Autonomous Robotic Satellite Capture Using Constrained Predictive Control

by

Richard McCourt

B.A.Sc., University of British Columbia, 2002

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES
(Department of Mechanical Engineering)

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA
July 2004

Library Authorization

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Richard McCourt — Name of Author *(please print)*

21/10/2004 — Date (dd/mm/yyyy)

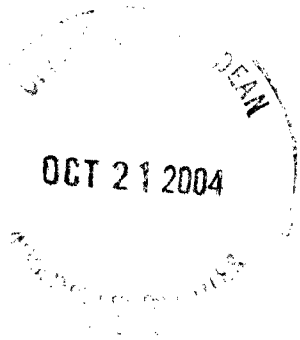
Title of Thesis:

Autonomous Robotic Capture of a Satellite Using Constrained Predictive Control

Degree: **Master of Applied Science** Year: **2004**

Department of **Mechanical Engineering**

The University of British Columbia
Vancouver, BC Canada



Abstract

This thesis investigates the use of model-based predictive control for the capture of a multi-degree-of freedom object that moves in a somewhat arbitrary manner, using a deployable manipulator. While the study is conducted through both computer simulation and ground-based experimental investigation, the intended application is focused on automating the robotic capture of a free-floating and spinning satellite. The main motivation for this application comes from the fact that robotic satellite capture in space, for retrieval, correction, repair, etc., can significantly eliminate the risks involved when astronauts are used in space walk scenarios to manually execute the necessary tasks. When a satellite is spinning, the maneuvering of a robotic manipulator by a human operator for a successful capture becomes increasing difficult. For this reason, satellite capturing using an autonomous robot is particularly attractive, and is studied in the thesis.

The present investigation uses an innovative manipulator known as the Multi-module Deployable Manipulator System (MDMS), which has been designed and built in our laboratory. It is a multi-purpose manipulator, which consists of a combination of revolute and prismatic joints and provides several advantages over the standard, all-revolute-joint manipulators that are used in space applications. The advantages include reduced dynamic coupling, fewer configuration singularities, easier obstacle avoidance, and simpler inverse kinematics, when compared to all-revolute-joint manipulators of the same size and degrees of freedom. Furthermore, our experience in designing, controlling, and experimentation with two models of the manipulator provides an added incentive for using the MDMS as the test bed for the present investigation.

In the controller that is developed, computer-simulated, implemented, and tested in the present research, the future path of the target satellite is predicted, and the controller uses this future knowledge of the target, and an internal model of the manipulator, to make optimal control decisions to minimize the tracking error between the target and the end-effector. Multi-parametric Quadratic Programming (mp-QP) techniques are used in order to obtain constrained optimal control decisions in real-time. The mp-QP algorithm offers fast solution times by explicitly solving the constrained quadratic programming problem offline and then using look-up tables in the real-time application. Unfortunately, the mp-QP solution is susceptible to numerical problems, and as a result the fully constrained predictive controller could not be implemented in real time. However, a sub-optimal version of the controller was implemented. The results show that the mp-QP

algorithm is still able to realize user defined constraints; therefore providing the benefits of constrained optimal control in the satellite capturing problem.

The results show that when the satellite motion is predicted the tracking performance is improved. Moreover, when the physical constraints of the system are formulated into the optimization, the controller becomes aware of its own limitations and the approach towards the satellite is improved by eliminating possible overshooting of the target.

Table of Contents

List of Tables

List of Figures

List of Symbols

A, B, C : State-space matrices describing the linearized and descretized model of the manipulator; Equation (2.4).

c_i : Vector of constraints on the optimal controller cost function at the future time, $t = k + i$; Equation (2.2).

$\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$: Vector of Coriolis and centripetal force terms used in the description of the system dynamics {see Appendix B}.

d_2 : Deploy length of the second joint in the satellite capturing manipulator {see Figure 3.1}.

$\mathbf{EYE_N}$: Matrix of N consecutive identity matrices; Equation (2.20)

$\mathbf{F_s}$, $\mathbf{F_v}$: Diagonal matrices containing the Coulomb and viscous friction terms, respectively {see Appendix B}

$\mathbf{F_i}$, $\boldsymbol{G_i}$: Gain and offset used to describe the affine control law in the i^{th} region; Equation (5.2)

$\mathbf{H_i}$, $\boldsymbol{K_i}$: Define the i^{th} region convex polytope in the parameter space $\Re^{N_p}$; Equation (5.1)

I : Identity matrix

I_i : Moment of inertia of the i^{th} link

k : Discrete time step, $t = k \cdot T_s$

$\mathbf{K_{MPC}}$: Feedback gain matrix used in the unconstrained predictive controller {Page 22}.

K : Conversion factor from servo card voltage, $\boldsymbol{u}(k)$, to joint torque/force, $\boldsymbol{\tau}(k)$; Equation (3.2)

l_w : The length from the end-effector to the center for the third joint of the manipulator; Equation (3.4)

l_t : The distance from the grappling point on the satellite to it's center of mass; Equation (3.5)

$\mathbf{M}(\boldsymbol{q})$: Mass matrix {see Appendix B}

M : Control horizon (or number of basis functions used in Predictive Functional Control)

N : Prediction horizon

N_p : Number of optimization parameters

N_r : Number of regions found in the explicit solution to the predictive control problem {see Appendix A}.

N_{Ci} : Number of facets (boundaries) describing the i^{th} region in the explicit predictive control solution

N_C : Total number of facets (boundaries) describing the N_r regions in the explicit predictive control solution; Equation (5.3)

n : Number of states used to describe the robotic manipulator

n_u, n_y, n_r : Number of inputs, outputs, and reference variables, respectively, for the robotic manipulator.

$\boldsymbol{q}$: Vector of joint positions $\begin{bmatrix}\theta_1 & d_2 & \theta_3\end{bmatrix}^T$

$\boldsymbol{r}(k)$: Vector of joint reference variables

$\mathbf{Q}(i)$, $\mathbf{R}(i)$: Cost function weights at future time, $t = k+i$, for the tracking error and control input moves, respectively. $\mathbf{Q}(i) \succeq 0$, $\mathbf{R}(i) \succ 0$

$\mathbf{Q}$, $\mathbf{R}$: Weighting matrices in the vector-matrix form of the predictive controller cost functions; Equation (2.11). $\mathbf{Q} \succeq 0$, $\mathbf{R} \succ 0$

$\mathbf{S_Q}$, $\mathbf{S_R}$: Cholesky decomposition of $\mathbf{Q}$ and $\mathbf{R}$, respectively; Equation (2.17)

T_s : Controller sampling time

$\boldsymbol{u}(k)$: Vector of servo card command voltages; inputs computed from the controller.

$\boldsymbol{u}_i(k)$: Vector of i^{th} coefficient in the basis function description of the inputs $\mathbf{u}(k)$ {Equation (2.3)}

$\Delta\boldsymbol{u}(k)$: The change in control inputs, $\Delta\boldsymbol{u}(k) = \boldsymbol{u}(k) - \boldsymbol{u}(k\text{-}1)$

$\Delta\boldsymbol{U}(k)$: Vector of current and future changes in control inputs; Equation (2.10)

$V(k)$: The cost function minimized by the controller at each time step.

$\mathbf{Y}$, $\mathbf{H}$, $\mathbf{F}$: Matrix coefficients used in the description of the controller cost function; Equation (3.9)

$\mathbf{G}$, $\boldsymbol{W}$, $\mathbf{E}$: Matrices used to describe the constraints on the controller cost function; Equation (2.20)

$\boldsymbol{x}$: Chaser robot state vector, $\boldsymbol{x} = \left[\boldsymbol{q}^T, \dot{\boldsymbol{q}}^T\right]^T \in \Re^n$

$\boldsymbol{x_r}$: Target state vector, $\boldsymbol{x_r}(k) = [\boldsymbol{r}(k)^T, \dot{\boldsymbol{r}}(k)^T, \ddot{\boldsymbol{r}}(k)^T]^T$

y : Chaser robot output vector, $y = \boldsymbol{x} = \left[\boldsymbol{q}^T, \dot{\boldsymbol{q}}^T\right]^T$

$\succeq$: Positive semi-definite

$\succ$: Positive definite

$\|\cdot\|_R^2$: weighted 2-norm (weighted by the matrix **R** in this case)

Greek Symbols

θ_i : Manipulator joint angles, $i \in \{1, 3\}$

θ_t : Orientation of the target satellite

λ : Vector of Lagrange multipliers

$\mu_i(k)$: Vector of i^{th} coefficient in the basis function description of the input moves $\Delta u(k)$ {Equation (2.12)}

$\mu(k)$: Vector of the M $\mu_i(k)$ vectors and the minimization argument for the Predictive Functional Controller cost function; defined in Equation (2.12)

$\xi(k)$: Vector containing the N_p optimization parameters in the controller cost function; $\xi(k) = [x(k)^T, u(k-1)^T, x_r(k)^T]^T$

τ : Vector of joint forces and torques {see Appendix B}

ω : Angular speed of the target satellite

Ψ, Γ, Θ : Matrix coefficients used in the description of the controller cost function {Equation (2.10)}

Λ : Transformation matrix from $x_r(k)$ to $T(k)$ {Equation (3.8)}

$T(k)$: Vector of the current and future reference variables, $r(k)$.

$Y(k)$: Vector of the current and future measured output, $y(k)$.

Ω : Transformation matrix from $\mu(k)$ to $\Delta U(k)$ {Equation (2.12)}

Abbreviations, Acronyms

APPE: Active Prediction Planning and Execution

CMG: Control Momentum Gyro

DARPA: Defense Advanced Research Projects Agency

HST: Hubble Space Telescope

KKT: Karush-Kuhn-Tucker

LP: Linear Programming

LQR: Linear Quadratic Regulator

LQG: Linear Quadratic Gaussian

MDMS:	Multi-module Deployable Manipulator System
MPC:	Model Predictive Control
mp-QP:	multi-parametric Quadratic Programming
MPT:	Multi-Parametric Toolbox
MSS:	Mobile Servicing System
NASA:	National Aeronautics and Space Administration
NASDA:	National Space Development Agency of Japan
OSV:	Orbital Service Vehicle
PFC:	Predictive Functional Control
QP:	Quadratic Programming
RHC:	Receding Horizon Control
RTX:	Real-Time eXtension
SCARA:	Selective Compliant Assembly Robot Arm
SPARTAN:	Shuttle Pointed Autonomous Research Tool for AstroNomy

A dot above a character refers to differentiation with respect to time.

A boldface italic character denotes a vector quantity.

A boldface character denotes a matrix quantity.

A '^' above a character denotes an estimation of that variable.

Subscripts 'e' and 't' correspond to the manipulator end-effector and target, respectively.

Subscripts 's' and 'd' refer to the slewing and deploying links, respectively.

Acknowledgements

I would like to thank my supervisor, Professor Clarence W. de Silva, for providing financial assistance, suggesting the research topic, and subsequent supervision and guidance of my research, which has led to the satisfactory conclusion of the present thesis. Also, I wish to thank Professor Vinod J. Modi who jointly with Professor de Silva, accepted me for the Master's program.

Thanks to Professor Elizabeth A. Croft and the rest of the Industrial Automation Laboratory for the knowledge and enlightenment shared in those morning meetings.

Thanks are also due to my colleagues and friends: Dr. Yang Cao, Mr. Parry Fung, Ms. Dana Kulić, Mr. Masaki Ohmiya, Mr. William Owen, Ms. Nazly Pirmoradi, Mr. Zhimin Qi, Dr. Keel-Soo Rhyu, Mr. Tao Sang, Mr. Poi Loon Tang, Mr. Ying Wang, Mr. Duminda Wijewardene, Mr. Kenneth Wong, and Mr. Jian Zhang. I greatly appreciate the knowledge, experience and culture that they have shared with me.

Thanks to my parents, Stephen and Jenny McCourt, my sister, Kristy McCourt, and the rest of my family for all of their support and encouragement that they have given me.

And last, but certainly not least, my thanks go to Tracy Ma, for her love and kindness, and for helping me through the good and the bad times.

Funding for this project has come from grants to Professor Clarence W. de Silva from the Natural Sciences and Engineering Research Council (NSERC) of Canada.

CHAPTER 1: INTRODUCTION

1.1 Background

One of the many reasons for the development of robotic manipulators is the ability for them to autonomously perform tasks that are not particularly suitable for humans; for example those involving long and repetitive operations and unhealthy, unpleasant and hazardous environments. Due to the particularly harsh environment of space, the application of robotics has received significant attention. The associated research activity and developments have led to many advances in space robotics that can be used to aid astronauts in performing extravehicular activities as well as numerous tasks related to the construction and operation of the International Space Station. Moreover, increased efforts have been made to develop autonomous and telerobotic systems that can be deployed in space while the human operators remain on Earth, thus eliminating some of the major risks involved in current launching procedures and also resulting in possible reductions of mission costs.

An important issue concerning the presence of humans in space is the servicing of the large number of satellites that orbit Earth. A special-purpose satellite that has received a considerable amount of attention is the Hubble Space Telescope (HST). In the past, astronauts, with the help of the Canadian robot, Canadarm, have been sent on board the Space Shuttle to perform various repairs on Hubble in order to keep it operational. However, recent efforts to repair the telescope have been halted due to increased safety concerns after the loss of the STS-107 crew in the space shuttle Columbia. Efforts to save the space telescope have led to heightened interests in the area of space robotics, with the emphasis on telerobotics. A prime candidate for Hubble servicing robots is a tele-operated android named Robonaut, designed by the Robotic Systems Technology Branch of NASA, in collaboration with the Defense Advanced Research Projects Agency (DARPA) of USA [Ambrose, et al., 2000; Bluethmann, et al., 2003].

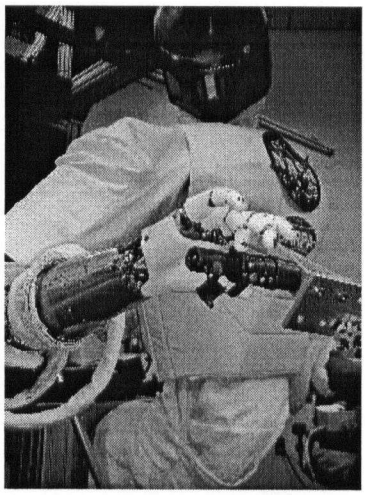


Figure 1.1 - Robonaut

The interest in Robonaut goes deeper than the maintenance of the HST. The successful robotic servicing of the Hubble telescope will undoubtedly give a further boost to the growing interest in unmanned orbital service vehicles (OSV's). These OSVs would be capable of remaining in orbit for very long periods of time and sufficiently versatile to service a large variety of orbiting space traffic. MacDonald, Dettwiler and Associates (MDA) and Boeing Phantom Works are doing some work in this area with their "Orbital Express" program, and so has the National Space Development Agency of Japan (NASDA) with their Hyper-OSV [Matsumoto, et al., 2002].

Control of the Robonaut will be achieved through human tele-presence control systems, allowing it to be operated safely from a ground control station on Earth. Unfortunately, the communication delay between Robonaut (or any future OSV) and the ground control station can be on the order of several seconds. This delay would add to the difficulty that a remote operator would face in maneuvering a manipulator in a complicated task, not the least of which is system instability.

One task that has proven to be particularly difficult in the past is the maneuvering of a robot in the capture of a moving target. Clearly, the difficulty increases manifold when the future trajectory of the target is incompletely known. Several satellites have been captured for servicing or repair with the help of the Canadarm robotic manipulator on the space shuttle; however, the task can become extremely difficult when the satellite is spinning or tumbling out of control. For example, during STS-87, an astronaut operating the Canadarm attempted to capture a free floating SPARTAN satellite. In this attempt the end-effector of the manipulator nudged the target satellite, causing it to tumble, thus making it more difficult to complete the capture, so, astronauts were

sent out on a space walk to capture the satellite by hand. Of course, in the case of Robonaut, or other unmanned OSVs, human aid will not be utilized.

Automating the satellite-capturing task would reduce the risks associated with these operations, reduce the mission costs, increase the likelihood of successful capture, and generally improve the effectiveness of the OSV. Some research has been done in the area of automated satellite-capturing, but so far the only autonomous satellite capture performed while in orbit has been done on NASDA's Engineering Test Satellite (ETS) VII [Oda, 2000]. In this case, the robotic manipulator on the chaser satellite removed and released a target satellite from its base and then successfully recaptured it.

If the future of space robotics rests on tele-operated and autonomous systems, then autonomous moving target capturing should be included in the repertoire of the manipulator's abilities. One such manipulator that possesses several advantages over standard all-revolute-joint space manipulators is the Multi-module Deployable Manipulator System (MDMS) as conceived in our laboratory.

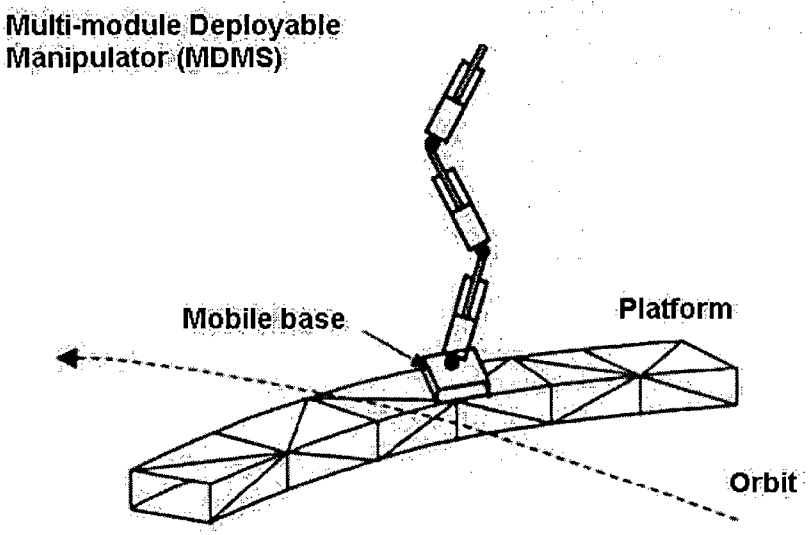


Figure 1.2 - Illustration of the MDMS on its space platform.

An illustration of the orbiting space platform based concept of this manipulator design is shown in Figure 1.2. The MDMS is comprised of a series of two-degree-of-freedom modules connected together in a chain topology. Each module is a combination of one slewing (revolute) link and one deploying (prismatic) link. The manipulator may be configured for a particular application by connecting as many modules as necessary. The MDMS has a variable geometry structure, in view of its deploying links, which provides particular advantages in obstacle avoidance. It has been designed as a multi-purpose manipulator, similar to the Mobile Servicing System (MSS) on the International Space Station; however, compared to conventional (all revolute joint) manipulators, like the MSS, the MDMS has several advantages. In addition to better obstacle avoidance, the MDMS has fewer singular configurations, reduced dynamic coupling between links, and simpler inverse kinematics.

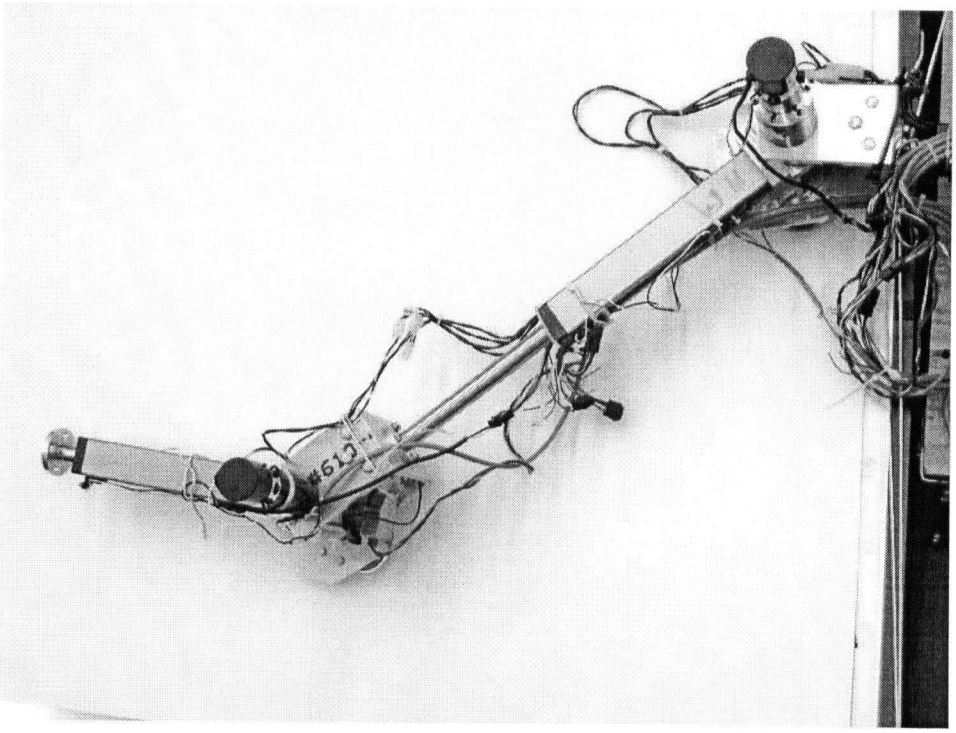


Figure 1.3 - The MDMS prototype used in the experiments.

A ground-based prototype of the MDMS as developed in our laboratory is used in the experimental investigations of the present research. In particular, a model-based predictive controller will be implemented on the MDMS, and its real-time performance will be investigated. The physical pro-

totype used in our experiments is a planar manipulator with rolling supports under the manipulator joints, to eliminate the gravity effects. The MDMS prototype is shown in Figure 1.3. The revolute joints of the manipulator use DC motors fitted with low backlash, harmonic drive gears, and deployment in the prismatic joints is provided by DC motors directly coupled to ballscrew actuators. Feedback motion signals from the joints are sensed using incremental optical encoders attached to the motor shafts, and fed to a servo card installed on the robot control computer running Windows 2000. The controller is programmed in Visual C++, and VenturCom's Real-Time Extension (RTX) provides the application programming interface (API) thereby giving real-time capabilities to Windows [de Silva, McCourt & Ohmiya, 2003].

1.2 Motivation

The investigation presented in this thesis focuses on the use of model-based predictive control for robotic capture of an object whose motion is not completely known. The main application considered is satellite capture using an autonomous robot for motion correction, retrieval, and repair. A multi-module deployable manipulator system (MDMS) as developed in our laboratory will be used as both the analytical/computer model and the experimental prototype in the present investigation. In this section, the motivation and justification will be provided for the use of MDMS and model-based predictive control in the present investigation.

The MDMS is a multi-purpose space manipulator with several advantages over conventional manipulators, and the satellite capture problem is investigated here as a specific application of this class of manipulators. The autonomous satellite-capturing controller must be able to smoothly maneuver the manipulator towards an expected interception point with the target and continue to track the target until the capture is complete. Since the target will be moving with incompletely known motions, it is desirable to have a controller that is capable of anticipating the future movement of the target. In addition, if the controller is aware of its own capabilities it can use this knowledge, and any available knowledge of the target, to help in choosing the best robotic action.

Model Predictive Control (MPC) is a model-based optimal controller, which uses open loop predictions of the system response into a finite, future length of time to repeatedly minimize a user-defined cost function, in achieving the specific control objective. Predictions can be made on

both the target and the robot to create the desired anticipative control action. The specific predictive controller used in the present work has the ability to handle user-defined constraints. These constraints can be used to specify the desired level of smoothness in the joint motions. For these reasons MPC is particularly suitable for use in the autonomous satellite capturing robots.

MPC has been successfully used in the process control industry; however, the application of predictive control in the aerospace industry appears to be relatively new. The recent applications of predictive control to high bandwidth aerospace systems have been aided by work in muti-parametric, explicit MPC solutions [Bemporad, et al. 2002] and with the constantly increasing speed of computers.

Several control techniques have been implemented on the MDMS in our laboratory, for trajectory tracking problems. Noteworthy are linear quadratic regulator (LQR), linear quadratic Gaussian (LQG) control, feedback linearization technique, modal control, intelligent fuzzy control, and neural-network adaptive control. These techniques, however, do not possess the "predictive" capability as useful in the satellite capture problem. The recent developments in efficient MPC solutions and the ever-increasing speed of digital computers have made possible the effective use of a real-time constrained model predictive control in high-bandwidth applications such as the MDMS. Satellite capture provides a good motivation for this implementation.

1.3 Literature Review

Autonomous satellite capture using robotic manipulators has been done before even though this is a relatively new topic. In particular, the ETS-VII satellite has performed an autonomous satellite capture while in orbit [Inaba & Oda, 2000]. A brief literature survey is given now to present some existing work in the areas of satellite catching and robotic predictive controllers, in the context of the present investigation.

In the area of satellite capturing, a focus has been the development of vision algorithms that are capable of supplying reference paths to standard robotic controllers [Jasiobedzki, et al., 2001]. Pose determination and machine vision are not considered in the present investigation, and it is assumed throughout the thesis that the current position of the target object is known.

In order to study the satellite capturing, it is important to understand how the satellite will be moving and how the manipulator can be used to capturing the satellite. In earlier work on the MDMS, Ueno and Modi [1998] considered the satellite capturing problem in computer simulation, where the satellite was assumed to be a point object, moving with constant velocity in a two degree-of-freedom environment.

The difficulty in capturing a satellite is increased when it is spinning or tumbling. This is because typically there is a specific grappling point where the manipulator must make contact with the satellite, and when the satellite is spinning, the grappling point is rotating around the satellite's center of mass. Nagamatsu, et al. [1996] studied the motion of a tumbling satellite and described this motion as the superposition of three rotations with constant angular velocities. First, the magnitude of these angular velocities was estimated. Then a manipulator with enough degrees of freedom aligned three joints with the three axes of rotation, and maintained alignment by spinning these joints at the same velocity as the corresponding axes. When done properly, the capturing of the satellite became simple because the relative velocity between the grappling fixture and the end-effector of the manipulator was very small.

In later studies, Nagamatsu [1997] considered a target that was translating as well as rotating, and predictions of the target position and orientation were used to compensate for time delays due to image processing. The position and orientation of the target were corrected and used as the desired position for a Jacobian inverse controller. Control of the manipulator was not considered further.

Richards and How [2003] have used MPC to autonomously guide a spacecraft towards a rendezvous point (e.g. satellite). The paper offered a guaranteed robust, finite time maneuver completion for an arbitrary target region. The work solved a minimum fuel consumption problem with added constraints to guarantee a successful rendezvous with the target. The MPC controller produced solutions that were more robust against unmeasured disturbances when compared to previously used control schemes for this problem. In addition, the use of parametric programming techniques allowed for more complicated cost functions to be minimized, and overall improved the performance of the maneuver.

Predictive control has been applied to a variety of mobile robotic applications, but fewer applications have been made on robotic manipulators. This is most likely because of the added difficulty

in dealing with the nonlinearities of the manipulator. Some robotic predictive controllers first linearize the robot using feedback of the inverse dynamics of the manipulator [Poignet & Gautier, 2000; Torres et al., 2001], while others have used Taylor series approximations [Hedjar et al., 2002] or linearization of the robot at each time step [Valle, Tadeo & Alvarez, 2002].

Poignet and Gautier [2000] made a comparison between the commonly used Computed Torque Control [Sciavicco & Sicilano, 2000] and a variation of MPC often used for high bandwidth systems known as Predictive Functional Control (PFC). In both controllers the plant is first linearized with the inverse dynamics model of the manipulator. Simulation results on the first two links of an industrial SCARA type robot showed that, in the absence of disturbances or model uncertainties, the computed torque controller could be tuned to give better tracking performance. However, the predictive functional controller performed significantly better in the presence of unmeasured disturbances and model uncertainties.

Valle, et al. [2002] applied constrained predictive control to a two degree-of-freedom direct-drive robot. A linearization of the manipulator was performed at each time step and then the constrained optimization of the predictive controller cost function was computed. Of course, this method was very computationally intensive, and even on a two degree-of-freedom robot with a very small prediction horizon, only computer simulations could be performed.

Typically, in the literature, good simulation results appear to have generated when predictive control was applied to a robot, but such claims have not been validated for the most part through experimental investigation and practical application. In the past work, MPC has been used to guide spacecraft towards satellites, and predictive control has been used on robot manipulators. The work presented in this thesis integrates such efforts in predictive control of robotic satellite capturing with the MDMS, using experimental implementation and verification as well as analysis and computer simulation.

1.4 Thesis Outline

In this introductory chapter the objectives of the present investigation are presented, the motivations are given, and the approach used in achieving the objectives is outlined. In particular, a need for autonomous robotic satellite capturing has been presented, along with some of the existing

work in this area. The multi-purpose MDMS robot as developed in our laboratory has been presented as a suitable application test platform for autonomous satellite capturing. Two prototype units of the MDMS have been developed in our laboratory. One is available to serve the role of the robot in autonomous satellite-capturing experiments, and the other unit as the moving satellite.

Model-based predictive control will be implemented on the MDMS for autonomously controlling the capture of a spinning satellite. Predictive control has been chosen here because of several pertinent advantages; in particular, it is an optimal controller that can be made to anticipate the future movements of the satellite target, under realistic constraints, and make improvements to the capturing process.

In Chapter 2 the theory behind model predictive control, as used in the present work, is presented. This chapter shows the general structure of the predictive controller and discusses how future set-points and constraints can be incorporated into the control decision. Then, a brief introduction to multi-parametric quadratic programming techniques is presented to show how the predictive controller can be implemented in high-bandwidth applications like the MDMS.

In Chapter 3 the autonomous satellite capturing problem is formulated. The dynamics of the MDMS and the target satellite are studied so that reference models of these systems can be incorporated into the controller. Next, the details of the integration of predictive control for the MDMS are discussed. The controller developed in this chapter is needed in the computer simulations and real-time experimentations with the MDMS, as presented in chapters 4 and 5, respectively.

In chapters 4 and 5 the predictive controller that is developed in Chapter 3 is used in robotic capture of a variety of satellite targets. In Chapter 4, in particular, the MDMS is simulated, and the optimization of the constrained predictive controller is performed at each time step to initially demonstrate the benefits of formulating constraints into the predictive controller. In Chapter 5, off-line solutions to the predictive controller optimization problem are computed in order to implement the constrained predictive controller at high control update rates, which are needed for the prototype MDMS.

The main conclusions of the work are given in Chapter 6, followed by some suggestions for future work in satellite capturing using predictive control on the MDMS.

CHAPTER 2:
PREDICTIVE CONTROL

2.1 Preamble

The predictive controller that is implemented in the multi-module deployable manipulator (MDMS) is an optimal controller, which is based on a predicted future response. It is known for its ability to operate close to state constraints, compared to a conventional controller [Maciejowski, 2002]. A typical optimal controller minimizes a user-defined cost function based on a model of the plant and a set of boundary conditions. The resulting optimal controller is obtained by solving the Hamilton-Jacobi-Bellman equation,

$$\frac{\partial V^*}{\partial t} + V\left(\boldsymbol{x},\boldsymbol{u},t\right) + \sum_{j=1}^{n} \frac{\partial V^*}{\partial \boldsymbol{x}_j} \boldsymbol{g}_j\left(\boldsymbol{x},\boldsymbol{u},t\right) = 0 \tag{2.1a}$$

Here, $V(\boldsymbol{x}, \boldsymbol{u}, t)$ is the user-defined cost function, and V^* is the value of the cost function at the optimal control input, $\boldsymbol{u} = \boldsymbol{u}^* \in U$, where U is a constrained subset of the n_u-dimensional space defined by the vector $\boldsymbol{u}$, $U \subseteq \Re^{n_u}$. The vector $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{u},t)$ describes the system dynamics as given by the set of n state equations $\dot{\boldsymbol{x}} = \boldsymbol{g}\left(\boldsymbol{x},\boldsymbol{u},t\right)$, where $\mathbf{x}$ is the n^{th} order state vector.

The optimal control input can be expressed as a function of the current states; i.e., as a state feedback control law. The optimal control law is given by the minimization of equation (2.1a), as

$$\boldsymbol{u}^*\left(\boldsymbol{x},t\right) = \arg\min_{\boldsymbol{u}\in \mathbf{U}} \left(V\left(\boldsymbol{x},\boldsymbol{u},t\right) + \sum_{j=1}^{n} \frac{\partial V^*}{\partial \boldsymbol{x}_j} \boldsymbol{g}_j\left(\boldsymbol{x},\boldsymbol{u},t\right) \right) \tag{2.1b}$$

Unfortunately, the partial differential equation (2.1a) is very difficult (and often impossible) to solve, so as to yield an explicit solution for (2.1b). Only in special cases can a solution be deter-

mined; for example, H_2/H_∞ control or the Linear Quadratic Regulator (LQR), which are computed in the absence of inequality constraints.

When inequality constraints are added to the problem the solution becomes a piecewise defined control law, and the explicit solution becomes further complicated. Only special cases of these problems, for example, the minimum time control, have been solved. Anti-windup schemes can be added to the standard optimal controller as a type of ad hoc solution to integrator windup when constraints are met, but the optimality of the solution is not guaranteed, with respect to the cost function. Model Predictive Control (MPC), on the other hand, provides a more formal solution to a constrained optimal control problem, where the constraints are explicitly taken into consideration, thereby reducing the possibility of constraint violations.

In the rest of the chapter gives an introduction to model predictive control (MPC). Only the theory that is relevant to this thesis is covered. First, the basic moving horizon is described. Next the formulation of both the unconstrained and the constrained model predictive controllers is presented. More details on the topic are found in [Maciejowski, 2002].

2.2 Receding Horizons

In MPC the optimal control problem is solved in each control period. A cost function is minimized based on open-loop predictions of the plant response and subject to user defined constraints. Naturally, this will significantly increase the computational effort of the controller, so, depending on the application, certain efficiencies have to be invoked or simplifications have to be made in the MPC algorithm to make the computational time feasible. One possible simplification is to only perform the optimization over a finite time horizon of N future time steps and make this horizon sufficiently small so that the computations can be made within the control period, T_s. A set of N optimal control input vectors ($\hat{\boldsymbol{u}}(k\,|\,k)^{\dagger}$, $\hat{\boldsymbol{u}}(k+1\,|\,k)$, ... , $\hat{\boldsymbol{u}}(k+N-1\,|\,k)$) for the current and the future time instants k, $k+1$, ..., $N-1$ are determined, based on the current knowledge and the control objective, by minimizing the cost function

---

† The notation $\hat{\mathbf{u}}(k+1\,|\,k)$ means the predicted control input at time $t=k+i$ based on measurements (or, knowledge) at time $t=k$

$$V(k) = \sum_{i=k+1}^{k+N} f\left(\hat{\boldsymbol{x}}(i|k), \hat{\boldsymbol{r}}(i|k), \hat{\boldsymbol{u}}(i-1|k), i\right) \tag{2.2}$$

$$\text{subject to } \boldsymbol{c}_i\left(\hat{\boldsymbol{x}}(i|k), \hat{\boldsymbol{r}}(i|k), \hat{\boldsymbol{u}}(i|k), i\right) \leq \mathbf{0},$$

$$\text{and } \hat{\boldsymbol{x}}(i|k) = \boldsymbol{g}\left(\hat{\boldsymbol{x}}(i-1|k), \hat{\boldsymbol{u}}(i-1|k)\right)$$

where:

$V(k)$ = total cost at current time, $t=k$,

$f\left(\hat{\boldsymbol{x}}(i|k), \hat{\boldsymbol{r}}(i|k), \hat{\boldsymbol{u}}(i-1|k), i\right)$ = cost at the future time, $t=i$,

$\boldsymbol{c}_i\left(\hat{\boldsymbol{x}}(i|k), \hat{\boldsymbol{r}}(i|k), \hat{\boldsymbol{u}}(i|k), i\right)$ = inequality constraints at future time, $t=i$,

N = prediction horizon,

and the state model $\hat{\boldsymbol{x}}(k+1|k) = \boldsymbol{g}\left(\hat{\boldsymbol{x}}(k|k), \hat{\boldsymbol{u}}(k|k)\right)$ describes the system dynamics.

Once the optimal solution has been determined, the corresponding input $\boldsymbol{u}(k) = \hat{\boldsymbol{u}}(k|k)$ is applied to the plant. The process is repeated at the start of the next control cycle. From equation (2.2) it can be seen that for a constant N, the terminal time, $k+N$, increases with k. This is known as a receding horizon, and because of this MPC is sometimes referred to as Receding Horizon Control (RHC).

2.3 Control Horizons and Predictive Functional Control

As mentioned earlier, the computation time of the predictive controller is a major limiting factor in the real-time implementation of MPC in high-bandwidth processes. Even with a fairly small prediction horizon the optimization problem can still be rather computationally intensive. As a further effort to decrease the computational load, control horizons are introduced.

When a control horizon M is introduced to the MPC algorithm, the cost function (2.2) is reformulated so that for some $M \leq N$, the control input does not change for times $t \geq k+M$. Specifically, the optimal solution consists of only M input vectors ($\hat{\boldsymbol{u}}(k|k)$, $\hat{\boldsymbol{u}}(k+1|k)$, ...,

$\hat{\boldsymbol{u}}(k+M-1|k)$), and the remaining *N-M* input vectors are all identical; i.e., $\hat{\boldsymbol{u}}(k+i|k)=\hat{\boldsymbol{u}}(k+M-1|k)$ for $M < i \leq N$. The value of the control horizon *M* can then be chosen to provide a more desirable sized optimization problem without having to place heavy restrictions on the prediction horizon, *N*.

The control horizon, *M*, also has some effect on the performance of the controller. The prediction horizon, *N*, determines the number of variables that are penalized in the cost function. When a control horizon is used, the same number of parameters must be minimized in the cost function, but now there are fewer degrees of freedom to choose from, and this will affect the solution of the optimization. Exactly how it affects the solution will depend on the system model. Typically, as the control horizon is decreased the controller becomes more aggressive.

Instead of using only *M* control moves in the beginning of the horizon, a more intuitive solution is to use a linear combination of *M* basis functions to describe the inputs throughout the horizon. Any basis functions could be used, but in this thesis a polynomial basis is chosen. Now, the current and future input vectors are related through an (*M*-1)-order polynomial function,

$$\hat{\boldsymbol{u}}(k+i|k)=\boldsymbol{u}_0(k)+\boldsymbol{u}_1(k)i+\cdots+\boldsymbol{u}_{M-1}(k)i^{M-1}\ , \tag{2.3}$$

and the *M* vectors $\boldsymbol{u}_0$(k), $\boldsymbol{u}_1$(k), ..., $\boldsymbol{u}_{M-1}$(k) become the argument of the original minimization problem in equation (2.2). The basis function parameterization of MPC is known as Predictive Functional Control (PFC) and was originally developed by Richalet, et al [1978, 1987] in the early 1980s for application of MPC on high-bandwidth servomechanisms.

PFC has several advantages over the control horizon form of the controller. In general, PFC is capable of making more accurate predictions for a better-fit solution to the minimization problem in (2.2), especially when tracking complex trajectories [Qin & Badgwell, 1997]. This is demonstrated in Figure 2.1 where an arbitrary setpoint trajectory was given to a simulated prismatic joint of the MDMS described in Chapter 1. A comparison is made between MPC with a control horizon of two and PFC with the future inputs described by two basis functions (a step and a ramp). A simple quadratic cost function penalizing tracking errors and control input changes was used. A further discussion on cost functions will be given in a subsequent section of the thesis.

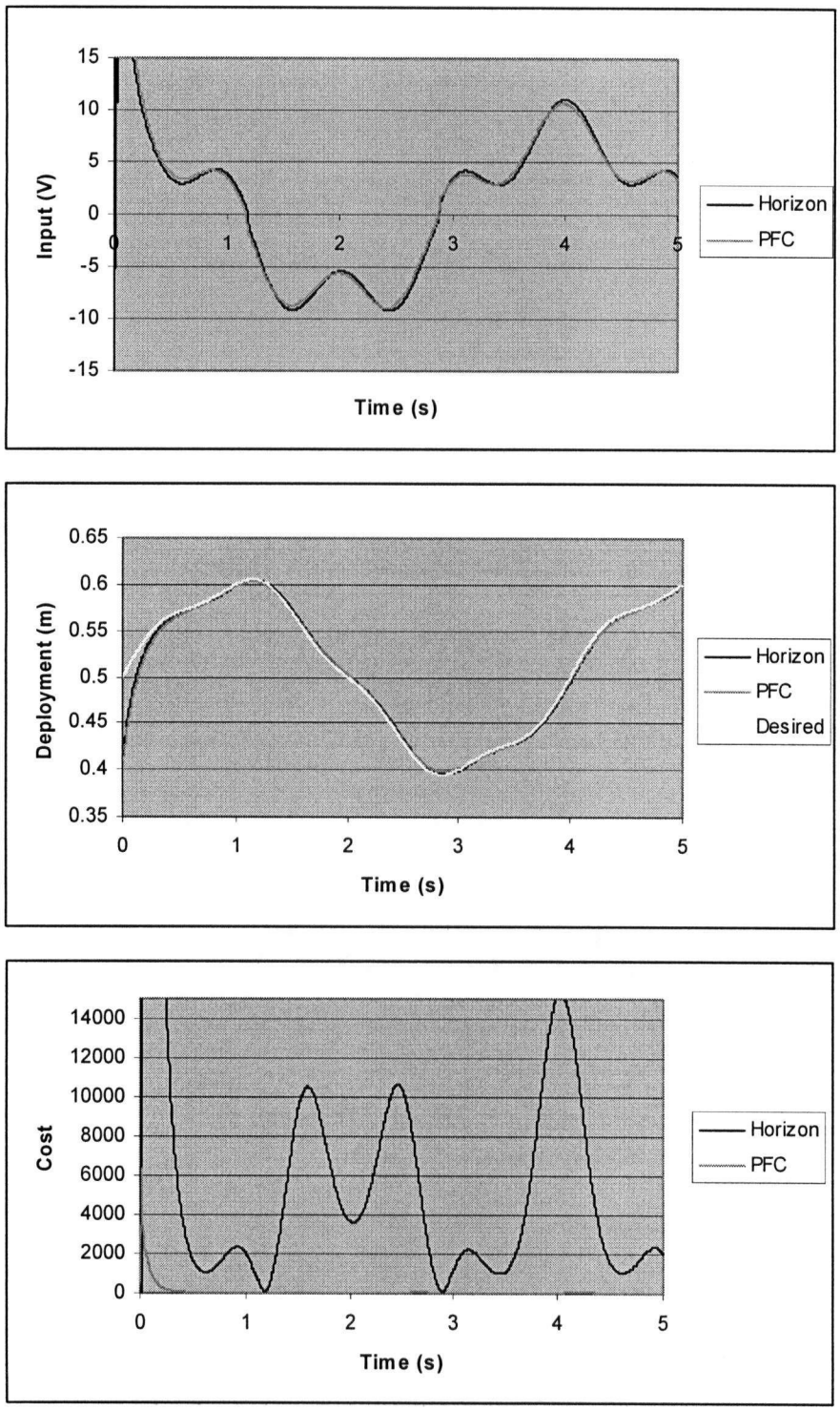


Figure 2.1 - Performance comparison between MPC using a control horizon, M=2, and MPC using two polynomial basis functions.

As shown in Figure 2.1, the tracking performance of the two controllers is almost identical. The major difference between the two controllers can be seen in the evaluation of the defined cost function. On average, the cost value of the control horizon form is 15,000 times larger than that of the basis function form. The more optimal solution has led some researchers to believe that PFC may provide an advantage when controlling nonlinear systems [Qin & Badgwell, 1997]. Moreover, by adjusting the cost function in the two controllers it was observed that PFC provided a more robust solution. The cost function parameters were adjusted until the control horizon formulation became unstable, yet the PFC controller maintained a very similar solution.

Unfortunately, the advantages of PFC do come at a cost. One disadvantage of PFC is that it tends to be more susceptible to numerical problems when trying to minimize equation (2.2). Another problem is that if the order of the input function is made too large, the calculated optimal solution can become rather oscillatory.

2.4 Internal Models

Model predictive control (MPC) uses a linear state-space model of the process in determining the control action. Of course, not all plants can be represented by linear models, and some effort has been made in extending MPC to nonlinear models [Qin & Badgwell, 1998]. However, the vast majority of developments and applications of MPC concern linear models. In fact, using a linear model makes it easier to solve the minimization problem in (2.2), and moreover, if the cost function is chosen properly, it is possible to show that the minimization returns a unique, global solution.

In this thesis a linear internal model is used in the implementation of MPC. Since the plant is a robotic manipulator, which is known to possess a variety of nonlinearities, some type of linearization must be done before MPC can be implemented on it. Details on how the plant is linearized and how the linear model is obtained will be discussed in Chapter 3. For now, the linearized, discrete-time, state-space model that is employed is expressed as

$$\begin{aligned} \hat{x}(k+1|k) &= \mathbf{A}\,\hat{x}(k|k) + \mathbf{B}\,\hat{u}(k|k) \\ \hat{y}(k|k) &= \mathbf{C}\,\hat{x}(k|k) \end{aligned} \tag{2.4}$$

where x is the n-dimensional state vector, u is the n_u-dimensional input vector, and y is the n_y-dimensional output vector. The rest of this chapter assumes that a linear model of this form is available for the robotic system.

2.5 Cost Function

The cost function of the optimal controller penalizes predicted reference tracking errors and the corresponding control decisions. Typically the 1, 2, or ∞-norm of the reference errors and control decisions are used. It can be shown that if the cost is formulated with one of these three norms there exists a unique, global solution to the minimization problem in equation (2.2) [Fletcher, 1987].

The 1 and ∞-norm cost functions are linear, and since a linear model of the plant is used, both of these forms lead to Linear Programming (LP) problems. The advantage to using a linear cost function is that there are many well known, and fast, algorithms for the LP optimization problem. Additionally, the ∞-norm cost function only penalizes the maximum predicted tracking error and tends to make the controller more robust against unmeasured disturbances [Maciejowski, 2002]. However, the disadvantage of using a linear cost function is that, as is well known [Fletcher, 1987], the solution to an LP problem always lies at the vertex of the polytope defined by the constraints, $c_i\left(\hat{x}(i|k), \hat{r}(i|k), \hat{u}(i|k)\right)$ in (2.2). This means that at all times n_u of the constraints will be active.

With a quadratic cost function, the minimization in (2.2) becomes a Quadratic Programming (QP) problem, and it is possible to find an unconstrained solution. Furthermore, the quadratic cost function can be formulated with weighting variables, which can be adjusted to affect the output performance, whereas in LP the constraints must be adjusted to affect the performance. In this thesis a quadratic (i.e., 2-norm) cost function is used. The cost function minimized in the tracking control problem is

$$V(k)=\sum_{i=1}^{N}\left[\left\|\hat{y}(k+i\,|\,k)-\hat{r}(k+i\,|\,k)\right\|_{\mathbf{Q}(i)}^{2}+\left\|\Delta\hat{u}(k+i-1\,|\,k)\right\|_{\mathbf{R}(i)}^{2}\right] \tag{2.5}$$

where $\left\|\hat{y}(k+i\,|\,k)-\hat{r}(k+i\,|\,k)\right\|_{\mathbf{Q}(i)}^{2}$ is the 2-norm of the tracking error at the future time $t=k+i$ weighted by the diagonal and positive semi-definite weighting matrix $\mathbf{Q}(i)\succeq 0$. The changes in control inputs, $\Delta\hat{u}(k+i\,|\,k)$, are penalized in the cost function and the amount at which they are penalized at time $t=k+i-1$ is weighted by the diagonal and positive definite matrix $\mathbf{R}(i)\succ 0$. This is the most commonly used form of the quadratic cost function.

The control increment $\Delta\hat{u}(k+i\,|\,k)$ is chosen as an argument in the cost function, as a means of providing offset free tracking when the steady-state control input is expected to be constant. However, if the steady state input is expected to be a ramp, then the second difference, $\Delta^{2}\hat{u}(k+i\,|\,k)=\Delta\hat{u}(k+i\,|\,k)-\Delta\hat{u}(k+i-1\,|\,k)$, should be used in (2.5), and the third difference, $\Delta^{3}\hat{u}(k+i\,|\,k)$, if the input is expected to be parabolic, and so on. For the position control of the MDMS when tracking a moving target the cost function form in (2.5) is adequate. This aspect will be revisited in Chapter 3.

In addition to providing offset-free tracking, the particular choice of the decision variable can also improve the smoothness of the control input. When the first difference, $\Delta u(k)$, is used then the control input to the plant is

$$u(k)=u(k-1)+\Delta u(k) \tag{2.6}$$

On the other hand, when the second difference is used, we have

$$\Delta^{2}u(k)=\Delta u(k)-\Delta u(k-1)=u(k)-u(k-1)-u(k-1)+u(k-2)$$

or,

$$u(k)=2u(k-1)-u(k-2)+\Delta^{2}u(k) \tag{2.7}$$

It follows that, increasing the order of the input difference is equivalent to increasing the order of a low-pass filter applied to the control input. However, applying a filter can affect the stability of the system, especially when dealing with high-bandwidth servomechanisms like a robot manipulator. In view of this, care should be taken when choosing the argument of the cost function. The results in chapters 4 and 5 show that in the satellite capturing problem the combination of PFC with a cost function argument of $\Delta\hat{\boldsymbol{u}}(k+i\,|\,k)$ generally produce smooth input functions and accurate tracking, hence, justifying the specific cost function given by (2.5).

2.6 Unconstrained Predictive Control

2.6.1 Controller Structure

Now that the cost function and the internal model have been defined, methods can be developed for solving the optimization problem. To start, the unconstrained controller is formulated. Specifically, the optimization problem in MPC becomes

$$\text{Minimize } V(k)=\sum_{i=1}^{N}\left[\left\|\hat{y}(k+i\,|\,k)-\hat{\boldsymbol{r}}(k+i\,|\,k)\right\|^2_{\mathbf{Q}(i)}+\left\|\Delta\hat{\boldsymbol{u}}(k+i-1\,|\,k)\right\|^2_{\mathbf{R}(i)}\right] \tag{2.8}$$

$$\begin{aligned}\text{Subject to } \hat{y}(k+i\,|\,k)&=\mathbf{C}\,\hat{\boldsymbol{x}}(k+i\,|\,k)\\&=\mathbf{C}\mathbf{A}\,\hat{\boldsymbol{x}}(k+i-1\,|\,k)+\mathbf{C}\mathbf{B}\,\hat{\boldsymbol{u}}(k+i-1\,|\,k)\end{aligned}$$

In this formulation (unconstrained), the process model is treated as the constraints of the optimization problem, as clear from (2.8). Since the cost function is formulated with $\Delta\hat{\boldsymbol{u}}(k+i\,|\,k)$, the constraints (i.e., the model) should also be formulated with $\Delta\hat{\boldsymbol{u}}(k+i\,|\,k)$. This can be done by setting

$$\hat{\boldsymbol{u}}(k+i\,|\,k)=\boldsymbol{u}(k-1)+\Delta\hat{\boldsymbol{u}}(k\,|\,k)+\Delta\hat{\boldsymbol{u}}(k+1\,|\,k)+\cdots+\Delta\hat{\boldsymbol{u}}(k+i\,|\,k) \tag{2.9}$$

By using this new formulation of the linear model, the predicted outputs of the plant can be written in the matrix form as

$$\underbrace{\begin{bmatrix}\hat{y}(k+1|k)\\ \hat{y}(k+2|k)\\ \hat{y}(k+3|k)\\ \vdots\\ \hat{y}(k+N|k)\end{bmatrix}}_{\boldsymbol{Y}(k)} = \underbrace{\begin{bmatrix}\mathbf{CA}\\ \mathbf{CA}^2\\ \mathbf{CA}^3\\ \vdots\\ \mathbf{CA}^N\end{bmatrix}}_{\boldsymbol{\Psi}} \boldsymbol{x}(k) + \underbrace{\begin{bmatrix}\mathbf{CB}\\ \mathbf{CAB+CB}\\ \mathbf{CA^2B+CAB+CB}\\ \vdots\\ \sum_{i=0}^{N-1}\mathbf{CA}^i\mathbf{B}\end{bmatrix}}_{\boldsymbol{\Gamma}} \boldsymbol{u}(k-1) + \underbrace{\begin{bmatrix}\mathbf{CB} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{CAB+CB} & \mathbf{CB} & \mathbf{0} & \cdots & \mathbf{0}\\ \sum_{i=0}^{2}\mathbf{CA}^i\mathbf{B} & \mathbf{CAB+CB} & \mathbf{CB} & \cdots & \mathbf{0}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ \sum_{i=0}^{N-1}\mathbf{CA}^i\mathbf{B} & \sum_{i=0}^{N-2}\mathbf{CA}^i\mathbf{B} & \sum_{i=0}^{N-3}\mathbf{CA}^i\mathbf{B} & \cdots & \mathbf{CB}\end{bmatrix}}_{\boldsymbol{\Theta}} \underbrace{\begin{bmatrix}\Delta\hat{u}(k|k)\\ \Delta\hat{u}(k+1|k)\\ \Delta\hat{u}(k+2|k)\\ \vdots\\ \Delta\hat{u}(k+N-1|k)\end{bmatrix}}_{\Delta\boldsymbol{U}(k)} \tag{2.10}$$

Now the constraints in (2.8) can be incorporated into the cost function through direct substitution, and the new cost function can be rewritten as

$$\begin{aligned} V(k) &= \left\|\boldsymbol{\Psi}\boldsymbol{x}(k)+\boldsymbol{\Gamma}\boldsymbol{u}(k-1)+\boldsymbol{\Theta}\Delta\boldsymbol{U}(k)-\boldsymbol{T}(k)\right\|_{\mathbf{Q}}^2 + \left\|\Delta\boldsymbol{U}(k)\right\|_{\mathbf{R}}^2 \\ &= \left[\boldsymbol{\Psi}\boldsymbol{x}(k)+\boldsymbol{\Gamma}\boldsymbol{u}(k-1)-\boldsymbol{T}(k)\right]^T \mathbf{Q}\left[\boldsymbol{\Psi}\boldsymbol{x}(k)+\boldsymbol{\Gamma}\boldsymbol{u}(k-1)-\boldsymbol{T}(k)\right] \\ &+ \ 2\Delta\boldsymbol{U}(k)^T\boldsymbol{\Theta}^T\mathbf{Q}\left[\boldsymbol{\Psi}\boldsymbol{x}(k)+\boldsymbol{\Gamma}\boldsymbol{u}(k-1)-\boldsymbol{T}(k)\right] \\ &+ \ \Delta\boldsymbol{U}(k)^T\left[\boldsymbol{\Theta}^T\mathbf{Q}\boldsymbol{\Theta}+\mathbf{R}\right]\Delta\boldsymbol{U}(k) \end{aligned} \tag{2.11}$$

where:

$$\boldsymbol{T}(k) = \begin{bmatrix}\hat{\boldsymbol{r}}(k+1|k)\\ \hat{\boldsymbol{r}}(k+2|k)\\ \hat{\boldsymbol{r}}(k+3|k)\\ \vdots\\ \hat{\boldsymbol{r}}(k+N|k)\end{bmatrix}, \qquad \mathbf{Q} = \begin{bmatrix}\mathbf{Q}(1) & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{0} & \mathbf{Q}(2) & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{Q}(3) & \cdots & \mathbf{0}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{Q}(N)\end{bmatrix},$$

$$\text{and } \mathbf{R} = \begin{bmatrix}\mathbf{R}(1) & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{0} & \mathbf{R}(2) & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{R}(3) & \cdots & \mathbf{0}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{R}(N)\end{bmatrix}$$

Note that an alternative means of incorporating constraints into an optimization problem would be through the use of Lagrange multipliers, but this increases the order of the overall optimization problem.

To get the final form of the cost function, the N terms of $\Delta \boldsymbol{u}$ are replaced by an $(M\text{-}1)^{\text{th}}$-order polynomial function.

$$\Delta \boldsymbol{U}(k) = \underbrace{\begin{bmatrix} \mathbf{I} & 0 & 0 & \cdots & 0 \\ \mathbf{I} & \mathbf{I} & \mathbf{I} & \cdots & \mathbf{I} \\ \mathbf{I} & 2\mathbf{I} & 4I & \cdots & 2^{M-1}\mathbf{I} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{I} & (N-1)\mathbf{I} & (N-1)^2 I & \cdots & (N-1)^{M-1}\mathbf{I} \end{bmatrix}}_{\boldsymbol{\Omega}} \underbrace{\begin{bmatrix} \mu_0(k) \\ \mu_1(k) \\ \mu_2(k) \\ \vdots \\ \mu_{M-1}(k) \end{bmatrix}}_{\mu(k)} \tag{2.12}$$

where $\mathbf{I}$ is an n_u^{th}-order identity matrix, and the μ_i(k)s are the coefficients of the polynomial basis function describing the future changes in control inputs, $\Delta \hat{\boldsymbol{u}}(k+i \mid k)$. Equation (2.12) can now be substituted into equation (2.11) to give

$$\begin{aligned} V(k) = & \left[\boldsymbol{\Psi} \boldsymbol{x}(k) + \boldsymbol{\Gamma} \boldsymbol{u}(k-1) - \boldsymbol{T}(k)\right]^T \mathbf{Q} \left[\boldsymbol{\Psi} \boldsymbol{x}(k) + \boldsymbol{\Gamma} \boldsymbol{u}(k-1) - \boldsymbol{T}(k)\right] \\ & + \mu(k)^T \mathbf{F}^T \left[\boldsymbol{\Psi} \boldsymbol{x}(k) + \boldsymbol{\Gamma} \boldsymbol{u}(k-1) - \boldsymbol{T}(k)\right] \\ & + \mu(k)^T \mathbf{H}\, \mu(k) \end{aligned} \tag{2.13}$$

where $\mathbf{F} = 2\mathbf{Q}\boldsymbol{\Theta}\boldsymbol{\Omega}$ and $\mathbf{H} = \boldsymbol{\Omega}^T \left[\boldsymbol{\Theta}^T \mathbf{Q} \boldsymbol{\Theta} + \mathbf{R}\right] \boldsymbol{\Omega}$.

Equation (2.13) is the PFC form of the cost function and is the form used in the present thesis for the tracking control problem of the MDMS. The optimal polynomial function comes from the minimization of this cost function.

In equation (2.5) the tracking error weighting matrices, $\mathbf{Q}(i)$, are defined as positive semi-definite matrices, and the control move weighting matrices, $\mathbf{R}(i)$, are defined as positive definite matrices. This makes the Hessian in equation (2.13) positive definite (because $\mathbf{H} = \boldsymbol{\Omega}^T \left[\boldsymbol{\Theta}^T \mathbf{Q} \boldsymbol{\Theta} + \mathbf{R}\right] \boldsymbol{\Omega}$), and therefore, the cost function, $V(k)$, is convex. This means that the optimal solution is both unique and global [Fletcher, 1987]. Equation (2.13) is now minimized in the absence of con-

straints. This can be done as usual, by taking the gradient of the cost function and setting it equal to zero.

$$\nabla_{\mu} V(k) = \mathbf{F}^{T}\left[\boldsymbol{\Psi} \boldsymbol{x}(k) + \boldsymbol{\Gamma} \boldsymbol{u}(k-1) - \boldsymbol{T}(k)\right] + 2\,\mathbf{H}\,\boldsymbol{\mu}(k) = \mathbf{0} \tag{2.14}$$

$$\boldsymbol{\mu}_{opt}(k) = \frac{1}{2}\mathbf{H}^{-1}\,\mathbf{F}^{T}\left[\boldsymbol{T}(k) - \boldsymbol{\Psi} \boldsymbol{x}(k) - \boldsymbol{\Gamma} \boldsymbol{u}(k-1)\right] \tag{2.15}$$

In the unconstrained case of MPC, the optimal solution in (2.15) is a function of the current state, the previous input, and the desired setpoints. A feedback matrix, $\mathbf{K}_{\mathbf{MPC}}$, can be computed offline in order to achieve fast implementation online. A block diagram of the controller is shown in Figure 2.2.

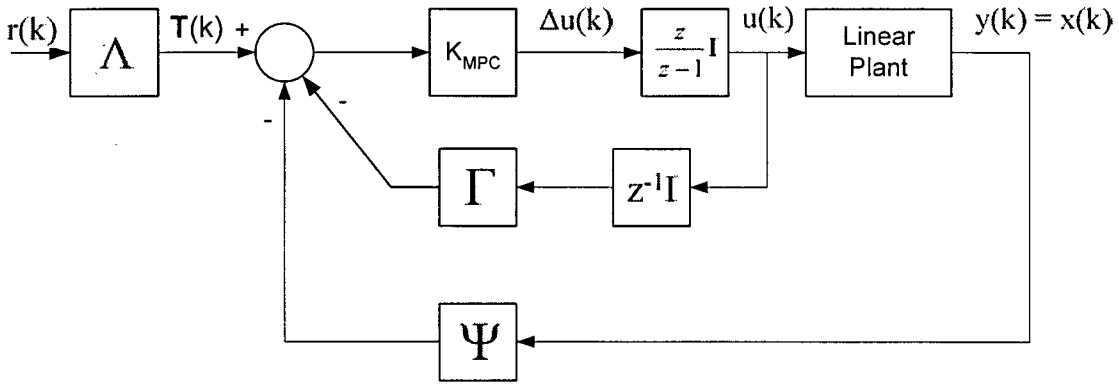


Figure 2.2 - Block diagram of the unconstrained predictive controller.

Unfortunately, inverting the Hessian in (2.15) can be rather difficult because MPC problems are known to be naturally ill conditioned. It is important to deal with these numerical issues early because the condition number of the Hessian directly affects the performance of the controller and will cause further problems when constraints are added to the optimization.

2.6.2 Numerical Issues of MPC

If the condition number of the cost function Hessian is large, then the numerical accuracy of (2.15) will be lost due to round off errors. In the worst case, if the condition number is too large, then the Hessian cannot be inverted and (2.15) will not provide a solution. Another way to see how the Hessian affects the performance of the controller is to look at the Hessian for the cost function in equation (2.13),

$$\mathbf{H} = \Omega^T \left[\Theta^T \mathbf{Q} \Theta + \mathbf{R} \right] \Omega \tag{2.16}$$

Here, **Q** and **R** are matrices containing the current and the future weights on the tracking errors and control input moves, respectively. It is easy to see how these matrices directly affect the Hessian, and therefore, the performance of the controller.

For the case of the unconstrained controller some of the numerical problems can be avoided by formulating the optimal solution as a least-squares problem. Maciejowski [2002] shows that the quadratic cost function is equivalent to the squared length of the vector in

$$\begin{bmatrix} \mathbf{S}_\mathbf{Q} \left\{ \Psi x(k) + \Gamma u(k-1) + \Theta \Omega \mu(k) - T(k) \right\} \\ \mathbf{S}_\mathbf{R} \Omega \mu(k) \end{bmatrix} \tag{2.17}$$

where $\mathbf{S}_\mathbf{Q}^T \mathbf{S}_\mathbf{Q} = \mathbf{Q}$ and $\mathbf{S}_\mathbf{R}^T \mathbf{S}_\mathbf{R} = \mathbf{R}$.

In view of this, by setting the length of this vector equal to zero, one can minimize the quadratic cost function. This can be accomplished by setting the vector itself equal to zero and then solving for μ(k). By setting the vector equal to zero rather than the length of the vector, unnecessary squaring of the ill-conditioned matrices Θ and Ω is avoided. Accordingly, the unconstrained solution in (2.15) becomes

$$\mu_{opt}(k) = \begin{bmatrix} \mathbf{S}_\mathbf{Q} \Theta \Omega \\ \mathbf{S}_\mathbf{R} \Omega \end{bmatrix} \backslash \begin{bmatrix} \mathbf{S}_\mathbf{Q} \\ 0 \end{bmatrix} \left\{ T(k) - \Psi x(k) - \Gamma u(k-1) \right\} \tag{2.18}$$

where \ is the "left division" operator (used in MATLAB). Many efficient methods exist for solving equation (2.18) without running into the numerical difficulties involved in taking the inverse of the Hessian, as in (2.15). A controller using the solution in (2.18) was implemented on the MDMS in [McCourt & de Silva, 2003].

Unfortunately, the least squares approach outlined above only works for the unconstrained solution. When constraints are involved, the length of the vector cannot always be set equal to zero, nor can the gradient of the cost function necessarily be set equal to zero as in equation (2.14). In this thesis active set methods are used to solve the constrained optimization problem. These methods typically involve taking the inverse of the Hessian. In view of this it is important to understand what factors affect its condition number so that ill posed problems can be avoided.

If λ_i is an eigenvalue of $\mathbf{A}$, then $c\mathbf{A}^k$ will have the eigenvalue $c\lambda_i^k$. Since the singular values are the square roots of the eigenvalues of $\mathbf{A}^H\mathbf{A}$[‡], it can be seen why the MPC problem tends to be ill conditioned. The Hessian in (2.16) contains the matrices Θ and Ω. In equation (2.10) it is seen that the matrix Θ contains increasing powers of the state transition matrix, $\mathbf{A}$ of (2.4). Assuming that the plant is stable, the eigenvalues of $\mathbf{A}$ will be on or within the unit circle and with increasing powers of $\mathbf{A}$ the corresponding eigenvalues (and therefore the singular values) will become smaller compared to unity. So Θ is ill conditioned and multiplying it by its transpose as in (2.16) will only increase the condition number.

The matrix Θ does make the Hessian ill conditioned, but not nearly as much as Ω. The formulation of Ω in equation (2.12) shows that as the number of basis functions is increased, the elements in the lower-right corner of the matrix become very large, while the top-left corner contains an identity matrix. Consequently, multiplying the Hessian by Ω and its transpose will greatly increase the condition number.

In conclusion, the two factors that most significantly affect the condition of the Hessian are the prediction horizon and the order of the polynomial function used to describe the control input moves. So, a convenient way to maintain the condition of the Hessian is to either keep the pre-

[‡] A^H is the conjugate transpose

diction horizon low or use a low-order polynomial basis function. The latter is what is usually done in PFC, with M=2 or M=3 in equation (2.3) being typical [Maciejowski, 2002].

2.7 Constrained Predictive Control

Naturally, in any system there will be physical limitations to the types and levels of inputs that can be applied and the nature of the resulting outputs. For the multi-module deployable manipulator system (MDMS), which is the plant that is used in the present investigation, the input constraints are the torque limits of the motors, and the output constraints are the joint motion limits. In the unconstrained controller, inputs are simply saturated when the constraint is violated and as a result the optimality with respect to the cost function is lost. If the constraints are formulated into the predictive controller, however, the controller will be aware of the expected constraints and should be able to take corrective action before the constraints are actually met. Furthermore, the addition of constraints also allows the predictive controller to naturally handle such common controller problems as integrator windup.

The constrained MPC problem may be expressed as the minimization of the cost function

$$\begin{aligned} V(k) = &\left[\boldsymbol{\Psi}\boldsymbol{x}(k)+\boldsymbol{\Gamma}\boldsymbol{u}(k-1)-\boldsymbol{T}(k)\right]^T \mathbf{Q}\left[\boldsymbol{\Psi}\boldsymbol{x}(k)+\boldsymbol{\Gamma}\boldsymbol{u}(k-1)-\boldsymbol{T}(k)\right] + \\ &\min_{\mu(k)}\left\{\boldsymbol{\mu}(k)^T \mathbf{F}^T\left[\boldsymbol{\Psi}\boldsymbol{x}(k)+\boldsymbol{\Gamma}\boldsymbol{u}(k-1)-\boldsymbol{T}(k)\right] + \boldsymbol{\mu}(k)^T \mathbf{H}\,\boldsymbol{\mu}(k)\right\} \end{aligned} \tag{2.19a}$$

subject to the constraints

$$\mathbf{G}\boldsymbol{\mu}(k) \le \boldsymbol{W} + \mathbf{E}\begin{bmatrix} \boldsymbol{x}(k) \\ \boldsymbol{u}(k-1) \end{bmatrix} \tag{2.19b}$$

where **G**, $\boldsymbol{W}$, and **E** are matrices used to describe the constraints on the system in terms of the coefficients in $\boldsymbol{\mu}$(k).

Since the argument of the minimization contains the coefficients of the control input function, all the constraints must be formulated in terms of these coefficients, as given in equation (2.19b). As a result of including the constraints, the controller changes from being a linear state feedback controller to a nonlinear controller. This happens because different constraints are active for different feedback states. If the constraints are linear, the optimal solution will become piecewise affine and the minimization of equation (2.19) is known as a multi-parametric quadratic programming (mp-QP) problem. The block diagram of the constrained predictive controller is shown in Figure 2.3. In comparison to Figure 2.2, it is noted that the predicted tracking errors are no longer simply multiplied by a matrix gain at each time step. Instead, the minimization of equation (2.19) must be performed at each sampling time to determine the optimal solution.

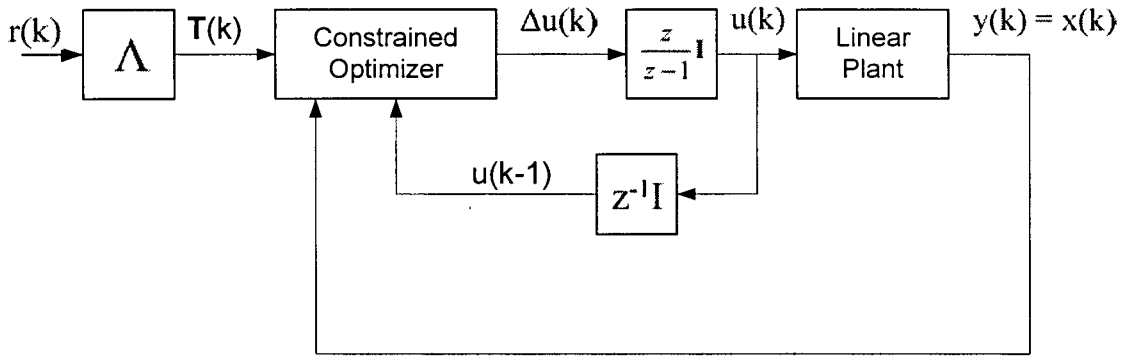


Figure 2.3 - Block diagram of the constrained predictive controller.

The online solution of the optimization problem (2.19) can be highly computational intensive. Therefore, methods for explicitly solving the mp-QP problem offline have to be considered. Such a scheme has been developed by Bemporad et al. [2002]. The basics behind determining the offline solutions to constrained QP problems are given in Appendix A.

2.7.1 Constraint Formulation

Before getting into the details of how to solve the constrained optimization problem, it is useful to formulate the constraints in terms of $\mu(k)$. If the constraints are placed only on the plant outputs, the inputs and the input changes, then using the linear model in (2.4) will result in linear constraints such as those in equation (2.19b). Linear constraints of this form define a convex poly-

tope in the $M \cdot n_u$-dimensional space and once again, the solution is guaranteed to be global, and unique [Fletcher, 1987].

Constraints on the outputs are related to the control input changes through equation (2.10). Similarly, constraints on the control inputs are related to the input changes through equation (2.9). Expressed in the form given in (2.19), these equations are:

$$\boldsymbol{Y}_{min} \le \boldsymbol{\Psi x}(k) + \boldsymbol{\Gamma u}(k-1) + \boldsymbol{\Theta\Omega\mu}(k) \le \boldsymbol{Y}_{max}$$

$$\boldsymbol{U}_{min} \le \underbrace{\begin{bmatrix} \mathbf{I} \\ \vdots \\ \mathbf{I} \end{bmatrix}}_{\mathbf{EYE_N}} \boldsymbol{u}(k-1) + \boldsymbol{\Omega\mu}(k) \le \boldsymbol{U}_{max}$$

$$\boldsymbol{\Delta U}_{min} \le \boldsymbol{\Omega\mu}(k) \le \boldsymbol{\Delta U}_{max}$$

which may be expressed as

$$\underbrace{\begin{bmatrix} \boldsymbol{\Theta\Omega} \\ -\boldsymbol{\Theta\Omega} \\ \boldsymbol{\Omega} \\ -\boldsymbol{\Omega} \\ \boldsymbol{\Omega} \\ -\boldsymbol{\Omega} \end{bmatrix}}_{\mathbf{G}} \boldsymbol{\mu}(k) \le \underbrace{\begin{bmatrix} \boldsymbol{Y}_{max} \\ -\boldsymbol{Y}_{min} \\ \boldsymbol{U}_{max} \\ -\boldsymbol{U}_{min} \\ \boldsymbol{\Delta U}_{max} \\ -\boldsymbol{\Delta U}_{min} \end{bmatrix}}_{\boldsymbol{W}} + \underbrace{\begin{bmatrix} -\boldsymbol{\Psi} & -\boldsymbol{\Gamma} \\ \boldsymbol{\Psi} & \boldsymbol{\Gamma} \\ \mathbf{0} & \mathbf{EYE_N} \\ \mathbf{0} & \mathbf{EYE_N} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{E}} \begin{bmatrix} \boldsymbol{x}(k) \\ \boldsymbol{u}(k-1) \end{bmatrix} \tag{2.20}$$

For the satellite tracking controller that is investigated in the present thesis, only a linear polynomial is needed (i.e., $M = 2$). Consequently, a simplification can be made for the constraints on $\Delta\hat{\boldsymbol{u}}(k+i \mid k)$ by constraining only the endpoints of the function ($\Delta\hat{\boldsymbol{u}}(k+i \mid k) = \boldsymbol{\mu}_0(k) + \boldsymbol{\mu}_1(k) \cdot i$). Specifically we have

$$\Delta\boldsymbol{U}_{min} \le \boldsymbol{\mu}_0(k) \le \Delta\boldsymbol{U}_{max}$$

and

$$\Delta \boldsymbol{U}_{min} \leq \boldsymbol{\mu}_0(k) + \boldsymbol{\mu}_1(k) \cdot (N-1) \leq \Delta \boldsymbol{U}_{max} \tag{2.21}$$

As mentioned earlier, the constrained predictive controller solves the optimal control problem in equation (2.19) at each time step. If active set methods are used for the optimization, then the worst-case computational time of the controller will increase with the number of constraints that are given in (2.20). In addition, each constraint in (2.20) defines a hyperplane in the $M \cdot n_u$-dimensional space of the control decision variables. As the number of constraints increases, the boundary of the convex polytope defined by (2.20) becomes smoother, and the neighboring hyperplanes become increasingly similar. Again, this can lead to numerical problems when neighboring constraints are active, so, care should be taken in how these constraints are defined. In particular, the explicit offline solution to the predictive control problem presented by Bemporad, et al. [2002] is very sensitive to these types of numerical problems. To understand why this happens, an introduction to offline multi-parametric quadratic programming (mp-QP) solution is presented next.

2.7.2 Multi-parametric Quadratic Programming

In the minimization of the cost function of the predictive controller, the parameters that are changing at each time step are the measured states of the plant (the robotic manipulator), the previous control inputs to the plant, and the state of the desired values (target position). The values of these parameters will determine the location of the unconstrained minimum point in the space defined by the optimization variables, $\boldsymbol{\mu}(k) \in \Re^{M \cdot n_u}$, and the location of this minimum point will correspond to a specific set of active constraints. In multi-parametric quadratic programming these regions of constant active sets are determined offline, and using the first-order Karush-Kuhn-Tucker (KKT) conditions, an affine solution is determined for each of these regions. The solution is piecewise affine in a space defined by the N_p parameters that change in the optimization at each time step; i.e., the n states, the n_u previous inputs, and the n_r current and predicted references. The parameter space is partitioned into a total of N_r regions.

In this thesis, the Multi-Parametric Toolbox (MPT) of MATLAB is used to find the N_r regions and the corresponding affine solutions. Each of the N_r regions is defined by an N_p-dimensional convex polytope. The boundaries of each polytope are defined such that the Lagrange multipliers

corresponding to the current active set of constraints remain positive and such that none of the constraints in equation (2.20) are violated (i.e., KKT conditions).

If any two constraints in equation (2.20) are very similar, then these will result in very long and thin partitioned regions. If these regions are too thin, when the mp-QP algorithm is crossing over the boundaries of one region to search for the solution in the next region, it might accidentally step over this very small region and create a hole in the partitioned space where the offline solution will not be defined. The step size of the search through the parameter space can be decreased to avoid the possibility of stepping over these thin regions, but this will increase numerical problems in the optimizations performed in this region. Further discussion on the implementation of offline mp-QP solutions will be found in chapters 4 and 5, where the simulations and real-time applications with mp-QP are performed. More details on the mp-QP algorithm are given in Appendix A [Bemporad, et al., 2002].

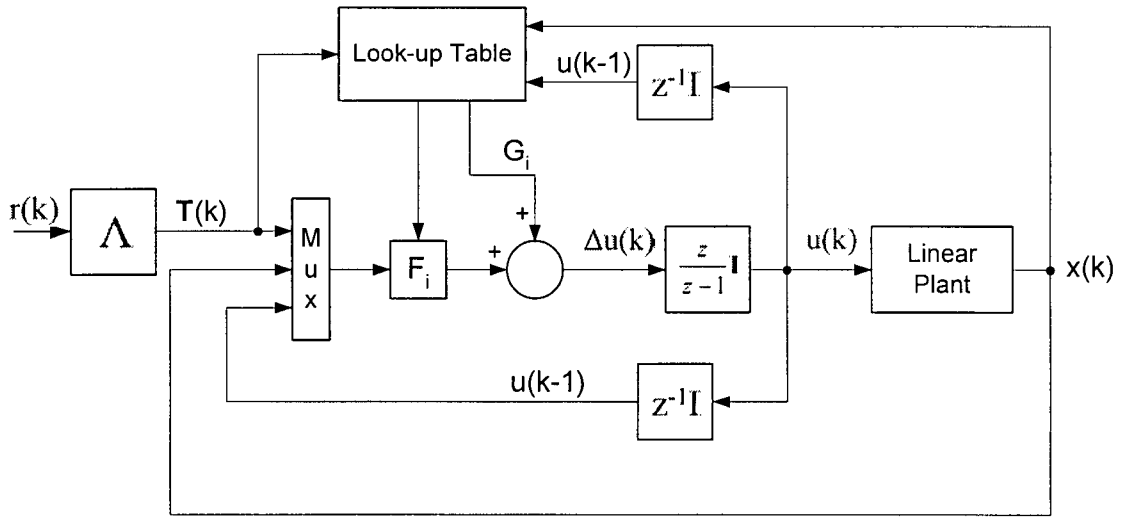


Figure 2.4 - Block diagram of the mp-QP controller.

In the block diagram shown in Figure 2.4, the optimizer of Figure 2.3 has been replaced by a gain, $\mathbf{F}_i$, and an offset, $\boldsymbol{G}_i$, which change depending on the current state of the manipulator and target, and the previous inputs.

2.8 Summary

In this chapter, the broad topic of model predictive control (MPC) was outlined. The choice of appropriate predictive control techniques was indicated for the robotic satellite-tracking problem addressed in Chapter 1. A predictive functional controller with an offline multi-parametric quadratic programming solution was formulated. It possesses the robustness and optimality of PFC with the online implementation speed of mp-QP. In the following chapters the details behind the construction and implementation of the predictive controller for the MDMS satellite-tracking problem will be presented.

CHAPTER 3:
PROBLEM FORMULATION

3.1 Preamble

In Chapter 1 the objectives, motivation, and the planned approach of the present investigation were outlined. The Multi-module Deployable Manipulator System (MDMS), which is the prototype manipulator that is used in autonomous satellite capture, as studied in the present thesis, was introduced. In Chapter 2, an introduction to Predictive Functional Control (PFC) and multi-parametric Quadratic Programming (mp-QP) was given, which forms the basis of the control approach that is used in the present work. In the present the overall problem that is studied in the thesis is formulated. In particular, the predictive controller is integrated into the MDMS, to form a joint-space controller for the MDMS, which will automate a satellite-capturing task.

The focus of the present chapter is to first build an analytical understanding for the dynamics of the target satellite and the chaser manipulator. The system dynamics forms the internal model used in the predictive controller. Furthermore, by understanding how the target (satellite) moves in the joint-space of the MDMS, the appropriate cost function to be penalized in the optimal control problem is determined.

3.2 Dynamics of the Manipulator System

The multi-module deployable manipulator system (MDMS) is the robot used in the in this thesis for investigating the control of the automated satellite capture problem. The prototype MDMS is restricted to planar motions only, in order to eliminate the gravity considerations. Furthermore, even though the MDMS has been designed to possess kinematic redundancy, this capability is not considered in the present investigation. Specifically, to reach the desired position and orientation of the target in planar motion, only three joints (3 degrees of freedom) are needed. Accordingly, only two modules of the MDMS are chosen, with the prismatic joint of the distal module not being used in the experimental investigation.

A schematic diagram of the manipulator as used in the present investigation is shown in Figure 3.1.

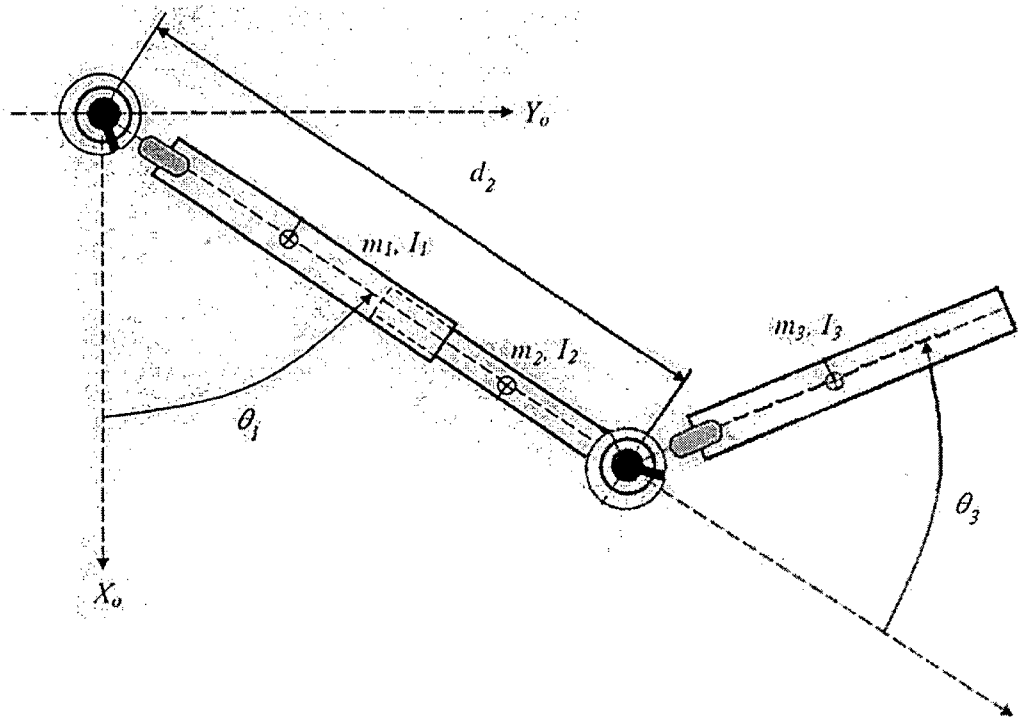


Figure 3.1 - Schematic diagram of the MDMS used for planar satellite capturing.

The manipulator in Figure 3.1 is assumed to be fully rigid, specifically with rigid links and rigid joints. A space-based manipulator may possess both joint and link flexibility, and in that case the corresponding model should include these effects [Cao, 1999]. Fortunately the prototype MDMS is known to be quite rigid, and the flexibility effects can be neglected for most practical purposes. If a flexible manipulator were to be used in the real-time verification of the controller developed in the present work, there should be sensors for measuring joint or link flexibility. In the absence of such sensors, an observer would be needed with the controller to estimate these states. However, the use of an observer to estimate joint and link flexibilities is beyond the scope of the present thesis.

The dynamics of the robotic system shown in Figure 3.1 may be determined using Lagrange equations, and then static and viscous friction terms may be added as generalized terms to give the vector-matrix equation,

$$\mathbf{M}(q)\ddot{q} + C(q,\dot{q}) + \mathbf{F}_s \operatorname{sgn}(\dot{q}) + \mathbf{F}_v\dot{q} = \tau \tag{3.1}$$

where $\mathbf{M}(q)$ is the inertia matrix for the robot and accordingly $\mathbf{M}(q)\ddot{q}$ contains the inertia force/torque terms, $C(q,\dot{q})$ is a vector of the centripetal and Coriolis force/torque terms, $\mathbf{F}_s \operatorname{sgn}(\dot{q})$ is the Coulomb friction, $\mathbf{F}_v\dot{q}$ is the viscous friction, and τ is a vector of joint forces and torques. The vectors $q, \dot{q}$ and $\ddot{q}$ are the joint positions, velocities, and accelerations, respectively. The absence of any gravity effects in equation (3.1) is due to the micro-gravity operating environment of the space-based system, and horizontal plane of operation of the ground-based manipulator. Details on the modeling of the MDMS in equation (3.1) are found in Appendix B.

As mentioned in Chapter 2, the predictive controller uses a linear model of the plant in the optimization of the cost function. However, robotic manipulators are globally nonlinear plants; therefore, a linearization of the MDMS must be made before the controller can be used. This may be achieved locally using Taylor series expansion about an operating point, or more appropriately using global linearization through feedback linearization where the inverse dynamics of the robot is applied in a feedback loop around the system as shown in Figure 3.2. This global linearization is possible in view of the availability of a complete nonlinear model of the MDMS.

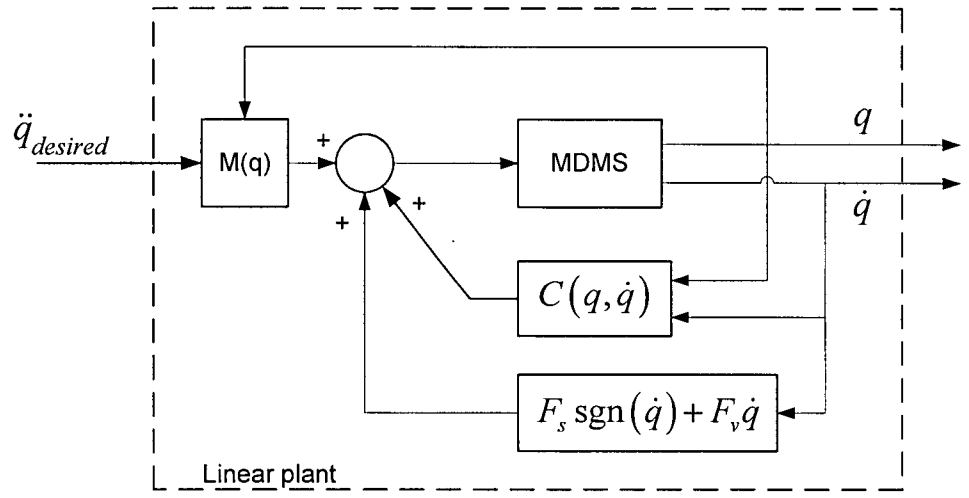


Figure 3.2 - Feedback linearization of the MDMS.

With the robot globally linearized as in Figure 3.2, the internal reference model in the controller becomes a double-integrator. The advantages of formulating the controller in this fashion is that the inputs have become the desired joint accelerations, and now, constraints can be put on the reference joint accelerations (the inputs) as well as the joint positions and velocities (the outputs). However, the inverse dynamics feedback loop does not take actuator constraints into account, so, the constraints on joint accelerations and velocities have to be sufficiently conservative to ensure that these constraints are not violated and the online optimization problem does not become infeasible.

Alternatively, a physical approach to linearizing a robot manipulator is to fit the joint motors with gear reducers with large speed reduction. In this manner, the joint torques/forces can be highly amplified and will form the inputs to the plant. The outputs will be the joint positions and velocities, and the nonlinear effects are left as unmeasured disturbances, which may be considered small when compared to the amplified inputs. Some of the disadvantages to placing large gear ratios on the joints are that the gear reduction greatly decreases the maximum speed of the joint, and the accuracy of the manipulator can be lost due to added nonlinearities in the gears, such as backlash and friction.

Fortunately, however, fast manipulation speeds are not needed and are undesirable for space manipulators because of their long, flexible structure and the sensitivity of the equipment that is being manipulated. In addition, the ground-based prototype of the MDMS has been designed with

harmonic gear drives, which virtually eliminate backlash and have very high gear ratios [de Silva, 1989; Wong, 2000]. For these reasons the predictive controller was implemented as a direct, low-level controller on the MDMS.

The joint positions and velocities are combined into a state vector, $x = \left[q^T, \dot{q}^T\right]^T \in \Re^n$, and the system dynamics in equation (3.1) may be expressed in the state-space form as

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} \dot{q} \\ -\mathbf{M}^{-1}(q)\left\{C(q,\dot{q}) + \mathbf{F}_s \operatorname{sgn}\dot{q} + \mathbf{F}_v\dot{q}\right\} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1}(q) \end{bmatrix} \mathbf{K} \cdot \boldsymbol{u} \tag{3.2}$$

where vector $\boldsymbol{u}$ represents the command voltages (inputs) from the servo card of the control computer and $\mathbf{K}$ is a conversion factor containing the corresponding gear ratios and motor torque constants.

The model is first linearized about an operating configuration with the prismatic joint fully retracted and the distal module at an angle of zero as defined in Figure 3.1. Then, the model is discretized with a sampling time of 5 milliseconds, which is large enough for the optimal solution to be found within the sampling period without compromising the performance of the controller. The linearization and discretization are done using the 'dlinmod' function in MATLAB, to obtain

$$x(k+1) = \begin{bmatrix} 1 & 0 & 0 & 0.005 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0.005 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0.0049 \\ 0 & 0 & 0 & 0.9951 & 0 & 0.0105 \\ 0 & 0 & 0 & 0 & 0.9928 & 0 \\ 0 & 0 & 0 & 0.0107 & 0 & 0.9619 \end{bmatrix} x(k) + \begin{bmatrix} 1.011E^{-5} & 0 & -1.887E^{-5} \\ 0 & 1.394E^{-6} & 0 \\ -2.214E^{-5} & 0 & 6.824E^{-5} \\ 0.004 & 0 & -0.0075 \\ 0 & 0.0006 & 0 \\ -0.0088 & 0 & 0.0271 \end{bmatrix} u(k) \tag{3.3}$$

3.3 Target Kinematics

In general, the controller developed in this thesis may be used for robotic tracking of any moving target subject to the kinematic and dynamic limitations and the operating bandwidth of the manipulator. However, in the simulations and experiments in chapters 4 and 5, the target is assumed

to be a free-floating satellite; i.e., passing by in the absence of external forces or torques. In Chapter 1, practical reasons were given for the need to have the MDMS capture a free-floating satellite.

As mentioned earlier, the present mode of operation of the MDMS possesses three degrees-of-freedom in planar motion. This is because unlike catching a ball, a satellite typically has a safe grappling point where the manipulator must try and grab it, and it is desirable for the manipulator to have both the correct position and orientation when the capture is being executed.

The automated capturing task is implemented to alleviate some of the difficulties encountered when capturing a spinning satellite. The grappling point on the satellite is not moving in a straight line, but rather spinning around the moving center of mass of the target, as shown in Figure 3.3.

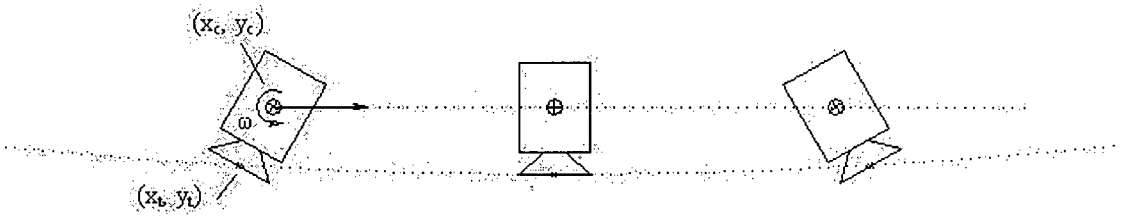


Figure 3.3 - A sequence of motion of a spinning satellite.

One of the many advantages of the MDMS is its simple inverse kinematics solution. For the non-redundant manipulator shown in Figure 3.1, the inverse kinematics solution is uniquely given by

$$\theta_1 = \text{atan2}(y_e - l_w \sin\theta_e,\quad x_e - l_w \cos\theta_e),$$

$$d_2 = \sqrt{(x_e - l_w \cos\theta_e)^2 + (y_e - l_w \sin\theta_e)^2}, \tag{3.4}$$

$$\theta_3 = \theta_e - \theta_1$$

where (x_e, y_e, θ_e) describes the end-effector position and orientation, and l_w is the length from the end-effector to the center for the third joint. Equation (3.4) can be used to express the position and orientation of the grappling point on the satellite in the joint space of the MDMS.

The target point (x_t, y_t) on the satellite is located a distance l_t from the center of mass of the satellite at (x_c, y_c), and is oriented at the angle θ_t. The center of mass is moving at a constant velocity of $v = \left[v_x, v_y\right]^T$, and spinning with constant angular velocity of ω. As a result, the position of the grappling point can be expressed as

$$\begin{aligned} x_t &= x_0 + v_x \cdot t - l_t \cos(\theta_0 + \omega \cdot t), \\ y_t &= y_0 + v_y \cdot t - l_t \sin(\theta_0 + \omega \cdot t), \\ \theta_t &= \theta_0 + \omega \cdot t, \end{aligned} \tag{3.5}$$

where (x_0, y_0, θ_0) is the position and orientation of the center of mass at $t=0$. The substitution of (3.5) into (3.4) will give the joint-space position of the target, and the result is formed into a vector of joint references, $\boldsymbol{r}(k)$, for the controller. From both the position of the target in (3.5) and the inverse kinematics of the robot in (3.4), it can be seen that $\boldsymbol{r}(k)$ is a very nonlinear function of the measured target velocities and time.

In the present controller the future position of the target is predicted, and the controller makes a decision on its control action based on the current and predicted joint tracking errors. These future reference vectors are chained together to form the vector $\boldsymbol{T}(k) = [\boldsymbol{r}(k+1|k)^T, \boldsymbol{r}(k+2|k)^T, \ldots]^T$ as discussed in Chapter 2. The vector $\boldsymbol{T}(k)$ is used in the definition of the cost function (2.11) and since it is not a function of the control input, it can be computed prior to the optimization.

In the unconstrained controller in equation (2.18), $\boldsymbol{T}(k)$ is computed and an error term is formed. The result is then multiplied by a gain matrix, which is computed offline as the optimal minimization of the cost function. We get

$$\boldsymbol{\mu}_{opt}(k) = \mathbf{K}_{\mathrm{MPC}} \cdot \left\{ \boldsymbol{T}(k) - \boldsymbol{\Psi} \boldsymbol{x}(k) - \boldsymbol{\Gamma} \boldsymbol{u}(k-1) \right\}$$

In the constrained controller, the vectors $\boldsymbol{T}(k)$, $\boldsymbol{x}(k)$ and $\boldsymbol{u}(k\text{-}1)$ are the N_p parameters of the optimization and in the mp-QP algorithm they form the parameter space $\Re^{N_p}$ that is partitioned in the offline optimization (see Appendix A). The mp-QP algorithm computes the constrained optimal minimization of the controller cost function offline, and produces a look-up table for fast online implementation. Unfortunately, it was found that both the length of time needed to compute the offline solution, and the size of the look-up table, increased very fast with the size of the parameter space $\Re^{N_p}$. In order to decrease the size of the parameter space and increase the computational efficiency, a second-order Taylor series approximation of the model is formulated into the problem. We get

$$\hat{\boldsymbol{r}}(k+i\,|\,k)=\boldsymbol{r}(k)+\dot{\boldsymbol{r}}(k)\cdot(T_s i)+\frac{1}{2}\ddot{\boldsymbol{r}}(k)\cdot(T_s i)^2 \tag{3.6}$$

Here $\dot{\boldsymbol{r}}(k)$ and $\ddot{\boldsymbol{r}}(k)$ are vectors of the current joint reference velocities and accelerations, respectively, and T_s is the sampling time of the controller. Equation (3.6) assumes that the current joint acceleration is constant throughout the prediction horizon. Since the window of time in which the controller looks into the future is small relative to the dynamics of the target, the approximation given by equation (3.6) is adequate.

When the joint velocities and accelerations are added to the problem, care must be taken when passing near, or through, singular configurations where the velocities and accelerations can become very large. Some examples of methods that deal with singularities involve reparameterizing the desired path [Loyd, 1998], or adding damping factors to the singular values of the manipulator Jacobian that are approaching zero [Nakamura, 1991]. Alternatively, Schreiber, et al. [1999] solved the singularity problem by constraint optimization. The controller developed in this thesis is a constrained optimal controller; therefore, it would be natural to adopt the constraint optimization solution to the singularity problem. There is particular advantage of the MDMS in this regard, as it has fewer singular configurations than a standard, all revolute joint manipulator. For the setup of the MDMS shown in Figure 3.1 there are no singularities in the operating space of the manipulator. The only singularity occurs when joints 1 and 3 occupy the same space, but the limits of the prismatic joint prevent the robot from ever reaching that configuration.

3.4 The Control System

The model of the manipulator has been formulated in the previous sections of the present chapter. The controller can be constructed now assuming that the expected joint-space position of the target is known. As mentioned earlier, the expected joint-space location of the target has been chosen as the reference value. Typically, in robot motion control, a trajectory is generated to specify how the controller should move the manipulator between any two points. However, since the problem addressed in the present thesis is a robotic capturing task, the robot must somehow span the distance between some initial starting position and the desired position of the target before it is able to follow the reference point like a standard trajectory tracking problem.

Some work has been done in the area of online trajectory planning for robot catching, for example, [Croft, et al., 1998] and [Park & Lee, 1992]. However, like for any online optimization problem, in the controller developed in this thesis, simplifications must be made in order to achieve acceptable computation speeds. In [Croft, et al., 1998] the planner is used to direct the robot from its initial configuration to a minimum time interception point using an active prediction planning and execution (APPE) algorithm, then switching to fine motion tracking and grasping algorithms once interception is achieved. In [Park & Lee, 1992] the target moves in a straight line along a conveyor and the initial position of the end-effector is overtop of the conveyor, so the trajectory optimization is reduced to a one-dimensional problem.

The controller developed in this thesis directly minimizes the joint motion differences between the reference target and the current state of the manipulator. However, it does this by trying to anticipate the motions of both the target and the manipulator. The controller is tuned to provide the desired tracking performance, and output constraints are used to control the speed at which the joints approach their targets.

3.5 Integration of Predictive Control

As mentioned in Chapter 2, the cost function of the present predictive controller minimizes the predicted reference tracking errors and the corresponding control actions. The cost function and the constraints are formulated as

$$V(k) = \left[\Psi x(k) + \Gamma u(k-1) - T(k)\right]^T \mathbf{Q} \left[\Psi x(k) + \Gamma u(k-1) - T(k)\right] + \min_{\mu(k)} \left\{ \mu(k)^T \mathbf{F}^T \left[\Psi x(k) + \Gamma u(k-1) - T(k)\right] + \mu(k)^T \mathbf{H}\, \mu(k) \right\} \tag{3.7}$$

$$s.t. \quad \mathbf{G}\mu(k) \le W + \mathbf{E}\begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix}$$

However, in the present problem the predicted joint references are a function of the current joint-space position, velocity, and acceleration of the target. This current state of the target is combined into a state vector, $x_r(k) = [r(k)^T, \dot{r}(k)^T, \ddot{r}(k)^T]^T$, and the future references can be expressed as

$$T(k) = \begin{bmatrix} \mathbf{I} & T_s\mathbf{I} & T_s^2\mathbf{I} \\ \mathbf{I} & 2T_s\mathbf{I} & (2T_s)^2\mathbf{I} \\ \vdots & \vdots & \vdots \\ \mathbf{I} & NT_s\mathbf{I} & (NT_s)^2\mathbf{I} \end{bmatrix} \begin{bmatrix} r(k) \\ \dot{r}(k) \\ \ddot{r}(k) \end{bmatrix} = \Lambda x_r(k) \tag{3.8}$$

The current state vector of the robot, the previous control input, and the current state vector for the target, can now all be combined into a single vector of optimization parameters, $\xi(k) = [x(k)^T, u(k-1)^T, x_r(k)^T]^T$ in the space $\Re^{N_p}$. Then the cost function becomes

$$V(k) = \frac{1}{2}\xi(k)^T \mathbf{Y}\, \xi(k) + \frac{1}{2}\mu(k)^T \mathbf{H}\, \mu(k) + \xi(k)^T \mathbf{F}\, \mu(k) \tag{3.9}$$

where:

$$\mathbf{Y} = 2\begin{bmatrix} \Psi & \Gamma & -\Lambda \end{bmatrix}^T \mathbf{Q} \begin{bmatrix} \Psi & \Gamma & -\Lambda \end{bmatrix}$$

$$\mathbf{H} = 2\Omega^T \left(\Theta^T \mathbf{Q} \Theta + \mathbf{R}\right)\Omega$$

$$\mathbf{F} = 2\begin{bmatrix} \Psi & \Gamma & -\Lambda \end{bmatrix}^T \mathbf{Q}\Theta\Omega$$

Equation (3.9) is now in the form of the cost function used in the mp-QP solver in the Multi-Parametric Toolbox (see Appendix A). The constraints in equation (2.20) also must be formulated in terms of the parameter vector, $\xi(k)$. This gives

$$\mathbf{G}\mu(k) \leq \boldsymbol{W} + \mathbf{E}\xi(k) \tag{3.10}$$

where:

$$\mathbf{G} = \begin{bmatrix} \boldsymbol{\Theta\Omega} \\ -\boldsymbol{\Theta\Omega} \\ \boldsymbol{\Omega} \\ -\boldsymbol{\Omega} \\ \boldsymbol{\Omega} \\ -\boldsymbol{\Omega} \end{bmatrix}, \boldsymbol{W} = \begin{bmatrix} \boldsymbol{Y}_{max} \\ -\boldsymbol{Y}_{min} \\ \boldsymbol{U}_{max} \\ -\boldsymbol{U}_{min} \\ \Delta\boldsymbol{U}_{max} \\ -\Delta\boldsymbol{U}_{min} \end{bmatrix} \text{ and } \mathbf{E} = \begin{bmatrix} -\boldsymbol{\Psi} & -\boldsymbol{\Gamma} & \mathbf{0} \\ \boldsymbol{\Psi} & \boldsymbol{\Gamma} & \mathbf{0} \\ \mathbf{0} & \mathbf{EYE}_N & \mathbf{0} \\ \mathbf{0} & \mathbf{EYE}_N & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

Since the matrices **Y**, **H**, and **F** from (3.9) and **G**, $\boldsymbol{W}$, and **E** from (3.10) are all that is needed for the mp-QP algorithm; essentially the problem formulation is a matter of determining these matrices. However, the number of constraints in equation (3.10) is proportional to the size of the prediction horizon, N. After implementing the mp-QP algorithm it was found that even if constraints are placed only on the inputs, the size of the resulting look-up table became unmanageably large, even for small values of N, for example N>5. In the implementation, the look-up table became too large, and besides that the solution would have taken days to compute, even on the 2.4 GHz Pentium4 PC, which was used. Furthermore, the mp-QP algorithm was used to geometrically partition the N_p-dimensional space, based on the constraints in (3.10). As the number of constraints increased, the size of the partitioned regions became very small and this caused numerical problems in the mp-QP solver.

To decrease the size of the problem, and reduce the number of numerical problems, the constraints are defined only at a select few points, rather than at every step into the horizon. This makes the control solution sub-optimal. Yet the controller is found to be capable of realizing the

defined constraints. The results in chapters 4 and 5 show an improvement in the performance of the controller over the unconstrained solution.

3.6 Integration of Constraints

The controller must take the manipulator from its initial configuration to an interception point, and then continue to track the target. The tracking may be aborted if the target moves out of range of operation. It is desirable to have the controller tuned so that the joint space tracking error is minimized during the time that the target is within the operating space of the manipulator. However, when the controller is tuned to give the desired tracking performance, the initially large reference errors lead to large inputs torques. In addition, as the manipulator approaches the target, large joint velocities are generated. These are undesirable effects because the target is a satellite floating in a micro-gravity environment, and any impact forces could possibly be damaging, and would cause the target path to change directions and possibly float away out of control.

Constraints are added to control how smoothly the manipulator approaches the target. Since the plant is linearized, input constraints can be used to put limits on the joint accelerations, and constraints on the change in the input can provide jerk limits. The joint velocities are outputs of the controller and output constraints are already included in equation (3.10).

The controller is implemented in joint space, as a physical necessity, but it is the task space maximum approach velocities that are important. However, the task space velocities are nonlinearly related to the joint space velocities through the robot kinematics, and constraints on the maximum task space velocity would lead to nonlinear constraints in joint space. Nonlinear constraints are not considered in this thesis, and consequently the constraints on the joint velocities only provide a rather conservative upper bound on the task space velocity.

Since the target can move in and out of the operational space of the manipulator, the joint limits can be added as constraints to the controller. By placing joint limits as constraints the controller is made aware of where these limits are ahead of the control actions, and will therefore be able to make better predictions on control actions.

3.7 Tuning the Controller

In the preceding development, we have come across many parameters that can be used to tune the performance of the controller. The available tuning parameters and a brief description on how they affect the performance are indicated in Table 3.1.

Table 3.1 - Tuning parameters for the predictive controller.

Tuning Parameter	Tuning Effect
Prediction horizon, N	The prediction horizon can be increased to allow the controller to anticipate the future movements of the target and the manipulator based on internal reference models
Order of input basis function, M	The order of the input basis function can be adjusted to affect the aggressiveness of the generated inputs
Weight matrices **Q** & **R**	The weighting matrices can be adjusted to determine the relative cost of the tracking errors and control input moves
Constraints	When constraints are formulated into the problem the controller becomes aware of its limitations and can make more knowledge-able decisions for the control inputs

There are some general guidelines for tuning these parameters [Maciejowski, 2002]. However, they are very general, and much of the tuning done on the present predictive controller is based on the experience and knowledge gained from simulations and experimentation with the MDMS.

There are many advantages to tuning the controller in simulation. One of the major advantages is that offline solutions to the optimal mp-QP problem are very computationally intensive and, depending on the size of the problem, computation times can range from a few minutes to several hours or even days. In simulation, the minimization of (3.9) subject to the constraints in (3.10) can be done at each time step using standard QP optimization techniques, and once the desired results have been achieved the offline solution can be computed using the mp-QP solver.

Unfortunately, tuning the controller in simulation will only bring the controller close to the actual tuned parameters of the experimental system. The fine-tuning of the controller on the prototype

MDMS is done through the tedious process of repeated parameter adjustments followed by off-line computations of the mp-QP problem. However, the fine-tuning adjustments are found to be needed only on the weighting parameters given in the matrices **Q** and **R**.

In addition to tuning the controller for the desired performance a dead-band compensator is added to adjust for the static friction in the prototype manipulator. The solution presented in [Tafazoli, 1996] is used in the present implementation, as given by

$$\Delta \boldsymbol{u}_{deadband}(k) = \boldsymbol{u}_d \operatorname{sgn}\left(\boldsymbol{r}(k) - \boldsymbol{y}(k)\right) \tag{3.11}$$

where $\boldsymbol{u}_d$ is the magnitude of the dead-band compensation term. The compensation term calculated in equation (3.11) is then added to current input decision.

3.8 Summary

A schematic diagram of the manipulator (MDMS) that is used for the satellite-capturing problem in the present thesis is shown in Figure 3.1. The dynamic model of this system was formulated in this chapter, and was used to create a discrete-time, linearized, internal reference model for the controller. One advantage of the chosen structure of the manipulator is that it provides a unique inverse kinematics solution for a given position and orientation of the end-effector. The inverse kinematics are used to map the position and orientation of the target into the joint space of the manipulator, and a second-order Taylor series approximation to the joint space path is used to predict the future joint space position of the target.

The model of the MDMS was integrated into a constrained predictive controller. This controller could be used to first bring the manipulator from its initial configuration to an interception point, and then continue to follow the target trajectory within the operating space of the manipulator. The developed predictive controller is summarized in the block diagram of Figure 3.4.

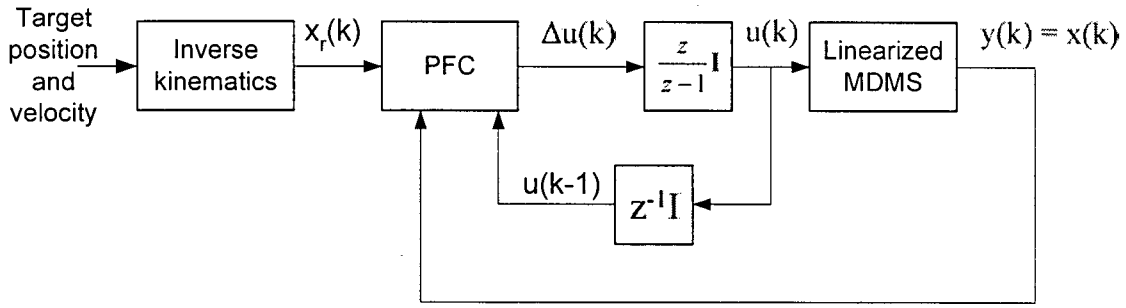


Figure 3.4 - Block diagram of the predictive controller implemented on the MDMS.

CHAPTER 4:
SIMULATION STUDY

4.1 Preamble

In this chapter the unconstrained and constrained predictive controllers, as formulated in the previous chapter, are used in a computer simulation of the ground-based prototype of the multi-module deployable manipulator system (MDMS). The conceptual space manipulator, which is used in the simulation studies, is similar to the ground-based system, but it is situated on a mobile base that traverses across an orbiting platform. Control momentum gyros (CMG's) located within the platform are used to sense the orientation and the flexibility in the platform. Previous work on the space-based manipulator has considered the regulation of the manipulator and the platform with flexibilities [Cao, 1999; Zhang, 2002]. As indicated before, flexibility issues are not considered in this thesis, even though their inclusion amounts to the modification of the dynamic model of the MDMS and is straightforward albeit with increased model complexity. Furthermore, effects of using redundant manipulators in the satellite-capturing problem are left for future work, even though the prototype MDMS has the capability of adding redundant degrees of freedom to the robotic system. In particular, the movement of the mobile base along the platform is not considered in the present work. The focus of the present investigation is to study the effectiveness of using constrained predictive optimal control in the interception and tracking of a moving target using an autonomous robot.

The flexibility and redundancy of the manipulator are not the only characteristics of the space-based system that are different from the ground-based system. In the space-based system the base of the manipulator is fixed to a free-floating platform, so the movement of the manipulator will affect the position of the target relative to the base of the robot. In this thesis it is assumed that 1) constraints can be placed on the operational speed of the manipulator to limit the forces at the base of the robot, and 2) the mass of the platform base is sufficiently large so that any forces generated at the base of the robot will result in a negligible movement of the base relative to the target.

4.2 Unconstrained Predictive Control

In the present investigation, first an unconstrained predictive controller was developed and implemented on the MDMS. From the initial tests using this control scheme, the prediction horizon, N, the order of the input function, M, and the location of the coincidence points were chosen. Much of the tuning in predictive control is based on the experience gained from computer simulations, and the simulations on the unconstrained controller have helped build the required knowledge base of tuning. The unconstrained controller has been tuned to give the desired tracking performance without considering the transient effects during the interception of the target.

In the following sections, a brief description is given on how the different controller parameters affect the controlled response. These parameters are discussed separately. Since the effects of these parameters are interconnected, in the actual tuning of the controller these parameters are adjusted concurrently.

4.2.1 Effects of Prediction Horizon

In many predictive controllers only the current reference point is used in the determination of the optimal control signal. However, it is known that if the future knowledge of the target position is penalized in the cost function, the tracking performance can be improved [McCourt & de Silva, 2003]. The predicted position allows the controller to anticipate the movement of the target and ultimately make better decisions to minimize the tracking error.

The predictions of the future position of the target are based on a second order approximation of the joint space position of the target (Equation (3.6)). Furthermore, some level of uncertainty will always be present in the internal model of the MDMS. In view of this, in the construction of the controller, the length of the prediction horizon is really a measure of how much trust is put into the prediction abilities of the models.

The degree of trust that could be put into the model varies depending on how fast the target is moving. In the simulations, it was found that there was not one exact prediction horizon that gave the best result for the tracking of every target, but values less than 50 were adequate. Pre-

diction horizons larger than 50 would still converge to the target position, but the response would either oscillate about the target position or would follow the target with a large steady state error.

In practice, however, even if exact models of the target and manipulator are used there is still a limit as to how large the prediction horizon can be made. As the prediction horizon, N, is increased, higher order basis functions (i.e., larger values of M) are need to fit the nonlinear path of the target. However, if both M and N are large, then the optimization becomes very ill-conditioned and the accuracy of the computed solution is lost (Chapter 2 gives details on PFC). Furthermore, when constraints are later added to the problem, the size of the problem will increase and it will take longer to converge to an optimal solution. Essentially then, the prediction horizon of the controller is limited by the numerical precision and CPU speed of the computer that is being used.

4.2.2 Order of the PFC Basis Function

In the formulation of the controller, the input moves, $\Delta\hat{\boldsymbol{u}}(k+i\,|\,k)$, are described by a set of M polynomial basis functions. In general, low values of M are used (e.g., 1 or 2) to try and keep the number of variables in the optimization low. Since the solution to the unconstrained problem can be determined offline, if the unconstrained controller is ever used, keeping the number of variables in the optimization low is not a major concern.

The present controller is implemented with constraints, and low values are used for M. It was found that using low values of M help generate smoother control signals. For the speed of the targets that were considered in the simulations, the higher values of M were found to improve the tracking performance by a negligible amount (if at all), but the inputs that were generated were very aggressive and oscillatory. A value of M=2 was used in the final version of the controller.

4.2.3 Location of Coincidence Points

As indicated in Chapter 3, to reduce the size of the constrained optimization problem, and decrease the numerical sensitivity, the constraints are defined only at select points throughout the horizon. In a similar fashion, the future tracking errors penalized in the controller cost function can be penalized only at specific points, known as coincidence points. Coincidence points are

typically used in PFC. In the unconstrained case, if the number of coincidence points is equal to the number of decision variables, M, then no optimization is needed. This is true because there are M tracking errors that are penalized in the cost function, and there are M decision variables, so an exact solution to the problem can be found. When an exact solution is computed, however, the controller is only concerned with minimizing the error at the M coincidence points and is not as concerned with staying on the reference path.

When there are more coincidence points than optimization variables, the computed solution provides a 'best fit' to the predicted target positions, so it only reduces the average predicted tracking error of the coincidence points. It was found that the performance of the controller was improved by selecting more coincidence points near the beginning of the prediction horizon, and using only one coincidence point at the end of the horizon. By placing the coincidence points in this fashion throughout the prediction horizon, the controller is more concerned with staying on the desired path at the beginning of the horizon, rather than at the end. The points at the end of the horizon serve a different purpose; they help the controller be more aware of upcoming constraints, and they provide the controller a general idea of where the target is moving towards in the near future.

4.2.4 Tuning of Weighting Matrices

Recall the cost function that is minimized in the predictive controller:

$$V(k) = \sum_{i=1}^{N} \left[\left\| \hat{y}(k+i \mid k) - \hat{\boldsymbol{r}}(k+i \mid k) \right\|_{\mathbf{Q}(i)}^2 + \left\| \Delta \hat{\boldsymbol{u}}(k+i-1 \mid k) \right\|_{\mathbf{R}(i)}^2 \right] \tag{4.1}$$

The matrices $\mathbf{Q}(i)$ determine the amount that tracking errors at the future time $t = k+i$ are penalized. Likewise, $\mathbf{R}(i)$ penalizes the control input changes at $t = k+i$. Adjustments of these matrices must be made relative to one another in order to affect the output of the controller. In particular, as intuitively clear, increasing all of the matrices by the same factor simply multiplies the cost function by that factor and does not change the location of the minimum solution.

In the tracking and interception of a satellite it is not only important that the manipulator has the same position as the target, but it should approach the target with the same velocity so that the robot does not crash into the target when trying to capture it. The outputs measured from the

MDMS are the joint positions and velocities, and likewise, the joint-space positions and velocities of the target grappling point are used as reference values. The matrix $\mathbf{Q}(i)$ now weighs the importance of minimizing position and velocity errors at time $t = k+i$.

The values in $\mathbf{Q}(i)$ must be chosen carefully. It is important to approach the target at the same velocity, but if the weights on the velocity tracking errors are made too large, and the target is moving away from the end-effector, then the response will be so highly damped that the manipulator cannot catch the target before it moves out of range. In the final selection of the controller weights, velocity tracking errors were penalized only at the end of the horizon. This allowed the controller to become aware of the target velocity, but focused the efforts on minimizing the position tracking error.

4.2.5 Results from the Unconstrained Controller

The optimal solution to the predictive controller is computed in the absence of constraints. Since in a real system there are always constraints, when the unconstrained controller is implemented the command signals computed by the controller are saturated at the maximum output of the servo card (±10 Volts).

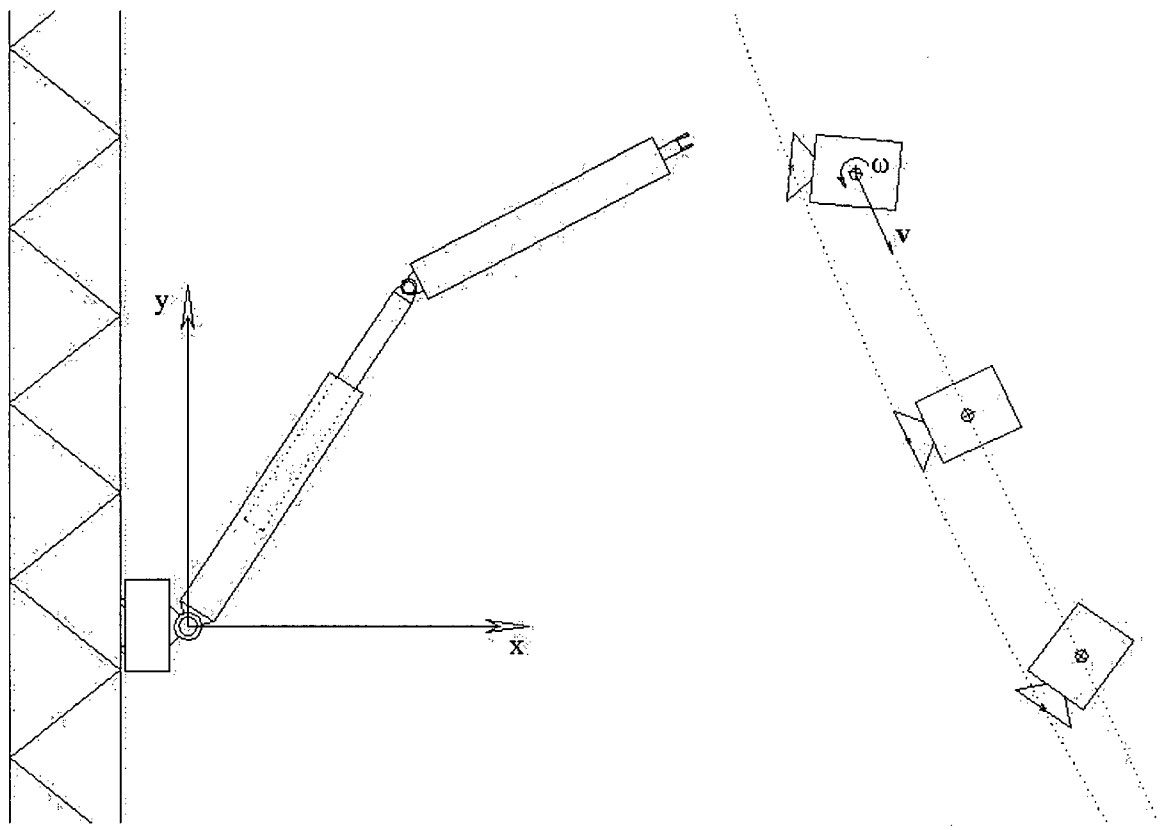


Figure 4.1 - The simulation model of robotic satellite capture.

In the simulations, the manipulator starts from rest, lying parallel to its platform base, and the MDMS attempts to catch a satellite that is moving with velocity $v = [v_x \quad v_y]^T$, and spinning with angular velocity ω, as shown in Figure 4.1.

Several different target cases are used. As an example, consider a target moving parallel to the platform with velocity $v = [0 \ -0.05]^T$ m/s, and rotating with angular velocity $\omega = \pi/60$ rad/s. In Figure 4.2 the advantages of using of using predictions in the controller are demonstrated. The simulated target point was initially 0.9 meters away from the end-effector with an initial mis-alignment of 75 degrees. When no predictions were used for the future positions of the target or the manipulator, the MDMS was able to reduce the steady tracking error to approximately 0.008 meters and 1 degree in 2 seconds. When a prediction horizon of 30 and the second order approximation of the target (Equation (3.6)) were used the tracking error was reduced to less than $5x10^{-4}$ meters and less than 0.5 degrees in the same amount of time.

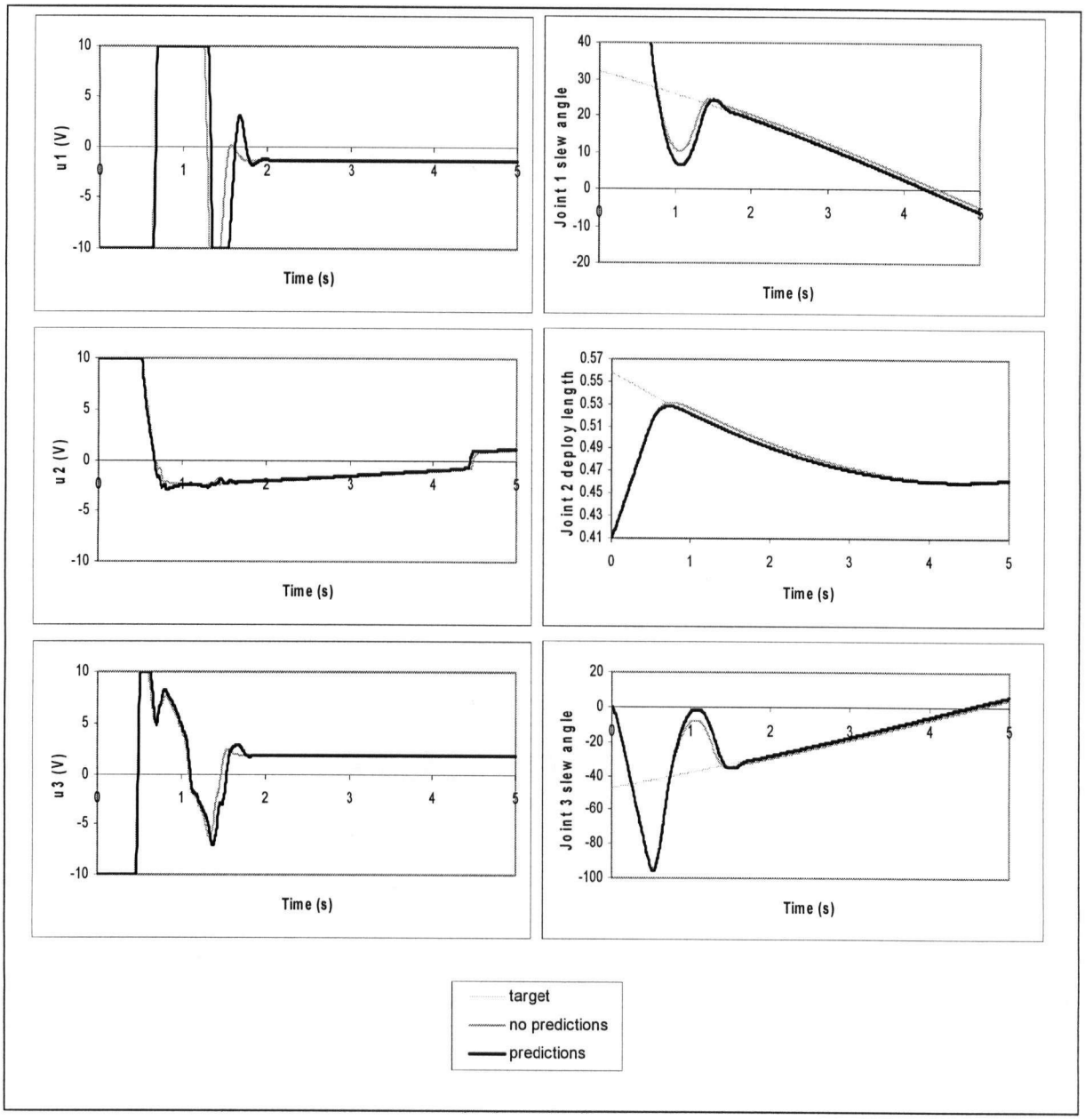


Figure 4.2 - Results from unconstrained controller showing the benefits of using predictions.

The results in Figure 4.2 show how the predictive controller can be tuned to produce accurate tracking results. However, in achieving these results, the manipulator joints overshoot their target positions. The only way this could be corrected in the unconstrained controller was by decreasing the weights on the position tracking errors relative to the weights on the velocity tracking errors.

Unfortunately, when less importance was placed on the joint position errors the tracking performance was found to suffer.

When the saturation of the control input signal was removed the degree of overshoot was significantly reduced; however, very large control inputs were needed. The decrease in overshoot was due to the fact that the controller was making more accurate predictions. Without the inputs being saturated the controller was able to apply the optimal solution that was calculated. So, when the controller is unaware of the abilities of the plant (i.e., its input and output constraints), the minimization of the plant error cannot be fully realized, and the performance degrades.

4.3 Constrained Predictive Control

When constraints are included in the predictive controller, the computations involved can become very time consuming and the online optimizer may not converge to the optimal solution within the control period. For this reason, most constrained optimal predictive controllers for high-bandwidth processes have been limited to simulations. In this thesis, since both simulation and experimentation are carried out, multi-parametric programming techniques are used to improve the computational speed of the optimization. However, the calculation of the explicit offline solution to the constrained predictive controller can be very time consuming, and the solution must be re-computed every time an adjustment is made to the tuning parameters. So, as a means of comparison, and to show the full benefits of constrained predictive control, the optimal solution is first computed by solving the optimal control problem at each time step using the 'quadprog' QP solver from the Optimization Toolbox in MATLAB.

4.3.1 QP Solution

The benefits of adding constraints into the predictive control problem can be seen from the results presented in figures 4.3 and 4.4. The tuning parameters were all kept the same, but the ±10V constraints on the inputs were formulated into the optimization. When the controller is aware of the input constraints it makes a much more cautious approach towards the target. The input constraints let the controller know how fast the MDMS can slow down when it gets to the target.

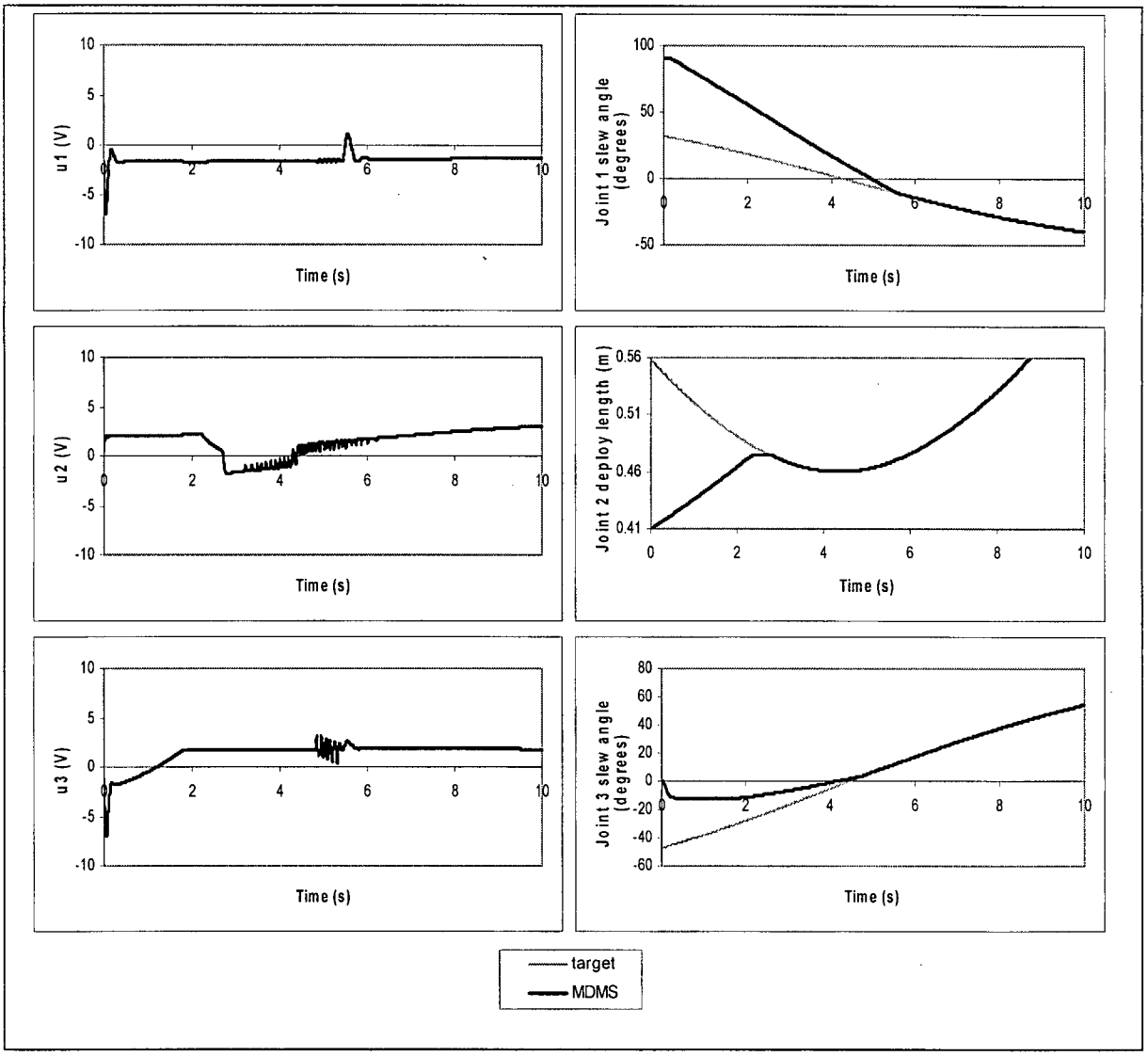


Figure 4.3 - Results of input constraint realization with unconstrained controller tuning parameters in joint space.

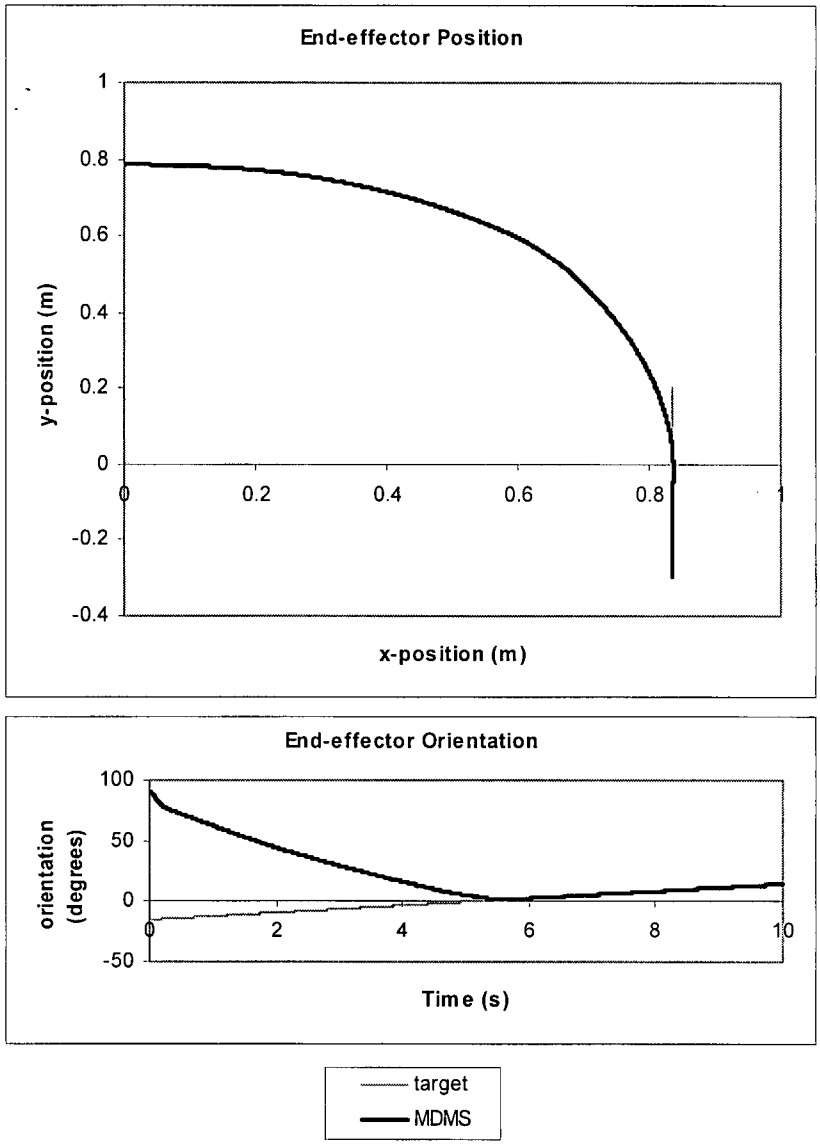


Figure 4.4 - Results of input constraint realization with unconstrained controller tuning parameters in task space.

When input constraints are used, the controller makes more knowledgeable decisions, and the tracking error weights can be adjusted to take advantage of this knowledge. In figures 4.5 and 4.6 the velocity tracking weights were reduced until the controller gave a critically damped response. The results show that with the inclusion of constraints the manipulator is able to achieve a steady state tracking error equal to that of the unconstrained controller in Figure 4.2, and in the same interval of time, but now without overshooting the target position.

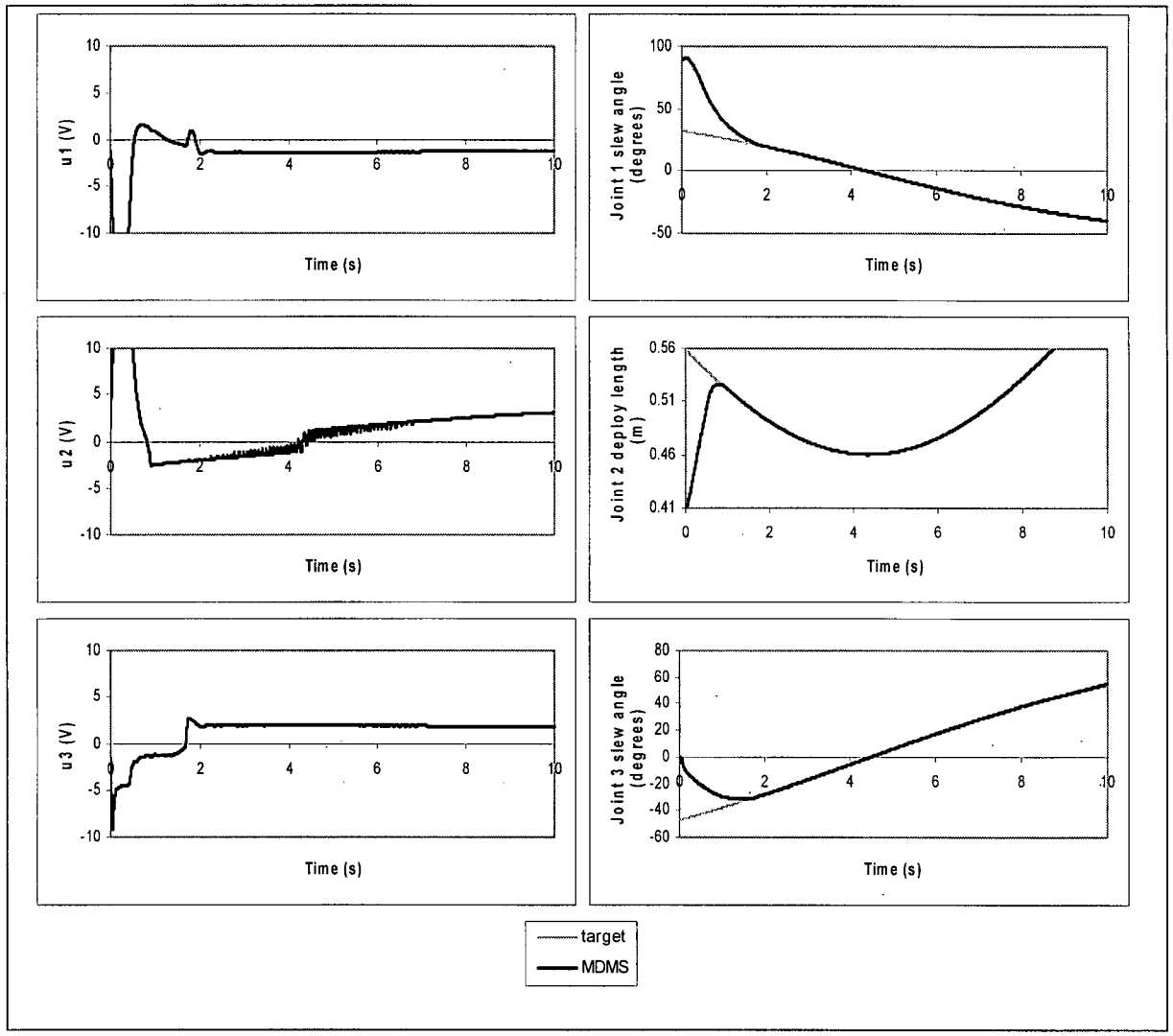


Figure 4.5 - Results from the re-tuned constrained controller with input constraints in joint space.

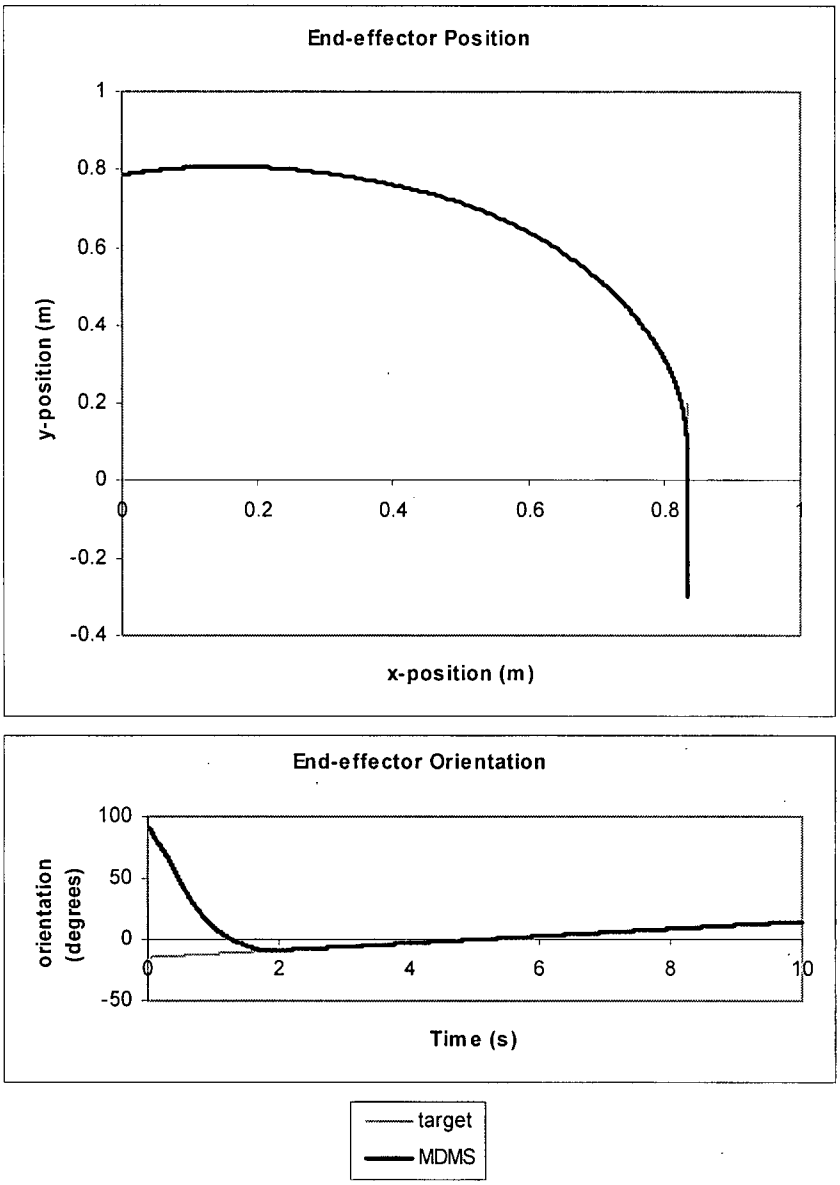


Figure 4.6 - Results from the re-tuned constrained controller with input constraints in task space.

The results shown in figures 4.5 and 4.6 have been obtained by tuning the controller to give the fastest response without any overshoot for the target traveling at velocity $v = [0 \ -0.05]^T$ m/s, and rotating at angular velocity $\omega = \pi/60$ rad/s. Not all targets will move and rotate at this speed, and as the speed of the target increases the accuracy of the model of the target will decrease. Furthermore, the internal model of the MDMS used for predictions is linear, and if it is linearized

using large gear ratios on the manipulator joints, the predictions from the linear model will become less accurate as the joint velocities become larger. However, even for faster moving satellites that were considered, the MDMS was able to achieve similar results to those shown in figures 4.5 and 4.6 without overshooting the target. Table 4.1 gives some typical results for several different target speeds.

Table 4.1 - Results obtained with different target speeds.

	Target details			Tracking accuracy	
Case No.	v_x (m/s)	v_y (m/s)	ω (rad/s)	Position	Orientation
1	0	-0.05	$\pi/60$	< 1mm	< 0.5°
2	0.03	-0.15	$\pi/16$	< 1mm	< 0.5°
3	0.05	-0.15	$\pi/8$	< 1mm	< 0.5°
4	0	-0.05	0	< 1mm	< 0.5°
5	0.005	-0.05	$\pi/12$	< 1mm	< 0.5°

In Table 4.1, constraints on the inputs are set at the maximum signal value that the servo card can send to the amplifiers of the robot drive system (±10V). As a result, when the input constraints become active, very large inputs are generated, which in turn lead to large joint velocities and accelerations on the approach towards the target. If the MDMS is capturing a satellite in space, these joint accelerations and velocities could be unacceptable because the space manipulator would be long and flexible, and the base of the manipulator would be attached to a space platform. The inputs from the controller can be further constrained to achieve the desired performance from the manipulator. When the input constraints are reduced to ±3V on the slewing joints and ±5V on the deploying joint, the MDMS is still able to catch every target listed in Table 4.1 without any overshoot and with the same tracking accuracy.

Alternatively, joint velocity constraints can be defined to improve the performance of the controller and restrict the motion of the manipulator to desirable operating limits. In figures 4.7 and 4.8, the target in Case 1 of Table 4.1 is caught using velocity constraints on the joints. The slewing velocity is constrained to less than 0.785 rad/s and the deployment velocity to less than 0.1 m/s. The maximum joint velocities produced in figures 4.7 and 4.8 are equal to the user defined constraints, therefore demonstrating the ability of the predictive controller to realize user defined output constraints.

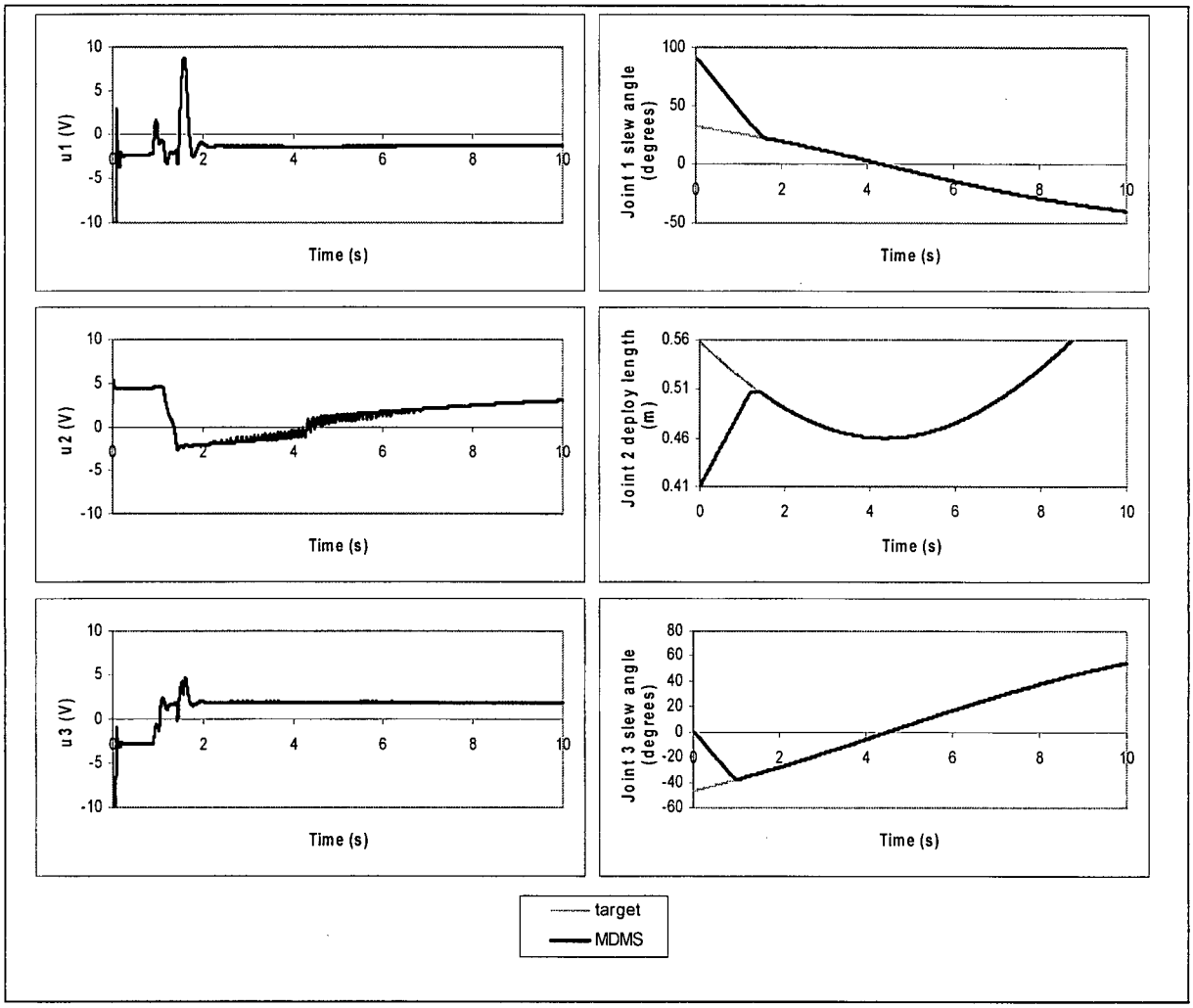


Figure 4.7 - Results from the predictive controller under MDMS output constraints in joint space.

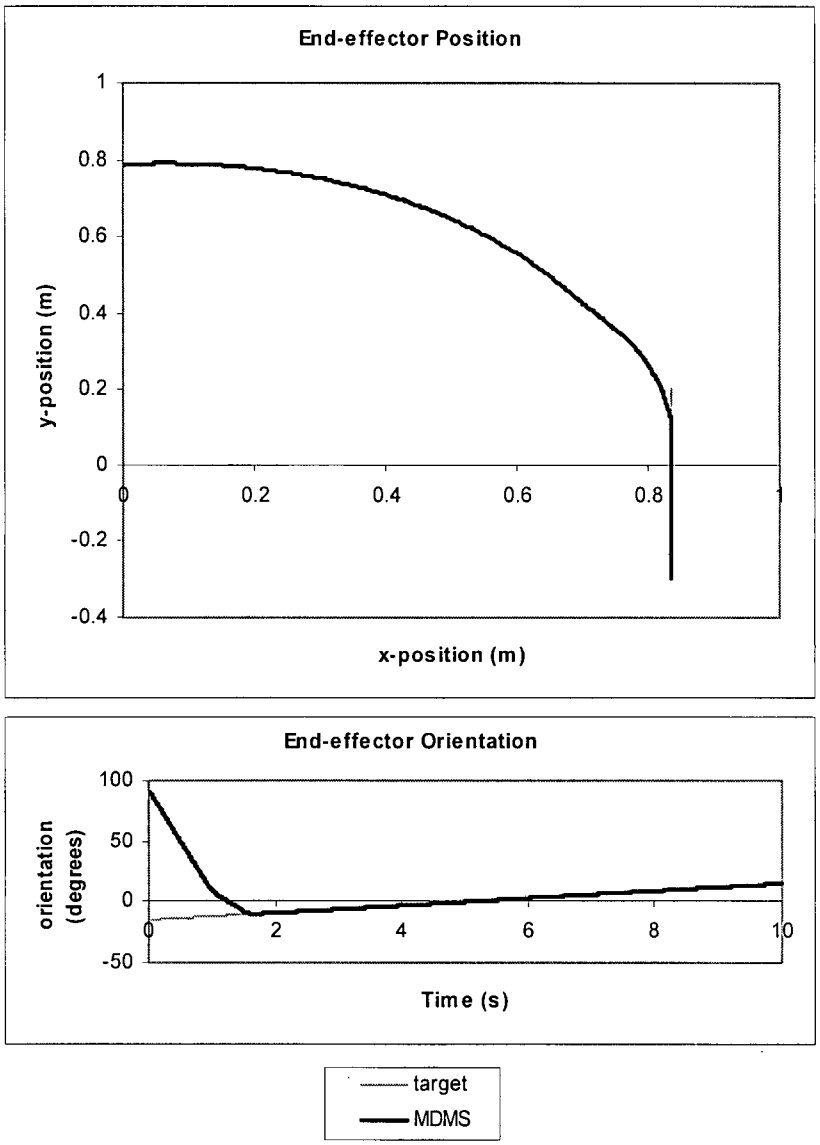


Figure 4.8 - Results from the predictive controller under MDMS output constraints in task space.

4.3.2 Mp-QP Solution

All things being equal, the mp-QP optimization should produce the exact same results as the online computation. Even though this could be true, the number of optimization parameters, N_p, in the MDMS satellite capturing problem can lead to incredibly large look-up tables. To find a solution in these look-up tables a search algorithm must determine to which region a point $\xi \in \Re^{N_p}$ belongs, and then apply the corresponding affine control law for that region. Efficient

methods of searching through these polytopes to find the region containing ξ have been developed [Borelli, 2002; Tondel, 2003], but these methods have not been incorporated into the multi-parametric toolbox.

In addition to the problem of large look-up tables, it was found that the mp-QP solver in the multi-parametric toolbox in MATLAB was much more sensitive to ill-conditioned problems than the QP solver that was used at each time step in the previous simulations. The numerical difficulties arose when the constraints were defined at every future time in the prediction horizon. Enforcing the constraints only at select points throughout the prediction horizon often mitigated the numerical problems, and it reduced the size of the look-up table; however, this also made the controller sub-optimal. The sub-optimal mp-QP controller was found to be better than the unconstrained controller because it had some knowledge of the input constraints, and it was still able to realize user defined output constraints.

To demonstrate the ability of the mp-QP controller, constraints were placed on the inputs only at the beginning and the end of the horizon. The result was a look-up table with $N_r = 384$ regions and took just over five minutes to compute on a Pentium4 2.4GHz processor in MATLAB 6.1. This set of constraints was chosen because when constraints were placed on the inputs over the entire horizon, the mp-QP solver ran into numerical problems. With this set of constraints defined the controller predicted that it was able to move slightly beyond the constraint in the middle of the horizon, but because the optimization was performed at each time step, and the maximum input constraints were satisfied at the beginning of the horizon, no constraint violations occurred.

In figures 4.9 and 4.10 the satellite in Case 1 of Table 4.1 is captured. The results show that despite the sub-optimal predictions, the controller is still able to intercept the target with only the first joint overshooting by just under 2 degrees. This level of overshoot is still considerably less than that of the unconstrained solution. Furthermore, restricting the inputs to smaller values can eliminate the overshoot. The results in figures 4.5 and 4.6 are superimposed in figures 4.9 and 4.10 to compare the performance of the fully constrained and sub-optimally constrained controllers. The results from the mp-QP controller are more aggressive because the controller is not fully aware of the constraints that will be met when the manipulator has to decelerate upon arrival at the target, and this is the cause of the overshoot in the first joint.

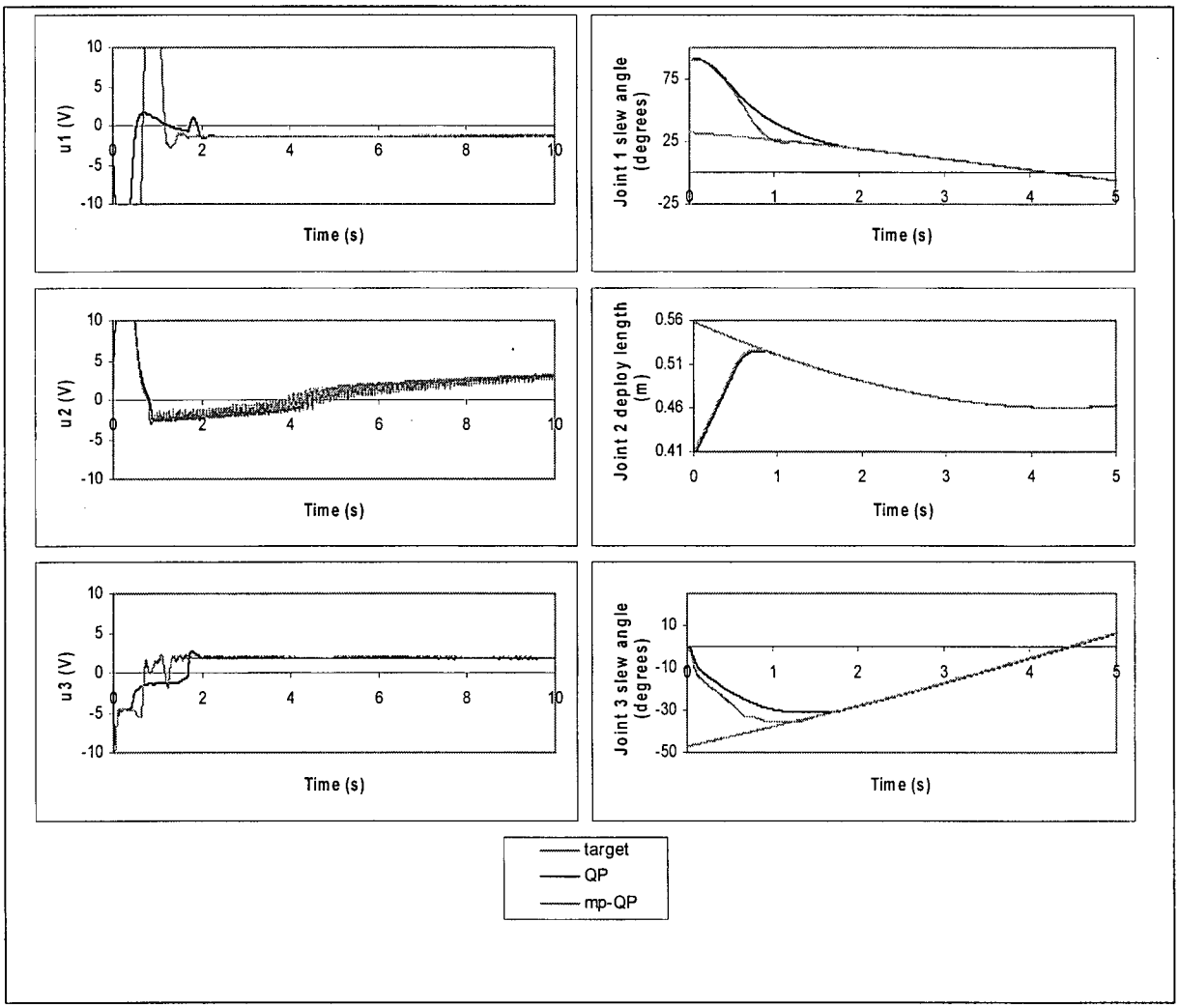


Figure 4.9 - Results from the mp-QP simulation in joint space.

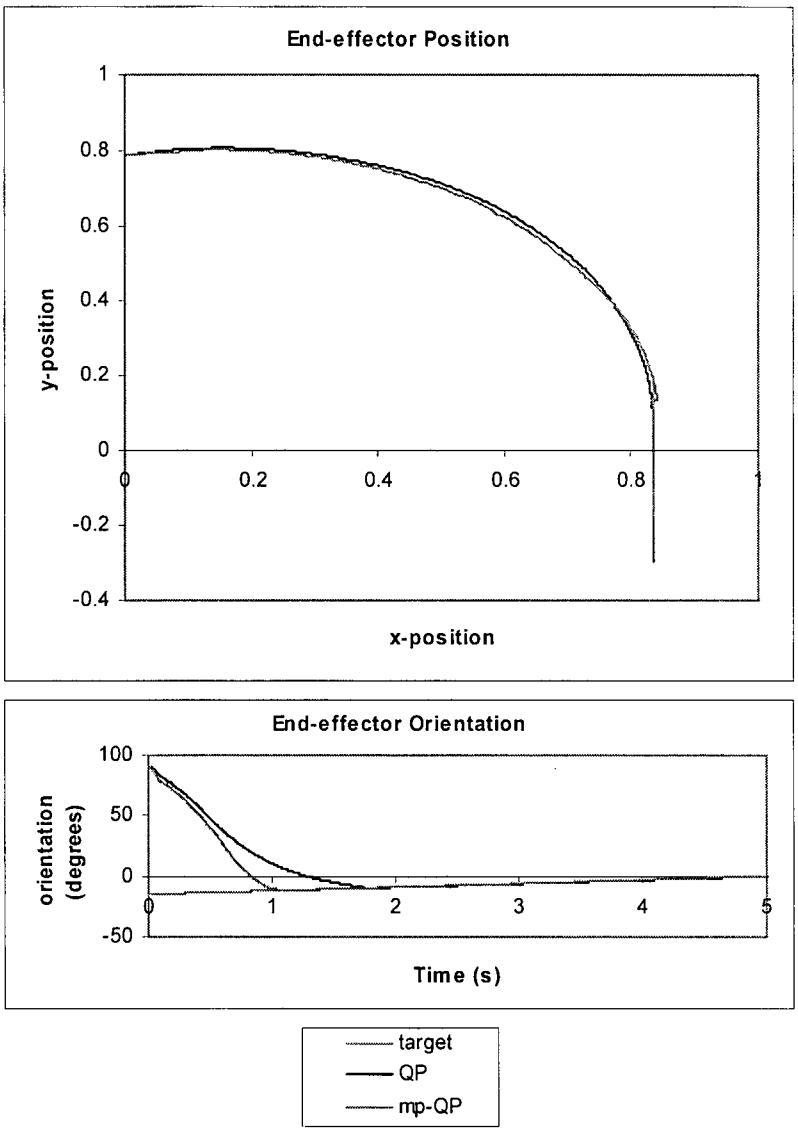


Figure 4.10 - Results from the mp-QP simulation in task space.

During the tracking of the target none of the constraints are active and the mp-QP solution produces the same results as the QP and unconstrained solutions. The tracking error is reduced to less than 1 mm and less than 0.5 degrees. So, other than the slightly more aggressive approach towards the target, the sub-optimally constrained mp-QP solution and the fully constrained QP solution produce very similar results.

4.4 Summary

The simulation results presented in this chapter have shown that the predictive controller is capable of successfully performing a robotic task of approaching and tracking a spinning satellite for the purpose of capturing it. The fact that the controller was able to make the robot track a variety of satellite targets with the same level of accuracy demonstrates the robustness of the predictive controller against the unmeasured (not modeled) disturbances due to the nonlinearities in the MDMS. The results showed the ability of the second order approximation of the target to provide predictions that helped to improve the performance of the controller.

Constraints were added to the predictive controller to demonstrate the benefits of making the controller aware of the physical limitation of the MDMS. The constraints were then used to show how the smoothness of the approach towards the target could be improved. Finally, output constraints could be placed on the MDMS joints to maintain the robot within the possible operation limits of its space-based system.

Multi-parametric quadratic programming was presented in this chapter to show that despite defining the constraints only at a select few points throughout the prediction horizon, the mp-QP controller was able to realize user defined constraints and overall increase the performance of the predictive controller during the satellite capture.

The model used in the simulations is not completely accurate and as a result the mp-QP controller developed in this chapter may not produce identical results when applied to the prototype manipulator in the experimental investigation. Experimental results from the unconstrained and constrained predictive controllers will be presented and discussed in the next chapter.

CHAPTER 5: EXPERIMENTAL IMPLEMENTATION

5.1 Preamble

Before the development of explicit solutions for model predictive control (MPC), the application of constrained predictive control to high bandwidth servomechanisms was found to be quite limited. Even with the current speed of computers the computational cost of performing an online optimization at high frequencies could severely limit the size of the problem to be solved. Of course, the explicit MPC solution does have its limitations. Offline solutions to the constrained optimal control problem require very large look-up tables to be stored and searched through in order to find the optimal solution at each time step, and this will also put limitations on the size of the problem, but not to the same degree. With these considerations in mind, a physical implementation of the constrained MPC was made on the prototype multi-module deployable manipulator system (MDMS) in our laboratory. In this chapter, this physical implementation is described and the experimental results are discussed. The results demonstrate the effectiveness of using a predictive controller in the satellite-capturing problem using a deployable manipulator.

5.2 Experimental Setup

The MDMS, as shown in Figure 5.1, was designed for the purpose of testing various control schemes on a physical system, with a focus on space-based applications. It is the second generation of prototype deployable manipulators that has been developed in our laboratory at the University of British Columbia. The new design is comprised of lighter links than its predecessor. Harmonic gear drives are used to help reduce the effects of backlash and the gear ratio has been increased [Wong, 2000]. All of these changes have, to some degree, mechanically linearized the prototype manipulator. In view of this, the linear predictive controller has been applied directly to the MDMS, without using the inverse dynamics of the robot. Real-time experiments were performed using this prototype manipulator system.

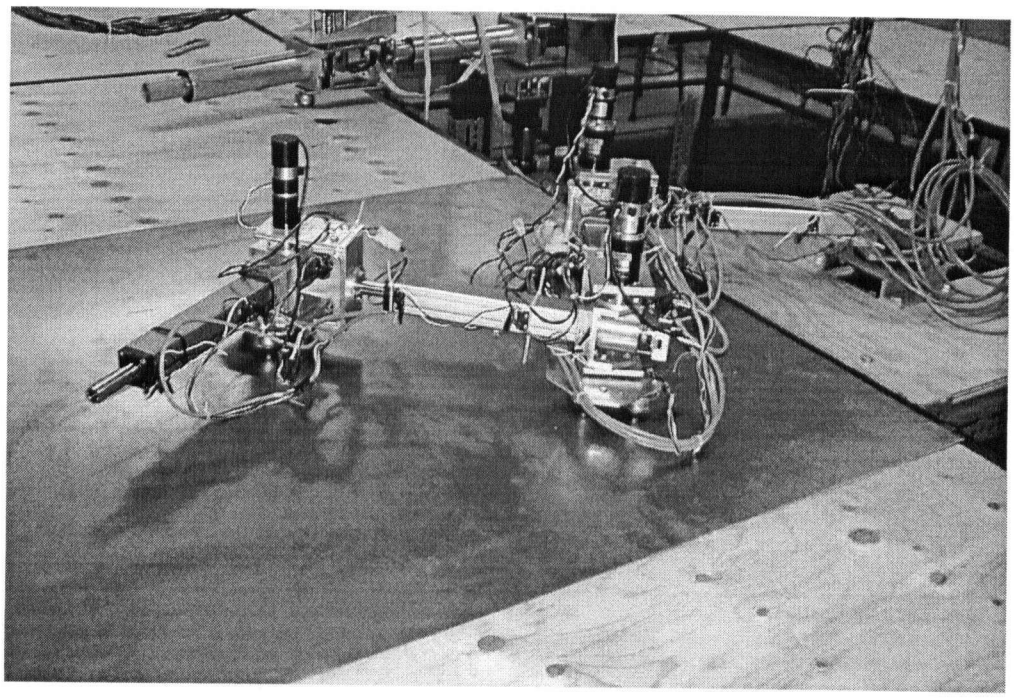


Figure 5.1 - MDMS used in the experimental investigation.

Feedback from the joints is obtained through incremental optical encoders attached to the motor shafts and is fed to a servo card installed on a desktop computer running Windows 2000. The controller is programmed in C++, and VenturCom's Real-Time Extension (RTX) provides an application programming interface (API) to add real-time capabilities to Windows.

The use of RTX is a recent addition to the prototype manipulator [de Silva, McCourt & Ohmiya, 2003]; previously QNX real-time operating system was used. One of the main advantages to RTX for Windows is the ease of use. This was certainly true for the predictive controller developed in this thesis. The offline multi-parametric quadratic programming (mp-QP) solution is computed in MATLAB using the Multi-Parametric Toolbox [Kvasnica, M., et al., 2004]. With RTX, the controller can be run on the same computer immediately after the mp-QP solution has been computed in MATLAB. This significantly decreases the time it takes to tune the controller, and therefore makes the development process much easier.

The MDMS used in the experiments does not have an end-effector capable of capturing targets. Consequently, for the experiments carried out as presented in this chapter a simulated target has been used. The controller is used to minimize the position and orientation error between the end-

effector and the target, and continue to track the target as long as it is in the operating range of the MDMS. The coordinate system for the experiments has been kept the same as shown in Figure 4.1, and the manipulator started the satellite capture from the initial configuration shown in Figure 5.2.

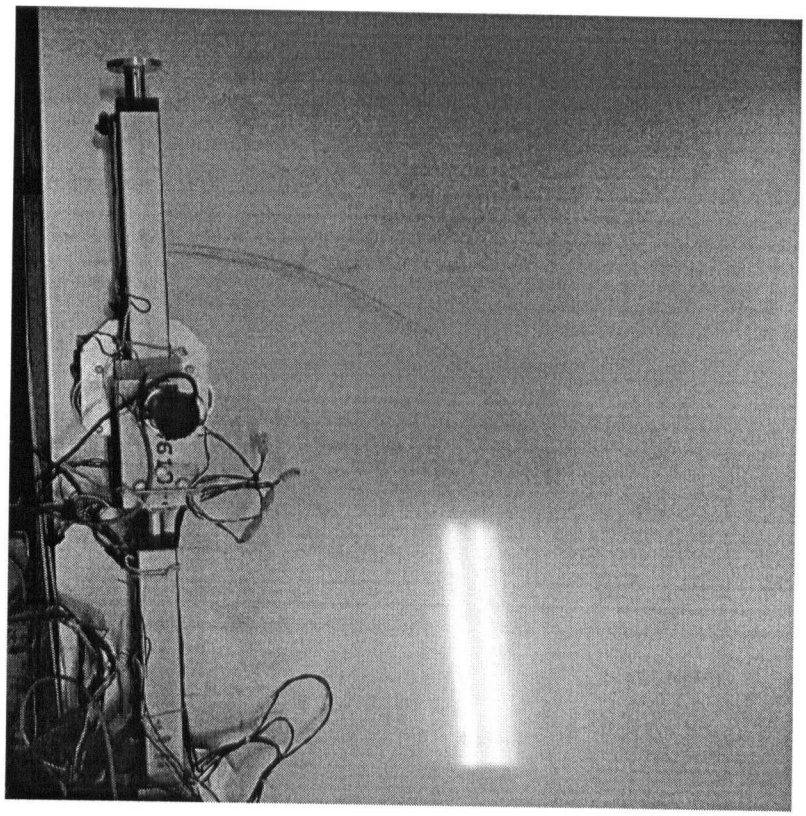


Figure 5.2 - Initial configuration of the manipulator for the experiments.

5.3 Results from the Unconstrained Controller

When the unconstrained predictive controller was applied to the prototype manipulator, the experimental results were found to be slightly more damped than the corresponding simulation results in Chapter 4, and the end-effector did not track as closely to the target as desired. In view of this, when running the subsequent tests on the manipulator, as reported in this chapter, the weighting matrices were adjusted to give the desired performance. The results from the simulated capture of a satellite traveling at $v = [0.03 \quad -0.15]^T$ m/s and rotating at $\pi/8$ rad/s (Case 3 of Table 4.1), are shown in figures 5.3 and 5.4.

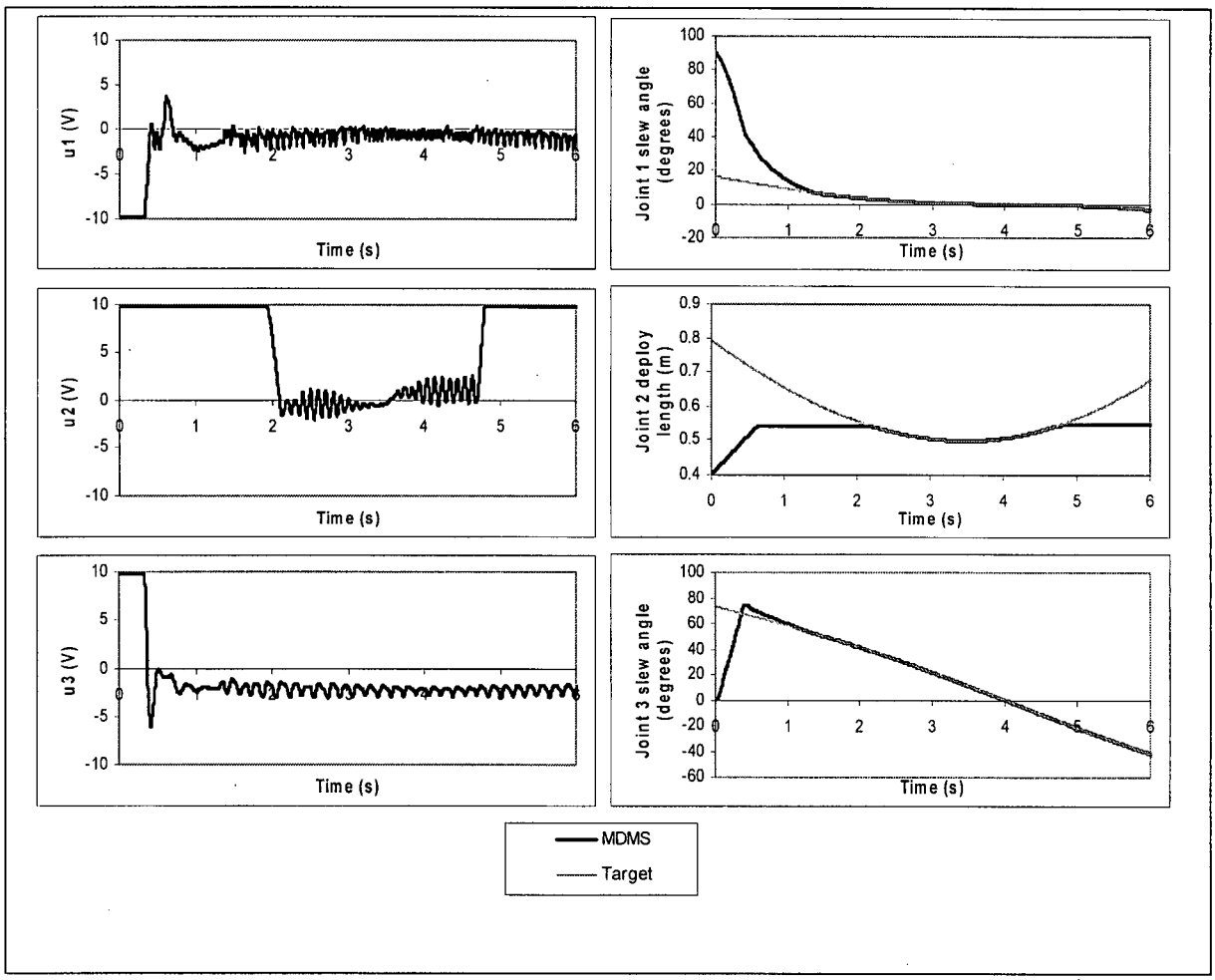


Figure 5.3 - Results from the unconstrained PFC in joint space.

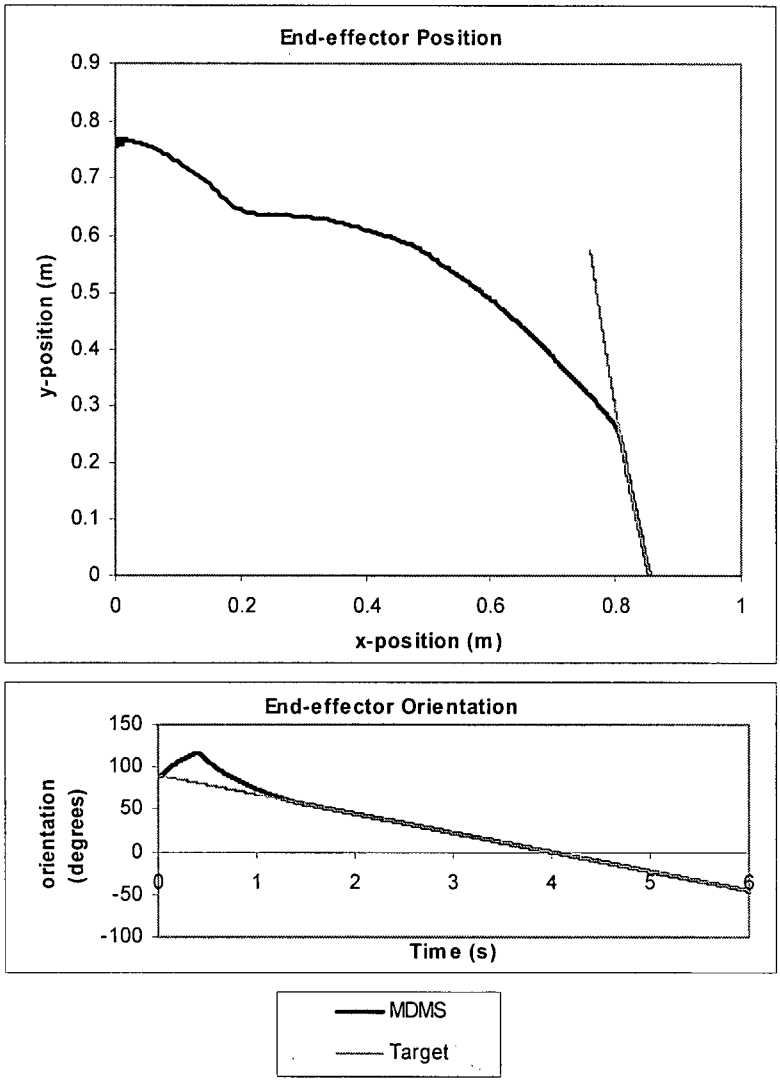


Figure 5.4 - Results from the unconstrained PFC in task space.

The results in figures 5.3 and 5.4 show that as the joints of the MDMS approach the target, only the third joint overshoots its desired value. However, the end-effector itself did not overshoot the satellites position during the approach. The results from target satellites at different velocities are summarized in Table 5.1.

Table 5.1 - Unconstrained controller results for various conditions of target satellite.

	Target details			Tracking accuracy	
Case No.	v_x (m/s)	v_y (m/s)	ω (rad/s)	Position	Orientation
1	0	-0.05	$\pi/24$	1.1±0.8mm	-0.09°±0.13°
2	0.03	-0.15	$\pi/8$	1.0±0.7mm	-0.12°±0.11°
3	0	-0.1	0	1.5±1.0mm	-0.17°±0.12°
4	0.005	-0.05	$\pi/12$	0.9±0.5mm	0.10°±0.11°

As seen, the unconstrained controller was able to minimize the error between the end-effector and the target with no overshoot. However, during the approach towards the target, the inputs generated by the controller were saturated at the maximum signal value of the servo card, which resulted in large joint velocities and accelerations. If this controller were implemented on a space-based system, then constraints should be placed on the maximum joint velocities and accelerations to ensure that the robot would operate within safe limits and furthermore, undesirable vibrations could be avoided.

5.4 Results from the Constrained Controller

Benefits of the constrained controller have been demonstrated in Chapter 4. Similar benefits are realized in the experimental operation, while demonstrating the ability of the prototype MDMS to realize user defined constraints, as presented now. The mp-QP solver, which is found in the multi-parametric toolbox in MATLAB, was used to determine the constrained solution to the predictive control problem. Unfortunately, the multi-parametric toolbox used in this thesis is known to be numerically sensitive to a large degree [Kvasnica, et al., 2004]. When a large number of constraints are defined for the problem, the regions in the space partitioned by the mp-QP solver become smaller. Within these small regions the mp-QP solver may face difficulty in computing the corresponding optimal control law (see Appendix A for details on mp-QP). Furthermore, when a large number of constraints are defined the number of regions created by the mp-QP solver becomes very large, and as a result it can be difficult to find the optimal solution in the look-up table that is produced. To limit the size of the problem, and overcome the numerical problems, in the present implementation the constraints are only defined at a select few points along the prediction horizon. By doing this, the computed solution does become sub-optimal.

However, depending on where these constraints are defined, the controller is still able to obey user defined constraint.

5.4.1 Searching for the Optimal Control Law

When the mp-QP algorithm computes the offline solution, it generates N_r different regions. Each of these regions are defined by the convex polytope

$$\mathbf{H}_i \cdot \xi(k) \leq \boldsymbol{K}_i \quad \text{for } i = 1,2,3, \ldots, N_r \tag{5.1}$$

where $\xi(k) = [x(k)^T, u(k-1)^T, x_r(k)^T]^T$ is a vector in the parameter space $\Re^{N_P}$. Each row of the relation (5.1) describes a facet of the N_p-dimensional polytope for the corresponding region. Within each of these N_r different regions the set of active constraints for the optimization problem remains constant; therefore, an affine control law can define the control solution, in the following form.

$$\mu(k) = \mathbf{F}_i \cdot \xi(k) + \boldsymbol{G}_i \qquad \text{for } i \in \{i \mid \mathbf{H}_i \cdot \xi(k) \leq \boldsymbol{K}_i\} \tag{5.2}$$

So, in order to compute the control solution, the controller must first determine the region in which $\xi(k)$ is located. The simplest way to do this is through a sequential search of all the regions: if the i^{th} region is defined by N_{Ci} facets (i.e., N_{Ci} rows in H_i and K_i), then to determine if a point is within a polytope, the controller must make $N_{Ci} \cdot N_p$ multiplications, $N_{Ci} \cdot (N_p - 1)$ additions and N_{Ci} comparisons, for a total of $2 \cdot N_p \cdot N_{Ci}$ arithmetic operations. In the worst case, the controller will have to search through every region for a total of $2 \cdot N_p \cdot N_C$ arithmetic operations, where

$$N_C = \sum_{i=1}^{N_r} N_{Ci} \tag{5.3}$$

Borrelli, et al. [2001] developed a method for searching through these polytope regions that chooses a "steepest descent" type search direction. The algorithm decreased the worst-case search time to $2 \cdot N_p \cdot N_r + N_C$ arithmetic operations. Tondel, et al. [2003b] improved on Borrelli's

method by formulating the polytopes in a binary search tree, which produce O(log(N_r)) search times. These search algorithms can allow for a problem of much larger size to be solved by the offline mp-QP scheme. However, these algorithms have not yet been incorporated into the multi-parametric toolbox and, as mentioned earlier, the number of constraints has been reduced because of numerical issues in the mp-QP solver in MATLAB. Consequently, in the examples that are demonstrated in this thesis the number of regions found in the mp-QP solution are fairly small, and in most cases a sequential search could still be performed within the sampling time of the controller. For simplicity then, only a sequential search algorithm is used.

5.4.2 Dealing with Infeasibilities

When standard predictive controllers are used with online minimization of the cost function, a method must be available for dealing with situations when the constrained optimization problem is infeasible. In mp-QP, the most optimal way of dealing with a set of infeasible optimization parameters is to choose the polytope region that is the shortest distance away from the infeasible point. The simplest way to find the closest region is via another sequential search through the N_r regions and by determining how far away the infeasible point is from each region. However, doing this would significantly increase the maximum search time needed to find a sub-optimal solution.

Instead, when the region containing $\xi(k)$ could not be found, the controller would revert to the unconstrained solution, but with much more restricting saturation values. This method was chosen because it ensured that the manipulator continued to move towards the target, however, it moved slower towards the target until a feasible parameter value could be found.

5.4.3 Experimental Results

In the tests with the unconstrained predictive controller, as given in Figure 5.3, the third joint generated the highest velocity of approximately 3.5 rad/s, and this joint was the only joint to overshoot its target value. Consequently, the first output constraint was defined so that the joint velocity of the third joint remained below 0.75 rad/s. This value was chosen so that the joint would still be able to track its target without violating the constraint.

The next constraint was placed on the joint velocity of the prismatic joint (joint #2). In the unconstrained controller the inputs to this joint were saturated during its approach toward the target, which resulted in a maximum joint velocity of approximately 0.225 m/s. A constraint was placed for the joint velocity to remain below 0.1 m/s, which was, again, chosen so that it would not limit the ability of the joint to track its target.

The joint constraints were defined only at the beginning and the end of the prediction horizon, which resulted in a controller consisting of $N_r = 92$ regions. A controller sampling frequency of 200 Hz was used, and the sequential search was able to search through these 92 regions in an average of 0.5ms on the Pentium4 2.4 GHz PC used in the experiments.

The results from the constrained predictive controller are given in figures 5.5 and 5.6. These results show the constrained joint velocities in the second and third joints, but more importantly they show an improvement in the way the manipulator approaches the target and the inputs generated by the controller. When the velocity of the third joint is constrained to 0.75 rad/s the level of overshoot is significantly reduced. Moreover, the total control effort used during the capture of the satellite is significantly reduced. In particular, in the unconstrained controller, the control inputs saturated whenever the second and the third joints were not at their target values. In the constrained controller the inputs only became saturated when the joint limits were reached. If the joint limits were formulated into the control problem then the controller would be aware of the limit that it has reached, and would not command the joint to extend any further.

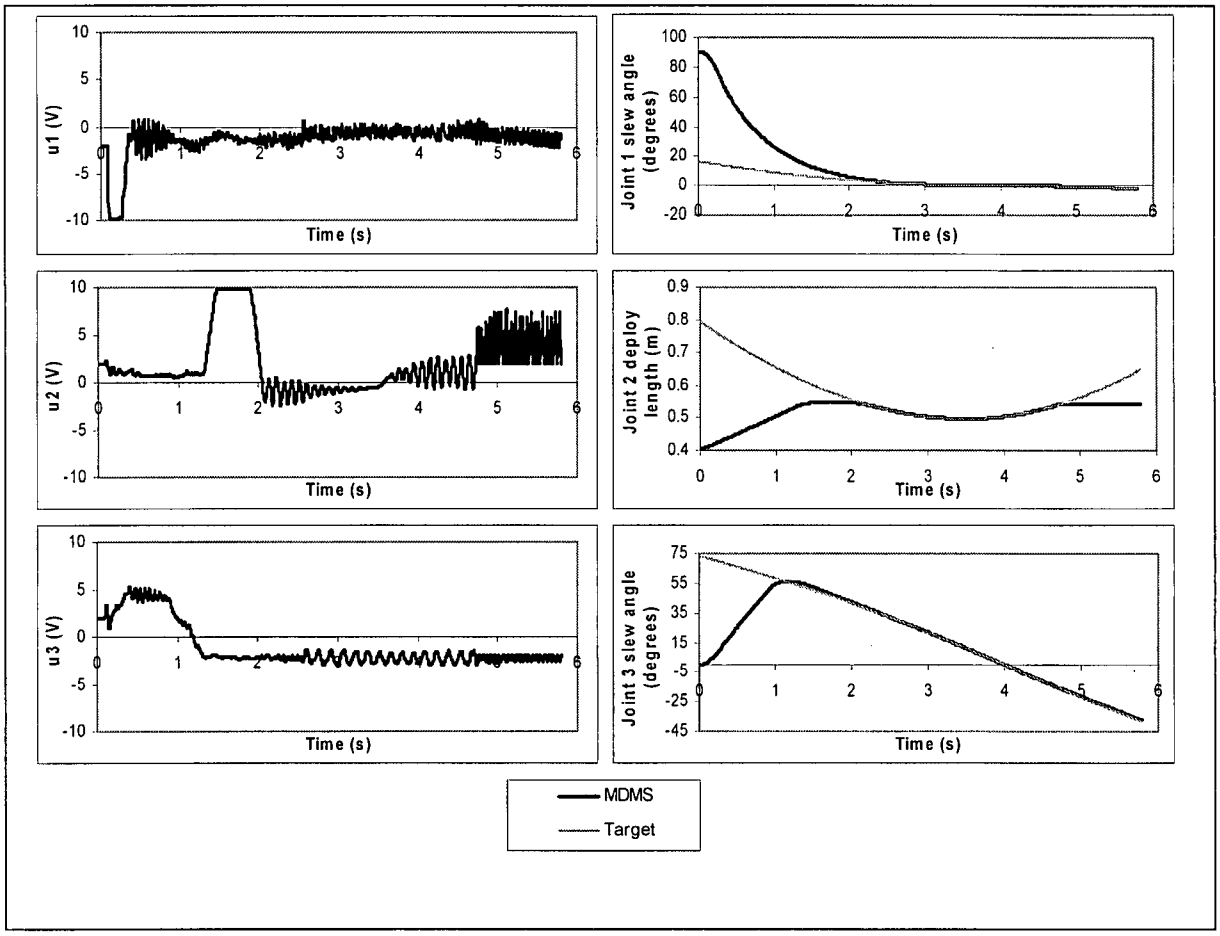


Figure 5.5 - Constrained predictive controller results obtained using the prototype MDMS in joint space.

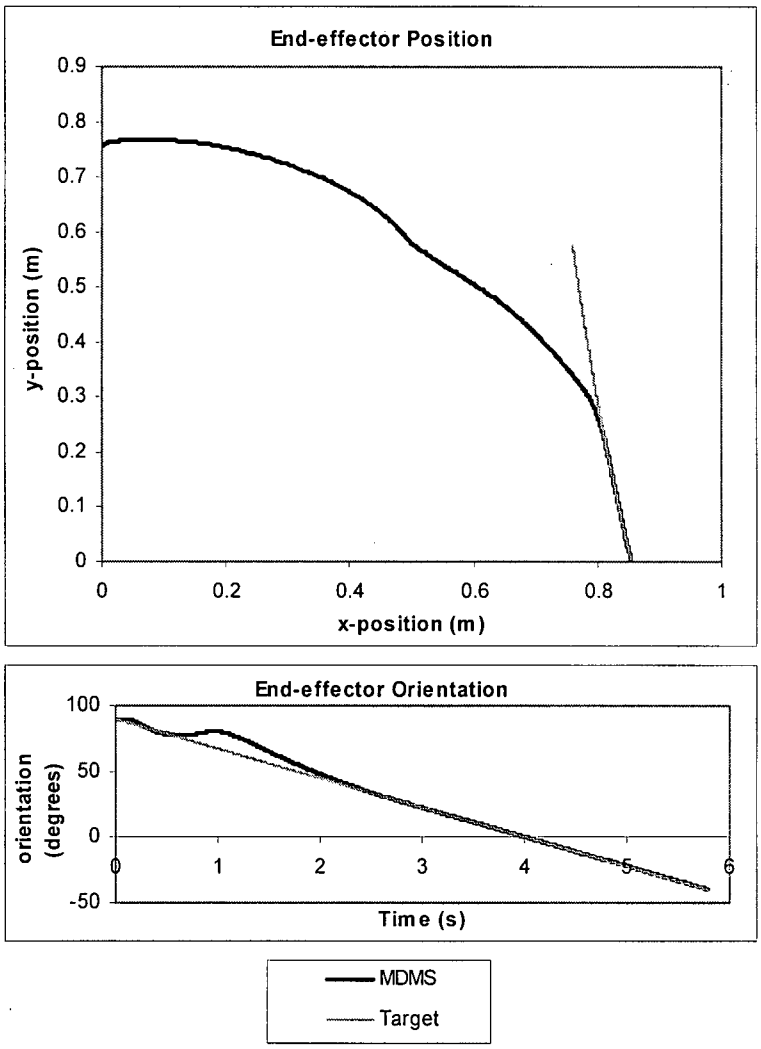


Figure 5.6 - Constrained predictive controller results obtained using the prototype MDMS in task space.

The results from the capture of other target satellites are very similar to those summarized in Table 5.1, with the exception that the joint velocities of the second and third joints have been constrained. Results from various other satellite targets are given in Appendix C.

5.5 Summary

In the previous chapter the benefits of predictive control were demonstrated through computer simulations. In the present chapter, the controller was implemented on the prototype MDMS and

experiments were carried out to study its performance on a real system. The experimental results were found to be slightly different from the corresponding simulations. Nevertheless, in the experiments the prototype MDMS was able to approach the target with very large joint velocities without overshooting the desired values. It should be emphasized that these joint velocities could be undesirable in space-based applications, and a solution to this problem was found with the constrained predictive control.

Constraints were added to the predictive control system via offline explicit solutions, resulting in a constrained predictive control problem. The offline solutions were compiled into a look-up table using a multi-parametric quadratic programming solver in the multi-parametric toolbox of MATLAB. The solutions were then read by the controller, which was programmed in C++ with the help of VenturCom's RTX for Windows.

The addition of the joint velocity constraints not only produced the required improvement in the joint motion as the MDMS approached the target, but also resulted in a significant decrease in the control effort that was needed. The constraints were defined only at the beginning and the end of the prediction horizon. Still, the controller was still able to restrict the joint velocities to the user-defined values.

CHAPTER 6: CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

6.1 Conclusions

This thesis investigated the use of model-based predictive control for the capture of a multi-degree-of freedom object that moves in a somewhat arbitrary manner, using a deployable manipulator. While the study was conducted through both computer simulation and ground-based experimental investigation, the intended application was focused on automating the robotic capture of a free-floating and spinning satellite. The main motivation for this application came from the fact that robotic satellite capture in space, for retrieval, correction, repair, etc., can significantly eliminate the risks involved when astronauts are used in space walk scenarios to manually execute the necessary tasks. Maneuvering a robot in the successful capture of a spinning satellite can be a difficult task. When a satellite is spinning, the maneuvering of a robotic manipulator by a human operator for a successful capture becomes increasing difficult. The difficulty of the task increases when the robot is tele-operated from a ground control station, because of communication delays. For this reason, satellite capturing using an autonomous robot is particularly attractive, and was studied in the thesis.

Predictive control was shown to be a feasible method for autonomously guiding a prototype manipulator towards a target satellite. When predictions are made the controller anticipates the future movements of both the target and the robot. Optimal control moves can then be calculated to minimize these predicted position and velocity errors between the target and the manipulator.

A predictive controller was developed and evaluated first using computer simulation. Next it was implemented in the joint space of a prototype manipulator called multi-module deployable manipulator system (MDMS robot), which has been developed in our laboratory. A linear model of the robot was used for predictions. The MDMS has many attractive features. Particularly important in the present investigation is the fact that it uses harmonic drives with large gear ratios on the joint motors to reduce the nonlinear effects. Any remaining nonlinear effects were left as unmeasured disturbances for the controller. Predictions of the target were also made in the joint

space of the manipulator and a second order Taylor series approximation of the target was used in these predictions. Despite the approximations in the internal models, the simulation and experimental results from the predictive control of the MDMS showed that the tracking of a moving target was considerably improved when predictions were incorporated.

In addition to the improvements in the tracking performance, predictive controller was capable of directly incorporating constraints into the optimal control problem. When constraints were explicitly incorporated, the controller was made to be aware of its limitations, thereby guarding against unrealistic control actions. In the satellite capturing problem this can, for example, indicate to the controller how fast it would be able to decelerate when reaching the target. Consequently, when the controller was predicting into the future, it would be aware of how early it had to decelerate in order to reach the target with the right position and velocity, and this knowledge could be used to improve the way the MDMS approached the target.

In order to realize these constraints in real-time, a multi-parametric quadratic programming (mp-QP) technique was used. With mp-QP the constrained predictive control problem was solved explicitly offline and a look-up table was used online. Unfortunately, the mp-QP algorithm was found to be more susceptible to numerical problem than the standard QP solver. These numerical issues were resolved by defining the constraints only at select points throughout the prediction horizon, but this made the predictive controller sub-optimal. Fortunately the sub-optimal controller was still able to obey the user defined constraints and the approach towards the target was improved using the constrained predictive controller on the prototype MDMS.

6.2 Suggestions for Future Work

This research has demonstrated the advantages of using predictive control in the autonomous robotic capturing of a spinning satellite. Even though this research was the first to apply predictive control techniques to the MDMS, there are a number of improvements that could be made over this first attempt. Some suggestions for future work on this research area are indicated below.

(1) Simulations should be performed on a complete space-based model of the MDMS. In the space-based system the MDMS is situated on a mobile carriage that can move along an orbiting platform. The flexibility of the manipulator joints and links, the flexibility in the

platform, and the dynamic coupling between the manipulator and the orientation of the platform must be considered when maneuvering, for improving the accuracy of the system model. A separate controller may be employed to regulate the orientation of the platform; however, if a controller is aware of the coupling between the manipulator and the platform, then more accurate control decisions can be made. Furthermore, if the mass of manipulator is sufficiently large when compared to the platform, then the motion of the manipulator can cause the platform to move. This will change the position of the target satellite relative to the base of the robot, and this effect must be considered in the controller.

(2) In the actual application of the predictive satellite-capturing controller, imaging and associated image processing techniques would most likely be used to determine the position and orientation of the target satellite. In that case, the controller will have to compensate for the delays in receiving the reference signal from the cameras. An observer may be built in order to estimate the current position of the target based on the known delays in the image processing.

(3) In this thesis, the approach and the tracking problem of the target satellite were considered. Future work should be done to study the completion of the capture and the retrieval of the satellite. The dynamics of the spinning satellite will directly affect the MDMS and its space platform when the capture is completed. Furthermore, the mass properties of the satellite may not be known, and the controller will have to adapt to compensate for the extra mass of the satellite at the end-effector.

(4) A rather ad hoc method was used in the thesis in dealing with infeasibilities. A more optimal solution to the infeasibility problem would be desirable. In the multi-parametric toolbox, if the optimization parameters define a point that is not within any region, then the search algorithm finds the region that is closest to this point and applies the corresponding affine control law. This solution was not used because of the increase in search time to find this closest region. One possible solution to this problem would be to create new regions that fill in the holes left by the mp-QP solver to ensure that no matter what the measured parameters are, they will always correspond to a defined region in the parameter space.

(5) It may be advantageous to have constraints that can change with the problem; for example, the joint velocity constraints could be dependent on how close the joint is to its target value. In this case the current value of the joint velocity constraint will become a parameter in the mp-QP problem, and the implementation becomes quite straightforward.

(6) In general, a satellite may possess a fairly more complex geometry that what is presented in figures 3.3 and 4.1. Solar panels, antennae and various other structures may be protruding from the body of the satellite. A more sophisticated controller should take them into account. A future version of the satellite capturing controller for the MDMS may use multiple and redundant modules and incorporate obstacle avoidance techniques when approaching the satellite to further ensure that the manipulator would not collide with the satellite during the capture.

Bibliography

Ambrose, R.O., Aldridge, H., Askew, R.S., Burridge, R.R., Bluethmann, W., Diftler, M., Lovchik, C., Magruder, D., & Rehnmark, F., (2000) "Robonaut: NASA's Space Humanoid", *IEEE Intelligent Systems Journal*, Vol. 15, No. 4, pp. 57-63.

Bemporad, A., Morari, M., Dua, V., & Pistikopoulos, E.N., (2002) "The explicit linear quadratic regulator for constrained systems," *Automatica*, Vol. 38, No. 1, pp. 3-20.

Bluethmann, W., Ambrose, R., Diftler, M., Askew, S., Huber, E., Goza, M., Rehnmark, F., Lovchik, C., & Magruder, D., (2003) "Robonaut: A Robot Designed to Work with Humans in Space", *Autonomous Robots*, 14, pp. 179-197.

Borrelli, F., (2002) *Discrete Time Constrained Optimal Control*, Ph.D. Thesis, Swiss Federal Institute of Technology, Zurich, Switzerland.

Borrelli, F., Baotic, M., Bemporad, A., & Morari, M., (2001) "Efficient On-Line Computation of Constrained Optimal Control," *Proc. IEEE Conference on Decision and Control*, Orlando, Florida, USA, pp. 1187-1192.

Cao, Y., (1999) *Dynamics and Control of Orbiting Deployable Multimodule Manipulators*, M.A.Sc. Thesis, University of British Columbia, Vancouver, Canada

Croft, E.A., Fenton, R.G. & Benhabib, B., (1998) "An On-line Robot Planning Strategy for Target Interception," *Journal of Robotic Systems*, Vol. 15, No. 2, pp. 97-114.

de Silva, C.W., (1989) *Control Sensors and Actuators*, Prentice-Hall, Englewood Cliffs, NJ.

de Silva, C.W., McCourt, R., & Ohmiya, M., (2003) "Control of a Multi-Module Deployable Manipulator Using RTX," *Proc. IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, Victoria, Canada, pp. 864-867

Fletcher, R., (1987) *Practical Methods of Optimization*, John Wiley & Sons, Chichester, England.

Hedjar, R., Toumi, R., Boucher, P., & Dumur, D., (2002) "Feedback Nonlinear Predictive Control of Rigid Link Robot Manipulators," *Proc. American Control Conference,* Anchorage, AK, pp. 3594-3599.

Inaba, N., & Oda, M., (2000) "Autonomous Satellite Capture by a Space Robot," *Proc. International Conference on Robotics & Automation,* San Francisco, CA, pp. 1169-1174.

Jasiobedzki, P., Greenspan, M., & Roth, G., (2001) "Pose Determination and Tracking for Autonomous Satellite Capture," *Proc. International Symposium on Artificial Intelligence and Robotics & Automation in Space,* Canadian Space Agency, St-Hubert, Quebec, pp. 1-8

Kvasnica, M., Grieder, P., Baotic, M., & Morari, M., (2004) "Multi-Parametric Toolbox (MPT)," *Hybrid Systems: Computation and Control*, Volume 2993 of Lecture Notes in Computer Science, pp. 448-462.

Loyd, J.E., (1998) "Desingularization of Nonredundant Serial Manipulator Trajectories Using Puiseux Series," *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 4, pp. 590-600.

Maciejowski, J.M., (2002) *Predictive Control with Constraints,* Prentice Hall, Harlow, England.

Matsumoto, S., Ohkami, Y., Wakabayashi, Y., Oda, M., & Ueno, H., (2002) "Satellite Capturing Strategy using agile Orbital Servicing Vehicle, Hyper-OSV," *Proc. IEEE International Conference on Robotics & Automation*, Washington, DC, pp. 2309-2314.

McCourt, R., & de Silva, C.W., (2003) "Applications of Predictive Control for Autonomous Satellite Capturing Using a Deployable Manipulator System," *Proc. ASME International Mechanical Engineering Congress*, Washington, D.C.

Nagamatsu, H., Kubota, T., & Nakatani, I., (1996) "Capture Strategy for Retrieval of a Tumbling Satellite by a Space Robotic Manipulator," *Proc. IEEE International Conference on Robotics & Automation,* Minneapolis, Minnesota, pp. 70-75.

Nagamatsu, H., Kubota, T., & Nakatani, I., (1997) "Autonomous Retrieval of a Tumbling Satellite Trajectory," *Proc. IEEE International Conference on Robotics & Automation,* Albuquerque, New Mexico, pp. 3074-3079.

Nakamura, Y., (1991) *Advanced Robotics: Redundancy and Optimization*, Addison-Wesley, Reading, Massachusetts.

Oda, M., (2000) "Experiences and Lessons Learned from the ETS-VII Robot Satellite," *Proc. IEEE International Conference on Robotics & Automation*, San Francisco, CA, pp. 914-919.

Park, T.H., & Lee, B.H., (1992) "An Approach to Robot Motion Analysis and Planning for Conveyor Tracking," *IEEE Trans. Systems, Man and Cybernetics*, Vol. 22, No. 2, pp.378-384.

Poignet, P., & Gautier, M., (2000) "Nonlinear Model Predictive Control of a Robotic Manipulator," *Proc. IEEE Advanced Motion Control Conference*, Nagoya, Japan, pp. 401-406

Qin, S.J., & Badgwell, T.A., (1997) "An overview of industrial model predictive control technology," *Chemical Process Control-V*, Vol. 93, No. 316, pp. 232-256. AIChe Symposium Series – American Institute of Chemical Engineers.

Qin, S.J., & Badgwell, T.A., (1998) "An overview of nonlinear model predictive control applications," Presented at *Nonlinear MPC Workshop*, Ascona, Switzerland.

Richards, A., & How, J., (2003) "Performance Evaluation of Rendezvous Using Model Predictive Control," *Proc. AIAA Guidance, Navigation, and Control Conference and Exhibit*, Austin, Texas, pp. 1-9.

Richalet, J. Rault, A., Testud, J., & Papon, J., (1978) "Model Predictive Heuristic Control: Applications to Industrial Process," *Automatica*, Vol. 14.

Richalet, J., Abu El Ata-Doss, S., Arber, C., Kuntze, H.B., Jacubasch, A., & Schill, W., (1987) "Predictive Functional Control: Applications to Fast and Accurate Robots," Proc. of IFAC 10th World Congress, Munich, Germany, pp. 251-258.

Sciavicco, L., & Siciliano, B., (2000) *Modelling and Control of Robot Manipulators*, Springer-Verlag, London, England.

Schreiber, G., Otter, M., Hirzinge, G., (1999) "Solving the Singularity Problem for Non-Redundant Manipulators by Constraint Optimization," *Proc. IEEE International Conference on Intelligent Robotics and Systems*, Korea, pp. 1482-1488.

Tafazoli, S., (1996) *Identification of Friction Effects and Structural Dynamics for Improved Control of Hydraulic Manipulators*, Ph.D. Thesis, University of British Columbia, Vancouver, Canada.

Tondel, P., (2003) *Constrained Optimal Control via Multiparametric Quadratic Programming*, Ph.D. Thesis, Norwegian University of Science and Technology, Trondheim, Norway.

Tondel, P., Johansen, T.A., & Bemporad, A., (2003a) "An algorithm for multi-parametric quadratic programming and explicit MPC solutions," *Automatica*, Vol. 39, No. 3, pp. 489-497

Tondel, P., Johansen, T.A., & Bemporad, A., (2003b) "Computation of piecewise affine control via binary search tree," *Automatica*, Vol. 39, No. 5, pp. 945-950

Torres, S., Mendez, J.A., Acosta, L., Sigut, M., Marichal, G.N., & Moreno, L., (2001) "A Predictive Control Algorithm with Interpolation for a Robot Manipulator with Constraints," *Proc. IEEE International Conference on Control Applications,* Mexico City, Mexico, pp. 536-541.

Ueno, S., & Modi, V.J., (1998) "Optimal Controller for a Space Platform Based Manipulator Capturing a Satellite," *Proc. AIAA Guidance, Navigation, and Control Conference*, Boston, MA.

Valle, F., Tadeo, F., & Alvarez, T., (2002) "Predictive Control of Robotic Manipulators," *Proc. IEEE International Conference on Control Applications,* Glasgow, Scotland, U.K., pp. 203-208.

Wong, H.K., (2000) *Design, Integration, and Dynamical Model of a Multimodule Deployable Manipulator System (MDMS)*, M.A.Sc. Thesis, University of British Columbia, Vancouver, Canada.

Zhang, J., (2002) *Dynamics and Control of a Novel Manipulator with Deployable and Slewing Links*, M.A.Sc. Thesis, University of British Columbia, Vancouver, Canada.

APPENDIX A: Multi-Parametric Quadratic Programming (mp-QP)

A.1 Preamble

Multi-parametric quadratic programming (mp-QP) has been used throughout this thesis. The algorithm provides an offline solution to the quadratic programming (QP) problem so that constrained optimal control problems of high-bandwidth systems can be implemented in real time. In addition, explicit solutions to the QP problem allow for a simpler online implementation, and provide a better understanding of the optimal controller.

In this appendix, a theoretical background is given for the explicit solution to constrained predictive control problems. Work in this area has been developed by several people including Bemporad, et al. [2002]. This appendix outlines the mp-QP algorithm developed by Bemporad, followed by the online implementation of the explicit solution. The Multi-Parametric Toolbox of MATLAB provides the offline optimization algorithm [http://control.ee.ethz.ch/~mpt/].

A.2 Multi-Parametric Quadratic Programming for Model Predictive Control

Recall the predictive controller cost function discussed in Chapter 3:

$$V(k)=\frac{1}{2}\xi(k)^T\,\mathbf{Y}\,\xi(k)+\frac{1}{2}\mu(k)^T\,\mathbf{H}\,\mu(k)+\xi(k)^T\,\mathbf{F}\,\mu(k) \qquad \text{(A.1)}$$

In the minimization of this cost function, the parameters that change at each time step are the measured states of the plant, $x(k)$, the previous control inputs to the plant, $u(k\text{-}1)$, and the state of the desired values, $x_r(k)$. These parameters have been combined into a single vector, $\xi(k)$, in equation (A.1). The values of these parameters will determine the location of the unconstrained minimum point in the space $\Re^{M\cdot n_u}$ defined by the optimization variables, $\mu(k)$.

In constrained optimization, $\mu(k)$ is confined to a subset $\mathbf{U} \subseteq \Re^{M \cdot n_u}$, which is defined in Chapter 3 as

$$\mathbf{G}\mu(k) \leq W + \mathbf{E}\xi(k) \tag{A.2}$$

The location of the minimum point will now correspond to an active set of these constraints. Since the Hessian of $V(k)$ is symmetric and positive definite, the necessary and sufficient conditions for a minimum of the cost function are given by the first-order Karush-Kuhn-Tucker (KKT) conditions:

$$\nabla_\mu \mathcal{L}(\mu^*, \lambda^*) = \mathbf{0} \tag{A.3a}$$

$$\mathbf{G}\mu^*(k) \leq W + \mathbf{E}\xi(k) \tag{A.3b}$$

$$\lambda_j^* \geq 0 \tag{A.3c}$$

$$\lambda_j^* \left(W_j + \mathbf{E}_j \xi(k) - \mathbf{G}_j \mu^*(k)\right) = 0 \tag{A.4d}$$

where $\mathcal{L}(\mu, \lambda) = V(k) - \lambda^T \left(W + \mathbf{E}\xi(k) - \mathbf{G}\mu(k)\right)$ is the Lagrangian, with corresponding Lagrange multipliers, λ, and the subscript j denotes the j^{th} row. By solving (A.3a) we get

$$\nabla_\mu \mathcal{L}(\mu^*, \lambda^*) = \mathbf{H}\mu^*(k) + \mathbf{F}^T \xi(k) + \mathbf{G}^T \lambda^* = \mathbf{0}$$

$$\mu^*(k) = -\mathbf{H}^{-1}\left(\mathbf{F}^T \xi(k) + \tilde{\mathbf{G}}^T \tilde{\lambda}\right) \tag{A.4}$$

where the $\tilde{\lambda}$ and $\tilde{\mathbf{G}}$ contain only the rows corresponding to the active set of constraints. For the inactive constraints, $\lambda_j^* = 0$ in (A.4d), and for the rows in the active set, equation (A.4d) gives

$$\tilde{W} + \tilde{\mathbf{E}}\xi(k) - \tilde{\mathbf{G}}\mu^*(k) = \mathbf{0} \tag{A.5}$$

Substituting (A.4) into (A.5) and solving for $\tilde{\lambda}$ we get,

$$\begin{aligned}
&\tilde{W}+\tilde{\mathbf{E}}\xi(k)+\tilde{\mathbf{G}}\mathbf{H}^{-1}\left(\mathbf{F}^T\xi(k)+\tilde{\mathbf{G}}^T\tilde{\lambda}\right)=\mathbf{0}\\
&\tilde{\lambda}=-\left(\tilde{\mathbf{G}}\mathbf{H}^{-1}\tilde{\mathbf{G}}^T\right)^{-1}\left(\tilde{W}+\left(\tilde{\mathbf{E}}+\tilde{\mathbf{G}}\mathbf{H}^{-1}\mathbf{F}^T\right)\xi(k)\right)\\
&\tilde{\lambda}=-\left(\tilde{\mathbf{G}}\mathbf{H}^{-1}\tilde{\mathbf{G}}^T\right)^{-1}\left(\tilde{W}+\tilde{\mathbf{S}}\xi(k)\right)
\end{aligned} \tag{A.6}$$

where $\widetilde{\mathbf{S}}=\widetilde{\mathbf{E}}+\widetilde{\mathbf{G}}\mathbf{H}^{-1}\mathbf{F}^T$. Equation (A.6) can now be substituted back into (A.4) to give,

$$\mu^*(k)=\mathbf{H}^{-1}\left(\tilde{\mathbf{G}}^T\left(\tilde{\mathbf{G}}\mathbf{H}^{-1}\tilde{\mathbf{G}}^T\right)^{-1}\tilde{\mathbf{S}}-\mathbf{F}^T\right)\xi(k)+\mathbf{H}^{-1}\tilde{\mathbf{G}}^T\left(\tilde{\mathbf{G}}\mathbf{H}^{-1}\tilde{\mathbf{G}}^T\right)^{-1}\tilde{W}, \tag{A.7}$$

which is affine in the optimization parameters, $\xi(k)\in\Re^{N_p}$.

At the beginning of each control cycle, a new set of parameters is defined and equations (A.3) must be satisfied to determine the set of active constraints. Within a certain neighborhood of a given set of parameters, ξ_0, the active set of constraints will remain constant, and an affine control law will be defined throughout this region. The mp-QP algorithm determines (offline) the regions where the KKT conditions can be satisfied with the same set of active constraints. In doing so, the parameter space, $\Re^{N_p}$, is partitioned into a total of N_r different regions. The regions are defined by the inequalities (A.3b) and (A.3c), which can be written in terms of the parameters, $\xi(k)$, using equations (A.6) and (A.7). As a result (A.3c) becomes,

$$\begin{aligned}
-\left(\tilde{\mathbf{G}}\mathbf{H}^{-1}\tilde{\mathbf{G}}^T\right)^{-1}\left(\tilde{W}+\tilde{\mathbf{S}}\xi(k)\right) &\leq \mathbf{0}\\
\left(\tilde{\mathbf{G}}\mathbf{H}^{-1}\tilde{\mathbf{G}}^T\right)^{-1}\tilde{\mathbf{S}}\xi(k) &\leq -\left(\tilde{\mathbf{G}}\mathbf{H}^{-1}\tilde{\mathbf{G}}^T\right)^{-1}\tilde{W}
\end{aligned} \tag{A.8}$$

and equation (A.3b) becomes,

$$\begin{aligned}
\mathbf{G}\mathbf{H}^{-1}\left(\tilde{\mathbf{G}}^T\left(\tilde{\mathbf{G}}\mathbf{H}^{-1}\tilde{\mathbf{G}}^T\right)^{-1}\tilde{\mathbf{S}}-\mathbf{F}^T\right)\xi(k)+\mathbf{G}\mathbf{H}^{-1}\tilde{\mathbf{G}}^T\left(\tilde{\mathbf{G}}\mathbf{H}^{-1}\tilde{\mathbf{G}}^T\right)^{-1}\tilde{W} &\leq W+\mathbf{E}\xi(k)\\
\left\{\mathbf{G}\mathbf{H}^{-1}\tilde{\mathbf{G}}^T\left(\tilde{\mathbf{G}}\mathbf{H}^{-1}\tilde{\mathbf{G}}^T\right)^{-1}\tilde{\mathbf{S}}-\mathbf{G}\mathbf{H}^{-1}\mathbf{F}^T-\mathbf{E}\right\}\xi(k) &\leq W-\mathbf{G}\mathbf{H}^{-1}\tilde{\mathbf{G}}^T\left(\tilde{\mathbf{G}}\mathbf{H}^{-1}\tilde{\mathbf{G}}^T\right)^{-1}\tilde{W}
\end{aligned} \tag{A.9}$$

The rows of equations (A.8) and (A.9) form the facets of an N_p-dimensional convex polytope, within which the active constraints at the minimizer of the cost function remain unchanged. Each $\xi(k) \in \Re^{N_p}$ will belong to one of the N_r polytopes, and the optimal solution is given by the corresponding affine control law. The mp-QP algorithm stores these N_r polytopes and their corresponding control laws into a look-up table.

A.3 Offline Algorithm

The offline mp-QP algorithm is used to partition the N_p-dimensional space into the N_r different regions. The algorithm developed by Tondel, et al. [2003a] is outlined here to show the main steps taken in generating the predictive controller look-up table.

Algorithm A.1

Let R be the regions explored.

Let nR be the number of new regions found.

1) Start with an arbitrary, feasible point, ξ_0;
   $R = 0$, $nR = 1$
2) Solve the QP problem for the initial set of parameters to obtain an initial active set $\mathcal{A}_0$.
3) Form the constraints (A.8) and (A.9), given the current active set.
4) Combine (A.8) and (A.9) and remove any redundant constraints.
5) Check if result of Step 3a is identical to any previous region stored in the look-up table. If not, store new region description.
6) If new critical region found, store the matrix gain, and offset terms that describe the affine solution for this region, i.e., terms in equation (A.7); $R = R + 1$
7) For each border in the current region, check if there exists a new region just outside of it.
8) For each new region found in Step 7, form the optimal active set on the other side of the border by examining the type of border that is being crossed:
   a. If a border of type (A.8) is being crossed, i.e., a Lagrange multiplier is becoming negative, then remove the corresponding constraint from the current active set.
   b. If a border of type (A.9) is being crossed, i.e., a user-defined constraint is being violated, then add the corresponding constraint to the current active set
   c. For each new active set formed, $nR = nR + 1$
9) Repeat steps 3 through 8 for each new region that has been found.

10) Repeat steps 3 through 9 until the number of explored regions, R, equals the number of regions found, nR

A.4 Online Algorithm

The online algorithm is responsible for searching through the look-up table produced by the mp-QP solver to determine which region the current measured set of parameters belongs to, and then applying the corresponding control law. Several efficient algorithms exist for performing this search [Borelli, et al., 2001; Tondel, et al., 2003b], however, a sequential search is used in Chapter 5 and it is a sequential search, which is outlined in Algorithm A.2.

Algorithm A.2

1) Measure the current optimization parameters, $\xi(k)$.
2) Sequentially search through the N_r regions, until the i^{th} region is found, which satisfies
   $\mathbf{H}_i \cdot \xi(k) \leq \boldsymbol{K}_i$ for $i = 1,2,3, \ldots, N_r$ {Equation (5.1)}
3) Implement the i^{th} control law,
   $\mu(k) = \mathbf{F}_i \cdot \xi(k) + \boldsymbol{G}_i$ {Equation (5.2)}

A.5 The Multi-Parametric Toolbox (MPT) of MATLAB

The multi-parametric toolbox (MPT) of MATLAB provided all the mp-QP offline solutions used in the present thesis. Version 1.2 of the toolbox was used. It provided the mp-QP solver 'mpt_mpqp().' Once equations (A.1) and (A.2) had been formed, the matrices **Y**, **H**, **F**, **G**, $\boldsymbol{W}$ and **E** were used as inputs to this function.

[*Pn*, *Fi*, *Gi*, *activeConstraints*, *Phard*] = mpt_mpqp(*Matrices*, *Options*)

Pn = an array containing the description of the N_r regions found in the partition

Fi = an array of the matrix gains used in the affine control law for the corresponding region {see Equation (5.2)}

Gi = array of the vector offsets used in the affine control law for the corresponding region {see equation (5.2)}

activeConstraints = list of constraints that are active in the corresponding regions

Phard = the union of all the explored regions; the convex hull of all feasible points in $\Re^{N_p}$

Matrices = structure containing the matrices **Y**, **H**, **F**, **G**, ***W***, and **E**

Options = allows the user to:

- select which LP/QP solvers are used;
- choose the level of debugging performed during the partitioning;
- define the step size taken when crossing over a facet

In addition to the mp-QP solver, the MPT provides several computational geometry algorithms, which are used by the mp-QP solver. These algorithms can be used to further analyze the explicit MPC solution. In particular, when only two parameters are considered in the optimization, the MPT provides several plotting functions that allow the user to graphically view the partitioned regions.

The multi-parametric toolbox is constantly being upgraded. Developers continue to improve the numerical stability of the algorithms, and add new features, like efficient search algorithms for online implementation. Material related to the MPT, along with the most recent version of the toolbox, can be found at http://control.ee.ethz.ch/~mpt/.

APPENDIX B:
Dynamic Formulation of the MDMS

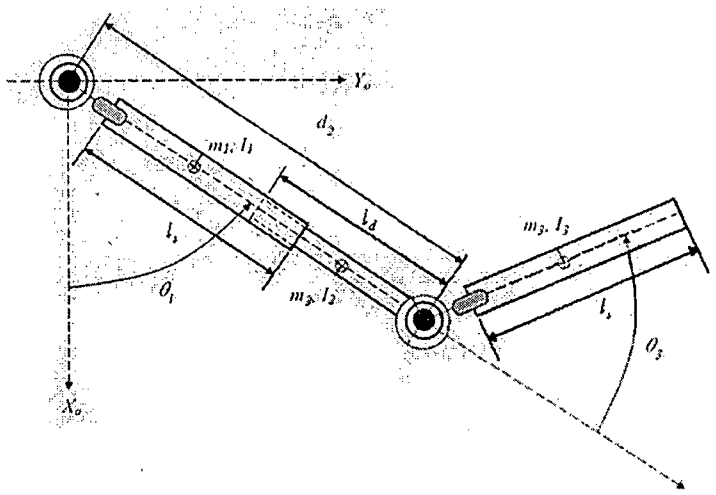


Figure B.1 - Schematic diagram of the MDMS used for planar satellite capturing.

Kinetic Energy of the System

Link 1 (slewing):

$$T_1 = \frac{1}{8} m_1 l_s^2 \dot{\theta}_1^{\,2} + \frac{1}{2} I_s \dot{\theta}_1^{\,2} \tag{B.1}$$

Link 2 (deployable):

$$T_2 = \frac{1}{2} m_2 \left(d_2 - \tfrac{1}{2} l_d\right) \dot{\theta}_1^{\,2} + \frac{1}{2} m_2 \dot{d}_2^{\,2} + \frac{1}{2} I_2 \dot{\theta}_1^{\,2} \tag{B.2}$$

Link 3 (slewing):

$$\begin{aligned} T_3 = {} & \frac{1}{2} m_3 \left(d_2^{\,2} + \tfrac{1}{4} l_s^{\,2} + l_s d_2 \cos\theta_3\right) \dot{\theta}_1^{\,2} + \frac{1}{2} m_3 \dot{d}_2^{\,2} + \frac{1}{8} m_3 l_s^{\,2} \dot{\theta}_3^{\,2} \\ & - \frac{1}{2} m_3 l_s \sin\theta_3 \dot{\theta}_1 \dot{d}_2 - \frac{1}{2} m_3 l_s \sin\theta_3 \dot{d}_2 \dot{\theta}_3 + \frac{1}{2} m_3 \left(l_s d_2 \cos\theta_3 + \tfrac{1}{2} l_s^{\,2}\right) \dot{\theta}_1 \dot{\theta}_3 \\ & + \frac{1}{2} I_3 \left(\dot{\theta}_1 + \dot{\theta}_3\right)^2 \end{aligned} \tag{B.3}$$

Total kinetic energy:

$$T = T_1 + T_2 + T_3 \tag{B.4}$$

N.B.: no potential energy terms are present because the planar manipulator is supported by roller bearings to remove the effects of gravity and joint and links are known to be fully rigid.

Applying the Lagrangian procedure, the equations of motion are given as

$$\mathbf{M}(q)\ddot{q} + C(q,\dot{q}) = \boldsymbol{\tau} \tag{B.5}$$

where:

q = vector of joint positions $\begin{bmatrix}\theta_1 & d_2 & \theta_3\end{bmatrix}^T$

$\dot{q}$ = vector of joint velocities $\begin{bmatrix}\dot{\theta}_1 & \dot{d}_2 & \dot{\theta}_3\end{bmatrix}^T$

$\ddot{q}$ = vector of joint accelerations $\begin{bmatrix}\ddot{\theta}_1 & \ddot{d}_2 & \ddot{\theta}_3\end{bmatrix}^T$

$\boldsymbol{\tau}$ = vector of joint forces, f, and torques, τ, $\boldsymbol{\tau} = \begin{bmatrix}\tau_1 & f_2 & \tau_3\end{bmatrix}^T$

$$\mathbf{M}(q) = \begin{bmatrix} \frac{1}{4}m_1 l_s^2 + I_1 + I_2 + I_3 & & \\ +m_2\left(d_2 - \frac{1}{2}l_d\right)^2 & -\frac{1}{2}m_3 l_s \sin\theta_3 & \frac{1}{2}m_3\left(l_s d_2 \cos\theta_3 + \frac{1}{2}l_s^2\right) \\ +m_3\left(d_2^2 + \frac{1}{4}l_s^2 + l_s d_2 \cos\theta_3\right) & & + I_3 \\ -\frac{1}{2}m_3 l_s \sin\theta_3 & m_2 + m_3 & -\frac{1}{2}m_3 l_s \sin\theta_3 \\ \frac{1}{2}m_3\left(l_s d_2 \cos\theta_3 + \frac{1}{2}l_s^2\right) + I_3 & -\frac{1}{2}m_3 l_s \sin\theta_3 & \frac{1}{4}m_3 l_s^2 + I_3 \end{bmatrix} \tag{B.6}$$

$$\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \begin{bmatrix} \left\{2m_2\left(d_2 - \frac{1}{2}l_d\right) + 2m_3 d_2 + m_3 l_s \cos\theta_3\right\}\dot{\theta}_1\dot{d}_2 \\ -m_3 l_s d_2 \sin\theta_3 \dot{\theta}_1\dot{\theta}_3 - \frac{1}{2}m_3 l_s d_2 \sin\theta_3 \dot{\theta}_3^{\,2} \\ -\left\{m_2\left(d_2 - \frac{1}{2}l_d\right) + m_3 d_2 + \frac{1}{2}m_3 l_s \cos\theta_3\right\}\dot{\theta}_1^{\,2} \\ -m_3 l_s \cos\theta_3 \dot{\theta}_1\dot{\theta}_3 - \frac{1}{2}m_3 l_s \cos\theta_3 \dot{\theta}_3^{\,2} \\ \frac{1}{2}m_3 l_s d_2 \sin\theta_3 \dot{\theta}_1^{\,2} + m_3 l_s \cos\theta_3 \dot{\theta}_1\dot{d}_2 \end{bmatrix} \tag{B.7}$$

Viscous ($\mathbf{F}_v$) and Coulomb ($\mathbf{F}_s$) friction terms were then added to give the complete model:

$$\mathbf{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \mathbf{F}_s \operatorname{sgn}(\dot{\boldsymbol{q}}) + \mathbf{F}_v\dot{\boldsymbol{q}} = \boldsymbol{\tau} \tag{B.8}$$

The model parameters used in the design of the controller are found in [Wong, 2000]. The parameters are:

- $m_1 = 2.3$ kg
- $m_2 = 0.75$ kg
- $m_3 = 3.05$ kg
- $l_s = 0.30$ m
- $l_d = 0.22$ m

Equation (B.8) assumes that the friction in the manipulator can be described by only static and viscous friction terms at the joints. Of course, this is a considerable simplification, and since the linear predictive controller is applied directly to the manipulator, the friction terms do not directly affect the internal model of the controller. So, values for $\mathbf{F}_s$ and $\mathbf{F}_v$ are chosen so that input commands to the simulated robot and the experimental setup display similar results.

APPENDIX C:
Further Results in Satellite Capturing Using Predictive Control

C.1 Control Horizon MPC Results

A control horizon form of the predictive satellite capturing controller was developed prior to the PFC version discussed in this thesis [McCourt & de Silva, 2003]. This unconstrained controller used the least squares solution as discussed in CHAPTER 2: . The full nonlinear predictions of the target satellite's position, and orientation, were used to form the vector of future setpoints, $\boldsymbol{T}(k)$, which was then used in the controller shown in Figure C.1.

A prediction horizon of 50 and a control horizon of 20 were used and three simulated satellite captures were performed. In the first, a simulated target passed by the manipulator at 0.1 m/s, parallel to the platform, and with no rotational velocity. In the second, the center of gravity of the target remained stationary relative to the base of the manipulator, and the target rotated at a constant angular velocity of 0.087 rad/s (5°/s). In the third scenario, the target both translated and rotated as it passed by the manipulator.

The results in Table C.1 are similar to the results in Figure 4.2, i.e., predictions of the future target position improve the tracking performance of the manipulator. This result is presented graphically in Figure C.2, where the joint space tracking results have been plotted for the capture of the spinning, and translating, satellite.

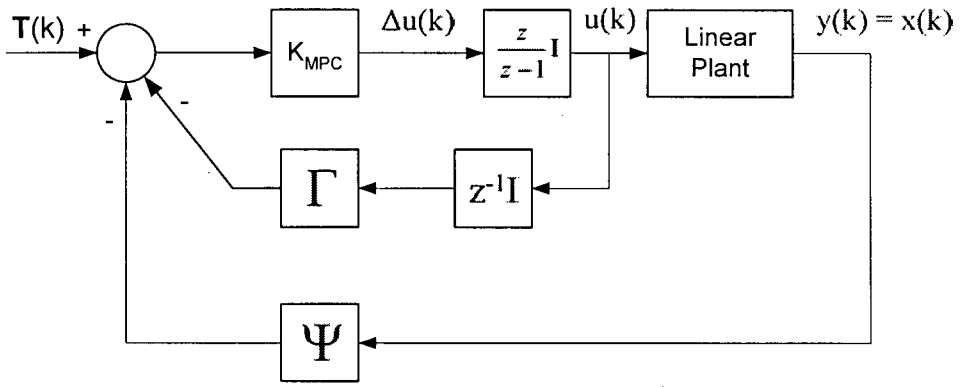


Figure C.1 - Block diagram of the unconstrained control horizon form of the predictive controller.

Table C.1 - Steady state tracking error results from the control horizon form of the predictive controller.

	Target Prediction?	Average Steady State Tracking Errors		
		θ_1 error	d_2 error	θ_3 error
Translating only	yes	0.02211°	0.000819 m	0.18336°
	no	-2.08490°	0.000692 m	1.83530°
Spinning only	yes	0.22539°	0.002350 m	0.35425°
	no	-1.12806°	-0.003110 m	1.81234°
Translating and Spinning	yes	-0.21157°	-0.002330 m	0.57952°
	no	-1.72780°	-0.003540 m	0.98413°

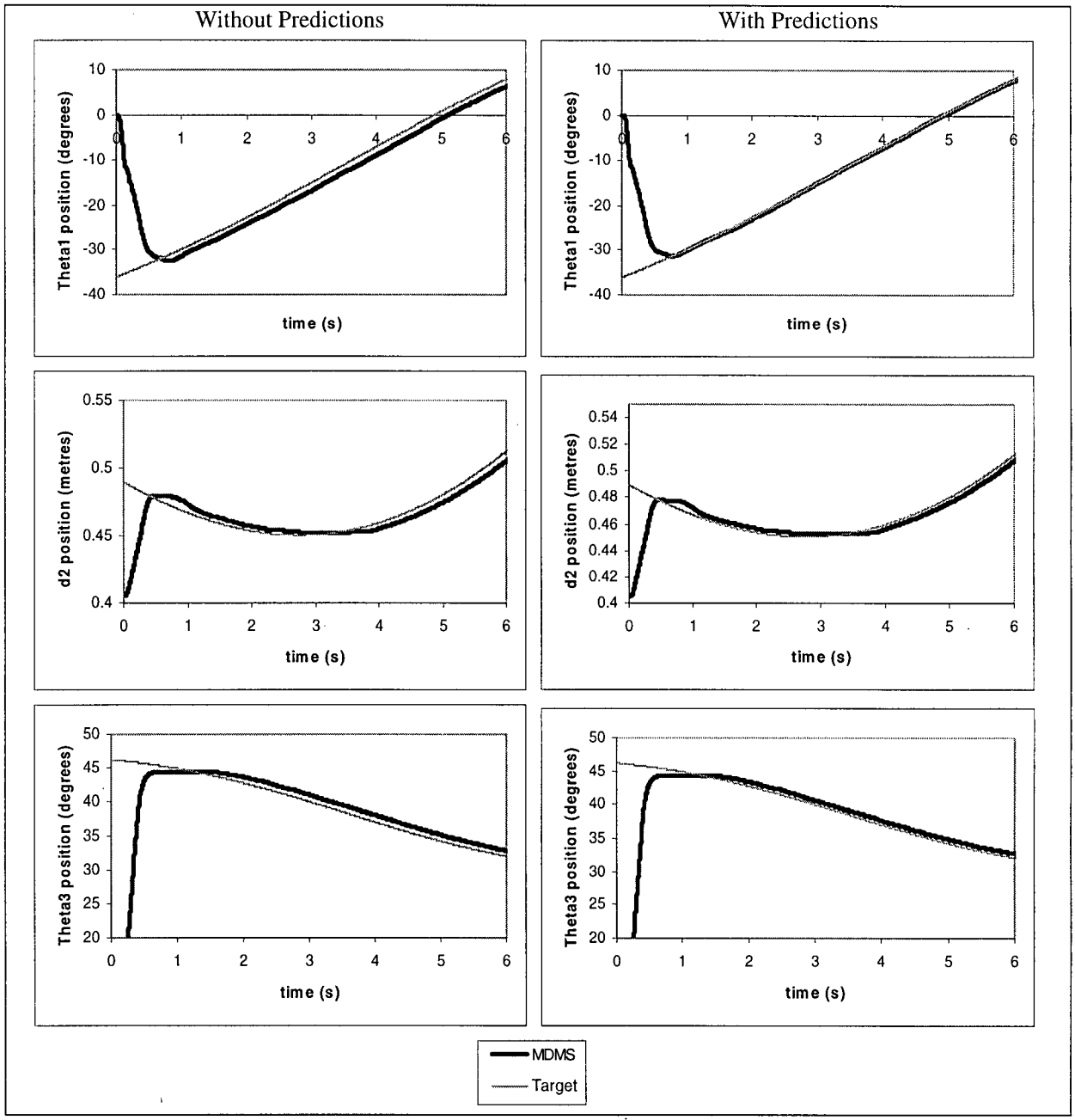


Figure C.2 - MDMS joint-space tracking error response with, and without, predictions of the targets position.

C.2 More Experimental Satellite Capture Results

Experimental results from Chapter 5 demonstrated the implementation of the unconstrained and constrained predictive controllers on the prototype manipulator in our laboratory. Results from satellite targets with different angular and translatory speed have been presented in the following table, and figures to show the ability of the linear predictive controller to guide the Multi-module Deployable Manipulator System (MDMS) towards a variety of spinning targets.

Table C.2 - Experimental results from various satellite captures

	Target details			Tracking accuracy		
Case No.	v_x (m/s)	v_y (m/s)	ω (rad/s)	Position	Orientation	Figure No.
1	0.005	-0.050	$\pi/12$	1.8±2.7mm	0.09°±0.14°	D.3, D.4
2	0.000	-0.050	$\pi/24$	1.1±0.8mm	0.15°±0.09°	D.5, D.6
3	0.000	-0.100	0	1.7±0.9mm	0.29°±0.08°	D.7, D.8

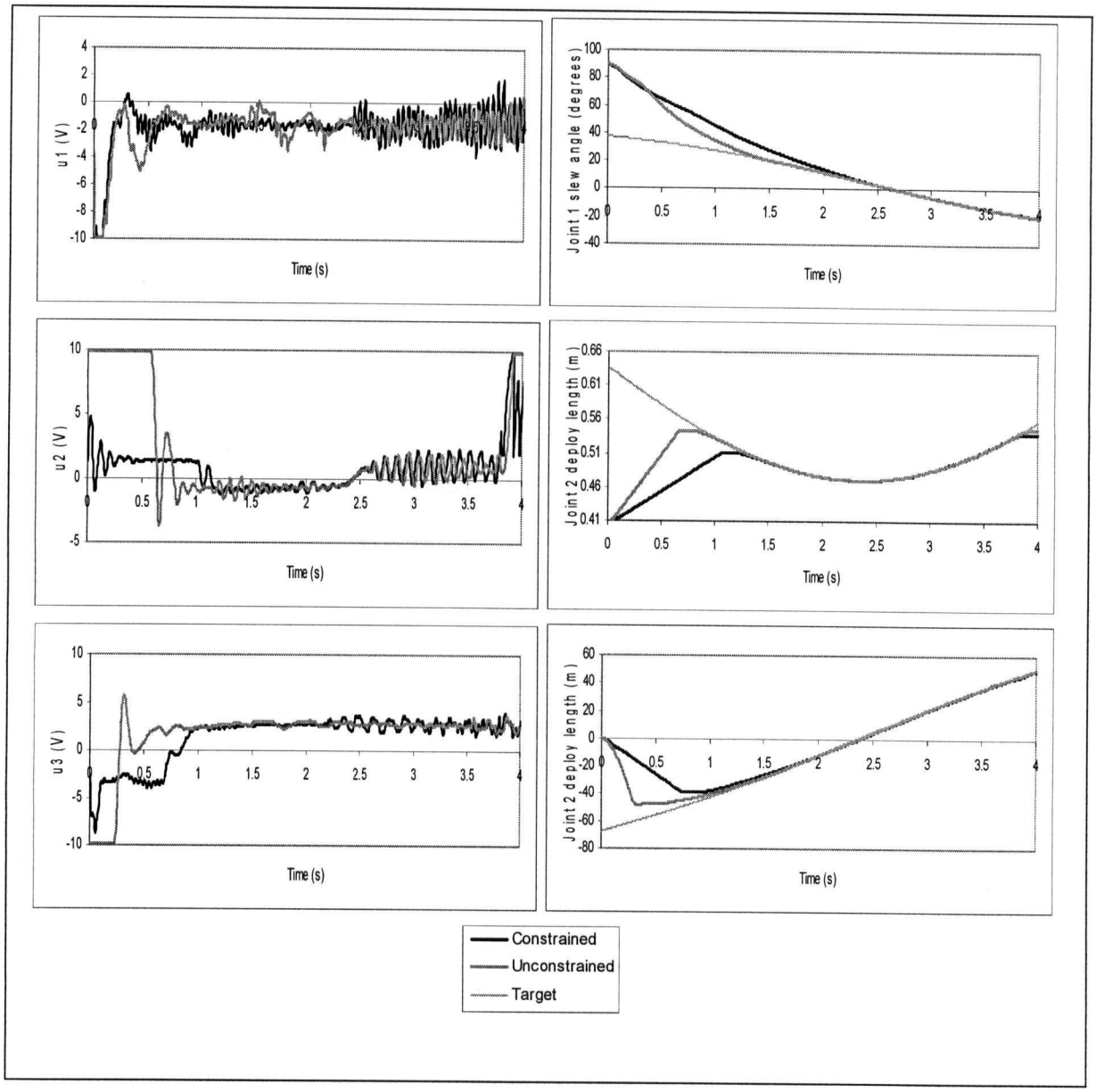


Figure C.3 – Joint space constrained predictive controller results for Case #1.

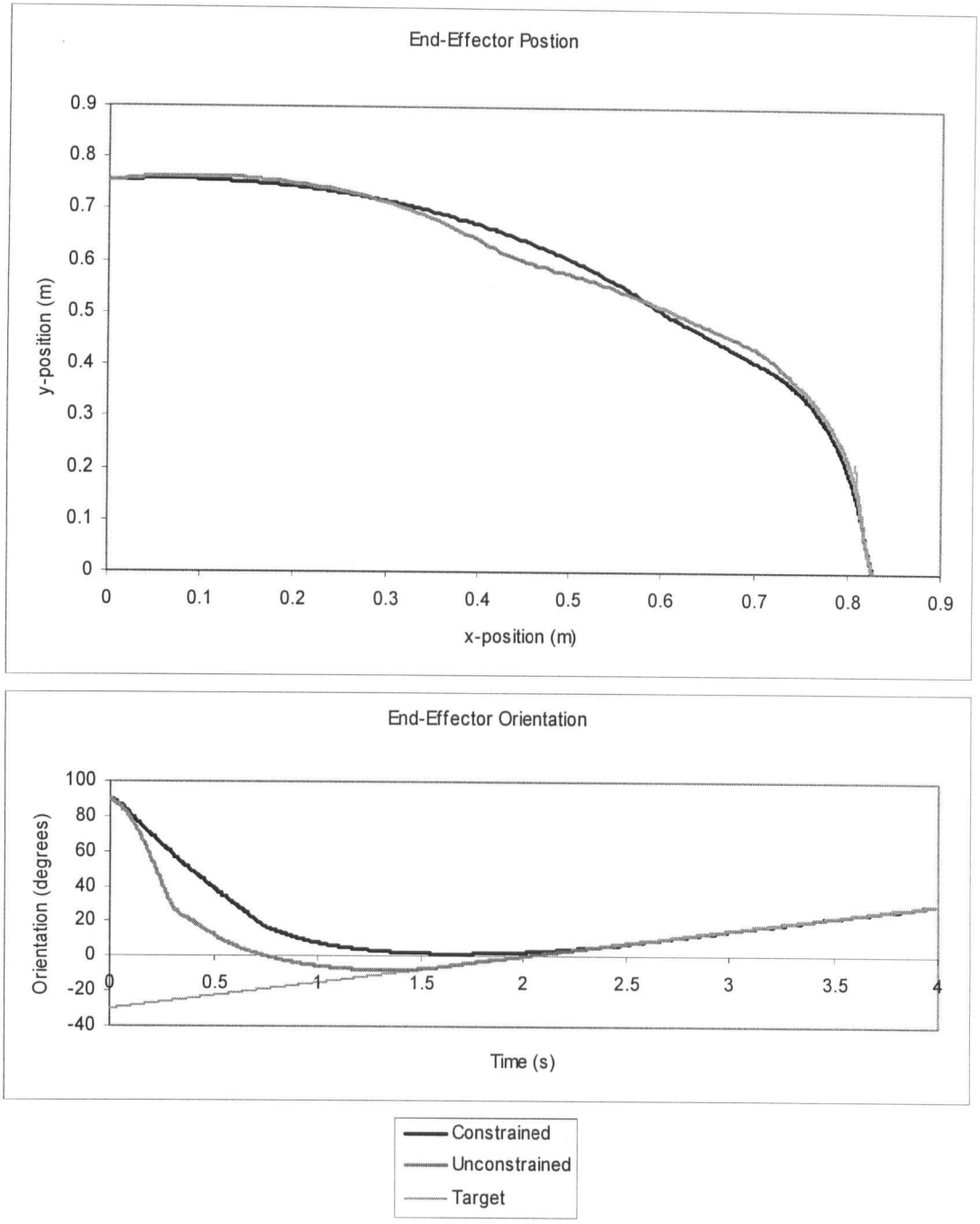


Figure C.4 - Task space constrained predictive controller results for Case #1.

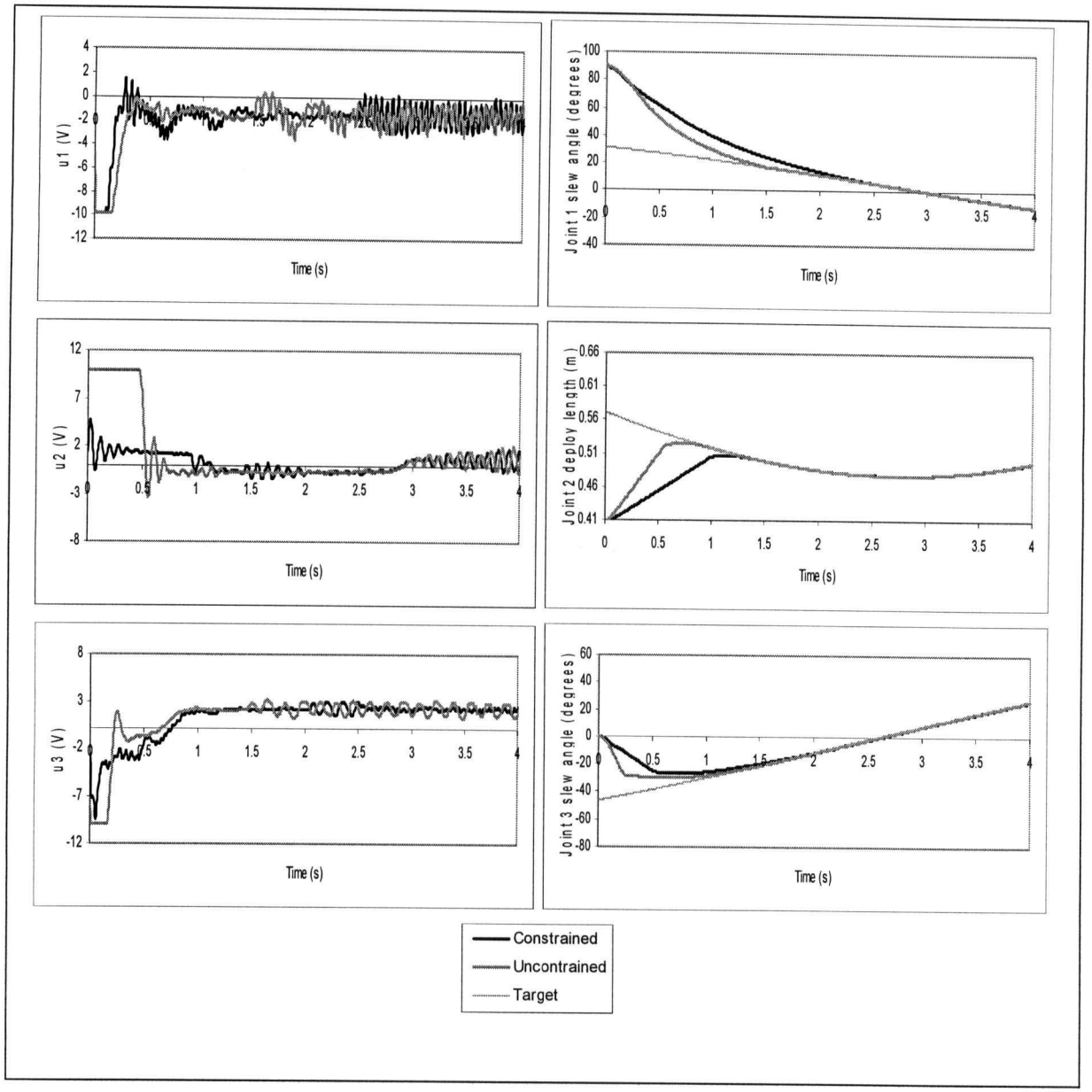


Figure C.5 - Joint space constrained predictive controller results for Case #2.

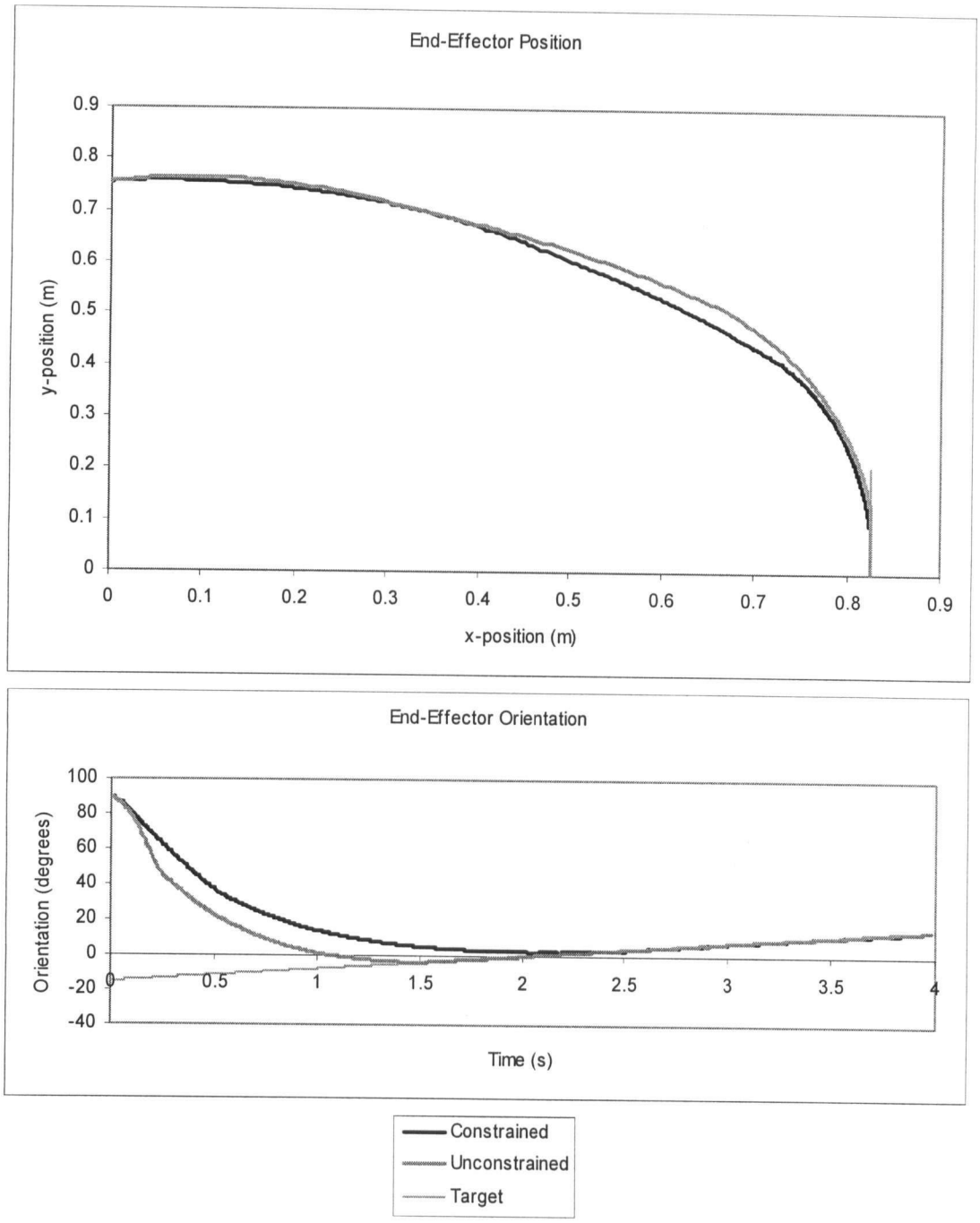


Figure C.6 - Task space constrained predictive controller results for Case #2.

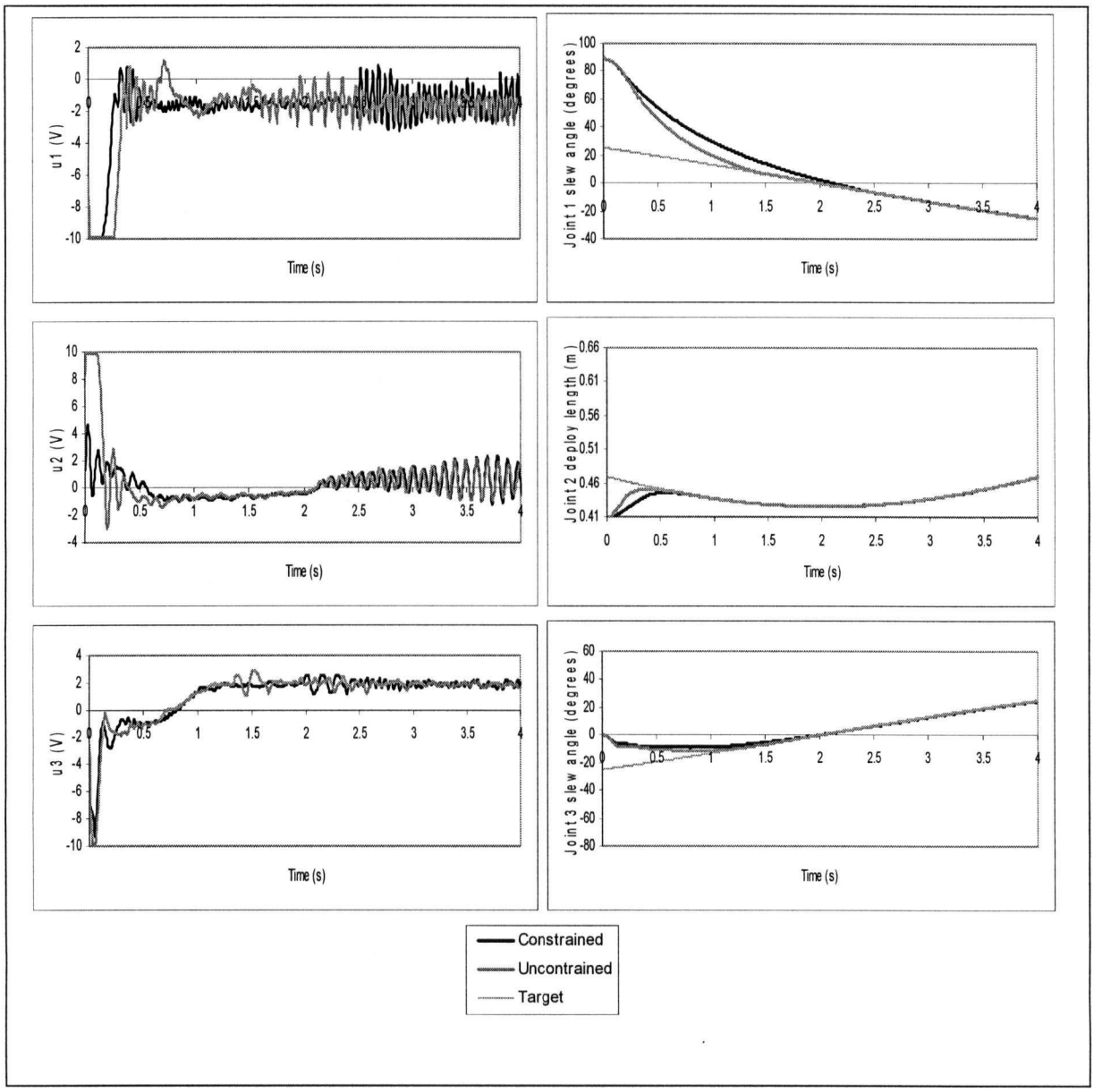


Figure C.7 - Joint space constrained predictive controller results for Case #3.

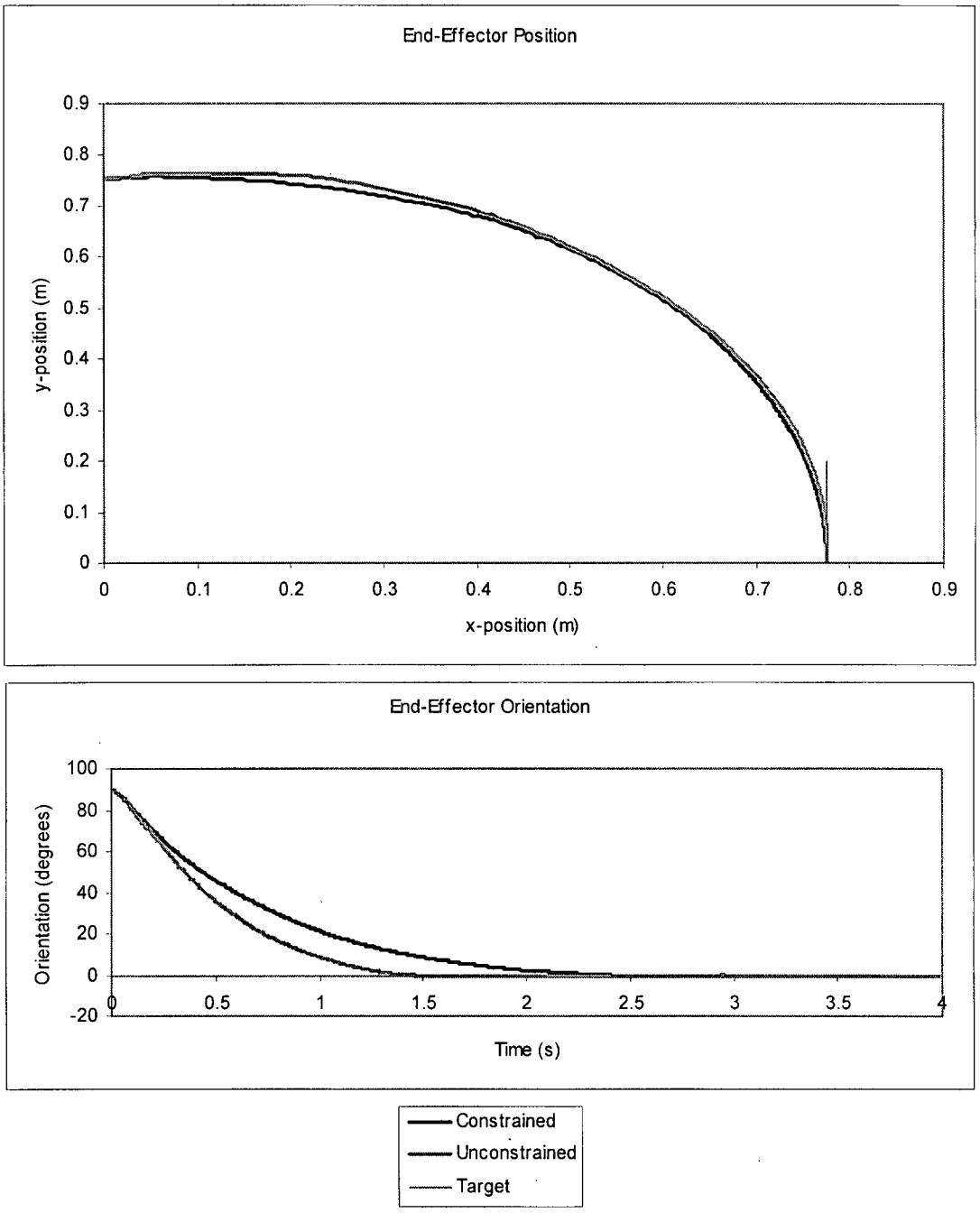


Figure C.8 - Task space constrained predictive controller results for Case #3.