might be employed, and the hydraulic cylinders and motors.

Linkage dynamics refers to the dynamics of the system of robot links moving in space, which are governed by the well known Newton-Euler equations ([5]). The actuator and linkage dynamics together determine the motion of the links of a robot.

Getting back to the problem statement, the task specification is in the form of a start and goal state, a state being composed of position and velocity variables. It can also be extended to a broader concept such as digging from a certain location and dumping at another, such that the (digging, free motion, and dumping) motions computed are optimal over the entire loading cycle. The proposed work will only address the non-digging section of the cycle. Optimizing digging is a sub-field in itself and will not be addressed by this work.

The problem statement assumes the use of a “low-level controller”, which refers to a position controller for the purpose of this research. However, in general, it could be a trajectory controller following a specified trajectory (position/velocity specified as a function of time). The advantage of using a low-level controller is that it minimizes the effect of external disturbances that may cause the robot to not achieve the command u(t) (Fig 3). The controller also improves system accuracy through the use of feedback. Thus, even if the optimal inputs to the controller are not in the form of a control law and are in the form of an open-loop control (or command sequence), they do not suffer from the disadvantage of sensitivity to disturbances. The good system positioning accuracy through the use of the controller allows the input u(t) to be well behaved since it does not have to compensate for low-level inaccuracies.

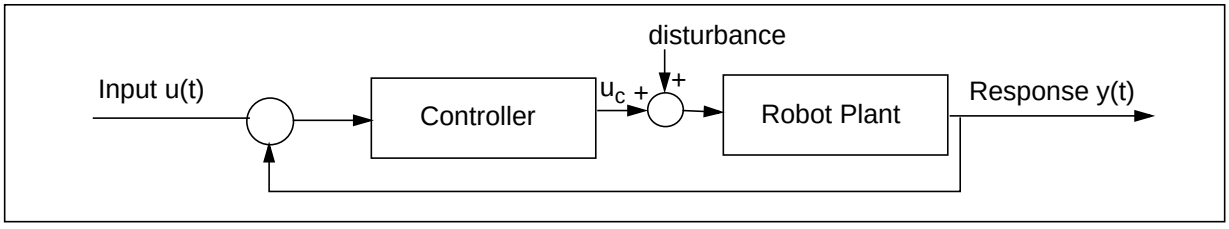


FIGURE 3. Controller can compensate for disturbances

One other point that should be noted is that the low-level controller does not affect the optimal plant motions. If an optimizer is used to compute the optimal u(t) for a given task for two different low-level controllers, the optimal controller input u_1*(t) (* denotes optimal) and u_2*(t) will (ideally) result in the same optimal plant input

$u_c^*(t)$, even though the optimal controller inputs $u_1^*(t)$ and $u_2^*(t)$ may be different. In the presence of disturbances $u_c^*(t)$ in the two cases may be different.

4. Review of previous work

The terms path and trajectory are used quite extensively in this document and they are defined as follows for clarity.The term “path” refers to a continuous curve in cartesian or robot configuration space connecting an initial and final configuration. A “trajectory” is a continuous curve in state space connecting initial and final states. Therefore a trajectory includes a path and the velocity at every point along that path.

Over the last decade, a number of researchers have been working on computationally tractable means of generating optimal controls for general open-chain manipulators for static and dynamic environments. Only work related to static environments will mentioned here since the nature of the problem class being addressed by this work involves slowly changing environments with few obstacles. (There do not exist any truly optimal methods for dynamic environments due to the computational complexity involved [6])

Most attempts at solving the optimal trajectory generation problem have used the purely control theoretic framework provided by Pontryagin’s Extremum Principle. When Pontryagin’s principle is applied to the time-optimal problem for a serial chain robot with n links, it leads to a set of 4n coupled non-linear differential equations with a two point boundary value problem, which is not very computationally tractable. To overcome this problem researchers have linearized the system of differential equations to obtain approximate solutions.

The difficulties with the control theoretic approach led to the use of methods involving state space discretization, followed by an exhaustive search for the minimum time trajectory.

The different approaches thus far can be classified under the following headings:

- Decoupled approaches: These approaches break the overall problem into two decoupled parts. First, the optimal path is computed, followed by computation of the time-optimal control to follow that path. The latter stage does not affect the path computed in the first stage and hence the name decoupled. The work done in computing the time-optimal control along a ***specified path***, i.e. the solution to the latter problem, includes that by Vukobratovic et al. [30], Shin et al.[27], Bobrow et al.[1], and Slotine et al. [28].
- Coupled approaches: These approaches do not go through the intermediate geometric path computation in the decoupled approaches. Even if they compute a geometric path they do not decouple the two stages - the creation of the path and computing the time-optimal control to follow it. They compute a collision-free path that is also optimal with respect to some performance index. Work by Rajan [23], Shiller et al.[26], and Bobrow [2] falls in this category.

- Reactive and Hybrid methods: These methods do not generate an explicit path. The desired state is computed as a function of the current measured state and some external potential field. For instance, the potential field method proposed by Khatib [9] does not require apriori trajectory generation. The trajectories are generated as the task is being executed.

Of the three approaches listed above, the first two classes are off-line methods while the methods in the last category are on-line techniques. Except for the coupled approaches, none of the methods explicitly optimize the robot path itself. That class of methods is therefore the most general and truly optimal approach. The approach proposed in this work, described in Sec 7, falls in this class of methods. The approaches proposed by Rajan[23] and Shiller et al.[26] are briefly described below.

Rajan proposed an approach with the following basic steps:

- Characterize the path in some manner
- Given a path determine the minimum time trajectory
- Vary the path until the minimum time path and trajectory are obtained.

In his examples the paths were characterized using splines, the minimum-time trajectory was computed using the approach by Bobrow et al., and a gradient descent was used to vary the spline parameters in the search for the minimum time path.

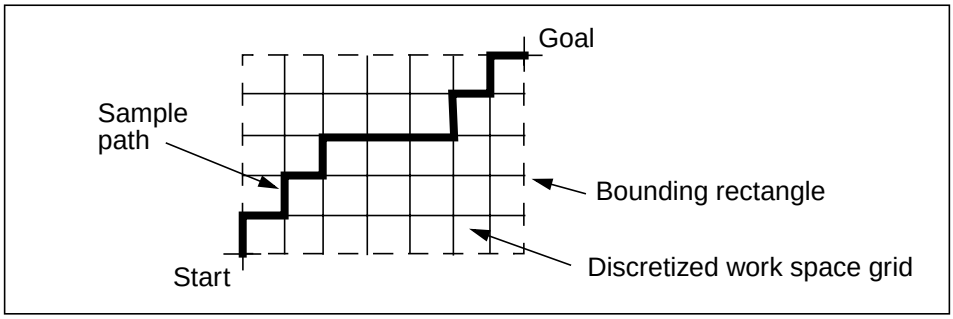


FIGURE 4. Bounding rectangle for paths examined by Shiller et al.

Shiller and Dubowsky proposed a scheme very similar to the one proposed by Rajan. Both methods used the same basic steps, described in Rajan's method, but the differences were in the path characterization, and the method used to search for the optimal path parameters. In Shiller's approach the paths were constructed by discretizing the work space of the robot, and a branch-and-bound exhaustive search was used to pick the optimal path. Shiller used progressively refined tests to evaluate the different trajectory alternatives. In the initial stages of the search the tests consisted of simple heuristics, such as ones that prefer simple smooth paths over devious "jagged" paths (Fig 4). In the later stages of their search the tests consisted of simulating the execution of different path alternatives. They used the time-optimal control computation algorithm pro-

posed by Bobrow et al.[19] to follow each path during the more refined simulation tests. The result of the each simulation predicted the value of the path option. This process is shown in Fig 5.

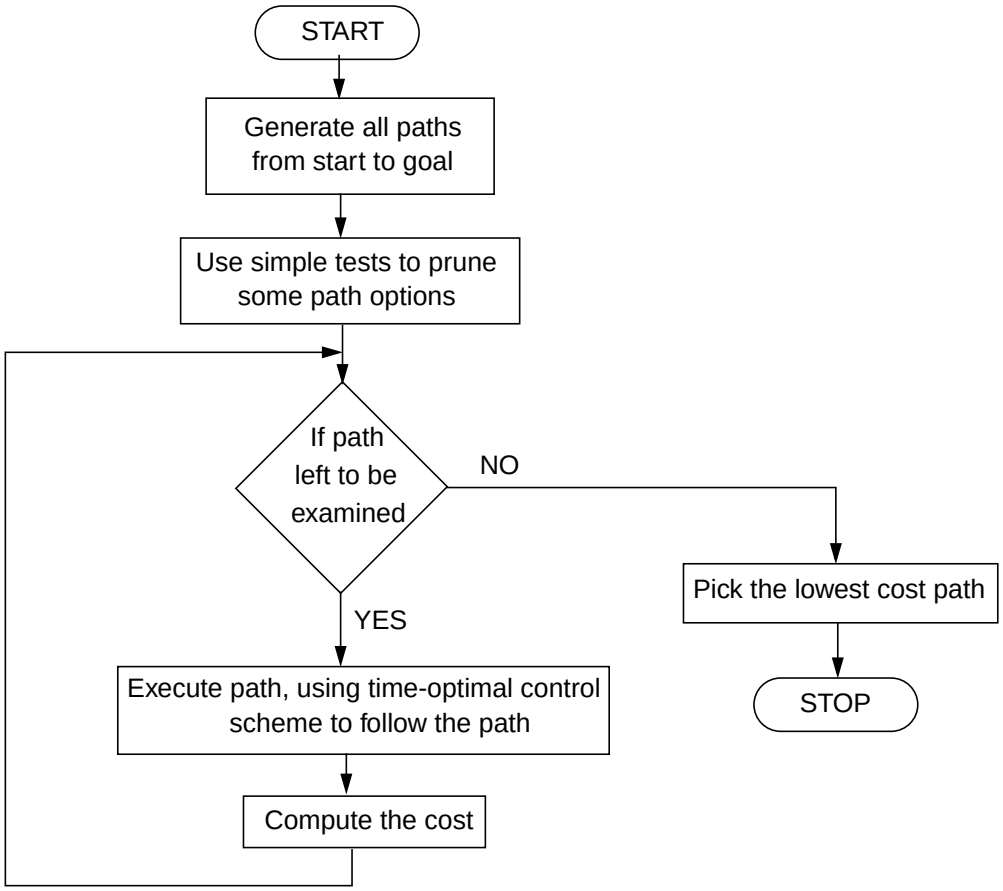


FIGURE 5. Flow-chart for scheme by Shiller and Dubowsky

The above approach by Shiller et al. examined a subset of path options between the start and goal states. The subset is defined by the bounding rectangle between the start and goal (Fig 5). They exhaustively searched through the subset of path options after discretizing the space. The drawbacks of this approach are that the exhaustive search is a somewhat inefficient method of extracting the optimum path, and that the optimum path may not necessarily lie within the bounding rectangle (or cuboid in cartesian 3-space).

We examined one of the examples mentioned in [26] - a 2 joint robot example - and were able to construct a faster control sequence which performed the same task in only 1.76 secs, while the “optimal” path computed using the Shiller’s method was 2.1 secs. This result would indicate that the above approach is not guaranteed to reach the global optimum.

More recently, Martin et al. [15] have proposed an approach to generating optimal motions by approximating the possible robot trajectories by a set of splines. The parameters of the splines completely define the trajectory and the dynamic constraints are transformed into constraints on the spline parameters. Therefore, a search for the optimal trajectory can be performed in the space of parameters of the B-spline curves that

make up the trajectory. This work is focused on computing linkage motions which advantageously use the robot dynamics to optimize the energy usage while performing a task for which the start and end positions are specified. It is not concerned with finding the optimal controls to execute the optimal trajectory computed.

Performing a search for an optimal path in cartesian space can be wasteful since many of the generated alternatives are infeasible. This problem can be avoided by operating in the space of possible robot actions, where all alternatives generated are feasible. Command-space search has been successfully used in the RANGER navigator system to navigate an Ackerman steered vehicle at high speeds [8]. However this approach is only feasible when the number of controlled degrees of freedom are few since the number of command options increases exponentially with the number of degrees of freedom.

All the approaches mentioned above attempt to exploit the robot linkage dynamics to optimize the objective function, which is typically the energy usage. However, since the primary focus of most research to date has been industrial manipulators driven by electric motors for which the actuator charactersitics are not very significant, little attention has been devoted to using the actuator effects advantageously. This is not the case for hydraulic robots where the actuator effects are very significant (and complex). The methods in the literature that allow actuator models assume a known torque limit curve for each joint actuator, which is how manufacturers specify electric motor characteristics. However, the torque limit curve is not easy to compute apriori for a hydraulic robot since the limit curve is a function of many variables due to the inter-actuator coupling. For instance, the force limit for the boom actuator on the HEX is dependent on a number of bucket hydraulic circuit variables, besides the boom circuit variables. Therefore optimal motion computation for hydraulic robots requires an approach different from those seen in the literature.

5. Problem characteristics

This work proposes to address the problem of optimal operation of machinery commonly used in excavation and mining. There are two main features which distinguish this class of problems from most other classes of automation described in the literature.

First, the robot's workspace in these applications is relatively free of obstacles. This is in contrast to most industrial manipulators which operate in restricted spaces making obstacle avoidance an important part of robot motion planning. For the class of problems being addressed in this work the robot's workspace is usually relatively free of obstacles, and the obstacles that are present, such as a truck waiting to be loaded, are not very dynamic. This suggests that in these applications it is more efficient to plan paths assuming no obstacles (or assuming fixed known obstacles) and modifying them if an obstacle enters the workspace, rather than trying to build a system for a dynamically changing environment with moving obstacles. This feature affects the solution strategy used for the motion optimization problem and is discussed further in Sec 7.

Second, the class of applications being addressed use hydraulic robots. Hydraulic robots, unlike most industrial manipulators have complex actuator interactions. The hydraulic actuators of a HEX receive high pressure oil from two pumps. When multiple actuators request flow simultaneously the power demand may (and usually does) exceed the capacity of the engine. The hydraulic system is forced to reduce the flow to the cylinders to keep the engine from stalling. Such a power limited condition is very common during normal operation of an excavator.

The actuator response therefore is an important part of the system dynamics and must be modeled if the motion optimization is to include actuator dynamics in addition to the linkage dynamics. However the complete solution, briefly described in Sec 6.1, requires the solution of an eighth-order non-linear system of 520 equations, which can be very time-consuming. A typical optimal motion computation requires a few thousand simulations and the optimization can take a few days (or weeks) if the complete eighth-order model is used. To address this practical issue we have developed an approach to constructing fast machine models[3] which effectively capture the non-linear actuator interactions. This approach uses a Memory-based Learning technique to learn the actuator model while using an analytical model of the linkage dynamics.

6. Approach

The optimal motion computation problem can be decomposed into two sub-problems:

- Construction of a usably fast machine model that captures the essential characteristics of the robot,
- Using the above model to compute optimal robot motions to perform a specified task.

The former sub-problem has been addressed by the work I have done to date, and the approach developed is described in the sections below (Sec 6.1-Sec 6.3). The latter sub-problem constitutes the bulk of the proposed work, and is described in Sec 7.

Sec 6.1 gives a brief description of the nature of the HEX modeling problem. It describes the equations involved to give the reader an idea of the complexity of the problem. The model construction details are described in Sec 6.2, followed by some results in Sec 6.3 which show the model's ability to simulate the response of the HEX.

3. The following notation is be used through the rest of this document: "Linkage dynamic model" refers to the system of Newton-Euler equations that describe the dynamics of the excavator's links, an "actuator model" describes the actuator dynamics, and "machine model" refers to a complete excavator model which includes actuator and linkage dynamics.

6.1 Hydraulic system modeling - Background

Fig 6 shows a simplified snapshot of the hydraulic system of a CAT 325 hydraulic excavator, which is the test-bed used for the work described in this document. The schematic is only a snapshot since it is valid for the case of boom, bucket, and stick cylinder extension (positive motion) and negative swing rotation. The set of boom (bk, st, sw) valves used for a different direction of boom (bk, st, sw) motion will be different.

The hydraulic system is driven by two hydraulic pumps which take low-pressure hydraulic oil from a tank (at atmospheric pressure) and output high-pressure oil. The power required to achieve the pressure rise is obtained from a single engine (not shown) which drives the pumps.

The high-pressure oil flows to the hydraulic cylinders, which in turn actuate the different joints. Each of the two pumps supplies two implement circuits, i.e. one pump supplies the boom and bucket cylinders while the other supplies the stick cylinder and swing motor. (For the rest of the discussion the tracks will not be mentioned since they are not used during the loading cycle. When they are used to reposition the base of the HEX, one pump is dedicated to each track motor).

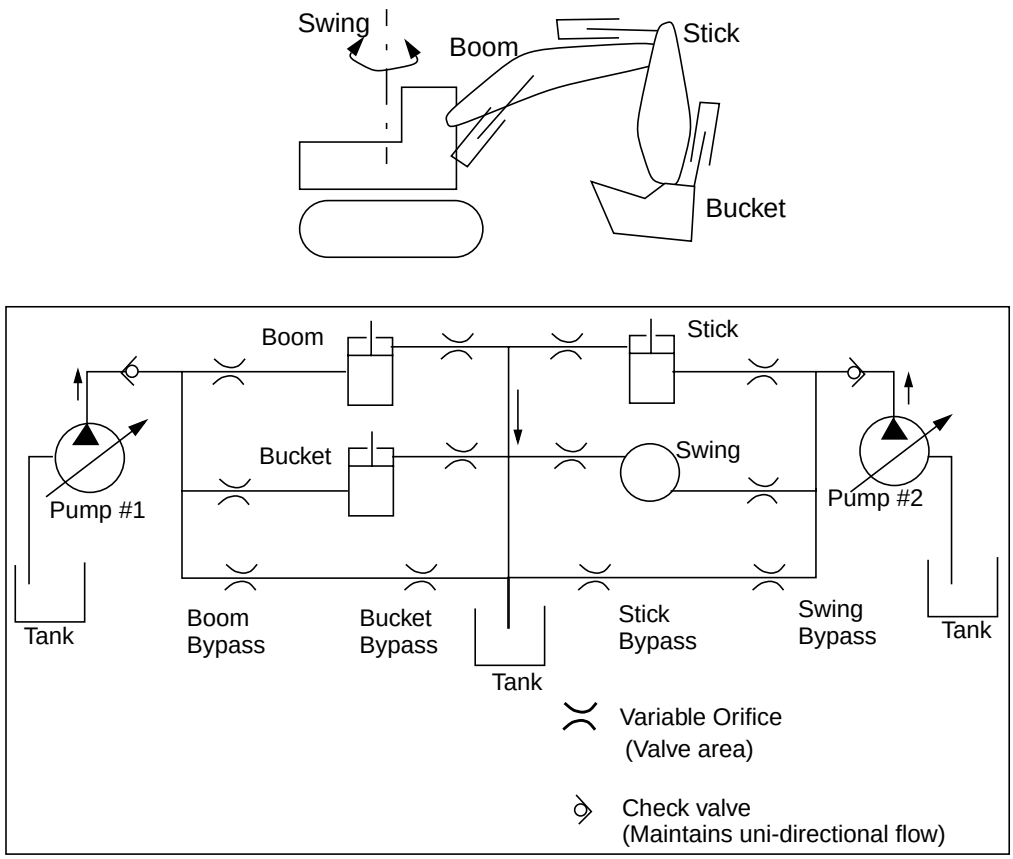


FIGURE 6. Snapshot of hydraulic system of a typical hydraulic excavator for the case of the boom, bucket and stick cylinders extending, negative rotation of the swing motor.

The flow from the pumps to the cylinders is controlled through variable orifices shown in Fig 6. If an orifice is completely closed no flow is supplied to that cylinder and no motion results. The hydraulic system shown

above is an open-center system. In an open-center system the pumps do not reduce their output to zero. When no actuator flow is being demanded, the pumps still output a non-zero flow - between 10 and 20% of maximum flow for the CAT325 testbed. This “idle” flow goes to the tank through the bypass (or “center”) passages shown in the figure. When the actuators are being commanded to move, the bypass passages slowly close and are fully closed when maximum velocity is being demanded.

The parallel arrangement shown in Fig 6 is the simplest mode of operation. More complex valve arrangements result when the HEX is used in other modes. For instance, in “trenching” mode the stick cylinder and swing motor are in series, and the stick can receive flow from the boom/bucket circuit through a set of valves not shown in the simplified schematic.

An analytical model of the hydraulic system includes orifice flow equations, fluid compressibility equations for all the oil volumes, as well as the force balance equations for all the cylinders. The orifice flow equation governing flow and pressure drop across an orifice (for turbulent flow) is:

$$Q = C_d A \sqrt{\Delta P} \qquad \text{(EQ 1)}$$

where Q is the flow rate through an orifice, C_d is the orifice coefficient of discharge (a constant), A is the orifice area, and ΔP is the pressure drop across the orifice. The concepts of flow and pressure drop are analogous to current and voltage drop in electrical circuits. Thus, an orifice can be viewed as a non-linear resistor. In general the valve orifice area A is a non-linear function of the valve displacement. It is non-linear to eliminate effects such as dead-bands in the valves and to obtain a desirable cylinder response characteristic.

The fluid compressibility equation, which captures the dynamics of the hydraulic oil is:

$$\dot{P} = -\frac{\beta Q}{V} \qquad \text{(EQ 2)}$$

where P is the pressure in a control volume, β is the bulk modulus of the oil, V is the volume of oil in the control volume, and Q is the flow rate through the control volume.

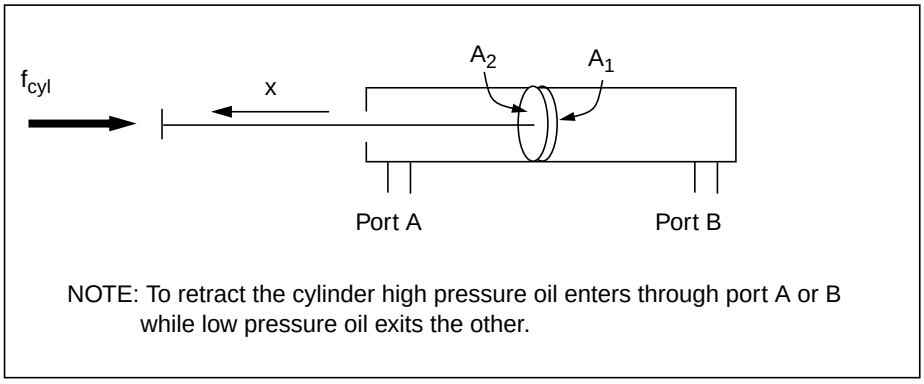


FIGURE 7. Schematic of bi-directional hydraulic cylinder

The force balance equation for a cylinder - boom, stick or bucket - is:

$$m\ddot{x} = P_1 A_1 - P_2 A_2 - f_{cyl} - f_{friction} \quad \text{(EQ 3)}$$

where m is the mass of the cylinder rod, P_i is the pressure with the subscripts indicating the two cylinder chambers, A_i is the surface area on the two sides of the cylinder piston, f_{cyl} is the force on the cylinder due to the linkages and $f_{friction}$ is the friction force on the piston. The force load is due to linkage dynamics and tip forces (if any).

The cylinder extension x is mapped to the joint position θ via a non-linear function: $x_{cyl} = C(\theta)$

The linkage force f_{cyl} appears in the excavator linkage dynamic equations:

$$M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) = \tau \quad \text{(EQ 4)}$$

where each joint torque τ_i is related to the corresponding cylinder force via the transform:

$$\tau_i = f_{cyl_i} \cdot \frac{\partial}{\partial \theta_i} C_i(\theta_i) \quad \text{(EQ 5)}$$

Thus, the solution of the response of even the simplified hydraulic system in Fig 6 involves the simultaneous solution of multiple orifice equations (Eqn 1), multiple compressibility equations for all the oil volumes (Eqn 2), and force balance equations for each cylinder (Eqn 3, Eqn 4, Eqn 5). A steady state solution would not include the fluid dynamics in Eqn 2.The excavator hydraulic system also has other non-linear components such as check-valves (shown in Fig 6) which prevent oil flow from the cylinder to the pump. These act as two-state diodes and make the solution more difficult due to their binary nature. Fig 6 only shows one configuration of the hydraulic valves. Depending on the direction of motion of the cylinders/motor a different set of valves are active.

A complete analytical model of the excavator which includes linkage and actuator dynamics is a coupled eighth-order non-linear system of 520 equations, and is partially described in [17]. A complete dynamic model of the HEX, which includes all hydraulic system elements and linkages, has been constructed using a proprietary numerical solver, and its performance has been verified using results obtained from the CAT 325 testbed. This detailed model takes 100-150 secs to simulate 1 sec. of a typical excavation cycle when running on a SUN Sparc20 workstation.

The above HEX model is complete but rather slow for use in optimal motion computation. Therefore a simplified model, which captures the important HEX characteristics while not being as computationally expensive, is desirable. Among the characteristics that must be captured by the simplified model are the actuator interactions which are important since they significantly affect the overall response. The interaction between the dif-

ferent actuators occurs due to the fact that they are powered by a limited power engine. When the sum total of the power demanded by the joints exceeds the power output of the engine, the pumps reduce their flow to keep the engine from getting overloaded. This causes the flow distribution between the different joints to be uneven.

Singh et al. [12] use a simple approach to handle the flow distribution between multiple hydraulic actuators. They assume a fixed max. pump flow (even though the pumps vary their output), and assume that the circuit with a valve closest to the pump gets all the flow it requires, and that the remaining flow is distributed among the rest. This approach is valid when the interacting cylinders have very different force loads, but not when the cylinders have similar force loads. There is no literature on how the flow distribution can be modeled more accurately without resorting to a detailed model. The following approach addresses that shortcoming.

6.2 Machine model construction - Approach

The goal of the model building exercise was to construct an open-loop excavator model capable of simulating typical excavator operations. The complete model has the ability to simulate machine motion reasonably accurately but much faster than a complete analytical model. The complete excavator model is partitioned into the actuator model and the linkage dynamics. Eqn 4 is used to capture the linkage dynamics, while memory-based learning is used to construct the actuator model as described below.

6.2.1 Actuator Model

Machine learning techniques such as locally weighted regression or neural networks have been shown to be capable of learning any arbitrary functional mapping. Neural networks require large training times while locally weighted regression offers a good and fast approach to learning. The disadvantages of the memory-based learning approach are the large amount of memory required to store the entire training data set, and the longer prediction times. The main advantage of memory-based learning is that since the internal representation of the learned function is simple it is possible to relate the training data to the prediction. This is not possible with neural networks since their internal representation is not explicit. Locally weighted regression was therefore used to learn the non-linear actuator characteristics while Eqn 4 was used to capture the linkage dynamics.

To understand the actuator model construction it would be useful to first understand the basic concept behind the physical operation of the hydraulic system shown in Fig 6. The boom-bucket part of the hydraulic circuit in Fig 6 can be viewed as a parallel combination of resistors as shown in Fig 8.

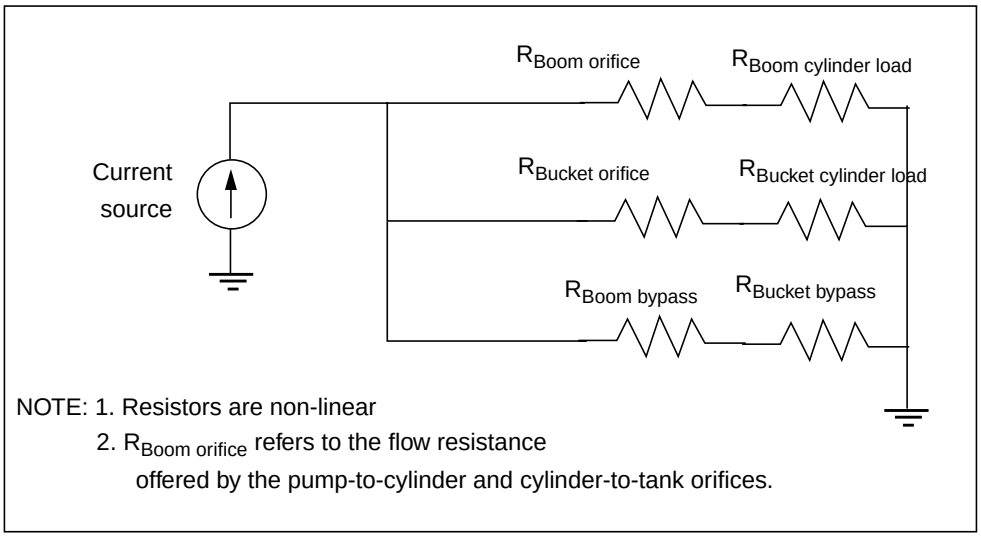


FIGURE 8. Electrical equivalent of hydraulic system

The flow of hydraulic oil into any cylinder determines the velocity of that cylinder. The flow (current) from the pump (current source) is distributed between the three parallel paths. The flow (current) going down each branch is determined by the ratio of the resistances of the different branches. For a given set of orifice areas - boom, bucket and bypass - and for a given set of boom and bucket cylinder loads, the distribution of flow between the different paths can be determined for a steady-state condition. This in turn allows the determination of the steady-state cylinder velocities.

This is an important approximation made in this approach, i.e. the actuator response is approximated by the *steady-state* actuator response for a given set of orifice areas and cylinder force loads. The force loads themselves are not restricted to be steady-state forces - they are computed using the dynamic model (Eqn 4) of the excavator. This approximation is made since learning the complete actuator dynamics requires the specification of a set of state variables in addition to the orifice areas and cylinder forces, which greatly increases the dimensionality of the space.

The approximation is justified since the machine model being developed is not focused on simulating short transient responses but is aimed at capturing the important “trends” (or features) in the excavator’s response over a multi-second loading cycle (See page 2 for a definition of a loading cycle). Based on analysis of a typical loading cycle it was observed that the total cycle time was more than 10 times the rise time for each joint. In fact the cycle time is usually much greater than (riseTime * numberOfRisesInACycle). Thus actuator transients can be ignored without significantly affecting the overall result, as borne out by the results in Sec 6.3.

The different orifice areas (Fig 6) are controlled by the position of a control spool. For example, the position of the boom control spool determines the boom pump-to-cylinder, cylinder-to-tank, and bypass areas. Thus a single spool position can represent all the orifice area variables for a joint.

A multivariate locally weighted regression technique was used to learn the mapping from the space of inputs - the spool positions and cylinder loads - to the space of outputs (the cylinder velocities). For instance, the boom actuator steady-state velocity was determined from knowledge of the boom and bucket spool positions, and the boom and bucket cylinder loads.

Fig 9 shows a portion of the response surface for the boom joint of the HEX. To create the 3-d plot, only 2 of the input dimensions are shown. The other two dimensions - boom and bucket cylinder forces - are fixed for the surface shown. The z-axis of the plot shows the boom cylinder velocity response. From the plot the effect of the interaction between the boom and bucket joints is clearly visible. For instance, the boom cylinder velocity at maximum boom spool position (boom control variable) is seen to drop sharply as the bucket spool position increases. This is due to the fact that the bucket cylinder has a lower force load than the boom cylinder, and since the boom and bucket are connected in a parallel arrangement, the bucket cylinder with the lower “resistance” draws a greater fraction of the total flow.

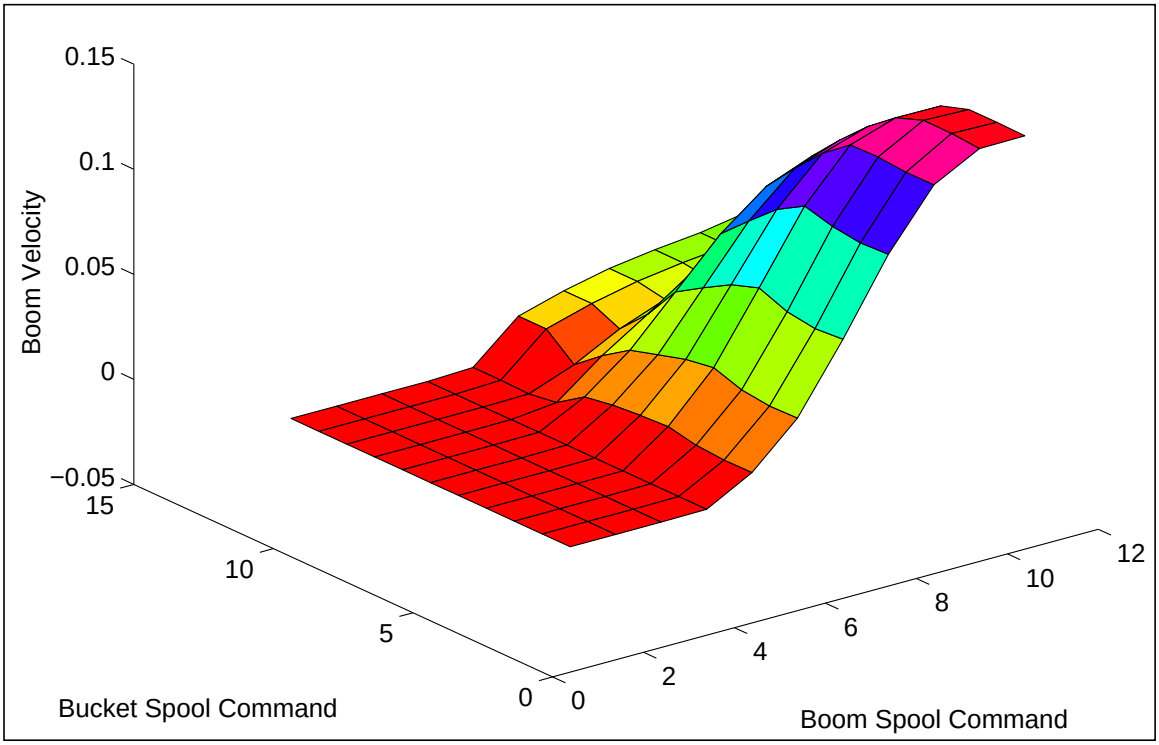


FIGURE 9. Section of the response surface for the boom joint of the HEX

The memory-based learning was performed using a software tool - *Vizier* - developed by Schneider et al. ([24][19][20]) at Carnegie Mellon University. *Vizier* allows the use of locally weighted linear regression to learn data sets and make predictions. The best parameters for the regression can be determined using a blackbox

utility available within *Vizier* which performs a number of leave-one-out type predictions on the learning data set before arriving at the best set of parameters.

Vizier was used to construct four Memory-Based Learning tables (or MBL tables) - one for each of the four joints of the excavator, i.e. the boom, stick, bucket, and swing. The input dimensions of the boom table (and bucket table) are boom and bucket spool positions, and boom and bucket cylinder force loads. The input dimensions of the stick table are swing and stick spool positions, swing inertia, and stick cylinder force load. The single output dimension for each of the three joints is the joint velocity.

The swing is more complicated to model since the inertial acceleration term in the dynamic equation is quite significant. This makes it impractical to use the steady-state assumption used for the other three joints. The other three joints - boom, stick, and bucket - are in a vertical plane and have a large gravity load term in the dynamic equation, which is absent for the swing. The acceleration phases of those joints are rather short (during a typical move) while the swing joint response is dominated by acceleration and deceleration phases.The swing table therefore uses the following input dimensions - swing and stick spool positions, stick cylinder force, swing inertia and swing velocity. The output dimension is swing acceleration.

Data Collection: Collecting training data is an important part of any machine learning exercise. Training data for learning the actuator model was collected using a slow analytical machine model[4]. The slow model was driven through a number of motion sequences to adequately cover the operating space for each learned model. For instance, the boom and bucket joints were actuated in various combinations to obtain adequate coverage of the boom-bucket space, whose dimensions are the boom and bucket spool positions, and boom and bucket force loads. While the spool positions are directly controllable the cylinder forces are not. The motions were therefore repeated for a fully loaded bucket, half-empty bucket, and completely empty bucket to cover the cylinder load dimensions. All motions were performed slowly to minimize transient effects.

The spool positions have a range of ± 11 mm. Data was sampled at a resolution of 1mm along the spool position axes. The cylinder forces were determined by the excavator's configuration. The boom and bucket tables had 5500 points each while the stick and swing had 6400 points each.

6.2.2 Complete Model Construction

The complete excavator model was constructed by partitioning the actuator model and linkage dynamic model into two separate problems. Instead of solving them simultaneously they are solved in a serial fashion. First, the linkage dynamic model (Eqn 4) is used to compute the forces for a given excavator state. This force is assumed to remain constant over the time period that the actuator response is simulated using the learned

4. The testbed was not used to collect training data due to our inability to measure spool positions directly on it. Also, the initial attempt at training used a large amount of training data which was much easier to collect from a simulation model (which had been verified to match the testbed).

actuator model. The results of the actuator simulation are used to compute a new state (which is used in the linkage dynamic model for force computation as the cycle continues). The steps are spelled out explicitly below.

- Step #1: The dynamic model of the excavator is used to compute the force loads on the different hydraulic cylinders.
- Step #2: The force loads computed in Step #1 are used with the input spool commands to predict the resulting cylinder velocities using the corresponding MBL tables. The swing table uses the current swing velocity to compute the swing acceleration.
- Step #3: The computed velocities (or accelerations) are integrated to obtain an updated excavator state. (Repeat steps 1 through 3)

6.3 Machine model construction - Results

The above approach was used to construct a complete model of a CAT 325 HEX. The performance of the model was evaluated by comparing it to the performance of the excavator testbed. The results from three tests are shown in Fig 10 through Fig 19. During the first test (Fig 10, Fig 11), the boom and bucket cylinders were actuated to demonstrate the interaction between them, while the second test (Fig 12) demonstrated interaction between the swing and stick joints. In the third test (Fig 14-Fig 19) an operator in the cab of the HEX performed operations similar to that during a normal loading cycle. The HEX was operated manually through joysticks, which control the position of the hydraulic control spools through proportional pilot valves. In the figures below, the “commands” refer to the displacement of the hydraulic spools (measured indirectly). The different orifice areas are controlled by moving the spools. By controlling the orifice areas the operator controls the flow to the hydraulic cylinders, and hence the velocity of the cylinders. Thus, the commands can be viewed as open-loop velocity commands to the machine.

Inter-actuator coupling is clearly demonstrated in the first two tests. In the first test the boom and bucket joints are both commanded to move but due to the actuator interaction the heavier boom joint does not get very much flow until the bucket has stopped moving.

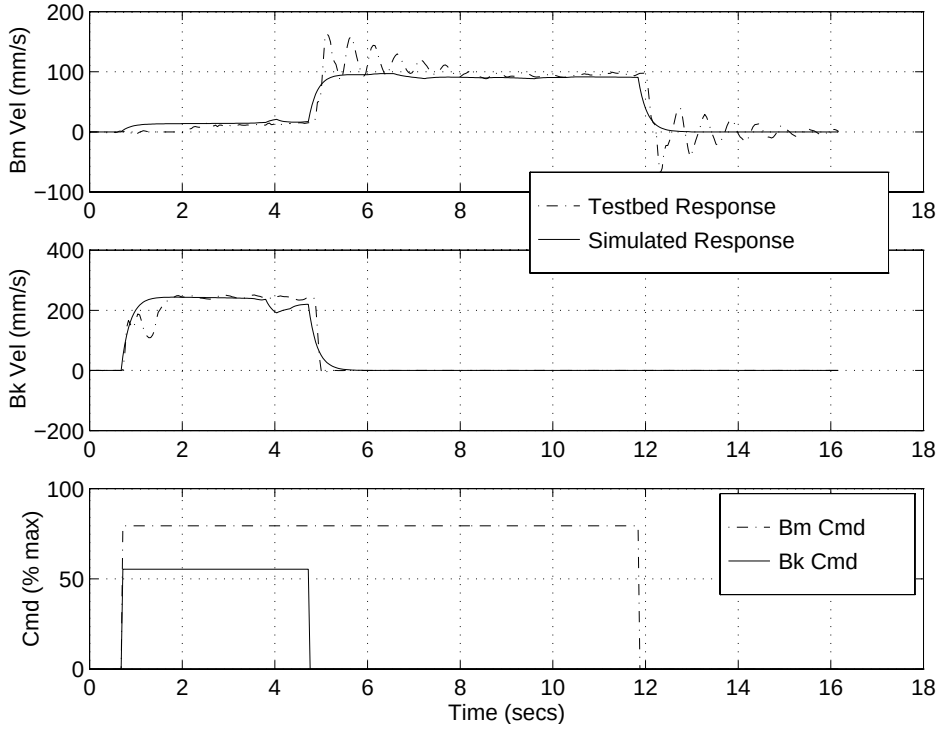


FIGURE 10. Boom/Bucket cylinder velocity plots (Test #1)

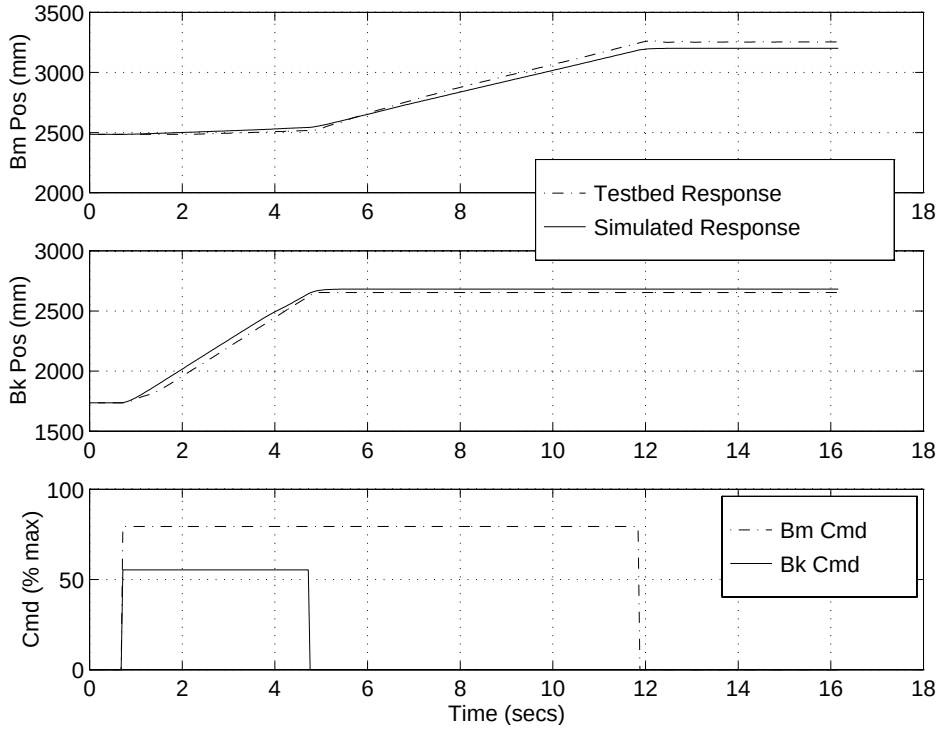


FIGURE 11. Boom/Bucket cylinder position plots (Test #1)

This inter-actuator coupling is primarily because the excavator reaches a power limit, resulting in insufficient actuator flow. The next set of figures show results from the second test where the swing and stick joints were commanded to move simultaneously causing the swing joint to slow down. The swing velocity can be seen to increase dramatically once the stick stops moving.

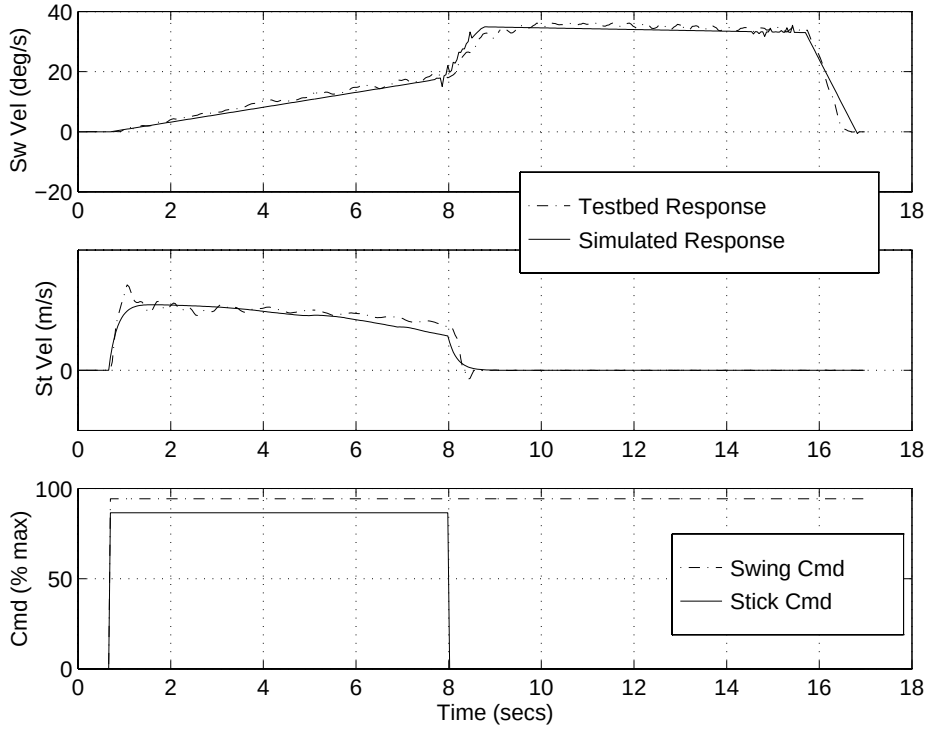


FIGURE 12. Swing/Stick velocity plots (Test #2)

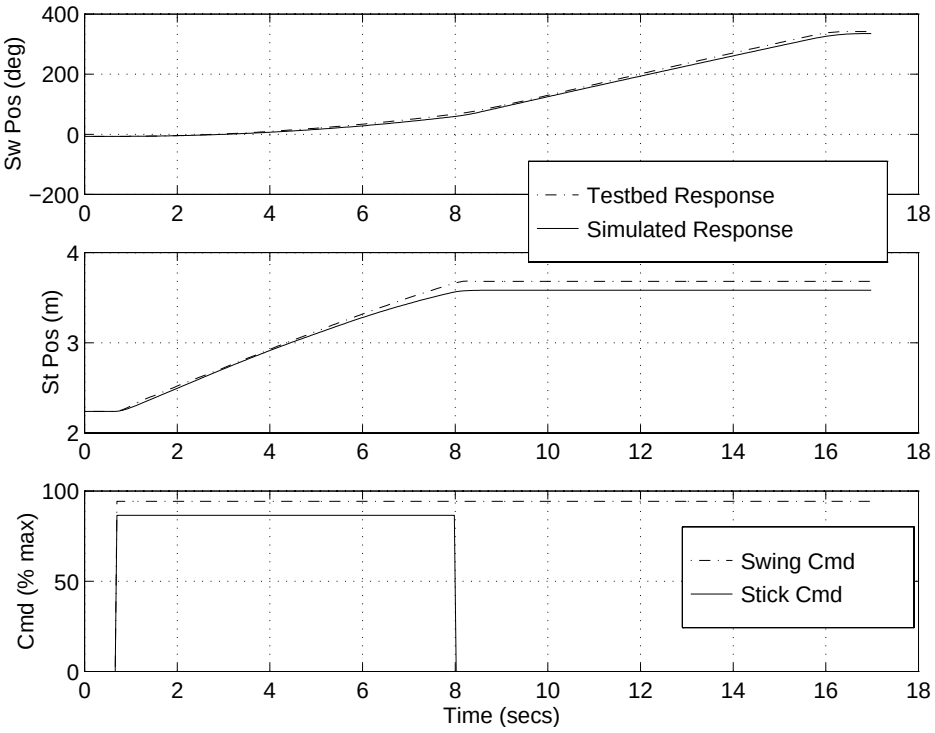


FIGURE 13. Swing/Stick position plots (Test #2)

The next set of figures are results from Test #3. Fig 14 through Fig 17 show the commands and response for each of the four joints of the excavator.

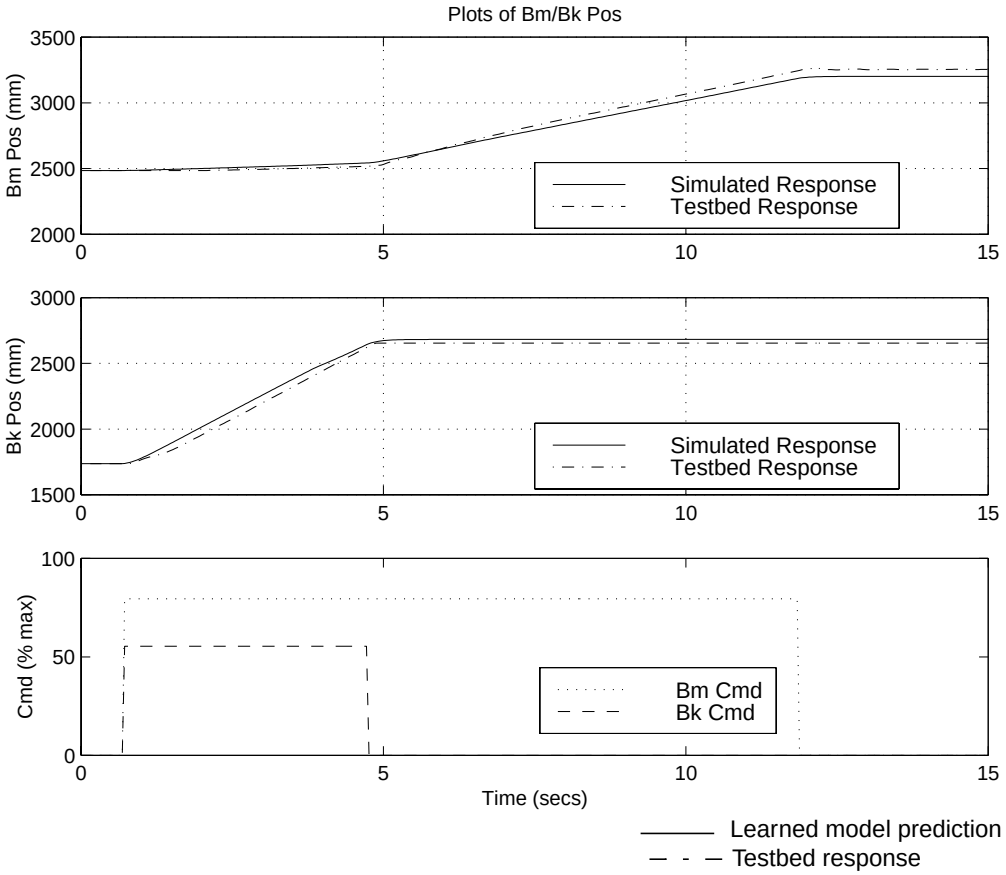


FIGURE 14. Boom Response (Test #3)
(a) Cylinder position - m (b) Cylinder velocity - m/s (c) Spool position command - mm

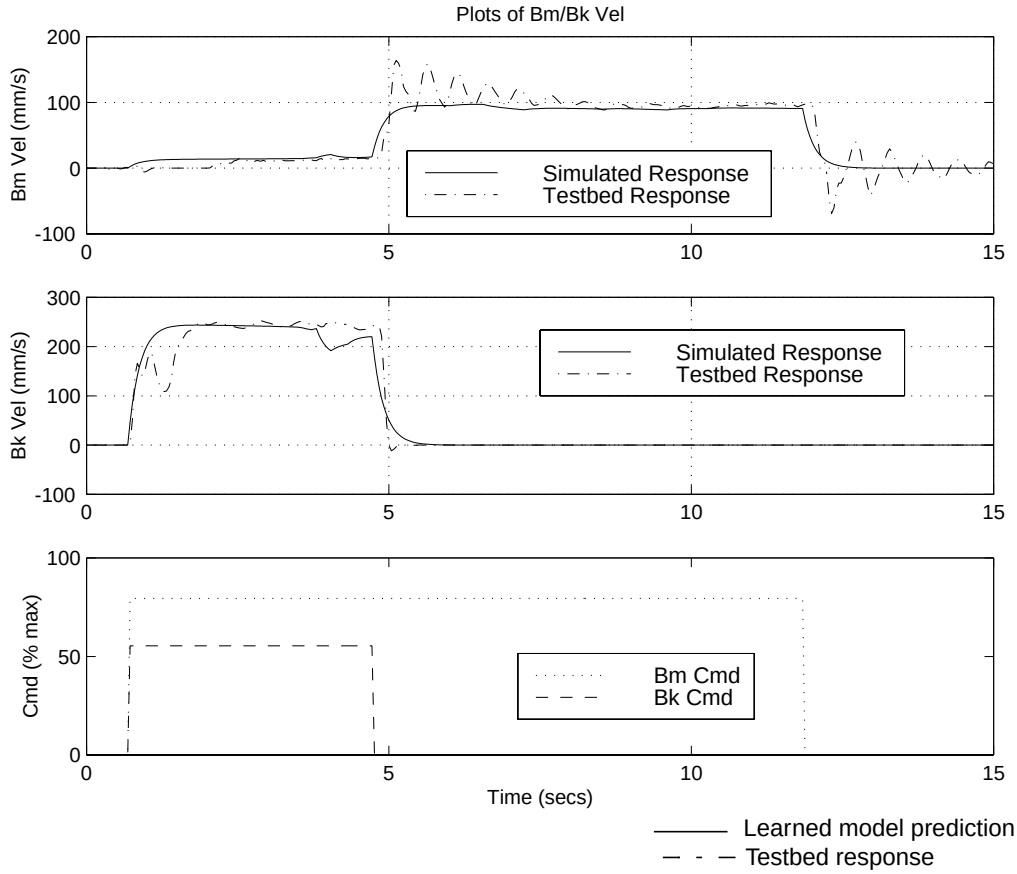


FIGURE 15. Bucket Response (Test #3)
(a) Cylinder position - m (b) Cylinder velocity - m/s (c) Spool position command - mm

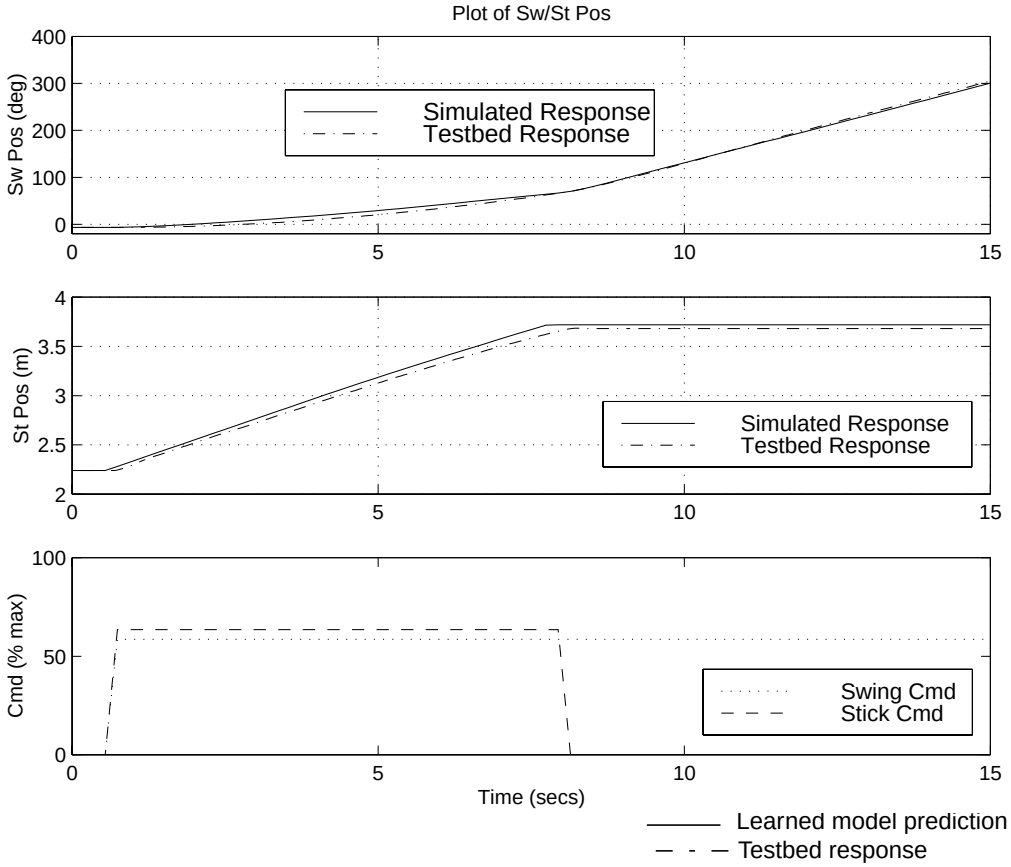


FIGURE 16. Stick Response (Test #3)
(a) Cylinder position - m (b) Cylinder velocity - m/s (c) Spool position command - mm

The current implementation of the learned model uses a locally-weighted regression scheme in conjunction with a table-lookup to learn the data. The model runs at a run-time:real-time ratio of 10:1, i.e. simulating ten seconds of motion requires 1 sec. of computation time on a SUN Sparc 20 workstation.

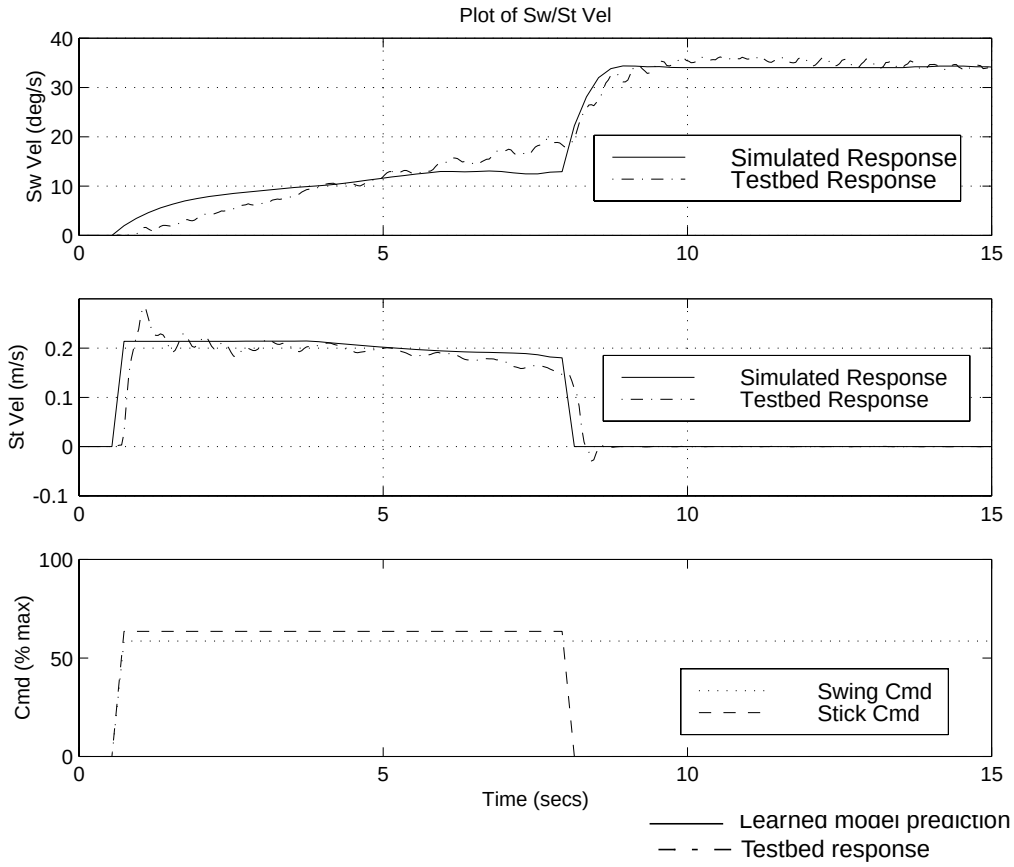


FIGURE 17. Swing Response (Test #3)
(a) Joint position - deg (b) Joint velocity - deg/s (c) Spool position command - mm

Fig 18 and Fig 19 show the error in the prediction of the position of the bucket tip. Fig 18 shows the contribution due to each of the four joints of the excavator. As expected, small errors in the swing and boom joints contribute to large errors at the bucket tip. These plots were created by using the Jacobian of the excavator to map joint angular errors to cartesian tip position errors.

Fig 19 shows the total error in the prediction of the bucket tip position. This is plotted along with the distance from the base of the boom joint to the bucket tip to give the reader an idea of the scale of the tip position error.

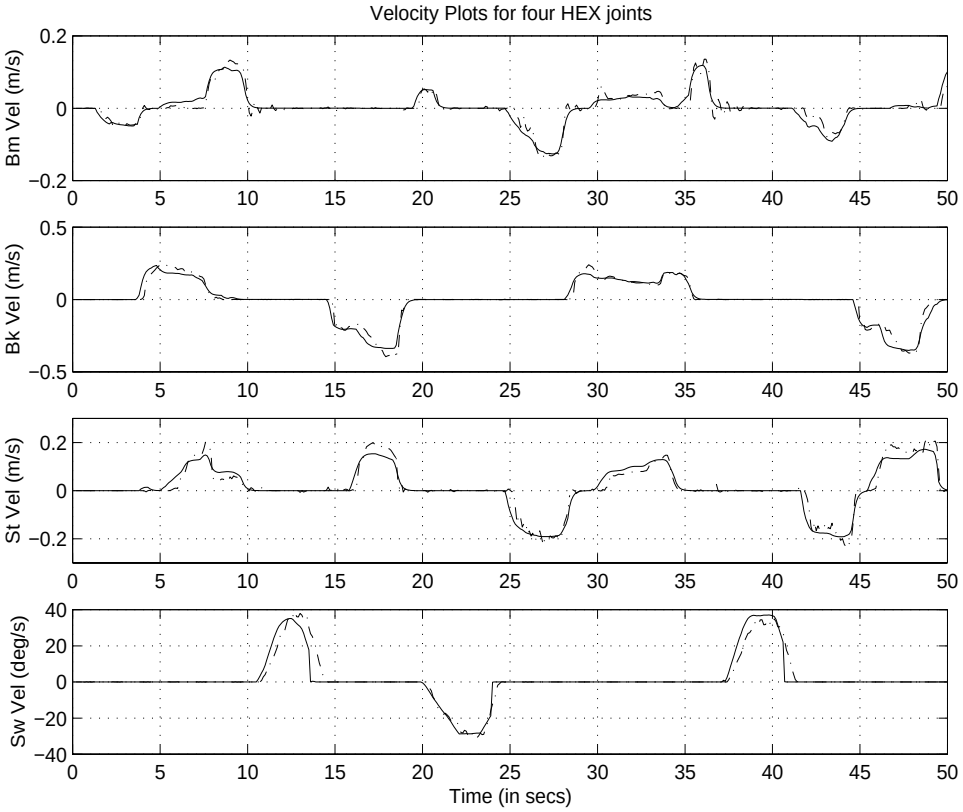


FIGURE 18. Tip error contribution due to each of the four joints (Test #3)

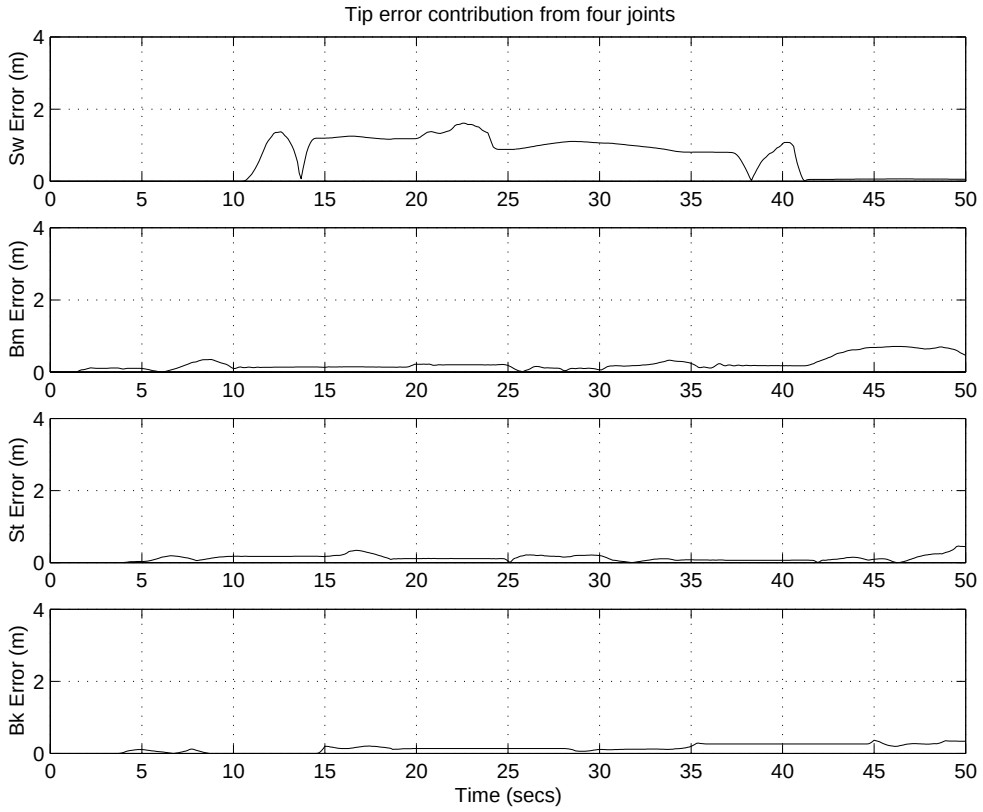


FIGURE 19. Overall bucket tip position prediction error over three loading cycles (Test #3)

The results show that the learned model does a very reasonable job of predicting the response of a hydraulic excavator and successfully captures the important actuator interactions. This completes the open-loop plant model of the excavator.

The complete model of the excavator robot requires a model of the robot controller, in addition to the open-loop plant model. The HEX testbed has individual joint controllers and each controller model was constructed in state-space form[5]. The complete machine model for the closed-loop excavator was constructed as shown in Fig 20.

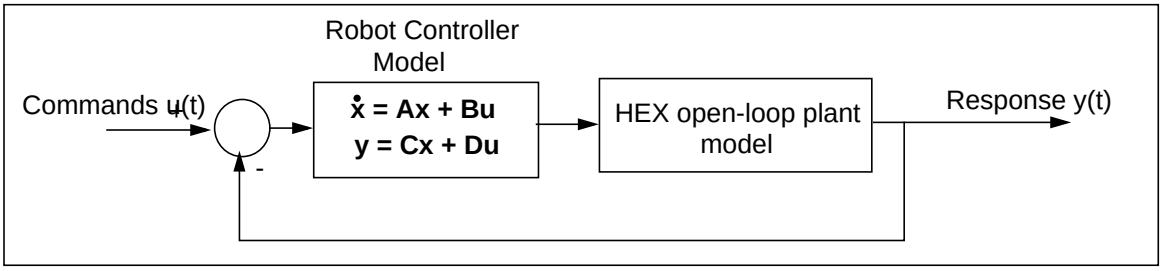


FIGURE 20. Complete closed-loop HEX machine model

5. The controller model could have been learned together with the open-loop plant model. However, as a general rule, it is always better to construct explicit models wherever possible since their fidelity will usually be much better that of a learned model.

This complete model is suitable for use in the optimal motion computation as a quick means of imposing the actuator effects and linkage dynamic constraints.

7. Proposed approach to optimal motion generation

The optimal motion computation problem is:

$$Minimize\ P = \int_{path} C(x_{initial}, x_{final}, u, t)dx \text{ where C =cost function;} \quad \text{(EQ 6)}$$

subject to the constraints imposed by the complete robot model which includes the actuator model and linkage dynamics.

The goal is to find a means of computing the input **u*(t)** to optimally execute a specified task for the controller-plant system shown in Fig 2 (Fig 2 is reproduced below for convenience). The advantage of having a low-level controller is that it results in a lower-order system (shown enclosed in the dashed box) which is more linear than the open-loop plant. This is important for hydraulic robots which have very non-linear characteristics. The low-level controller therefore makes the optimal input **u*(t)** more smooth, thus allowing a coarser discretization of the input **u(t)** when performing a search. The proposed approach to input discretization and search is described later in this section.

The proposed approach does not constrain the robot to have a particular controller - it only assumes the existence of a controller whose model is known. The optimal input **u*(t)** computed for the system shown below is the optimal input sequence for a **given low-level controller** - a different controller may yield a different optimal input **u*(t)**. Note that the existing time-optimal control schemes, such as the method proposed by Bobrow[2], are not directly applicable to hydraulic robots since the actuator constraints cannot be expressed as simple torque limit curves.

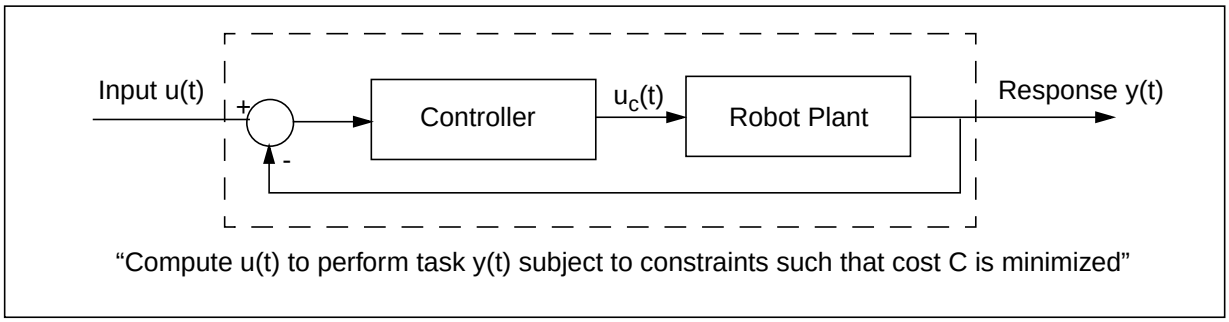


FIGURE 21. Reproduction of Fig 2

If available computing resources were infinite one approach to optimization would be to construct an infinitely long search vector of inputs defining the continuous-valued input **u(t)**. A search in the space defined by the

search vector could yield the desired optimal input. However, since infinite resources are not available, the discretization of the input **u(t)** must be finite.

Large and heavy systems, such as the hydraulic robots described in Sec 1, act as low-pass filters to any input they receive since they have a low bandwidth of operation. Thus, the (optimal) input **u(t)** can be approximated by a (relatively) short sequence **u[k]** (The square brackets and the integer argument k are used to indicate the discrete domain version of **u**). A search technique can then be used to search the space of the discretized representation of **u(t)** to compute the optimal set of inputs **u[k]**. This search approach is feasible if the input discretizations is not too fine - light robots with higher bandwidths may not be solvable using this approach due to the much larger dimensionality of the search vector **u[k]**.

The controller inputs are usually in the form of desired joint or end-effector positions or velocities or both. The CAT 325 testbed used for much of this work has joint position controllers for each of the four joints. The input **u** used for the rest of this work is therefore a stream of position commands, which specifies a **commanded** end-effector path. The optimization search is therefore performed in the space of commanded end-effector paths.

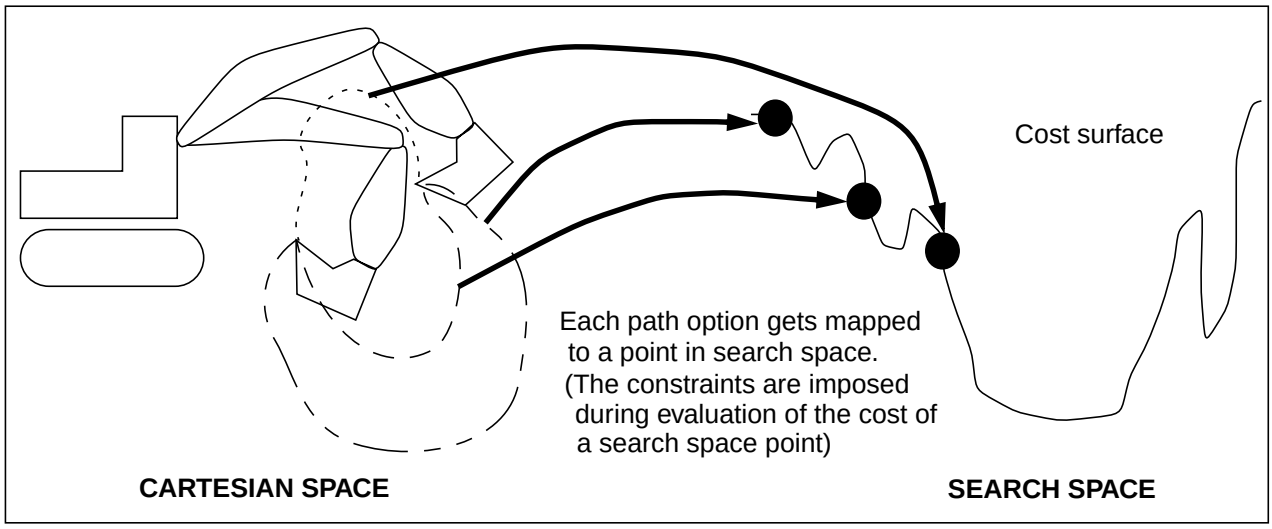


FIGURE 22. Relationship between cartesian space and search space

The essential elements of the search approach to the optimization are:

- Search space:

  The search space for the optimization is the space of all possible paths between the start and goal states. Each feasible path option is a point in the search space.

- Cost function:

  This provides the evaluation of the path options examined. The ultimate goal of the optimization is to minimize this objective function.

- Constraint surface:

  This reflects physical constraints such as kinematic limits, as well as the actuator and linkage dynamic constraints. All path options must be evaluated for feasibility. The advantage of performing the search in command space is that all options generated are necessarily feasible if actuator limits have been imposed. That however is not the case if the search is performed in path space since all paths may not be achievable by the robot.

  The approach to constructing a fast robot model described in Sec 6.2 offers a quick method for imposing the kinematic and dynamic constraints on a candidate path.

- Optimization algorithm:

  The specific optimization algorithm used dictates the method of exploring the search space starting from an initial (sub-optimal) path, to ultimately move to the optimal path. For instance, a simple gradient descent would attempt to move in the direction of steepest descent.

  The process of optimization can be viewed as a walk in the search space towards the point of least cost, with each point along the walk lying on the constraint surface, i.e. each path option satisfies all the constraints.

Thus, the proposed approach to optimal motion computation consists of exploring the search space by evaluating different points in the search space using the cost function while imposing the dynamic constraints, and using the cost information to move towards the optimal cost point in the search space.

The following sections describe the above four elements in greater detail.

7.1 Search space (Problem representation)

There are two approaches to constructing an invertible mapping - spatial and temporal. These approaches are described below. Each has its own advantages and disadvantages and currently both are candidates for the problem representation.

7.1.1 Spatial methods:

The search space for the path optimization should be constructed such that the mapping from cartesian space to search space is invertible - the invertibility will allow the optimal search space point to be mapped to an optimal path. One easy method of constructing an invertible path mapping is by discretizing the cartesian space path using intermediate “knot” points. The vector consisting of the knot points can constitute the search vector, and the dimensions of the search space would be the knot point coordinates. Fig 23 shows a search vector. Note that it does not include orientation of the end-effector at the knot points - if the end effec-

tor orientation is specified at the knot points, then additional dimensions need not be included in the search vector to account for end-effector orientations since it is already specified.

To specify a general trajectory (position and velocity) the same approach can be used by adding the velocity to each knot point specification. Thus each knot point will have six parameters associated with it. The excavator testbed available for this work has a position controller and therefore each knot point is only specified by its position (three parameters) for the rest of this discussion.

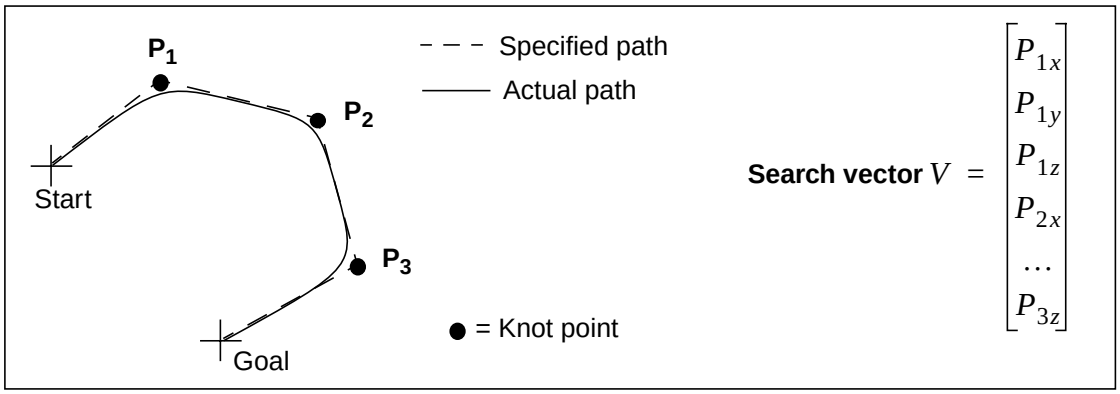


FIGURE 23. Defining a search vector

Note that the path specified by the search vector is not the path that will necessarily be followed exactly. It is the path specified to the robot controller as a series of knot points. The controller moves the robot towards knot point P_1 and starts moving towards knot point P_2 once it is “close” to P_1. The tolerance for being “close” enough to a knot point before starting to move towards the next is determined by the proximity to obstacles.

The imperfect robot controller will be unable to follow the specified path exactly and will follow a slightly different path, shown as “Actual path” in Fig 23. Thus the knot points used above are conceptually similar to the knot points defining a B-spline, where moving the knot points shapes the spline. In this case moving the knot points shapes the robot path accordingly.

The disadvantages of using a search vector as described above are:

- Any generated path would have to be checked for being in the robot’s workspace. If the robot’s workspace is a convex surface then it is sufficient to ensure that the knot points lie in the robot’s workspace since all intermediate points are then guaranteed to lie within the workspace.
- The dimensionality of the search space is dependent on the discretization of the path chosen - the finer the discretization the higher the dimensionality of the search space. As explained earlier in this section, the input for large and heavy hydraulic systems can be approximated by a coarsely discretized input and therefore the dimensionality problem is not very serious for the class of problems addressed here.

An alternative method of constructing the search vector is by using a joint space path instead of an end-effector path. The swing/boom/stick/bucket joint path from start-state to end-state is discretized and the search vector consists of a series on joint knot points. These joint knot points are similar to the end-effector path knot points in that the controller attempts to move to swing knot point 1, and moves to knot point 2 once it is “close” to knot point 1. The controller handles the knot points for the other joints in a similar manner. Note that the joints are controlled independently.

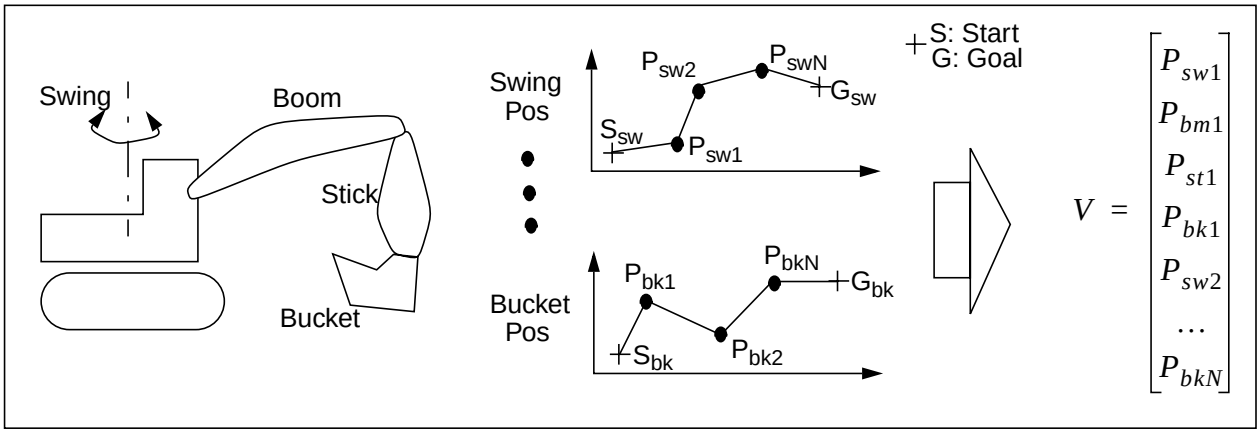


FIGURE 24. Another approach to generating search vector

The advantage of choosing the cartesian space path to map to the search space is that the dimensionality of the search space is independent of the number of robot joints. Also, since an explicit end-effector path is specified, obstacles in the workspace can be easily taken into account by severely penalizing path options causing collisions with obstacles. This is not easy to do with the joint space approach where an explicit end-effector path is not specified - the joint commands are not coordinated to specify an end-effector path.

On the other hand, with the joint based approach it is very easy to impose joint limits (and hence robot workspace limits) on any generated path since the dimensions of the vector are joint positions. Another minor advantage is that robot joint coordination is easier to visualize using this search vector scheme as compared to the previous one where the joint motions have to be extracted from the end-effector motions using the inverse kinematic mapping. The joint coordination can yield insights into how one set of motions uses the robot dynamics to be more optimal than another.

Both the above approaches to map from cartesian space to search space yield a bijective mapping i.e. the mapping is one-to-one and invertible. Thus every point in search space corresponds to a unique path in cartesian space. The existence of an inverse is required to ensure that the optimal point in the search space can be mapped back to a cartesian space path. The uniqueness of the inverse mapping, although helpful, is not a necessary condition.

7.1.2 Temporal method:

The search vector created using the above spatial approaches specify a path through space which is followed to some tolerance that is specifiable. An alternative approach to creating a search space is to specify an input stream discretized in time instead of space. The search vector would be 4*n dimensional for the 4-joint HEX, where n is the temporal discretization chosen. Each search vector thus consists of a stream of commands.

$$V = \begin{bmatrix} U_{sw}(k=0) \\ U_{bm}(k=0) \\ U_{st}(k=0) \\ U_{bk}(k=0) \\ U_{sw}(k=1) \\ \cdots \\ U_{bk}(k=n) \end{bmatrix}$$

; where $U_{jointName}$ is the input to a particular joint

; time t = 0...kT...nT where T is the time step size and n is the number of discretizations. (k is an integer)

FIGURE 25. The temporal search space vector

The spatial and temporal approaches each have their advantages and disadvantages. The temporal approach is essentially a command-space search, while the spatial approaches described above are combinations of command-space search and desired-path search. A spatial approach which only specifies knot point positions, and therefore only a spatial path (instead of a trajectory), has the disadvantage of being sub-optimal due to the “slow-down” when approaching each knot point. Specification of the velocity at the knot points would alleviate this problem. The temporal approach on the other hand does not suffer from this problem. However, the disadvantage of using the temporal approach is that the optimization must be performed at multiple path “durations” (Since each temporal path is a time-sequence of inputs, the length of the search vector is also the duration of the input). An optimization performed using a temporal search vector will yield an optimal input sequence of the **same** duration as the search vector. A search vector of a shorter duration may yield an equally (or more) optimal input for the same task.

7.2 Cost function

The cost function can be constructed to include quantities such as time and energy consumption. It can also be used as a convenient method of avoiding obstacles by penalizing paths that intersect (known and fixed) obstacles in the workspace. It can also be used to penalize paths that lie outside the robot’s workspace.

The cost function can be constructed as a linear weighted combination of all the above factors. The proposed work will use a cost function which considers time from start to goal, and an estimate of energy consumption.

In addition it will also have penalizing factors for paths that collide with fixed obstacles and for paths that violate workspace limits.

The energy consumption can be computed as:

$$E = \sum_{joints} \left(\int_{path} F_i dx \right) \quad \text{(EQ 7)}$$

where F_i is the force (or torque) generated by the i'th cylinder (or motor). The obstacle collision factor is:

$$O = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{matrix} \text{; if the path does not collide with an obstacle} \\ \text{; if the path collides with an obstacle} \end{matrix} \quad \text{(EQ 8)}$$

The kinematic limit factor is:

$$K = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{matrix} \text{; if the path does not cause violation of kinematic limits} \\ \text{; if the path lies outside the workspace} \end{matrix} \quad \text{(EQ 9)}$$

The cost function is:

$$C_{path} = \lambda_{time} T + \lambda_{energy} E + \lambda_{obstacle} O + \lambda_{kinematic} K \quad \text{(EQ 10)}$$

where the weights λ_{time}, λ_{energy}, $\lambda_{obstacle}$, $\lambda_{kinematic}$ can be adjusted to adjust the relative importance of the competing criteria. In the tests done to verify this approach (described later in this document) λ_{time} was set to 1 while the others were set to 0. Note that the above list of terms in the cost function is not exhaustive. The cost function definition is flexible enough to allow the addition of any other terms desired.

The computation of the "time" cost of a path is the most computationally expensive part of the cost calculation. This computational cost can be somewhat alleviated by the use of heuristics to approximate costs for path options that are very far from the region of interest. The cost function could therefore be somewhat intelligent about deciding how much expense to devote to computing the cost of a path option.

The energy consumption term E refers to the energy expended at the cylinder since it is computed as a product of the force/torque at the cylinder/motor and the displacement. It would be desirable to add hydraulic system losses to E since the sum of the losses and work done at the cylinders is the true estimate of the power delivered by the engine. However, using the model structure outlined earlier (Sec 6.2), it is not possible to compute the hydraulic losses. The use of a more detailed but slower analytical model can address that issue, but this work does not intend to incorporate hydraulic losses.

7.3 Constraint surface

The constraints due to the actuator and linkage dynamics are imposed through the machine model described in Sec 6.2. Each search space vector is evaluated by predicting the response of the robot to the command sequence represented by the search vector. This process of predicting the response allows the imposition of the dynamic constraints.

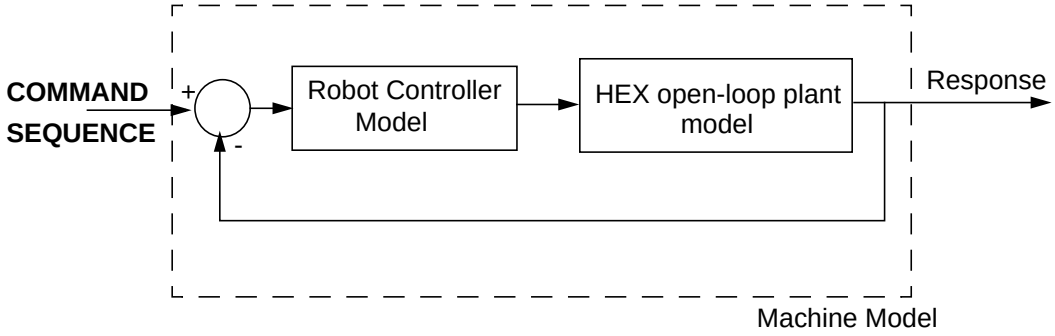


FIGURE 26. Machine model imposes dynamic constraints

One of the factors in determining a good loading cycle is load spillage which is required to be minimal. Excessive spillage can require additional cleanup operations which result in wasted time. There is no easy way of incorporating a “spillage cost”. However heuristics used to minimize spillage can be incorporated as constraints. For instance, during a trench digging (or trenching) operation, the mouth of the bucket is kept level when lifting it out of the trench, which could be incorporated as a constraint in the robot model.

7.4 Path optimization algorithm

There are a number of algorithms that can be used for a problem of this nature. To gauge the nature of the cost surface a simple downhill simplex algorithm was implemented. This algorithm does not require gradient information - it only uses function evaluations to move towards the optimum. This is important since gradient information is not readily available. The search space can only be evaluated at a point using the scheme shown in Fig 26. Note that each search space point corresponds to a robot path option to go from start to the goal.

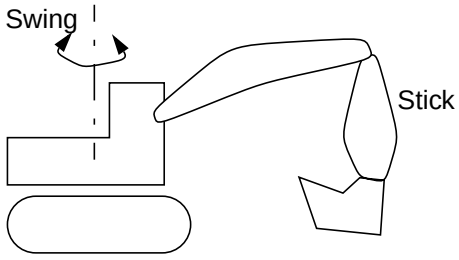


FIGURE 27. Simple two joint robot (Constructed by fixing the boom and bucket joints)

A simple two-joint approximation of the HEX model was constructed to study the use of different optimization algorithms for the HEX path optimization. The simple model had first-order approximations for the swing and stick joints, but variable time constants for the first-order approximations. The time constant for the swing joint was dependent on the position of the stick joint. This modeled the effect of the stick joint on the swing inertia. The stick time constant was dependent on the swing velocity to model the effect of centrifugal force on the stick joint.

$$\tau_{sw}\dot{\Theta}_{sw} + \Theta_{sw} = \Theta_{swCommand} \quad \text{(EQ 11)}$$

$$\tau_{st}\dot{x}_{stickCyl} + x_{stickCyl} = x_{stCommand} \quad \text{(EQ 12)}$$

The time constant τ_{sw} varies linearly from 1.0 secs to 2.0 secs as the stick joint moves from minimum to maximum, while τ_{st} varies from 0.2 secs for stick link moving outwards at maximum swing velocity to 0.6 secs when the stick is moving inwards (against the centripetal force) at maximum swing velocity. No hydraulic system effects were included in this model.

The results from the downhill simplex algorithm indicated that the cost surface (for the simplified system) had a number of local minima. The results of the downhill simplex optimization were dependent on the start point for the optimization and yielded a different result for sufficiently distant points in the search space. The algorithm would converge to the same result when starting from search space points which were not very distant from each other.

The results of the downhill simplex algorithm suggest that the optimization requires a method of getting close to the global minimum. One well known optimization technique well suited for problems of this nature is simulated annealing - SA for short. SA is a well known optimization technique first introduced by Kirkpatrick et al.[10] in 1983. It has gained a lot of prominence in recent years due to its success in solving problems in different domains which had previously been considered unsolvable, including effectively “solving” the NP hard travelling salesman problem. The main idea behind SA is drawn from thermodynamics, specifically with the way that liquids freeze and crystallize. At high temperatures the molecules of a liquid move freely with respect to one another. If the liquid is cooled thermal mobility is lost. However if the cooling is done slowly the atoms are able to line themselves up and form a pure crystal that is completely ordered over distances of over a billion times the size of an atom. This crystal is the state of minimum energy for this system. The same result is not seen when liquids are cooled quickly or quenched, in which case the liquid ends up in a polycrystalline or amorphous state with higher energy. So, the essence of the process is slow cooling, allowing ample time for redistribution of the atoms as they lose mobility, and is essential for ensuring that a low energy state will be achieved.

The algorithm starts off at a high temperature and changes from state 1 to state 2 with probability:

$$p = e^{\left[-\frac{(E_2 - E_1)}{kT}\right]} \quad \text{(EQ 13)}$$

where E_1 is the energy (or cost) at state 1 and E_2 is the energy at state 2. If $E_2<E_1$ then the move has a probability greater than 1 and is therefore always accepted. An uphill move ($E_2>E_1$) is accepted with probability p given by Eqn 13. When the temperature is high, uphill moves are more readily accepted while at lower temperatures the probability of an uphill move is low.

SA was used with the simple two-joint model (Eqn 11, Eqn 12). The SA routines were implemented using a software library - *Acacia* - provided by Tamal Mukherjee ([21]). This library has been compiled over the years by researchers in the Electrical and Computer Engineering department at CMU, and has been tested for problems in the domain of Electronic Computer-Aided Design.

SA was consistently successful at moving close to the same global minimum. Since SA uses a stochastic process to determine whether or not to accept an uphill move, it is not guaranteed to converge to a global minimum. For instance, at a low temperature the algorithm may decide to move to a higher energy state and then take a long time to get out of the local minima. The recommended procedure is to start SA from a number of different start positions and use the results of all the runs to pick the minimum. Alternatively SA could be used to move the search close to the global minimum and a simple method, such as the downhill simplex be used to move to the global minimum. However, it is difficult to determine when the search has reached “close enough” to the global minimum so that a downhill simplex can be used.

SA requires four principal components:

- **Problem representation:** For the two-joint example a joint space search vector was constructed as described in Sec 7.1. The search vector contained two knot points.
- **Move set:** The move-set is a function that generates a new point in the search space for the annealing to consider as a candidate move. The move-set generator for this problem consisted of changing one of the dimensions of the search vector (which corresponds to the position of one of the joints of the robot) to a random value within the kinematic limits of the joint position.This was implemented by generating a random number between 0 and 1, with 0 corresponding to the minimum joint position and 1 corresponding to the maximum joint position.
- **Cost:** The cost of a point was simply evaluated as the time to execute the path represented by the search space point. No heuristics were used to simplify the cost computation - all paths were simulated irrespective of how absurd they might have been. Note that the use of a spatial approach to search vector generation guarantees that the goal will be reached.

- **Cooling schedule:** The tests used a cooling schedule proposed by Huang, Romeo, and Sangiovanni-Vincentelli[7]. Using this schedule the hot temperature (or start temperature) is determined using White's [32] method. In White's method the initial temperature for the optimization is determined by performing a certain number of moves at a very high temperature, $T_{infinity}$ to determine the standard deviation of the cost function $\sigma_{infinity}$. Then the following formula is used to compute the starting temperature:

$$T_{white} = \frac{(-3)\sigma_{\infty}}{\log(accept)} \quad \text{(EQ 14)}$$

where *accept* is a parameter that must be specified by the user. The parameter corresponds to the desired acceptance ratio at $T_{infinity}$ and was set to 0.85. The temperature during the optimization is updated by the formula:

$$T_{i+1} = T_i \exp\left(\frac{-\lambda T_i}{\sigma_i}\right) \quad \text{(EQ 15)}$$

where s_i is the standard deviation of the accepted costs at the previous temperature. For further details of the HRSV schedule see [7].

The results from using SA to compute the optimal motion using the simple model are described in Sec 8 below.

8. Results

The results from the simulated annealing optimization for the two joint problem are shown in the plots below. The robot was commanded to move from a start position of (swing = 0^o; stick = -90^o) to a goal position of (swing = 180^o; stick = -90^o). Although the stick positions at the initial and final configurations were the same,

the optimal set of motions involved moving the stick to obtain a speedup of the swing joint. This is clearly visible in Fig 31 where the optimized and unoptimized motions are compared.

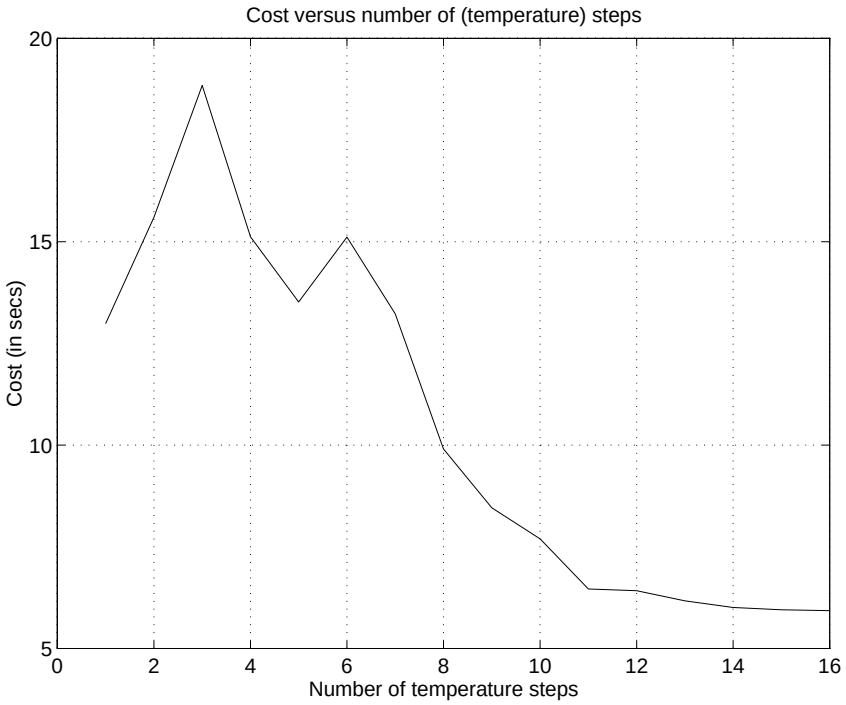


FIGURE 28. Cost as a function of number of temperature steps

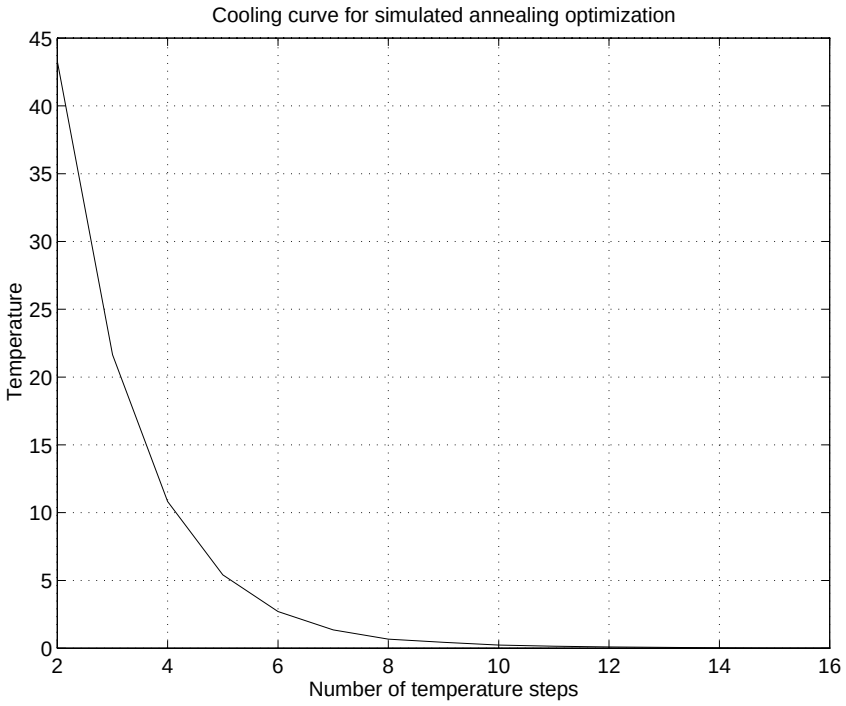


FIGURE 29. Cooling curve

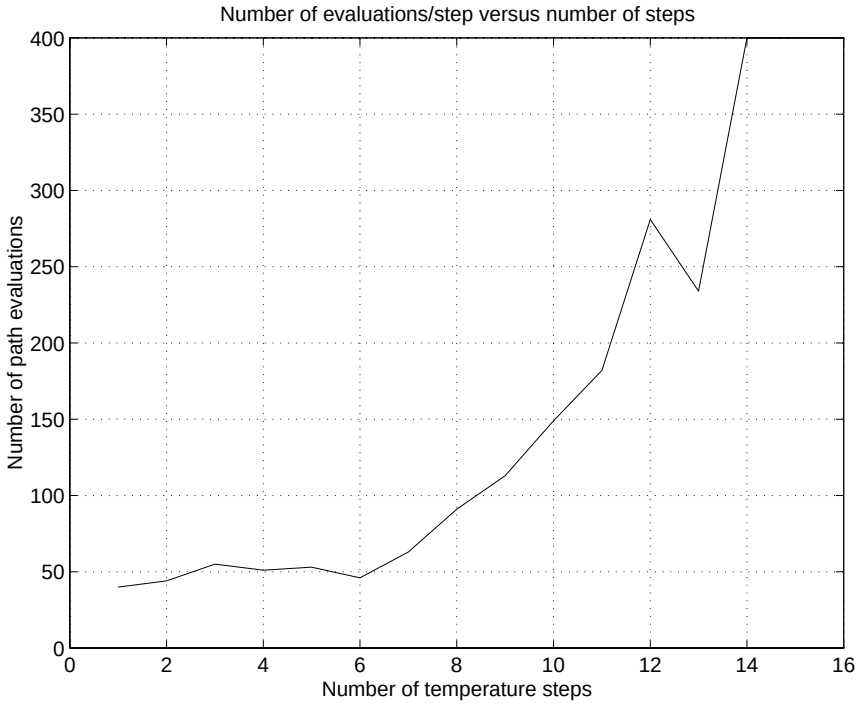


FIGURE 30. Cost evaluations per step

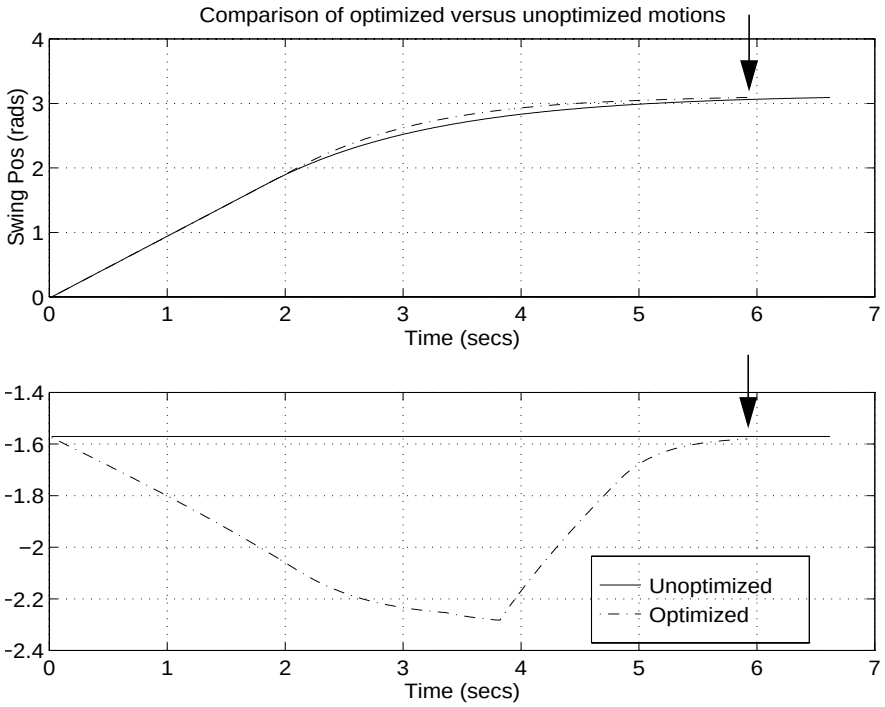


FIGURE 31. Optimized versus unoptimized motions
(Goal reached in 5.9 secs using optimized motions - 12% faster)

To study the effect of increasing the discretization of the input **u(t)**, which is reflected in the length of the search vector **u[k]**, the optimization was performed for 1 knot point (which results in a 2 dimensional search vector for the two-jointed robot) through 25 knot points (50 dimensional search vector). The SA procedure was used with the same parameters as above, and with the same start and goal configurations as before.

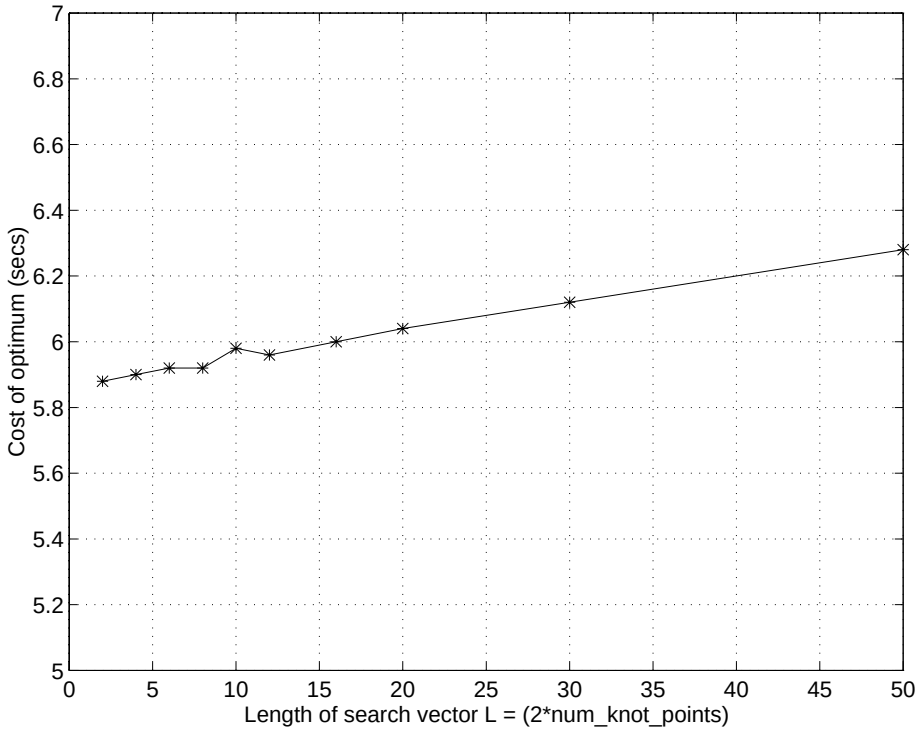


FIGURE 32. Optimum cost versus input discretization

The optimum costs in the above plot were obtained for a single run; a “run” consisted of starting the annealing algorithm from a starting point and letting it run until it had reached steady-state. “Steady-state” is defined through a cost change tolerance - if the cost does not change by more than the tolerance over some number of cycles then steady-state has been reached.

The cost of the optimum as a function of the path discretization seems to increase linearly with the number of knot points. The change in the value of the optimum is less than 7% for a change in number of knot points from 1 to 25. I think that this effect might be due to the fact that adding dimensions to the search space causes more local minima and searching among the local minima gets more complicated as the dimensionality increases. The question of how to pick the “best” number of knot points requires more attention.

Fig 33 indicates that the increase in the computation time with increasing dimensionality of the search space is low when the number of knot points is low and increases at a less than exponential rate for higher dimensions. It appears that for this case the computation time is bounded by $L^{2.2}$ - the function $L^{2.2}$ is also plotted

on Fig 33. It is encouraging to observe that the increase in computation is not exponential in the number of search space dimensions.

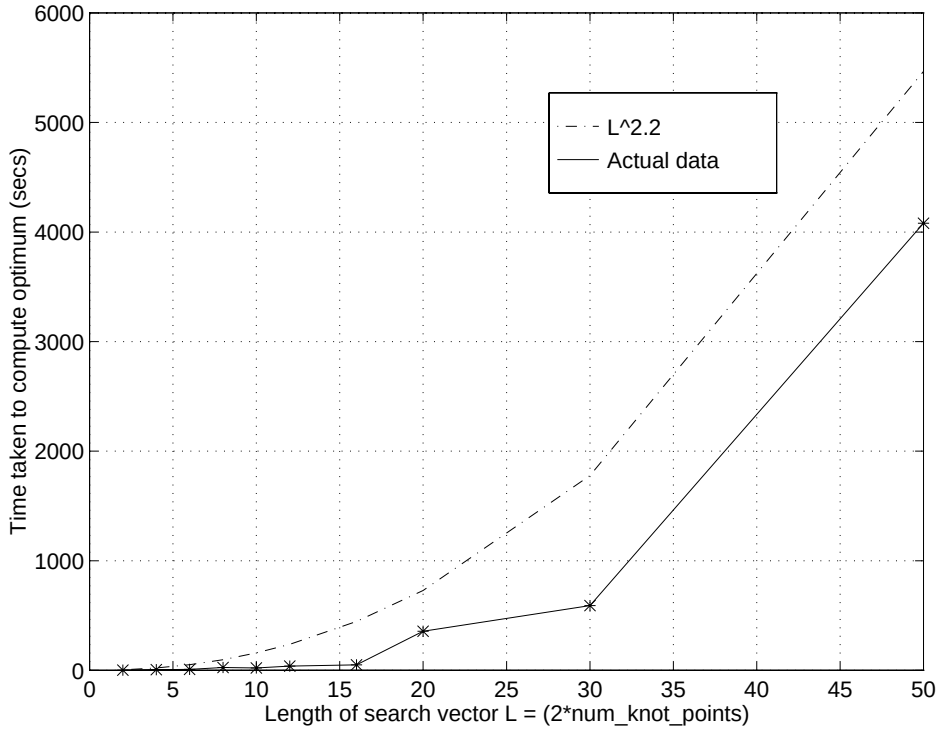


FIGURE 33. Computation time to reach optimum versus discretization of the search vector

9. Summary

This document proposes an approach to solving the optimal motion generation problem for a hydraulic robot. The work done thus far provides much of the groundwork needed to implement the optimization methodology. This includes:

- Fast hydraulic robot models created using memory-based learning which capture the significant hydraulic system effects. These fast models will allow the cost computation to be reasonably fast and accurate. Slower analytical models would render the whole scheme too impractical due to their computational expense.

  The fast HEX model currently takes 1 sec of computation time to simulate 10 sec of machine motion.

- Identification of an optimization algorithm (Simulated Annealing implemented using *Acacia* [21]) that has been shown to be reasonably effective in solving the optimal motion computation problem.

10. Tasks/Schedule

The proposed schedule of work is described below:

- March-August 1998
  - Complete and test the complete HEX model (described in Sec 6.1-Sec 6.3)
  - Incorporation of known obstacles in the workspace. Obstacles can be introduced in the workspace by penalizing paths that intersect obstacles.
  - Identify loading configurations for which optimal motions will be computed. This will consist of identifying start and end locations for the bucket, as well as locations of the obstacles, such as the truck.
  - Identify the actual form of the cost function to be used for this research.
  - Select approach to search space construction (Spatial/Temporal).
  - Identify a way of selecting the discretization of the path (i.e. selecting the number of knot points to use).
  - Use an “expert” human to perform the above identified tasks and record the motions. This will serve as a benchmark or reference by which the computed optimal motions will be measured.

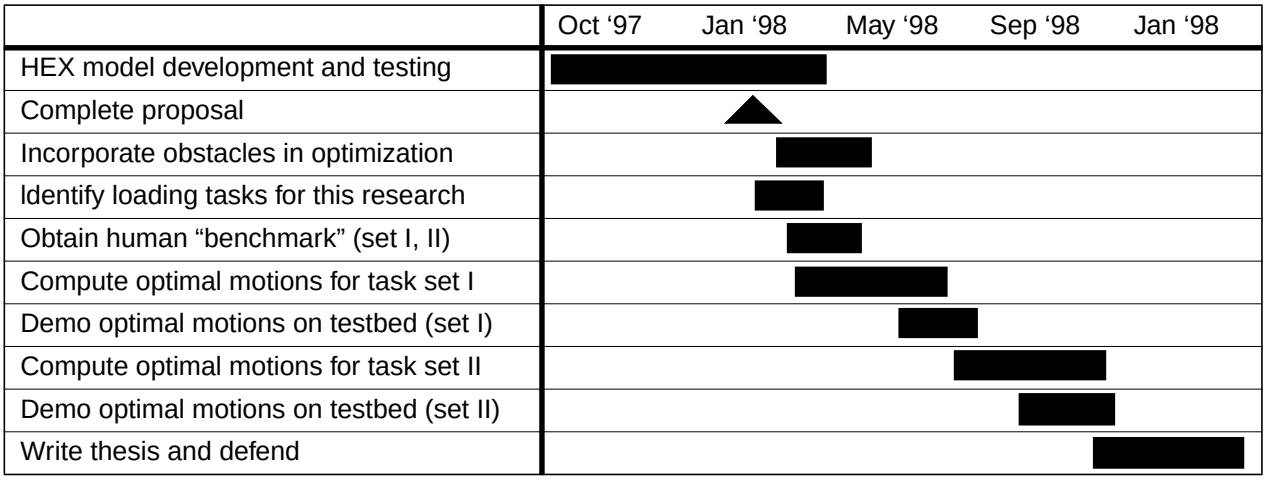
    The above operations will be performed twice:

    Set I: Here the “loading” operation only consists of free space motion with no soil (or load) in the bucket. This is the task for which optimal motions will be computed during the first phase of the proposed research, as described in the two bullets below.

    Set II: Here, the operation will be performed with a load in the bucket and including the dumping operation in the task. Digging will not be included in either case. Therefore the task specification will start upon completion of the dig, will involve moving the bucket to the goal, dumping it there, and moving back to the start. This case (involving a load in the bucket, and the dumping operation) will be handled in the second phase of the proposed research (August-December 1998).
  - Use proposed approach to compute optimal motions for the selected loading tasks (set I).
  - Implement the above computed optimal motions for task set I on the HEX testbed and compare against the performance of the “expert”.
- September 1998 - December 1999
  - Study the effect of variable loads in the bucket. This will involve testing the HEX model in conditions of varying bucket loads. The tests will include a combination of step and ramp tests.
  - Compute optimal motions for task set II (which includes the dumping as part of the task).

- Implement the above computed optimal motions for task set II on the HEX testbed and compare against the performance of the “expert”.

- January-March 1999
  - Thesis writing and graduation!!!



11. Challenges

Some key challenges in performing the above mentioned schedule of work are:

- The tasks to be optimized must be chosen carefully. The HEX model is not perfect and has limited fidelity. For instance, if the model can estimate the time to move from point A to point B to within 1 sec. then the task chosen must be such that the total time of the task is much larger than 1 sec.
- Introduction of obstacles in the optimization may be non-trivial. An incorrect choice of “obstacle cost” could easily cause the optimization to get stuck.
- Handling the dumping maneuver is a difficult problem. The dumping maneuver typically consists of a combination of bucket and stick motion which causes the load to spill at the desired location. Imposing this stick/bucket motion in the optimal motion generation may be a challenge. If it does turn out to be more than I can handle, I intend to use a fixed dumping strategy, i.e. a fixed set of motions for the dumping which will be incorporated in the complete set of motions.
- The HEX model has not yet been tested with different bucket loads. The model can probably handle different bucket loads but that remains to be seen. There is also the question of estimation of the bucket load. Although the HEX response is insensitive to small differences in bucket load, the differ-

ence in response between a full bucket and a half-full bucket has not yet been quantified. This may affect the second phase of the proposed schedule of work. At worst I expect that this will require another round of data collection to rebuild/re-tune the model.

12. Contributions

The contributions of the proposed work are described below.

- The main contribution of the proposed work is in the incorporation of complex actuator effects in optimal motion planning for hydraulic robots. This work focuses on the class of large hydraulic robots and addresses one of the key features that distinguishes such systems from typical electric drive robots - the complex actuator interactions.
- Constructing fast hydraulic robot models using memory-based learning is a new and innovative approach which shows considerable promise as an effective means of capturing complex actuator interactions in a hydraulic robot. This approach may be extended to learn a machine model directly from a testbed by collecting the training data needed during operation of the machine.
- This is the first time that SA has been used to solve the optimal motion generation problem. Although SA does not guarantee that the global minimum will be reached, it improves the chances of reaching the global minimum by accepting uphill moves early in the exploration of the search space. Methods in the literature have either used a gradient-based search (which tends to get stuck in local minima) or search space discretization followed by an exhaustive search of the discretized grid.

13. References

[1] Bobrow, J.E., "*Optimal robot path planning using the minimum-time criterion*", IEEE Journal of Robotics and Automation, vol. 4, No. 4, pp. 443-451, August 1988.

[2] Bobrow, J.E. Dubowsky, S., Gibson, J.S., "*Time Optimal Control of Robotic Manipulators along Specified Paths*" International Journal of Robotics Research, Vol. 4, No. 3, 1985.

[3] Castellote, G.P., Cannon, R.H., "*Proximate Time-Optimal Algorithm for On-Line Path Parameterization and Modification*", Proceedings of the IEEE Conference on Robotics and Automation, Minneapolis, April 1996.

[4] Centikunt, S., Chiu, H.T., "*A Study of Learning Controllers for Tip Position Control of a Flexible Arm using Neural Networks*", Proc. of the Winter Annual Meeting of the ASME, 1991.

[5] Craig, John J., "*Introduction to Robotics: Mechanics and Control*", 2nd Ed., Addison-Wesley Publishing Company, 1989.

[6] Fiorini, P, Shiller, Z, "*Time-Optimal Trajectory Planning in Dynamic Environments*", Proceedings of the IEEE Conference on Robotics and Automation, April 1996.

[7] Huang, M.D., Romeo, F., Sangiovanni-Vincentelli, A., "*An Efficient General Cooling Schedule for Simulated Annealing*", Proceedings of the International Conference on Computer Aided Design, Nov. 1986, pp. 381-384.

[8] Kelly, A., Stentz, A., "*An Approach to Rough Terrain Autonomous Mobility*", To appear April 1 1998 in Autonomous Robots.

[9] Khatib, O., "*Real time obstacle avoidance for manipulators and mobile robots*", International Journal of Robotics Research, Vol. 5, No. 1, 1986.

[10] Kirkpatrick, S., et al., "*Optimization by Simulated Annealing*", Science, Vol. 220, No. 4598, pp. 671-680, May 1983.

[11] Lawrence, P.D., et al., "*Coordinated and Force-Feedback Control of Hydraulic Excavators*", Fourth International Symposium on Experimental Robotics, ISER '95.

[12] Singh, N., et al., "*Coordinated Motion Control of Heavy Duty Industrial Machines with Redundancy*", Robotica, 1995, Vol. 13, pp. 623-633.

[13] Li, Z.D., Corke, P.I., Gurgenci, H., "*Modeling and Simulation of an Electro-Hydraulic Mining Manipulator*", Proceedings of the IEEE Conf. on Robotics and Automation, April 1997.

[14] Lin, L.C., Yih, T.W., "*Rigid Model-Based Neural Network Control of Flexible-Link Manipulators*", IEEE Transaction on Robotics and Automation, Vol. 12, No. 4, Aug. 1996.

[15] Martin, B.J., Bobrow, J.E., "*Minimum Effort Motions for Open Chain Manipulators with Task Dependent End-Effector Constraints*", Proceedings of the IEEE Conference on Robotics and Automation, April 1997.

[16] McDonell, B.W., Bobrow, J.E., "*Modeling, Identification, and Control of a Pneumatically Actuated Robot*", Proceedings of the IEEE Conference on Robotics and Automation, April 1997.

[17] Medanic, J., Yuan, M., Medanic, B., "*Robust Multivariable Non-linear Control of a Two Link Excavator: Part I*", To be presented at the IEEE Conference on Decision and Control Systems, Dec 1997.

[18] Merritt, H.E., "*Hydraulic Control Systems*", Wiley, New York, 1967.

[19] Moore, A., Atkeson, C., Schaal, S., "*Locally Weighted Learning for Control*", 1997, To appear in AI Review.

[20] Moore, A., Atkeson, C., Schaal, S., "*Memory-based Learning for Control*", 1995, CMU Technical Report CMU-RI-TR-95-18.

[21] Ochotta, E., Mukherjee, T., "*Programmers Guide to the Reconfigurable Simulated Annealing Library*".

[22] Press, W.H., et al., "*Numerical Recipes in C - The Art of Scientific Computing*", Second Edition, 1996, Cambridge University Press.

[23] Rajan, V.T., "*Minimum Time Trajectory Planning*", Proceedings of the IEEE Conference on Robotics and Automation, St. Louis, MO, 1985.

[24] Rowe, P., Stentz, A., "*Parameterized Scripts for Motion Planning*", Proceedings of the International Conference on Intelligent Robots and Systems, Grenoble, France, Vol. 2, pp. 1119-1124.

[25] Schneider, J., Moore, A.W., "*A Locally Weighted Learning Tutorial using Vizier 1.0*".

[26] Shiller, Z., Dubowsky, S., "*On Computing the Global Time-Optimal Motions of Robotic Manipulators in the Presence of Obstacles*", IEEE Transactions on Robotics and Automation, Vol. 7, No. 6, 1991.

[27] Shin, K.G. and McKay, N.D., "*Minimum-time Control of Robotic Manipulators with Geometric Path Constraints*", IEEE Transactions on Automatic Control, Vol AC-25, 1985.

[28] Slotine, J.J., Yang, H.S., "*Improving the Efficiency of Time-Optimal Path-Following Algorithms*", IEEE Transactions on Robotics and Automation, Vol. 5, no. 1, pp. 118-124, Feb 1989.

[29] Song, B., Koivo, A.J., "*Neural Adaptive Control of Excavators*", Proceedings of the Intelligent Robots and Systems Conference, Pittsburgh, PA, April 1995.

[30] Vukobratovic, M., Kircanski, M., "*A Method for Optimal Synthesis of Manipulation Robot Trajectories*", Transactions of the ASME, Vol. 104, June 1982.

[31] Wapenhans, H., et al., "*Optimal Trajectory Planning with Application to Industrial Robots*", International Journal of Advanced Manufacturing, Vol. 9, No. 1, 1994.

[32] White, S., "*Concepts of Scale in Simulated Annealing*", IEEE International Conference on Computer Design, Port Chester, New York, 1984, pp 646-651.

[33] Zhang, J., Herp, A., "*Design of a Fuzzy Controller for Optimal Execution of Subgoal-Guided Robot Motions*", pp. 2986, Proceedings of the IEEE Conf. on Robotics and Automation, 1994.

[34] Zomaya, A.Y., "*An Error-learning Neural Network for Tuning of Robot Dynamic Models*", International Journal of Systems Science, Vol. 26, No. 1, 1995.