Optimal motion computation for hydraulic robots

Murali Krishna

CMU-RI-TR-99-30

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania - 15213

September 30th 1999

UMI Number: 9964600

Bell & Howell Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Carnegie Mellon

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890

Thesis

Optimal Motion Computational for Hydraulic Robots

Murali Krishna

Submitted in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the field of Robotics

ACCEPTED:

John Bares	Thesis Committee Chair	8-27-99 Date
Matthew T. Mason	Program Chair	September 2, 1999 Date
James H. Morris	Dean	9/2/99 Date

APPROVED:

Paul Christiano	Provost	7 September 1999 Date

Abstract

Optimal motion planning for hydraulic robots is an important problem that must be solved as the traditional domains of hydraulic machines, e.g. construction, excavation, forestry etc., embrace varying levels of automation and robotics. There is almost no previous work on optimal motion computation for hydraulic robots. Most work done to date on optimal motion planning has focused on electric-drive manipulators, which are fundamentally different from hydraulic robots due to the physical nature of fluid power actuation versus electric power. Also, hydraulic robots suffer from significant interactions between different joint actuators due to their coupling through the hydraulic system; this is in addition to the non-linear linkage dynamic coupling. These differences preclude adapting previous approaches for hydraulic robots.

In this thesis, we propose and demonstrate an approach to computing optimal motions for hydraulic robots; this problem has never been addressed before in the literature. Although we only optimize free-space motions (due to the difficulty in modeling soil-tool interaction) we allow different end-effector loads. We use a robust search technique (Simulated Annealing) to search the robot's discretized command space for the temporal command sequence that minimizes an objective function while performing a given task (with specified start and end states) subject to obstacle avoidance and kinematic and dynamic constraints. The objective function for the optimization can be composed of any quantifiable measures such as time, energy, joint forces etc. Our approach is the first to allow the use of a diverse set of measures in the objective function. All existing methods focus almost exclusively on time-optimality.

During the search, we use a robot model to evaluate the cost of a candidate command sequence. This hydraulic robot model was constructed using a novel approach, also developed in this research. This modeling approach allows the construction of computationally inexpensive hydraulic robot models that capture the significant actuator interactions that are typical of hydraulic robots.

We demonstrate our optimization approach by computing optimal motions for a Caterpillar 325A hydraulic excavator(HEX) robot. We present examples of computing time-optimal motions for 7 different tasks, as well as energy-optimal motions for the same 7 tasks. The time-optimal motions for all 7 tasks are demonstrated on a HEX testbed. The optimal motions computed for 4 of the 7 tasks are compared to that of a human expert performing the same tasks. The comparisons show that the computed optimal motions are as good as, or better, than the expert's motions. The energy-optimal motions offer some interesting insights into the operation of the HEX, while demonstrating the power of our method in handling different measures in the objective function.

To Amma and Appa

Acknowledgements

I've had a very enjoyable and fruitful "n" years at CMU and have had John Bares as a friend and advisor for "n-1" of those years. Besides providing me with a small but steady paycheck, he has helped pull me out whenever I got stuck in the graduate-school-bog. He was also the prime mover for my embarking on this rather long (doctoral) chapter of my life. Thanks John.

Thanks also to...

- Steve Lunzman and Lonnie Devier of Caterpillar Inc., who initiated me to the world of hydraulic systems, and gave me the opportunity to drive some really large and cool machines,
- Andrew Moore and Rob Rutenbar who helped direct my work by gracing my thesis committee,
- Sanjiv Singh and Tony Stentz for advice along the way....
- To a number of RoboGrads, assorted graduate students, and staff who chipped in with (mostly!) helpful advice and support. In particular I'd like to thank Marcel Bergerman (for stimulating discussions and literature on optimal motion planning), Tamal Mukherjee (for providing a lot of the Simulated Annealing code), Patrick Rowe (for discussions and help putting on demos), Bob McCall (for a lot of graphics support and discussions on random topics), Scott "Fix-all" Boehmke, Mike Rondinelli (for help with graphics editing), Jim Frazier (for spending many hours in the excavator testbed while I refined my demos), and Jorgen Pedersen (for keeping me awake all night for the first time in my life),
- To my siblings Meera, Rekha and Kannan for helping me feel that I was doing something useful all these years.....
- To the entire Autonomous Loading System(ALS) Team at CMU and Caterpillar Inc.

Table of Contents

CHAPTER 1 *Introduction* 1

Problem Description 5
Research Issues. 7
Solution approach 9

CHAPTER 2 *Previous work* 11

Hydraulic robot modeling 11
Optimal motion planning. 13
Optimal motion planning for hydraulic robots 13
Optimal motion planning for electric-drive robots 15
Optimization techniques 20
Summary. 21

CHAPTER 3 *Robot Modeling* 23

Description of the detailed HEX robot model. 25
Detailed Linkage Model. 26
Detailed actuator model. 27
Construction of the complete detailed robot model 34
Simplified Actuator Model. 35
Modeling the FF HEX operating mode. 37
Modeling the TL HEX operating mode. 38
Data collection. 40
Modeling actuator delays 41
Implementation detail 42
Fudge factors 42
Simplified HEX model (Using simplified actuator model) 43
Robot model results. 43
Fine-finish mode tests (Used MBL function approximators) 44
Truck-loading mode tests (Used neural network function approximators). 47
Extended free-space motion test 50
HEX joint controller model 53
Concluding remarks 54

CHAPTER 4 *Optimal Motion Planning* 55

Discretize the robot's command space 57
Use a robust search technique to find the optimal 57
Search space 58
Constraint surface 62
Cost function 62
Time term 63
Energy term 63
Obstacle term 64
Spillage term 66
Other terms 67
Optimization algorithm 67
Cooling schedule 70
Move set 73
Implementation details 81
Knot point selection 81
Selection of seed search vector 83
Random number selection 83
Concluding remarks 83

CHAPTER 5 *Results* 85

Computing minimal-time motions 88
Loading Configuration #1 (Tasks #1,2) 90
Loading configuration #2 (Tasks #3, 4) 95
Loading Configuration #3 (Tasks #5, 6) 98
Configuration #4 (Task #7) 101
An interesting result 102
Computing minimal energy motions 104
Loading configuration #1 (Tasks #1, #2) 105
Loading configuration #2 (Tasks #3, 4) 107
Loading configuration #3 (Tasks #5, 6) 109
Configuration #4 (Task #7) 111
Tabulation of all optimal motion planning results 112
Computation time for optimal motion planning 112
Cost variation studies 117
Case #1: 117

Case #2: *119*
An example application 122
Summary 124

CHAPTER 6 *Conclusions and Contributions* *126*

Summary of the research 126
Conclusions 127
Research issue: Hydraulic system modeling *127*
Research issue: Working with saturated actuators *128*
Research issue: Using measures other than time in the objective function *129*
Research issue: Avoiding obstacles *129*
Computation time for optimal motion planning *130*
Potential application *130*
Application to computing optimal paths for mobile robots *130*
Effect of using a low-level controller *131*
Obstacle buffers *131*
Unmodeled dynamics *132*
Contributions 133

CHAPTER 7 *Future work* *135*

CHAPTER 8 *References* *137*

CHAPTER 9 *Glossary* *143*

CHAPTER 10 *Appendix A: Details of HEX linkage dynamic equations* .. *146*

Mass Matrix 148
Vector of non-linear terms 151
Gravity terms 154
Actuator torque vector 155
Boom cylinder force to joint torque transform: *155*
Stick cylinder force to joint torque: *156*
Bucket cylinder force to joint torque: *157*

CHAPTER 11 *Appendix B: Robot modeling results* **159**

Table of Figures

FIGURE 1. A typical HEX truck-loading operation (Digging) 2
FIGURE 2. A typical HEX truck-loading operation (Dumping) 2
FIGURE 3. Problem description 5
FIGURE 4. Controller can compensate for disturbances 6
FIGURE 5. Illustration of example task used by Rowe (a) HEX over dig face (b) HEX over truck 14
FIGURE 6. Bounding rectangle for paths examined by Shiller et al. 17
FIGURE 7. Flow-chart for scheme by Shiller and Dubowsky 18
FIGURE 8. Description of the HEX robot as a combination of controllers, and open-loop plant (linkages + actuators) 24
FIGURE 9. HEX robot viewed as a combination of linkages and actuators 26
FIGURE 10. Centers of mass for loaded and empty buckets 27
FIGURE 11. Illustration of the operation of a spool valve 28
FIGURE 12. HEX hydraulic schematic (FF Mode) 29
FIGURE 13. HEX hydraulic schematic (TL Mode) 30
FIGURE 14. Schematic of bi-directional hydraulic cylinder 33
FIGURE 15. Section of the Bm response surface (FF Mode) 36
FIGURE 16. A typical 3-layer feed-forward neural network used for approximating an actuator response surface (TL Mode); Details of a single hidden (i'th) unit are shown in the lower figure 39
FIGURE 17. Illustration of the two chambers of a hydraulic cylinder 41
FIGURE 18. Bm, Bk Cylinder Velocity Plots (Test #1 - FF Mode) 45
FIGURE 19. Bm, Bk Cylinder Position Plots (Test #1 - FF Mode) 45
FIGURE 20. Sw Joint, St Cylinder Velocity Plots (Test #2 - FF Mode) 46
FIGURE 21. Sw Joint, St Cylinder Position Plots (Test #2 - FF Mode) 46
FIGURE 22. Bm, St, Bk Cylinder Velocity Plots (Test #1 - TL Mode) 48
FIGURE 23. Bm, St, Bk Cylinder Position Plots (Test #1 - TL Mode) 48
FIGURE 24. Sw Joint, St Cylinder Velocity Plots (Test #2 - TL Mode) 49
FIGURE 25. Sw Joint, St Cylinder Position Plots (Test #2 - TL Mode) 49
FIGURE 26. Velocity plots of the four HEX joints during the extended motion test 51
FIGURE 27. Tip error contributions due to each HEX joint during the extended motion test 51
FIGURE 28. (a) Tip position error (b) Bk tip velocity during extended motion test 52
FIGURE 29. Schematic of closed-loop HEX robot (Reproduction of Figure 8) 53
FIGURE 30. Reproduction of Figure 3 (Chapter 1) 55

FIGURE 31. Schematic showing our approach to optimal motion computation..... 56
FIGURE 32. Construction of the search vector..... 59
FIGURE 33. Relationship between cartesian space and search space..... 60
FIGURE 34. Triangular mesh of a pile of soil. The mesh has been filled in and shaded... 65
FIGURE 35. Flow-chart of SA search algorithm..... 69
FIGURE 36. Description of the search vector (Reproduction of Figure 32)..... 74
FIGURE 37. The list of all 46 move-sets. For a description of how perturbRadFactor is used, ref Equation 31, Equation 32. A move-set M is selected at the start of each temperature..... 76
FIGURE 38. Flow-chart showing the generation of a new search vector using move-set #M..... 79
FIGURE 39. Details of how the perturbation of a search vector element is done..... 80
FIGURE 40. Details of move-set #46, which creates a new vector that points in the direction of the minimum cost vector seen thus far in the optimization. ... 81
FIGURE 41. Initial seed search vector construction..... 83
FIGURE 42. Illustrations showing the loading configurations and tasks..... 86
FIGURE 43. Loading configuration #1 (a) start and (b) final HEX positions. These images were taken from a simulator. The terrain shown was constructed using a laser range scan of the actual terrain..... 91
FIGURE 44. Task #1 (Loading configuration #1) - TL mode - loaded bucket. 92
FIGURE 45. Task #2 (Loading configuration #1) - TL mode - empty bucket..... 92
FIGURE 46. Joint commands corresponding to the seed search vector (Initial Cmd Trace) and the optimal search vector (Final Cmd Trace) for task #1. Two knot points are used to discretize the command trace for each joint. .. 94
FIGURE 47. Joint commands corresponding to the seed search vector (Initial Cmd Trace) and the optimal search vector (Final Cmd Trace) for task #2. Three knot points are used to discretize the command trace for each joint. . 95
FIGURE 48. Loading configuration #2 (a) start and (b) finish HEX positions..... 96
FIGURE 49. Task #3 (Loading configuration #2) - FF mode - loaded bucket..... 97
FIGURE 50. Task #4 (Loading configuration #2) - FF mode - empty bucket..... 97
FIGURE 51. Loading configuration #3 (a) start and (b) finish HEX positions..... 99
FIGURE 52. Task #5 (Loading configuration #3) - TL mode - loaded bucket..... 100
FIGURE 53. Task #6 (Loading configuration #3) - TL mode - empty bucket..... 100
FIGURE 54. Task #7 (Configuration #4) (a) Initial seed motion (b) Real optimal motion. 101
FIGURE 55. Task #7 (Loading configuration #4) - FF mode - empty bucket..... 102

FIGURE 56. Comparison of time-optimal motions for task #5 using TL and FF HEX operating modes. 103
FIGURE 57. Comparison of time-optimal motions for task #6 using TL and FF HEX operating modes. 104
FIGURE 58. Task #1 (Loading configuration #1) - TL mode - loaded bucket. 106
FIGURE 59. Task #2 (Loading configuration #1) - TL mode - empty bucket. 107
FIGURE 60. Task #3 (Loading configuration #2) - FF mode - loaded bucket. 108
FIGURE 61. Task #4 (Loading configuration #2) - FF mode - empty bucket. 108
FIGURE 62. Task #5 (Loading configuration #3) - TL mode - loaded bucket. 110
FIGURE 63. Task #6 (Loading configuration #3) - TL mode - empty bucket. 110
FIGURE 64. Task #7 (Configuration #4) - FF mode - empty bucket. 111
FIGURE 65. Plots of the number of candidate motions examined during the time-optimal motion computation for the 7 tasks. 114
FIGURE 66. Plots of the average time taken to compute the time-optimal motions for all tasks. These plots use the data tabulated in Table 8. 114
FIGURE 67. Cost and Temperature variation as annealing optimization progresses. 118
FIGURE 68. Number of candidates examined during the annealing optimization for task #1. 119
FIGURE 69. Variation in cost for 3 instances each of 5 knot point cases, when computing time-optimal motions for task #1 (KP = Knot Points). 120
FIGURE 70. Variation in the cost of multiple runs (6 runs per knot point) of the SA optimization for task #1, and using the same seed vector. Each data point represents the results from one optimization. The thick line connects the mean values of the cost for each knot point. 121
FIGURE 71. Comparison of task #1 optimal motions for a normal Bm, and 50% faster Bm.123
FIGURE 72. HEX coordinate axes 146
FIGURE 73. Figure showing boom joint control points (Not to scale) 155
FIGURE 74. Figure showing stick joint control points (Not to scale) 156
FIGURE 75. Figure showing bucket joint details (Not to scale). 157
FIGURE 76. Velocity plots for test with St - and Sw + motions. 161
FIGURE 77. Position plots for test with St - and Sw + motions. 161
FIGURE 78. Velocity plots for test with Bm +, St +, and Bk - cylinder motions. 162
FIGURE 79. Position plots for test with Bm +, St +, and Bk - cylinder motions. 162
FIGURE 80. Velocity plots for test with Bm +, St -, and Bk + cylinder motions. 163
FIGURE 81. Position plots for test with Bm +, St -, and Bk + cylinder motions. 163

FIGURE 82. Velocity plots for test with Bm +, St -, and Bk - cylinder motions. 164
FIGURE 83. Position plots for test with Bm +, St -, and Bk - cylinder motions. 164
FIGURE 84. Velocity plots for test with Bm -, St +, and Bk + cylinder motions. 165
FIGURE 85. Position plots for test with Bm -, St +, and Bk + cylinder motions. 165
FIGURE 86. Velocity plots for test with Bm -, St +, and Bk - cylinder motions. 166
FIGURE 87. Position plots for test with Bm -, St +, and Bk - cylinder motions. 166
FIGURE 88. Velocity plots for test with Bm -, St -, and Bk - cylinder motions. 167
FIGURE 89. Position plots for test with Bm -, St -, and Bk + cylinder motions. 167
FIGURE 90. Velocity plots for test with Bm -, St -, and Bk - cylinder motions. 168
FIGURE 91. Position plots for test with Bm -, St -, and Bk - cylinder motions. 168

List of Tables

TABLE 1: Inputs/Output for HEX actuator MBL tables (FF Mode). 38
TABLE 2: Inputs/Output for HEX actuator model networks (TL Mode) 40
TABLE 3. Cooling schedule and freezing parameters for the HRSV cooling schedule. 72
TABLE 4. Possible HEX joint permutations that perturb at least one joint. 75
TABLE 5. Summary of tests conducted. 87
TABLE 6. Summary of tests conducted. 89
TABLE 7. Tabulation of optimization results. 112
TABLE 8. Average time taken in hrs. (over 3 parallel SA optimizations) to compute time-optimal motions for the 5 cases for each task. The number in parentheses indicate the average number of search vectors examined during the simulation. 113
TABLE 9. Table of mean, standard deviation, and minimum cost values for the optimization runs shown in Figure 70 (Task #1). 122
TABLE 10. Table of HEX linkage parameters. 149
TABLE 11. List of tests involving simultaneous Bm, St, and Bk motion. 160

CHAPTER 1 Introduction

Motion planning[1] for robot manipulators has been the subject of study for a number of years. With the development of robots for the manufacturing industries there arose a need to specify paths/trajectories for the commonly used open-chain industrial manipulator arms. The early motion planning methods were mostly concerned with avoiding other objects in the workplace, while performing any one of a myriad of tasks ranging from the simple pick-and-place to the more complex spray painting or welding jobs. All this work on manipulator arms led to the development of a number of motion planning approaches which are comprehensively summarized in [Hwong 92].

Optimal motion planning for robots was only a logical extension of the motion planning concept - i.e. optimizing time (or some other objective function) while generating a robot motion plan. The late 80's and early 90's saw much research conducted on the development of time-optimal controls for robots. All this focus was directed towards the commonly used electric-drive industrial manipulators. Not much attention was paid to hydraulic and pneumatic robots since the industrial manipulator world had renounced fluid power actuation fairly early, and used electric drives for almost all manufacturing robots.

However hydraulic and pneumatic machines dominate application areas like excavation, mining, forestry, and construction, due to the high forces needed for tasks such as digging in a pile of rock or carrying a tree. Hydraulic power offers an overwhelming edge over electric motors in such heavy

1. The terms motion planning and path planning are used inter-changeably in the literature. The former is usually used in the context of robot manipulators while the latter is used for mobile robots.

material handling applications due to the desirable characteristics of high force-to-weight ratios and actuator simplicity and durability. Electric motors tend to do well where operating speeds are high and forces are relatively low.

A commonly used hydraulic machine is shown in Figure 1, Figure 2. It shows a Hydraulic Excavator, commonly referred to as a HEX. It has four rotary joints and two independently controlled tracks that allow the HEX to move. The six actuated joints are -

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

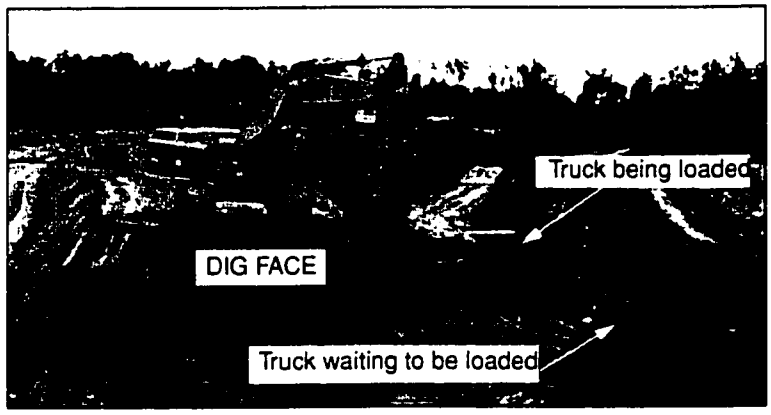


FIGURE 1. A typical HEX truck-loading operation (Digging).

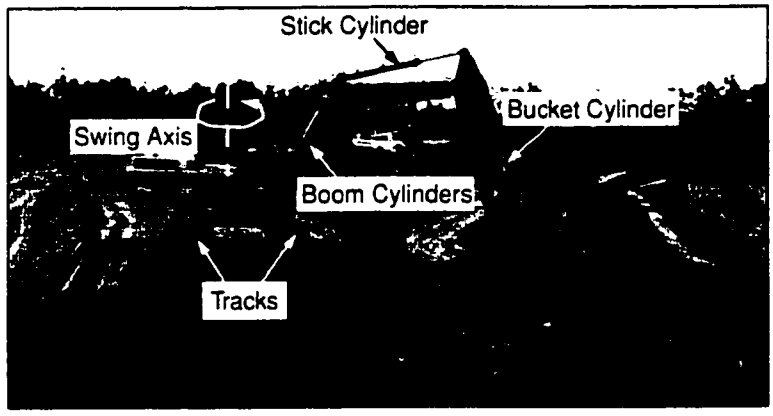


FIGURE 2. A typical HEX truck-loading operation (Dumping)

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 (linear actuators) visible in Figure 1, Figure 2. The swing joint and the tracks use rotary hydraulic actuators (hydraulic motors) not visible in the figure.

The HEX's two independently actuated tracks give it the ability to turn-in-place. The excavation activities are performed using the swing (Sw), boom (Bm), stick (St), and bucket (Bk) joints. The tracks are usually not actuated when excavating; they are periodically used to reposition the excavator.

The HEX is powered by a single diesel engine which drives two hydraulic pumps. The hydraulic pumps take low-pressure hydraulic oil and supply it at high pressure to 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 limited power source (single or multiple engines) 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(s) can supply. This causes the available power to be non-uniformly distributed amongst the different actuators.

Throughout this document, the term "hydraulic machine" or "hydraulic robot" will refer to a typical hydraulic machine or robot with the above characteristics.

A typical operating scenario for a HEX is shown in Figure 1 and Figure 2, where it is sitting on a pile (or bench) of soil. The HEX is being used to dig in the soil face in front of it, unload the bucket in a waiting truck, and return to the dirt pile for the next bucket load. One such dig-unload-return operation is commonly called a *loading pass*[1]. A HEX may take 3-6 *loading passes* to fill a truck, depending on the relative sizes of the HEX bucket and the truck bed. Once a truck is full, it pulls away and is replaced by a waiting empty truck, and the cycle continues. This type of loading operation is called a *truck-loading* operation. The HEX will occasionally use the tracks to back up on the bench before resuming the loading process. This *truck-loading* operation is repeated for many days or months in a typical *mass excavation* scenario. The repetitive nature of the process makes it very amenable to automation.

Besides the above task there are other scenarios involving other hydraulic machines which are equally amenable to either semi-automation or full automation. In a semi-automated scenario the HEX opera-

1. All terms marked in italics are special terms. Besides their definition in the document, they are also described in the glossary for easy reference at any time.

tor may be equipped with a smart cruise control that would perform a portion of the loading cycle, thus reducing operator fatigue while maintaining consistently high productivity.

As we look towards applying robotic technologies to hydraulic machines there arises the need to perform motion planning for them. In many cases automation can be a practical reality only if the robotic machinery is more productive (higher output in tons/hour) than a manually operated one, while offering lower production cost (cost/ton of material excavated). A mine operator or other end-user does not invest in new technology for the sake of technology alone. The user is primarily interested in higher productivity and lower costs while maintaining safe operation. For example, manufacturing robots allowed a drastic reduction in the manpower requirement on the factory floor and also offered consistent quality at high production rates. This offered a direct cost and quality benefit to the manufacturers who embraced robotics. Similar cost benefit must be apparent to the buyers of robotic excavation, mining and forestry machines in order for them to invest in the technology.

Automated machines 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 yield further gains in productivity since manually operated machines are designed for operator comfort and ergonomics. For instance, manual machines are designed to minimize operator jerk since that is a major cause of operator fatigue. Removal of such constraints (such as minimizing jerk) may allow the designer to make the robotic machine more productive and efficient than the manual 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, at least in the near-term future.

The most significant productivity gains can be realized by observing that most human operators use the machines sub-optimally[1] almost all the time, and the few expert operators cannot sustain high-productivity throughout an entire shift[2]. 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 develop a virtual trainer which can train operators to perform more efficiently by coaching them during operation. Optimal motion computation is thus not limited in its application to robotic machinery alone.

1. The term "optimal operation" here refers to machine operation while optimizing productivity. In general, the function being optimized could be any quantifiable measure.
2. This fact was gleaned from conversations with field test engineers at Caterpillar Inc., a leading manufacturer of excavation and mining equipment.

Optimal motion computation also has use in improving machine design to improve operating efficiency. Today there is no easy means of gauging how a particular design change affects, for instance, a *truck-loading* operation. The effect of the design change is measured by performing step and ramp tests for the redesigned joints. The results are extrapolated to estimate the effects on a typical operation. Qualitative input is also solicited from expert machine operators who operate a prototype with the design changes. Besides being a time-consuming process, all of this does not quite provide a direct mapping from design change to change in performance, i.e. it does not answer the question, "How does a particular design change affect the peak performance obtainable from the machine?". An optimal motion computation system can answer that question.

From the above discussion it is clear that there is a need to solve the optimal motion planning problem for hydraulic robots. The problem is defined below.

Section 1.1: Problem Description

The problem being addressed in this work is: "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".

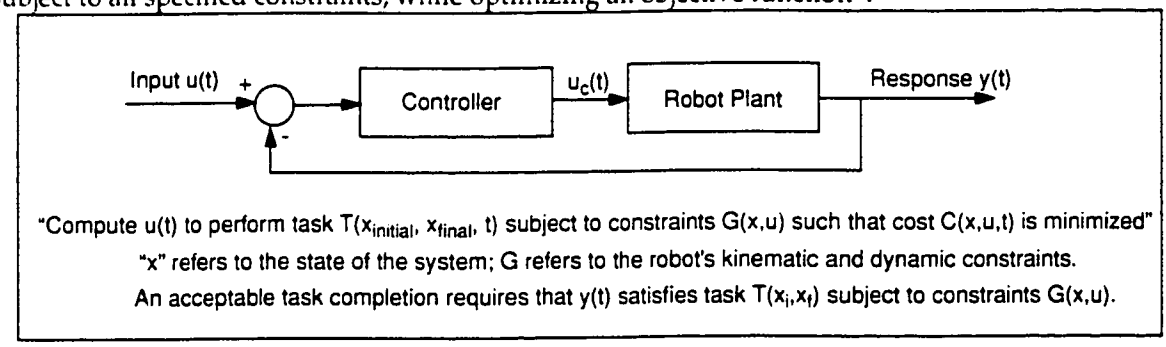


FIGURE 3. Problem description

The problem statement requires some clarifications.

- 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 C(x,u,t) can be 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 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 - 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 ([Craig 89]). The actuator and linkage dynamics together determine the motion of the links of a robot.
- The task specification is in the form of a start and goal state, where a state is composed of position and velocity variables. This work only addresses the non-digging part of the cycle since optimizing the digging is a very hard problem, and is itself a broad research topic. Optimizing digging is difficult due to the need to model the soil-tool interaction, and due to the unpredictable nature of the soil which may change radically from one dig to another.
- 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 controller is not required to be sophisticated enough to account for the non-linear robot characteristics. The advantage of 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) (Figure 4). 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 a command sequence, they do not suffer from the disadvantage of sensitivity to disturbances.

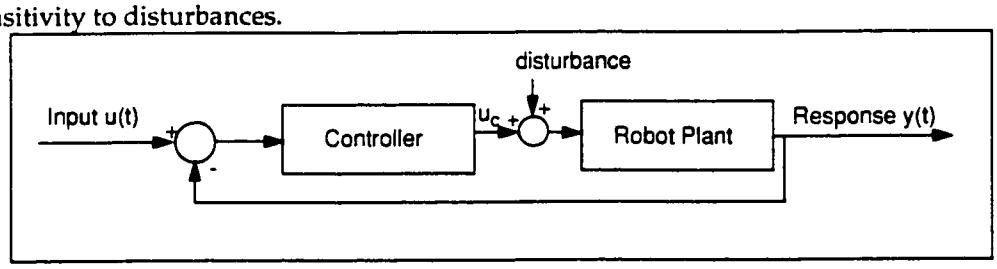


FIGURE 4. Controller can compensate for disturbances

It should be noted 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 different disturbances, $u_c^*(t)$ in the two cases would likely be different.

Section 1.2: Research Issues

There are several key characteristics of the problem described above that distinguish it from previous research. These characteristics combine to create a substantial research challenge.

1. The most important distinguishing characteristic is the use of hydraulic robots by this work, while previous studies have looked at electric drive robots. Unlike most industrial manipulators, hydraulic robots have complex actuator interactions. For instance, consider the operation of a HEX where the hydraulic actuators 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 hydraulic pumps, or the engine itself. The hydraulic system is forced to reduce the flow to the cylinders, thus reducing the joint velocities, to keep the engine from lugging or stalling. Such a power limited condition is very common during normal operation of a HEX. Pneumatic robots avoid flow limit problems by maintaining a large reservoir of pressurized gas. It is however not practical to have a reservoir of pressurized oil since the bulk modulus of liquids is much higher than that of gases.

   Electric drive industrial manipulators do not suffer from this power-limit condition. The individual joint motors have limitations, and these are defined by their torque-versus-speed curves, but it is usually possible for all the joints to move at their peak power consumption without reaching a total power limit.

   This characteristic makes the robot modeling problem much harder since it is not possible to capture these non-linear power-limit induced actuator interactions using simple linear approximations (The nature of the modeling problem is described in Chapter 3). These interactions are significant and cannot be ignored either. Hence, the challenge is to develop a computationally inexpensive way of modeling these interactions.

2. The other distinguishing factor is related to path following and actuator saturation. Most existing optimal motion planning methods rely on computing paths that are implemented on a robot using the time-optimal control scheme developed by Bobrow et al. [Bobrow 85]. This scheme computes a time-optimal control to follow a specified path; the "specified path" is the optimal cartesian or joint space path computed by the motion planning method.

   These methods were developed for industrial manipulators, which perform tasks such as spray painting and welding. Painting and welding require the robot tip to follow a specified spatial path. Following a specified path requires the robot's joint actuators to not saturate; if they are saturated,

the robot may not be able to follow the path. The time-optimal control scheme developed by Bobrow is based on this fundamental idea of avoiding actuator saturation.

Our application involves tasks that are similar to pick-and-place tasks where only the start and end states are specified, e.g. transporting a loaded bucket from the dig face to a truck. In our tasks, it is not important to follow a specified path. It is only important that the robot avoid colliding with obstacles in its workspace, and not tip itself over.

The implication of this feature is that, unlike existing methods, we do not have to avoid actuator saturation. Since actuator saturation is a very common feature of hydraulic robot operation, avoiding it would require significant reduction in the usage of the robot's capabilities, which defeats the purpose of optimal motion computation, i.e. improving robot productivity. Avoiding saturation necessarily causes under-utilization of the robot's capabilities. For instance, consider a control for an electric-drive robot which avoids motor saturation. To achieve that, it must ensure that its peak torque demand is below the torque limit of the motors. Since there will always be some uncertainty in the torque limit curve, the peak torque demand must remain below the limit by at least the amount of the uncertainty. This causes the under-utilization of the robot's capabilities.

For our tasks, any motion that results in a collision-free path is an acceptable candidate, and we wish to search among these candidates to select the motion that minimizes some objective function (e.g. time). Working with actuator saturation is a marked departure from the approach used by most researchers. It makes the problem more challenging since we can no longer search in the space of desired paths, as existing methods do. We cannot compute a collision-free spatial path and expect the robot to follow it, since actuator saturation may disallow following the specified path.

3. The other distinguishing factor is related to the method of computing optimal motions. Most previously developed methods rely on computing paths that are implemented on a robot using the time-optimal control scheme developed by Bobrow et al. [Bobrow 85]. This method computes a time-optimal control to follow a specified path (The "specified path" would be the optimal cartesian or joint space path computed by the motion planning method).

   The Bobrow method uses a heuristic based on the robot's (motor) torque limit curves to compute the control for following a specified path. Due to the inherently different physical processes that govern electric motor and fluid power actuation, it is not possible to merely transplant existing methods to solve the optimal motion planning problem for hydraulic robots. Electric motors produce a certain torque in response to a drive current and most robot control laws are based on that fact. Hydraulic robots do not function in that manner; a certain control input (hydraulic spool valve opening) results in a corresponding oil flow, and hence a certain actuator velocity. This velocity may depend on the external joint torque load.

   Therefore, since we do not have the luxury of a means of computing the time-optimal control for a specified path, the complete motion planning and implementation must be solved, i.e. we must compute the control inputs that will cause the robot to execute the desired motion in an optimal manner, and not just the optimal motion (or path).

4. Another important difference is in the use of a general user-specified objective function in our work. Our objective function can be composed of any quantifiable measures. This is in contrast to

almost all previous work which has focused exclusively on time optimality. The requirement of being able to use any objective function makes the problem more difficult since it is not possible to use heuristics that are tailored to time-optimal motion planning. We need a method that is applicable for a wide variety of objective functions.

5. The alleviating factor in our problem is the lack of clutter in the robot's workspace. For the class of applications that this work considers, the robot's workspace is relatively obstacle free. The obstacles that are present, such as a truck waiting to be loaded, are not very dynamic. This suggests that it is more efficient to plan paths assuming no obstacles (or assuming fixed known obstacles) and modifying them (e.g. halt and re-plan) 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.

   However, the lack of clutter does not mean that the space of possibilities has few local minima (or maxima). The cost surface is non-smooth due to the non-linear interactions between the different joints of a hydraulic robot. Even if the robot has to move from one point to another through a completely empty space (no obstacles in the environment), the cost surface will have multiple local minima due to the interactions between the different joints, i.e. their relative coordination can cause variations in the cost (time or other measure). For instance, moving the stick joint of our testbed excavator causes the swing joint to slow down. This interaction, although not negligible, is lower if the swing is already moving. If the swing is stationary, then it may not be able to move at all if the stick is being used. Such non-linear interactions cause the cost surface to be "bumpy" even though there are few obstacles in the environment.

Section 1.3: Solution approach

Having identified the characteristics that make our problem challenging and distinct from previous work, we looked at different means of addressing these challenges.

The foremost challenge was in developing computationally inexpensive hydraulic robot models that capture the actuator interactions. Our first attempt was to write the non-linear system of equations describing the HEX robot, and use a numerical solver to solve for the robot response. This approach was extremely computationally expensive when using a numerical solver of adequate robustness. We then constructed linear approximations of the HEX hydraulic system but soon realized that they were unable to capture all the actuator interactions. At this stage, the options were to use piece-wise linear approximations, or non-linear function approximations. The drawback of the former approach is the difficulty in determining how to decompose the non-linear actuator response functions to create the piece-wise linear approximations. The drawback of the latter approach is the need for data needed for the non-linear function approximation using methods such as locally weighted linear regression, or neural networks.

We chose the latter approach since although both approaches offered similar accuracy, the latter offered the promise of being easily extendable to other hydraulic robots. The non-linear function approximator approach was quite successful at modeling the actuator interactions. This actuator model is used in conjunction with a linkage model to build the complete robot model[1]. The model, and the results from using the model for a HEX, are described in Chapter 3.

Once the modeling problem had been addressed, the challenge was in developing an approach to using such a model to compute optimal hydraulic robot motions. Since the most feasible approach involved an explicit search of the space of possibilities, we selected a robust search approach to search the discretized space of robot commands. We use a general purpose optimization algorithm called *Simulated Annealing* to perform the optimization. We discretize the space of robot commands and search in that space. Each point in the search space corresponds to a temporal sequence of commands. As we search the space, we use the robot model to evaluate each candidate point by simulating the motion resulting from the corresponding command sequence. This approach is described in Chapter 4, and the optimal motion planning results are in Chapter 5.

The rest of the document lists the conclusions from the results (Chapter 6), and describes possible extensions to this work (Chapter 7).

1. 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.

CHAPTER 2 Previous work

Hydraulic robot modeling is a distinct sub-problem of the optimal motion planning problem. We therefore present a brief survey of previous work on hydraulic robot modeling, besides work on optimal motion planning for hydraulic and electric robots.

We also present a section on optimization techniques. The field of optimization methods is fairly large and is covered extensively by a number of textbooks. We therefore only present a brief summary of the different types of optimization methods, and refer readers to other texts for details.

Section 2.1: Hydraulic robot modeling

The model of an open-chain manipulator is a combination of its linkage and actuator dynamics. The simultaneous solution of the two sets of equations yields the complete robot response. The linkage dynamic equations for electric-drive and hydraulic robots are identical; the differences are in the actuator dynamic equations. Electric motor dynamics are commonly modeled using the linear equation:

$$J_M \ddot{q}_M + B\dot{q}_M + F_M + R\tau = K_M \upsilon \qquad \text{(EQ 1)}$$

where $q_M = vec\{q_{M_i}\}$ is the vector of rotor position angles, the control input is the motor voltage vector υ ,and F_M is the friction of the i'th rotor. The actuator coefficient matrices are constants and are given by:

$$J_M = diag\{J_{M_i}\} \quad \text{(EQ 2)}$$

$$B = diag\{B_{M_i} + (K_{b_i} K_{M_i})/R_{a_i}\} \quad \text{(EQ 3)}$$

$$R = diag\{r_i\} \quad \text{(EQ 4)}$$

$$K_M = diag\{K_{M_i}/R_{a_i}\} \quad \text{(EQ 5)}$$

where the i'th motor has inertia J_{M_i}, rotor damping constant B_{M_i}, back emf constant K_{b_i}, torque constant K_{M_i}, armature resistance R_{a_i}, and r_i is the gear ratio of the coupling from the i'th motor to the i'th link.

Unlike electric actuator response, hydraulic system response cannot be easily approximated using linear equations due to the inherently non-linear nature of fluid flow, and due to the highly non-linear characteristics of the plumbing for practical hydraulic systems. Hydraulic systems also have many more components that contribute to the complexity of the response. Even the simplest system requires a pump, valves to regulate the flow of hydraulic oil, and cylinder/motor to convert the fluid energy to shaft energy; in comparison, an electric drive system requires an electric motor and some gearing.

The fundamental equations governing hydraulic system behavior have been known for many years. More recently, thanks in part to the attention focused on hydraulic robots, there has been a resurgence in the number of papers on hydraulic system modeling. Among the most recent papers available on the subject are those by Li [Li 97], and Bilodeau [Bilodeau 97]. These authors describe their efforts in the modeling, parameter identification, and control of an electro-hydraulic mining manipulator and a single rotary hydraulic joint respectively. These implementations, although novel, simply use textbook methods for their particular applications. Other researchers with contributions to the literature are Tafazoli [Tafazoli 96] (modeling and parameter identification for a mini-excavator) and Kuntze [Kuntze 95] (modeling, and controller development for a multi-joint hydraulic robot arm).

McDonell et al. [McDonell 97] describe their work on the modeling, identification and control of a three degree-of-freedom pneumatically actuated robot arm. This robot does not suffer from any actuator interactions due to the use of a large reservoir of pressurized air. This reservoir is able to provide additional power during peak demands thus insulating the actuators from the effects of the fluctua-

tion in power demand. This approach is applicable to pneumatic systems but cannot be used for hydraulic systems due to the higher pressures needed to compress liquids.

Lawrence et al. [Lawrence 95] describe the retrofit of a hydraulic excavator, and development of coordinated and force-feedback control algorithms. Their work uses a testbed that is very similar to the excavator used in our work. However, they significantly modified the hydraulic system of their excavator to reduce the complex actuator interactions by dedicating an independent hydraulic pump for each cylinder and motor. Although this approach simplifies the control problem, we do not use it to simplify our HEX hydraulic system since all practical hydraulic machines have multiple actuators using a common pump. All hydraulic machines minimize the number of pumps since extra pumps add significantly to the cost, besides adding more complexity due to the required hydraulic plumbing. Thus, it is important to address the problem of actuator interactions, rather than avoiding the problem, may limit applicability of the research.

Some researchers have worked on improving the control of hydraulic excavators using non-linear approaches to overcome the non-linearities in the system. Work by Yao [Yao 98] and Medanic [Medanic 97] fall in this category.

In summary, all the existing work on hydraulic system modeling in the literature uses the traditional approach of writing the system of differential equations that capture the dynamics, and then solving them. This approach is computationally expensive, especially for a system like a hydraulic excavator. This might be acceptable when the models are used for control design, and linearized versions used in model-based controllers. It is however not acceptable when the model is needed to perform tens of thousands of robot simulations, as in our optimal motion planning, since the time required is prohibitively high. We describe our approach to solving the modeling problem in the following chapter.

Section 2.2: Optimal motion planning

In this section we discuss previous work on optimal motion planning. We first describe work by Rowe on optimal motion planning for hydraulic robots. The next sub-section (Section 2.2.2) discusses previous work on optimal motion planning for electric-drive manipulator robots.

2.2.1: Optimal motion planning for hydraulic robots

As mentioned in the introduction, the literature does not have any references to research on globally optimal motion planning for hydraulic robots. Rowe et al.[Rowe 99] are the only researchers to pursue optimal motion planning for hydraulic robots. They addressed the problem of performing free-space minimum-time motion planning for a hydraulic robot engaged in *mass-excavation* operation. Their

approach is to constrain the global optimization problem using motion templates (called *scripts*), which specify a sequence that the robot motions must follow. Their approach uses robot feedback to perform constrained local optimization of the motions. They used the same HEX testbed, and joint position controllers used in our work. The basic steps in their approach are listed below.

- Use human expert knowledge to construct a *script* for the given *loading configuration*. The script is defined as a sequence of robot joint motions. For example, the *script* for the *task* of moving from the dig face to the truck and back (Figure 5) could be:
  - Raise the Bm until the Bm angle is greater than *BmAngle1* (until the bucket is higher than the tail gate of the truck).
  - Next, issue a Sw joint position command (*SwAngle1*) to move the Sw clockwise (towards the truck).
  - Once the Sw joint has passed the tail gate of the truck (the truck position is known before the start of the *task*), issue a Bk joint position command (*BkAngle1*) to start opening the Bk (to unload the bucket load in the truck).
  - After the Bk is fully open (it is assumed that the bucket is empty once the bucket is fully open), issue a Sw joint command (*SwAngle2*) to move the Sw back towards the dig face.
  - Issue a Bm position command (*BmAngle2*) to move the Bm down, once the Sw is past the tail gate of the truck (to get the bucket to the soil face).

  We use a simplified version of a *script* in the above example since we only wish to illustrate the basic idea behind this approach. Rowe's scripts involve more joint coordination between the HEX joints. Notice that the *script* has adjustable parameters, shown in italics and underlined.
- Use a gradient descent to compute the set of *script* parameters that optimize the time taken for each *loading pass*. The cost of any given set of *script* parameters is evaluated by using the parameters to execute an actual *loading-pass* on the machine. The time taken for the *loading-pass* is the cost.

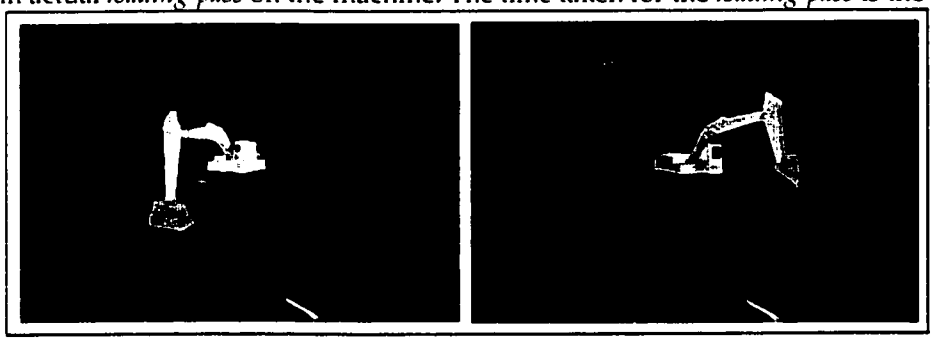


FIGURE 5. Illustration of example *task* used by Rowe (a) HEX over dig face (b) HEX over truck

Rowe's method uses heuristics to improve the gradient descent search and minimize the number of *loading-passes* that must be performed to compute the minimum-time set. The results from Rowe's method are quite impressive. The method was used to load 10 trucks (60 *loading-passes*) during one demonstration and the time taken for each loading pass for the 10th truck was very comparable to that of a human expert HEX operator. Also, since the method uses robot feedback to compute the set of best *script* parameters, it is not affected by variations in friction or wear characteristics from one HEX robot to another. However, the disadvantage of the method is that it requires the apriori construction of a *script*. There is no guarantee that the *script*, with the minimum-time set of *script* parameters, is the globally optimal motion since optimizing the *script* parameters is only a (constrained) local optimization. Also, since the optimization is perform on-line, the *script* parameter search has to be limited in order to guarantee safety at all times. The method also does not allow for the use of other objective functions, such as energy.

2.2.2: Optimal motion planning for electric-drive robots

Although optimal motion planning for hydraulic robots has not received any attention until now (aside from the above cited work by Rowe), a number of researchers have been working over the last 15 years on computationally tractable means of generating optimal controls for electric-drive robot manipulators for static and dynamic environments. This work is closely related to the problem addressed in this thesis. Only work related to static environments is referenced 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 [Fiorini 96].

The terms path and trajectory are used quite extensively in this document and they are now defined 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.

Most attempts at solving the optimal trajectory generation problem have used the purely control theoretic framework provided by Pontryagin's Extremum Principle [Kirk 70]. 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. These methods can be classified under the following headings:

- Decoupled approaches: These approaches break the overall problem into two decoupled parts. First, the 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 this category includes that by [Vukobratovic 82], [Shin 85], [Bobrow 85], and [Slotine 89].
- Coupled approaches: These approaches do not compute the geometric path as in the decoupled approaches. The optimal path computation in these methods couples the two stages - the path creation and computing the time-optimal control to follow it. A minimization method is used to vary the parameters of the spatial path. For each path, the time-optimal control to follow the path is computed, which is then used to compute the path's cost. The cost information is used by the minimization method. Work by [Rajan 87], [Shiller 91], and [Bobrow 88] 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 86] 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 last category has on-line techniques. Only the coupled approaches optimize the robot path itself, and are therefore the most general and truly optimal. The approach developed in this research also belongs to this category. The coupled approaches proposed by Rajan [Rajan 87] and Shiller [Shiller 91] 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 cubic splines, the minimum-time trajectory (for a given path) was computed using the approach by Bobrow et al. [Bobrow 85], and a gradient descent was used to vary the spline parameters in the search for the minimum time path.

The first drawback of this approach is the use of a simplistic gradient descent method. The author describes the implementation of the method for a two degree-of-freedom arm. The maximum size of his search vector is 8. The paper does not clearly describe the sensitivity of the method to changes in initial conditions for the gradient descent, i.e. "Is the final solution different if a different initial trajectory is used?". We believe that this simple approach will work for low degree-of-freedom robots, but the approach will break down very quickly as the number of joints increase[1]. This is because the cost surface becomes more complex and a simple gradient descent will no longer be able to sufficiently explore the space to yield an optimal (or near optimal) solution. Also, the search is performed in the

space of end-effector trajectories (described by cubic splines). This method is not as efficient as a command space search where, unlike the trajectory space, all points in the search space are feasible.

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 path alternatives. In the initial stages of the search the tests consisted of simple heuristics, such as ones that prefer simple paths over "jagged" paths (Figure 6).

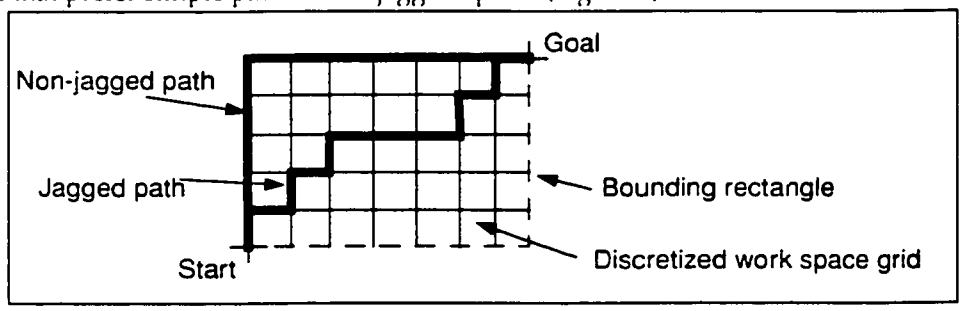


FIGURE 6. Bounding rectangle for paths examined by Shiller et al.

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 proposed by Bobrow [Bobrow 85] 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 Figure 7.

1. This observation is based on our experience with the cost function for the hydraulic robots examined in this thesis.

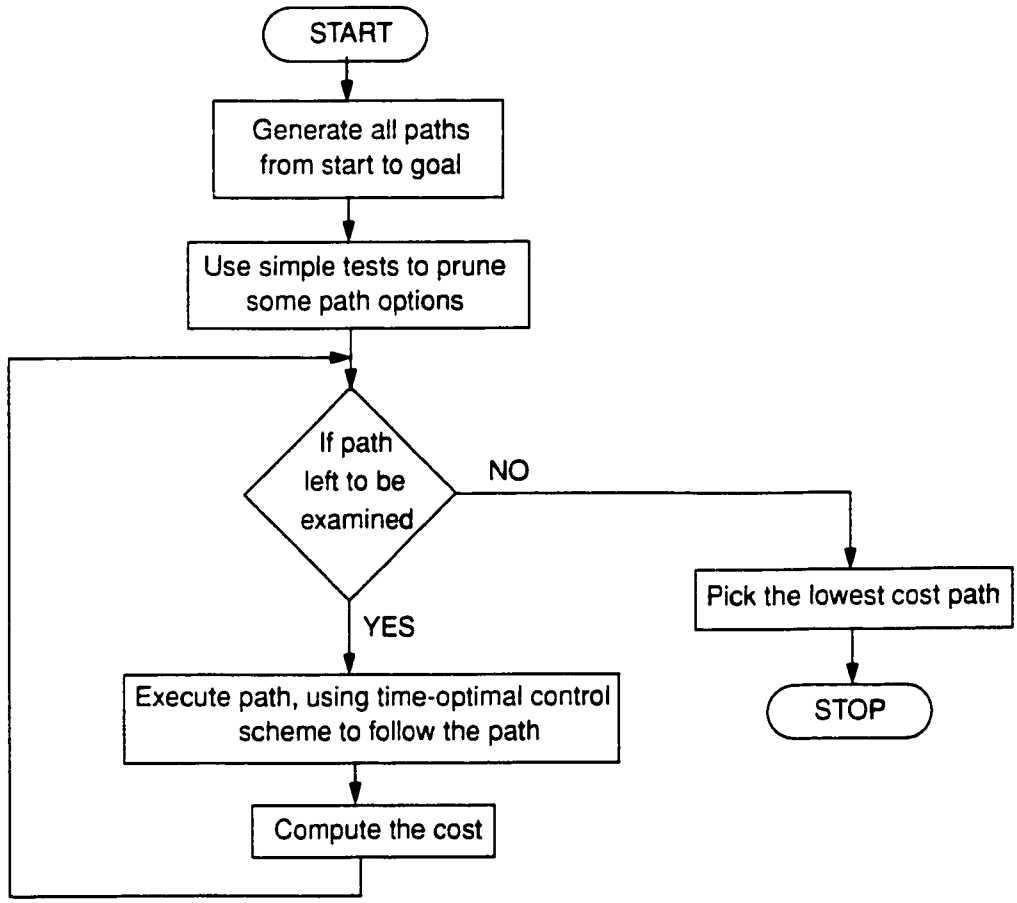


FIGURE 7. Flow-chart for scheme by Shiller and Dubowsky

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 (Figure 6). 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[2], and that the optimum path may not necessarily lie within the bounding rectangle (or cuboid in cartesian 3-space).

2. However, an exhaustive search is the only way to guarantee global optimality (at the selected discretization).

More recently, Martin ([Martin 97]) has proposed an approach to generating optimal linkage 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 does not compute the optimal controls to execute the optimal trajectory computed.

Performing a search for an optimal trajectory 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 [Kelly 98]. 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 time usage. However, since the primary focus of most research to date has been industrial manipulators driven by electric motors for which the actuator characteristics 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 use 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 (or particularly relevant) for a hydraulic robot since the limit curve is a function of many variables due to the inter-actuator coupling. For instance, due to its particular plumbing, 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. Also, unlike electric motors where a certain drive current produces a known torque, hydraulic robots do not usually control torque (or force) at the hydraulic motor (or cylinder). A particular hydraulic valve opening causes a pressure drop, which, along with the cylinder/motor force load determines the flow to the cylinder/motor.

Due to these basic differences in the nature of electric motor driven manipulator arms and hydraulic robots, previously developed optimal motion computation methods cannot be applied to hydraulic robots. We therefore need an new approach for hydraulic robots.

Section 2.3: Optimization techniques

Optimization methods came of age after the advent of computers. Prior to that there were no practical ways of solving large optimization problems. The methods derived from the work of mathematicians such as Newton were only applicable to well behaved and simple cost functions, i.e. functions with no local minima/maxima. After the advent of computers, researchers began to develop methods to optimize multi-variable functions with tens (or hundreds) of variables, which led to their application to a wide range of real-world problems.The applications for optimization methods are many and varied; from manufacturing process to machine design optimization.The problems can be divided into two broad classes: constrained and unconstrained optimization.

Constrained problems are those whose variables have bounds. Most real world problems are constrained optimization problems since nothing in the physical world can have infinite value, and is therefore bounded. However, some constrained problems can be formulated as unconstrained problems (through the use of Lagrange multipliers), which are easier to solve. The most well developed area of constrained optimization is linear-programming, where both the function to be optimized and the constraints are linear functions of the independent variables. The well known Simplex method is an approach to solving linear programming problems. The area of non-linear programming has also had some successes. However, the commonly used methods [Bazaraa 79]: quadratic programming, convex programming, quasi-Newton methods etc., rely on a particular form of the cost function (e.g. quadratic) or the constraints (convex simplex method), or require gradient information (reduced gradient methods). Most of these methods are best suited for cost surfaces without many local minima.

On the other hand, the area of unconstrained optimization has received a lot of attention and has had quite some success in the development of methods for the optimization of general linear and non-linear functions. The different methods available can be broadly classed into gradient-based and non-gradient-based methods. The gradient-base methods use the derivative of the cost function to move in the downhill[3] direction. The methods that do not require gradient information were developed for problems where it is difficult (or impossible) to get good estimates of derivatives of the cost function.

For our optimal motion planning problem, the derivative of the cost surface is an indication of how the cost of a robot motion changes as the input commands are slightly perturbed. We obtain the cost of a robot motion by using a robot model to simulate the motion. Due to the "black-box" nature of our model (it only gives us the cost of a motion), it is difficult to get good estimates of the derivative. Computing a derivative by numerical differencing does not usually yield an acceptable estimate since numerical differentiation is more prone to inaccuracies due to noisy function estimates. We therefore have to rely on non-gradient-based methods.

3. The maximization and minimization problems are dual problems; one can be turned into the other by merely changing the sign of the objective function. We therefore refer only to minimization problems for clarity.

The downhill-simplex and Powell's method [Press 96] are unconstrained optimization methods that does not use gradient information. Powell's method uses repeated one-dimensional bracketing of the minimum. It is not as robust as the downhill-simplex method by Nelder and Mead [Press 96] which is quite remarkable in its ability to move to the bottom of cost surfaces with long winded valleys. However, even the downhill-simplex method performs poorly in the presence of large local minima which trap the minimization process. Our problem has large local minima and requires an optimization method that can overcome them.

The Simulated Annealing algorithm is another optimization method that falls in the same category of non-gradient-based unconstrained optimization methods. It was first introduced by Kirkpatrick [Kirkpatrick 83] and has since shown good results on a number of NP-hard problems, most notably the travelling salesman problem. It has also been used very successfully in solving combinatorial optimization problems in VLSI design [Rutenbar 89]. This algorithm constructs a Monte-Carlo approximation of the cost surface by acquiring multiple samples, and then uses the approximation to move in the downhill direction. It is described in greater detail in Section 4.4 (Chapter 4). We use it to solve our optimal motion planning problem since it is extremely robust to local minima. Our results confirm that statement.

In summary, there are a number of unconstrained and constrained optimization methods available but only a small subset (non-gradient-based methods) are applicable to our optimal motion planning problem. Of these, the Simulated Annealing method is the most suitable given the ill-behaved nature of our cost surface.

Section 2.4: Summary

We need a hydraulic robot model for optimal motion planning since tens of thousands of robot motion simulations may be performed when computing the optimal motions for a task. To ensure that the optimal motion planning can be achieved in a reasonable length of time, it is important that the robot model be fast, while retaining the ability to model the important hydraulic system characteristics. From a look at the previous work on hydraulic/pneumatic robot modeling it is clear that existing approaches are either too slow, or involve drastic simplifying assumptions that compromise model accuracy. We therefore need a new approach to hydraulic robot modeling to satisfy our requirements of speed, while not significantly compromising accuracy. Chapter 3 describes our approach to solving this problem.

The subject of globally optimal motion planning for fluid-powered machines has been an untouched subject until now. A literature survey has yielded no previous work in this area. The previous work on optimal motion planning only addresses electric-drive robots and the existing methods cannot be adapted for hydraulic robots. Our approach to solving the optimal motion planning problem for

hydraulic robots is described in Chapter 4. From a survey of the available optimization techniques it is apparent that only a small subset (non-gradient-based unconstrained optimization methods) are applicable to our problem. Of these methods, the Simulated Annealing method is the most robust to the problem of local minima and is selected for use in our optimal motion planning approach.

CHAPTER 3

Robot Modeling

Our approach for solving the optimal motion planning problem for hydraulic robots is to use a robust search algorithm to search the space of robot commands. Thus, each point in the search space is a temporal sequence of robot commands. The search algorithm, described later in this document, navigates the search space and moves towards progressively better robot command alternatives. The evaluation of a point as being "better" or "worse" is performed by using a robot model to simulate the robot response to the command sequence corresponding to that point.

Since the search space is high-dimensional (from 10 and 100), searching it effectively requires evaluating many points in the search space, which in turn requires tens of thousands of robot simulations. In order for the search to conclude in a reasonable length of time, the robot model must be computationally inexpensive, while capturing the important robot characteristics.

One of the important characteristics we wish to capture are the significant actuator interactions seen in hydraulic robots. Traditionally, the approach to dealing with saturation effects (and the resulting actuator interactions) has been to reduce actuator speed, thus keeping the system in a linear regime of operation. While the speed reduction prevents actuator saturation and improves controllability, it also limits the peak performance of the robot. Having a model that captures power/flow saturation effects enables us to compute optimal motions that allow robot operation in a saturated condition.

At the same time, we do not need an extremely detailed since the fine detail will not augment the final result significantly. For instance, modeling the second-order effects in the response of a hydraulic spool valve does not add significantly to the fidelity of a first-order response model. It is far more important to capture important "trends" in the robot response, rather than the exact values of factors

such as friction and bucket load. The basis of this argument lies in the fact that robot models can never be perfect; they can be made quite accurate but there will always be discrepancies between the model and the real robot. The discrepancies will also vary from one robot to another due to different friction, wear, or manufacturing variations.

Given this reality, we suggest using a coarse robot model to compute the optimal motions. The coarse model captures the important "trends" but not necessarily the exact parameter values. The computed optimal motions can be adjusted during implementation of the motions[1]. To implement the computed motions on an actual robot, we suggest the use of a scheme which uses real robot feedback to fine-tune the motions. One such scheme is that proposed by Rowe et al. [Rowe 97], which improves its motions by observing the results, and performing a local optimization (We do not implement our optimal motions using this scheme since we only use one testbed to demonstrate our motions, and our model is closely tuned to the testbed)

This chapter describes our approach to constructing computationally inexpensive hydraulic robot models that capture the important "trends" in the robot's behavior, the actuator interactions being one of the most significant "trends". We demonstrate our approach by constructing a HEX model.

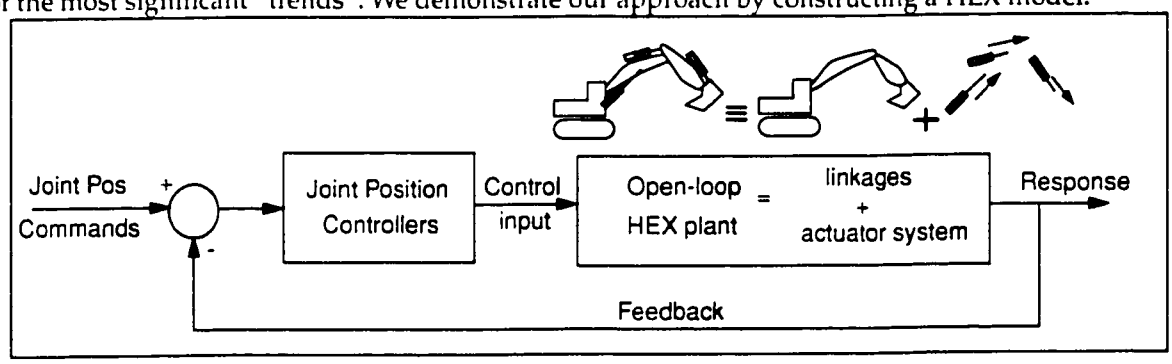


FIGURE 8. Description of the HEX robot as a combination of controllers, and open-loop plant (linkages + actuators).

The HEX robot (shown in Figure 8) consists of two main parts: the joint controllers and the open-loop plant. The joint controllers used are PD (proportional derivative) controllers. As shown in the figure, the inputs to our robot are joint position commands. The controller uses the error signal (difference between the commanded and current joint positions) to compute the control input to the open-loop

1. We thus make the implicit assumption that the optimal motion solution for the approximate plant is close to the solution for the true plant. If the model is perfect, the two solutions will be the same.

plant. The open-loop plant consists of two distinct parts: the linkages, and the actuator system. We refer to the entire hydraulic system (and engine) as being part of the actuator system.

In this chapter we describe the modeling of the three parts of the robot: the linkages, the actuator system, and the joint controllers, in that order. The challenge in the modeling lies in constructing fast actuator models that capture the important actuator interactions in the different HEX operating modes. We construct actuator models for two different HEX operating modes: the *fine-finish*(*FF*) and the *truck-loading*(*TL*) mode. The linkage and controller models are easy to construct using textbook methods.

We first describe the construction of the open-loop HEX plant model. Section 3.1 gives a brief description of the equations involved in the detailed modeling, and also describes how the equations can be used to construct a complete open-loop HEX model (linkage and actuator dynamics). This approach, which uses a numerical solver, is the most common approach used in the literature. The next section, Section 3.2, details our approach to constructing fast HEX actuator models for the *FF* and *TL* operating modes. We also describe the construction of a complete HEX model using our fast actuator model. Section 3.4 shows some comparisons of HEX motion simulations performed using the open-loop plant model, to real testbed motion data.

The next section, Section 3.6, describes the modeling of the joint controllers. The open-loop plant model, in conjunction with the joint controller model, can be used to construct a model of the closed-loop HEX robot. We do not show comparisons of motion simulations performed using the closed-loop HEX robot model, to real testbed data, since that comparison is demonstrated by the optimal motion planning results in Chapter 5. We complete the chapter with some concluding remarks.

Section 3.1: Description of the detailed HEX robot model

The model of a serial chain manipulator, such as the HEX, can be viewed as a combination of two parts:

- linkage dynamics and,
- actuator dynamics.

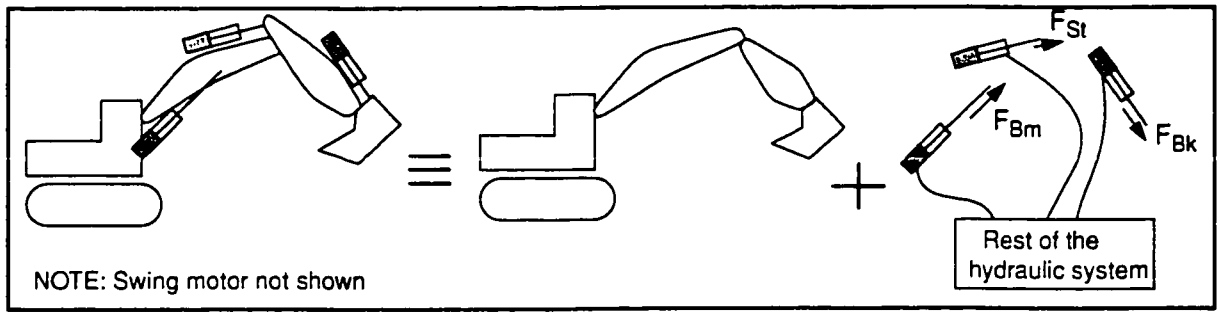


FIGURE 9. HEX robot viewed as a combination of linkages and actuators.

The simultaneous solution of the equations describing the linkage and actuator dynamics yields the overall robot response. The linkage and actuator HEX models are described below.

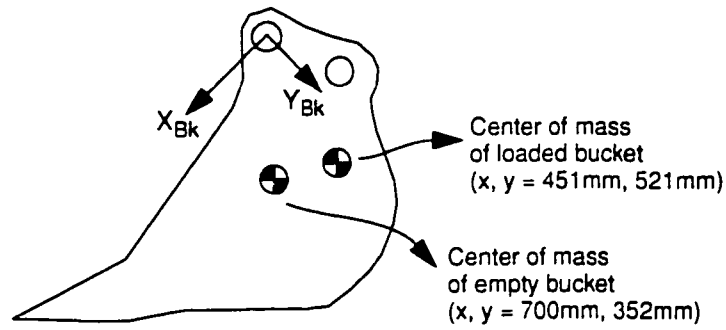
3.1.1 : Detailed Linkage Model

The linkage dynamics for a serial chain manipulator like the HEX can be described using the well known Newton-Euler equations [Craig 89]. The general form of the equations, as commonly seen in the literature, is:

$$M(\Theta)\ddot{\Theta} + V(\Theta, \dot{\Theta}) + G(\Theta) = \Upsilon_{actuator} + \Upsilon_{external} + \Upsilon_{friction} \qquad \text{(EQ 6)}$$

where M is the inertia matrix, Θ is the vector of joint angles, V is the vector of the non-linear coriolis and centripetal terms, G is the vector of gravity terms, $\Upsilon_{actuator}$ is the vector of torques applied on the linkages by the hydraulic actuators, $\Upsilon_{external}$ is the vector of torques applied on the linkages due to interaction of the linkages with the environment, and $\Upsilon_{friction}$ is the vector of friction torques. In this work we do not consider any soil-tool interaction and hence the associated torques, $\Upsilon_{external}$, are zero.

We also set $\Upsilon_{friction}$ to zero since, for our purposes, it is negligibly small in comparison to $\Upsilon_{actuator}$. The validity of this assumption is borne out by the results described later in this chapter.



Note: X_{Bk} and Y_{Bk} are the bucket coordinate system axes (Ref Appendix A for details)

FIGURE 10. Centers of mass for loaded and empty buckets

Although we do not model digging, we do model HEX operation with a loaded bucket. The load in the bucket is modeled by treating the bucket-soil combination as a lumped system of masses, i.e. we combine the masses and find the new center of mass of the lumped system. The mass distribution of the bucket i.e. the moments of inertia, are assumed to not change after it has been loaded. For a typical loaded bucket the center of mass is shifted as shown in Figure 10. The amount of this shift was determined empirically. This assumption is valid for tasks where the bucket is filled with soil or gravel, as it is in our case. However, it may not be as valid if (for instance) the end-effector was carrying a tree-trunk. In such a case, the tree would significantly alter the inertia of the end-effector, and would require more detailed modeling.

The bucket soil load was determined by recording the hydraulic cylinder pressures when the HEX was stationary, and using them in a static force analysis.The details of the terms in Equation 6 - the inertia matrix terms, the gravity terms and the non-linear terms - are given in Appendix A at the end of this document.

3.1.2 : Detailed actuator model

The actuator system for a hydraulic robot such as a HEX consists of the entire apparatus responsible for generating the drive at the cylinders/motor. This includes the engine which supplies the power, the pumps that use engine shaft power to convert low pressure hydraulic oil to high pressure, the hydraulic valves that regulate flow to the different cylinders/motor, the fluid volumes (such as the hoses), the solenoids that move the hydraulic *spool* valves, the *spool* valves themselves, and check

valves that prevent flow from moving in undesirable directions. Figure 11 illustrates how a *spool* valve controls the hydraulic flow to a cylinder. On our HEX testbed, the position of the hydraulic *spool* is used as the control input to the open-loop HEX plant (See Figure 8).

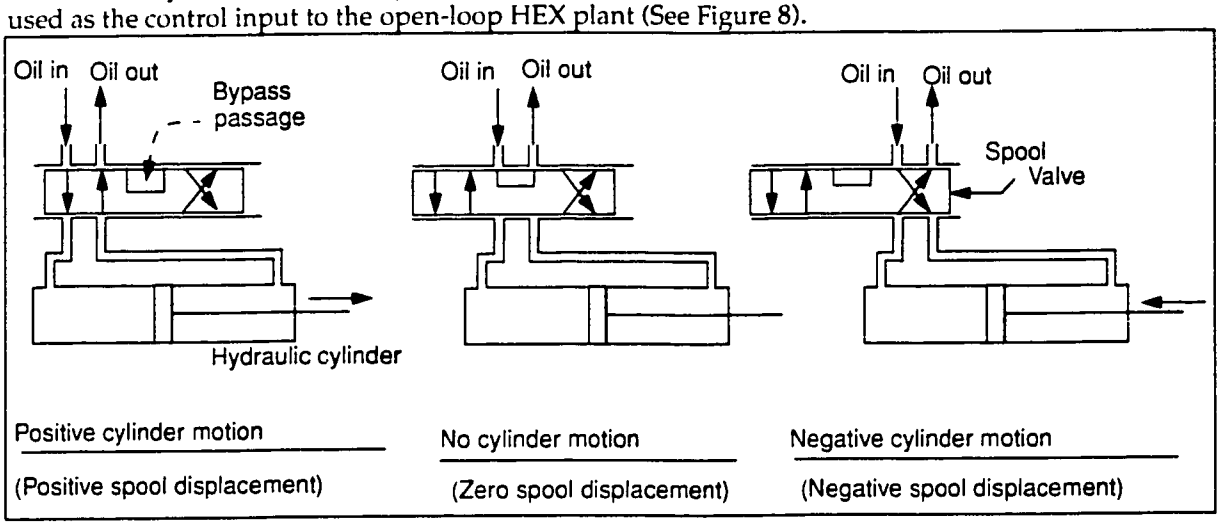


FIGURE 11. Illustration of the operation of a *spool* valve

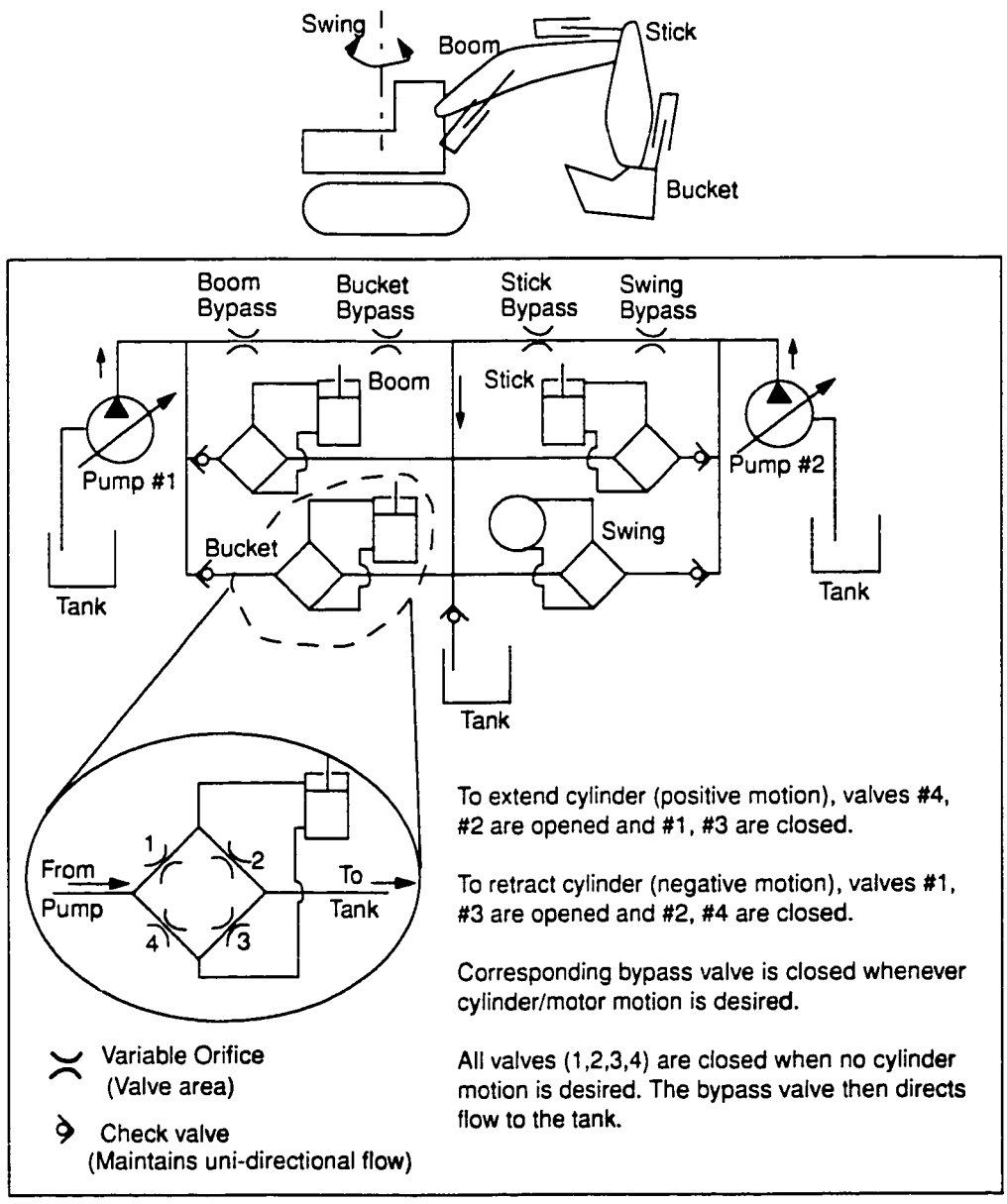


FIGURE 12. HEX hydraulic schematic (*FF* Mode)

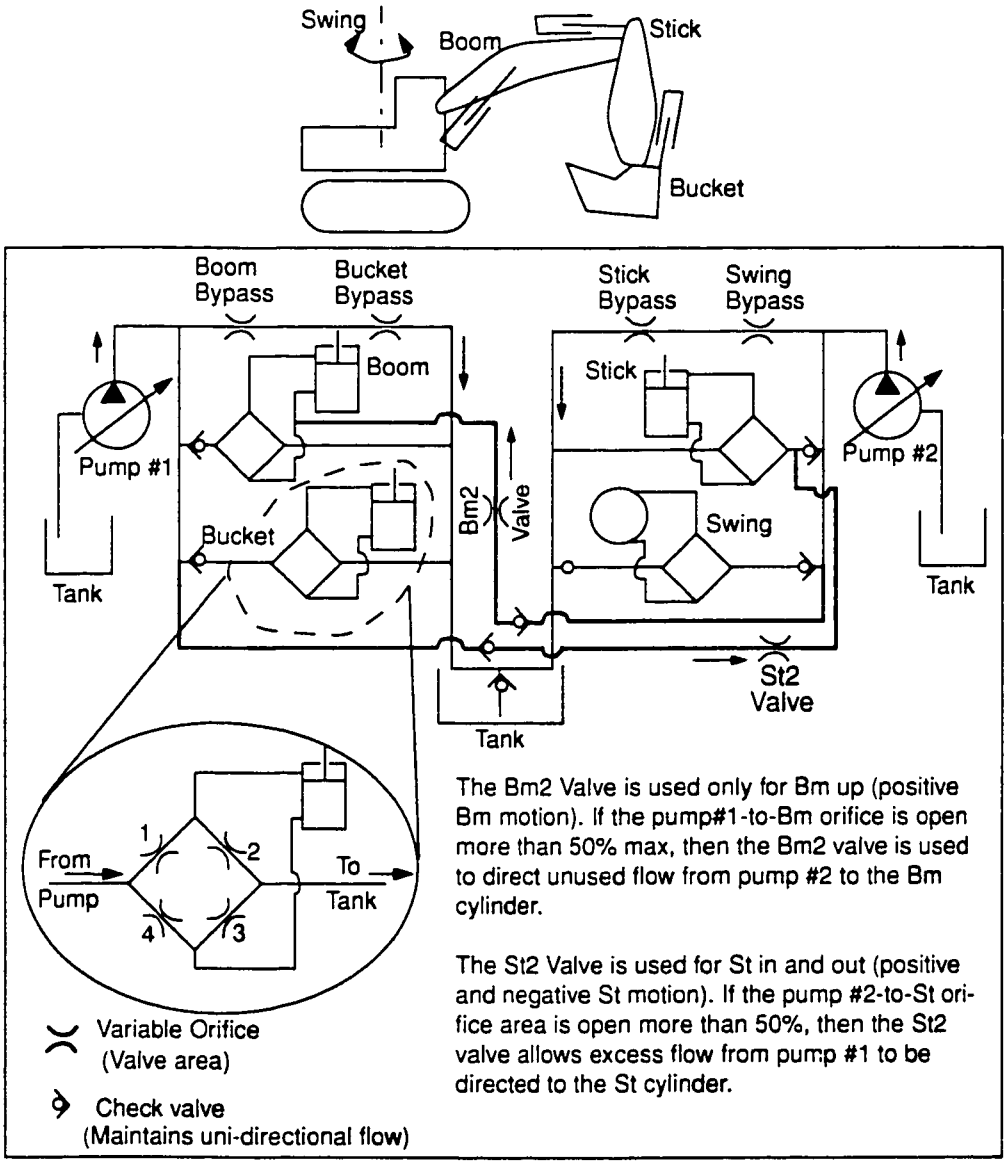


FIGURE 13. HEX hydraulic schematic (*TL* Mode)

Figure 12 and Figure 13 show schematics of the HEX hydraulic systems in the *FF* and *TL* modes. In these figures, the engine that drives the two pumps is not shown. Figure 12 describes the bridge network of valves used for each cylinder/motor, and shows the valves that are opened when extending or retracting a cylinder/motor. The schematic shown in Figure 12 is for the *fine-finish* (*FF*) mode; the name indicates that the mode is used to perform finishing operations using the HEX implement. The flow routing for the more complex *truck-loading* (*TL*) operating mode is shown in Figure 13. This mode is used when the HEX is used in a mass excavation scenario.

As shown in Figure 12 and Figure 13, the hydraulic system is driven by two hydraulic pumps which take low-pressure hydraulic oil from a tank (at slightly above 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 primarily supplies two implement circuits[1], i.e. one pump supplies the boom and bucket cylinders while the other supplies the stick cylinder and swing motor. In the *TL* operating mode, unused flow from pump #1 is used to drive the stick while unused flow from pump #2 is used to supply the boom. This *cross-over* flow is achieved through two valves indicated as the Bm2 and St2 valves in Figure 13. The *cross-over* arrangement proves most advantageous when the boom and stick are not moved simultaneously; the advantage is realized in the form of an almost 100% increase in boom and stick cylinder velocities. This flow routing allows human operators to perform truck loading operations most effectively.

The flow from the pumps to the cylinders is controlled through variable orifices. If an orifice is completely closed no flow is supplied to that cylinder and no motion results. The Caterpillar 325A HEX hydraulic system 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. 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. An open-center hydraulic system offers a responsive system since the pump does not have to stroke up from zero output when flow is demanded after a pause in operation. The drawback of such a system is the wasted power when the implements are not being used. When the implements are idle, the pumps are still supplying flow which is sent through the bypass valves to the tank.

1. 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.

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:

$$Q = C_d A \sqrt{\Delta P} \qquad \text{(EQ 7)}$$

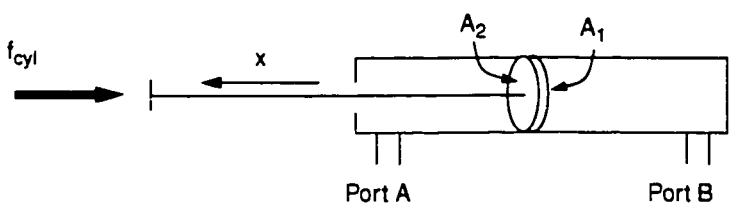
where Q is the flow rate through an orifice, C_d is the orifice coefficient of discharge (a constant), A is the orifice area, and ΔP is the pressure drop across the orifice. The concepts of flow and pressure drop are analogous to current and voltage drop in electrical circuits. Thus, an orifice can be viewed as a non-linear resistor. The different orifice areas in Figure 12 are controlled by the position of a control *spool*, as shown in Figure 11. 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. In general the valve orifice area A is a non-linear function of the *spool* 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:

$$\dot{P} = -\frac{\beta Q}{V} \qquad \text{(EQ 8)}$$

where P is the pressure in a control volume, β is the bulk modulus[1] of the oil, V is the volume of oil in the control volume, and Q is the flow rate through the control volume.

1. The bulk modulus of the oil is a function of temperature. The oil temperature can vary from -20 to +100 deg F during operation. A rise in temperature from -20 to +20 deg F can cause a significant change in the response characteristic of the hydraulic system due to changing bulk modulus and oil viscosity. However, the change in response is quite small for temperatures between 20 and 100 deg F. In our analysis we assume fixed oil properties, which is a good assumption for oil temperatures between 20 and 100 deg F.



NOTE: To retract the cylinder high pressure oil enters through port A or B while low pressure oil exits the other.

FIGURE 14. Schematic of bi-directional hydraulic cylinder

The force balance equation for a cylinder (Bm, St, or Bk) is:

$$m\ddot{x} = P_1A_1 - P_2A_2 - f_{cyl} - f_{friction} \qquad \textbf{(EQ 9)}$$

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).

Each cylinder extension x_{Bm}, x_{St}, x_{Bk} is mapped to the joint position $\theta_{Bm}, \theta_{St}, \theta_{Bk}$ via a non-linear function:

$$x_i = C_i(\theta_i) \qquad \textbf{(EQ 10)}$$

The linkage force f_{cyl} appears in the excavator linkage dynamic equations (Equation 6) where each joint torque τ_i in the torque vector Υ is related to the corresponding cylinder force via the transform:

$$\tau_i = f_{cyl_i} \cdot \frac{\partial}{\partial \theta_i} C_i(\theta_i) + \tau_{diggingForce} \qquad \textbf{(EQ 11)}$$

In our work we have not considered soil-tool interactions and hence the digging forces are zero. The detailed expressions for the non-linear function C_{cyl} in Equation 10, and $\frac{\partial}{\partial \theta_i} C_i(\theta_i)$ in Equation 11 are given in Appendix A.

Thus, the solution of the response of even the simplified hydraulic system in Figure 12 involves the simultaneous solution of multiple orifice equations (Equation 7), multiple compressibility equations for all the oil volumes (Equation 8), and force balance equations for each cylinder. A steady state solution would not include the fluid dynamics in Equation 8. The HEX hydraulic system also has other non-linear components such as check-valves (shown in Figure 12) 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.

3.1.3 : Construction of the complete detailed robot model

The complete HEX response can be determined by the simultaneous solution of the linkage and actuator dynamics for a given set of inputs to the HEX. A simulation of the robot response to any set of inputs is thus possible by progressive integration of the results using the Runge-Kutta [Press 96] or similar method. At each time step, starting from t=0, the initial conditions for that time step and input commands are used to simultaneously solve the linkage and actuator dynamics. The result is integrated numerically and used to provide the initial conditions for the next time step.

A complete model of the excavator which includes linkage and actuator dynamics is a coupled eighth-order non-linear system of a few hundred equations, and is partially described in [Medanic 97]. 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 HEX testbed. This detailed model takes 100-150 secs to simulate 1 sec. of a typical excavation cycle when running on a SUN Sparc20 workstation. Even a simplified steady state version of the above model takes 1 second to simulate 2 secs. of a typical HEX motion.

Section 3.2: Simplified Actuator Model

From the above discussion it is quite obvious that a traditional approach to modeling the HEX hydraulic system is too slow for use in optimal motion computation. A typical optimal motion search may require performing tens of thousands of robot motion simulations, which would take months or a year to compute using the aforementioned model. We thus require a HEX actuator model that is at least an order of magnitude faster (computationally inexpensive) than the above cited approach.

We construct a fast actuator model by using a function approximation method to approximate the non-linear actuator characteristics and interactions. We describe below the form of the actuator response functions, and their approximation using non-linear function approximators.

As mentioned before, the different orifice areas in Figure 12 are controlled by the position of a control *spool* (Figure 11). The *spool* position can thus represent all the orifice area variables for a joint.

From Figure 12 it can be seen that the steady state flow available to any cylinder/motor is a function of the orifice area of the relevant valves, and the ratio of force loads between the cylinders/motors competing for flow. For instance, if the boom inlet orifice area were zero, no flow could be supplied to it. To supply flow to the boom cylinder, the boom *spool* is shifted and the area opened. This allows high-pressure oil to flow from the pump to the cylinder. If, in addition, the bucket *spool* were to be fully shifted, the bucket cylinder will "steal" part of the boom's flow since the bucket cylinder is lightly loaded as compared to the boom cylinder. This is true if the total pump flow is inadequate to supply both cylinders at the same time - if the bucket or boom orifice areas are only partially open, the distribution of flows between the cylinders will be different.

This effect is clearly shown in Figure 15. The x-axis of that figure shows the boom *spool* position (0 to 11mm), the y-axis shows the bucket *spool* position (0 to 11 mm), and the z-axis shows the boom cylinder velocity (mm/sec). The boom cylinder velocity is an indicator of the amount of flow received by the boom cylinder. The *spool* positions indicate the orifice opening from the pump to the appropriate cylinder. 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.

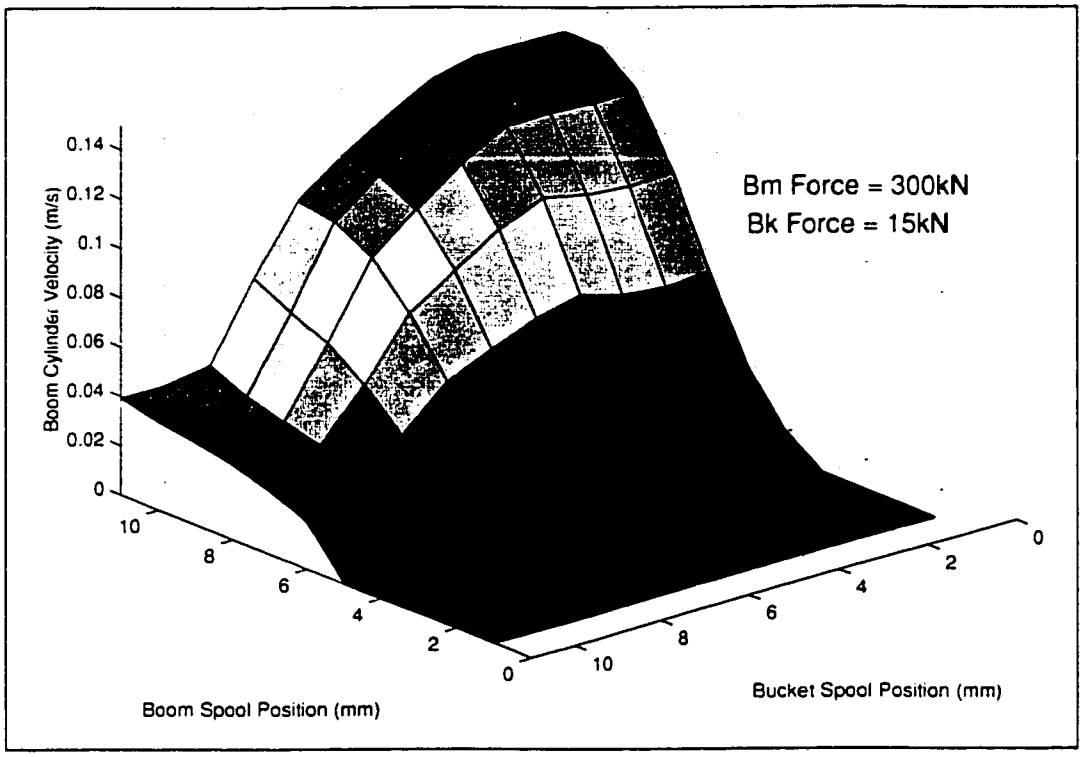


FIGURE 15. Section of the Bm response surface (*FF* Mode)

The boom cylinder velocity at maximum boom *spool* position (Bm *Spool* Command = 11mm) is seen to drop sharply as the bucket *spool* position increases from 0 to 11mm. 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.

Our approach to modeling these interactions is to use a non-linear function approximator to approximate this functional mapping from the space of inputs - the *spool* positions (which determine the different orifice areas) and cylinder forces, to the outputs - the flow to each cylinder/motor, which in turn determines its steady-state velocity. We settle for a steady-state actuator model since the transient response is not significant for cylinder motions that are much longer than the cylinder's time constant, and we need the model for just such a purpose, i.e. simulating a few seconds of machine motion. For instance, the worst cylinder response transients are seen in the Bm joint, and these only last for 1-1.5

secs (Ref Figure 18). We thus avoid the extra expense of modeling the actuator transients. The transient response is a function of more variables than those used for a steady-state model (the *spool* positions and the cylinder forces). The validity of this assumption that the effect of the transient response is not significant when considering a few seconds of machine motion, is borne out by the results shown in Section 3.4.

The next two subsections describe the use of different function approximation methods for modeling the *fine-finish(FF)* and *truck-loading(TL)* operating modes.

3.2.1 : Modeling the FF HEX operating mode

In our first attempt at using a function approximator to construct the actuator model we used a locally weighted linear regression technique (also referred to as a Memory-based Learning (MBL) technique in the literature) to model the *fine-finish* (*FF*) HEX operating mode.

The memory-based learning was performed using a software tool - *Vizier* - developed by Schneider et al. [Schneider 96] [Atkeson 97] 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.

We used *Vizier* in conjunction with a lookup scheme to speed the predictions. The basic idea behind this is to store the prediction information from all the previous queries so that new queries can possibly use multi-linear interpolation to compute the result, instead of computing the value from the data set (which requires matrix inversion). This is implemented as follows:

1. Construct a regularly gridded table with the same dimensions as that of the function being approximated. For instance, the Bm response function has two *spool* positions (Bm and Bk) and two forces (Bm and Bk cylinder forces) as the dimensions.
2. When a query is made, check to see if the grid points of the cell that the query lies in have a prediction value associated with them. At the start, the grid points will not have any value associated with them.

   If the grid points surrounding the query do not have a value, then use the standard prediction procedure to compute their values (from the data set).

   Once all the surrounding grid points have values, use multi-linear interpolation to obtain the prediction value for the query.

This approach makes the initial set of predictions take longer since multiple predictions are made for each query (to compute the values for the neighboring grid points), and each such query requires

matrix inversions. As the process continues, and the table becomes populated, the predictions require less time as they can more frequently use the multi-linear interpolation.

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 inputs for each MBL table were chosen after studying the flow routing schematics for the HEX *FF* operating mode. The inputs and outputs for each table are listed in Table 1. The swing is different from the other joints since it is a rotational joint with a large inertia; the inertia of the swing is a function of the positions of the linkages and the load in the bucket. For instance, the swing inertia is very different for a fully extended and fully retracted stick cylinder (corresponding to the stick being tucked in and being stretched out respectively). Due to the larger inertia, the swing has long acceleration and deceleration phases, unlike the three joints driven by linear cylinders. This requires that the input dimensions be slightly different from that for the other three joints.

	Bm	**St**	**Bk**	**Sw**
Input #1	Bm Spool Position	Sw Spool Position	Bm Spool Position	Sw Spool Position
Input #2	Bk Spool Position	St Spool Position	Bk Spool Position	St Spool Position
Input #3	Bm Cyl. Force	Sw Inertia	Bm Cyl. Force	Sw Inertia
Input #4	Bk Cyl. Force	St Cyl. Force	Bk Cyl. Force	St Cyl. Force
Input #5	-	-	-	Sw Velocity
Output	Bm Cyl. Velocity	St Cyl. Velocity	Bk Cyl. Velocity	Sw Acceleration

TABLE 1: Inputs/Output for HEX actuator MBL tables (*FF* Mode)

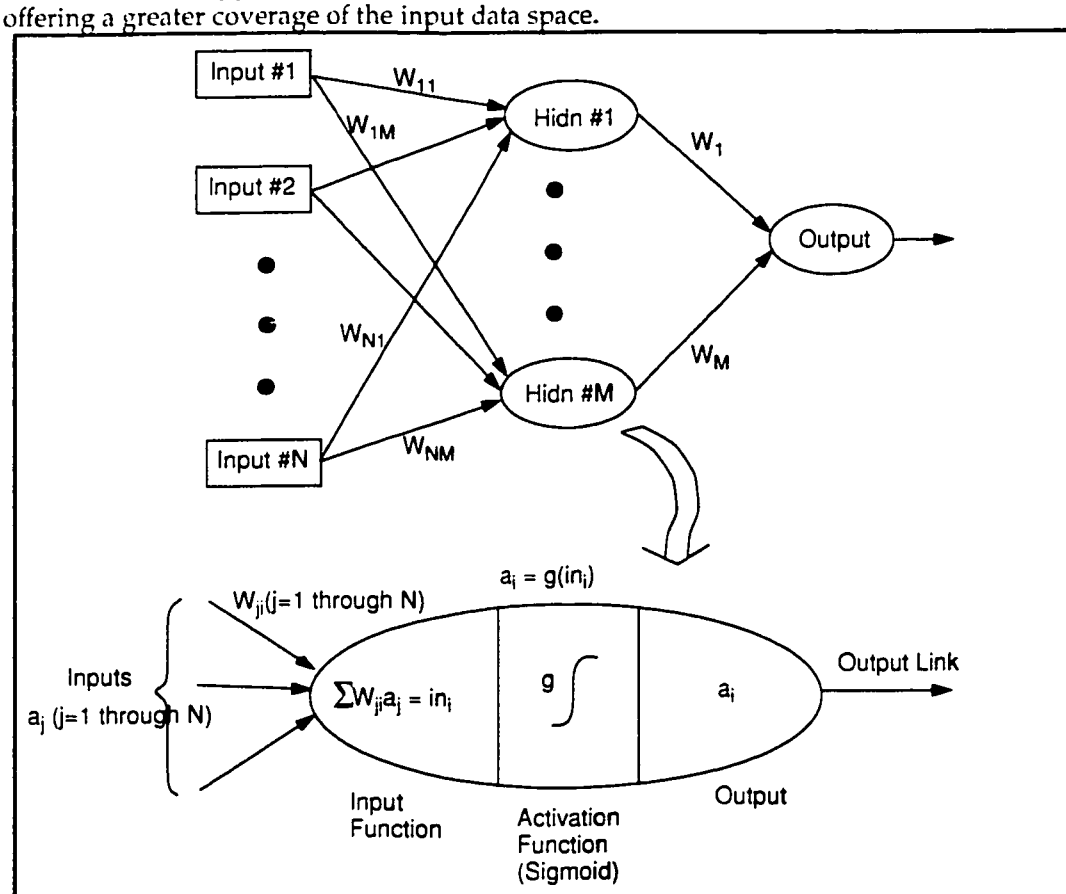
The following sub-section, Section 3.2.2, describes the model construction for the *TL* operating mode. The results comparing the predictions of the *FF* and *TL* mode models are given in the next section (Section 3.4).

3.2.2 : Modeling the TL HEX operating mode

During the next attempt, when modeling the truck-loading (*TL*) mode, we used neural networks for the function approximation. This was necessitated by the fact that the actuator response surfaces in the *TL* mode are more complex and have a larger number of input units. The MBL method, although providing comparable results, has slower prediction times when compared to a neural network. In addition, the prediction times increase with the number of inputs.

However, neural networks do not offer much control over the approximation created. For instance, it is possible that the neural network may use an erroneous approximation in a region of the input space where no data is available. It is not possible to force the network to use a different (for instance,

smoother) approximation in that region alone. This is not the case with the MBL method which offers more control over the approximating function. We alleviate this problem by using more training data, thus offering a greater coverage of the input data space.



NOTE: 1. W_{ij} is the weight associated with the connection between input i and hidden unit j.
2. W_j is the weight for the connection between hidden unit j and the output unit.
3. The structure of the output unit is identical to that of the hidden units. In the above detail of the hidden unit, replacing W_{ij}(j=1 through N) by W_j(j=1 through M), and the inputs by a_k(k=1 through M) yields the output unit.

FIGURE 16. A typical 3-layer feed-forward neural network used for approximating an actuator response surface (*TL* Mode); Details of a single hidden (i'th) unit are shown in the lower figure.

For the *TL* mode, a total of four neural networks are used - one for each joint of the HEX. Each network has many inputs and one output, and one hidden layer. The networks are the commonly used feed-forward networks (Figure 16 [Russell 95]). We used the backpropagation algorithm of Bryson and Ho [Bryson 69] to train the networks. The outputs of the boom, stick and bucket networks are steady-state cylinder velocity values, while the swing output is swing acceleration. The network inputs were chosen after studying the flow routing schematics for the HEX *TL* operating mode. Table 2 lists the inputs, outputs and number of hidden units for the networks used to model the *TL* operating mode.

Note that the inputs for the Bm response function do not include the St Force and Sw Inertia. This is because, based on analysis of the training data, we found the effect of those inputs on the Bm response was negligible. The same is the reason for not including St Force and Sw Inertia in the list of Bk function inputs, not including the Bm and Bk Forces in the St and Sw list of inputs.

	Bm	**St**	**Bk**	**Sw**
Input #1	Bm Spool Position	Bm Spool Position	Bm Spool Position	Bm Spool Position
Input #2	St Spool Position	St Spool Position	St Spool Position	St Spool Position
Input #3	Bk Spool Position	Bk Spool Position	Bk Spool Position	Bk Spool Position
Input #4	Sw Spool Position	Sw Spool Position	Sw Spool Position	Sw Spool Position
Input #5	Bm Cyl. Force	St Cyl. Force	Bk Cyl. Force	Sw Inertia
Input #6	Bk Cyl. Force	Sw Inertia	Bm Cyl. Force	St Cyl. Force
Input #7	-	-	-	Sw Velocity
Output	Bm Cyl. Velocity	St Cyl. Velocity	Bk Cyl. Velocity	Sw Acceleration
Num. hidden units	9	9	9	9

TABLE 2: Inputs/Output for HEX actuator model networks (*TL* Mode)

3.2.3 : Data collection

Training data for the actuator models was collected using a slow but complete analytical machine model[1]. The slow model was driven through a number of motion sequences to adequately cover the operating space of each network. 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 com-

1. The testbed was not used to collect training data due to our inability to measure *spool* positions directly on it.

pletely 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 1 mm along the *spool* position axes. The cylinder forces were determined by the excavator's configuration.

3.2.4 : Modeling actuator delays

The HEX actuators exhibit a certain delay between the issuing of the command and the actuator response. There are two main reasons for HEX actuator delays. One is due to the fact that a smaller hydraulic system (*pilot* system) is used to move the larger main *spool* valves. This results in a delay between when the *spool* valve motion is initiated by activating the pilot system, and when the main *spool* valve actually shifts. The second cause of the delay is due to the finite pump response, and the time it takes to fill a cylinder chamber. For instance, consider the case where the head end (Figure 17) of a hydraulic cylinder is pressurized (connected to the pump) and the rod end is connected to the tank (low pressure). If the control *spool valve* were to be suddenly closed, the high pressure oil in the head end will not have time to leave, and hence will be trapped in that chamber. The rod end chamber will be at low pressure for the same reason. If the cylinder were to now be moved in a negative direction (connect the rod end to the high pressure pump), it will take some time for the rod-end chamber to fill with enough pressurized fluid so that motion can be initiated. This delay is not present if the cylinder is moved in a positive direction (instead of negative), since the head end is already at high pressure.

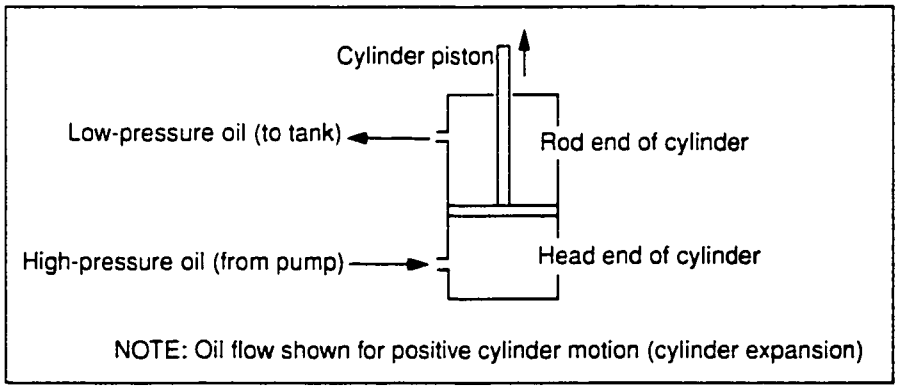


FIGURE 17. Illustration of the two chambers of a hydraulic cylinder.

Our model only captures the former delay since it is easy to model as a first-order lag. The latter cause of the delay is not modeled since it is dependent on the state of the pump and cylinder chambers when motion is initiated. Our steady-state actuator model cannot capture such effects.

3.2.5 : Implementation detail

While training the networks (for the *TL mode* model) it was found that dividing the input space was desirable to allow the networks to better approximate the functions. Hence, each joint's input space was divided into 8 non-overlapping sections resulting in a total of 4*8=32 networks. The 8 represents the number of permutations possible with regard to the direction of motion of the Bm, St, Bk, and Sw *spool* valves[1]. The Bm, St, and Bk *spool* valves can move in or out, resulting in 2^3 permutations, and hence 8 sub-spaces; the Sw is symmetric and hence positive and negative *spool* motions (clockwise and counter-clockwise swing motions) are identical. The network for each sub-space was then trained using the data points in that sub-space.

The MBL approach used to model the *FF* HEX operating mode did not require any such division of the data space since it was able to approximate the response functions fairly well. This was most likely due to the lower number of input dimensions for each joint response function in the *FF* case.

3.2.6 : Fudge factors

No method can be complete without its share of fudge factors. In our actuator models it was observed that in parts of the operating space, the boom and stick cylinder velocities predicted using the model were slightly higher than the values obtained from the detailed analytical model (and the testbed). This was attributed to the fact that *flow forces* might not be allowing the hydraulic *spools* to move to the desired displacement. *Flow forces* are forces applied to the hydraulic *spool* valves by the fluid whose flow they control. In our testbed, solenoid actuators move the hydraulic *spool* valves in an open-loop manner. In response to a certain input current, they attempt to move the *spool* to a desired position. However *flow forces* prevent the *spool* from displacing to the desired extent. The problem is therefore in the prediction of the *spool* displacement in response to an input current, and not in the prediction of the cylinder velocities. To address the problem we decrease the boom and stick *spool* displacements by an empirically determined factor (0.95 for the boom and 0.9 for the stick) in some portions of the boom and stick operating space.

1. Recall that each joint actuator's response surface is a function of the four *spool* valve positions, in the *TL* mode.

Section 3.3: Simplified HEX model (Using simplified actuator model)

The complete HEX model is constructed by partitioning the actuator dynamics and linkage dynamics into two separate problems. Instead of solving them simultaneously they are solved in a serial fashion. First, the linkage dynamic model (Equation 6) 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 approximated 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. We use a fixed time step of 0.02 secs for the above progressive integration. The steps are spelled out explicitly below.

- Step #1: The linkage 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 compute the resulting cylinder velocities using the corresponding neural networks or MBL tables. The swing table uses the current swing velocity to compute the swing acceleration.

  Note that the input *spool* commands are the control inputs for the open-loop plant model (Figure 8). They may be supplied by the position controller (when operating the HEX robot in a closed-loop mode), or by a human operator of the HEX via control joysticks (in a manual operation mode).
- Step #3: The computed velocities (or accelerations) are integrated to obtain an updated excavator state. (Repeat steps 1 through 3).

Section 3.4: Robot model results

We present some results to demonstrate the effectiveness of our modeling approach. We have conducted many tests to explore different combinations of Bm, St, and Bk cylinder directions for the two HEX operating modes - the *fine-finish* and *truck-loading* modes. The following two sections - Section 3.4.1 and Section 3.4.2, show the results from four multi-joint tests (two for each operating mode) along with an explanation of the results. The results from all the other tests are presented in Appendix B. In all the tests that follow, the HEX was used in open-loop mode, i.e. the hydraulic *spool* valves were used to actuate the different joints and no controllers were used (Refer Figure 8). Hence, the "commands" shown in all the plots below refer to the hydraulic *spool* position, as a percentage of maximum *spool* valve displacement. Controlling the orifice areas allows control of the flow to the hydraulic cylinders/motor, and hence their velocity.

As mentioned in our problem statement (Section 1.1), we only address free-space motion of the HEX robot. Hence, no digging was involved in any of the tests and therefore no digging forces were applied at the bucket tip.

3.4.1 : Fine-finish mode tests (Used MBL function approximators)

In the first test (Figure 18, Figure 19) the boom and bucket joints are actuated simultaneously, and the swing and stick joints are actuated together in the second test (Figure 20, Figure 21). These tests show the extent of the actuator interaction. In the *FF* mode of operation one hydraulic pump supplies hydraulic oil to the swing and stick joints while the other pump supplies oil to the boom and bucket joints. Thus, the Bm-Bk joints are closely coupled, as are the St-Sw joint pair; hence our choice of the following tests involving combinations of Bm and Bk joints, or St and Sw joints. The interaction between the Bm-Bk and the St-Sw sets of joints is very small; the only coupling between them is through the engine.

In the first test the Bm and Bk joints are simultaneously commanded to move (see the last subplot in the figures) but due to the actuator interaction the heavier Bm does not get very much flow until the bucket has stopped moving (at approx. t = 5 secs). The model is able to capture this interaction quite well. Note that the boom velocity oscillation[1] in Figure 18 is not captured by the steady-state model. However, the effect of not modeling the oscillation is quite small, as seen in the Bm position plot in Figure 19.

1. The Bm oscillation is due to the compressibility of the hydraulic oil in the cylinder. The oscillation is most significant for the Bm since the cylinder supports the linkage from below, like a spring supporting a mass on it.

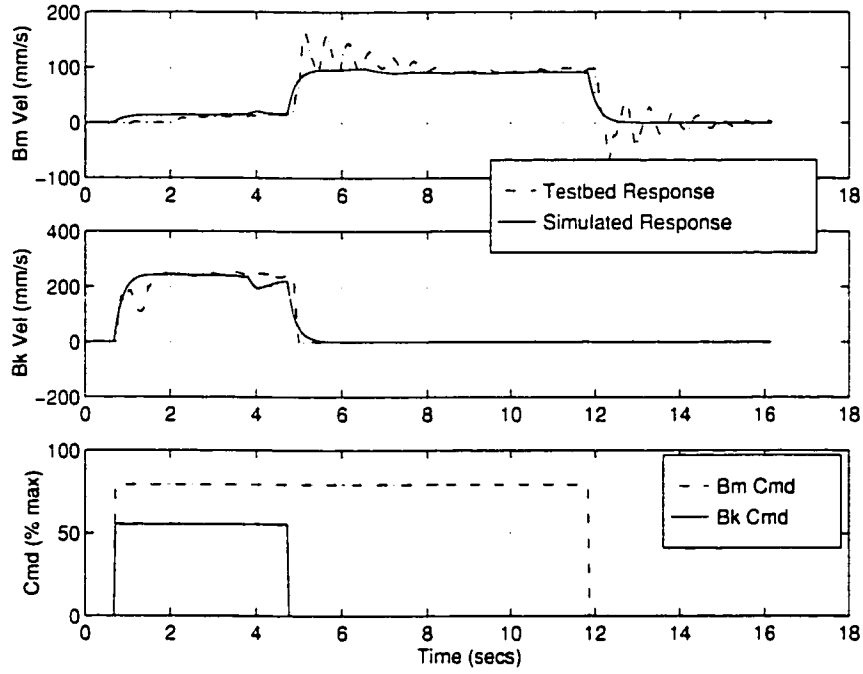


FIGURE 18. Bm, Bk Cylinder Velocity Plots (Test #1 - *FF* Mode)

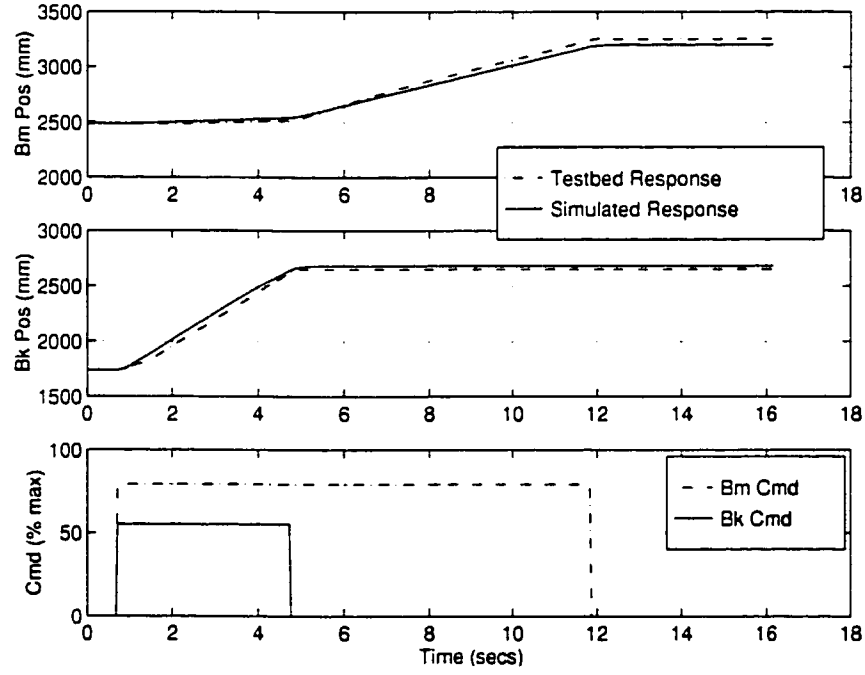


FIGURE 19. Bm, Bk Cylinder Position Plots (Test #1 - *FF* Mode)

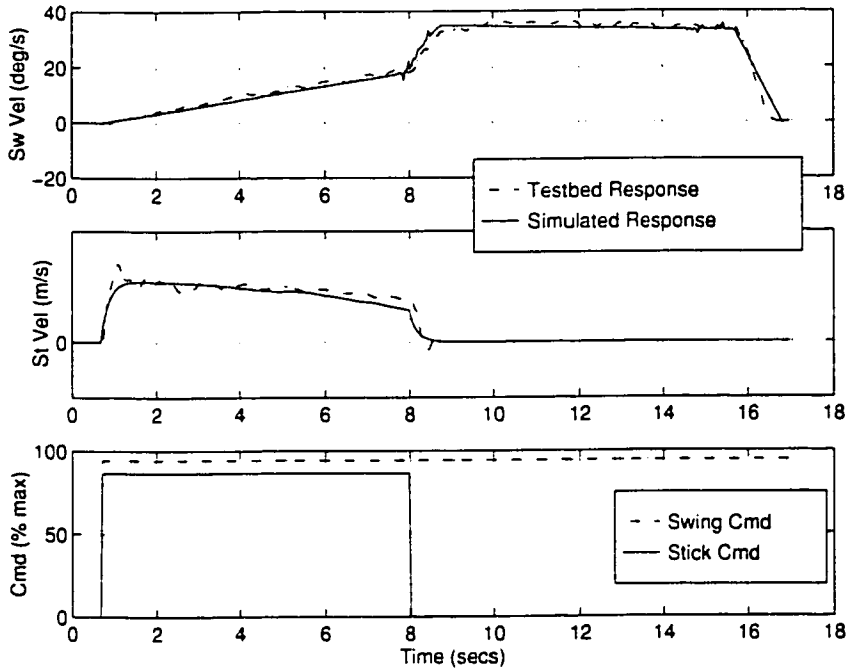


FIGURE 20. Sw Joint, St Cylinder Velocity Plots (Test #2 - *FF* Mode)

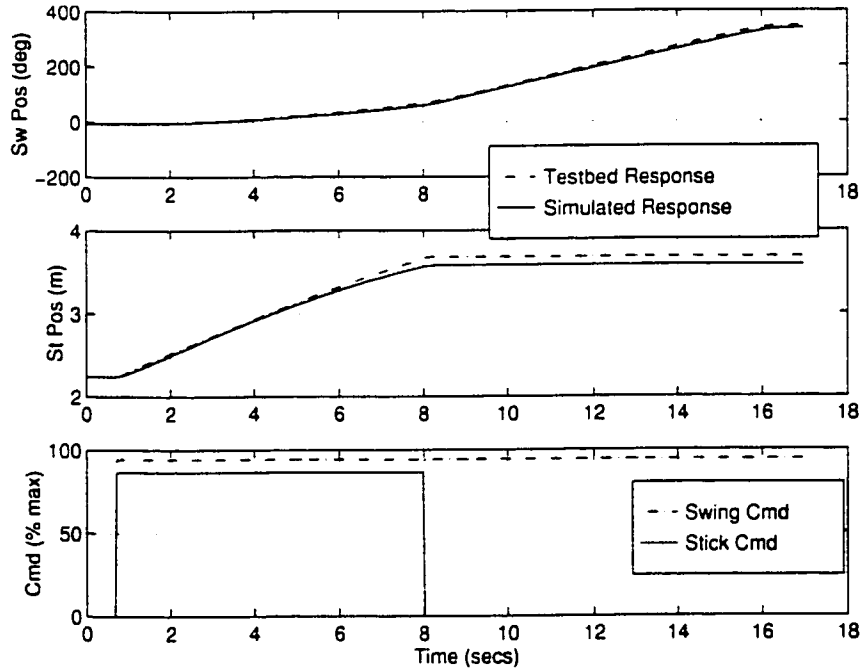


FIGURE 21. Sw Joint, St Cylinder Position Plots (Test #2 - *FF* Mode)

In the second test, the swing and stick joints are both simultaneously actuated by moving the *spool* valves of both joints. The stick joint does not seem significantly affected although the swing acceleration is significantly reduced. This is underscored by the sudden increase in swing acceleration when the St *spool* orifice is closed (at t = 8 secs). The model does a good job of predicting the joint responses in all cases.

The current implementation of the HEX *FF* mode model uses a locally-weighted regression scheme in conjunction with a table-lookup to approximate the data. This model runs at a run-time:real-time ratio of 40:1, i.e. simulating 40 seconds of motion requires 1 sec. of computation time on a SUN Sparc 20 workstation.

We profiled the model program code on a 143 MHz single processor SUN Ultra-1 workstation, which is faster than a Sparc-20 workstation. The model took an average of 1.7 msec (250,000 CPU cycles) to simulate 100 msecs of machine motion, i.e. simulating 40 secs of motion took 0.68 secs of computation time.

3.4.2 : Truck-loading mode tests (Used neural network function approximators)

We show here the results from the joint interaction tests conducted in the *truck-loading* mode. Recall that the hydraulic flow routing in this mode is more involved than in the *fine-finish* mode. Refer Figure 13 for an illustration of the plumbing for the *TL* mode. The joint interactions are no longer mainly confined to being between the Bm-Bk or the St-Sw joints. In the *TL* mode, the interaction between the Bm-St-Bk joints and Bm-St-Sw joints is most significant while there is little interaction between the Sw and Bk joints. These observations are made from analysis of the plumbing for the *TL* mode, and they form the basis for our choice of the tests shown in this section.

During the first test (Figure 22, Figure 23) the Bm, St and Bk cylinders were actuated simultaneously (as shown by the fourth subplot in all the figures) to demonstrate the interaction between them, while the second test (Figure 24, Figure 25) demonstrates interaction between the swing and stick joints. As in the *FF* mode tests, there is a significant Bm velocity oscillation, but its effect seem rather small when considered over a time period longer than that of the oscillation. In Figure 22, the boom is almost completely stalled when the Bm, St, and Bk are actuated. However, unlike the *FF* mode, the Bm does have a significant (albeit reduced) velocity when the St orifice is closed and only the Bm and Bk are actuated. Recall that the Bm was almost completely stalled in the *FF* mode when the Bm and Bk were actuated simultaneously (Figure 18). The enhanced Bm velocity in the *TL* mode is due to the use of the Bm2 valve (Figure 13) which provides excess flow from the pump supplying the St and Sw joints. This excess flow is minimal when the St (or Sw) is being actuated, but is significant when the St (and Sw) orifices are closed.

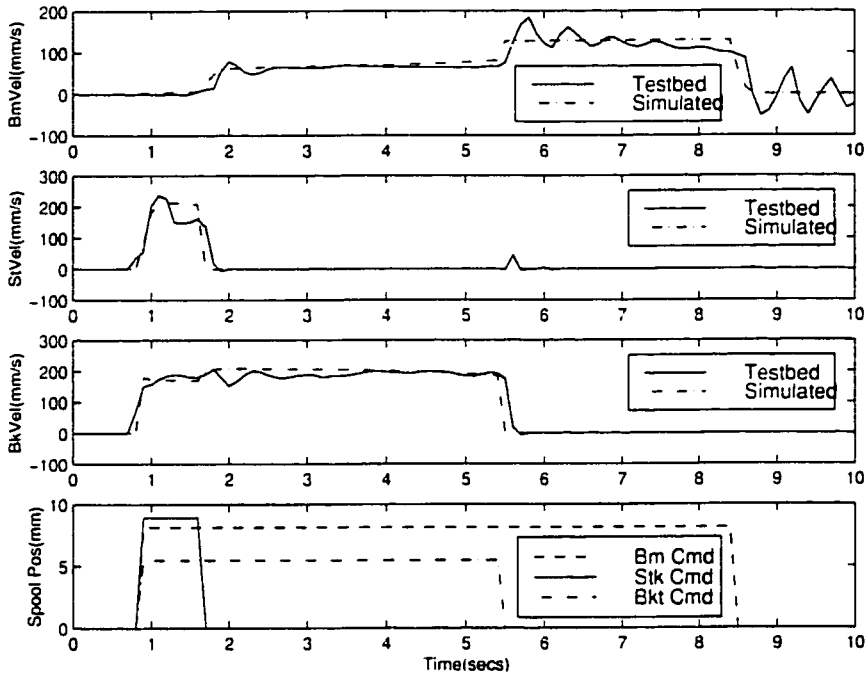


FIGURE 22. Bm, St, Bk Cylinder Velocity Plots (Test #1 - *TL* Mode)

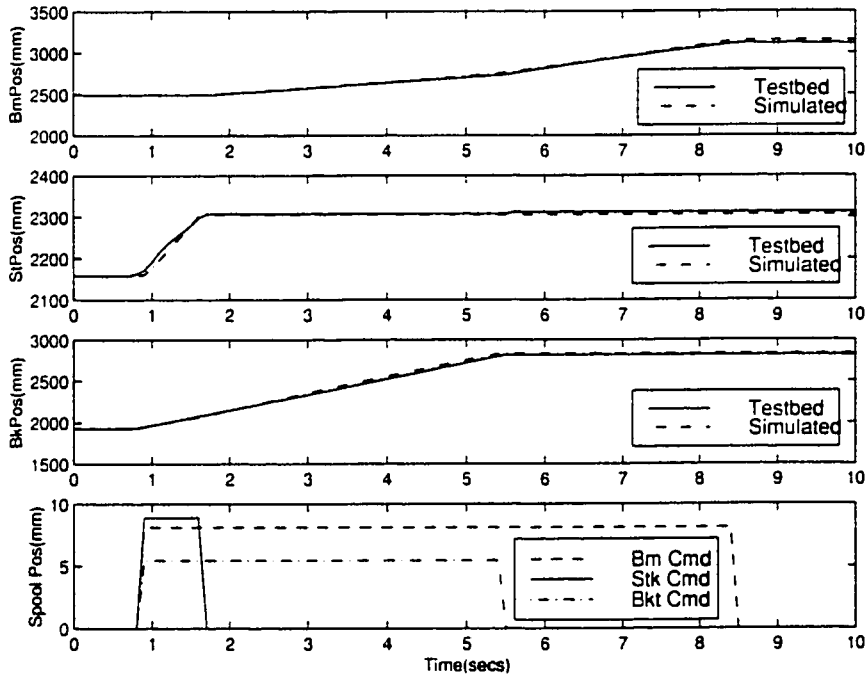


FIGURE 23. Bm, St, Bk Cylinder Position Plots (Test #1 - *TL* Mode)

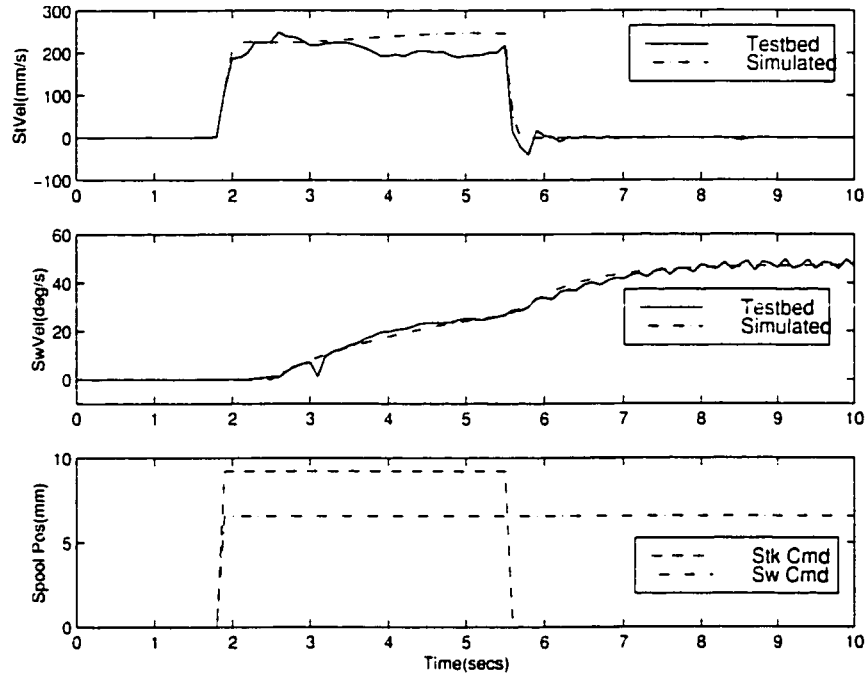


FIGURE 24. Sw Joint, St Cylinder Velocity Plots (Test #2 - *TL* Mode)

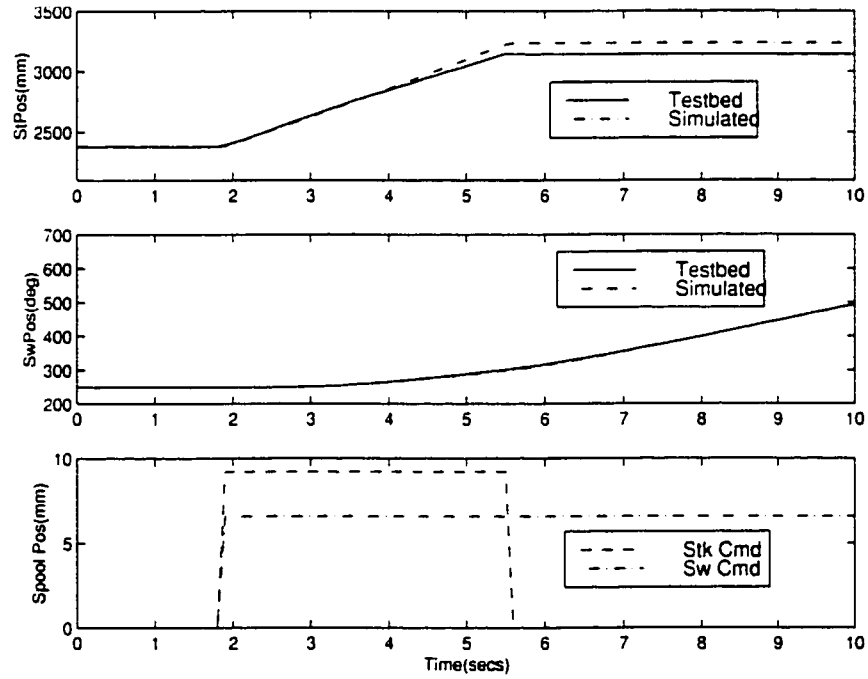


FIGURE 25. Sw Joint, St Cylinder Position Plots (Test #2 - *TL* Mode)

The second test (Figure 24, Figure 25) shows the simultaneous actuation of the St and Sw joints. The Sw joint was actuated only in combination with the St, and not with all the other joints, due to the logistics of ensuring that the HEX did not collide with the environment. The Sw and St actuators tend to interact most closely due to the nature of the hydraulic plumbing, although there is some interaction between the Bm and Sw joints as well. In hindsight, we realize that we overlooked performing any Bm-Sw joint tests, which may have been desirable to further characterize the performance of the models.

The results from both tests show that the model is able to model the joint interactions quite accurately in all cases. The current implementation of the HEX *TL* mode model uses 32 networks to cover the entire space for four joints. The model runs at a run-time:real-time ratio of 75:1, i.e. simulating 75 seconds of motion requires 1 sec. of computation time on a SUN Sparc 20 workstation.

We profiled the model program code on a 143 MHz single processor SUN Ultra-1 workstation, which is faster than a Sparc-20 workstation. The model took an average of 1 msec (150,000 CPU cycles) to simulate 100 msecs of machine motion, i.e. simulating 75 secs of motion required 0.75 secs. of computation time.

Section 3.5: Extended free-space motion test

Shown below are the results from a test where a human operator in the cab of the HEX performed operations similar to that during a normal loading cycle while the HEX was in the *FF* operating mode. The HEX was operated manually through joysticks, which control the position of the *hydraulic spool valves*. No soil interaction was involved and the bucket was empty for the entire duration of the motion.

Figure 26 shows the velocity response of the HEX joints on the testbed, along with the velocity predicted using the *FF* mode model.Figure 27 and Figure 28 show the error in the prediction of the position of the bucket tip. Figure 27 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.

Figure 28 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. The tip error has an average value of 1.0 m during most of the 50 secs while the maximum error is 1.8 m.

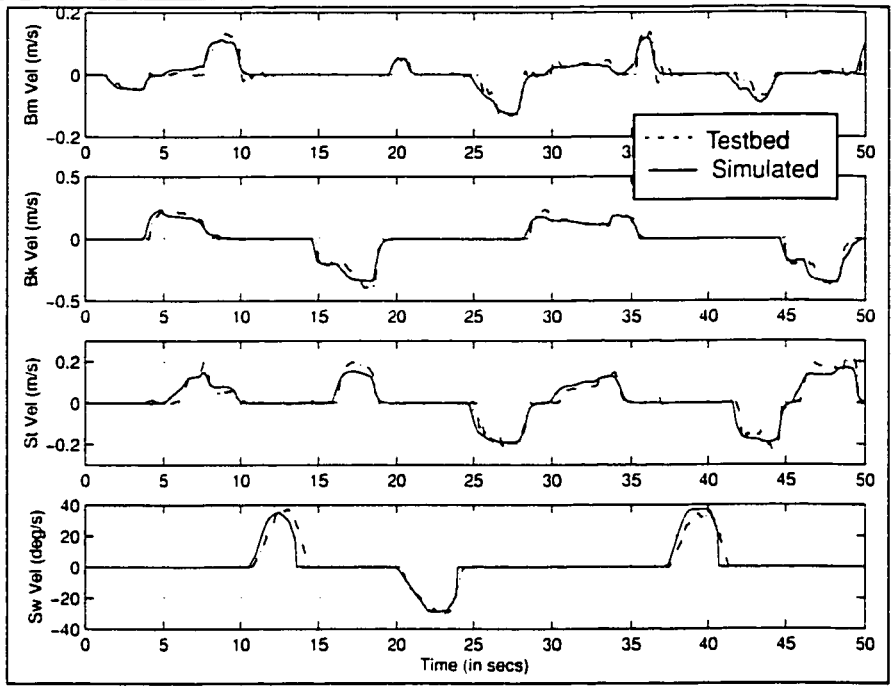


FIGURE 26. Velocity plots of the four HEX joints during the extended motion test.

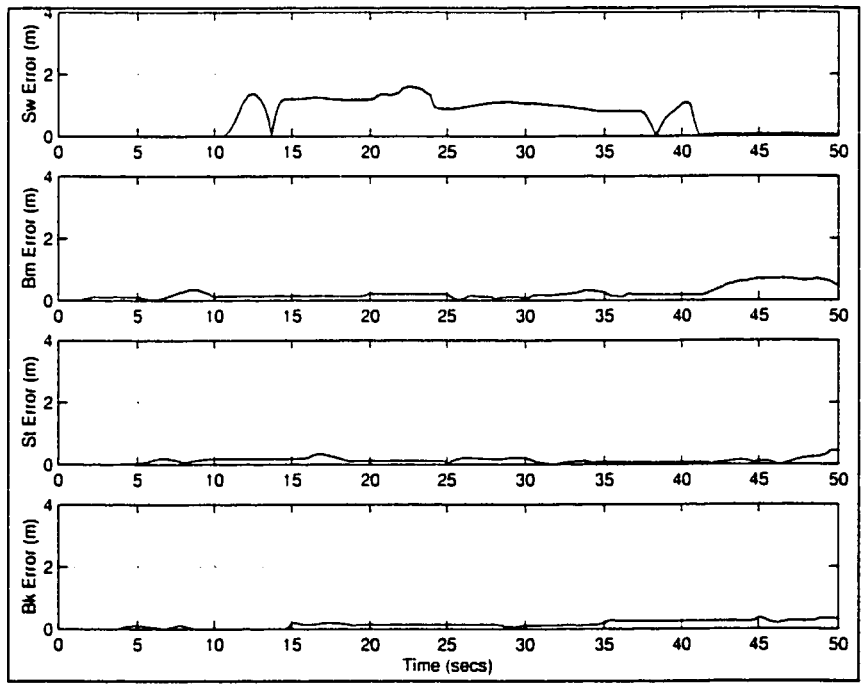


FIGURE 27. Tip error contributions due to each HEX joint during the extended motion test.

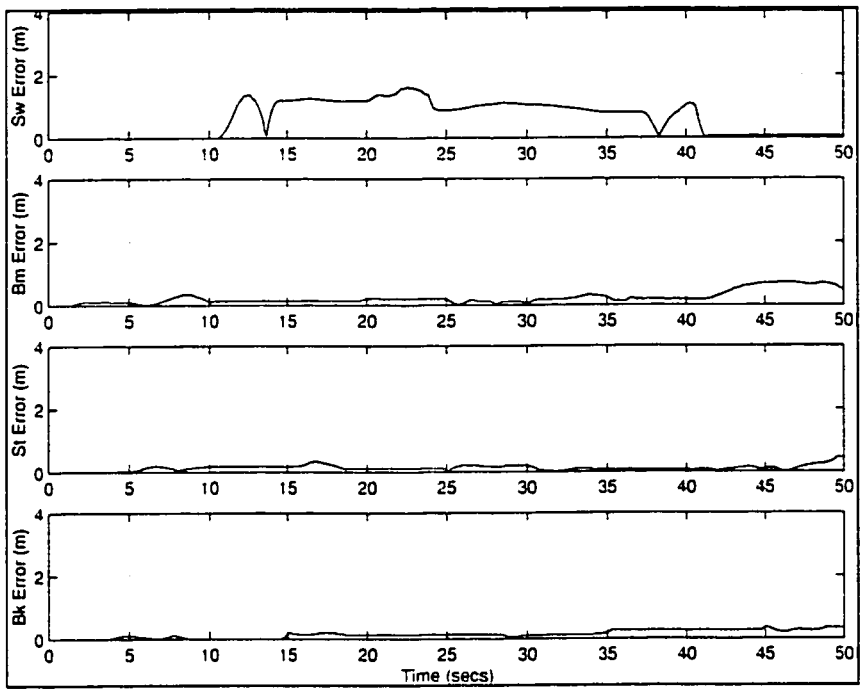


FIGURE 28. (a) Tip position error (b) Bk tip velocity during extended motion test.

These results show that the model prediction is good even for such an extended duration motion (50 secs). This duration is much higher than the duration of the tasks (in the following chapter) for which optimal motions are computed. We are encouraged by the fact that the error between the testbed joints and predicted motion in the first 10 secs. of motion remains quite low (0.1 m approx), especially when compared to the 1.5m width of the bucket. Most of our tasks have a duration of approximately 10 secs.

Section 3.6: HEX joint controller model

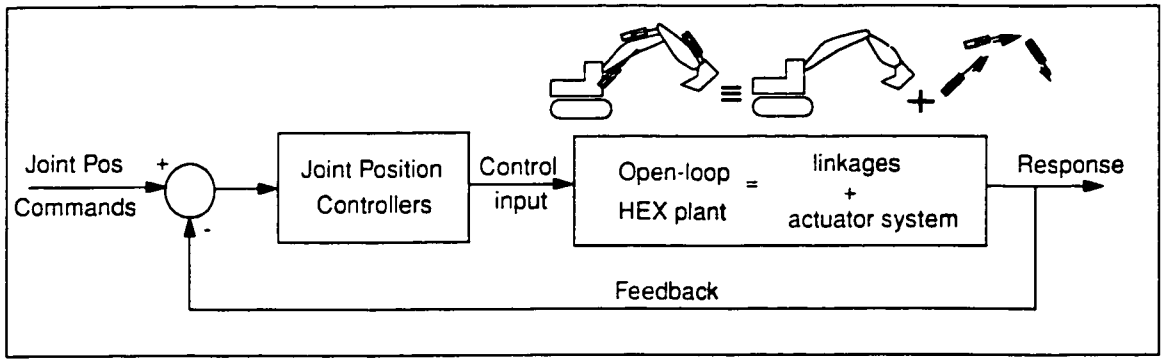


FIGURE 29. Schematic of closed-loop HEX robot (Reproduction of Figure 8).

The HEX testbed has independent PD (proportional-derivative) joint position controllers for each of its four joints. The joint controllers are modeled in state-space form as:

$$\dot{\underline{x}}_{joint} = A_{joint}\underline{x}_{joint} + B_{joint}u_{joint} \qquad \text{(EQ 12)}$$

$$\underline{y}_{joint} = C_{joint}\underline{x}_{joint} + D_{joint}u_{joint} \qquad \text{(EQ 13)}$$

where, u_{joint} is the position command to the joint, $\underline{x}_{joint}$ is a 5 element vector of state variables, y_{joint} is the *spool* position command, and A_{joint}, B_{joint}, C_{joint}, D_{joint} are the state-space matrices for the controllers. These controllers were not developed in this research. They cannot be described here in greater detail for proprietary reasons.

We can use the above joint controller model in conjunction with the open-loop HEX plant model to construct a model of the closed-loop HEX robot shown in Figure 29. This closed-loop HEX robot model is used in our optimal motion planning approach described in the next chapter.

Section 3.7: Concluding remarks

A model of a hydraulic excavator has been constructed by decomposing the complete 8th order non-linear system of equations into two separate models - the 2nd order linkage dynamic model and the actuator model. The models were used in a serial fashion, with the results of one being input to the other at every time step. Two non-linear actuator models were constructed - one to model the *fine-finish* HEX operating mode, and the other to model the *truck-loading* mode. The *FF* mode model was our first attempt at using function approximators to construct actuator models. It was constructed using a locally weighted linear regression approximator. The *TL* mode model was constructed later and uses a set of 32 neural networks to approximate the actuator response surfaces.

As an alternative to our approach of using non-linear function approximators, it is possible to approximate the actuator interactions using sets of linear surfaces. However, separating the actuator response surfaces into linear sections is non-trivial.

The main disadvantage of using a function approximator is the need to supply it with enough data to ensure that the approximated function is close to the desired function. Neural networks offer a significant speed advantage over non-linear regression-based function approximation techniques. However, they offer very little control over the function approximation created, thus requiring more extensive coverage of the data space.

The results shown above and in Appendix B show that both models are promising; they are able to accurately capture the complex actuator interactions that dominate HEX operation in the different operating modes. However, the models cannot simulate actuator transients since they are steady state models. The approach is therefore most applicable to robots where the steady-state response is dominant as compared to the transient response, which is true for most hydraulic robots due to the damping provided by the hydraulic fluid. However, there always remains the possibility that some unmodeled dynamics may cause undesirable oscillations (or other effects) which will not be detected when using a steady-state model to compute the optimal motions. This can be overcome by using a truly dynamic model to improve upon the motion computed using the steady-state model by perturbing it locally, i.e. through small modifications of the steady-state result. The perturbations required should be small since the steady-state response is dominant as compared to the transient response.

The HEX models are much faster than comparable analytical models. Our benchmark tests show that the HEX model using the regression-based actuator model is 20 times faster than a comparable numerical solver-based model, while the HEX model that uses the neural network based actuator model is 2.5 times as fast as the regression-based model. Our hydraulic robot models are ideal for applications where the model is used for extensive simulation, and where computation time is a precious resource. They are therefore suitable for use in optimal motion planning for hydraulic robots. The next chapter addresses the problem of optimal motion computation, given just such a model.

CHAPTER 4

Optimal Motion Planning

In the previous chapter, Chapter 3, we developed a method for modeling the complex actuator interactions of a hydraulic robot. We used the resulting actuator model, in conjunction with linkage dynamics and joint controller models, to build a complete HEX closed-loop robot model (See Figure 29, Chapter 3).

This chapter presents an approach to optimal motion planning for hydraulic robots that uses the above-mentioned robot model. As before, we use the HEX to illustrate our approach. The optimal motion computation problem is shown in Figure 30.

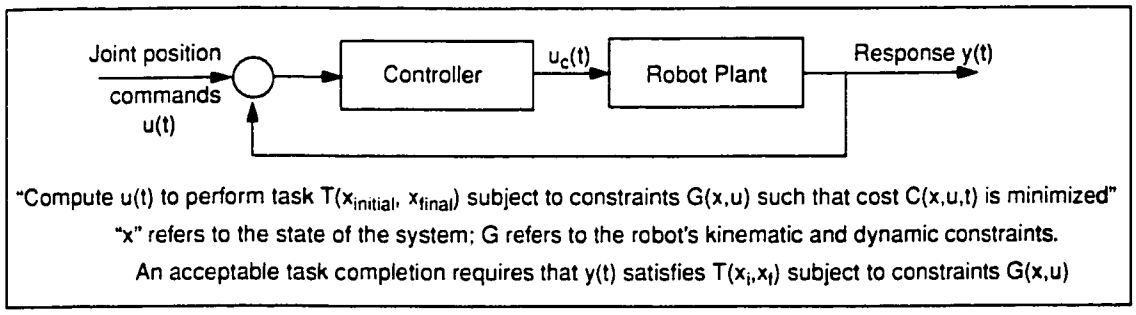


FIGURE 30. Reproduction of Figure 3 (Chapter 1)

In the language of optimization, the optimal motion planning problem is:

$$\underset{u(t)}{Minimize} \quad C = \int_{path} c(x, u, t)dx \qquad \text{where C = cost function;} \tag{EQ 14}$$

s.t. $y(t) \models T(x_{initial}, x_{final}), G(x, u)$ where T refers to the task, and G refers to the set of constraints.

The term "constraints" refers to robot motion constraints, i.e. the limitations such as a finite response time and magnitude, which are imposed by the robot model. Factors commonly referred to as constraints in the literature, such as avoiding obstacles in the environment, avoiding spillage etc., are fundamentally different in nature from the robot motion constraints, and are not referred to as such in this document. The difference lies in the fact that while it is impossible for the HEX robot to move at infinite speed (and this is hence a constraint), the robot can collide with the environment, or spill the load in the bucket (although both those are undesirable). The latter are therefore not constraints, but undesirable properties of a robot motion. We can avoid them by penalizing motions which cause those effects. The minimization process will thus automatically avoid their occurrence.

The goal of optimal motion planning is to find a means of computing the input **u*(t)** (* denotes optimal) to optimally execute a specified task for the system shown in Figure 30.

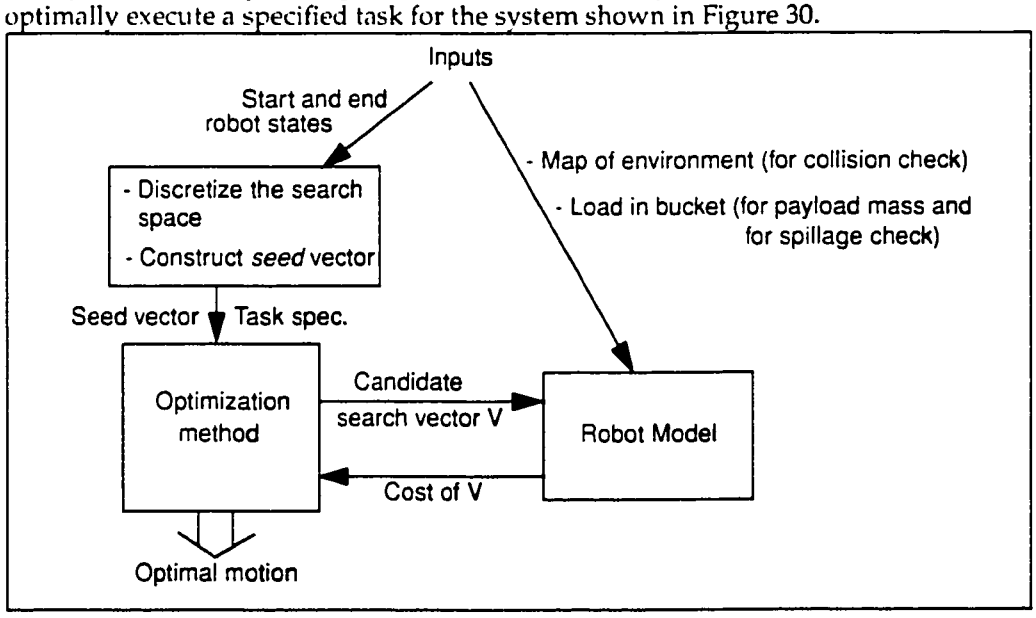


FIGURE 31. Schematic showing our approach to optimal motion computation.

Our approach to optimal motion planning, illustrated in Figure 31, consists of two steps:

- Discretize the robot's command space.
- Use a robust search technique to traverse the discretized space and find the sequence of robot commands that perform a specified task optimally.

4.0.1 : Discretize the robot's command space

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

Hydraulic robots are usually large and significantly damped systems which act as low-pass filters to any input they receive. The use of a set of joint controllers (as proposed in our problem statement, Section 1.1) further reduces the order of the robot system. Thus, the (optimal) input **u(t)** (joint position commands) to the robot 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 discretization is not too fine. Robots with higher bandwidths may require a much finer discretization, and hence a longer search vector **u[k]**, which will result in a longer search to find the optimal.

4.0.2 : Use a robust search technique to find the optimal

Each point in the search space is a temporal sequence of robot commands (joint position commands for our HEX robot). The search technique starts the search from an initial point (the *seed*). It uses the robot model to simulate (and thus evaluate the cost of) the command sequence associated with the *seed* (or current search space point). The cost evaluation is used to navigate towards a better point, and ultimately the optimal command sequence.

The robot model used in the simulation and evaluation is the one developed in Chapter 3. Since the optimization involves an extensive search, it is important to use such a fast model, to ensure that the search (involving tens of thousands of motion simulations) will conclude in a reasonable length of time.

The essential elements of our optimal motion planning approach are:

Search space: The search space for the optimization is the space of all possible joint command sequences.

Constraint surface: This represents the robot's physical constraints: the kinematic limits, actuator and linkage dynamic constraints, and constraints due to the controller.

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

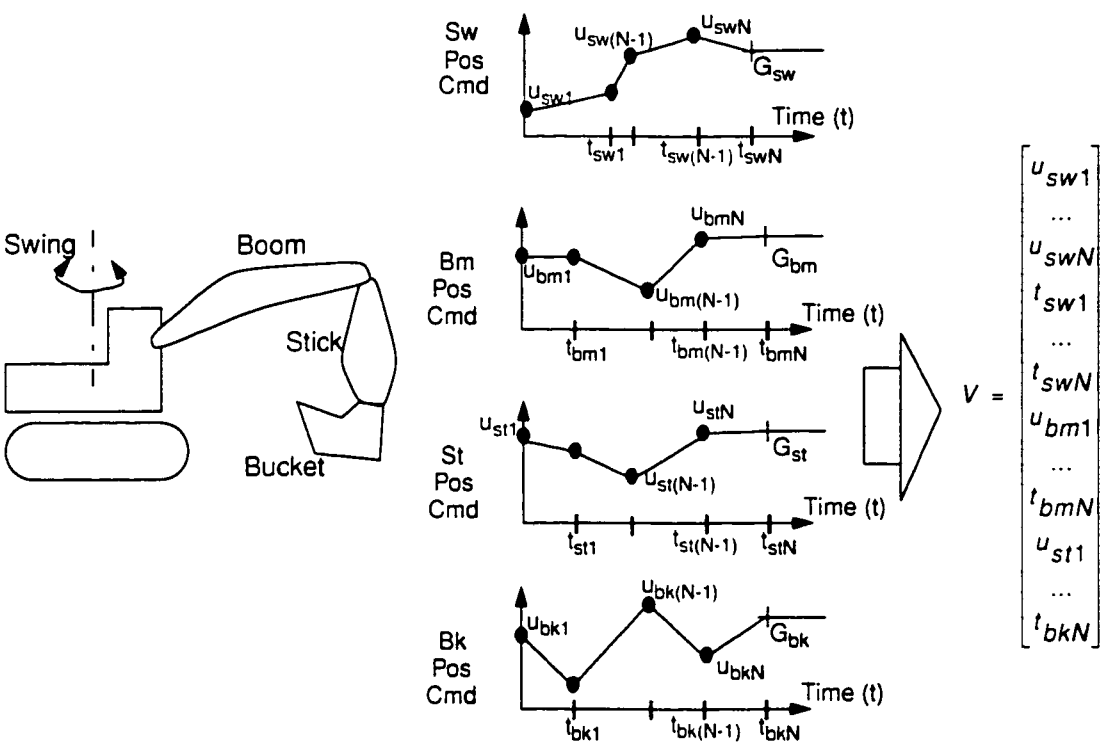
Optimization algorithm: The specific optimization algorithm used dictates the method of exploring the search space, starting from an initial (sub-optimal) path and moving to the optimal path. For instance, a simple gradient descent would attempt to move in the direction of steepest descent.

The following sections (Section 4.1 through Section 4.4) describe these four elements in greater detail. We discuss some implementation specifics in Section 4.5, and conclude the chapter with a summary in Section 4.6.

Section 4.1: Search space

The search space used in our approach is the space of joint position commands. Figure 32 shows our approach to the construction of a search vector V. Each temporal joint command sequence is expressed as a piece-wise linear connection of N+1 *knot points*. The coordinates of each knot point consist of a command value (u) and a time value (t). Although N+1 knot points are used for each joint, only 2N elements (instead of 2*(N+1)) in the search vector correspond to each joint. This is because the time coordinate of the first knot point of each joint is fixed (zero), and the last knot point command value is equal to the goal value (G_{joint}). Note that the joint command value after the (N+1)'th knot point remains at the goal value for that joint (G_{joint}). Therefore, each joint's command sequence is described using 2N values: N u and N t values.

The 8N u and t values are stacked together to construct the search vector V, as shown in the figure below. Using this search space definition, the command value for any of the four joints at any time t, can be obtained by linear interpolation between the neighboring knot points.



NOTE: (1) u_{swi}, u_{bmi}, u_{sti}, u_{bki} are the command values for the i'th Sw, Bm, St, Bk knot points respectively.

(2) The t_{swi}, t_{bmi}, t_{sti}, t_{bki} are the time values for the i'th Sw, Bm, St, Bk knot point respectively.

(3) G_{sw}, G_{bm}, G_{st}, G_{bk} are the goal values for the Sw, Bm, St, Bk respectively. In the plots they mark the (N+1)'th knot point describing the joint command sequence.

FIGURE 32. Construction of the search vector

The above search vector approach yields a surjective mapping i.e. the mapping from search space to cartesian space is many-to-one (Figure 33). In other words, every point in the search space corresponds to a unique path in cartesian space, the inverse of which is however not true. The validity of the statement that each search space point maps to a unique cartesian space point (under the assumption that

the initial states for all the search space points are the same) follows from the definition of the state[1] of a system. To prove the inverse with an example, consider the simple case of single joint motion. Two different command sequences, both of which however result in saturating the joint being actuated, will yield the same response. Thus, two search space points could map to the same cartesian space path. The uniqueness of the mapping from search space to cartesian space ensures that every search space point can be mapped to a unique cartesian path, which can be examined to determine whether any the robot collides with the environment along that path.

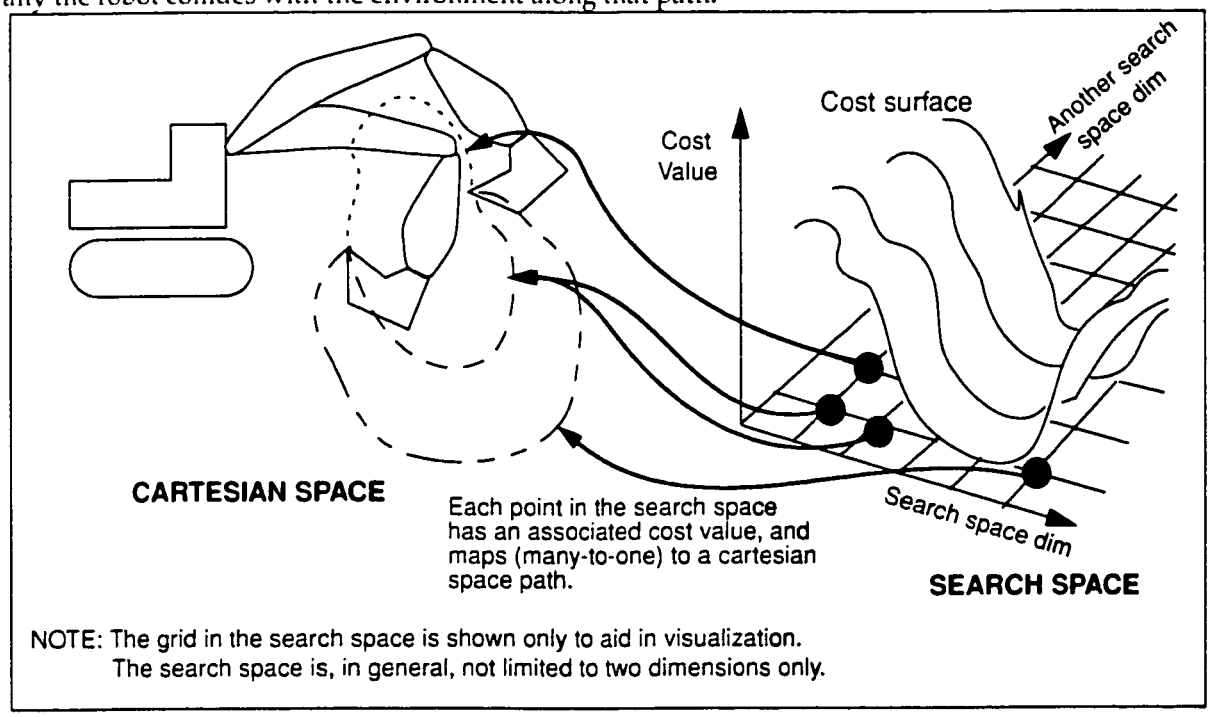


FIGURE 33. Relationship between cartesian space and search space

There are two main alternatives to the use of the above search space:

1. One alternative is to use the space of end-effector paths. The paths could be described in cartesian or joint space. The approach used by Shiller and Dubowsky [Shiller 91] is one example. Their search was in the space of possible cartesian end-effector paths (Ref Section 2.2 for more details).

1. A *state* of a system is defined as the smallest set of variables such that the knowledge of these variables at $t=t_0$, together with the input for $t >= t_0$ completely determines the behavior of the system for time $t >= t_0$ [Ogata 90].

The main disadvantage of searching in the space of "desired" paths, instead of the command space, is that all alternatives in the desired-path space are not feasible, i.e. it may not be possible to follow some of the end-effector path candidates. In contrast, all points in the command space are feasible. Thus, the search will not spend time examining infeasible candidates.

2. The other alternative is to use a cartesian command space, as opposed to the proposed joint command space. In this alternative, the commands would be described as a sequence of cartesian space knot points, similar to the joint space knot points used in our approach.

   The disadvantage of such a space (which applies to any cartesian space approach: command space or desired-path space), is the difficulty in imposing kinematic joint limits (and hence robot workspace limits) since the inverse kinematics have to be performed to ensure that each point along the commanded cartesian space path is within the workspace. In contrast to this, it is easy to impose joint limits in our search space since the our commands are in joint space.

   The existence of a unique mapping is required to ensure that the optimal point in the search space can be mapped back to a cartesian space path. This property is especially useful when the robot has redundant linkages since the inverse kinematic mapping is not unique; a cartesian approach is difficult to use for such robots.

The above definition of the search vector allows the user to select a desired discretization of the command stream - the finer the discretization, the more the knot points, and hence the higher the dimensionality of the search vector. The length of the search vector is given by:

$$SrchVectorLength = 2N \times (numJoints) \quad \text{(EQ 15)}$$

If the number of knot points is 2, i.e. N=2 in Figure 32, the search vector length for the HEX is:

$$SrchVectorLength = 4 \times 4 = 16 \quad \text{(EQ 16)}$$

As shown in Figure 32, each joint command is set to the goal position for that joint after the last knot point, e.g. at $t_{sw,N}$ the command value is set to G_{sw}, and stays at that value thereafter. This ensures that the goal is always reached, unless a collision prevents it. This is an important addition since now the goal is reached by all search vectors that do not collide with the environment (as long as the low-level controller does its job adequately). It may be reached after a long time and thus very sub-optimally, but nonetheless it will be reached.

Most optimization techniques require an initial instance of the search vector. This *seed search vector* is used to seed the search by providing a starting location for the exploration of the search space. In the ideal case, the exploration strategy (or optimization algorithm) will be completely insensitive to the choice of the *seed search vector*. However, all optimization methods are somewhat sensitive to the choice of the initial guess. Our goal is to select a method that is as insensitive to the choice as possible. This issue is addressed further in the section below on the optimization algorithm selection.

Section 4.2: Constraint surface

Since we use a search approach to optimal motion planning, the search technique used (described in Section 4.4 below) traverses the search space in search of the optimal command sequence. It moves from one point to another, using the cost of each successive point as a guide to the optimal. As explained in Section 4.1, our search space is the robot's command space, i.e. each point in the search space is a temporal sequence of robot commands.

The evaluation of each search space point is done by using a model to simulate the robot's response to the command sequence associated with the point. Thus, the kinematic and dynamic constraints (actuator and linkage dynamics), and constraints due to the controller are automatically imposed during the cost evaluation.

The robot model used for the cost evaluation in our HEX implementation of the optimal motion planning is the model developed in Chapter 3.

Section 4.3: Cost function

The general form of the cost function is:

$$C_{path} = \lambda_{time} t + \lambda_{energy} E + \lambda_{obstacle} O + \lambda_{spillage} S \qquad \text{(EQ 17)}$$

where the λ_i's are the weights, t is the time taken to execute the motion, E is the energy consumed in performing the motion, O is the extent of the collision, and S is the amount of spillage.

The cost function is constructed to allow a user to select between time and energy consumption as the objective function to minimize.

Optimizing time is akin to optimizing productivity, which is defined as:

$$productivity = \frac{soilThroughput}{timeTaken} \qquad \text{(EQ 18)}$$

Thus, by minimizing the time taken to perform a task (for a given bucket load), we are maximizing the productivity, which is commonly referred to in industry parlance as tons-per-hour (tons/hr.).

In a similar vein, optimizing energy is akin to optimizing fuel efficiency, which is defined as:

$$fuelEfficiency = \frac{soilThroughput}{massOfFuelConsumed} \quad \text{(EQ 19)}$$

In an energy optimization we are minimizing the fuel consumption for transporting a certain amount of soil to a truck, hence maximizing the fuel efficiency.

We allow the user to select between time and energy as the objective function. However, we have not implemented a function that allows the combination of the two quantities. Merging the two in a single cost function would require careful weighting of the time and the energy values since they are disparate quantities. Therefore only one of either λ_{time} or λ_{energy} is set to a non- zero value. If λ_{time} is non-zero, it is set to 1, while λ_{energy} is set to 0.01 when it is non-zero.

The weights associated with the obstacle collision and spillage terms are set very high so that the optimizer considers them both to be extremely undesirable. We set $\lambda_{obstacle}$ to 10000 and $\lambda_{spillage}$ to 5000. Based on these choices of the weights, the costs of different motions will be in one of three ranges: those that cause collisions (highest cost), those that cause spillage (but no collisions), and those that neither cause collisions not spillage (lowest cost). (As explained in Section 4.3.3, we terminate the simulation of a search vector as soon as either spillage or collision is detected. Thus, the O and S terms in Equation 17 will never be non-zero at the same time).

The user has the option of setting $\lambda_{spillage}$ to zero in case the motion being optimized involves an empty bucket, since doing that relaxes the constraints on the optimization and allows more motions to be examined.

The details of the computation of the terms in Equation 17 are discussed below.

4.3.1 : Time term

The time taken to execute a particular command sequence (or search vector) is obtained by using the robot model to simulate the robot response. The time computed, t, is in seconds.

4.3.2 : Energy term

We have used two different approaches to computing the energy consumption during robot operation. For the FF operating mode, the energy consumption is computed as:

$$E = \sum_{joints} \left(\int_{path} F_i dx \right) \quad \text{(EQ 20)}$$

where F_i is the force (or torque) generated by the i'th cylinder (or motor).

This energy term computed is the energy expended at the cylinder since it is computed as a product of the force/torque at the cylinder/motor and the displacement. We refer to this energy measure as the *implement energy*, since it reflects the energy expended at the implements. 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, and hence the fuel usage (This measure, albeit close, will still not be a complete measure of the fuel consumption since it will not include losses within the engine).

We tried to address this problem when constructing the model for the TL mode of operation. We constructed a neural network to approximate the power usage. The 5 inputs for this network were the joint velocities of all the four HEX joints, and the swing acceleration. The network output was the energy consumption rate at that instant. The data to train this network was collected from the detailed HEX model. The energy rate was obtained by tapping the engine shaft torque and velocity in the detailed HEX model; the product of the torque and velocity gave us the engine's energy output at that instant. This neural network provides a model of the engine output power and includes hydraulic and other losses in the system, in addition to the energy expended at the cylinders/motor. We refer to this energy measure as the *engine energy*.

The energy consumption for a given HEX motion is computed by integrating the energy usage rate (power) at every time step during the motion. The energy consumption term, E, is computed in kJ.

4.3.3 : Obstacle term

Collision avoidance is an important part of robot operation. To inform the optimization about the obstacles in the environment, a terrain map of the region around the robot is input to the robot model (Figure 31). A terrain map is a grid of height values for the region around the robot, i.e. each grid point on the horizontal plane has an associated height value. The map is a 2.5-dimensional view of the world, i.e. each grid point is only allowed one height value. This representation does not allow overhangs to be represented. However, it is found to be adequate for the terrains commonly seen in HEX operation. We construct our terrain maps using data acquired from a laser range scanner mounted on the HEX robot. This range scanner is capable of scanning all around the HEX (360^0 azimuth). For

regions where range data is missing due to occlusion, we interpolate the surrounding values to patch the hole in the data.

FIGURE 34. Triangular mesh of a pile of soil. The mesh has been filled in and shaded.

We convert the terrain map to a triangular mesh representation (as shown in Figure 34) where each triangle covers half of a 0.25m-by-0.25m patch. We also approximate the robot as a set of triangles. Checking for collisions thus involves checking for intersections between the two sets of triangles: the triangles from the HEX robot, and the environment mesh. The triangulation allows the collision checking to proceed very quickly. We check for intersections between triangles using a publicly available software routine called *Rapid*. This software is available from the Computer Science Department of the University of North Carolina at Chapel Hill (http://www.cs.unc.edu/~geom/collision_code.html) [Gottschalk 96].

At each time step during the simulation of the robot motion, a collision check is performed. If the goal is reached and no collision has been detected, then the O term in Equation 17 is set to zero. If a collision is detected during the simulation, the simulation is stopped and a collision penalty is assigned to the search vector being evaluated, i.e. the t, E, S terms in Equation 17 are assigned zero while the O term is set to:

$$O = NumOfIntersectingTriangles + (|\Delta Sw| + |\Delta Bm| + |\Delta St| + |\Delta Bk|) \quad \textbf{(EQ 21)}$$

$$\Delta Jt = CurrentJtPosition - JtGoalPosition;\ Jt = Sw, Bm, St, Bk \quad \textbf{(EQ 22)}$$

The t, and E terms in Equation 17 are set to zero since the goal has not been reached. Therefore, those terms do not have any significance in this case. The S terms is set to zero since no spillage has occurred (more on spillage in Section 4.3.4 below).

The high weight ($\lambda_{obstacle} = 10000$) associated with the obstacle term O (ref Equation 17) ensures that the cost of the search vector (that caused the collision) is set to a high value. The second set of terms in Equation 21 add a measure of the distance of each joint from the goal, at the time of the collision. This helps provide a gradient, i.e. it results in a higher cost for collisions further away from the goal. Using a barrier cost function (fixed high cost) does not give the optimization any information on the direction that will lead to a lower cost, and is therefore undesirable.

The above approach of looking for intersections between any two triangles also easily allows the examination of self-intersections. We look for collisions between triangles representing the different links of the robot; an intersection indicates a collision of two robot links. A self-collision is considered similar to a collision with the environment.

4.3.4 : Spillage term

A spillage constraint is imposed when the HEX bucket is loaded. This constraint is imposed by checking the bucket joint position to see if the bucket opens before it is over the truck. The truck location and bucket load are two inputs that the user supplies when starting the optimization. The bucket angle is checked at each time step of the robot response simulation to see if it is greater than the start bucket angle. If it is, then the location of the bucket in cartesian space is computed. If the bucket is not over the truck bed then the simulation is stopped and that command stream (or search vector) is assigned a spillage penalty, i.e. the t, E, O values in Equation 17 are assigned zero while the S term is set to:

$$S = |\Delta Sw| + |\Delta Bm| + |\Delta St| + |\Delta Bk| \qquad \textbf{(EQ 23)}$$

$$\Delta Jt = CurrentJtPosition - JtGoalPosition;\ Jt = Sw, Bm, St, Bk \qquad \textbf{(EQ 24)}$$

If no spillage is detected throughout the simulation, the S term in Equation 17 is set to zero. The spillage cost creates a gradient, just as in the collision cost (Section 4.3.3). The cost of spillage is higher when the spillage occurs further from the goal. The optimization algorithm can thus use that information to move towards a solution where the spillage is at the goal (which is the desired result for the HEX applications we have considered).

Since we do not model the dynamics of the robot actuators (we only use a steady-state actuator model), we do not look for spillage due to jerk. We have not found this to be a problem, but it is possible that the need to include this may be felt in the future. It can be solved by using a dynamic model of the robot, and by modeling the dynamics of the soil in the bucket (instead of using the lumped-mass assumption used in Section 3.1.1).

4.3.5 : Other terms

There is always the possibility of adding other terms to the cost function. One of the factors that affects the productivity[2] of human operators is the amount of careful co-ordination and simultaneous modulation of multiple joints. Reducing the frequency of operator input usually enhances the productivity of the operator. Thus, one measure of interest might be the "busyness" of the command sequence, i.e. it may be desirable to find the input with the least modulation of the commands. The "busyness" can be measured as a combination of the time during which at least one of the joint commands is modulated (this prefers step inputs over ramp or other continuous modulation inputs) and the number of command transitions (which keeps the number of step inputs to a minimum). Thus, a command sequence with few step inputs will have a lower cost than one that requires ramp inputs, or one with a larger number of step inputs.

Section 4.4: Optimization algorithm

The process of optimization can be viewed as a walk on the cost surface towards the point of locally least cost, such that each point along the walk satisfies all the constraints.

Since our robot model can only evaluate a search vector but not provide gradient information, we have to rely on non-gradient-based methods. (Numerical differencing to obtain gradient information is not a generally recommended approach [Press 96]).

We used the downhill simplex minimization algorithm [Press 96, pp. 408] in our first attempt to find the optimal motion for a given task. A simplex is a geometrical figure consisting, in K dimensions, of K+1 points or vertices, and all their interconnecting line segments, polygonal faces etc. A two dimensional simplex is a triangle, while a three dimensional one is a tetrahedron, not necessarily regular. For an K-dimensional minimization, the downhill simplex algorithm is given a set of K+1 points that define an initial K-dimensional non-degenerate simplex, i.e. that which encloses a finite inner K-dimensional volume. The algorithm then takes a series of steps, with most steps just moving the point of the simplex where the function is largest (highest point) through the opposite face of the simplex to a lower point. This is called a reflection operation. The algorithm also employs non-rigid transformations such as anisotropic contraction or expansion. These operations make the algorithm quite robust by allowing it to "squeeze" through narrow gaps in the cost surface. For more details of the downhill simplex method refer [Press 96, pp. 408].

2. This information was gleaned from conversations with Steve Lunzman and Lonnie Devier of Caterpillar Inc.

The results indicated that the downhill simplex did very well if no local minima were present. However, similar to most other downhill-only methods, it is quite susceptible to local minima because the simplex only moves in a downhill direction and does not accept uphill moves at any time. This makes it almost impossible for the algorithm to get out of a valley if all the vertices of the simplex are contained in the valley.

We tried using the downhill simplex algorithm with multiple restarts. For each restart we chose a different starting location and different extent of the initial simplex. The results were not encouraging even for N=1 (we used search vectors that were 8 to 40 elements long; N=1 through 5). The algorithm continued to get mired in local minima and was very sensitive to the location of the initial simplex.

We therefore needed an optimization method that was robust to local minima, and did not require gradient information. A 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.[Kirkpatrick 83]. It has gained 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. If, however, the cooling is performed 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 and allowing ample time for redistribution of the atoms as they lose mobility. This is essential to ensure that a low energy state will be achieved.

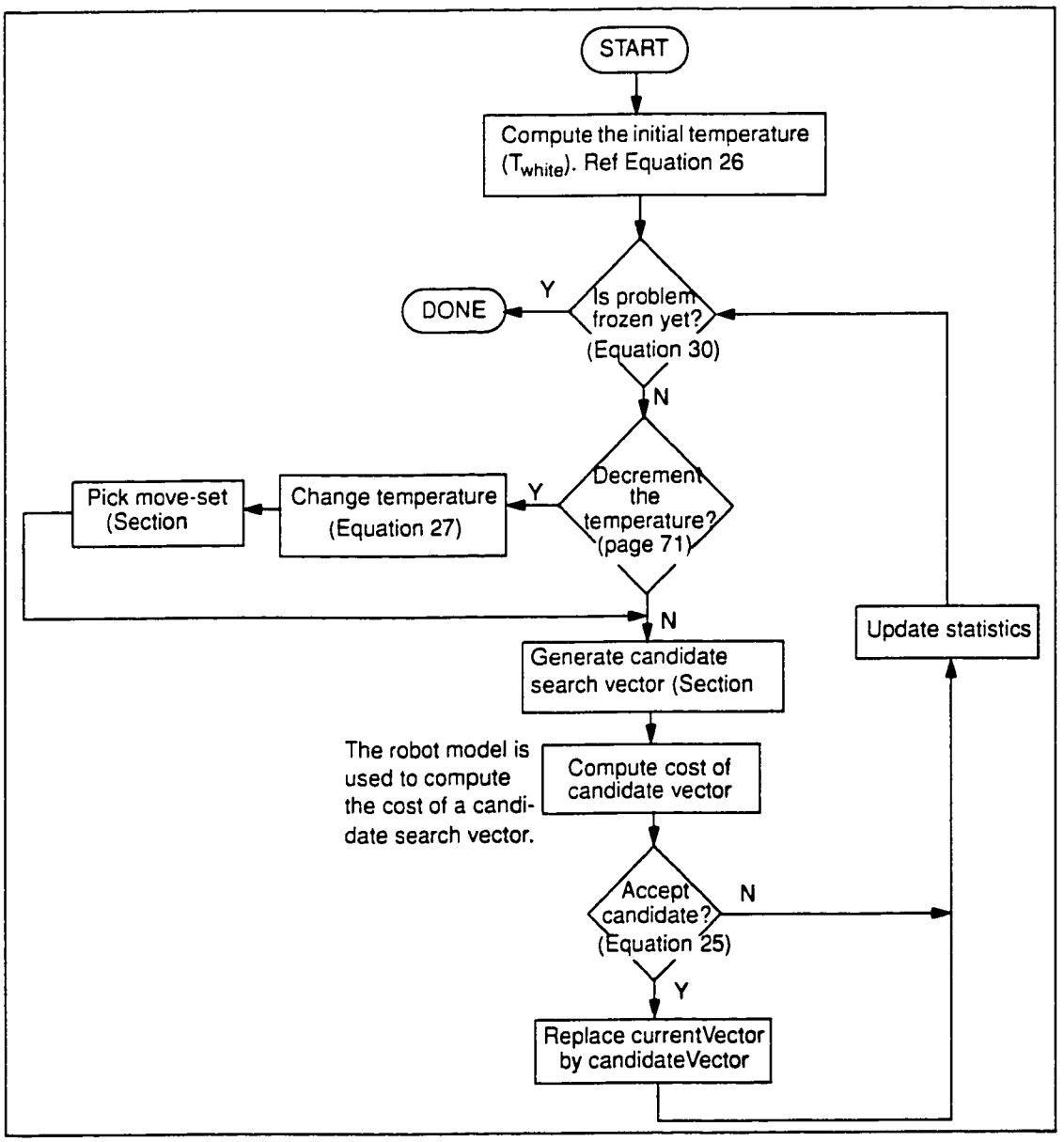


FIGURE 35. Flow-chart of SA search algorithm

Our implementation of the SA algorithm is shown in the flow-chart in Figure 35. The SA algorithm starts off at a high temperature and the change from state 1 to state 2 is accepted with probability:

$$p = e^{\left[-\frac{(C_2 - C_1)}{T}\right]} \qquad \text{(EQ 25)}$$

where C_1 is the cost at state 1, C_2 is the cost at state 2, and T is the current temperature. If $C_2<C_1$ then the move has a probability greater than 1 and is therefore always accepted. An uphill move ($C_2>C_1$) is accepted with probability p given by Equation 25. Uphill moves are more readily accepted at high temperatures, while the probability of an uphill move is low at lower temperatures.

The SA routines were implemented using a software library, *Acacia*, provided by Tamal Mukherjee [Ochotta 95]. This library has been compiled over the years by researchers in the Electrical and Computer Engineering department at CMU, and has been tested in the domain of Electronic Computer-Aided Design.

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 slightly higher energy state and then be unable to get out of the local minimum. It is therefore recommended that multiple instances of the annealing be performed to increase the probability of the SA reaching the global minimum.

Aside from the search space representation and cost function already described earlier in this chapter, the SA algorithm requires the definition of a cooling schedule and move-set, which are detailed below.

4.4.1 : Cooling schedule

The tests used a cooling schedule proposed by Huang, Romeo, and Sangiovanni-Vincentelli [Huang 86]. Using this schedule the hot temperature (or start temperature) is determined using White's [White 84] 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_∞, to determine the standard deviation, σ_∞, of the cost function.Then the following formula is used to compute the starting temperature:

$$T_{white} = \frac{(-3)\sigma_\infty}{\log(acceptRatio)} \qquad \text{(EQ 26)}$$

where $acceptRatio$ is a user-specified parameter which corresponds to the desired acceptance ratio at T_∞. The temperature decrement is performed by the formula:

$$T_{i+1} = T_i \exp\left(\frac{-\lambda T_i}{\sigma_i}\right) \quad \text{(EQ 27)}$$

where σ_i is the standard deviation of the accepted costs (Ref Equation 25 for details of move acceptance) at the previous temperature, and λ is a user-supplied parameter. Note that the subscript *i* indicates the previous temperature, while *i+1* indicates the new temperature. If

$$\sigma_i = 0, then, T_{i+1} = T_i \times r \ . \quad \text{(EQ 28)}$$

where r is user supplied ratio. Also, if

$$\exp\left(\frac{-\lambda T_i}{\sigma_i}\right) < minRatio \quad \text{(EQ 29)}$$

then $T_{i+1} = T_i \times minRatio$, where $minRatio$ is another user-supplied parameter.

The number of moves per temperature is adjusted to keep the annealing state near equilibrium[3]. The algorithm to do this is as follows:

1. If at any time $maxTriesPerTemp$ is reached, then move on to next temperature.
2. Perform at least $minMovesPerTemp$ accepted moves (Recall that Equation 25 is used to decide whether to accept a move). If unable to get enough accepts before reaching $maxTriesBeforeEquil$, then done at this temperature.
3. Determine if at equilibrium by matching a histogram to the cost of accepted moves. We use two buckets - inside and outside. The boundary between the buckets is set as $\mu_i \pm 0.5\sigma_i$, where μ_i and σ_i are the mean and standard deviation of the costs to date at the present temperature. Each accepted move increments the count of either the inside or outside bucket. There are counting limits for each bucket.

   $insideBucketLimit = 3 \times 0.38 \times equilMultiplier$, and

   $outsideBucketLimit = 3 \times 0.62 \times equilMultiplier$;

   where *equilMultiplier* is a user-supplied parameter (set to 2 in our implementation).

3. At each temperature, the simulation must proceed long enough for the system to reach a steady-state, also called the equilibrium state. In this context, equilibrium is defined as the establishment of a steady-state probability distribution of the generated states. The equilibrium condition ensures that the search has sufficiently explored the space of possibilities, before decrementing the temperature.

4. If the count on the inside bucket exceeds its limit before the outside count, then done at this temperature.
5. If the count on the outside bucket exceeds its limit first, then reset the counts and try again.

For further details of the HRSV schedule see [Huang 86].

We also tried a cooling schedule proposed by Swartz et al. [Swartz 90] which was derived from work done by Lam [Lam 88]. The final results however were not as good as those obtained using the HRSV schedule and this cooling schedule was discarded.

In addition to the cooling schedule there is the issue of selecting a freezing condition. Since the annealing temperature will take infinite time to reach a temperature of zero, we need to pick a condition to decide when the optimization should be deemed to be frozen, and thus complete.

We use a simple freezing criterion where we look at the acceptance rate. If the acceptance rate is below *freezeAcceptanceRatio* then we look at the mean and standard deviation of the final costs for the last *freezeNumTemps* temperatures. The problem is considered frozen if:

$$\frac{\sigma}{\mu} < \varepsilon \qquad \textbf{(EQ 30)}$$

where ε (*freezeEpsilon*) is a user-specified parameter. The cooling schedule and freezing parameters used for all the optimizations are tabulated in Table 3.

Name of parameter	Parameter value used	Comments
acceptRatio (Equation 26)	0.85	Increasing acceptRatio results in higher starting temperature. The acceptRatio must be chosen to be less that 1.
lambda (Equation 27)	0.7	Increasing lambda causes slower cooling of the problem. lambda must be greater than 0.
r (Equation 28)	0.5	Higher r causes slower cooling of the problem. r (and Equation 28) usually only get used at low temperatures.
minRatio (Equation 29)	0.5	Higher minRatio causes slower cooling of the problem.

TABLE 3. Cooling schedule and freezing parameters for the HRSV cooling schedule.

NOTE: 'N' is the length of the search vector.

Name of parameter	Parameter value used	Comments
maxTriesPerTemp	100*N	Simulating of more candidate vectors. This parameter is usually set to a high value and the number of simulations at any temperature rarely reaches this value. The annealing is therefore not sensitive to this parameter (if it is set to a high value). It is used to ensure that the annealing does not go on forever.
minMovesPerTemp	2*N	Increasing minMovesPerTemp causes quicker transitions from one temperature to another (which may also result in poorer results).
maxTriesBeforeEquil	20*N	Higher maxTriesBeforeEquil causes slower transition from one temperature to another, since the annealing waits for longer to get enough accepted moves.
equilMultiplier	2	Higher equilMultiplier results in the examining of more candidates at each temperature. If set too high, it can prolong the reaching of the equilibrium condition.
freezeAcceptanceRatio	0.6	Increasing freezeAcceptanceRatio causes earlier freezing of the problem.
freezeNumTemps	3	Higher freezeNumTemps delays freezing of the problem since the annealing waits to satisfy Equation 30 for a greater number of consecutive temperatures.
freezeEpsilon (Equation 30)	0.001	Higher freezeEpsilon causes earlier freezing of the problem.

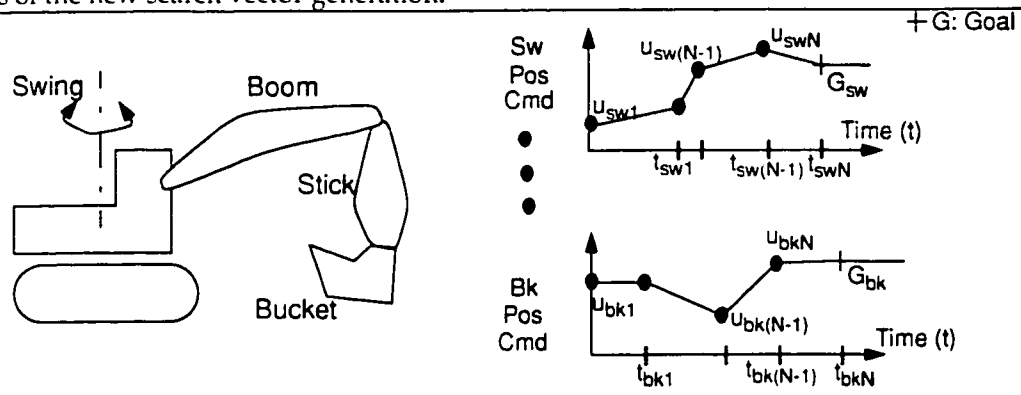
TABLE 3. Cooling schedule and freezing parameters for the HRSV cooling schedule.

NOTE: 'N' is the length of the search vector.

4.4.2 : Move set

One of the most important pieces of the SA algorithm is the function that generates new search vector candidates. Each candidate vector generated is evaluated (using the robot model) and Equation 25 is

used to determine whether to select the candidate or discard it. In this sub-section we discuss the details of the new search vector generation.



$$srchVec = \left[u_{sw1} \cdots u_{swN}\ t_{sw1} \cdots t_{swN}\ u_{bm1} \cdots u_{bmN}\ t_{bm1} \cdots t_{bmN}\ u_{st1} \cdots t_{stN}\ u_{bk1} \cdots t_{bkN}\right]^T$$

NOTE: Each u value represents a joint command value.

FIGURE 36. Description of the search vector (Reproduction of Figure 32)

A new search vector is generated by perturbing some elements of the current search vector. The two steps in this process: selecting which elements to perturb, and how to perturb them, are described below.

Selecting the search vector elements to perturb: There are a number of ways to select the elements-to-be-perturbed. These different ways are called move-sets. The move-sets guide the search by offering it the choice of certain directions that may yield near-optimal results much faster than an unguided approach. For instance, a move-set used for our problem perturbs a subset of elements u_{sw1} through u_{swN} and t_{sw1} through t_{swN} (See Figure 36). If such a move-set was used, the search would then be modifying only the swing command profile of the current search vector by perturbing the associated knot points. Using this move-set would be appropriate for a *task* (or part of a *task*) that only requires Sw joint motion. In the absence of the guidance provided by this move-set, all the elements of the search vector will be treated as being equally important and the search would needlessly spend time perturbing the elements corresponding to the Bm, St, and Bk joints (Note the use of a move-set does not restrict the search by disallowing the exploration of other combinations of joint motions; it only offers it some suggestions on possible search directions).

We use 46 different move-sets for our problem. The design of the move-sets was based on two observations:

- In certain situations the search could get trapped if it only searched along one joint direction. For instance, consider a situation where the robot has to move the Bm and Sw joints in a coordinated fashion to avoid an obstacle. In this case, a move-set that only allows perturbation of the Bm and Sw search vector elements is more efficient at finding the best coordination, than a broad approach that perturbs all joints, since the space of possibilities in the latter case is larger. We do not disallow the exploration of other move-sets that perturb a different combination of joints, which may potentially yield a better solution. We continuously examine the different move-sets to select the one that gives us the best result.

  Our move-sets are selected to allow the perturbation of different sets of joints.Since our HEX robot has 4 joints, there are (2^4-1=15) joint permutations that perturb at least one joint. The possible permutations are listed in Table 4.

Permutation #	Perturb Sw?	Perturb Bm?	Perturb St?	Perturb Bk?
1	N	N	N	Y
2	N	N	Y	N
3	N	N	Y	Y
4	N	Y	N	N
5	N	Y	N	Y
6	N	Y	Y	N
7	N	Y	Y	Y
8	Y	N	N	N
9	Y	N	N	Y
10	Y	N	Y	N
11	Y	N	Y	Y
12	Y	Y	N	N
13	Y	Y	N	Y
14	Y	Y	Y	N
15	Y	Y	Y	Y

TABLE 4. Possible HEX joint permutations that perturb at least one joint.

- The second observation is that the search is more efficient if the size of the perturbations decreases with temperature. At low temperatures, the SA algorithm is more reluctant to move to a higher cost option. Therefore, it is inefficient to generate large perturbations since they result in a higher cost candidate and are therefore not accepted. It is possible to use a fixed decrement in perturbation size with temperature. However, based on our experience, it is difficult to determine a good decrement

rate. If the decrease is too quick, the search gets "walled in" too quickly; if it is too slow, the search is inefficient.

To counter this problem, we allow the search to select the perturbation radius by having three classes of move-sets with three different radii, i.e. three levels of perturbation magnitudes. Each level has an associated *perturbRadFactor* (Perturbation Radius Factor) of either 0.1, 0.6 or 1.0. These values were chosen to provide enough equally-spaced levels of perturbation radii. The *perturbRadFactor* is used when perturbing the search vector elements (explained in the next sub-section).

We therefore have 45 move-sets, as described above: three classes of different perturbation radii with 15 move-sets each. In addition, we have one other move-set (described in the flowchart in Figure 40) which creates a new vector that (approximately) points in the direction of the minimum cost search vector seen until then in the search. Figure 37 summarizes all the different move-sets used.

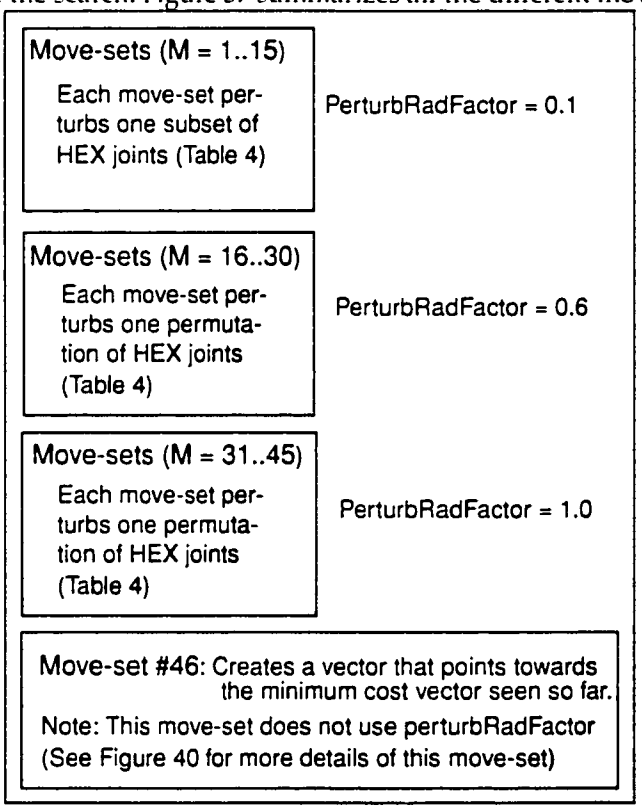


FIGURE 37. The list of all 46 move-sets. For a description of how perturbRadFactor is used, ref Equation 31, Equation 32. A move-set M is selected at the start of each temperature.

A scheme proposed by Ochotta [Ochotta 94] is used to select a move-set at each temperature. At the start of a temperature, each move-set is used to generate 10 new search vectors, and the statistics are compiled for the costs of the new vectors generated. The most promising (lowest mean cost) move-set is selected for use at that temperature; we do not change the move-set during a temperature. By re-doing this selection at each temperature, we allow the use of different exploration strategies at each temperature.

Once a move-set has been selected, it defines the search vector elements that can be perturbed (by defining the joint whose elements can be perturbed), and the perturbation radius (*perturbRadFactor*). We do not perturb all the elements that can be perturbed. The following procedure is used to select the elements (corresponding to a particular joint) that are perturbed (Ref flowchart in Figure 38).

- Generate a random integer (R_2) between 1 and 2*N (N = Number of knot points used in defining the search vector). Recall that a search vector has 2*N elements that correspond to each joint (Ref Figure 36). R_2 is the number of elements that will be perturbed.
- Generate R_2 unique integer values between 1 and 2*N. Add an offset to these values based on which joint's elements are being perturbed: 0 if it is the Sw, 2*N for the Bm, 4*N for the St, and 6*N for the Bk (Recall that the first 2*N elements of the search vector define the Sw command, the next 2*N define the Bm, while the St and Bk follow, in that order).

  These R_2 integers define the indices of the elements that will be perturbed.
- Perturb the selected elements: This is described in the following section.

Perturbing the selected elements: Once the elements to be perturbed have been selected (and the perturbation radius factor is known), each of those elements ($e_{i,current}$) are perturbed to generate the new value ($e_{i,new}$) as follows. (This is also described as flowchart in Figure 39).

- Generate a random real number (R_3) between -1 and 1
- If $r<0$ then

$$e_{i,new} = e_{i,current} + perturbRadFactor \cdot R_3 \cdot (e_{i,current} - e_{i,MIN}) \quad \text{(EQ 31)}$$

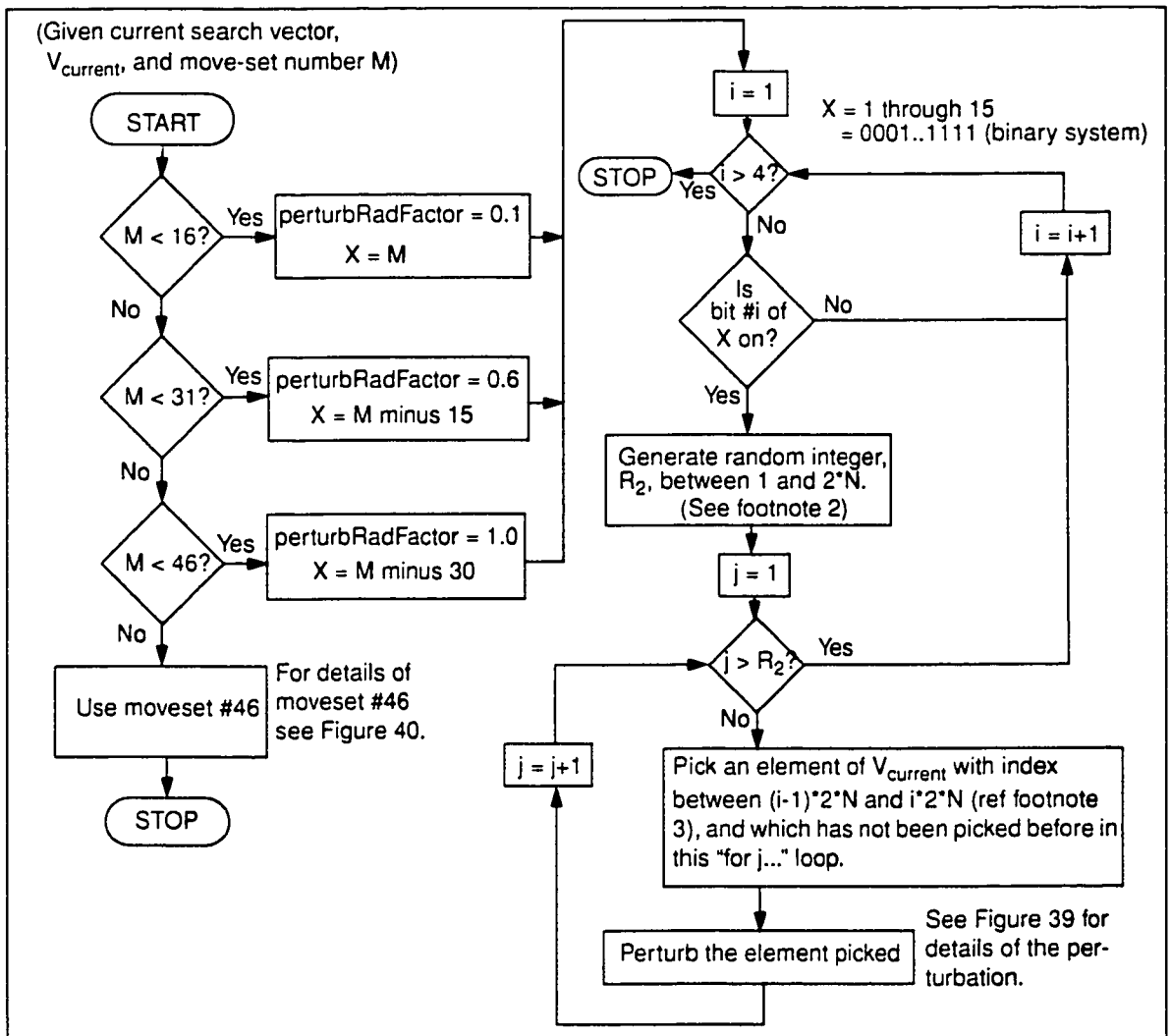
else

$$e_{i,new} = e_{i,current} + perturbRadFactor \cdot R_3 \cdot (e_{i,MAX} - e_{i,current}) \quad \text{(EQ 32)}$$

In the above equations, $e_{i,MIN}$ is the minimum value that element e_i can take, and $e_{i,MAX}$ is the maximum allowable value for e_i. For instance, $e_{i,MIN}$ and $e_{i,MAX}$ could be the joint limits if e_i is a joint command (one of the u_i values in the search vector; ref Figure 36). If it is a time value (one of the t_i parameters of the search vector) then it is limited to having a value between t_{i-1} and t_{i+1} (to ensure that the time values are monotonically increasing). We enforce a minimum spacing of 0.1 secs

between the time values to disallow discontinuities in the command profile, which will occur if two knot points with different u values have the same t values. t_N can have a maximum value of 20 secs; t_1 can have a minimum value of 0.1 secs. The value of 50 secs was a very generous estimate of the maximum time that any of our tasks could take. The selection of 0.1 secs as the minimum time spacing was arbitrary.

The flow-charts below illustrate the creation of a new search vector using the currently selected move-set M. The move-set selection is done at the start of each temperature of the SA optimization.



NOTE: (1) All random numbers are picked from a uniform distribution.

(2) There are 2*N elements in $V_{current}$ which define the commands for any joint. R2 is the number those elements that will be perturbed.

(3) We pick an element of $V_{current}$ between these two indices since the elements of $V_{current}$ that correspond to joint i lie between them (i=1 is the Sw, 2 is the Bm, 3 is the St, 4 is the Bk).

FIGURE 38. Flow-chart showing the generation of a new search vector using move-set **#M**.

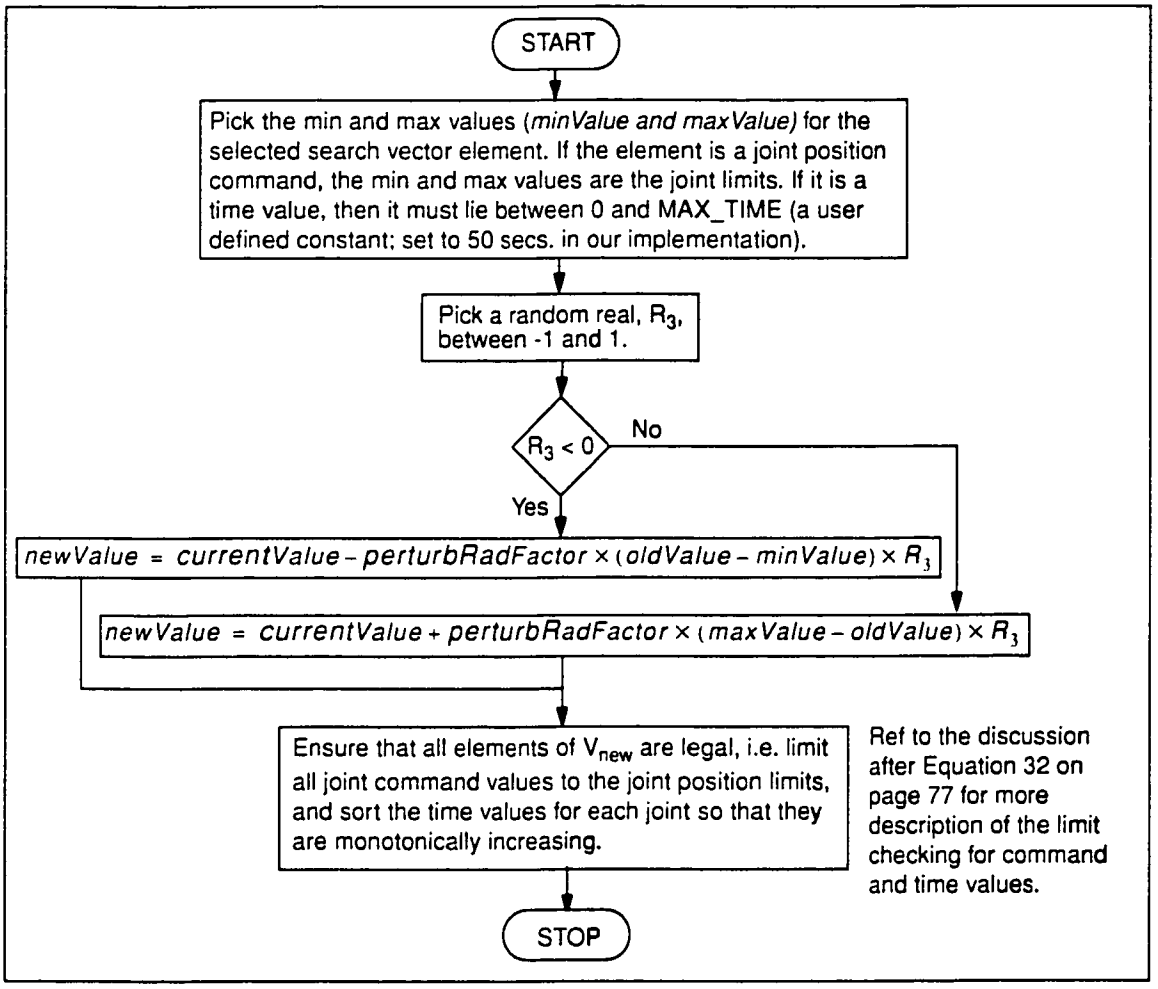


FIGURE 39. Details of how the perturbation of a search vector element is done.

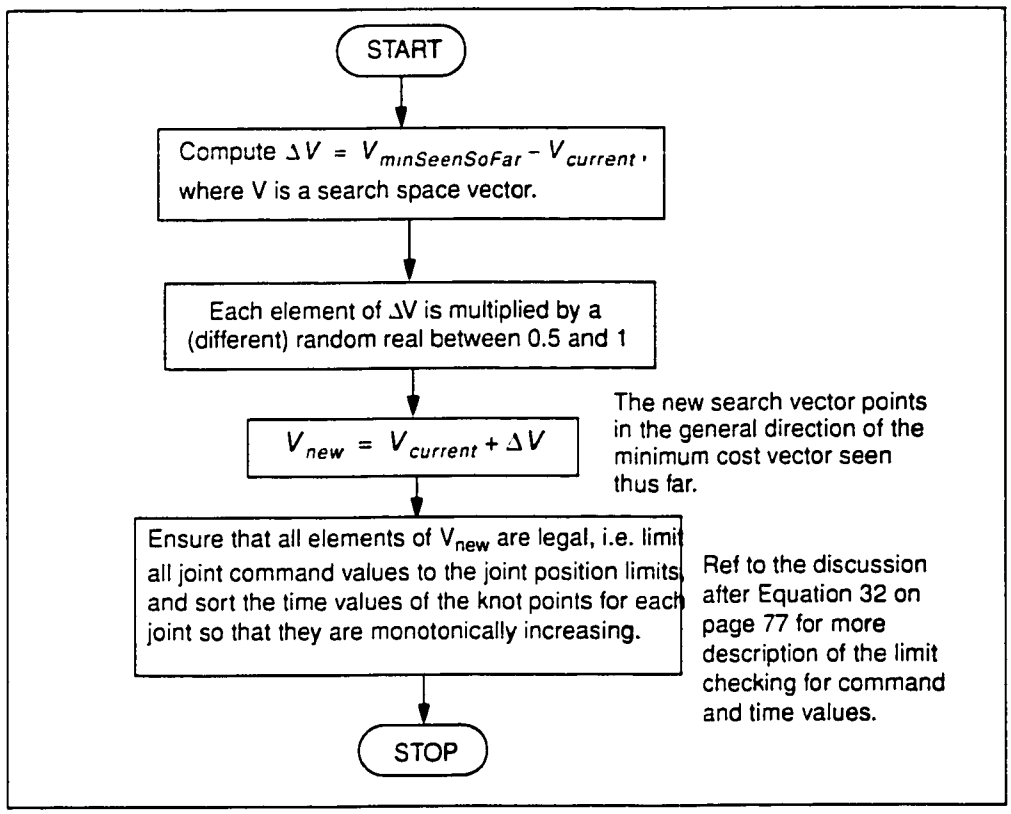


FIGURE 40. Details of move-set #46, which creates a new vector that points in the direction of the minimum cost vector seen thus far in the optimization.

Section 4.5: Implementation details

There are few details specific to our implementation of the SA algorithm for optimal motion planning. These are discussed below.

4.5.1 : Knot point selection

There is no easy method of selecting the number of knot points to use to construct the search vector. Using too many knot points can results in an rather long search vector, which increases the size of the

search space (See Equation 15 for the relation between the size of the search vector and the number of knot points). If the number of knot points is too small, the search vector may not have the degrees-of-freedom necessary to approximate the optimal command curve.

One promising way to address the problem is by dynamically adding or removing knot points as the search proceeds. This allows the search vector length to be part of the search process. In this work we have not used this approach due to lack of time. Future work may explore this approach.

Another way of selecting the desired discretization is by studying the standard deviation of the cost over multiple optimization runs for the different knot point cases. The standard deviation is usually minimum for a certain discretization, which is also the discretization that usually has the lowest mean cost values. This is discussed further in the next chapter (Chapter 5, page 122).

All of the above discussion assumes that the command discretization for each joint is the same, i.e. the same number of knot points are used to describe each joint command curve. This may not be the best approach since depending on the amount of joint motion involved, some joints may require more (or less) knot points to describe their joint command curves.

We have not implemented the non-uniform knot point usage for the different joints; our implementation requires that each joint use the same number of knot points. We also do not allow the search vector length to vary during an optimization due to the difficulty with deciding when to add or delete a knot point, and which one to add or delete.

We run our optimization for 5 different cases, with each case using between 1 and 5 knot points per joint. We then select the best solution from the 5 cases. Each case was run in parallel on 3 different computers since the SA algorithm is a stochastic search method and requires each optimization to be run multiple times. Thus, for each task, the optimization was run 15 times.

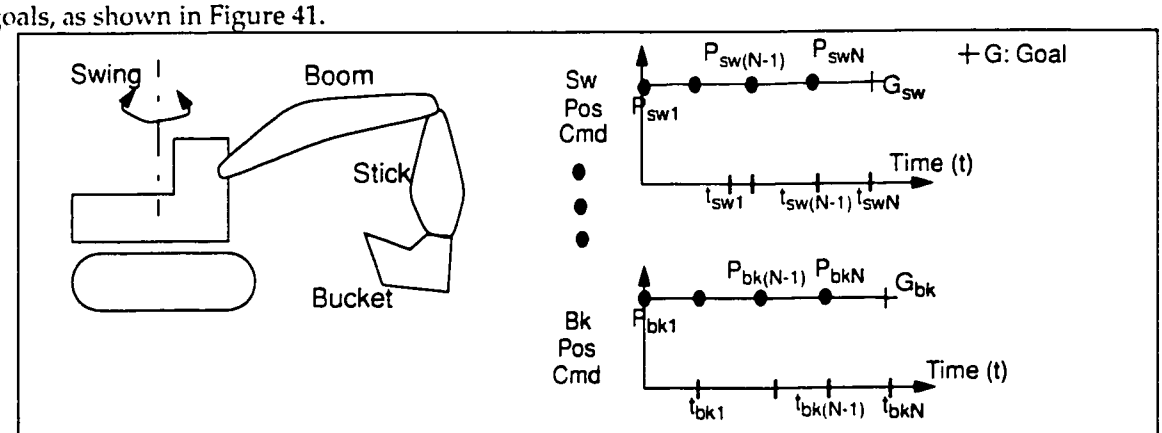
The choice of using 5 cases, with 1 through 5 knot points per joint, was based on the observation that this set usually provided a good result, for the computation time devoted to it. We used the same annealing parameters for all the 5 cases (as a function of the number of knot points). We tried to limit the computation time to a few hours, and hence the choice of the annealing parameters.

Better results can be obtained for a higher number of knot points by changing the annealing parameters so that the search simmers for a longer duration at each temperature, thus performing a wider search. However, doing so requires more computation time.

The choice of running 3 instances of each of the 5 cases for each task was a function of the available time and computing resources.

4.5.2 : Selection of seed search vector

The *seed search vector* for all cases was constructed by setting all the joint knot points to the final joint goals, as shown in Figure 41.



$$seedSrchVec = \left[G_{sw} \cdots G_{sw}\ t_{sw1} \cdots t_{swN}\ G_{bm} \cdots G_{bm}\ t_{bm1} \cdots t_{bmN}\ G_{st} \cdots t_{stN}\ G_{bk} \cdots t_{bkN}\right]^T$$

FIGURE 41. Initial *seed search vector* construction.

The joint knot points are position commands to the joint position controllers. Thus, the initial search vector would result in moving the HEX towards the goal. This is not always the best solution, especially if an obstacle lies on the way from the start to the goal. It also does not offer the gains from coordinating the different joints to exploit the dynamic and hydraulic system characteristics.

4.5.3 : Random number selection

All random numbers selected in different parts of the annealing algorithm, whether integer or real, are drawn from a uniform distribution over the selected intervals.

Section 4.6: Concluding remarks

In this chapter we have outlined our approach to optimal motion computation for hydraulic robots, We use the robot model developed in Chapter 3 in a command space search which uses the Simulated Annealing algorithm. We discretize the command space of the robot and search in the command space for the control (temporal sequence of commands) that minimizes a user-defined objective function,

while performing a specified *task* (with specified start and end states) and satisfying the kinematic and dynamic constraints.

The advantages of using a command space search is that all the candidates in the command space are feasible. This is in contrast to searching in the space of desired spatial paths and trajectories, where many of the candidates are infeasible.

We use a novel search vector construction technique that guarantees that all candidate search vectors will take the robot to the goal (if the joint position controllers do not fail). Therefore, the search vector construction itself automatically prunes motions that do not reach the goal, thus making the search more efficient.

The cost function used for our optimization is a weighted linear combination of different cost terms. It is flexible enough to allow the introduction of any quantifiable measure in the cost function. For instance, it is possible to compute optimal motions that minimize the force load or usage of a joint. We use additional terms in the cost functions for spillage or collisions with obstacles. These terms are weighted heavily in order to penalize candidate vectors that cause them. The minimization automatically avoids such candidates due to the high cost associated with them.

We impose the kinematic and dynamic constraints on the candidate command vectors through the robot model by simulating the robot's response to the commands. The simulated response is used to compute the cost of the candidate command vector.

We use the above described approach to compute optimal motions for a number of HEX robot *tasks*. The resulting optimal motions, and the analysis of the results, are presented in the following chapter, Chapter 5.

CHAPTER 5

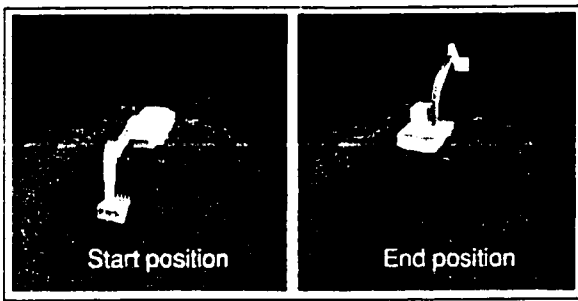
Results

The optimal motion planning approach detailed in the previous chapter was used to compute the optimal motions for a HEX placed in a number of different *loading configurations*; a *loading configuration* being an arrangement of HEX, truck, and terrain.

For a description of the *loading cycle* see Figure 1 and Figure 2 in Chapter 1. In this work we have only focused on the free-space motion of the HEX, not on the digging sections of the *loading pass*. Thus, a typical *task* for our application is one where a loaded bucket has to be emptied in a truck, or an empty bucket must return from the truck to the dig face. Both these do not involve any soil-tool interaction, except for the unloading of the soil in the truck. The load in the bucket for all the *tasks* was assumed to be 35000N (loaded bucket) or zero (empty bucket).

We computed minimum-time and minimum-energy motions for 7 *tasks*. The choice between time and energy was made by setting the appropriate weights in the cost function (see Equation 17, Chapter 4). Each *task* is defined by the following parameters.

- Start and end locations of the Sw, Bm, St, and Bk.
- Load in the bucket.
- Location of truck: Used to detect spillage, if the bucket is loaded.
- Map of the environment around the HEX: Used to define the obstacles around the HEX.



Start and end HEX positions for *configuration* #1

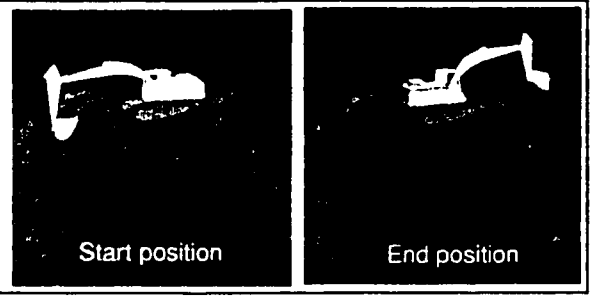
- *Task* #1: Start -> End (Loaded bucket)
- *Task* #2: End -> Start (Empty bucket)

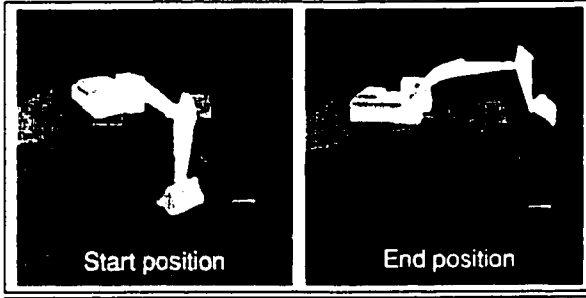
Both *tasks* performed in *truck-loading mode.*

Start and end HEX positions for *configuration* #2

- *Task* #3: Start -> End (Loaded bucket)
- *Task* #4: End -> Start (Empty bucket)

Both *tasks* performed in *fine-finish mode.*





Start and end HEX positions for *configuration* #3

- *Task* #5: Start -> End (Loaded bucket)
- *Task* #6: End -> Start (Empty bucket)

Both *tasks* performed in *truck-loading mode.*

Start and end HEX positions for *configuration* #4

- *Task* #7: Start -> End (Empty bucket)

This *task* performed in *fine-finish mode.*

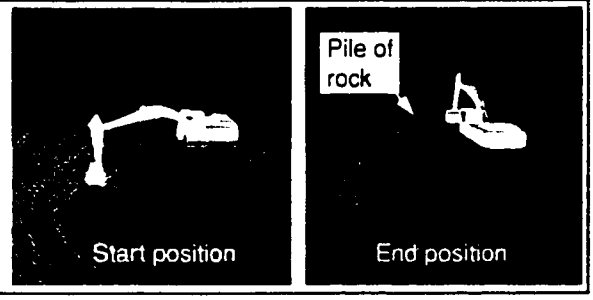


FIGURE 42. Illustrations showing the *loading configurations* and *tasks.*

Figure 42 shows the four different *loading configurations* used for the 7 *tasks*:

- *configuration* #1 for *tasks* 1 and 2,
- *configuration* #2 for *tasks* 3 and 4,
- *configuration* #3 for *tasks* 5 and 6, and
- *configuration #4 for task* 7.

For *configurations* 1, 2, and 3 the two associated *tasks* are:

- One starting from the start position with a loaded bucket and moving it to the end position (over the truck), while avoiding collisions with the environment, and avoiding spillage of the bucket load outside the truck.
- The other moving the (now) empty bucket back to the start position, while still avoiding collisions. Spillage is not an issue for this *task* since the bucket is empty.

These three *configurations* were chosen to obtain a rich set of HEX *truck-loading* arrangements. To obtain some diversity in the HEX operating modes, the tasks in *configurations* #1 and #3 were performed in the *truck-loading mode*, and tasks in *configurations* #2 and #4 were performed in the *fine-finish mode*. We chose *configuration* #4 (and the associated task #7) to demonstrate the obstacle avoidance capabilities of the optimizer, since it involved avoiding a pile of rock placed between the start and the goal positions of the HEX.

The following table summarizes the above information.

Loading Configuration	*Tasks* in this configuration	HEX operating mode
1	1, 2	TL
2	3, 4	FF
3	5, 6	TL
4	7	FF

TABLE 5. Summary of *tests* conducted.

The steps used to compute the (time or energy) optimal motions for each *task* are listed below.

1. A complete (360° of azimuth) laser scan was taken of the area surrounding the HEX after it was positioned in the desired arrangement relative to the truck. A scanner mounted on the HEX testbed was used to acquire the scan.
2. The start and end HEX locations were identified. (More on this in Section 5.1)

3. The following parameters were input to the optimization:
   - Complete 360° azimuth laser scan: This is used to construct a triangulated mesh, which is then used for collision avoidance.
   - The truck location: This allows computation of the *spillage* cost, i.e. if the bucket is loaded and spillage is to be checked, then spillage is assumed to occur if the bucket angle is greater than the start bucket angle at any time during the motion, when the bucket is not over the truck bed.

     We manually identified the corners of the truck bed from the laser scan, and input their coordinates to the optimization.
   - The start and end HEX positions.
   - *Seed search vector*: This is used to provide a starting point for the optimization. The Simulated Annealing algorithm is fairly insensitive to the choice of the initial search vector. In the early stages of the optimization when the temperature is high, the algorithm allows an expansive exploration of the search space by accepting almost all moves generated as candidates, even if they increase the cost significantly.
   - Spillage term: Used to specify whether the cost function should contain spillage terms.
4. A time or energy minimization was performed for the defined task. The output of the minimization is a vector of joint commands (See Figure 36, Chapter 4) and a simulated HEX response to those joint commands (as predicted by the robot model used in the optimization). We refer to this simulated response as the *simulated optimal motion* response.

Section 5.1 shows the results from the time-optimal tests while Section 5.3 shows the results from the energy-optimal tests. An interesting result from the time-optimal tests is described in Section 5.2. All the results are compiled and tabulated in Section 5.4. Section 5.5 gives a description of the computation time used for planning optimal motions for the 7 *tasks*. Section 5.6 describes the results from some cost variation studies, while Section 5.7 describes a possible use of our optimal motion planning approach: as a machine designer's aid. We conclude the chapter with a short summary.

Section 5.1: Computing minimal-time motions

In this set of tests, we computed the minimum-time motions for the 7 different HEX *tasks*. Since there is no way to determine how well our computed motions compare to the globally optimal motion, we compare the motions to the next best alternative at our disposal: the HEX motions when operated by a human expert.

We compare the human expert's HEX motions to the time-optimal motions (computed for the same loading configuration) since human experts tend to primarily optimize time, while performing the task smoothly. The human expert comparisons were performed for *loading configurations* 1, 2 (*tasks* 1, 2, 3, 4). Each comparison used data collected from our Caterpillar 325A HEX testbed while a human expert performed the *truck-loading* operation as fast as possible. He was allowed a few trials to familiarize himself with each *loading configuration*. The following variables were recorded during the human expert's operation:

- 360^0 range scan (before start of operation)
- Joint angles,
- Joint velocities,
- Hydraulic cylinder pressures,

In addition to the human expert comparison, we compared our results for *loading configuration* #3 (*tasks* 5, 6) to the results obtained by Rowe et al. [Rowe 99] for a truck loading operation. We recorded the same set of variables as for the human expert comparison, including a 360° laser range scan.

To ensure that the start and end locations for the human expert (and Rowe's method) and our optimal planning system were the same for each *task*, we analyzed the motions recorded from the expert's operation (and Rowe's method); these motions are referred to as *comparison motions*. The analysis of the *comparison motions* for each *task* allowed us to identify the HEX joint locations immediately after completion of the dig, which we then defined as the start position for the *task*. The joint locations after the unloading of the bucket in the truck defined the end location. There is a natural short pause (a second or less) in the *comparison motions* as soon as the digging is done, as well as after the unloading of the bucket. This made it easy to pick the start and end locations.

Table 6 summarizes the *task* information thus far.

Loading Configuration	*Tasks* in this configuration	Compare to....	HEX operating mode
1	1, 2	Human expert	TL
2	3, 4	Human expert	FF
3	5, 6	Rowe's method	TL
4	7	None	FF

TABLE 6. Summary of tests conducted.

The time-optimal motion results obtained for all 7 *tasks* were implemented on the HEX testbed to prove that the optimal motions were indeed as fast as the motion planning system predicted. This is

done by sending the vector of joint commands (obtained as a result from the optimal motion computation) to the closed-loop HEX testbed, and collecting the response data. This response is referred to as *real optimal motion* response.

In the result plots (later in this section), we plot the *simulated* and *real optimal motions* along with the *comparison motions*. The results for each loading configuration (and *task*) follow.

5.1.1 : Loading Configuration #1 (Tasks #1,2)

The loading configuration is illustrated in Figure 43. The images in the figure were taken from a graphical simulator. The HEX and truck are synthetic models while the terrain, visible in brown, is a triangulated mesh created from an actual laser range scan of the terrain around the HEX. The scan was taken just before the human expert's operation, using a laser scanner mounted on the Caterpillar 325A testbed. Although the dig face changed during the operation, due to the HEX digging in the dig face, the

change was quite small and was ignored, i.e. we assumed that the terrain stayed unchanged between the taking of the scan and the free-space motion task.

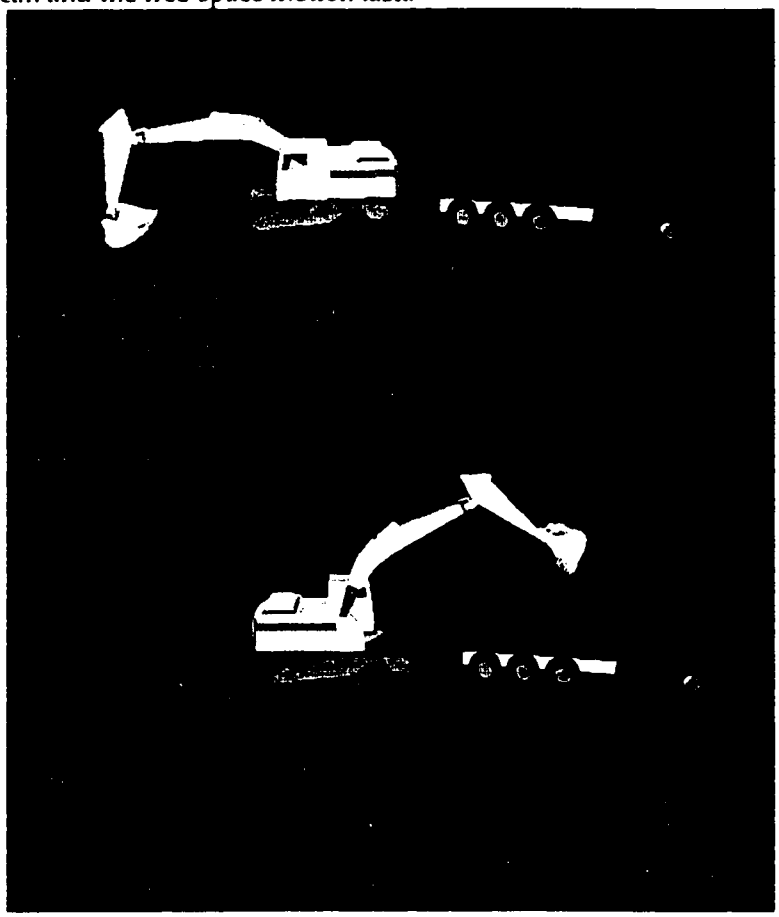


FIGURE 43. Loading configuration #1 (a) start and (b) final HEX positions. These images were taken from a simulator. The terrain shown was constructed using a laser range scan of the actual terrain.

The results from the optimization are plotted along with the expert's motions on Figure 44, Figure 45.

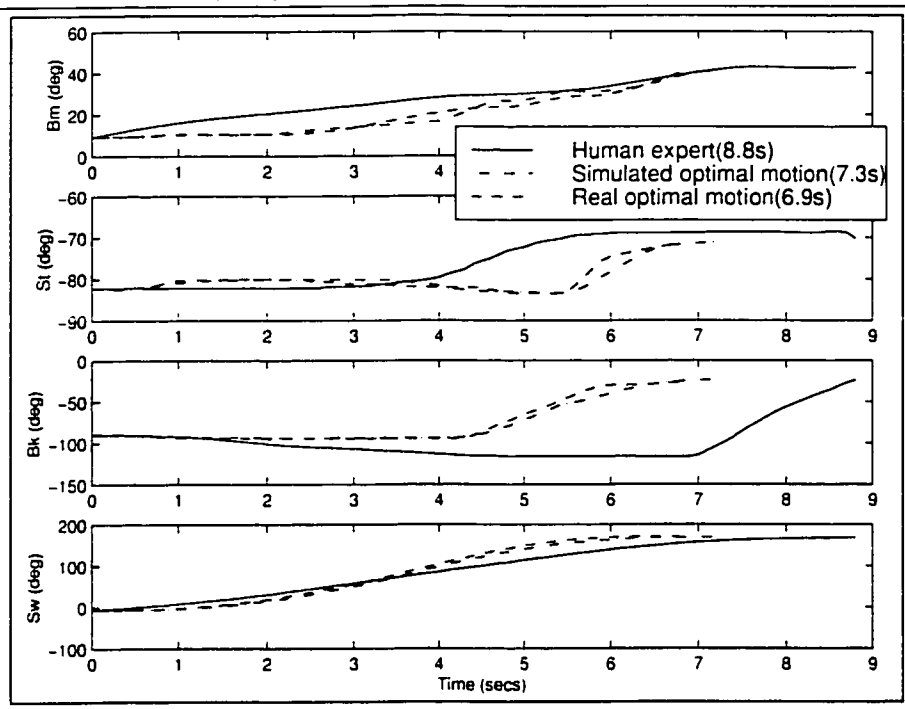


FIGURE 44. Task #1 (Loading configuration #1) - TL mode - loaded bucket.

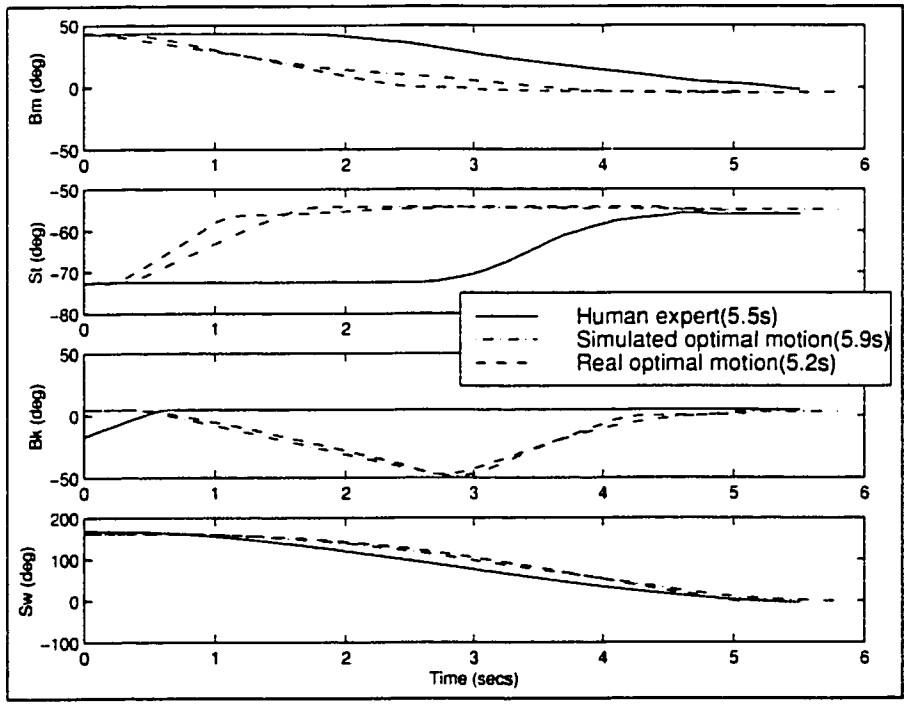


FIGURE 45. Task #2 (Loading configuration #1) - TL mode - empty bucket.

From Figure 44 we can see that the optimal motion computed for *task* #1 was 1.5 secs faster than the motion demonstrated by the human expert. This task involved moving from the start position over the dig face (with a loaded bucket) to the final position over the truck. The legend in the plots shows the time taken for each of the traces. For instance, the time taken by the expert in Figure 44 was 8.8 secs. The *simulated optimal motion* for the same task took 7.3 secs, and the *real optimal motion* took 6.9 secs. The prediction of the model appears to be fairly accurate in these tests. The slightly shorter time taken by the *real optimal motion* (as compared to the *simulated optimal motion*) is largely due to the error in predicting the swing deceleration in the *truck-loading mode* model.

The expert performed 6 loading passes during the recording of the comparison data. The plots above show his third loading pass which was one of his fastest loading passes. We expect that the expert was comfortable with the flow of the truck loading operation by then, but was not yet tired (or bored).

The above comparison shows that the time taken by the human expert and the computed optimal motions are very comparable. We can only speculate why the computed motions are faster than the expert's motions in *task* #1. The difference is partly due to the human expert's larger margin-of-safety, i.e. he stays further away from the truck when approaching it to unload the bucket. To achieve this he raises the boom much earlier than in the case of the optimal motion. The lower swing speed in the expert's motions also appear to have contributed to the difference. The lower swing speed could have been due to different swing command modulation or different joint coordination, e.g using multiple joints at the same time may have reduced the peak swing acceleration.

There is also the issue of variation in the operation of the particular expert, i.e. it is possible that over a larger number of *loading passes*, he may do as well as the computed optimal. We did not attempt to make the expert perform many *loading passes* with the same start and end positions since that is rather difficult to orchestrate. While loading a truck, the expert digs in different locations on the dig face (to always get a full bucket load), and thus has a different start position for each *loading pass*. Forcing him to start and end in a marked location would require him to stop after each dig (i.e. after each time that the bucket was filled) in order to position the HEX at the designated start location. This would break the "flow" of the loading operation and would therefore be slower.

Finding the exact reasons for the differences requires more detailed analysis of the two motions, which is beyond the scope of this thesis. It should be noted that the expert used for our comparison is one of

the very best. However, most HEX operators are not as good; they usually operate at 60-80% the speed of an expert.

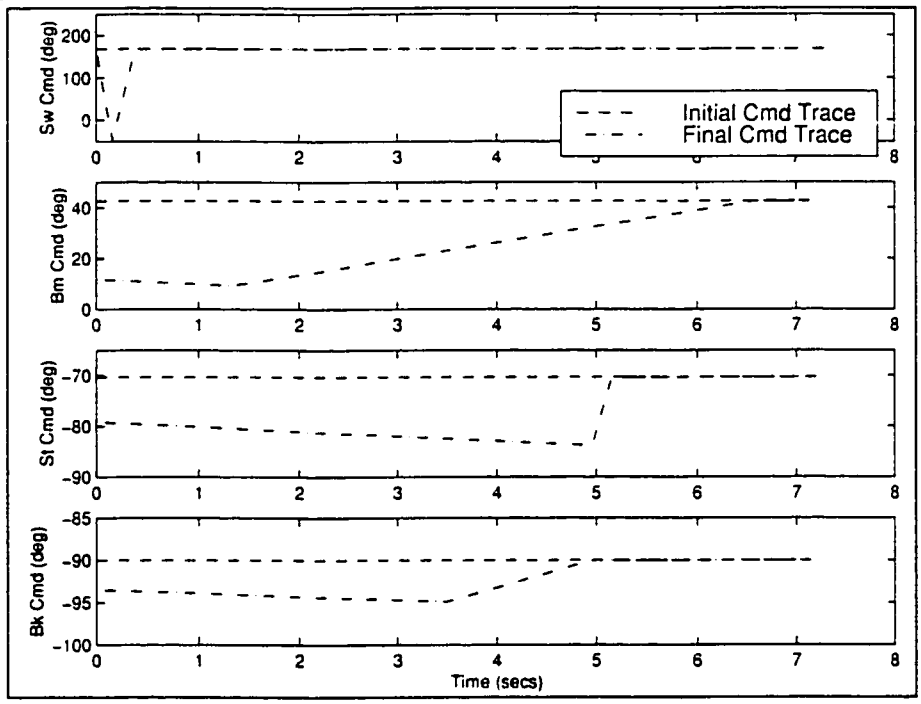


FIGURE 46. Joint commands corresponding to the *seed search vector* (Initial Cmd Trace) and the optimal search vector (Final Cmd Trace) for *task* #1. Two knot points are used to discretize the command trace for each joint.

Figure 46 and Figure 47 show the joint commands corresponding to the initial and final (time-optimal) search vectors for *tasks* #1 and #2. We present these to give the reader an idea of the change in the command traces from the start of the optimization to the end. Note that the knot points in the *seed vector* were simply set to the goal joint values.

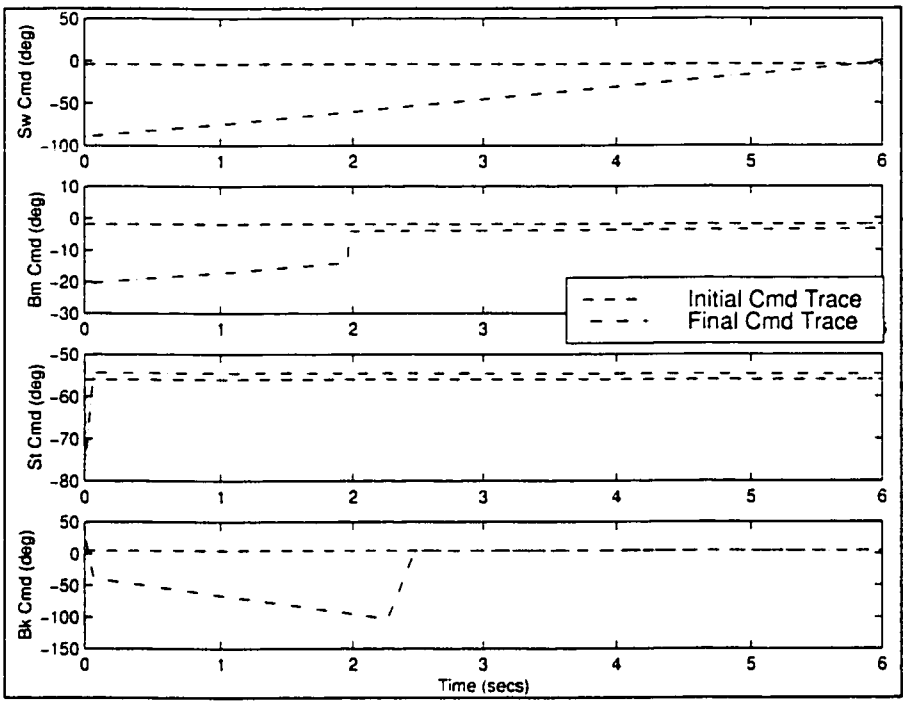


FIGURE 47. Joint commands corresponding to the *seed search vector* (Initial Cmd Trace) and the optimal search vector (Final Cmd Trace) for *task* #2. Three knot points are used to discretize the command trace for each joint.

5.1.2 : Loading configuration #2 (Tasks #3, 4)

Loading configuration #2 is illustrated in Figure 48. This loading arrangement was chosen since it forced the bucket to be raised up and over the front of the bench when moving from the start position to the truck. The dig face in Figure 48 is just below the bucket when the HEX is at the start position. The truck appears partially buried in the (brown) terrain because the truck is picked up in the laser scan. We do not perform any object recognition in the scanned data and therefore the truck is not very distinguishable in the mass of brown; everything in the terrain map is colored brown, since the HEX

must not collide with anything that appears in the laser scan. The synthetic truck model was added to clearly show the truck location.

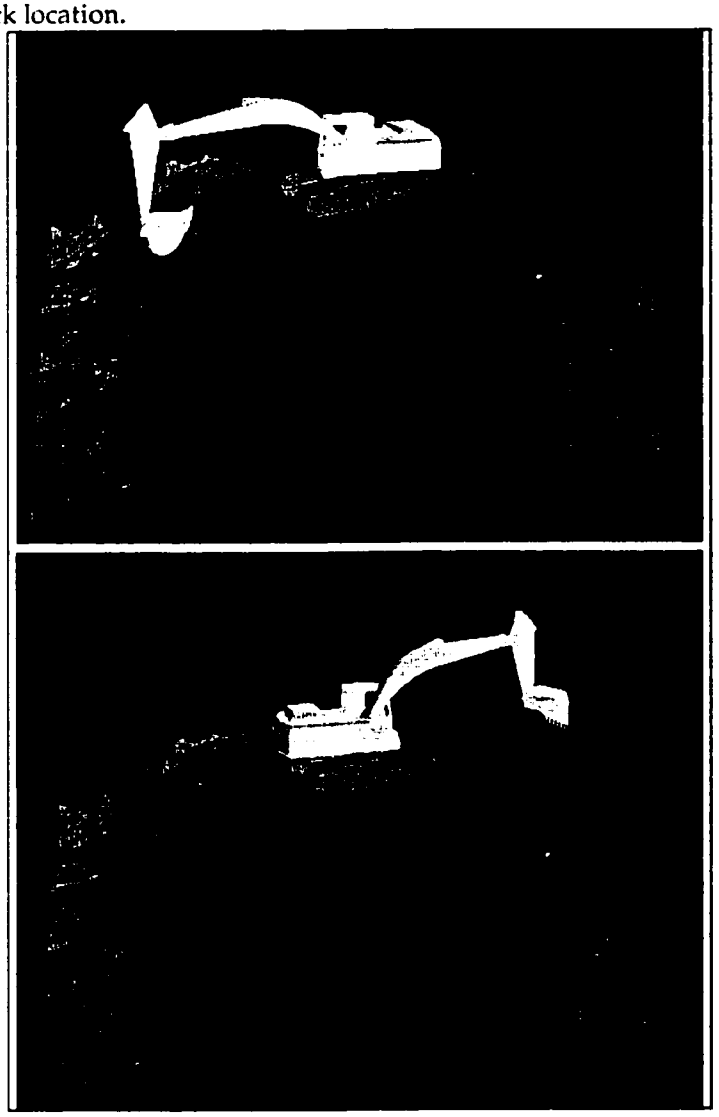


FIGURE 48. Loading configuration #2 (a) start and (b) finish HEX positions.

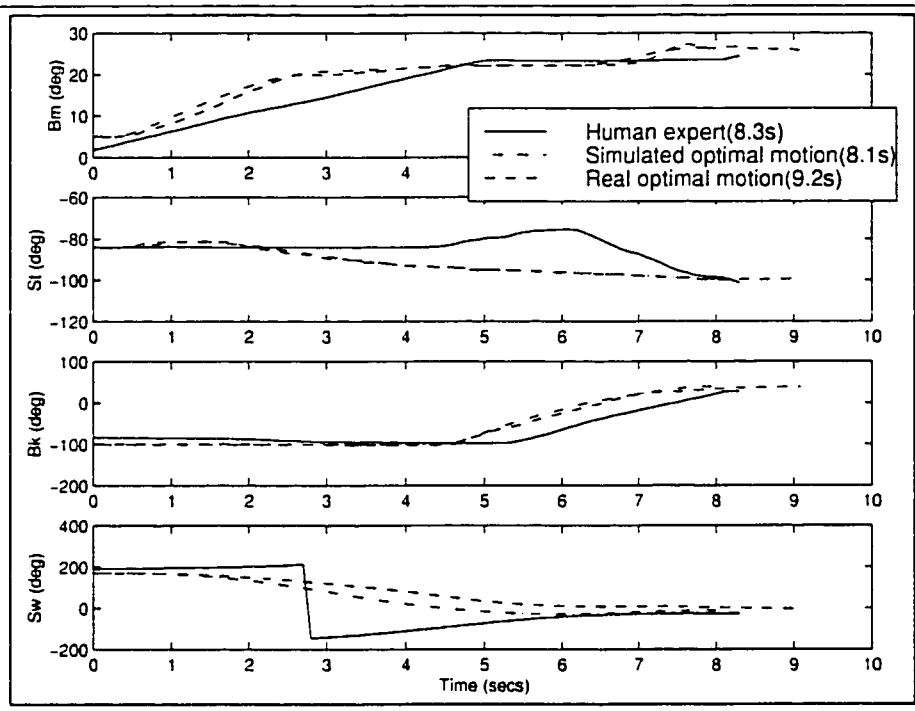


FIGURE 49. Task #3 (Loading configuration #2) - FF mode - loaded bucket.

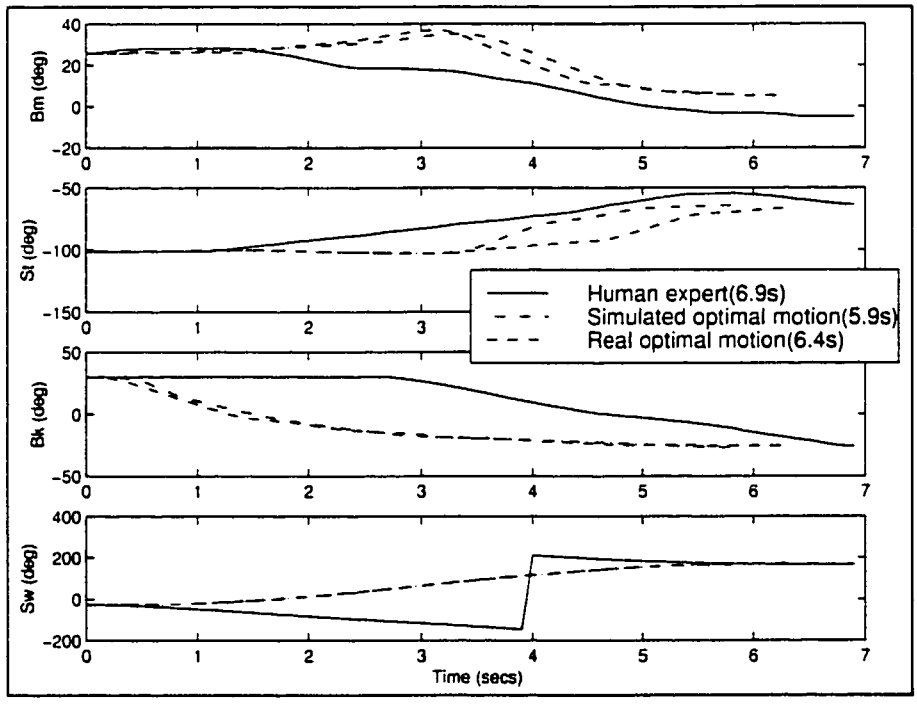


FIGURE 50. Task #4 (Loading configuration #2) - FF mode - empty bucket.

Task #3 shows the motion from the start position to the unloading position over the truck. *Task* #4 starts from the unloading position and returns to the dig face. The sudden jump in the swing joint angle (Figure 49, Figure 50) is not a physical leap but merely reflects the fact that the swing is a rotary joint, with an angular range from -180° to +180°. Thus, the step in the swing angle plot is when the swing wraps around from +180° to -180° (or vice-versa), although they are the same physical location.

In this set of motions, the human chose to swing in a positive direction in task #3 (Positive direction of the axis of rotation Z is upwards) while the computed optimal motion involved swinging the other way. Human operators prefer to rotate to their left (positive swing rotation) since the operator's cab is located to the left side of the machine. The operator has a large blind spot when rotating to the right since the implements (boom, stick and bucket) obstruct the view.

The optimal motions computed in this case are comparable to the expert's motions. The difference in the *real* and *simulated optimal motion* time values in *task* #3 is due to the inaccuracy in the Sw joint model (Figure 49). The error seems to be high only when the bucket is loaded; we expect that this is due to insufficient Sw data in that region of the input space (for the Sw joint response approximation). This can be corrected by the use of additional data in the *FF* mode model.

5.1.3 : Loading Configuration #3 (Tasks #5, 6)

This loading configuration is a one of the most preferred loading configurations for the HEX since it involves very short swing joint motions. The HEX starts from the location just above the dig face after capturing a bucketful of soil. Next, the boom and swing joints need to move in a coordinated fashion to clear the tailgate of the truck; if the swing is too fast and the boom has not raised enough, the HEX will hit the tailgate of the truck; if the swing is too slow, then it results in a longer cycle time.

The motion time for *tasks* #5 and 6 (Figure 52, Figure 53) again appear to be very comparable for our optimal motion planning method and Rowe's method. The results from Rowe's method were obtained after 9 trucks (54 bucket loads) had been loaded ([Rowe 99]). Rowe's method starts out with a sub-optimal loading strategy but improves with experience. It monitors the truck-loading performance from one loading pass to another and performs a local optimization to speed up the cycle times, as the loading proceeds.

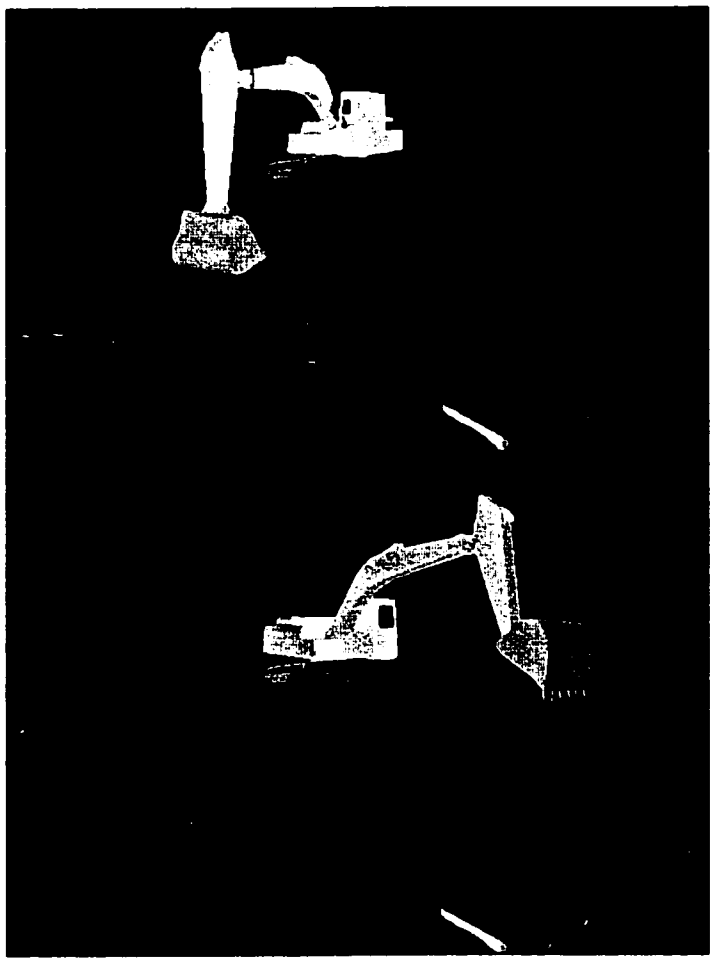


FIGURE 51. Loading configuration #3 (a) start and (b) finish HEX positions.

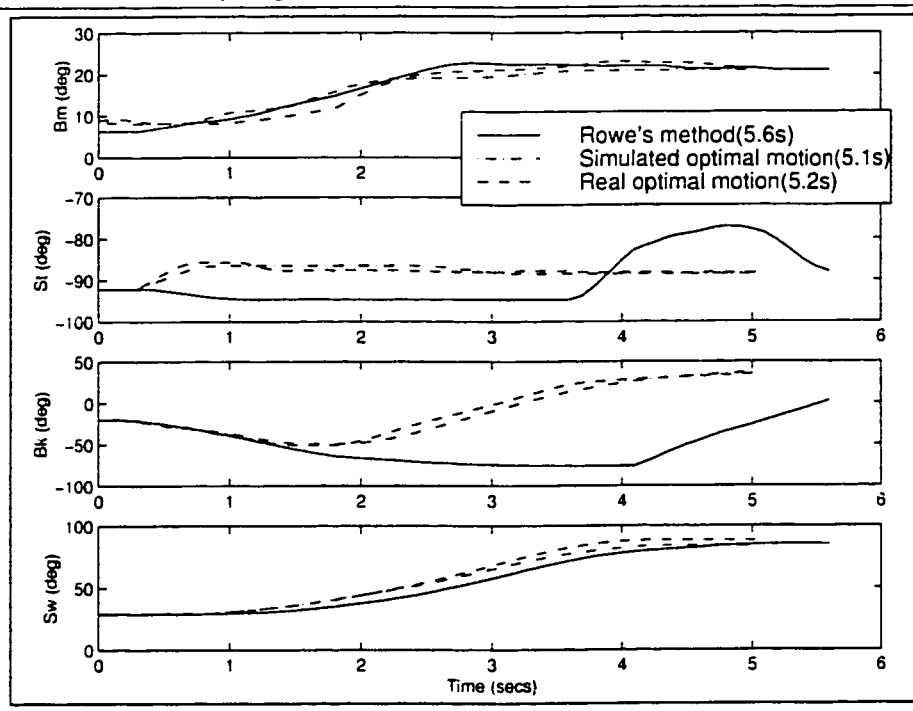


FIGURE 52. Task #5 (Loading configuration #3) - TL mode - loaded bucket.

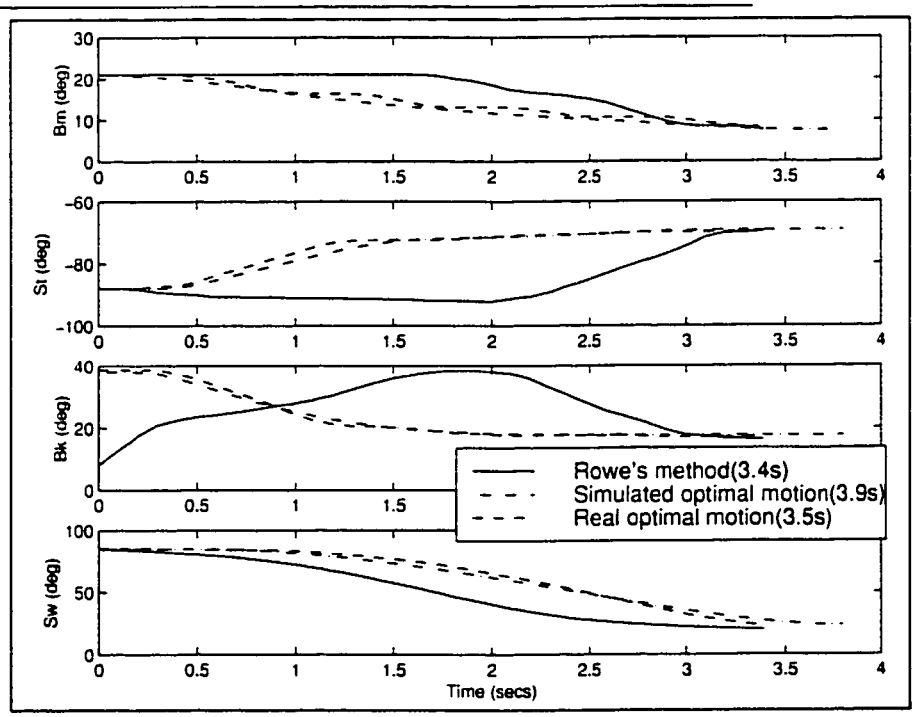


FIGURE 53. Task #6 (Loading configuration #3) - TL mode - empty bucket.

5.1.4 : Configuration #4 (Task #7)

Shown below are the motion responses for the *seed search vector* and the *optimal search vector*. The time taken for the *seed motion* is 9.8 secs, while that for the optimal motion is 5.0 secs.

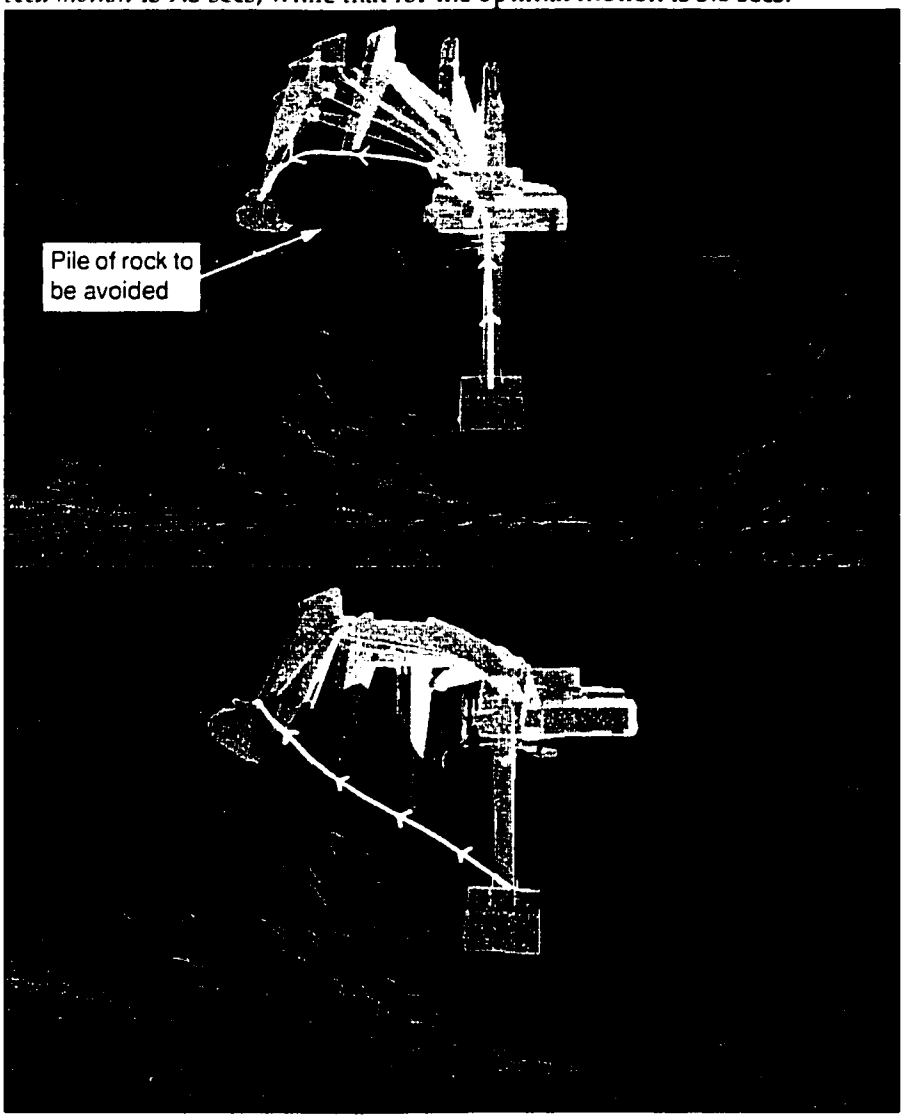


FIGURE 54. Task #7 (Configuration #4) (a) Initial seed motion (b) Real optimal motion.

In this task the optimization exploits the fact that the boom is slower than the stick joint. Thus, instead of booming up and raising the implements up and over the rock pile, as in the initial seed motion, it located a better motion which involved moving the stick out, thus straddling the rock pile under the boom and minimizing the upward motion of the boom joint.

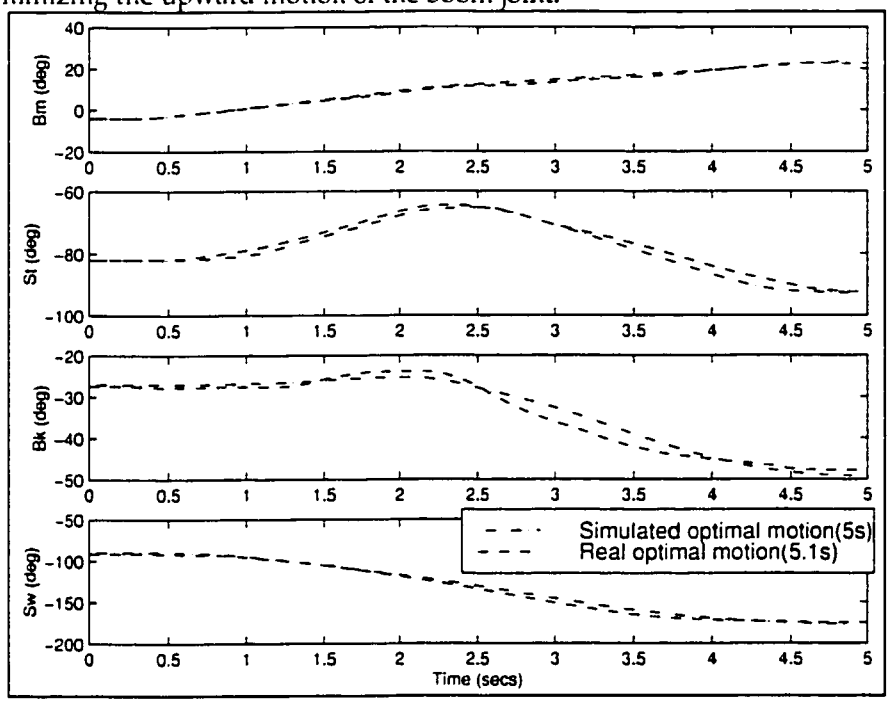


FIGURE 55. Task #7 (Loading configuration #4) - FF mode - empty bucket.

Section 5.2: An interesting result

An interesting result was obtained by computing optimal motions for the same task using different HEX operating modes. The figures below, show the time-optimal motions computed for tasks #5 and #6 using the *fine-finish* and the *truck-loading modes*. The results plotted are *real optimal motions*, i.e. they are motions as demonstrated on the HEX testbed. The motion time for the two results are not that dif-

ferent, although we initially expected that the *fine-finish mode* result would be slower since the hydraulic oil routing in the *truck-loading mode* is optimized for fast motions.

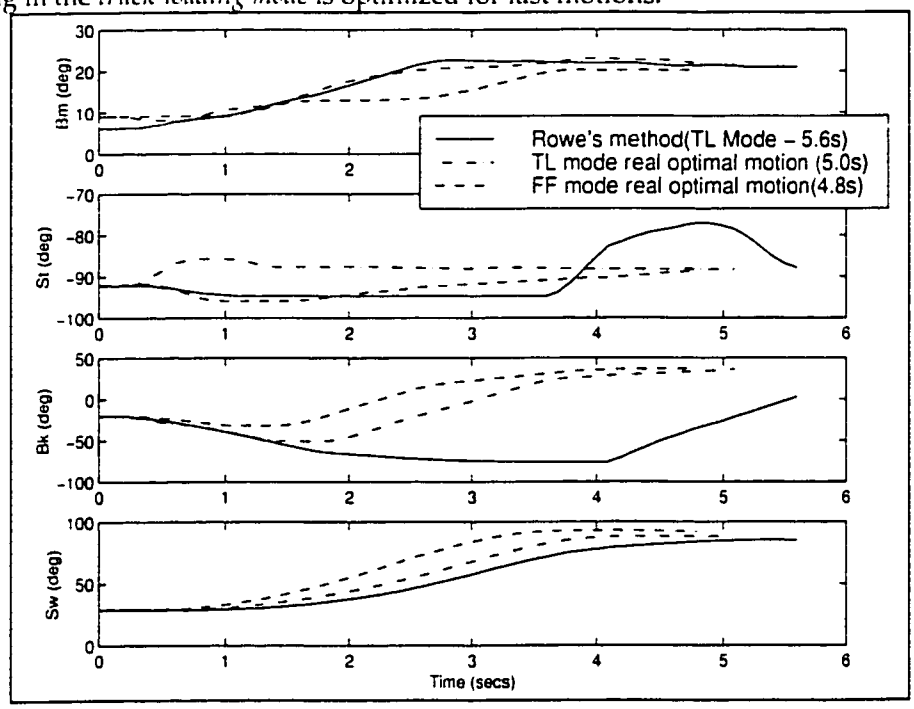


FIGURE 56. Comparison of time-optimal motions for task #5 using TL and FF HEX operating modes.

The similar motion time values (the 4.8 and 5.0 sec values are close enough that the motions can be considered as being equally good) can be explained by the fact that although the hydraulic oil routing is different for the two modes, the engine power capacity is still the same. The optimization manages to find a motion sequence that works around the constraints imposed by the restricted flow routing in the *fine-finish mode* and performs the task as fast as in the *truck-loading mode*. This result is significant, especially when demonstrated for the most common truck loading configuration since it dispels the popular belief that the *TL* mode fundamentally improves the ability of the HEX. As this test demonstrates, it is possible to achieve the same (time) performance in both modes, for the most common *truck-loading task*. It is likely that the *TL* mode improves the performance for a manually operated HEX since the characteristics of the *TL* mode are ergonomically superior. Human operators are limited in the number of joints they can coordinate simultaneously. Careful coordination is required to achieve the high performance in the *FF* mode, which is not an easy feat for a human operator. On the other

hand, the *TL* mode allows the human to achieve the same performance by requiring less careful coordination of the joints.

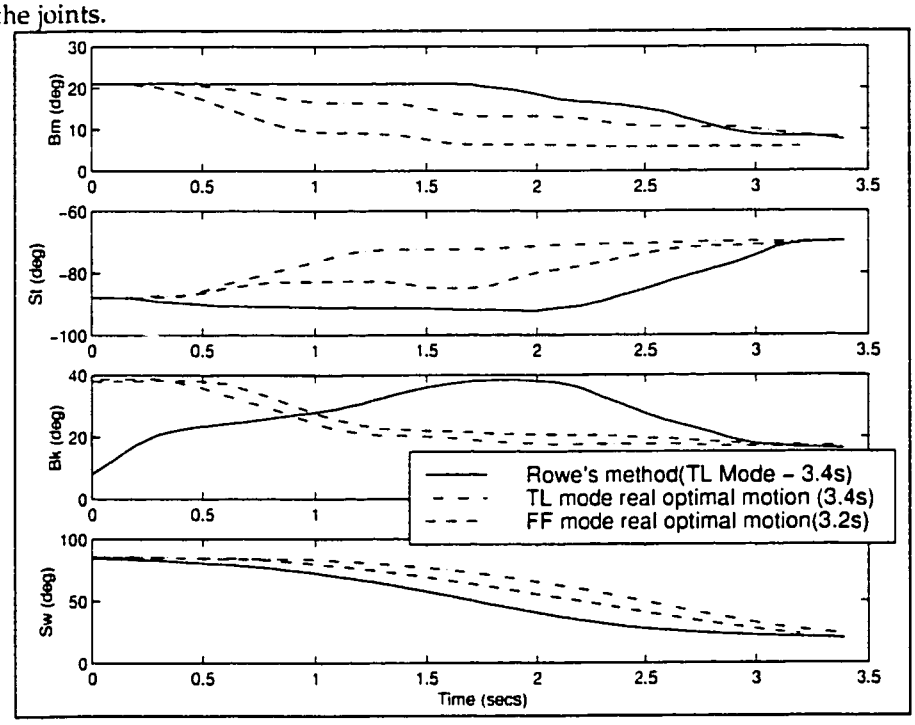


FIGURE 57. Comparison of time-optimal motions for task #6 using TL and FF HEX operating modes.

Section 5.3: Computing minimal energy motions

In this set of tests we computed the energy-optimal motions for the same 7 HEX *tasks* used in the minimum-time tests (Section 5.1). In an energy optimization we minimize the energy consumed while performing a *task*.

As mentioned in Section 4.3, we use two different methods for computing the energy term - the *implement energy* and the *engine energy*. The energy consumption in the *fine-finish* HEX operating mode is computed as:

$$E = \sum_{joints} \left(\int_{path} F_i dx \right) \quad \text{(EQ 33)}$$

This energy term is a measure of the energy consumed at the cylinders/motors; it does not reflect the losses in the hydraulic system, which can sometimes be quite significant. We refer to this energy term as the *implement energy*, since it is the energy consumed at the implements.

The second (and more complete) measure of energy consumption is the *engine energy*, which is used when modeling the *truck-loading* HEX operating mode[1]. It is computed by constructing a neural network that approximates the engine output power (rate of output of energy) as a function of the HEX hydraulic cylinder and motor velocities. This network approximates the power output of the engine, and hence accounts for all hydraulic and other losses downstream of the engine.

The differences in the two measures causes an interesting difference in the results obtained. The *task* times for the *tasks* performed in the *truck-loading mode* (*tasks* #1, 2, 5, and 6) are quite close to those obtained for the time-optimal case. This result is not seen for the *tasks* done in the *fine-finish mode*: *tasks* #3, 4, and 7. The time taken for these *tasks* is much longer than that taken by the corresponding time-optimal motions. This difference is explained by the fact that the *implement energy* measure does not add to the cost when no cylinders/motors are being moved. However, in reality, energy is being consumed even if no implements are moving since the engine is not switched off. The *engine energy* on the other hand does add to the cost even when no implements are being moved.

Thus, although time does not explicitly figure in the cost function, it is implicitly present in the energy computation for the *truck-loading* mode. This is not true for the *fine-finish* mode energy computation.

5.3.1 : Loading configuration #1 (Tasks #1, #2)

This loading configuration and associated tasks are the same as that defined in the time-optimal motion computation section (Figure 43). These tasks were done in the *truck-loading mode*. Note how the motion time for energy optimal results are quite close to that of the corresponding time-optimal

1. The addition of the more complete *engine energy* was only after the first round of model construction for the *fine-finish* operating mode. We did not compute the *engine energy* measure for that mode since that would have required us to redo the data collection for that mode, and that was not possible in the available time. However, the use of that measure for the *truck-loading* mode has proved its desirability over the *implement energy* measure.

motions. The results of the energy optimization are plotted below along with the results of the time-optimization, as well as the human expert's motions.

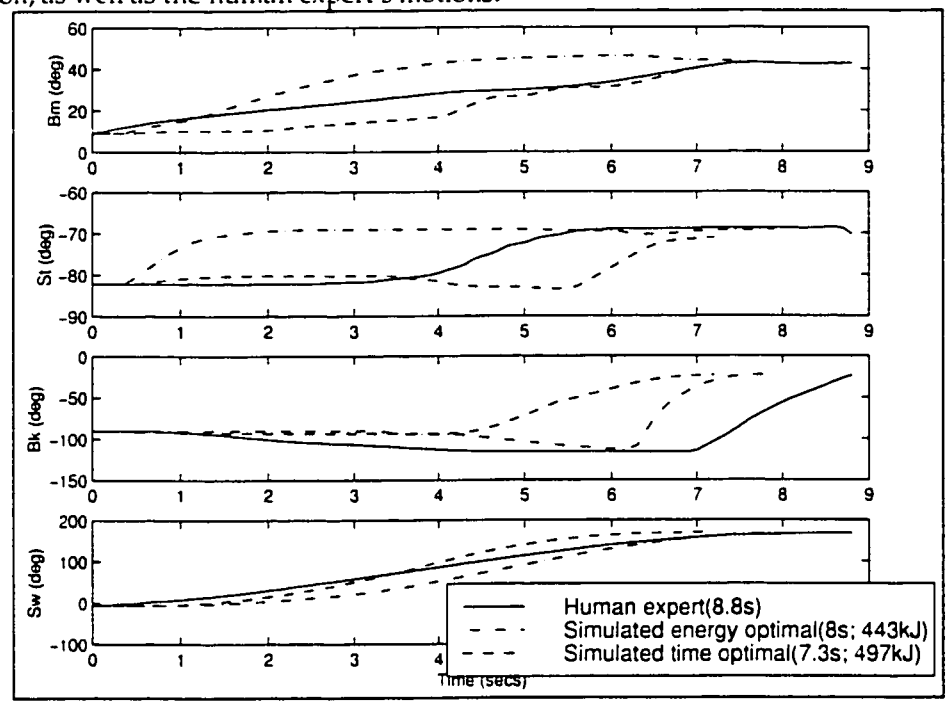


FIGURE 58. Task #1 (Loading configuration #1) - TL mode - loaded bucket.

The energy consumptions for the different traces - the *simulated time optimal* and *simulated energy optimal* motions - are also shown in the legend of each plot. These energy-optimal motions were not implemented on the testbed since it was not instrumented to measure the engine power consumption. This is also the reason why the expert's motions do not have a measured energy value, in the legend box of each plot.

However, for comparison, we computed the *engine* and *implement energies*. The *engine energy* was computed to be 483 kJ. This is slightly lower than the energy cost of the time-optimal motion in Figure 58. Since the *engine energy* measure does not indicate the energy used by each implement, we computed the *implement energy* (using Equation 33). The energy consumptions for the Bm, St, Bk, and Sw joints were 95 kJ, 33 kJ, 1 kJ, and 140 kJ respectively. Note that the sum of the *implement energies* is less than 483 kJ, which is the *engine energy* computed. This is because the *implement energies* do not account for losses in the system. The individual *implement energies* show that the Sw uses the most energy, fol-

lowed by the Bm joint. The energy usage of the St and Bk are quite small as compared to that of the other two joints.

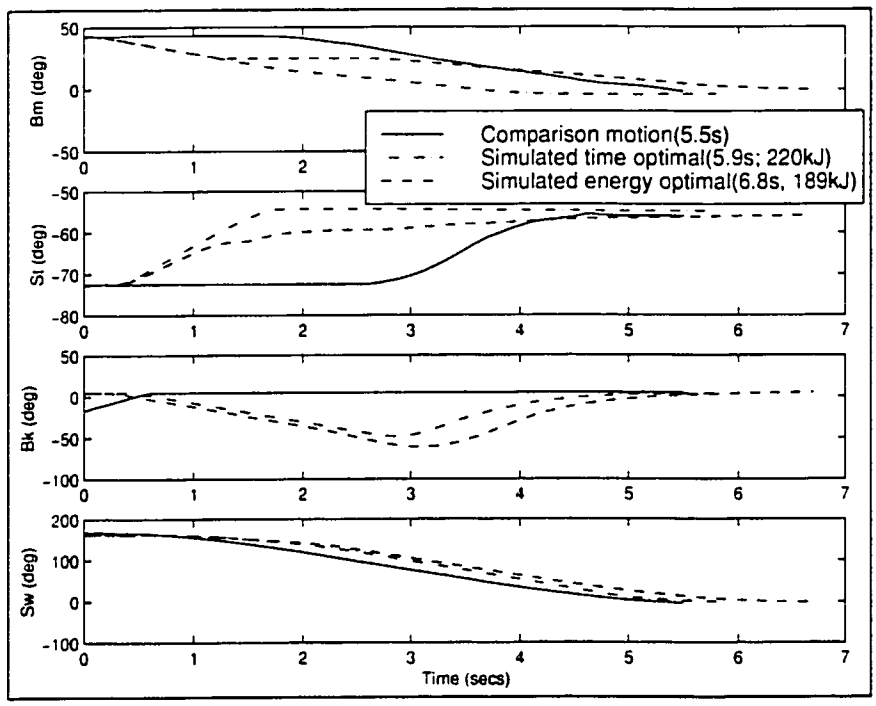


FIGURE 59. Task #2 (Loading configuration #1) - TL mode - empty bucket.

5.3.2 : Loading configuration #2 (Tasks #3, 4)

The loading configuration is the same as that illustrated previously in Figure 48. This task is performed in the *fine-finish mode*. Note that unlike task #1 and #2 above, the task times for tasks #3 and #4 are much longer than that for the corresponding time-optimal motions. This is because slowing down the joint motions offers an energy reduction which is exploited by the optimization. However there is no penalizing factor imposed by the *implement energy measure* (used in the *fine-finish mode* model) for the extra time taken, during when the engine is still expending energy.

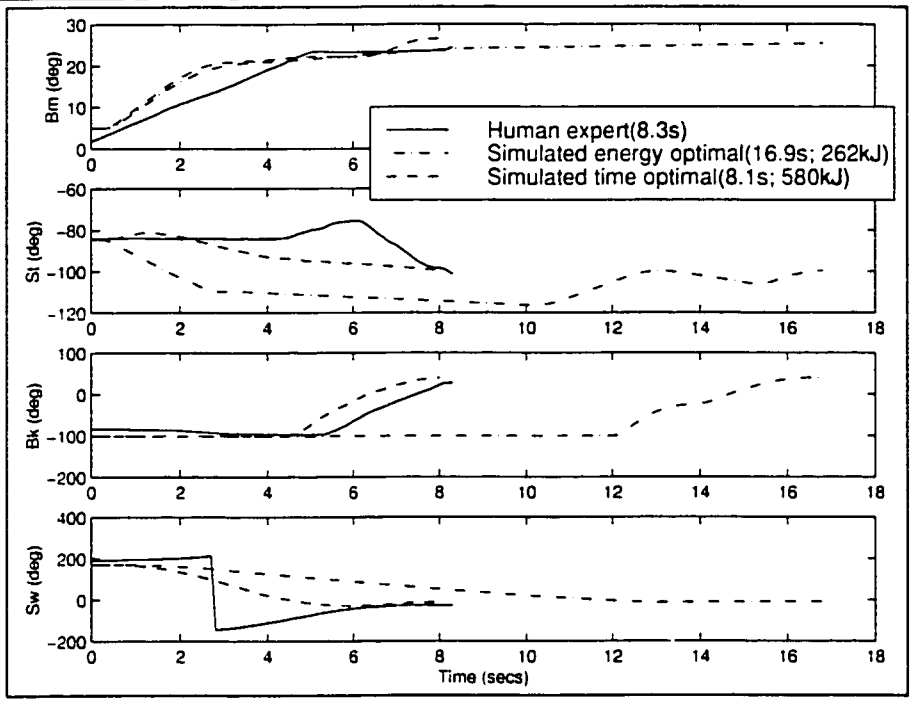


FIGURE 60. Task #3 (Loading configuration #2) - FF mode - loaded bucket.

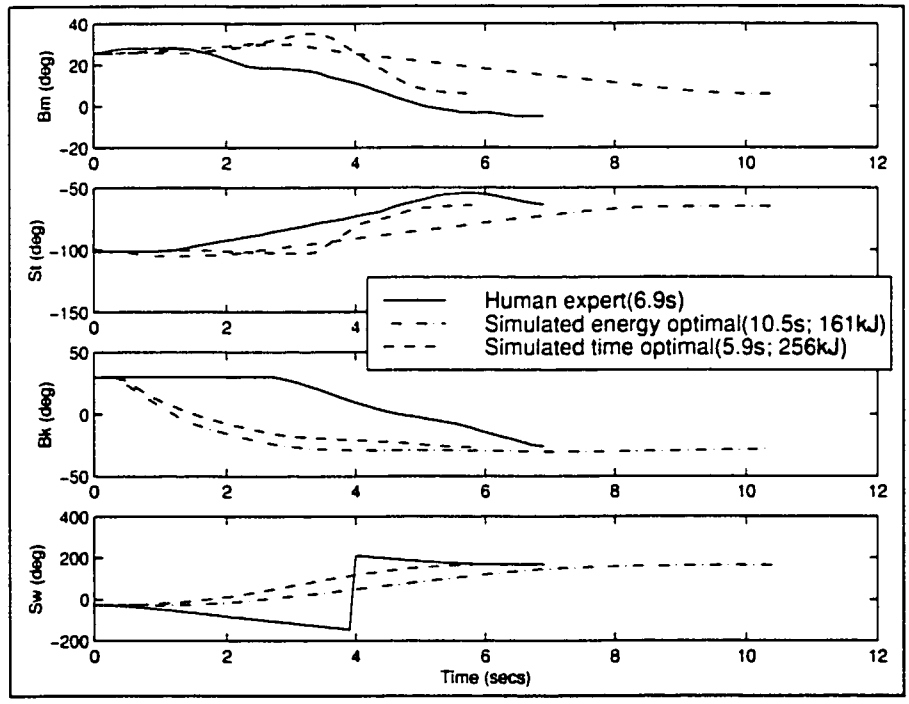


FIGURE 61. Task #4 (Loading configuration #2) - FF mode - empty bucket.

5.3.3 : Loading configuration #3 (Tasks #5, 6)

The configuration is illustrated in Figure 51. The results are shown below. The energy optimal motion appears to be gaining its energy savings in task #6 by reducing the deceleration rate of the swing joint. This makes intuitive sense since high acceleration and deceleration of the swing joint waste energy. Note that the deceleration and acceleration are not reduced so far that the losses due to the engine idling (or at low load) offset the energy savings.

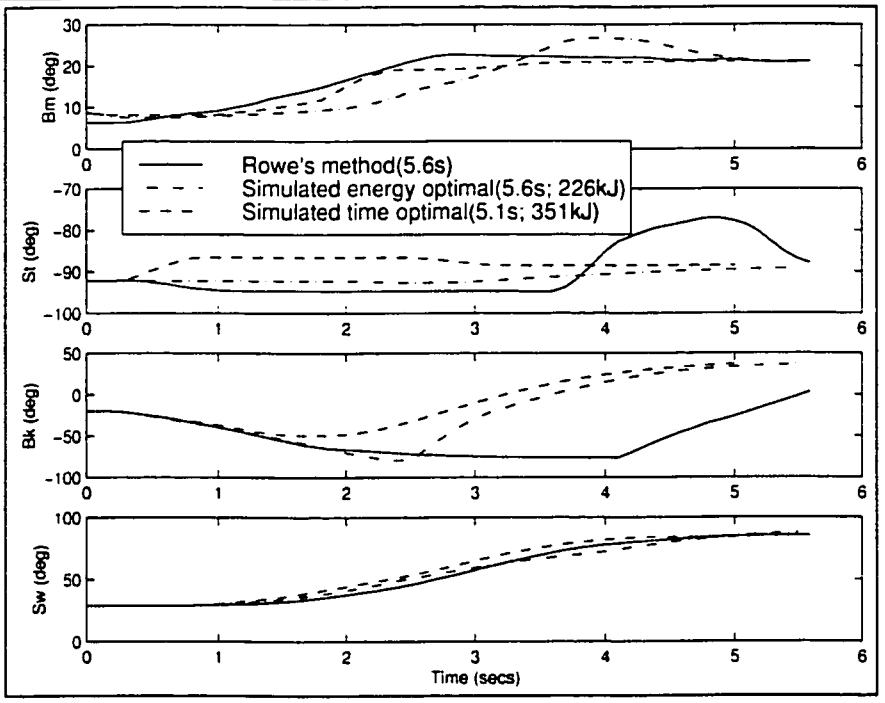


FIGURE 62. Task #5 (Loading configuration #3) - TL mode - loaded bucket.

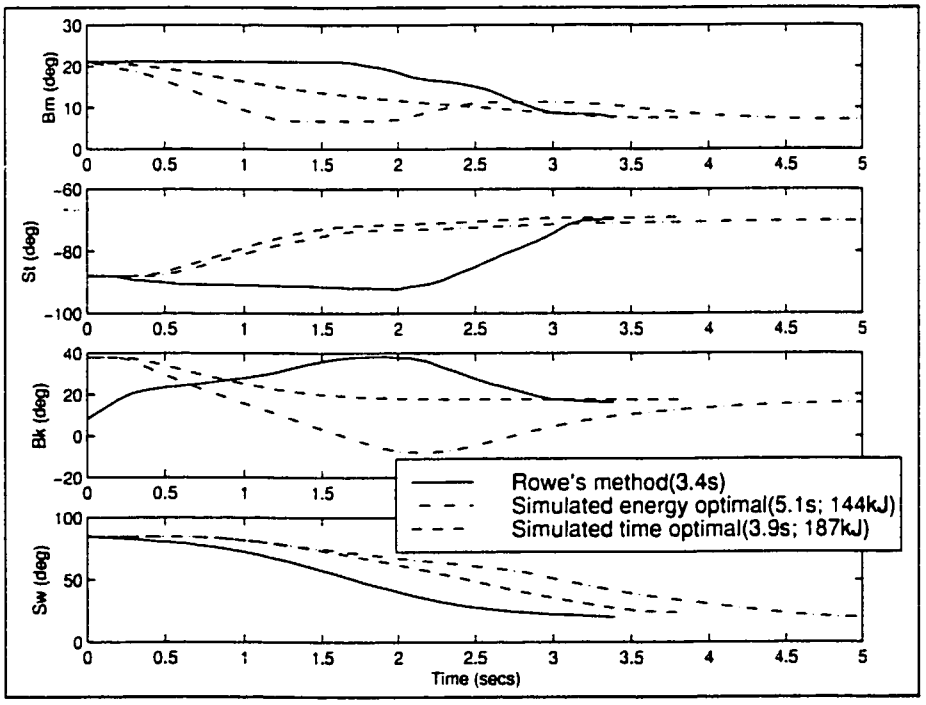


FIGURE 63. Task #6 (Loading configuration #3) - TL mode - empty bucket.

5.3.4 : Configuration #4 (Task #7)

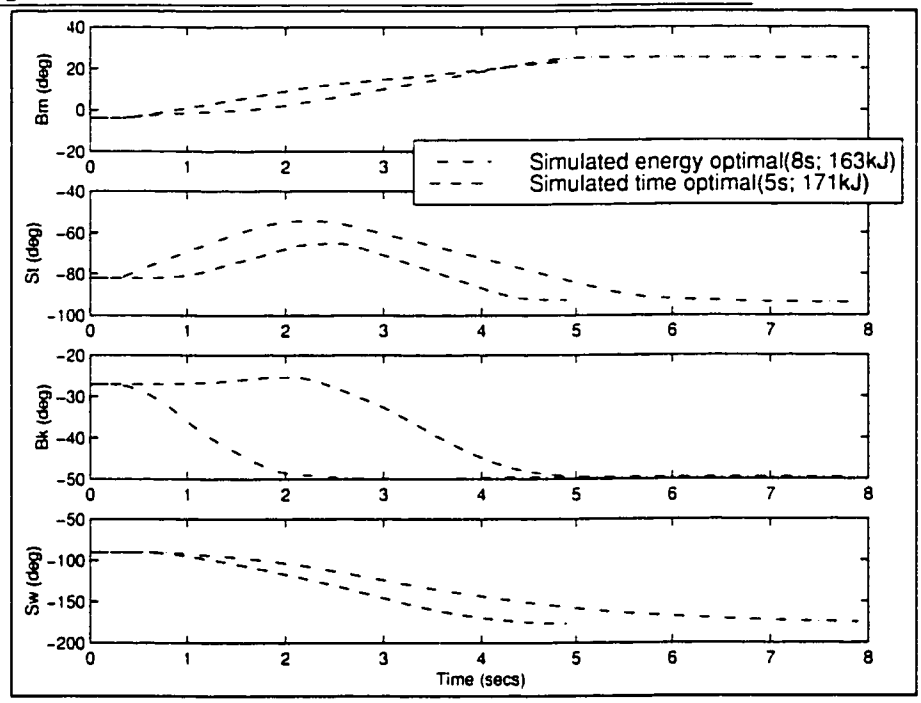


FIGURE 64. Task #7 (Configuration #4) - *FF* mode - empty bucket.

The configuration for this task is illustrated in Figure 54. This task was performed in the *fine-finish mode* (and uses the energy measure that does not penalize engine idling). Again, as for *task* #3, #4, the energy savings are obtained by decreasing the acceleration and deceleration rates.

Section 5.4: Tabulation of all optimal motion planning results

All the results are tabulated in the following table.

		Time Optimization		Energy Optimization	
	Time for *comparison motion* (secs)	**Time, in secs (%)**	**Energy (kJ)**	**Time (secs)**	**Energy (kJ)**
Task #1	8.8 (human expert)	7.3 (-17%)	497	8.0	443
Task #2	5.5 (human expert)	5.9 (+15%)	220	6.8	189
Task #3	8.3 (human expert)	8.1 (- 3%)	580	16.9	262
Task #4	6.9 (human expert)	5.9 (-15%)	256	10.5	161
Task #5	5.6 (Rowe's method)	5.1 (- 9%)	351	5.6	226
Task #6	3.4 (Rowe's method)	3.9 (+14%)	187	5.1	144
Task #7	-	5.0	171	8.0	163

TABLE 7. Tabulation of optimization results.

NOTE: (1) The bracketed percentage values in column two indicate how much lower/higher the optimal motion time values are than the corresponding *comparison motion* (in column one). Values are expressed as a percentage of the *comparison motion* time.

(2) The *comparison motions* only have time values since the measurement of energy (engine shaft power) was not possible on the Caterpillar 325A excavator testbed.

Section 5.5: Computation time for optimal motion planning

As described in Section 4.5, the SA optimization is performed 15 times for each *task*; it is performed 3 times for each of the 5 cases where the search uses 1 through 5 knot points per joint. The average time

taken to compute the optimal motions for *tasks* #1 through #7, and the number of candidate motions examined in each case are listed in Table 8, and are plotted in Figure 65 and Figure 66.

	Avg. time for 1 knot point per joint, in hrs. (Num srch vectors examined)	**Avg. time for 2 knot point per joint, in hrs. (Num srch vectors examined)**	**Avg. time for 3 knot point per joint, in hrs. (Num srch vectors examined)**	**Avg. time for 4 knot point per joint, in hrs. (Num srch vectors examined)**	**Avg. time for 5 knot point per joint, in hrs. (Num srch vectors examined)**
Task #1	1.0 (7314)	2.0 (15864)	1.6 (24253)	2.5 (36143)	3.5 (49159)
Task #2	1.6 (7077)	5.5 (23580)	5.3 (37833)	6.4 (45924)	6.6 (44658)
Task #3	1.1 (5785)	2.4 (12335)	2.1 (19237)	2.8 (59160)	3.1 (65233)
Task #4	1.3 (5813)	2.5 (10702)	2.0 (18269)	2.7 (19104)	3.0 (32575)
Task #5	1.3 (8278)	3.0 (21370)	3.4 (40036)	3.7 (46959)	5.1 (59209)
Task #6	1.3 (5932)	4.9 (19542)	4.5 (31354)	4.9 (38251)	4.9 (35544)
Task #7	1.7 (6941)	2.8 (11887)	2.6 (18834)	3.5 (24281)	3.8 (30813)
Type of computer used (for all corresponding *tasks*)	SUN Sparc-20	SUN Sparc-20	SUN Ultra-1	SUN Ultra-1	SUN Ultra-1

TABLE 8. Average time taken in hrs. (over 3 parallel SA optimizations) to compute time-optimal motions for the 5 cases for each *task*. The number in parentheses indicate the average number of search vectors examined during the simulation.

NOTE: The SUN Ultra-1 machines were 143MHz single processor workstations. The SUN Sparc-20 machines were 100MHz single processor workstations.

Looking across the columns of any row in Table 8, we can see the increase in computation time (and number of candidate motions examined) as the number of knot points used increases (for a given *task*); looking down any given column shows the variation in computation time for different *tasks*, for the same number of knot points used. The variation over different *tasks* (for the same number of knot points) is a reflection of the differing nature of the cost surfaces. If a cost surface contains many local minima, the optimization may take much longer to navigate towards the minimum since it is much harder to discern the structure of the cost surface (thus requiring the examination of more candidate motions).

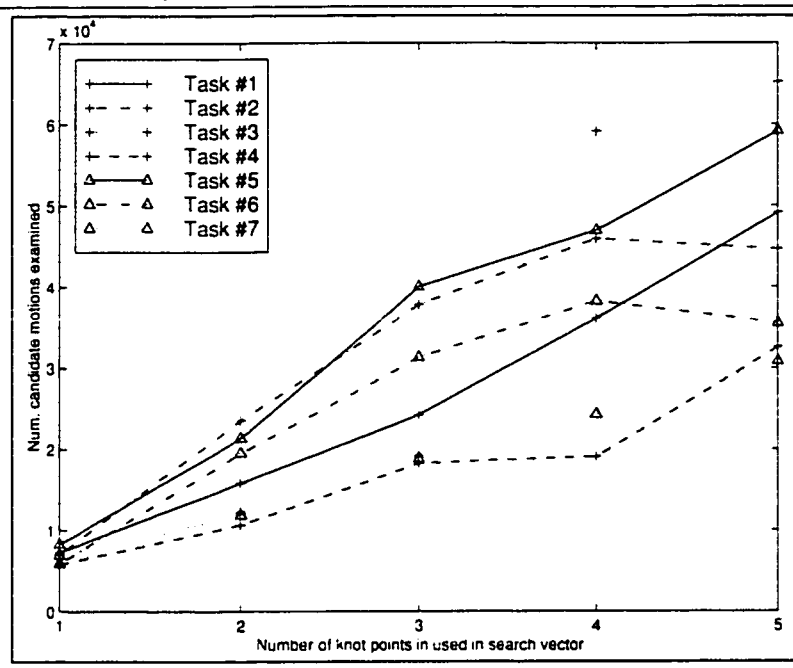


FIGURE 65. Plots of the number of candidate motions examined during the time-optimal motion computation for the 7 *tasks*.

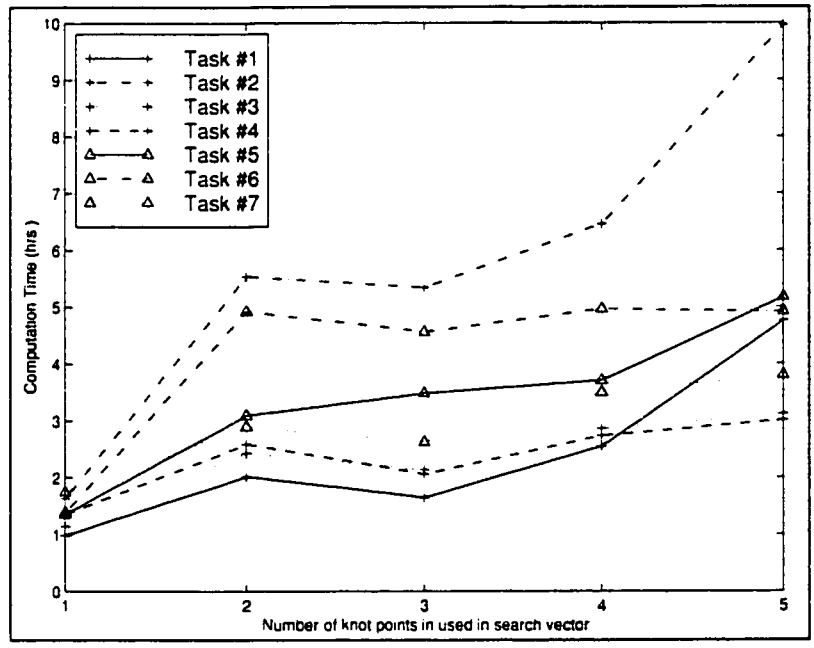


FIGURE 66. Plots of the average time taken to compute the time-optimal motions for all *tasks*. These plots use the data tabulated in Table 8.

Overall, the above time-optimal motion computations take anywhere from 1 to 10 hours, depending on the number of knot points used in the search vector. Although not shown here, the computation time for the energy optimization tests are similar to the time optimization tests. Figure 66 shows a drop in the computation time for nearly all tasks in the case of 3 knot points (as compared to the time required for 2 knot points). This is due to the fact that the newer Ultra-1 workstations (used for the 3 and 4 knot point cases) are faster than the SUN Sparc-20 workstations. No such dip is seen in Figure 65 since that plots the number of candidate motions examined; this metric is independent of the speed of the computer on which the optimization is running.

The plot for *task* #3 in Figure 65 shows a marked increase in the number of motions examined as the number of knot points increases from 3 to 4 (and higher). Although it is not easy to predict the exact reason for the sharp increase, it is likely that it is due to the cost surface becoming more "pocked" with valleys (local minima), as the number of knot points goes from 3 to 4 (and higher). This causes the minimization to take a longer time to get to the optimal result.

Although the search vector length increases by a factor of 8 for each additional knot point, the computation time does not increase dramatically. The plots above indicate that the relationship between the computation time (or more importantly, the number of candidate motions examined) and number of knot points is almost linear. (In comparison, the multi-dimensional conjugate gradient method is of $O(N^2)$ as are the quasi-Newton methods. See [Press 96] for more details).

To better understand the CPU usage, we performed a detailed profiling of the optimization program code for two test runs of the SA optimization; one for *task* #1 and another for *task* #3 (This profiling test is distinct from any of the other tests). Both these tests used 1 knot point per joint, i.e. the search vector was 8 elements long. For the *task* #1 (*truck-loading mode*) profiling test, a total of $3.4X10^{11}$CPU cycles (2383 secs) were used and 10050 candidate motions were explored. Of the total usage,$3.1X10^{11}$ cycles (2172 secs) were used for machine simulation. In the profiling test for *task* #3 (*fine-finish mode*), the total CPU usage was $2.0X10^{11}$ cycles (1344 secs) for 5189 candidate motions explored. Of the total usage, $1.8X10^{11}$ cycles (1268 secs) were used for the machine simulation. Both tests were run on a 143 MHz SUN Ultra-1 workstation. From the above figures it can be seen that between 96 and 99% of the total CPU usage was by the machine simulation function. The rest of the CPU usage was consumed by the SA functions, e.g. generating new candidate motions, implementing the cooling schedule, etc.

A future implementation of this optimal motion planning approach can be sped up significantly in the following ways:

- Multi-resolution approach: Performing the optimization using 5 different sets of knot points is wasteful. A multi-resolution approach can dynamically vary the number of knot points used to describe the joint commands. Additionally, the number of knot points used for each joint can be different; thus a joint that does not have much variation in its command has fewer knot points

describing the command function than a joint whose command has large variations. Both these approaches (dynamically adding and deleting knot points, and using different numbers of knot points for each joint) ensure that the search vector length is only as long as needed, thus making the search more efficient. We did not implement this approach in this work due to lack of time.

This is one of the most promising ways of decreasing the computation time. It may be possible to increase the speed by up to a factor 5 using this approach.

Another approach to selecting a desired discretization is described below on page 122. That approach can also help reduce the computation by eliminating the need to perform the optimization at multiple resolutions.

- There are a number of ways of parallelizing the annealing search. It is possible to run a batch of optimizations which quench the problem quickly and provide a starting point for a second round of annealing optimizations. It may be possible to obtain a linear speedup by adding more parallel processes in the optimization. The number of parallel optimizations is dependent on the amount of available computing.
- It may be possible to use larger integration steps in the robot model during the initial stages of the optimization, and using smaller step sizes as the optimization progresses. This is a valid approach if the larger integration step size provides an upper bound on the total cost.
- A more detailed analysis of the cost surface may allow the use of a combination of a downhill method in conjunction with the SA optimization. Although such a combination was unsuccessful in our test cases, it may be possible to use some specific problem features to make it work.
- Re-coding: The current implementation is a combination of program code culled from a variety of sources and is not very efficient.
- Better hardware: These computation times are from running the optimization on workstations that are about 7 years old. Current workstations are at least 3-4 times as fast.
- In spite of the above suggestions to speed up the motion computation, it may still be desirable to use methods such as look-up tables to cache the solutions computed off-line (or in the background). For a given loading situation (i.e. for a given arrangement of robot and obstacles) it is possible to construct a lookup table indexed on the basis of discrete start and end locations; intermediate solutions can be obtained by linear interpolation. The class of machines examined in this work perform fairly repetitive motions (i.e. the start and end locations do not vary significantly), the environments (the soil face) only vary gradually, and the obstacles (such as trucks and dozers) are predictable in their locations relative to the robot. Hence, the size of the lookup table may not be prohibitively large. For instance, a realistic discretization of the start and end spaces might be 100 voxels each. Thus, the lookup table would have a very reasonable size of 10^4 entries. Each entry in the table would have an associated command sequence for each of the robot's joints.

Section 5.6: Cost variation studies

In the results shown in the previous sections we showed the best result obtained but did not discuss the spread of the results obtained from multiple runs of the optimization for the same *task*, or the details of how the cost decreases as the optimization progresses. These analyses help give a better picture of the performance of our optimal motion planning approach.

The following cases are discussed in this section:

- Case #1: Cost variation with decreasing temperature during an annealing optimization for different *tasks*.
- Case #2: The cost variation if an annealing optimization is run multiple times for the same *task* (using the same inputs). We show two plots: the cost variation as a function of the annealing temperature for different knot point cases, and the spread in the final cost values.

All the cost values shown in this section are for time-optimizations. Extremely high costs indicate collisions or spillage.

5.6.1 : Case #1:

The first variation: cost versus annealing temperature, is shown in Figure 67. In this figure, "temperature index" is plotted along the x-axis. This quantity refers to the index number of the temperature levels[2], with the start (hottest) temperature having an index of 1.

2. The annealing progresses by maintaining the temperature at fixed values and decrementing the temperature when the optimization has simmered enough (reached equilibrium) at that temperature (the termination condition for each temperature is described in Section 4.4).

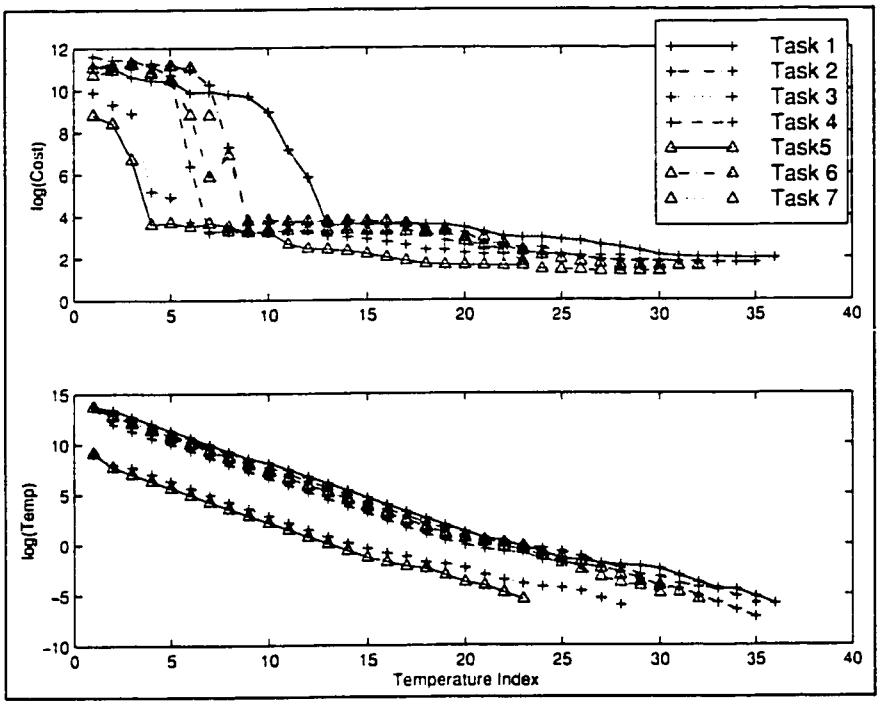


FIGURE 67. Cost and Temperature variation as annealing optimization progresses.

The upper plot shows the natural logarithm of the mean cost value at each temperature, for the 7 *tasks*. The cost values are very high for motions that cause collisions with the environment, or cause spillage. The precipitous drop seen in the cost as the optimization cools marks the transition to collision and spillage-free motions. At high temperatures the annealing explores a wide range of motions, even though they cause collisions/spillage, thus increasing the mean cost at high temperatures.

The lower plot shows the natural logarithm of the annealing temperature. It shows that the temperature drops exponentially (since the log plot is approximately linear).

Figure 68 shows the number of search vectors examined at each temperature for *task* #1; to minimize clutter, we do not plot the same for the other *tasks*. The annealing seems to examine most of the candidates in the mid-temperatures and spends relatively little time in the lower temperatures[3].

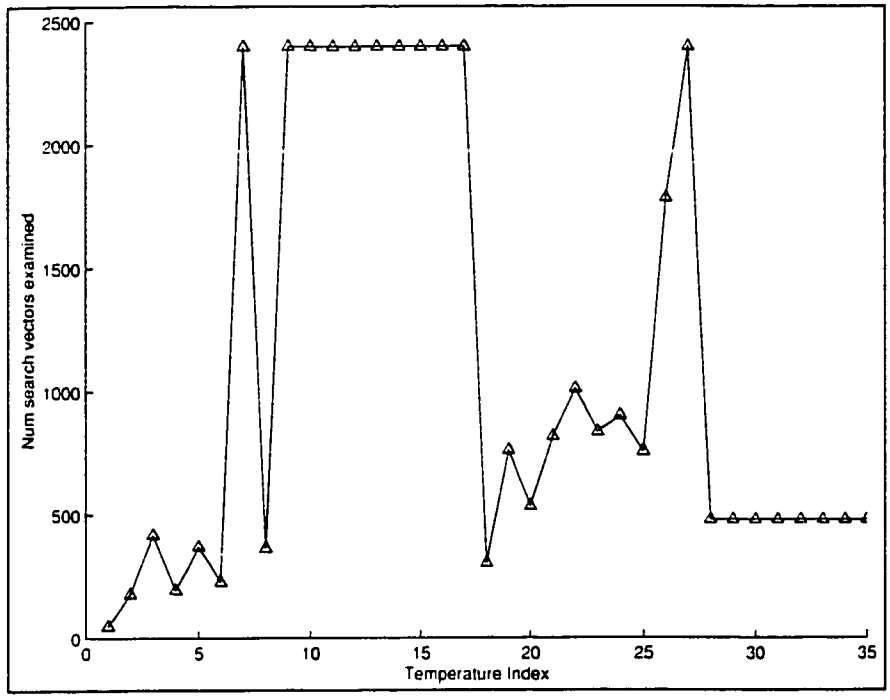


FIGURE 68. Number of candidates examined during the annealing optimization for task #1.

5.6.2 : Case #2:

As noted in Section 4.5, we run the SA optimization for 5 different knot point cases, i.e. using 1 through 5 knot points per joint. Three instances of each of these 5 cases is run in parallel. All 15 optimizations (3 instances of 5 cases) for each *task* use the same *seed vector.*

3. Some researchers have suggested terminating the annealing early and using a simpler downhill method in the later stages of the annealing optimization. This is based on the observation that a typical annealing optimization spends a fair amount of time examining candidate at low temperatures but the gain in cost is very small. We examined this approach on some optimization tests by stopping the annealing early and using a downhill simplex minimization starting from the best search vector at that time. This did not have much success - the downhill simplex method was unable to significantly improve the motion, while the annealing was able to do so.

This sub-section shows two sets of plots. The first set (Figure 69) shows the decrease in the cost of the time-optimal motions for *task* #1 as a function of the annealing temperature, for the 15 optimizations. The figure plots the logarithm of the mean cost as a function of the temperature index. It can be seen that the higher knot point cases reach collision/spillage-free motions at lower temperatures. This may be because the cost surfaces get more "peaks" (and valleys) as the dimensionality increases. For instance, while the cost for the 1 knot-point case (thick line) drops at around temperature index of 5, the cost of the 3 knot-point case (thick line) is still hugging the high-cost "peaks" (which may not have existed for the 1 knot point case) at temperature index of 10.

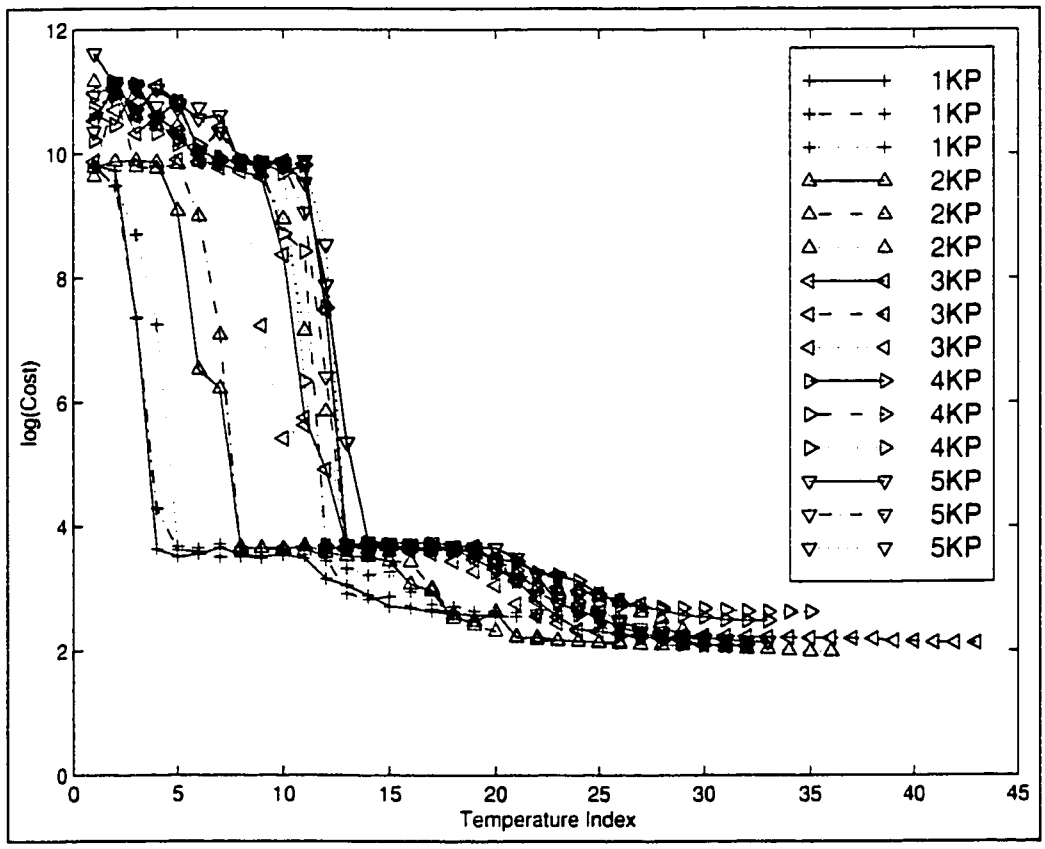


FIGURE 69. Variation in cost for 3 instances each of 5 knot point cases, when computing time-optimal motions for *task* #1 (KP = Knot Points).

Figure 70 shows the spread in the cost values from the 15 optimizations described earlier (for *task* #1). We performed another 15 optimizations to improve the statistical estimates; there are thus a total of 30 data points in Figure 70.

The single line in the figure connects the mean values of the cost for the different knot point cases. The widgets at each knot point case show the cost of the optimal-motion obtained from each of the 30 optimizations. The spread in the cost values increases as the number of knot points increase since the cost surface becomes more labyrinthine, thus offering making it harder to get close to the global optimum.

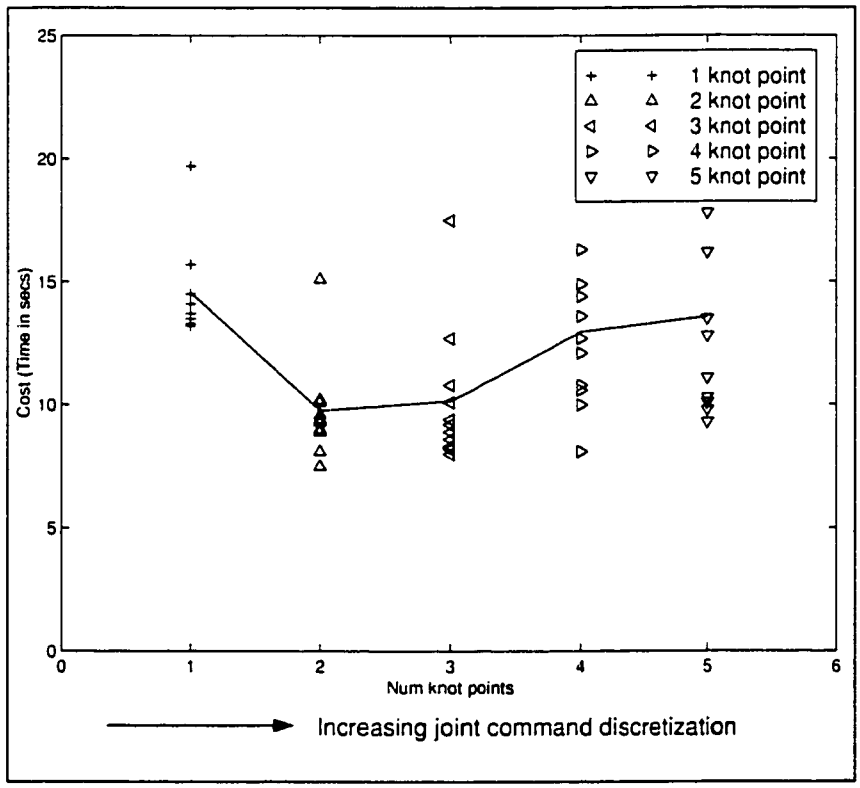


FIGURE 70. Variation in the cost of multiple runs (6 runs per knot point) of the SA optimization for *task* #1, and using the same *seed* vector. Each data point represents the results from one optimization. The thick line connects the mean values of the cost for each knot point.

Table 7 shows the mean and standard deviation values for the data in Figure 70. It can easily be seen that the standard deviation of the cost values for the 2 knot point case is the lowest. The spread of the

standard deviation values suggest that for a given selection of annealing parameters (as was the case for the data in Figure 70) there is an optimum number of knot points for which the standard deviation of the cost is the lowest. This observation is consistent for all the examples in this work. Therefore, one way of selecting the desired command discretization (i.e. the number of knot points to use for the command discretization) is by studying the variation of standard deviation values for different knot point cases and selecting the one with the lowest. Once the number of knot points is selected, the optimization need only be performed for that discretization, instead of a range of discretizations. This would greatly reduce the computational cost by reducing the number of times that the optimization has to be performed.

Number of knot points	Mean of cost (secs)	Std. dev. of cost (secs)	Min. cost (secs)
1	13.5	1.8	12.2
2	8.9	1.7	7.3
3	9.1	2.8	7.8
4	11.9	3.1	8.0
5	12.6	3.9	8.3

TABLE 9. Table of mean, standard deviation, and minimum cost values for the optimization runs shown in Figure 70 (Task #1).

As a comparison, the mean and standard deviation values of the cycle times for the human expert operator for task #1 were 8.7 secs and 0.48 secs respectively. The mean and standard deviation were computed for the 6 loading passes that the human performed. Thus, comparing the human's operation to our method it would be fair to say that if an optimization system were used to compute optimal motions by performing the optimization at the (optimum) selected discretization, on average the two would be comparable. However, it is possible that the best result from the optimization is better than that of the human, as was the case in our Task #1 example.

Section 5.7: An example application

One use of our optimal motion planner is as an aid to machine designers. Today, designers of machines such as the HEX do not have an easy way of determining the effect of a design change on the end application, such as *truck-loading*. The effect of the change is gleaned by performing step and ramp tests on individual joints, and extrapolating those results. Occasionally prototypes may be built and tested by expert operators whose qualitative comments provide some evaluation of the design change. The tests used today do not provide a direct mapping from design change to effect on the end application. However, our optimal motion planner can provide that information to a designer.

To illustrate this application, we recomputed the time-optimal motion results for *task* #1 (Figure 43 on page 91) after increasing the boom cylinder response by 50% (we scaled the Bm joint velocities obtained from the robot model by 50%). From a cursory examination of the task it would appear that increasing the boom speed will decrease total cycle time since the boom is one of the heaviest and slowest joints, and the *task* involves significant boom joint motion.

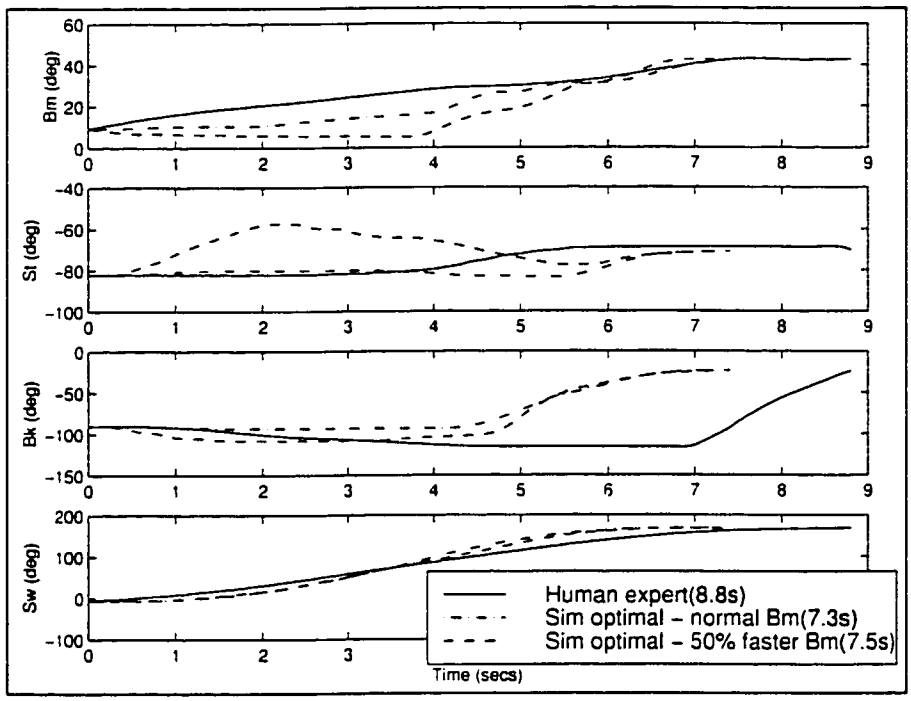


FIGURE 71. Comparison of *task* #1 optimal motions for a normal Bm, and 50% faster Bm.

Our results show that the intuitive observation is not necessarily true. The time-optimal motion for the "enhanced boom" takes essentially as long as the normal ("un-enhanced") boom case. The Bm rises more quickly in the enhanced case (than in the normal case) but that does not reduce the overall time. This is because the bottleneck in the operation is not the Bm, but the Sw joint. Results such as these can greatly increase the machine designers' insight into such application *tasks*.

Section 5.8: Summary

In this chapter we have demonstrated results from using our approach to optimal motion planning for a HEX robot. We have computed collision and spillage-free time-optimal motions for 7 *tasks*, as well as the energy-optimal motions for the same *tasks*. We have demonstrated the time-optimal motions on a HEX testbed and have thus proved that the robot models used for optimal motion planning are very adequate in predicting robot motion. This is the first time that globally time-optimal motions have been computed (and demonstrated) for a hydraulic robot.

We compare the time-optimal motions (Section 5.1) for *tasks 1,2,3,4* to those of a human expert for the same *truck-loading tasks*. The execution time for the two sets of motions are very comparable. The computed time-optimal motions are as good as, or faster than the human expert. It should be noted that most HEX operators are not as good as the expert used for our tests; the expert operates the HEX 20-25% faster than an average operator. We have also compared the motions for *tasks 5,6* to those generated by Rowe's method. Our results again are as good or better than, the results from Rowe's approach. Rowe's results were obtained after unloading 54 bucket loads in a truck (9 truck-loads).

The energy-optimal motions (Section 5.3) computed for the same 7 *tasks* offer a good contrast to the time-optimal motions. More importantly, the use of energy as an alternative objective function underscores the flexibility of our method in allowing the inclusion of any quantifiable measure (time, energy, joint torques etc.) in the objective function.

Our current implementation of the optimal motion computation takes a few hours to compute the optimal motions for a specified *task* (Section 5.5). We have offered ways by which this computation time may be reduced for future implementations. These include more efficient code implementations, as well as developing an approach to determining the best discretization for each joint command. Our current approach is inefficient since we explore 5 different discretization levels; incorporating the selection of the discretization in the search promises to significantly reduce the computation time by eliminating the multiple optimizations (once for each discretization level).

Besides running the optimization for each command discretization level (number of knot points), we run the optimization multiple times for each discretization level. In our examples we use 5 levels (using one through five knot points for each joint) and run the optimization 3 times for each level. The need to perform multiple optimizations at each level is due to the stochastic nature of the Simulated Annealing algorithm. Performing the optimization multiple times for each *task* improves the probability of the algorithm reaching the global minimum (Section 5.6). Although, as mentioned in the previous paragraph, it is possible to avoid performing optimizations for each discretization, it is not possible to avoid running the optimization multiple times for the same discretization.

Although the long computation times detract from our approach, it is still a valuable utility which can be used as a design tool, or for off-line computation of optimal motions. Besides our approach, there do not exist any methods capable of providing the information that our optimal motion computation method provides for hydraulic robots. Shortening the computation time will only make the approach more attractive for these applications, besides opening up other areas.

The utility of our method as a design tool is illustrated by an example in our results (Section 5.7). Today's designers do not have any means of determining the effect of a machine design change on the final application, such as *truck-loading*. Our method can provide that information. This can be done by computing the optimal motions for a specified *task* using the model of the re-designed machine, as well as the model of a baseline machine (for comparison). The comparison of the two sets of motions can give designers insights into the effects of the design change on the application *tasks(s)*.

We have thus demonstrated the effectiveness of our approach in computing optimal motions for hydraulic robots, and have illustrated its utility as a design tool. The following chapter, Chapter 6, lists conclusions from this research, and its contributions. Possibilities for future extensions of this research are described in Chapter 7.

CHAPTER 6

Conclusions and Contributions

In this chapter we look back at the research, and our approach to solving the optimal motion planning problem. Section 6.1 gives a summary of our research. Section 6.2 discusses our solutions to the research issues identified in Chapter 1 and conclusions from this work. Section 6.3 lists our research contributions to the state-of-the-art.

Section 6.1: Summary of the research

In this thesis, we have proposed and demonstrated an approach to computing optimal free-space motions for hydraulic robots. We use a robust search technique called Simulated Annealing to search the robot's command space for the temporal command sequence that optimally performs a given *task* (with specified start and end conditions), while avoiding obstacles, spillage of bucket payload, and satisfying kinematic and dynamic constraints. The objective function is composed a weighted linear combination of different cost terms. It can include any quantifiable measure such as time, energy, joint forces, etc. Our optimal motion computation approach is the first to allow the use of a diverse set of measures (other than time), in the objective function. All other methods in the literature (for electric-drive or hydraulic robots) focus almost exclusively on time-optimality.

During the search, we evaluate the cost of a candidate command sequence using a hydraulic robot model. This model was developed using a novel approach, also proposed and demonstrated in this thesis. This powerful modeling approach allows the construction of computationally inexpensive

hydraulic robot models that capture the significant actuator interactions that are typical of hydraulic robots.

We demonstrate our optimization approach by computing optimal motions for a Caterpillar 325A hydraulic excavator (HEX) robot. We present examples of computing time-optimal motions for 7 different *tasks*, as well as energy-optimal motions for the same 7 *tasks*. The time-optimal motions for all 7 *tasks* are demonstrated on a real HEX testbed. The motions computed for 4 *tasks* are compared to the motions of a human expert. The comparisons show that the computed optimal motions are as good as, or better, than the expert's motions. We also compare the motions for two *tasks* with those obtained from Rowe's method. The results from our method and Rowe's method are quite comparable. The energy-optimal motions offer some interesting insights into the operation of the HEX, while demonstrating the power of our method in handling different measures in the objective function.

Besides their use in optimal motion computation to improve the productivity and efficiency of hydraulic robots, we offer a possible application of our research as a design tool for hydraulic machine designers. We illustrate this application with an example to demonstrate its utility.

Section 6.2: Conclusions

The goal of this research was to develop an approach to computing globally optimal motions for hydraulic robots. We have achieved that goal, and have demonstrated our approach by computing time and energy optimal motions for a retro-fitted Caterpillar 325A HEX robot. It can be seen from the results (Chapter 5) that our approach to optimal motion planning is feasible, and produces motions that compare well to those obtained from other sources, such as a human expert operator.

Although the approach looks promising, we need to qualify the choices made along the path to the solution; choices that may restrict the use of our approach for other applications and robots. We began the research, and this thesis document, with a discussion of the issues that made our problem unique (Chapter 1). We now revisit those issues, and talk about how we have addressed those challenges. After a discussion of the issues, we list some other conclusions from this research.

6.2.1: Research issue: Hydraulic system modeling

One of the main issues at the start of this research was the question of constructing computationally inexpensive hydraulic system models. We needed computationally inexpensive models since optimal motion planning required many thousands of robot motion simulations. The hydraulic robot models therefore had to be fast so that the optimal motions could be computed in a reasonable length of time.

From a survey of the state-of-the-art in hydraulic system modeling it was quite apparent that the available methods were too slow. The most common approach is to use a numerical solver to solve the set of equations that model the hydraulic system. This approach can yield accurate results but is too slow for a time-sensitive application. Linear approximations of the non-linear hydraulic system model compromise the accuracy by not capturing the significant actuator interactions, that are typical of hydraulic robots.

Our approach solved this problem by using non-linear function approximators (regression-based methods and neural networks) to approximate the non-linear response functions for each of the robot's actuators. Such an approximation is possible under the steady-state assumption, i.e. the behavior of the robot is in steady-state (no transients).

This modeling approach is able to model the overall HEX response (including actuator interactions) quite accurately due to the fact that the HEX response is dominated by its steady-state behavior. The modeling approach therefore can be used for other robots that demonstrate the same characteristic, i.e. the steady-state response is a significant part of the overall response.

Hydraulic robots are generally large and heavy since they are used for high force applications; they are also damped since the flow of hydraulic oil through orifices is generally turbulent, which results in energy dissipation. Thus, the joints are akin to a spring-mass-damper system. For such a system, higher mass and damping reduce the fundamental vibration frequency while higher spring constants increase the fundamental frequency. Our modeling approach is applicable for most hydraulic robots since their mass and damping parameters are significant enough to minimize the transients in the response. The greater the effect of transients, the higher the error when using the steady-state assumption.

6.2.2: Research issue: Working with saturated actuators

Most optimal motion computation approaches developed for electric-drive robot use the method by Bobrow, Dubowsky, and Gibson [Bobrow 85] for computing the time-optimal control to follow a specified path. This method avoids actuator saturation to compute a control that follows the path. The class of *tasks* considered in this research are similar to pick-and-place tasks where only the start and end states are specified. Since the robot is not required to follow a specified path, we allow the robot actuators to be saturated, which results in maximum utilization of the robot's capabilities.

However, working with actuator saturation disallows a search in the space of desired spatial paths, an approach used by existing methods. We address this by searching in the command space of the robot, where each candidate is a temporal sequence of robot commands. We simulate each candidate using a robot model and check the resulting spatial path for collisions. Our search avoids candidates that cause collisions since we assign them a high cost. Our approach thus computes the robot control (temporal sequence of robot commands) to perform the given *task* optimally, while avoiding collisions.

We were able to search in the robot's command space since we assumed the use of joint position controllers for the robot's joints. The robot's command space was therefore the space of joint position commands. The joint controllers also reduced the order of the system; if we had attempted to computed the control for the open-loop robot (i.e. compute the sequence of hydraulic *spool* valve positions), the problem would have been harder since the discretization of the joint commands would have to be finer (resulting in a much longer search vector). Computing the optimal might have been more difficult in that case since the longer search vector would likely have resulted in a labyrinthine cost surface (due to the higher order dynamics of the open-loop robot), which would be difficult to navigate in.

6.2.3: Research issue: Using measures other than time in the objective function

Unlike all previous approaches, our optimal motion computation method allows the use of any quantifiable measure in the objective function. Thus, it is possible to compute motions that minimize the motion of one joint, or minimize the force load on one joint.

The price of this flexibility lies in our black-box approach to the optimal motion planning problem. Since we do not exploit the nature of the equations of our robot model, our search approach is more inefficient than those that exploit the nature of the equations. Thus, our approach would be slower (i.e. take longer) to compute the time-optimal motions for an electric-drive robot than Shiller's [Shiller 91] method, although the quality of the results from both methods would be comparable.

However, it is not possible to easily exploit the hydraulic robot model equations since they are far too complex. Thus, our approach is certainly the best alternative for computing optimal motions for hydraulic robots, but not necessarily for electric-drive robots.

6.2.4: Research issue: Avoiding obstacles

One of the alleviating factors in the nature of the optimal motion planning problem for hydraulic robots was the lack of clutter in the workspace. We exploited this characteristics by assuming that all obstacles in the environment are known at the start of the optimization, and do not move (or change) thereafter, until after the *task* execution.

However, this is only true for the class of *tasks* we considered. It cannot be said to be generally true for all hydraulic robots. There are mining applications where multiple loading machines work simultaneously on a dig face. In such cases, we expect that a strategy such as mutual communication between the robots can keep them from interfering with each other.

6.2.5: Computation time for optimal motion planning

The computational time taken by our implementation of the optimal motion planning may appear to be the greatest cause for concern. The computation for each *task* took about 10 hours when running the optimization on 6 workstations (in parallel). We used multiple workstations since we performed multiple instances of the annealing optimization, and selected the best result.

There are a few ways of overcoming this drawback and reducing the run-time. These are described in Section 5.5. We feel confident that future implementations can be made much faster. It will however be some time (even if we assume that Moore's law of doubling processor speed every 18 months holds) before this could be used in real-time.

Inspite of the long computation times, our approach has a valuable utility for optimal motion computation. Besides our approach, there do not exist any methods capable of providing the information that our optimal motion computation method provides for hydraulic robots. Our approach has utility in off-line optimal motion computation, and as a design tool. Shortening the computation time will only make the approach more attractive for these applications, besides opening up other areas.

6.2.6: Potential application

Our optimal motion planning method can be used as a tool for machine designers. This potential deserves more attention than we have afforded it. A designer could use the planning method to see the effect of design changes on, for instance, a typical *truck-loading* task. It would answer the question: "How does a particular design change affect the peak performance obtainable from the machine?".

Today's designers obtain this information by making the change, constructing a prototype, and having an expert operate the prototype. This approach, besides being expensive and time consuming, offers a mostly qualitative assessment of the design change (based on the operator's comments). Our planning method can provide a quantitative answer the above question. The designer could specify one or more target application *tasks*, and study the effect of the design change on the optimal performance. This would greatly reduce the cost and time for each design cycle.

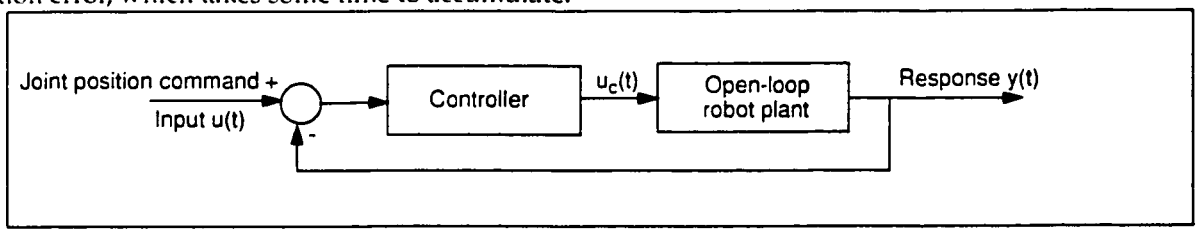
6.2.7: Application to computing optimal paths for mobile robots

The general framework of the proposed method does not disallow its use in computing optimal paths for a mobile robot. However, the challenge lies in the fact that unlike a rotary or linear joint that is limited in its total travel, a mobile base has no such restriction. This greatly widens the space of possibilities, which may make a general search approach impractical. For example, while a joint position is limited between its limits of travel, there is no limit on the position of a mobile robot - it can travel any-

where in its workspace. However, if the space of possibilities was restricted, e.g. the robot is in a fenced area, then it may be possible to use our approach.

6.2.8: Effect of using a low-level controller

Our assumption of the use of a set of joint (position) controllers prevents us from reaching the global optimum in the general case. Pontryagin's principle requires that a time-optimal motion involve the fastest acceleration and deceleration of the joints. The joint controllers necessarily reduce the bandwidth of the open-loop robot plant and disallow very high-frequency inputs to the open-loop plant. For example, if a high step (position) command is input to the closed-loop robot, the open-loop plant input (u_c) will not be a step. The joint controller's output to the open-loop plant is based on the position error, which takes some time to accumulate.



However, the search will attempt to overcome this handicap. It will attempt to compute the u(t) that results in a $u_c(t)$ that is close to the globally optimal input by "fooling" the controller. For instance, the search may come up with a u(t) that issues the motion command to a joint earlier since it will realize the delay between the issuing of the command and the motion of the joint. The limitation in this "fooling" of the controller are the controller input limits. In our controller, the inputs are limited to the joint position limits.

Thus, although the use of a low-level controller significantly aids our approach, it has an associated drawback. This drawback can be minimized by having a responsive controller.

6.2.9: Obstacle buffers

In our examples we did not use any buffers around the obstacles in the environment. However, it is desirable to expand the obstacles by an amount related to the accuracy of the model. In our examples, the motion of the bucket tip was within 0.5 m of the predicted position during all the motions. Hence, increasing the size of the truck by 0.5m in all directions would guarantee that the robot never collided with the truck.

6.2.10: Unmodeled dynamics

The steady-state robot actuator model used in our approach raises the possibility of undesirable dynamic effects (such as joint oscillation) in an optimal motion computed using the steady-state model. To avoid such an occurrence it would be wise to use a slow dynamic model to examine the optimal motion computed. If the computed optimal has some undesirable characteristic then it can either be modified to correct the problem, or a local optimization using the dynamic model can be performed to find a good locally optimal solution that is free of the problem. This local optimization would not require many robot motion simulations and it should be possible to use a slower dynamic model for that purpose.

Section 6.3: Contributions

The primary contribution of this research is in developing a solution to the problem of computing globally optimal motions for hydraulic robots. Although this problem has been an untouched area of study until now, it promises to become increasingly important as hydraulic machines are used in complete or semi-automated applications. Our approach can be used to compute the control for performing specific *tasks* optimally, which could then be used by an automated hydraulic robot. The computation could be done off-line (due to the long computation time) and stored in a lookup table for a range of start and end conditions. These could then be used in real-time to perform the *task* on a robotic machine.

The solution also has use in aiding the designers of hydraulic machines, as mentioned in Section 6.2.6. In either of these roles: as a machine designer's tool or to improve hydraulic robot performance, our approach offers a means of maximizing productivity and efficiency of operation, which are key evaluation metrics for users of such machines.

A secondary contribution of this research is the flexible approach used for solving the optimal motion planning problem. It allows the user the ability to construct an arbitrary objective function to use for the optimization. Although it appears to be a simple notion, it is quite powerful since it greatly extends the use of this research for seemingly remote applications, such as minimizing the force load on a single (over-loaded) joint of a hydraulic robot. All previous work on optimal motion planning has focused almost exclusively on time-optimality.

Another very significant contribution of this research is our effort at drawing attention to the important problem of the complex (non-linear) actuator interactions in hydraulic machines, as well as our approach to efficiently modeling them. This problem has been detected by researchers before, but was avoided [Lawrence 95]. Singh et al. [Singh 95] tried to address the problem using a very simple model for distribution of hydraulic oil between competing actuators. In their distribution model, the lightly loaded actuator always received its maximum flow, and any extra flow was directed to the second actuator. This simple model is not realistic.

The key ideas in our computationally inexpensive approach to modeling the actuator interactions were:

- Making the steady-state approximation, since the effects of the transients in actuator response are small, when a few seconds (or more) of motion is considered.
- Identifying the parameters that define the (steady-state) non-linear response surfaces for the different actuators of the hydraulic robot.
- Using non-linear function approximators to construct approximations of the response surfaces for all the joint actuators.

This approach has proven to be quite powerful at capturing the important actuator interactions. Besides its use in our optimal motion planning, it has other applications. The speed of such a model is an asset that allows it to be used in model-based controllers to help overcome the non-linearities in hydraulic machine operation.

A final contribution of this research is in the demonstration of optimal motion computation on a real hydraulic robot testbed. This included developing robot models, calibrating them to the testbed, and finally demonstrating the computed optimal motions on the testbed for multiple *tasks*. The significance of this contribution lies in the fact that most optimal motion computation is only performed in simulation. Demonstrations on a testbed ensure that the practical issues of working with a real robot have been addressed.

CHAPTER 7 Future work

Our optimal motion planning approach for hydraulic robots offers a number of avenues for further extension.

- The motion planning approach can be implemented for other hydraulic and electric-drive robots. An interesting case-study would be to examine the effectiveness of this approach in optimal motion planning for electric drive robots. We expect our approach to not be as fast and efficient as the time-optimal motion planning methods that have been suggested by various researchers, since those exploit the specific characteristics of electric-drive robots, such as the torque limit curves etc. Our approach is general purpose and does not make assumptions about specific robot characteristics. However, the results in both cases should be comparable.
- Another interesting avenue for future work is to examine the construction of other forms of the objective function. We have used the basic forms of the time and energy cost terms. It is possible that a different combination of the terms in the cost function may yield a more traversable cost surface.
- Dynamic Programming is a promising optimization technique and has been used previously in motion planning [Vukobratovic 82]. Since 1982, a number of researchers have come up with improved implementations of the DP algorithm [Munos 99] and it offers a promising avenue of exploration. The advantage of DP is its ability to quickly compute an optimal motion from any point to any other, once the initial time-consuming step of cost map creation is complete.
- At a more mundane level, there are a few changes which can be made to future implementations which will greatly speed up the motion planning:

- The current implementation is rather inefficient in its use of the robot model. Since more than 95% of the CPU usage during an optimization is for the robot simulation, it is important to make the robot model as efficient as possible. In the current implementation the simulation of a search vector (sequence of commands) is performed using fixed time steps of 0.1 secs. At every time step the robot model function is called, the simulation performed for 0.1 secs, and the new configuration checked for collisions. This process continues until the goal is reached, or a collision is detected. A more efficient alternative would be to use the *interrupt* model, where the simulation continues until an interrupt is issued to stop the simulation. The interrupt would be triggered if the goal is reached or a collision detected.
- The current implementation only allows fixed search vector lengths during an optimization. Adding the ability to change the search vector length and making it a part of the search process would make the search more efficient. It would also be desirable to allow different numbers of knot points to be used for the different joints. Currently, the discretization of the command curve for each joint is the same.

CHAPTER 8

References

[Atkeson 97] Atkeson, C.G., Moore, A.W., Schaal, S.A., *"Locally Weighted Learning"*, AI Review, Vol. 11, pp. 11-73, '97.

[Bazaraa 79] Bazaraa, Mokhtar S., Shetty, C.M., *"Nonlinear Programming: Theory and Algorithms"*, Published by John Wiley and Sons, New York, 1979.

[Bilodeau 97] Bilodeau, G., Papadopoulos, E., *"Modelling, Identification and Experimental Validation of a Hydraulic Manipulator Joint for Control"*, Proceedings of the 1997 IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems (IROS '97), Grenoble, France, Sept 1997, pp. 331-6.

[Bobrow 85] 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, '85.

[Bobrow 88] 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, Aug. '88.

[Bryson 69] Bryson, A.E., Ho, Y.C., *"Applied Optimal Control"*, Blaisdell, New York.

[Castellote 96] 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 '96.

[Centikunt 91] 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, pp. 15-19, '91.

[Craig 89] Craig, John J., *"Introduction to Robotics: Mechanics and Control"*, 2nd Ed., Addison-Wesley Publishing Company, '89.

[Fiorini 96] Fiorini, P, Shiller, Z, *"Time-Optimal Trajectory Planning in Dynamic Environments"*, Proceedings of the IEEE Conference on Robotics and Automation, April '96.

[Gottschalk 96] Gottschalk, S., Lin, M.C., Manocha, D., *"OBB-Tree: A Heirarchical Structure for Rapid Interference Detection"*, Technical Report TR96-013, Department of Computer Science, University of North Carolina, Chapel Hill, NC - 27599-3175.

[Huang 86] 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. '86, pp. 381-384.

[Hwang 92] Hwang, Yong K., Ahuja, Narendra, *"Gross Motion Planning - A Survey"*, ACM Computing Surveys, Vol. 24, No. 3, September 1992.

[Kelly 98] Kelly, A., Stentz, A., *"An Approach to Rough Terrain Autonomous Mobility - Part 1"*, Autonomous Robots, Vol. 5, pp. 129-161, '98.

[Khatib 86] Khatib, O., *"Real time obstacle avoidance for manipulators and mobile robots"*, International Journal of Robotics Research, Vol. 5, No. 1, '86.

[Kirk 70] Kirk, Donald E., *"Optimal Control Theory"*, 1970, Prentice-Hall Inc., Englewood Cliffs, New Jersey.

[Kirkpatrick 83] Kirkpatrick, S., et al., *"Optimization by Simulated Annealing"*, Science, Vol. 220, No. 4598, pp. 671-680, May '83.

[Koivo 96] Koivo, A.J., Nease, A.D., *"Adaptive Self-Tuning Control of Excavators"*, Proc. of the Conf. on Robotics for Challenging Environments, pp. 220-6, June 1-6 '96, Pub. by the ASCE.

[Krishna 98] Krishna, M., Bares, J., *"Hydraulic System Modeling through Memory-based Learning"*, Proc. IEEE Conf. on Intelligent Robots and Systems, (IROS), Victoria, B.C., Canada, Oct '98, pp. 1733-8.

[Krishna 99a] Krishna, M., Bares, J., *"Constructing Hydraulic Robot Models using Memory-Based Learning"*, ASCE Journal of Aerospace Engg., April '99, Vol. 12, No. 2, pp. 34-42.

[Krishna 99b] Krishna, M., Bares, J., "Constructing Fast Hydraulic Robot Models for Optimal Motion Planning", Proc. of the Field and Service Robotics Conf., Pittsburgh, PA, August 1999, pp. 220-5.

[Kuntze 95] Kuntze, H.B., et al., *"On the dynamic control of a hydraulic large range robot for construction applications"*, Automation in Construction, Vol. 4, pp. 61-73, 1995, Published by Elsevier Science B.V.

[Lam 88] Lam, J., *"An Efficient Simulated Annealing Schedule"*, Ph.D. dissertation, Department of Computer Science, Yale University, Fall 1988.

[Lawrence 95] Lawrence, P.D., et al., *"Coordinated and Force-Feedback Control of Hydraulic Excavators"*, Fourth International Symposium on Experimental Robotics, Proc. ISER '95, Pub. by Springer-Verlag, June 30 - July 2 '95.

[Lewis 93] Lewis, F.L., Abdallah, C.T., Dawson, D.M., *"Control of Robot Manipulators"*, 1993, MacMillan Publishing Company, 866, Third Ave., New York, NY - 10022.

[Li 97] 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, Albuquerque, N.M., pp. 1663-8, April '97.

[Lin 96] 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, pp. 595-602, Aug. 1996.

[Martin 97] 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, pp. 2044-9, April 1997.

[McDonell 97] McDonell, B.W., Bobrow, J.E., *"Modeling, Identification, and Control of a Pneumatically Actuated Robot"*, Proceedings of the IEEE Conference on Robotics and Automation, pp. 124-9, April 1997.

[Medanic 97] Medanic, J., Yuan, M., Medanic, B., *"Robust Multivariable Non-linear Control of a Two Link Excavator: Part I"*, Proc. IEEE Conference on Decision and Control Systems, pp. 4231-6, Dec 1997.

[Merritt 67] Merritt, H.E., *"Hydraulic Control Systems"*, Wiley, New York, 1967.

[Moore 97a] Moore, A., Atkeson, C., Schaal, S., *"Locally Weighted Learning for Control"*, AI Review, Vol. 11, pp. 75-113, 1997.

[Moore 97b] Moore, A.W., Schneider, J., Deng, K., *"Efficient Locally Weighted Polynomial Regression Predictions"*, Proc. 1997 Intl. Machine Learning Conf., Morgan Kaufmann Publishers, San Francisco, CA.

[Munos 99] Munos, Remi, Moore, A.W., *"Variable resolution discretization for high-accuracy solution for optimal control problems"*, To appear in Proc of Int'l Joint Conf. on Artificial Intelligence, 1999.

[Narendra 90] Narendra, K.S., Parthasarathy, K., *"Identification and control of dynamical systems using neural networks"*, IEEE Tran. on Neural Networks, Vol. 1, pp. 4-27, 1990.

[Ochotta 95] Ochotta, E., Mukherjee, T., *"Programmers Guide to the Reconfigurable Simulated Annealing Library"*, 1995.

[Ochotta 94] Ochotta, E., *"Synthesis of High-Performance Analog Cells in ASTRX/OBLX"*, Ph.D. thesis, Research Report No. CMUCAD-94-17, February 1994.

[Ogata 90] Ogata, Katsushiko, *"Modern Control Engineering"*, 2nd Ed., Prentice Hall, Englewood Cliffs, N.J., 1990.

[Press 96] Press, W.H., et al., *"Numerical Recipes in C - The Art of Scientific Computing"*, Second Edition, 1996, Cambridge University Press.

[Rajan 85] Rajan, V.T., *"Minimum Time Trajectory Planning"*, Proceedings of the IEEE Conference on Robotics and Automation, St. Louis, MO, pp. 759-764, '85.

[Rowe 97] 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, Sep. '97.

[Rowe 99] Rowe, Patrick, *"Adaptive Motion Planning for Mass Excavation"*, Ph.D. dissertation, Technical Report CMU-RI-TR-99-09, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA - 15213.

[Russell 95] Russell, Stuart J., Norvig, Peter, *"Artificial Intelligence: A Modern Approach"*, 1995, published by Prentice-Hall Inc., Upper Saddle River, New Jersey - 07458.

[Rutenbar 89] Rutenbar, Rob A., *"Simulated Annealing Algorithms: An Overview"*, IEEE Circuits and Devices Magazine, January 1989, pp. 19-26.

[Schneider 96] Schneider, J., Moore, A.W., *"A Locally Weighted Learning Tutorial using Vizier 1.0"*, 1996.

[Shiller 91] 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, Dec. '91.

[Shin 85] Shin, K.G. and McKay, N.D., *"Minimum-time Control of Robotic Manipulators with Geometric Path Constraints"*, IEEE Transactions on Automatic Control, Vol AC-30, No. 6, pp. 531-541, June '85.

[Singh 95] Singh, N., et al., *"Coordinated Motion Control of Heavy Duty Industrial Machines with Redundancy"*, Robotica, Vol. 13, pp. 623-633, '95

[Slotine 89] 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 '89.

[Song 95] Song, B., Koivo, A.J., *"Neural Adaptive Control of Excavators"*, Proceedings of the Intelligent Robots and Systems Conference, Pittsburgh, PA, pp. 162-7, April 1995.

[Swartz 90] Swartz, W., Sechen, C., *"New Algorithms for the placement and routing of Macrocells"*, Proc. of the IEEE/ACM International Conf. on Computer-Aided Design, November, 1990, pp. 336-9.

[Tafazoli 96] Tafazoli, S., Lawrence, P.D., et al., *"Parameter Estimation and Actuator Friction Analysis for a Mini-Excavator"*, Proc., of the IEEE Int'l Conf. on Robotics and Automation, Minneapolis, Minnesota, April 1996, pp. 329-334.

[Vaha 91] Vaha, P.K., Skibniewski, M.J., Koivo, A.J., *"Kinematics and Trajectory Planning for Robotic Excavation"*, Proc. ASCE Construction Congress II, Cambridge, MA, 1991.

[Vaha 93] Vaha, P.K., Skibniewski, M.J., *"Dynamic Model of Excavator"*, ASCE Journal of Aerospace Engineering, Vol. 6, No. 2, April 1993.

[Vukobratovic 82] Vukobratovic, M., Kircanski, M., *"A Method for Optimal Synthesis of Manipulation Robot Trajectories"*, Journal of Dynamic Systems, Measurement, and Control, Vol. 104, pp. 188-193, June '82.

[Wapenhans 94] Wapenhans, H., et al., *"Optimal Trajectory Planning with Application to Industrial Robots"*, International Journal of Advanced Manufacturing, Vol. 9, No. 1, pp. 49-55, '94.

[White 84] White, S., *"Concepts of Scale in Simulated Annealing"*, IEEE International Conference on Computer Design, Port Chester, New York, pp 646-651, '84.

[Yao 98] Yao, Bin, Zhang, Jiao, et al., *"High Performance Swing Velocity Tracking Control of Hydraulic Excavators"*, Proc. of the American Controls Conference, 1998, pp. 818-22, Vol. 2.

[Zhang 94] Zhang, J., Herp, A., *"Design of a Fuzzy Controller for Optimal Execution of Subgoal-Guided Robot Motions"*, Proceedings of the IEEE Conf. on Robotics and Automation, pp. 2986-2991, '94.

[Zomaya 95] Zomaya, A.Y., *"An Error-learning Neural Network for Tuning of Robot Dynamic Models"*, International Journal of Systems Science, Vol. 26, No. 1, pp. 13-31, 1995.

CHAPTER 9

Glossary

HEX	A hydraulic excavator.
CAT 325 HEX	A 25-ton hydraulic excavator manufactured by Caterpillar Inc., called the model 325.
Closed-loop HEX	A HEX with joint position controllers.
Truck-loading operation	An operation where trucks are continuously loaded by a digging machine (See Figure 1, Chapter 1).
Fine-finish operation	An operation such as smoothing the sides of a pile of soil, which does not requires high forces but requires good control of the implement.
Mass excavation operation	This refers to an excavation operation where large amounts of soil are excavated and moved using a continuous cycle of trucks, and the truck-loading typically is done 24 hrs. a day for weeks. Such an operation is usually for a large scale project such as an airport, highway etc.
Truck-loading, Fine-finish mode	These are two commonly used HEX operating modes. The hydraulic oil flow routing is different in the two cases due to the different requirements of a truck-loading and fine-finish opera-

tion. These modes are also referred to as the FF and TL modes. See Section 3.1 (page 29)

Loading configuration — An arrangement of the HEX, truck and terrain in a truck-loading operation.

Loading pass — A section of a truck loading operation consisting of digging a bucket load of soil, unloading it in a truck, and returning to the dig face for the next dig.

Hydraulic spool valve — A *spool* valve is used to control the hydraulic flow to a hydraulic actuator (cylinder or a motor). On our HEX testbed, solenoids control the position of the *spool* valves for the different joints. The position of a *spool* valve is the control input to the open-loop HEX plant. See Figure 11 and Figure 8 (Chapter 3) for more details.

Cross-over valves — These are special valves that are used in the truck-loading mode. They re-direct flow to allow more efficient use of the available energy. (Ref Figure 13 on page 30).

Task — A task is defined by specifying the start and end HEX positions, the location of the truck (if relevant), the load in the bucket, and a terrain map (for collision avoidance). See *Chapter 5* for more details.

Seed Search Vector — The initial instance of the search vector used to seed the exploration of the search space.

Comparison motion — This term, and the next two, are used in Chapter 5. It refers to the HEX motion performed by the human expert, or by the implementation of Rowe's method. These motions are used to compare with the results of the optimal motion planning approach described in this document.

Simulated optimal motion — The HEX response, as predicted by the HEX robot model, to the optimal HEX joint command vector obtained at the end of the optimization.

Real optimal motion — The motion that results when the optimal command vector is fed to the closed-loop HEX.

Implement energy, Engine energy	These two measures of energy consumption are used in our implementation. For details refer the section on *Energy term on page 63* in *Chapter 4*.

CHAPTER 10

Appendix A: Details of HEX linkage dynamic equations

The coordinate axes for the HEX are shown in Figure 72.

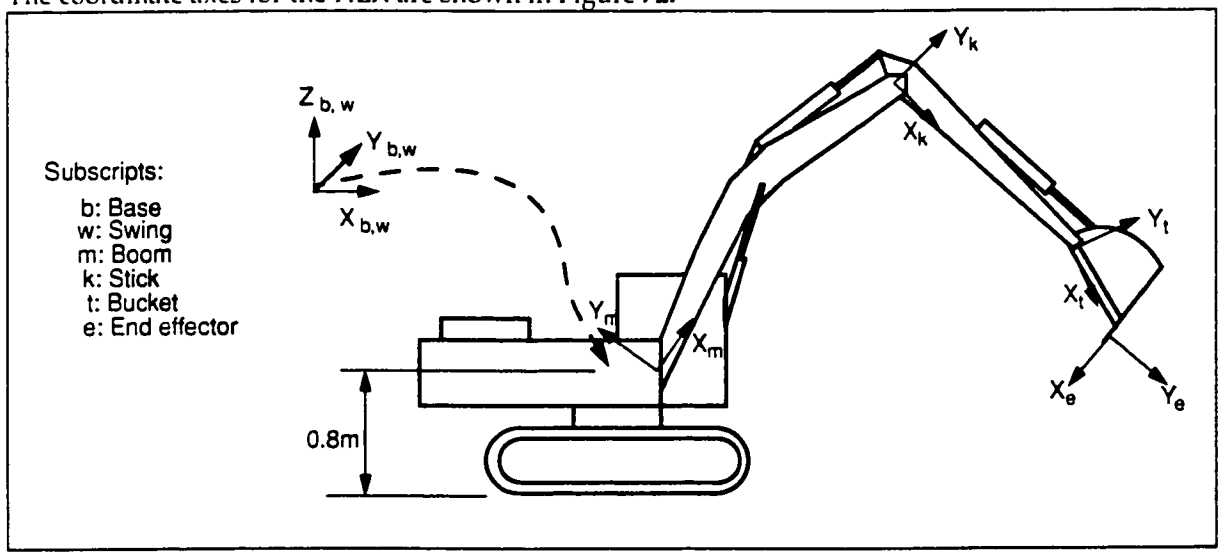


FIGURE 72. HEX coordinate axes

The z-axes of all joint frames are coincident with the axes of rotation of the joint. The base frame does not move relative to the ground; hence the x and y axes of the swing are only coincident with the x and y axes of the base joint when the swing rotation angle is zero. The base and swing coordinate axes have coincident origins and z-axes. The naming convention used above is consistent with the Denavit-Hartenburg convention[Craig 89].

The transformation matrices between the different coordinate frames are given below. In each equation c_i stands for "cosine of joint angle i", and s_i stands for "sine of joint angle i". The indices 1, 2, 3, 4 refer to the swing, boom, stick and bucket joints respectively.

$$T_{w \to b} = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(EQ 34)}$$

$$T_{m \to w} = \begin{bmatrix} c_2 & -s_2 & 0 & l_1 \\ 0 & 0 & -1 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(EQ 35)}$$

where $l_1 = swingLinkLength = 0.165m = distFromSwingOriginToBoomOrigin$.

$$T_{t \to m} = \begin{bmatrix} c_3 & -s_3 & 0 & l_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(EQ 36)}$$

where $l_2 = boomLinkLength = 6.15m$.

$$T_{k \to t} = \begin{bmatrix} c_4 & -s_4 & 0 & l_3 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(EQ 37)}$$

where $l_k = stickLinkLength = 2.65m$.

The general form of the linkage dynamics for the HEX open-chain manipulator arm is:

$$M(q)\ddot{q} + V(q,\dot{q})\dot{q} + G(q) = \Upsilon_{actuator} + \Upsilon_{external} + \Upsilon_{friction} \quad \text{(EQ 38)}$$

Here, q is the vector of joint angles, $M(q)$ is the mass (or inertia) matrix, $V(q,\dot{q})$ is the matrix of non-linear coriolis and centripetal terms, and $G(q)$ is the vector of gravity torques. The vector of joint torques, $\Upsilon_{actuator}$, is the torque applied at the joints by the hydraulic cylinders and swing motor. In the class of tasks we examine, there is no forceful interaction between the HEX and the environment and hence the vector $\Upsilon_{external}$ is zero. The vector of friction torques, $\Upsilon_{friction}$, is neglected in our analysis.

Section 10.1: Mass Matrix

The inertia (or mass) matrix $M(q)$ is a 4X4 positive definite symmetric matrix defined as:

$$m_{jk}(q) = \sum_{i=max(j,k)}^{n} Tr\left[\frac{\partial T_i}{\partial q_j} \cdot I_i \cdot \frac{\partial T_i^T}{\partial q_k}\right] \quad \text{(EQ 39)}$$

where $j, k \in (1, 2, 3, 4)$ (Ref [Lewis 93], pp. 75). The q_i; i=1,2,3,4, refer to the swing, boom, stick and bucket joint angles respectively. The Tr operator is the matrix trace operator, and the T_i (i=1, 2, 3, 4) matrices are the transformation matrices defined as:

$$T_1 = T_{w \to b} \quad \text{(EQ 40)}$$

$$T_2 = T_{m \to b} = T_{w \to b} \cdot T_{m \to w} \quad \text{(EQ 41)}$$

$$T_3 = T_{t \to b} = T_{w \to b} \cdot T_{m \to w} \cdot T_{t \to m} \quad \text{(EQ 42)}$$

$$T_4 = T_{k \to b} = T_{w \to b} \cdot T_{m \to w} \cdot T_{t \to m} \cdot T_{k \to t} \quad \text{(EQ 43)}$$

The I_i; i = 1, 2, 3, 4, are the pseudo-inertia matrices for the swing, boom, stick and bucket joints. These are defined as:

$$I_i = \begin{bmatrix} \frac{-I_{xx_i} + I_{yy_i} + I_{zz_i}}{2} & I_{xy_i} & I_{xz_i} & m_i\bar{x}_i \\ I_{xy_i} & \frac{-I_{xx_i} + I_{yy_i} + I_{zz_i}}{2} & I_{yz_i} & m_i\bar{y}_i \\ I_{xz_i} & I_{yz_i} & \frac{-I_{xx_i} + I_{yy_i} + I_{zz_i}}{2} & m_i\bar{z}_i \\ m_i\bar{x}_i & m_i\bar{y}_i & m_i\bar{z}_i & m_i \end{bmatrix} \quad \text{(EQ 44)}$$

where m_i is the mass of link i, $\bar{x}_i$, $\bar{y}_i$, $\bar{z}_i$ define the location of the center of gravity of link i in the coordinate frame of link i, I_{xx_i}, I_{yy_i}, I_{zz_i} are the moments of inertia of link i, and I_{xy_i}, I_{yz_i}, I_{xz_i} are the cross-products of inertia. The centers of gravity and moments of inertia for all four links are given in the table below.

	Swing	Boom	Stick	Bucket
Link mass (kgs)	11230	2040	946	1200
I_{xx} (N-m-s^2)	8000	475.6	113	491
I_{yy} (N-m-s^2)	22000	8875	994	371
I_{zz} (N-m-s^2)	14000	9208	1084	412
I_{xy} (N-m-s^2)	0	418	131	85
I_{yz} (N-m-s^2)	0	0	0	0
I_{xz} (N-m-s^2)	0	0	0	0
$\bar{x}$ (mm)	-1930	3342	861	700
$\bar{y}$ (mm)	1500	700	322	353
$\bar{z}$ (mm)	0	0	0	0

TABLE 10. Table of HEX linkage parameters

NOTE: The centers of gravity are given in the coordinate frames attached to the respective links.

The partial derivative terms in Equation 39 are given below.

$$\frac{\partial T_1}{\partial q_1} = \begin{bmatrix} -s_1 & -c_1 & 0 & 0 \\ c_1 & -s_1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{(EQ 45)}$$

$$\frac{\partial T_2}{\partial q_1} = \begin{bmatrix} -s_1 c_2 & s_1 s_2 & c_1 & -s_1 l_1 \\ c_1 c_2 & -c_1 s_2 & s_1 & c_1 l_1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{(EQ 46)}$$

$$\frac{\partial T_2}{\partial q_2} = \begin{bmatrix} -c_1 s_2 & -c_1 c_2 & 0 & 0 \\ -s_1 s_2 & -s_1 c_2 & 0 & 0 \\ c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{(EQ 47)}$$

$$\frac{\partial T_3}{\partial q_1} = \begin{bmatrix} -s_1 c_{23} & s_1 s_{23} & c_1 & -s_1(c_2 l_2 + l_1) \\ c_1 c_{23} & -c_1 s_{23} & s_1 & c_1(c_2 l_2 + l_1) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{(EQ 48)}$$

$$\frac{\partial T_3}{\partial q_2} = \begin{bmatrix} -c_1 s_{23} & -c_1 c_{23} & 0 & -c_1 s_2 l_2 \\ -s_1 s_{23} & -s_1 c_{23} & 0 & -s_1 s_2 l_2 \\ c_{23} & -s_{23} & 0 & c_2 l_2 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{(EQ 49)}$$

$$\frac{\partial T_3}{\partial q_3} = \begin{bmatrix} -c_1 s_{23} & -c_1 c_{23} & 0 & 0 \\ -s_1 s_{23} & -s_1 c_{23} & 0 & 0 \\ c_{23} & -s_{23} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{(EQ 50)}$$

$$\frac{\partial T_4}{\partial q_1} = \begin{bmatrix} -s_1 c_{234} & s_1 s_{234} & c_1 & -s_1(c_{23}l_3 + c_2 l_2 + l_1) \\ c_1 c_{234} & -c_1 s_{234} & s_1 & c_1(c_{23}l_3 + c_2 l_2 + l_1) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{(EQ 51)}$$

$$\frac{\partial T_4}{\partial q_2} = \begin{bmatrix} -c_1 s_{234} & -c_1 c_{234} & 0 & -c_1(s_{23}l_3 + s_2 l_2) \\ -s_1 s_{234} & -s_1 c_{234} & 0 & -s_1(s_{23}l_3 + s_2 l_2) \\ c_{234} & -s_{234} & 0 & c_{23}l_3 + c_2 l_2 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{(EQ 52)}$$

$$\frac{\partial T_4}{\partial q_3} = \begin{bmatrix} -c_1 s_{234} & -c_1 c_{234} & 0 & -c_1 s_{23} l_3 \\ -s_1 s_{234} & -s_1 c_{234} & 0 & -s_1 s_{23} l_3 \\ c_{234} & -s_{234} & 0 & c_{23} l_3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{(EQ 53)}$$

$$\frac{\partial T_4}{\partial q_4} = \begin{bmatrix} -c_1 s_{234} & -c_1 c_{234} & 0 & 0 \\ -s_1 s_{234} & -s_1 c_{234} & 0 & 0 \\ c_{234} & -s_{234} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{(EQ 54)}$$

Section 10.2: Vector of non-linear terms $V(q, \dot{q})$

The 4X4 matrix of non-linear coriolis and centripetal terms is given by:

$$v_{kj} = \sum_{i=1}^{n} \frac{1}{2}\left\{\frac{\partial m_{kj}}{\partial q_i} + \frac{\partial m_{ki}}{\partial q_j} + \frac{\partial m_{ij}}{\partial q_k}\right\}\dot{q}_i \quad \text{(EQ 55)}$$

where v_{kj}; $k, j \in (1, 2, 3.4)$ are the terms in the $V(q, \dot{q})$ matrix, m_{cd} are the (c,d) terms of the mass matrix. From Equation 39 it follows that:

$$\frac{\partial m_{cd}}{\partial q_i} = Tr\left[\frac{\partial^2 T_d}{\partial q_i \partial q_c} I_d \frac{\partial T_d}{\partial q_d} + \frac{\partial T_d}{\partial q_d} I_d \frac{\partial^2 T_d}{\partial q_i \partial q_d}^T\right] \quad \text{(EQ 56)}$$

The first derivative terms in the above equations have been given in the previous section (on the Mass matrix terms). The second derivative terms are:

$$\frac{\partial^2 T_1}{\partial q_1^2} = \begin{bmatrix} -c_1 & s_1 & 0 & 0 \\ -s_1 & -c_1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{(EQ 57)}$$

$$\frac{\partial^2 T_2}{\partial q_1^2} = \begin{bmatrix} -c_1 c_2 & c_1 s_2 & -s_1 & -c_1 l_1 \\ -s_1 c_2 & s_1 s_2 & c_1 & -s_1 l_1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{(EQ 58)}$$

$$\frac{\partial^2 T_2}{\partial q_1 \partial q_2} = \begin{bmatrix} s_1 s_2 & s_1 c_2 & 0 & 0 \\ -c_1 s_2 & -c_1 c_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \frac{\partial^2 T_2}{\partial q_2 \partial q_1} \quad \text{(EQ 59)}$$

$$\frac{\partial^2 T_2}{\partial q_2^2} = \begin{bmatrix} -c_1 c_2 & c_1 s_2 & 0 & 0 \\ -s_1 c_2 & s_1 s_2 & 0 & 0 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{(EQ 60)}$$

$$\frac{\partial^2 T_3}{\partial q_1^2} = \begin{bmatrix} -c_1 c_{23} & c_1 s_{23} & -s_1 & -c_1(c_2 l_2 + l_1) \\ -s_1 c_{23} & s_1 s_{23} & c_1 & -s_1(c_2 l_2 + l_1) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{(EQ 61)}$$

$$\frac{\partial^2 T_3}{\partial q_1 \partial q_2} = \begin{bmatrix} s_1 s_{23} & s_1 c_{23} & 0 & s_1 s_2 l_2 \\ -c_1 s_{23} & -c_1 c_{23} & 0 & -c_1 s_2 l_2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \frac{\partial^2 T_3}{\partial q_2 \partial q_1} \qquad \text{(EQ 62)}$$

$$\frac{\partial^2 T_3}{\partial q_1 \partial q_3} = \begin{bmatrix} s_1 s_{23} & s_1 c_{23} & 0 & 0 \\ -c_1 s_{23} & -c_1 c_{23} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \frac{\partial^2 T_3}{\partial q_3 \partial q_1} \qquad \text{(EQ 63)}$$

$$\frac{\partial^2 T_3}{\partial q_2^2} = \begin{bmatrix} -c_1 c_{23} & c_1 s_{23} & 0 & -c_1 c_2 l_2 \\ -s_1 c_{23} & s_1 s_{23} & 0 & -s_1 c_2 l_2 \\ -s_{23} & -c_{23} & 0 & -s_2 l_2 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{(EQ 64)}$$

$$\frac{\partial^2 T_3}{\partial q_2 \partial q_3} = \begin{bmatrix} -c_1 c_{23} & c_1 s_{23} & 0 & 0 \\ -s_1 c_{23} & s_1 s_{23} & 0 & 0 \\ -s_{23} & -c_{23} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \frac{\partial^2 T_3}{\partial q_3 \partial q_2} = \frac{\partial^2 T_3}{\partial q_3^2} \qquad \text{(EQ 65)}$$

$$\frac{\partial^2 T_4}{\partial q_1^2} = \begin{bmatrix} -c_1 c_{234} & c_1 s_{234} & -s_1 & -c_1(c_{23} l_3 + c_2 l_2 + l_1) \\ -s_1 c_{234} & s_1 s_{234} & c_1 & -s_1(c_{23} l_3 + c_2 l_2 + l_1) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \text{(EQ 66)}$$

$$\frac{\partial^2 T_4}{\partial q_1 \partial q_2} = \begin{bmatrix} s_1 s_{234} & s_1 c_{234} & 0 & s_1(s_{23} l_3 + s_2 l_2) \\ -c_1 s_{234} & -c_1 c_{234} & 0 & -c_1(s_{23} l_3 + s_2 l_2) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \frac{\partial^2 T_4}{\partial q_2 \partial q_1} \qquad \text{(EQ 67)}$$

$$\frac{\partial^2 T_4}{\partial q_1 \partial q_3} = \begin{bmatrix} s_1 s_{234} & s_1 c_{234} & 0 & s_1 s_{23} l_3 \\ -c_1 s_{234} & -c_1 c_{234} & 0 & -c_1 s_{23} l_3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \frac{\partial^2 T_4}{\partial q_3 \partial q_1} \tag{EQ 68}$$

$$\frac{\partial^2 T_4}{\partial q_2^2} = \begin{bmatrix} -c_1 c_{234} & c_1 s_{234} & 0 & -c_1(c_{23} l_3 + c_2 l_2) \\ -s_1 c_{234} & s_1 s_{234} & 0 & -s_1(c_{23} l_3 + c_2 l_2) \\ -s_{234} & -c_{234} & 0 & -s_{23} l_3 - s_2 l_2 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{EQ 69}$$

$$\frac{\partial^2 T_4}{\partial q_2 \partial q_3} = \begin{bmatrix} -c_1 c_{234} & c_1 s_{234} & 0 & -c_1 c_{23} l_3 \\ -s_1 c_{234} & s_1 s_{234} & 0 & -s_1 c_{23} l_3 \\ -s_{234} & -c_{234} & 0 & -s_{23} l_3 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \frac{\partial^2 T_4}{\partial q_3 \partial q_2} = \frac{\partial^2 T_4}{\partial q_3^2} \tag{EQ 70}$$

$$\frac{\partial^2 T_4}{\partial q_2 \partial q_4} = \begin{bmatrix} -c_1 c_{234} & c_1 s_{234} & 0 & 0 \\ -s_1 c_{234} & s_1 s_{234} & 0 & 0 \\ -s_{234} & -c_{234} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \frac{\partial^2 T_4}{\partial q_4 \partial q_2} = \frac{\partial^2 T_4}{\partial q_3 \partial q_4} = \frac{\partial^2 T_4}{\partial q_4 \partial q_3} = \frac{\partial^2 T_4}{\partial q_4^2} \tag{EQ 71}$$

Section 10.3: Gravity terms

The last term on the left side in Equation 38 is the vector of gravity terms. The elements of that vector are:

$$G(q) = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} \tag{EQ 72}$$

$$g_1 = 0 \tag{EQ 73}$$

$$g_2 = g[m_1(\bar{x}_1 c_2 - \bar{y}_1 s_2) + m_2((c_{23}\bar{x}_3 - s_{23}\bar{y}_3) + c_2 l_2) + m_4((c_{234}\bar{x}_4 - s_{234}\bar{y}_4) + c_{23} l_3 + c_2 l_2)] \quad \text{(EQ 74)}$$

where g is the gravitational constant (9.8 m/s^2), and $(\bar{x}_i, \bar{y}_i)$ is the center of gravity of link i, in the coordinate frame of link i.

$$g_3 = g[m_3(c_{23}\bar{x}_3 - s_{23}\bar{y}_3) + m_4((c_{234}\bar{x}_3 - s_{234}\bar{y}_3) + c_{23} l_3)] \quad \text{(EQ 75)}$$

$$g_4 = g[m_4(c_{234}\bar{x}_4 - s_{234}\bar{y}_4)] \quad \text{(EQ 76)}$$

Section 10.4: Actuator torque vector $\Upsilon_{actuator}$

The torque vector on the right side of Equation 38 contains the torques applied on the joints by the different hydraulic actuators. The swing joint has a motor that applies torque to the swing joint. The boom, stick, and bucket joints are actuated by hydraulic cylinders; the linear motion is transformed into rotary motion at the joint. The relationships that transform cylinder force to joint torque for the boom, stick, and bucket joints are given below.

10.4.1 : Boom cylinder force to joint torque transform:

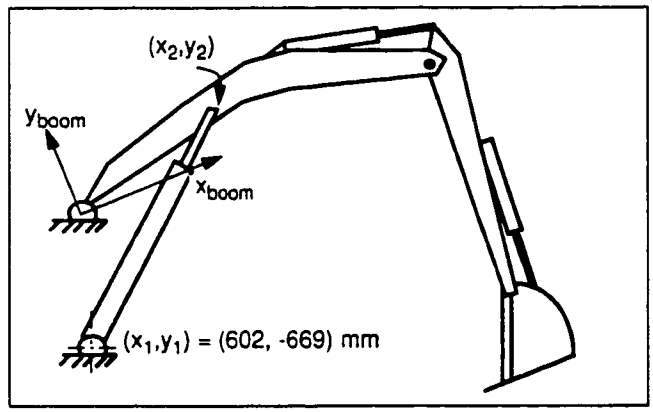


FIGURE 73. Figure showing boom joint control points (Not to scale)

From the above figure we get the relationship between boom cylinder force and joint torque as:

$$\tau_{boom} = F_{boom}\frac{(x_2 \cdot \Delta y - \Delta x \cdot y_2)}{\sqrt{\Delta x^2 + \Delta y^2}} \qquad \text{(EQ 77)}$$

where $\Delta x = x_2 - x_1$; $\Delta y = y_2 - y_1$, and $x_2 = 2736 \cdot \cos(q_2 + 21.8^o)$; $y_2 = 2735 \cdot \sin(q_2 + 21.8^o)$. All lengths are in *mm*, forces are in *N*, and torques are in *N-mm*.

10.4.2 : Stick cylinder force to joint torque:

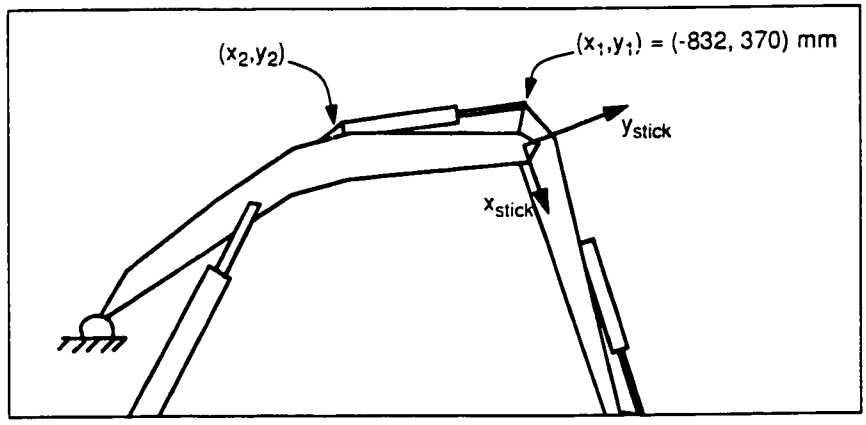


FIGURE 74. Figure showing stick joint control points (Not to scale)

The relationship between stick cylinder force and joint torque is:

$$\tau_{stick} = F_{stick}\frac{(x_1 \cdot \Delta y - y_1 \cdot \Delta x)}{\sqrt{\Delta x^2 + \Delta y^2}} \qquad \text{(EQ 78)}$$

where $\Delta x = x_1 - x_2$; $\Delta y = y_1 - y_2$, and $x_2 = -2632 \cdot \cos(q_3) - 1217 \cdot \sin(q_3)$;
$y_2 = -2632 \cdot \sin(q_3) + 1217 \cdot \cos(q_3)$. All lengths are in *mm*, forces are in *N*, and torques are in *N-mm*.

10.4.3 : Bucket cylinder force to joint torque:

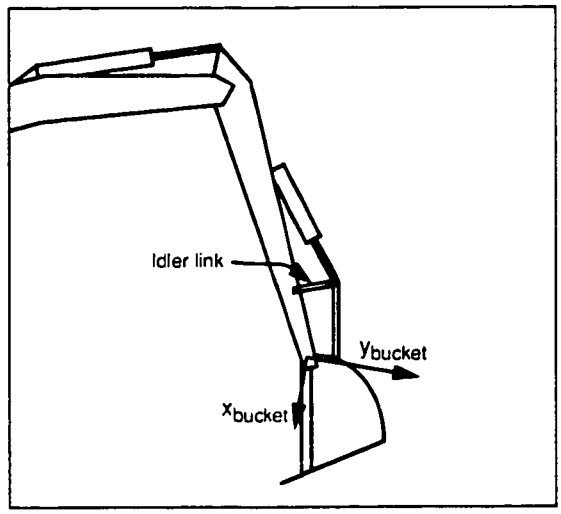


FIGURE 75. Figure showing bucket joint details (Not to scale)

The analysis of the bucket mechanism is complicated by the use of the idler link. The idler greatly improves the range of motion of the bucket but makes the geometry rather complicated. The relationship between the bucket cylinder force and joint torque is:

$$\tau_{bkt} = F_{bkt} \cdot [512 \cdot (n_{2x} \cdot \sin(95.5^o + q_4) - n_{2y} \cdot \cos(95.5^o + q_4))] \cdot \frac{(n_{1x} \cdot \sin(q) - n_{1y} \cdot \cos(q))}{(n_{2x} \cdot \sin(q) - n_{2y} \cdot \cos(q))} \quad \text{(EQ 79)}$$

where

$$n_{1x} = \frac{2214 + 685 \cdot \cos(q)}{\sqrt{2355^2 + 685^2 - 2 \cdot 2355 \cdot 685 \cdot \cos(160^o - q)}} \quad \text{(EQ 80)}$$

$$n_{1y} = \frac{-804 + 685 \cdot \sin(q)}{\sqrt{2355^2 + 685^2 - 2 \cdot 2355 \cdot 685 \cdot \cos(160^o - q)}} \quad \text{(EQ 81)}$$

$$n_{2x} = \frac{512 \cdot \cos(95.5^o + q_4) - 685 \cdot \cos(q) + 480}{640} \quad \text{(EQ 82)}$$

$$n_{2y} = \frac{512 \cdot \sin(95.5^{o} + q_4) - 685 \cdot \sin(q) - 35}{640} \quad \text{(EQ 83)}$$

$$q = \operatorname{asin}\left(\frac{512}{A}\sin(80^{o} - q_4)\right) + \operatorname{acos}\left(\frac{685^2 + A^2 - 640^2}{2 \cdot 685 \cdot A}\right) - 5^{o} \quad \text{(EQ 84)}$$

$$A = \sqrt{481^2 + 512^2 - 2 \cdot 481 \cdot 512 \cdot \cos(80^{o} - q_4)} \quad \text{(EQ 85)}$$

CHAPTER 11

Appendix B: Robot modeling results

Aside from the plots shown in Chapter 3, we have conducted more tests in the *TL* HEX operating mode to explore the performance of the model. These tests are similar to those shown in Chapter 3 (Section 3.4.2) but differ in the directions of motion of the Bm, St, and Bk joints. The tests are summarized in which shows the direction of motion for the different joints. The 8 tests - 7 in this appendix and 1 in Chapter 3 - involving simultaneous Bm, St, and Bk motions use the 8 (= 2^3) permutations possible in the directions of Bm, St, and Bk joints. Our model only uses one direction of motion of the Sw since it is symmetric, i.e. both directions of swing motion are identical (assuming that the HEX is on a flat surface). Recall that the *TL* mode actuator model divided each joint space into 8 sections and used a neural network for each section (See Section 3.2.5 for more details). The tests shown in this chapter characterize 7 of those 8 sections; the 8th section is characterized by the tests in Chapter 3.

In addition to the Bm-St-Bk tests, we also have a Sw-St test, corresponding to negative St motion (collapsing St cylinder) and positive Sw motion (counter-clockwise rotation of the swing joint). Test #2 shown in Section 3.4.2 (*TL* mode tests) used positive St motion and positive Sw motion. This Sw-St test is shown in the first set of figures, Figure 76 and Figure 77. The list of all the tests is summarized in Table 11.

Bm Direction	St Direction	Bk Direction	Figures showing the results for corresponding Bm-St-Bk test
+	+	+	Figure 22, Figure 23 (Chapter 3)
+	+	-	Figure 78, Figure 79
+	-	+	Figure 80, Figure 81
+	-	-	Figure 82, Figure 83
-	+	+	Figure 84, Figure 85
-	+	-	Figure 86, Figure 87
-	-	+	Figure 88, Figure 89
-	-	-	Figure 90, Figure 91

TABLE 11. List of tests involving simultaneous Bm, St, and Bk motion.

NOTE: Positive cylinder motion corresponds to cylinder extension, and negative to cylinder contraction.

The results shown in the following figures show that the HEX *TL* model is able to capture all the different actuator interactions and provides accurate estimates of the joint responses. There are a few noticeable anomalies (related to actuator delays) worth pointing out (Refer to Section 3.2.4 for an explanation of the various delays in actuator response).

- In Figure 80 and Figure 82, the Bm velocity response on the testbed shows a distinct delay - about 0.5 secs in the two figures. This delay in the real response is most likely due to the initial state of the boom at the start of the tests. For instance, if the head end of the boom cylinder was at tank (low) pressure at the start of the test, it would take some time to pressurize the cylinder chamber before motion could start. This delay is dependent on the initial state of the cylinder, which is not modeled in our steady-state model. Hence, our model response in those figures starts 0.5 secs earlier than the testbed response.
- A similar error is seen in Figure 76 in the swing velocity plot. In this case, the testbed response starts earlier since the swing motor chamber was possibly at high pressure (trapped pressure) at the start of the test. This is another case of state-dependent delay which is not captured by our steady-state model.

However, the effect of not modeling the state dependent delays is not significant, as seen from the corresponding position plots.

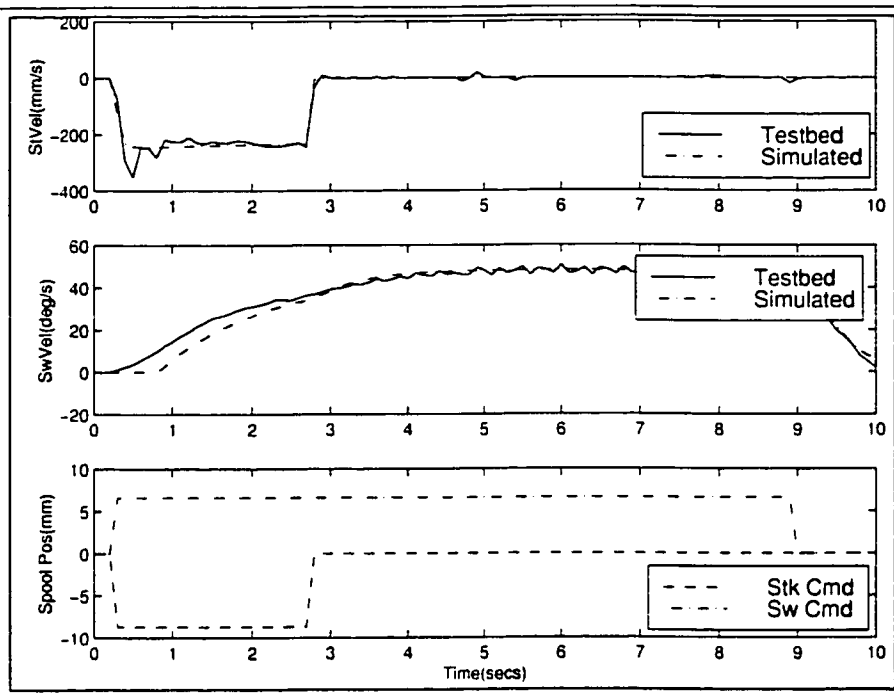


FIGURE 76. Velocity plots for test with St - and Sw + motions.

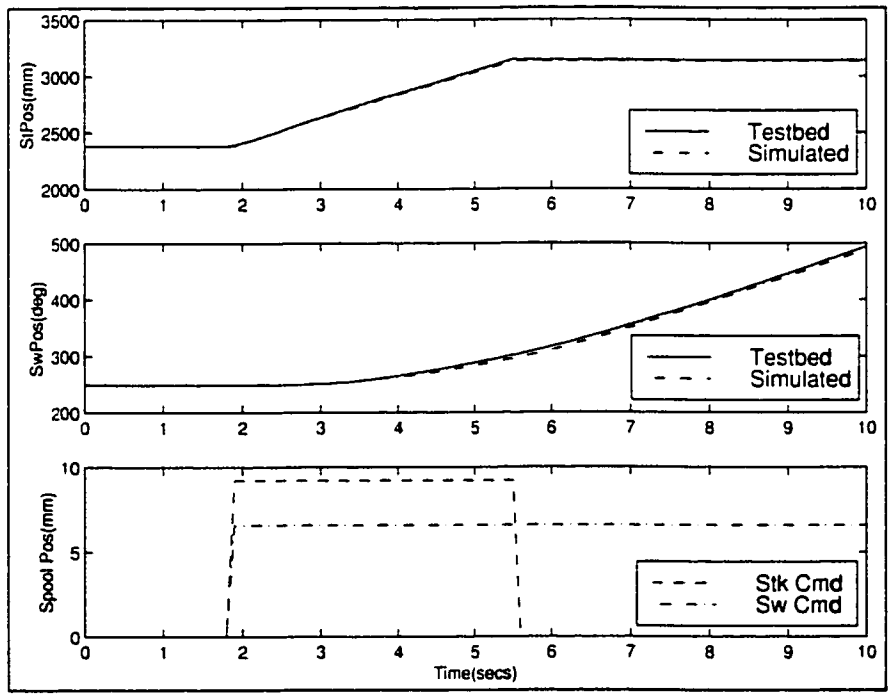


FIGURE 77. Position plots for test with St - and Sw + motions.

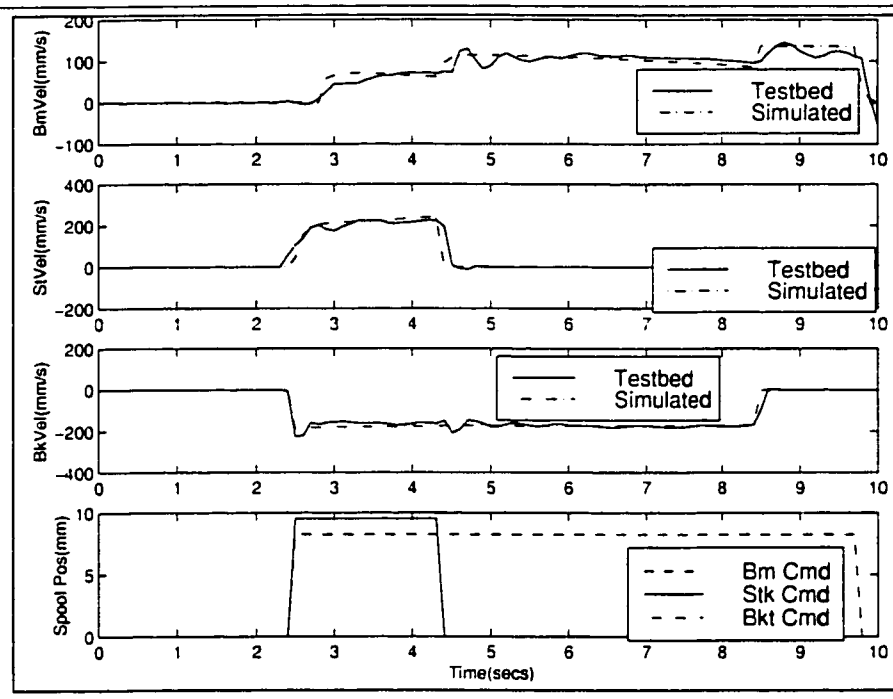


FIGURE 78. Velocity plots for test with Bm +, St +, and Bk - cylinder motions.

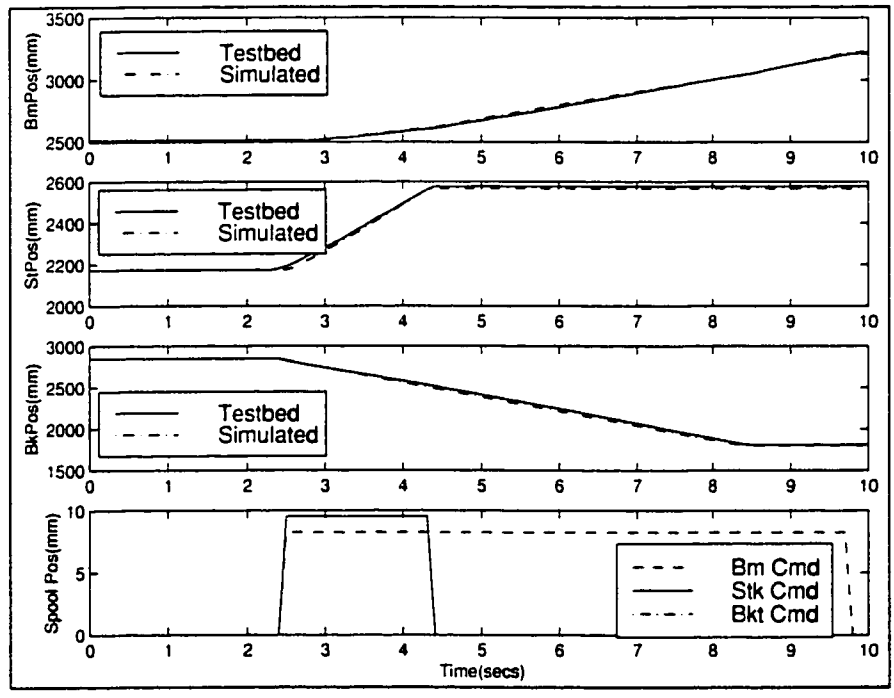


FIGURE 79. Position plots for test with Bm +, St +, and Bk - cylinder motions.

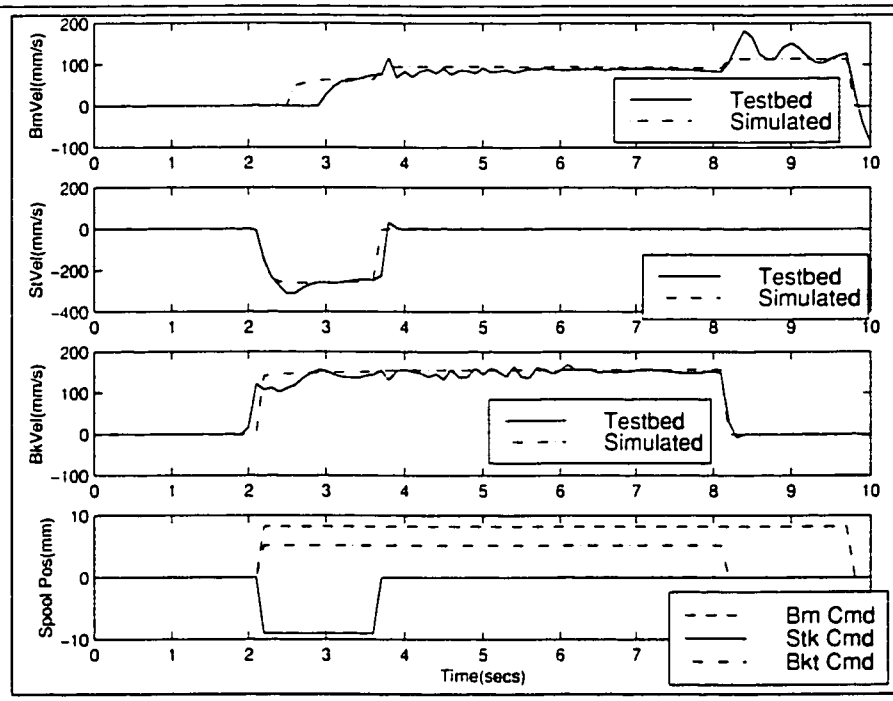


FIGURE 80. Velocity plots for test with Bm +, St -, and Bk + cylinder motions.

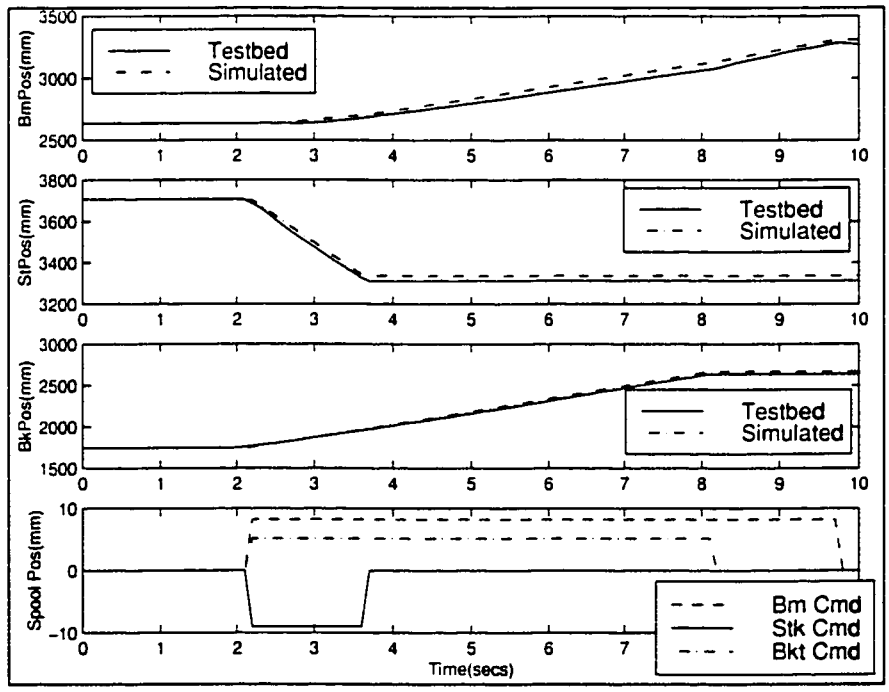


FIGURE 81. Position plots for test with Bm +, St -, and Bk + cylinder motions.

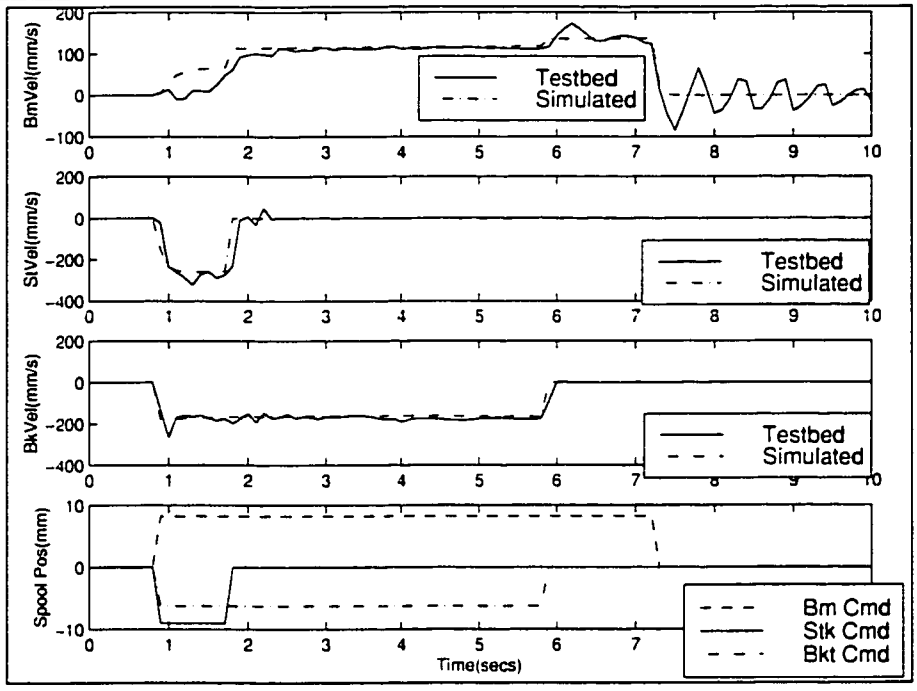


FIGURE 82. Velocity plots for test with Bm +, St -, and Bk - cylinder motions.

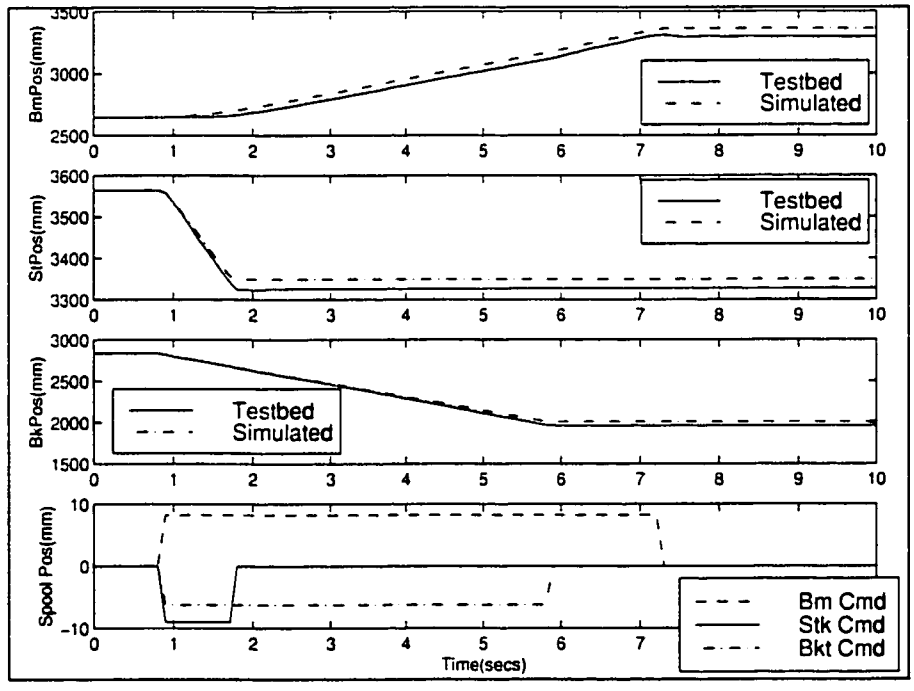


FIGURE 83. Position plots for test with Bm +, St -, and Bk - cylinder motions.

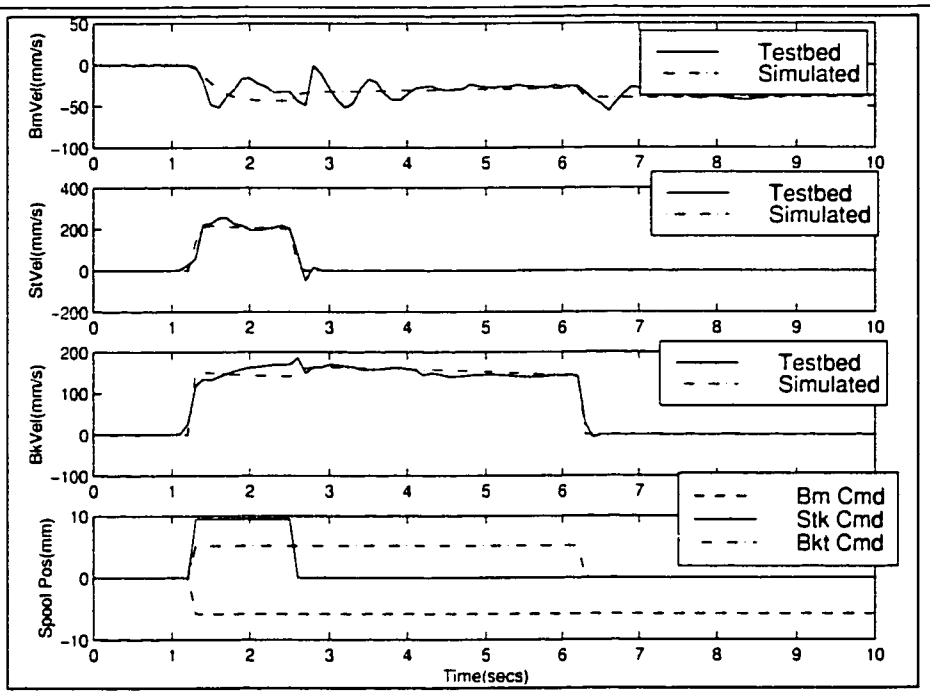


FIGURE 84. Velocity plots for test with Bm -, St +, and Bk + cylinder motions.

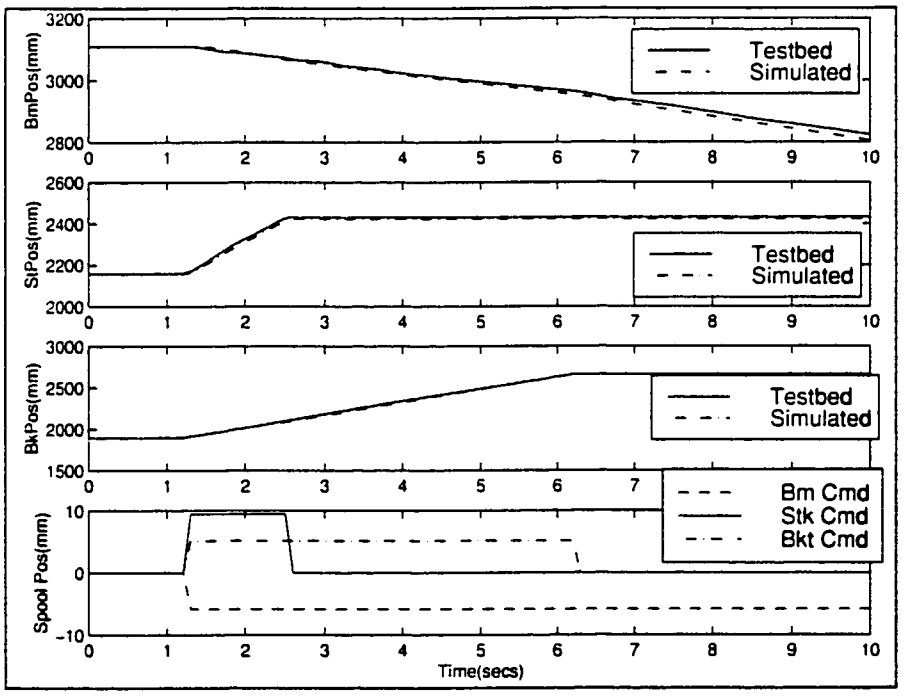


FIGURE 85. Position plots for test with Bm -, St +, and Bk + cylinder motions.

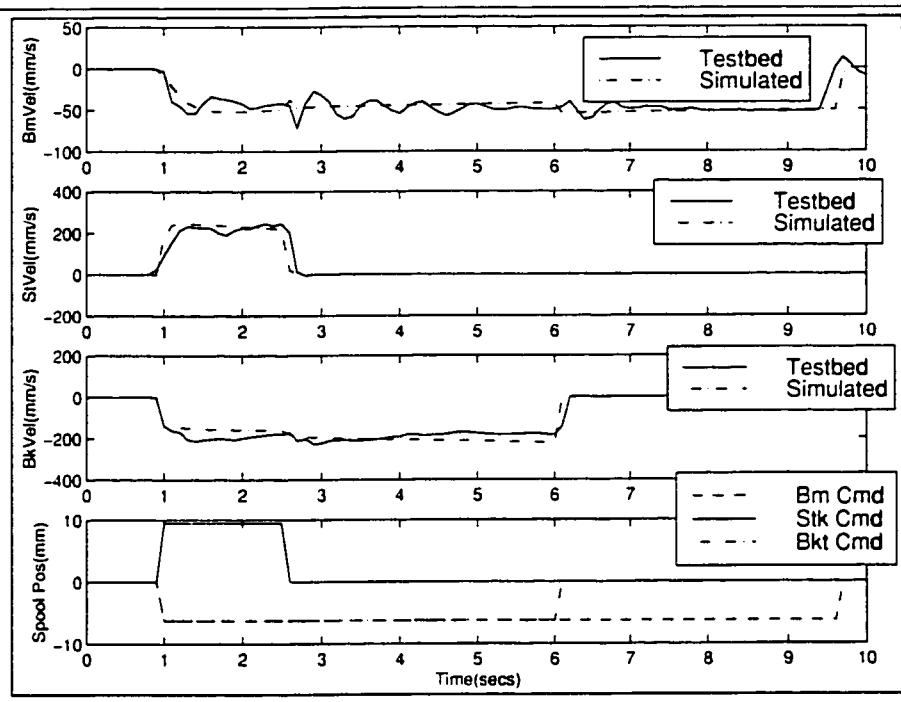


FIGURE 86. Velocity plots for test with Bm -, St +, and Bk - cylinder motions.

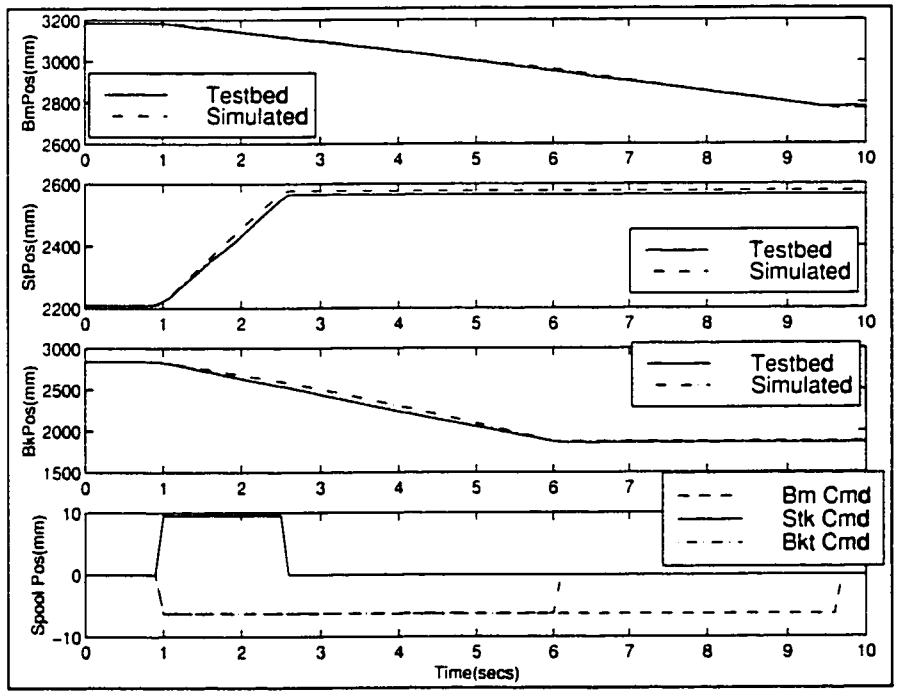


FIGURE 87. Position plots for test with Bm -, St +, and Bk - cylinder motions.

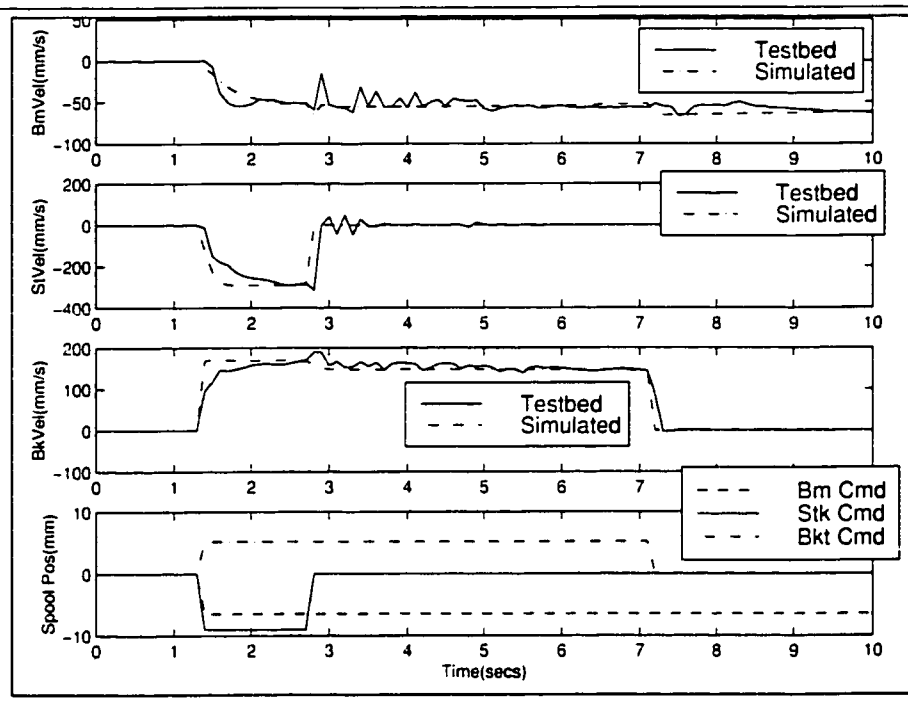


FIGURE 88. Velocity plots for test with Bm -, St -, and Bk - cylinder motions.

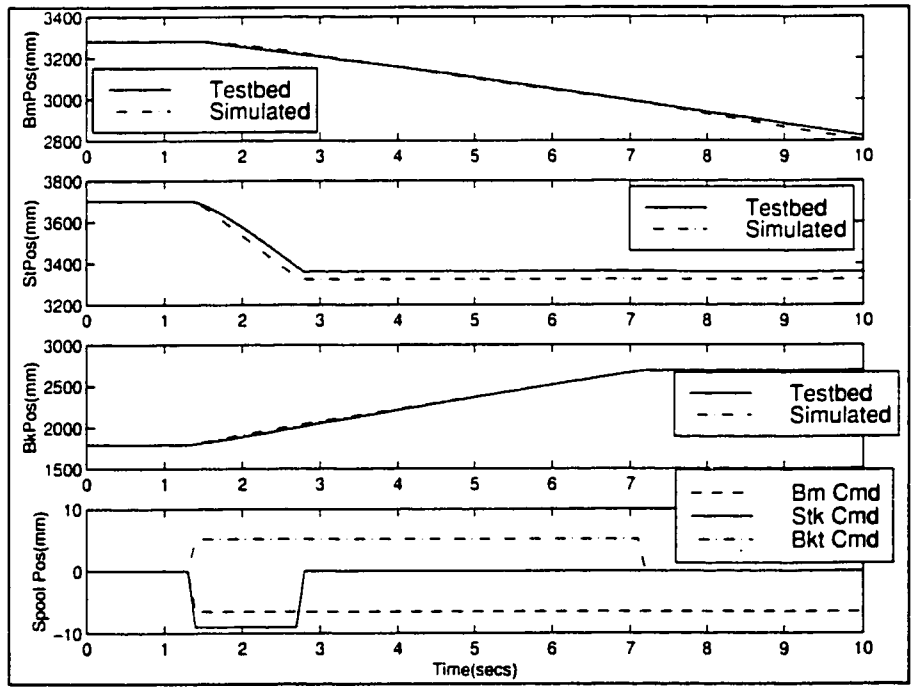


FIGURE 89. Position plots for test with Bm -, St -, and Bk + cylinder motions.

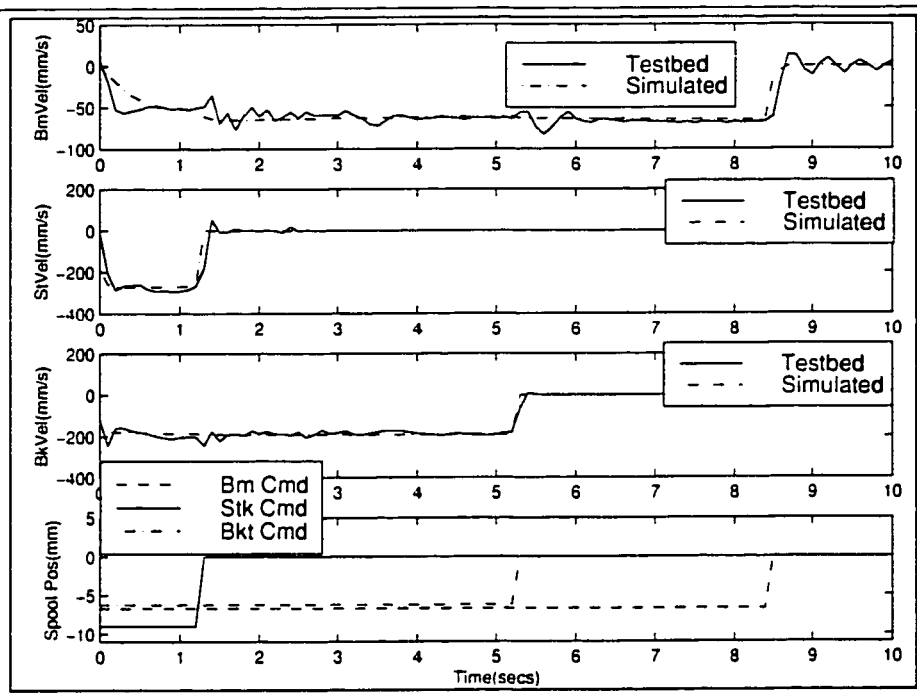


FIGURE 90. Velocity plots for test with Bm -, St -, and Bk - cylinder motions.

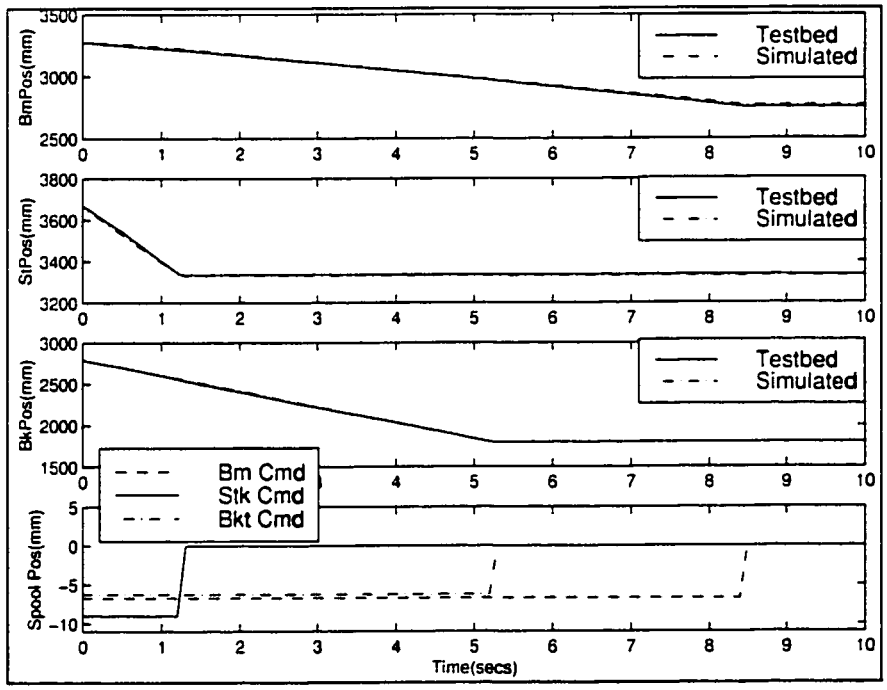


FIGURE 91. Position plots for test with Bm -, St -, and Bk - cylinder motions.