A Model Predictive Approach to Optimal Path-Following and Contouring Control

Denise Lam

Submitted in total fulfilment of the requirements of the degree of Doctor of Philosophy

Department of Mechanical Engineering
THE UNIVERSITY OF MELBOURNE

September 2012

Produced on archival quality paper.
Copyright © 2012 Denise Lam

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm or any other means without written permission from the author.
Abstract

In the multi-billion dollar motion control industry, systems are often required to traverse along a geometric path, where the speed at which the path is traversed is not pre-determined. Such path-following tasks involve competing control objectives of maximising accuracy while minimising traversal time. Existing trajectory planning routines optimise the path speed with respect to some of these objectives. However, trajectory planning is conducted in an open loop fashion, relying on trajectory tracking controllers to achieve the desired motion. Meanwhile, closed-loop path-following control schemes do not explicitly consider the control objectives.

This thesis investigates the development of closed-loop path-following control schemes based on minimisation of a cost function within a receding horizon structure. The cost function is selected to reflect the desired objectives such as minimising traversal time, and is minimised subject to the system constraints. The receding horizon formulation allows for feedback to be taken into account at each sample, in contrast to open loop trajectory planning.

A receding horizon path-following control scheme is first developed with the intention of achieving time-optimal path-following. The controller is designed to advance along the path as far as possible within a finite horizon while staying exactly on the path and honouring the constraints. The conditions under which the receding horizon approach replicates the time-optimal path-following solution are identified for second order differentially flat systems. In particular, it is shown that minimum-time path-following is achieved when the horizon is sufficiently long.
Subsequently, a contouring control scheme for biaxial systems based on a similar framework is developed where the exact path-following constraint is relaxed in favour of penalising contouring error in the finite horizon cost function, which reflects the trade-off between competing control objectives of contouring accuracy, productivity and minimisation of control deviations. The penalty weights may be tuned to favour some objectives over others. A computationally efficient algorithm is proposed, allowing the scheme to be implemented in real time on an X-Y table. Experimental results demonstrate how the behaviour of the controller can be adjusted in a systematic manner by tuning the penalty weights. Results also show that the proposed scheme achieves superior contouring performance compared with tracking controllers implemented with constant path speed.

The proposed contouring control approach is then extended to more complicated multi-axis systems involving position and orientation control. A multi-rate control architecture is proposed to facilitate the implementation of the contouring control scheme on industrial hardware. Simulations are conducted with an industrial profile cutting machine model, and results show similar behaviour to the biaxial version. Alternative implementations requiring varying degrees of modification to existing hardware are proposed and tested in simulation, highlighting the trade-off between contouring performance and hardware modification.
Declaration

This is to certify that

1. the thesis comprises only my original work towards the PhD,
2. due acknowledgement has been made in the text to all other material used,
3. the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Denise Lam, September 2012
Acknowledgements

I would like to thank Associate Professor Chris Manzie and Professor Malcolm Good for their guidance and support over the last 3 years. I count myself lucky to have received such professional and helpful supervision, as well as having a pleasurable learning experience. In particular, I acknowledge the inordinate amount of patience that Chris and Malcolm have displayed during the drafting of this thesis and related publications.

I was also fortunate enough to receive support from the industry partner, ANCA Motion. I would like to particularly thank David Fisher and Troy Robinette for giving me the opportunity to work on a project with relevance to industry. In addition, the assistance provided by Craig Fletcher, Peter Harding, James Little and Migara Dharmadasa is greatly appreciated. I also gratefully acknowledge the support of the Elizabeth and Vernon Puzey Scholarship.

Throughout my PhD I have benefited from collaborating with other researchers around the world. In particular, my work in the area of time-optimal control was conducted in collaboration with Professor Robert Bitmead from the University of California, San Diego. Robert’s encouragement and technical insight allowed me to look at my research from a different perspective, and his warm personality and sense of humour allowed me to thoroughly enjoy doing so. I would also like to thank Timm Faulwasser from the Otto-von-Guericke University in Magdeburg, Germany, whose work forms the basis of my own, and who has also fostered collaboration between our two research groups.

I am grateful to have been part of a vibrant research group in the Mechanical Engineer-
ing Department at the University of Melbourne. Its members have come and gone over the years and are too numerous to mention. I would particularly like to thank fellow PhD student and office buddy Michael Stephens, who blazed the trail for my project in many ways. Franz Bauer and Maria Weiß, both visiting students from Germany, provided invaluable assistance with my laboratory work.

As a female PhD student in Mechanical Engineering I am in the minority; however I never experienced any isolation and felt extremely well supported over the last 3 years. This is in no small part due to the actions of Professor Doreen Thomas, who has made a great effort to form a support network for women in the department.

Finally, I would like to thank John Papandriopoulos for the use of his excellent thesis template.
4.3 Linear time-varying model predictive contouring control ........................................ 65
4.4 Guaranteed path completion ...................................................................................... 69
4.5 Conclusion ................................................................................................................. 74

5 Application of biaxial MPCC to an X-Y Table ............................................................ 75
5.1 Test rig description ....................................................................................................... 75
5.2 X-Y table modelling ..................................................................................................... 77
5.3 MPCC implementation ................................................................................................. 81
5.4 Benchmarking controllers ........................................................................................... 82
5.5 Experimental results .................................................................................................... 85
5.6 Conclusion .................................................................................................................... 91

6 Multi-axis model predictive contouring control with industrial system architecture ................................................................. 93
6.1 Industrial system architecture ...................................................................................... 93
6.2 The multi-axis contouring control problem ................................................................ 95
6.3 Multi-axis model predictive contouring control formulation .................................... 100
6.4 Linear time-varying multi-axis MPCC ....................................................................... 103
6.5 Conclusion .................................................................................................................... 110

7 Application of multi-axis MPCC to an industrial machine model .................................. 111
7.1 Machine description ..................................................................................................... 111
7.2 Simulation model ......................................................................................................... 114
7.3 Joint controllers .......................................................................................................... 116
7.4 MPCC implementation with joint feedback .............................................................. 117
7.5 Simulations .................................................................................................................... 119
7.6 Conclusion .................................................................................................................... 126

8 Alternative MPCC implementations ............................................................................ 127
8.1 MPCC implementation with motor feedback ............................................................ 127
8.2 Open loop MPCC implementation .............................................................................. 133
8.3 Comparison of MPCC implementations .................................................................... 138
8.4 Conclusion .................................................................................................................... 140
List of Figures

1.1 Typical motion control system ............................................. 1
1.2 A laser profiling machine [44] ........................................... 2
1.3 Comparison of tracking and path-following .......................... 3
1.4 Corner profile ................................................................... 4

2.1 Conventional architecture with separate trajectory planning and tracking ........................................... 9
2.2 Minimum-time trajectory in the $\theta$-$\dot{\theta}$ phase plane ......... 11
2.3 Reference governor architecture [49] ................................. 16
2.4 Hierarchical control structure with reference governor and velocity adaptation [114] ......................... 18
2.5 Path governor [10] ............................................................. 19
2.6 Cascaded PI control architecture ....................................... 20
2.7 ZPETC control scheme [117] ............................................ 21
2.8 Contour error versus tracking error [64] ............................. 23
2.9 Cross-coupled control structure [112] ................................. 24
2.10 Generalized predictive cascade control architecture [14] ....... 27
2.11 Path-following control architecture ................................... 30
2.12 Velocity field for a circular path [76] ................................. 33

3.1 Illustration of proof of Lemma 3.1 ....................................... 46
3.2 Elliptical path .................................................................... 54
3.3 Input trajectories for elliptical path ..................................... 55
3.4 Flower-shaped path .......................................................... 56
3.5 Plots of $\dot{x}$, $\dot{y}$ and $v$ versus $\theta$ with varying horizon lengths for flower-shaped path ......................... 57
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.10</td>
<td>Contouring error for joint feedback MPCC for ( q_\theta = 0.05 ) and ( q_\theta = 2 )</td>
<td>123</td>
</tr>
<tr>
<td>7.11</td>
<td>Workpiece error for joint feedback MPCC for ( q_\theta = 0.05 ) and ( q_\theta = 2 )</td>
<td>124</td>
</tr>
<tr>
<td>7.12</td>
<td>Traversal time and RMS contouring error vs. traversal time weighting for MPCC with joint feedback</td>
<td>125</td>
</tr>
<tr>
<td>8.1</td>
<td>MPCC with motor feedback</td>
<td>128</td>
</tr>
<tr>
<td>8.2</td>
<td>Joint command acceleration and jerk for motor feedback MPCC for ( q_\theta = 0.05 ) and ( q_\theta = 2 )</td>
<td>129</td>
</tr>
<tr>
<td>8.3</td>
<td>Path speed for motor feedback MPCC for ( q_\theta = 0.05 ) and ( q_\theta = 2 )</td>
<td>130</td>
</tr>
<tr>
<td>8.4</td>
<td>Contouring error for motor feedback MPCC for ( q_\theta = 0.05 ) and ( q_\theta = 2 )</td>
<td>130</td>
</tr>
<tr>
<td>8.5</td>
<td>Workpiece error for motor feedback MPCC for ( q_\theta = 0.05 ) and ( q_\theta = 2 )</td>
<td>131</td>
</tr>
<tr>
<td>8.6</td>
<td>Traversal time and RMS contouring error vs. traversal time weighting for MPCC with motor feedback</td>
<td>132</td>
</tr>
<tr>
<td>8.7</td>
<td>Open loop MPCC</td>
<td>133</td>
</tr>
<tr>
<td>8.8</td>
<td>Joint command acceleration and jerk for open loop MPCC for ( q_\theta = 0.05 ) and ( q_\theta = 2 )</td>
<td>135</td>
</tr>
<tr>
<td>8.9</td>
<td>Path speed for open loop MPCC for ( q_\theta = 0.05 ) and ( q_\theta = 2 )</td>
<td>136</td>
</tr>
<tr>
<td>8.10</td>
<td>Contouring error for open loop MPCC for ( q_\theta = 0.05 ) and ( q_\theta = 2 )</td>
<td>136</td>
</tr>
<tr>
<td>8.11</td>
<td>Workpiece error for open loop MPCC for ( q_\theta = 0.05 ) and ( q_\theta = 2 )</td>
<td>137</td>
</tr>
<tr>
<td>8.12</td>
<td>Traversal time and RMS contouring error vs. traversal time weighting for open loop MPCC</td>
<td>137</td>
</tr>
<tr>
<td>8.13</td>
<td>RMS contouring error versus traversal time</td>
<td>138</td>
</tr>
<tr>
<td>8.14</td>
<td>RMS workpiece error versus traversal time</td>
<td>139</td>
</tr>
<tr>
<td>8.15</td>
<td>RMS contouring error versus traversal time with reduced PI gain</td>
<td>141</td>
</tr>
<tr>
<td>8.16</td>
<td>RMS workpiece error versus traversal time with reduced PI gain</td>
<td>141</td>
</tr>
<tr>
<td>A.1</td>
<td>Friction models and measured data for the X-axis</td>
<td>167</td>
</tr>
<tr>
<td>A.2</td>
<td>X-Y table system-identification experiment</td>
<td>169</td>
</tr>
<tr>
<td>A.3</td>
<td>Closed-loop X-axis model validation</td>
<td>170</td>
</tr>
<tr>
<td>A.4</td>
<td>Open loop X-axis model validation</td>
<td>171</td>
</tr>
<tr>
<td>B.1</td>
<td>Simplified geometry used to calculate rotational inertia matrices</td>
<td>174</td>
</tr>
<tr>
<td>B.2</td>
<td>Scale diagram showing centre of gravity locations for links 4 and 5</td>
<td>176</td>
</tr>
<tr>
<td>B.3</td>
<td>Frequency response from ( T_1'' ) to ( q_1 )</td>
<td>177</td>
</tr>
</tbody>
</table>
3.4 Frequency response from $T_2^m$ to $\dot{q}_2$
List of Tables

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Evaluation of servo controllers [63]</td>
<td>25</td>
</tr>
<tr>
<td>3.1</td>
<td>Rigid X-Y table model parameters</td>
<td>54</td>
</tr>
<tr>
<td>4.1</td>
<td>Values of $l$ and $m$ for increasing $A$</td>
<td>69</td>
</tr>
<tr>
<td>5.1</td>
<td>X-Y table model parameters</td>
<td>80</td>
</tr>
<tr>
<td>7.1</td>
<td>D-H parameters for profile cutting machine</td>
<td>112</td>
</tr>
<tr>
<td>7.2</td>
<td>Parameters for multi-axis MPCC simulation</td>
<td>120</td>
</tr>
<tr>
<td>7.3</td>
<td>Contouring error and traversal time for $q_1 = 0.05$ and $q_2 = 2$ for joint feedback MPCC</td>
<td>125</td>
</tr>
<tr>
<td>8.1</td>
<td>Contouring error and traversal time for $q_1 = 0.05$ and $q_2 = 2$ for motor feedback MPCC</td>
<td>131</td>
</tr>
<tr>
<td>8.2</td>
<td>Contouring error and traversal time for $q_1 = 0.05$ and $q_2 = 2$ for open loop MPCC</td>
<td>134</td>
</tr>
<tr>
<td>A.1</td>
<td>X-axis friction parameters</td>
<td>167</td>
</tr>
<tr>
<td>A.2</td>
<td>Y-axis friction parameters</td>
<td>168</td>
</tr>
<tr>
<td>A.3</td>
<td>Identified X-Y table model parameters</td>
<td>169</td>
</tr>
<tr>
<td>B.1</td>
<td>Dimension values for simplified geometry</td>
<td>175</td>
</tr>
<tr>
<td>B.2</td>
<td>Motor, coupling and gear parameters for profile cutting machine model</td>
<td>176</td>
</tr>
<tr>
<td>B.3</td>
<td>Serial link parameters for profile cutting machine model</td>
<td>177</td>
</tr>
<tr>
<td>C.1</td>
<td>Cascaded PI gains for profile cutting machine joint controllers</td>
<td>179</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Motion control involves the use of actuators, such as electric motors, to control the position and/or velocity of a machine. Applications include robot control, cutting and grinding machines, food processing, computer disk drives and many others. In 2006, the market size for motion control annual sales in the United States was estimated to be between 1.25 and 2.25 billion US dollars \cite{85}. Another source states that the size of the global robotics market is around 24 billion US dollars \cite{16}.

A typical electromechanical motion control system consists of a number of servo motors and drives, as shown in Figure 1.1. The drives apply voltage to the motors in order to achieve a particular motion, such as rotation about a specified axis, or translation in a specified direction. Translational motion is achieved either with a linear motor or by combining a lead screw or pulley with a conventional rotary motor.

Figure 1.1: Typical motion control system
Feedback is available to the servo drives in the form of motor current and position measurements. Additionally, in some applications, measurements of load position are also available. The motor and load velocities can also be employed as feedback, either by direct measurement or by differentiation of the position feedback signals.

Typically, each drive controls the motion of a single motor, and is unaware of the motion of the other axes. A higher level system, referred to in Figure 1.1 as the motion co-ordinator, communicates with the servo drives in order to co-ordinate the axes such that the desired motion is achieved. The motion co-ordinator is known as the Computer Numerical Controller (CNC) in machine tool applications.

A motion control system may be required to perform a number of tasks, such as moving from one position to another, or tracing a desired geometric path. Examples of the latter task include laser profiling (see Figure 1.2) and autonomous vehicle control. If motion is not required to be synchronised with some external process, the velocity at which the path is traversed may be selected by the motion control system. The task of traversing a specified geometric path is known as path-following control, and forms the subject of this thesis.

Figure 1.2: A laser profiling machine
1.1 Path-following versus tracking

It is important to emphasise the difference between path-following and tracking, which is more widely addressed in the control literature.

Tracking, shown in two dimensions in Figure 1.3(a), requires the machine position to track a time-parameterised reference trajectory \((x^d_k, y^d_k)\), where the superscript \(d\) indicates a desired trajectory, and the subscript \(k\) denotes the time step. That is, the machine is commanded to be at a particular position at each point in time. The tracking controller aims to manipulate the plant input signal such that the system output matches a pre-determined reference trajectory. The axis tracking errors \(e^x\) and \(e^y\) are defined as the deviation between the measured position and the reference trajectory in the \(x\)-and \(y\)-axis directions.

\[
\begin{align*}
\text{(a) Tracking} & : (x_0, y_0) \rightarrow (x_1, y_1) \rightarrow (x_2, y_2) \\
\text{(b) Path-following} & : (x_0, y_0) \rightarrow (x_1, y_1) \rightarrow (x_2, y_2)
\end{align*}
\]

Figure 1.3: Comparison of tracking and path-following

In contrast, path-following only requires the machine position to travel along a desired path, which is not time-parameterised, as shown in Figure 1.3(b). The path-following controller therefore selects the speed at which the system traverses the path, as well as manipulating the control inputs. The contouring error \(\epsilon^c\) is defined as the minimum deviation from the measured position to the path.
1.2 Control objectives

Path-following control involves a number of control objectives. Clearly, it should be ensured that the machine accurately follows the path; in other words, the contouring error should be kept as small as possible. In addition, it is desirable to traverse the path quickly in order to maximise productivity and manufacturing throughput. As a result, minimising the time taken to traverse the path is another objective. Note that these two objectives, accuracy and productivity, are competing with one another: traversing the path quickly is likely to increase the contouring error, and vice versa.

This is illustrated in the corner profile shown in Figure 1.4. Given a system with bounded acceleration, the system is required to come to a complete stop in order to trace the profile exactly. However, if some path deviation is allowed, a higher path speed may be maintained, which also results in shorter traversal times. As the amount of allowable deviation increases, the path speed and hence productivity also increases.

In this thesis, path-following control approaches are developed with respect to these control objectives. Two related, but subtly different, control tasks are examined. The first is the minimum-time path-following problem, where the system is required to traverse the path with no contouring error.
Minimum-time path-following problem. *Given a geometric path and plant model with constraints, determine an input signal such that the system follows the path exactly in minimum time.*

In contrast, following the idea illustrated in Figure 1.4, an alternative problem can be posed where the system is allowed to deviate from the path and a trade-off between accuracy and productivity is considered. In this thesis, this will be referred to as the contouring problem, stated below.

**Contouring problem.** *Given a geometric path and plant model with constraints, determine an input signal that minimises a cost function representing competing objectives of minimising contouring error and traversal time.*

The aim of this thesis is to develop control schemes which address the minimum-time path-following and contouring problems.

### 1.3 Industry partner

This research is supported by ANCA Motion Pty. Ltd., an Australian company specialising in developing control systems for a variety of motion control applications. ANCA Motion’s history stems back to 1974, when its parent company ANCA was created. ANCA designed and built Australia’s first Computer Numerical Controlled (CNC) machine, and has long been at the forefront of CNC technology [3].

In 2008, ANCA Motion was created to focus on the development of control systems for a variety of machine manufacturers, including ANCA. Many of the applications for ANCA Motion control systems, such as CNC machines and laser or water jet profiling, are directly relevant to the minimum-time path-following and contouring problems addressed in this thesis.
1.4 Thesis layout

The remainder of the thesis is laid out as follows. In Chapter 2, existing techniques relevant to the minimum-time path-following and contouring problems are assessed. It is identified that current approaches which explicitly consider control objectives such as minimising traversal time involve offline optimisation of the reference trajectory. As a result, disturbances or modelling errors may lead to poor performance. While closed-loop path-following control schemes exist, these focus mainly on accurate path-following, and do not directly address other control objectives such as traversing the path quickly. Therefore, there is an opportunity for further research in the development of closed-loop optimal path-following and contouring control schemes.

A receding horizon approach to the minimum-time path-following problem is developed in Chapter 3. Since the receding horizon formulation allows feedback to be taken into account, the proposed approach is a closed-loop optimal path-following control scheme. The main contribution of Chapter 3 is the derivation of conditions under which the receding horizon scheme achieves the exact minimum-time solution. More practical approaches using a similar framework are then proposed in subsequent chapters.

In Chapter 4, the receding horizon formulation is used to address the contouring problem for biaxial systems. The developed model predictive contouring control scheme focuses on addressing the trade-off between productivity and accuracy, as well as minimising computational complexity in order to facilitate real-time implementation. The proposed controller is implemented experimentally on an X-Y table in Chapter 5, and results demonstrate how the controller can be tuned to favour productivity over accuracy, or vice versa.

The model predictive contouring control (MPCC) approach is extended to multi-axis systems in Chapter 6, generalising some of the concepts introduced in Chapter 4. Multi-axis contouring requires consideration of both the position and orientation of the end effector, as well as the machine forward kinematics. The desired path and contouring error are expressed in terms of the end effector pose, while the system dynamics and
constraints remain in joint co-ordinates. A multi-rate control architecture is proposed where the MPCC controller generates joint position commands to be tracked by low level joint controllers, in order to facilitate the implementation of the contouring control scheme on industrial hardware.

In Chapter 7, the multi-axis model predictive contouring control approach is applied to a simulation model of an industrial profile cutting machine. The MPCC controller receives measurements of joint position for use as feedback. Simulation results demonstrate how the cost function weights affect contouring behaviour.

Direct measurements of joint position, which are required for the MPCC implementation presented in Chapter 7, are not available with the current hardware of the profile cutting machine. In Chapter 8, alternative implementations of model predictive contouring control are presented, either using only motor feedback or in an entirely open loop fashion, as these approaches require less modification to existing hardware. Simulation results highlight the trade-off between contouring performance and system modification.

Finally, Chapter 9 summarises the contributions of this thesis and highlights potential areas for further research.
In traditional path following systems, the desired path is converted to a time-dependent reference trajectory using an appropriate trajectory planning technique. Trajectory tracking controllers are then employed to track the reference trajectory. The conventional path following control architecture is illustrated in Figure 2.1.

![Figure 2.1: Conventional architecture with separate trajectory planning and tracking](image)

Existing techniques for trajectory planning and tracking are discussed in the following sections.

### 2.1 Optimal trajectory planning

The trajectory planning routine is used to determine the speed at which the system travels along the path, thereby converting the desired path into a reference trajectory. As mentioned in Chapter 1, it is often desirable to optimise the reference trajectory such that the path is completed in minimum time. As a result, a variety of approaches have been proposed with relation to optimal trajectory planning.
The earliest solutions to the minimum-time trajectory planning problem were proposed independently by Bobrow et al. [12], Shin and McKay [104] and Pfeiffer and Johanni [89] for rigid fully-actuated robotic manipulators governed by the following equation of motion:

\[ M(q)\ddot{q} + h(q, \dot{q}) = T, \tag{2.1} \]

where \( q \in \mathbb{R}^n \) denotes the joint positions for an \( n \) degree-of-freedom robot, \( T \in \mathbb{R}^{nu} \) is the vector of input torques, \( M(\cdot) \) is the inertia matrix and \( h(\cdot) \) is the combined effect of Coriolis and viscous friction torques and gravity. The system is subject to joint-dependent actuator constraints, i.e.:

\[ T_{\text{min}}(q, \dot{q}) \leq T \leq T_{\text{max}}(q, \dot{q}). \tag{2.2} \]

The desired path is parameterised by a path parameter \( \theta \in [\theta^s, \theta^f] \), and it is assumed that the joint position, velocity and acceleration can be expressed as functions of \( \theta, \dot{\theta} \) and \( \ddot{\theta} \):

\[
\begin{align*}
q &= q(\theta), \\
\dot{q} &= \dot{q}(\theta, \dot{\theta}), \\
\ddot{q} &= \ddot{q}(\theta, \dot{\theta}, \dddot{\theta}). \tag{2.3}
\end{align*}
\]

By combining (2.1)-(2.3), the torque constraints can be expressed as a constraint on the path acceleration \( \dddot{\theta} \) as a function of \( \theta \) and \( \dot{\theta} \):

\[ \dddot{\theta}_{\text{min}}(\theta, \dot{\theta}) \leq \dddot{\theta} \leq \dddot{\theta}_{\text{max}}(\theta, \dot{\theta}). \tag{2.4} \]

The minimum-time trajectory planning problem can then be interpreted as an optimal control problem with a scalar control input \( \dddot{\theta} \). Bobrow et al. [12] show that the optimal trajectory consists of alternating between minimum and maximum path acceleration, and algorithms are proposed to find the switching points, based on forward and back-
ward integration. It follows from (2.4) that no feasible path acceleration exists when \( \ddot{\theta}_{\text{min}}(\theta, \dot{\theta}) > \ddot{\theta}_{\text{max}}(\theta, \dot{\theta}) \). This corresponds to an implicit constraint on \( \dot{\theta} \) as a function of \( \theta \) which appears as an inadmissible region in the \( \theta-\dot{\theta} \) phase plane. The algorithms proposed by Bobrow et al. [12], Shin and McKay [104] and Pfeiffer and Johanni [89] all rely on the idea of avoiding inadmissible regions, as illustrated in Figure 2.2.

![Figure 2.2: Minimum-time trajectory in the \( \theta-\dot{\theta} \) phase plane](image)

Shiller and Lu [103] show that these solution methods fail at so-called singular points, where at least one actuator does not contribute to the acceleration along the path, and propose a modification of the phase plane approach to remedy this problem. Shiller also derives a necessary condition for the existence of singular points [102].

Finding inadmissible regions in the phase plane involves searching over the whole range of \( \theta \) and \( \dot{\theta} \), which may become computationally expensive. Slotine and Yang [102] propose a more efficient approach by categorising all possible points where the phase plane trajectory just touches, but does not enter, an inadmissible region. These points are denoted characteristic switching points, as the time-optimal trajectory may switch from minimum to maximum path acceleration at these points. An algorithm is proposed to find all of the characteristic switching points by searching once over the range...
of $\theta$. Forward and backward integration from the identified characteristic switching points yields limit curves which are then used to compute the minimum-time solution. It is claimed that the new method improves efficiency by a factor of 50 compared with [12].

Trajectory optimisation methods based only on torque constraints produce potentially discontinuous torque signals, which may be undesirable for the real machine. In [24], Constantinescu and Croft extend the phase plane approach to include torque rate constraints, in order to avoid undesirable discontinuities in the torque signal. It is shown that torque rate constraints can be expressed as state-dependent constraints on the pseudo-jerk, defined as the third derivative of the path parameter:

$$
\dddot{\theta}_{\text{min}}(\theta, \dot{\theta}, \ddot{\theta}) \leq \dddot{\theta} \leq \dddot{\theta}_{\text{max}}(\theta, \dot{\theta}, \ddot{\theta}).
$$

(2.5)

The control input then becomes the pseudo-jerk $\dddot{\theta}$, rather than $\dot{\theta}$ which is the case for the earlier phase plane approaches. An optimisation problem is posed by parameterising trajectories in the $\theta$-$\dot{\theta}$ phase plane trajectory with cubic splines, and solved using the flexible tolerance method. In the flexible tolerance method, a function $\mathcal{F}()$ is defined that is zero when the constraints are satisfied and positive otherwise. The solution is then obtained by iteratively solving a simpler optimisation problem with a single constraint on $\mathcal{F}()$.

Phase plane approaches rely on the reduction of the optimal trajectory planning problem to a much simpler problem defined only in terms of the path parameter $\theta$ and its derivatives. The approaches discussed previously are restricted to rigid robotic manipulators with equation of motion (2.1). Faulwasser et al. [46] show that this reduction is possible for any differentially flat system, and take advantage of the simplified problem to investigate the existence of solutions to the trajectory planning problem.

The techniques discussed thus far focus only on minimising traversal time, and do not consider other control objectives such as minimising control effort. A dynamic programming (DP) approach to optimal trajectory planning problems was proposed
by Vukobratović and Kirčanski [125] with a cost function of the form:

$$J = \int_{t_0}^{t_f} L(x_t, u_t) dt + g(x_{t_f}),$$  \hspace{1cm} (2.6)$$

where $t_0$, $t_f$ are the start and end times and $L(\cdot), g(\cdot)$ are arbitrary stage and terminal costs. The continuous-time problem is discretised with a sample interval $\delta t$, allowing for an optimisation problem with a finite number of decision variables to be posed. The problem is solved using dynamic programming techniques, iterating backwards in time from the final state.

Shin and McKay [105] and Pfeiffer and Johanni [89] also propose dynamic programming approaches where instead of using time discretisation as in [125], the $\theta$-\$ phase plane is divided into a discrete grid. The cost function is therefore expressed in $\theta$ coordinates, rather than time. Shin and McKay also show that the DP solution converges to the true minimum-time solution as the grid size decreases. Another DP approach proposed by Singh and Leu [107] is applicable to paths described by a sequence of points, rather than parameterised paths. An advantage of the dynamic programming approaches is that they can be applied not only to minimum-time problems, but also to problems including other performance criteria, such as control effort or energy. However, DP solution methods have a high computational cost.

The problem of trajectory planning has also been studied using optimal control theory. Chen et al. [21] use the extended Pontryagin’s Minimum Principle to describe the general structure of time-optimal solutions. It is shown that at least one actuator is at its limit in every finite time interval along the trajectory. Shiller [100] uses Pontryagin’s Maximum Principle to obtain a solution to the time-energy optimal path following problem, where the cost function to be minimised includes a term penalising control effort and $T$, as before, is the vector of input torques:

$$J = \int_{t_0}^{t_f} \left(1 + \epsilon^2 T^2\right) dt,$$  \hspace{1cm} (2.7)$$

where $\epsilon$ is a tuning parameter determining the emphasis placed on minimising control
effort. The solution is computed by solving a two point boundary value problem, which arises from the application of Pontryagin’s Maximum Principle. It is shown that the resulting control inputs are smooth, avoiding problems associated with discontinuous input signals produced by the time-optimal solutions. As expected, the time-energy optimal trajectory approaches the minimum-time trajectory as $\epsilon$ approaches zero. As with dynamic programming approaches, finding the optimal solution using this approach involves a large computational burden.

In a different approach proposed in the robotics literature, Verscheure et al. [122] reformulate the optimal path-following problem as a convex second-order cone program (SOCP), resulting in efficient calculation of the optimal solution. An online version based on log-barrier solution methods is also proposed [123, 124] for situations where the desired path is not completely known a priori. Other objectives such as thermal energy may be traded off against time-optimality. However, a main drawback of this approach is that viscous friction must be neglected in order to reformulate the optimisation as a convex problem.

In the manufacturing literature, Renton and Elbestawi [93] propose a computationally efficient two-pass algorithm, entitled Minimum-Time Path Optimisation (MPTO), to calculate the reference trajectory based on a kinematic system model. Renton and Elbestawi also show that the velocity and acceleration limits of a motor are not constant, but in fact depend on the instantaneous velocity and acceleration [93]. The proposed MPTO utilises the full acceleration and velocity envelope, allowing for faster feed rates compared to algorithms that specified constant acceleration and velocity limits. However, as a kinematic system model is used, the dynamics of the system are not considered. The two-pass algorithm is extended to include contour error and bandwidth constraints by Dong and Stori [33, 34], with a dynamic system model. Dong and Stori also provide a formal optimality proof using phase plane analysis. Jerk constraints are later incorporated in [35].

Imamura and Kaufman [55] introduce the idea of simultaneously optimising the reference trajectory and feedback controller gains subject to contour error, current and
2.1 Optimal trajectory planning

voltage constraints. Linear Quadratic Regulator (LQR) feedback controllers were used. The optimisation is completed offline, and is conducted in two phases. First, the feed rate is optimised to minimise tracking time. Then, controller gains are tuned via minimisation of an objective function. The procedure is then repeated until an optimum tracking time is reached. Since the optimisation problem is not convex, it is unlikely a global optimum will be found by this algorithm [34]. Imamura and Kaufman also point out that there is no systematic way of selecting the cost function to be used in the controller tuning phase of the optimisation.

In another trajectory planning approach, spline fitting techniques are used to smooth out sharp corners in the contouring path in [43]. The cornering feed rate is then tuned manually using a simulation model of the closed-loop system such that the corner is traversed in minimum time while achieving a desired level of accuracy. Huang and Tomizuka propose a self-paced fuzzy tracking controller for biaxial contouring which adjusts the speed of the reference trajectory depending on the curvature of the path and its rate of change [54]. Based on a set of pre-determined rules, the controller slows down as the curvature of the path increases and speeds up along straight sections.

It is important to note that in all of the trajectory planning approaches discussed thus far, the reference trajectory is calculated in an open loop fashion, relying on feedback controllers to achieve the desired motion. In addition, if the reference trajectory is time-optimal, at least one actuator constraint is active at all times [21]. Therefore, in the presence of disturbances or modelling errors, tracking performance could severely deteriorate, or constraint violation could occur.

Several researchers have attempted to address this problem. Shin and McKay [106] propose a trajectory planning algorithm that is robust to payload uncertainties. Assuming a particular bound on the amount of uncertainty, the trajectory is planned for the worst case. Clearly, the traversal time is increased in comparison to the nominal minimum-time trajectory, but with the benefit of a robustness guarantee. Trajectory preshaping is used to compensate for friction, motor dynamics and the dynamics of the tracking controller in [101]. The nominal time-optimal reference trajectory is modified using the
inverse of the plant model and controller so that in closed-loop, the desired behaviour is achieved. Kieffer et al. [59] experimentally identify disturbances for a particular path and then use the information to plan a robust near time-optimal reference trajectory.

The reference governor is a nonlinear device that modifies the reference trajectory online in order to avoid constraint violation [39]. Consider a linear discrete-time model of the closed-loop plant

\[ x_{k+1} = \Phi x_k + G g_k, \quad (2.8) \]

where \( g_k \) is the input to the closed-loop system, which would be equal to the nominal reference trajectory \( r_k \) in the absence of constraints, and \( x_k \) is the state. The system constraints are expressed as

\[ c_k = H_c x_k + D g_k \in C, \quad (2.9) \]

where \( c_k \) is the vector of quantities that must lie in the set \( C \). The reference governor selects \( g_k \) with the aim of being as close as possible to \( r_k \) while still honouring the constraints (2.9). The general architecture of the reference governor approach is given in Figure 2.3.

![Reference governor architecture](image)

The mechanism by which \( g_k \) is selected varies among authors. Gilbert et al. [50] select \( g_k \) to be the output of a low-pass filter with state-dependent bandwidth. Bemporad et al. [6] solve a constrained optimisation problem at each time step. The optimisation selects \( g_k \) to minimise a quadratic cost function penalising the difference between the
nominal and modified reference trajectories, subject to the constraints (2.9) over a finite time horizon. It is shown that there exists a horizon length, which can be determined \textit{a priori}, such that satisfaction of the constraints is guaranteed for all time. This approach is applied to an inverted pendulum in [19]. Note that Bemporad \textit{et al.} initially call their approach a “command governor” rather than reference governor, however the terminology becomes consistent in Bemporad’s more recent work [2]. Reference governors have also been extended to handle nonlinear systems [2,19].

The reference governor adjusts only the reference position to avoid constraint violation. The time parameterisation of the trajectory remains unchanged, and poor tracking performance with respect to the original path can occur. In the machine tool literature, Susanu and Dumur [114] combine the reference governor approach with velocity adaptation, which regenerates a new, slower trajectory when the reference governor adjusts the trajectory so much that unacceptable tracking errors occur. This is implemented in a hierarchical structure, as shown in Figure 2.3. The trajectory tracking controller is at the lowest level, with the trajectory supervisor, which acts as a reference governor, and velocity adaptation residing in the upper levels.

Dahl and Nielson [26] adjust the speed of the reference trajectory online using trajectory time scaling. The path acceleration and velocity constraints are calculated online, and if violation of these constraints is detected, a time scaling factor is applied to slow the reference trajectory down. Canudas de Wit and Roskam apply the time scaling approach to a two-degree-of-freedom wheeled mobile robot [18]. An extension to include torque rate constraints is provided by Gerelli and Guarino Lo Bianco [18]. A similar scheme is proposed by Kieffer \textit{et al.} [59] where the path acceleration is governed by a feedback control law which is designed to compensate for worst-case disturbances. Kieffer \textit{et al.} also provide a methodology for choosing trajectory tracking controller gains to achieve accuracy within a prescribed tolerance. These approaches all require a pre-determined reference trajectory, and only modify the reference to reduce performance deterioration in the presence of disturbances.

The path governor, introduced by Bemporad \textit{et al.} [10], generates the reference trajec-
Figure 2.4: Hierarchical control structure with reference governor and velocity adaptation online and utilises feedback at every time step, as shown in Figure 2.5. Therefore, a pre-determined reference trajectory is not required. A constrained scalar optimisation problem is solved at time intervals of length $\Delta t$. A virtual time parameterisation of the path parameter $\theta$ is proposed of the following form:

$$\theta(\tau, k\Delta t, \theta_\infty) = \theta_\infty + [\theta_{k\Delta t} - \theta_\infty]e^{\alpha\tau},$$

(2.10)

where $\tau \in [0, \infty)$ denotes the prediction time, $k\Delta t$ the current time, $\alpha$ a fixed parameter and $\theta_\infty$ a free scalar to be determined. The aim is to select $\theta_\infty$ in order to maximise

$$J(\theta_\infty) = \theta(\Delta t, k\Delta t, \theta_\infty)$$

(2.11)

while ensuring that the constraints of the system are satisfied. The optimised virtual parameterisation is used to generate the reference trajectory for the time interval $[k\Delta t, (k+1)\Delta t]$, after which the optimisation is repeated.
Maximising $\theta(\Delta t, k\Delta t, \theta_{\infty})$ corresponds to traversing as far along the path as possible in the interval $[k\Delta t, (k+1)\Delta t]$, and hence minimising traversal time. However, because of the structure of the virtual parameterisation (2.10), the resulting reference trajectory is at best near time-optimal.

Regardless of which trajectory planning algorithm is used, it is necessary to use feedback controllers to track the reference trajectory accurately. The following section examines the problem of trajectory tracking and some approaches documented in the literature.

### 2.2 Trajectory tracking

There is a vast array of trajectory tracking approaches documented in the literature. A brief discussion of a selection of approaches is provided here, beginning with the current industry standard approach.

#### 2.2.1 Cascaded PI control

By far the most widely used control strategy for motion control drives is the cascaded PI architecture, consisting of nested current, velocity and position control loops. The cascaded PI structure is shown in Figure 2.6.

PI control offers simplicity of controller design with little or no knowledge of the system dynamics. Derivative action can be added to form proportional-integral-derivative
(PID) control. Given a plant that can be described by a linear transfer function of low order, the closed-loop poles can be placed arbitrarily using a single (not cascaded) PID controller [23]. However, performance can degrade when faced with more complicated systems and actuator saturation. Changing plant dynamics due to different loading conditions and mechanical wear can also increase tracking errors [41]. As a result, a wide range of alternative control approaches have been attempted, only a few of which are discussed below.

2.2.2 Inverse-model-based feedforward control

As the reference trajectory is usually known ahead of time, the desired outputs several time steps into the future are known to the controller. As a result, feedforward control can be used in addition to feedback control to improve tracking performance. Inverse-Model-Based Feedforward (IFF) control uses knowledge of future set points to compensate for lag in the closed-loop dynamics. Ko discusses a variety of IFF controllers and associated implementation issues in [61].

Zero phase error tracking control, proposed by Tomizuka [117], works on the principle of pole-zero cancellation to achieve perfect tracking. Since the zeros of the plant to be controlled become the poles of the controller, any zeros on or outside the unit circle on the z-plane cannot be cancelled out by the controller, as this would result in the controller being open loop unstable. Tomizuka shows that perfect tracking cannot be achieved in the presence of these zeros.
To remedy this, the ZPETC algorithm seeks to cancel out only the zeros that lie within the unit circle. To handle any remaining zeros, the controller is designed such that the phase error introduced by these zeros is completely eliminated. The overall ZPETC scheme is shown in Figure 2.7. The input to the ZPETC feedforward controller is the \((d+s)\)-step ahead commanded output \(y_k^d\) where \(d\) is the number of delay steps in the closed-loop system and \(s\) is the number of closed-loop zeros on or outside the unit circle. The feedforward controller generates a reference signal \(r_k\) based on the desired future output and a model of the closed-loop system. The reference \(r_k\) is then applied to the closed-loop system consisting of the feedback controller and plant.

![ZPETC control scheme](image)

Figure 2.7: ZPETC control scheme

Tomizuka demonstrates successful application of ZPETC in machine tool control in [116]. The ZPETC scheme has also been combined with a number of other control techniques such as disturbance observers [40, 71] and friction compensation [71, 115]. Torfs et al. propose extended bandwidth zero phase error tracking control (EBZPETC), introducing additional feedforward terms to further reduce tracking error [119].

Ko proposes Inverse-Model-Based Velocity Feedforward Control (IFFV), which combines velocity feedforward control with inverse-model-based feedforward control [61]. In an IFFV scheme, the velocity feedforward gain is replaced by the inverse of the velocity loop. Benefits of the IFFV controller include zero steady-state error for ramped inputs and improved robustness compared to IFF control.

Inverse-model-based feedforward control schemes can achieve excellent tracking performance with a high overall bandwidth [61]. However, they require knowledge of the closed-loop system dynamics, and their performance is sensitive to plant parameter
variations. An adaptive version of ZPETC was put forward by Tsao and Tomizuka in [121] to address the issue of parameter variation. In addition, the inverse transfer function approach has a tendency to cause large control signals [63], and makes no attempt to account for actuator saturation.

2.2.3 Contouring error control

The trajectory tracking techniques discussed thus far focus on reducing tracking error, defined as the difference between the reference and measured output at time $k$. However, as mentioned earlier, in path following or contouring applications it is not necessary to track the time parameterised reference trajectory accurately, as long as the system follows the desired path. In such applications the contouring error, which is the deviation from the desired contouring path, is much more important than the tracking error. Figure 2.8 illustrates the difference between the two metrics for a two-axis contouring application. The point $R$ represents the value of the reference at the current time, while $P$ is the actual tool position. The point $A$ represents the point on the desired path closest to $P$. The tracking errors in the $x$ and $y$ directions are measured from the point $R$, while the $x$ and $y$ components of the contouring error are measured from the point $A$.

Cross coupled control (CCC), introduced by Koren [62], seeks to minimise contouring errors instead of tracking errors. This is accomplished by estimating $\epsilon_x$ and $\epsilon_y$ based on the tracking error of the $x$ and $y$ axes $e_x$ and $e_y$ using a contouring error model. A feedback control law is designed to minimise these contouring errors, and the output of the contouring error controller is added to the control inputs of the individual axes. A block diagram of a CCC scheme is shown in Figure 2.9. $G_x$ and $G_y$ represent the feedback control laws acting to minimise contouring errors $\epsilon_x$ and $\epsilon_y$. The contour error controllers and individual X and Y axis controllers can be implemented using any feedback control strategy. It should be emphasised that the cross coupled control approach is designed to complement an existing control architecture, not to replace it.

Koren and Lo conducted comparative experiments to test the performance of PID,
ZPETC and CCC controllers [63]. The cross-coupled controller was implemented with a proportional control law. Results of the evaluation are summarised in Table 2.1. However, it should be noted that Table 2.1 shows the performance of ZPETC with only a proportional feedback controller. It is likely that improved performance would be achieved if ZPETC were to be implemented in conjunction with a disturbance observer and friction compensation, such as in [71]. Yao et al. show that the combination of ZPETC and adaptive robust control (proposed by Yao and Tomizuka in [128]) is capable of achieving tracking errors of around 2 microns [127].

Several other control approaches aim to minimise contouring error rather than only tracking error. McNab and Tsao [83] implement receding horizon linear quadratic control using a cost function penalising contouring error as well as tracking error and control effort. It is shown that the contour becomes more accurate as the penalty weighting assigned to contouring error is increased. A similar approach using model predictive control is proposed by Khalick and Uchiyama [58], where separate weighting factors are applied to the errors orthogonal and tangential to the desired path. Improved contouring performance is achieved by increasing the penalty on orthogonal errors.

Another approach to contouring error control is proposed by Chiu and Tomizuka [22], where contouring control is performed in a transformed co-ordinate frame by repre-
senting the error dynamics in tangential and normal components. The controller can then focus on reducing the normal (contouring) component of the error. Chen and Wu [20] propose a contouring control scheme based on equivalent errors, where instead of approximating the contouring error, an equivalent error is formulated and it is shown that reducing the equivalent error results in a reduction of contouring error.

Contouring error controllers attempt to address the path following control problem indirectly by placing more emphasis on deviation from the path rather than tracking error. However, these approaches are still fundamentally trajectory tracking control schemes, with no ability to affect the path evolution. Another trajectory tracking control approach is model predictive control, discussed in the following Section.

2.2.4 Model predictive control

Model Predictive Control, or MPC, is an advanced control strategy which has been gaining interest over the last 25 years. A survey by Qin and Badgwell [91] shows that the number of MPC applications doubled in the years from 1995 to 1999. According to Maciejowski [79], MPC is the only advanced control technique to have made a signifi-


\[ x_{k+1} = f(x_k, u_k) \]
\[ y_k = h(x_k, u_k). \]  

(2.12)

In tracking applications, MPC seeks to minimise the following cost function:

\[ J(k) = \sum_{i=1}^{H_p} ||y_{k+i,k} - y^d_{k+i,k}||_Q^2 + \sum_{i=1}^{H_u} ||\Delta u_{k+i-1,k}||_R^2, \]  

(2.13)

where

- \( y_{k+i,k} \) is the predicted output,
- $y_{k+i,k}^d$ is the reference trajectory,
- $\Delta u_{k+i,k}$ is the predicted change in input,
- $H_p$ and $H_u$ are the prediction and control horizons,
- $Q$ and $R$ are weighting matrices, and
- $||e||_Q^2$ represents the quadratic form $e^T Q e$.

Constraints can be explicitly considered in the controller by minimising (2.13) subject to input and state constraints. Explicit constraint management is a key advantage of the model predictive control formulation. The solution to the optimisation problem is a sequence of control inputs $u_{k+i,k}$. The first control input $u_{k,k}$ is then applied to the plant, and the optimisation is repeated in the next time step, in what is often referred to as a Receding Horizon (RH) fashion. The number of time steps ahead used for prediction is known as the prediction horizon, while the control horizon specifies the number of optimised future control inputs.

In minimising the cost function (2.13), the control signal seeks to fulfil two possibly conflicting control objectives. First, in order to minimise tracking error, the difference between desired and predicted outputs should be minimised. Since the predicted reference trajectory $y_{k+i,k}^d$ appears in the cost function, advance knowledge of the reference trajectory can be utilised. A penalty is also assigned to the change in control input across the horizon, in order to enforce smooth control signals. The choice of the weighting matrices $Q$ and $R$ determine the relative importance of these two objectives. It is also possible to include different objectives in the cost function; for example control effort can be penalised instead of, or as well as, control deviations.

If the plant is linear and there are no constraints, the minimisation of the cost function (2.13) has a closed form solution, and the optimal control sequence can be computed using matrix inversion. If constraints are present, the optimal control sequence for a linear system is the solution to a convex quadratic program (QP). If the system model is nonlinear, the optimisation problem becomes a general nonlinear, non-convex problem which is considerably more difficult to solve efficiently.

Model predictive control has been applied to trajectory tracking by several researchers.
In the machine tool literature, Boucher et al. implement predictive control for machine tool drives by combining the cascaded structure of conventional machine tool control with generalised predictive control (GPC), a variant of MPC \[14\]. The generalised predictive cascade control (GPCC) architecture is shown in Figure 2.10.

This idea is extended in \[13\], where the multirate architecture of the cascaded control system is explicitly accounted for in the controllers. An advantage of the multirate structure is that it takes advantage of the faster dynamics of the inner loop, allowing faster sampling of the inner loop without oversampling the external loop.

Further extensions of Boucher et al.’s work include adaptive control to handle changing plant parameters such as variations of inertia in flexible transmissions \[37\], GPC with terminal equality constraints \[36\] and robust control \[15\]. Maaziz et al. implement GPC for the speed and flux control of an induction motor with a nonlinear plant model \[78\]. Feedback linearisation is applied to obtain linear models to be used by the GPC controller. More recent work by Dumur et al. \[38\] has focussed on the complete GPC design procedure, from machine modelling and system identification to controller design, simulation, and finally implementation on the CNC. It is demonstrated that predictive control is a viable option for industrial machining axes.

A key advantage of MPC is that actuator constraints can be considered explicitly. However, Boucher et al. implement only unconstrained GPC. In 1988, Tsang and Clarke extended GPC to account for constraints \[120\]. A quadratic programming approach to solving the constrained optimization problem was discussed, and at the time, found to be too computationally intense for machine tool applications. However, considering the advances in computational power in the last twenty years, this may no longer be the case.
Del Re et al. [31] implement model predictive control in simulation on a two-axis machine using a linear discrete-time state space model with input constraints. Results demonstrate that the model predictive controller performs better than cascaded PID and pole placement controllers when tracking a corner profile. However, there is no mention in [31] of any measures taken to reduce the computational load of constrained MPC, and it is unclear if the proposed controller is computationally efficient enough to be implemented on a real machine.

Real-time implementation of model predictive control, particularly with constraints, is restricted by its computational burden. Explicit MPC, introduced by Bemporad et al. [9], seeks to reduce the online computation requirements of MPC by solving the optimisation problem offline, and using a lookup table to determine the correct control input online. The state space is divided into a number of regions and it is shown that within each region, the optimal control can be calculated using a simple state feedback law. The regions and state feedback controllers are computed offline, so that the online computation is reduced to identifying the correct region from the current state and then evaluating the corresponding control law.

If there is a large number of regions, the computational burden of searching for the correct region online may become too high for practical implementation. Furthermore, storage requirements also increase with the number of regions. Unfortunately, the number of regions grows rapidly with the control horizon. In addition, if advanced knowledge of the reference trajectory over the prediction horizon is to be used in the optimisation, the number of regions grows so much as to make implementation of explicit MPC on fast systems impractical [47]. Stephens et al. [113] reduce the number of regions associated with tracking MPC by introducing a trajectory horizon into the cost function:

$$J(k) = \sum_{i=1}^{H_t-2} \| y_{k+i,k} - y_{k+i,k}^d \|^2_Q + \sum_{i=H_t-1}^{H_p} \| y_{k+i,k} - \hat{y}_{k+i,k}^d \|^2_Q + \sum_{i=1}^{H_u} \| \Delta u_{k+i-1,k} \|^2_R,$$  \hspace{1cm} (2.14)

where $H_t$ is the trajectory horizon and $\hat{y}_{k+i,k}^d$ is the extended reference trajectory. The extended reference trajectory is calculated using a first-order hold approximation of
the reference trajectory at the end of the trajectory horizon:

\[
\hat{y}_{k+i,k}^d = \hat{y}_{k+H_{t-1},k}^d + (i - H_t + 1)(\hat{y}_{H_{t},k}^d - \hat{y}_{H_{t-1},k}^d),
\]

\[i \in \{H_{t-1}, \ldots, H_p\}.\] (2.15)

Effectively, (2.15) assumes that the velocity of the reference trajectory remains constant from the end of the trajectory horizon to the end of the prediction horizon. It is shown that this assumption works well in the context of machine tools, which typically follow smooth trajectories. The trajectory horizon can be made significantly shorter than the prediction horizon, reducing the number of regions required for explicit MPC. The optimal control laws are stored in a binary search tree, as proposed in [118], which is then searched online to find the control input. Using this approach, Stephens et al. are able to implement tracking MPC on an industrial machine tool with a sample period of 500 µs, and demonstrate a significant improvement in tracking accuracy over cascaded PI control.

Model predictive control has also been applied to trajectory tracking of mobile robots, where a prediction model of the tracking error is used to design the controller. The error dynamics model for wheeled robots is nonlinear. This leads to a nonlinear optimisation problem being posed at each sample, which is impractical for online implementation. To overcome this problem, Kuhne et al. [65] use successive linearisation of the error model around the reference trajectory and then apply linear MPC, reducing the computational load. Similar approaches are proposed by Gu and Hu [52] and Klančar and Škrjanc [60]. Akiba et al. [3] point out that linearisation around the reference trajectory is only valid if the robot is sufficiently close to the reference. They propose a piecewise affine model involving linearisation around a number of equilibrium points.

As mentioned earlier, trajectory tracking approaches focus only on reducing tracking or contouring error with a fixed reference trajectory. Path following is achieved by combining a trajectory planning scheme with a trajectory tracking controller, as shown in Figure 2.1. While this technique of specifying a reference trajectory to be tracked by feedback controllers can simplify the design process, the separation between the opti-
misation routine and the inputs that are actually applied to the plant can affect system performance, particularly if disturbances arise. While some methods of adjusting the reference trajectory online to avoid such problems have been discussed, there is a benefit in using a single controller to achieve path following directly, without the need for a separate trajectory planning scheme.

### 2.3 Path-following control

In contrast to the trajectory planning/tracking approaches discussed in Sections 2.1 and 2.2, path-following controllers address the path-following control problem directly. Such controllers determine the evolution of the path and the plant inputs simultaneously using available feedback. A general architecture for path following control is provided in Figure 2.11.

![Path-following control architecture](image)

Figure 2.11: Path-following control architecture

Path-following control arose from the unfortunate fact that even in the absence of constraints, non-minimum phase systems are incapable of exact trajectory tracking. Qiu and Davison [92] show that there is a fundamental lower bound on the $L_2$-norm of the tracking error for such systems. The same is true for nonlinear systems with unstable zero dynamics [98]. Aguiar et al. show that this performance limitation is removed if path following control is considered, rather than trajectory tracking, for both linear [2] and nonlinear systems [11].

Dačić and Kokotović [27] propose a path-following controller for linear non-minimum phase systems where a feedback law is designed for the path parameter $\theta$ to stabilise the zero dynamics. Another feedback law is then used to determine the plant input...
which drives the system output along the path. For a linear system with state vector $x$, output $y$ and desired path function $y^d(\theta)$, the proposed controller is shown to achieve the following objectives:

1. Asymptotic convergence to the path: $y_t \to y^d(\theta_t)$ as $t \to \infty$,
2. Forward motion along the path,
3. Boundedness of the state.

Dačić et al. [28] extend the approach to nonlinear systems with unstable zero dynamics, where only practical convergence to the path is guaranteed: given any $\epsilon > 0$, a path following controller can be designed such that $\limsup_{t \to \infty} ||y_t - y^d(\theta_t)|| \leq \epsilon$. Hence, path following to a specified accuracy is achieved. Subsequently, Dačić et al. [29] propose a path-following control design based on averaging for problems where the path is periodic in the path parameter.

Hauser and Hindman [53] propose a method to convert trajectory tracking controllers to path-following controllers for feedback-linearisable systems using an appropriate mapping from the current state to the path parameter $\theta$. Consider a feedback-linearised system with state $x$ and input $u$:

$$\dot{x} = Ax + Bu, \quad (2.16)$$

and a trajectory tracking control law:

$$u_t = \tilde{\beta}(x_t, \zeta_t, \nu_t) := \nu_t + K(x_t - \zeta_t) \quad (2.17)$$

where $(\zeta_t, \nu_t)$ is a time parameterised reference trajectory for $(x, u)$ and $A_c = A + BK$ is Hurwitz. A mapping is defined from the state $x$ to the path parameter,

$$\pi(x) = \argmin_{\theta} ||x - \zeta(\theta)||^2_P, \quad (2.18)$$

where $P$ is positive definite. The mapping $\pi(x)$ can be interpreted as the value of $\theta$ corresponding to the closest point on the path to $x$ with respect to the norm defined by
the matrix $P$. The path-following control law is obtained by replacing $t$ with $\pi(x_t)$ in (2.17):

$$u_t = \bar{\beta}(x_t, \zeta(\pi(x_t)), \nu(\pi(x_t)))$$

so that a time-parameterised reference trajectory is no longer required. It is shown that the converted control law (2.19) provides exponentially stable path following. Hauser and Hindman’s approach has subsequently been applied to flight control problems [4] and motorcycle control [94].

Skjetne et al. [108] reformulate the path-following problem into two tasks: a geometric task, where the system is driven towards the path, and a dynamic task, where the path speed is made to converge to some desired function. A recursive backstepping technique is used to satisfy the geometric task. Then, an update law is designed to achieve the required speed assignment. The control approach is shown to achieve closed-loop input-to-state stability.

Path-following control schemes also arise in the autonomous vehicle literature. An early approach proposed by Samson [95] for wheeled mobile robots relies on computing a projection of the vehicle onto the desired path, similar to the mapping used in [53]. The technique was extended to chained systems with application to a car pulling a series of trailers [96] and autonomous underwater vehicles [39]. A drawback of these projection-based methods is that the initial position of the vehicle must be sufficiently close to the path to ensure uniqueness of the projection. Soentanto et al. [110] overcome this issue by introducing the concept of a virtual vehicle to be tracked along the path. The path-following error is then defined as the distance between the real and virtual vehicles, and the speed of the virtual vehicle is treated as an additional control input. The technique has been applied to control of autonomous underwater vehicles [69, 70] and unmanned surface vehicles [111].

Velocity field control [74] is another, quite different approach to path following control. A time invariant velocity field is created in the output space, so that for each output location $q$ there is a corresponding desired velocity $V^d(q)$ which drives the system to-
Path-following control

wards, or along, the desired path. The system velocity is then controlled to be a scalar multiple of the desired velocity, i.e. \( \dot{q}_t = \alpha V^d(q_t) \). A velocity field for a circular path in two dimensions is shown in Figure 2.12. A similar approach is proposed by Nelson et al. for path following for miniature air vehicles. Here, vector fields around straight line and orbital paths are used to direct the vehicle towards the desired path.

Passive velocity field control (PVFC), proposed by Li and Horowitz, additionally requires that passivity of the closed-loop system be maintained. This guarantees that interaction between the control system and passive environments is stable. Clearly, velocity field control relies on the ability to create velocity fields for the desired path. Li and Horowitz provide a methodology for generating velocity fields for parameterised curves in [77]. Velocity field control has been applied to the control of robot manipulators [75], mobile robots [32] and machine tools [74]. An adaptive version is given in [73], where PVFC is extended to applications with unknown inertia parameters.

The path-following control approaches discussed thus far do not consider any constraints on the system. In addition, these control schemes do not directly address the
control objectives discussed in Chapter [1] but rather focus on driving the output towards the path and ensuring forward motion. Kanjanawanishkul et al. [56] combine path-following control with trajectory tracking control for mobile robots using nonlinear model predictive control. A nonlinear system describing the error dynamics is formulated where the velocity of the virtual vehicle to be followed by the robot is included in the control input vector. In addition, a term penalising the error between the robot state and a time-parameterised reference is included in the cost function, which has the effect of encouraging time convergence in addition to path following. The MPC framework allows for constraints to be taken into account. Simulation and experimental results demonstrate that using an appropriate combination of path following and trajectory tracking, the robot converges to the desired path in a smooth manner and then acts to reduce trajectory tracking error. However, experiments were conducted with a relatively long (0.1 s) sample period, perhaps due to the high computational load of nonlinear MPC.

Faulwasser et al. [45] propose model predictive path-following control (MPFC) for general nonlinear continuous time systems. Following a model predictive control framework, a cost function is minimised, subject to constraints, at each sample. A timing law $g(\cdot)$ is introduced for the path parameter, driven by a virtual input $v$:

$$\dot{\theta}_t = g(\theta_t, v_t), \quad v_t \in \mathcal{V}. \quad (2.20)$$

In contrast to tracking MPC, model predictive path-following control determines the plant inputs and path parameter evolution simultaneously. The MPFC cost function contains a stage cost $F(\cdot)$ as a general function of the state, path parameter, plant inputs and virtual input, as well as a terminal cost $E(\cdot)$ used to guarantee closed-loop stability:

$$J = \int_{t_k}^{t_k+H_p} F(\bar{x}_\tau, \bar{u}_\tau, \theta_\tau, v_\tau) \, d\tau + E(\bar{x}_{t_k+H_p}, \theta_{t_k+H_p}), \quad (2.21)$$

where $\bar{x}$ and $\bar{u}$ are the predicted states and inputs, $t_k$ is the current sampling instant and $H_p$ is the prediction horizon.
The MPFC framework offers an opportunity to include specific control objectives in the stage cost $F(\cdot)$. However, Faulwasser et al. do not discuss the effect of the choice of cost function on the behaviour of the controller, instead focusing on achieving accurate and stable path following. Furthermore, the controller is implemented in simulation only, with no attempt made to reduce the computational burden of nonlinear MPC.

2.4 Conclusion

The extra degree of freedom afforded by the path-following control problem presents an opportunity for performance improvement. However, optimal path-following control in the presence of constraints is a difficult task. Control schemes based on optimal open loop reference trajectories risk performance deterioration and constraint violation if disturbances or modelling errors arise. While online path following controllers have the ability to reject disturbances, the path following control schemes examined in the literature focus on driving tracking error to zero, and do not consider other control objectives such as traversing the path quickly.

This review has highlighted an opportunity for further research in developing online optimal control schemes for both minimum-time path-following and contouring control. Combining the ideas behind trajectory planning and path-following control, this approach has the potential to reject disturbances while also attempting to achieve the designated control objectives.

The recently proposed model predictive path-following control approach [45] involves the minimisation of a cost function over a finite horizon. As a receding horizon structure is used, feedback may be taken into account at each time step, while the cost function may be selected to reflect desired control objectives such as minimising traversal time or contouring error.
2.5 Research aims

The following research aims are proposed:

1. Apply and extend the MPFC framework to minimum-time path-following and contouring problems.
2. Investigate the practical implementation of the proposed control schemes on existing state-of-the-art hardware.

2.6 Research plan

A strategy for addressing the research aims established in this Chapter is outlined in the following. First, the MPFC framework will be applied to minimum-time path-following, and a theoretical investigation of the time optimality properties of the control scheme will be conducted. The application of receding horizon schemes to contouring problems will then be investigated, with an emphasis on practical implementation.

In biaxial contouring, the contouring error may be estimated from the axis tracking errors in a relatively straightforward manner, allowing contouring error control approaches such as [63] to focus on minimising contouring error. Therefore, it is proposed to initially adapt the receding horizon framework to biaxial contouring problems, taking advantage of the contouring error estimation techniques in the literature.

In addition, as a biaxial laboratory test rig is available for use, the developed biaxial contouring control approach may be implemented experimentally, in contrast to [45], where only simulation results are presented. Subsequently, the approach will be extended to more complicated multi-axis systems, with particular emphasis on industrial applications.

The research plan is summarised below:

1. Develop a receding horizon path-following control scheme with the intention of addressing the minimum-time path-following problem
2. Identify conditions under which the receding horizon approach generates the time-optimal solution.

3. Within a similar framework, relax the exact path-following constraint and develop a control approach for biaxial systems where contouring control objectives of accuracy and productivity are reflected in the finite horizon cost function.

4. Implement the developed model predictive contouring control approach in real-time and demonstrate how the control objectives are addressed by the cost function through appropriate experiments.

5. Generalise the model predictive contouring control algorithm to more complicated multi-axis systems with a view towards industrial implementation.

6. Apply the multi-axis contouring control scheme to a simulation model of an industrial contouring machine.
Chapter 3
Receding Horizon
Time-Optimal Control

A receding horizon approach to the minimum-time path-following problem is proposed, where it is desired to traverse along the path exactly as quickly as possible, in order to maximise productivity. The finite horizon cost function and constraints are selected so that the system advances as far as possible along the path within the horizon, in contrast to [15] which focuses only on accurate path-following. Unlike offline time-optimal control approaches, the receding horizon strategy utilises available feedback information at each sample, and hence has the potential to reject disturbances.

The conditions under which the proposed formulation reproduces the solution to the minimum-time path-following problem are established. It is shown that for a class of differentially flat systems, time optimality is achieved when the horizon is sufficiently long. Moreover, the proposed formulation guarantees exact path-following irrespective of horizon length. Simulations conducted with a rigid X-Y table model verify the theoretical results.
3.1 The minimum-time path-following problem

Consider a continuous-time nonlinear system

\[ \dot{x}_t = f_x(x_t, u_t), \]
\[ z_t = f_z(x_t), \]  

(3.1)

where \( x_t \in \mathcal{X} \subset \mathbb{R}^n, u_t \in \mathcal{U} \subset \mathbb{R}^m \) and \( z_t \in \mathbb{R}^m \) represent the state, input and output vector respectively, with input and state constraint sets \( \mathcal{U} \) and \( \mathcal{X} \) which contain the origin.

The control task is to steer the system output along a regular curve, parameterised by a path parameter \( \theta \) from \( \theta_s \) to \( \theta_f \):

\[ z^d(\theta) : \mathbb{R} \to \mathbb{R}^m, \theta \in [\theta_s, \theta_f]. \]  

(3.2)

The path function \( z^d(\theta) \) is assumed to be twice differentiable with bounded second derivative with respect to \( \theta \). Given an initial time \( t \), path parameter \( \theta_t \) and state \( x_t \), the aim is to traverse forwards along the path to \( \theta_f \) in minimum time, subject to the constraints of the system (3.1). Define the following dynamics for \( \theta \):

\[ \dot{\theta}_t = v_t, \quad v_t \geq 0, \]  

(3.3)

where \( v_t \) is a virtual input which determines the path evolution. The constraint \( v_t \geq 0 \) enforces forward motion along the path. The minimum-time path-following problem can be posed as:

\[ P_{\infty}(t, \theta_t, x_t) : \quad \text{Minimise} \quad \int_t^{t_f} 1 \, d\tau, \]

Subject to \( \dot{\theta}_\tau = v_\tau, \quad v_\tau \geq 0, \quad \theta_{tf} = \theta_f \).
3.2 Receding horizon formulation

\[\dot{\xi}_\tau = f_\xi(\xi_\tau, u_\tau),\]
\[u_\tau \in \mathcal{U}, \xi_\tau \in \mathcal{X},\]
\[f_z(\xi_\tau) = z^d(\theta_\tau),\]
\[\theta_\tau \in [\theta^a, \theta^f],\]
\[\forall \tau \in [t, \infty),\]
\[(3.4)\]

where \(\{u_\tau, v_\tau\}\) are the control inputs and \(t_f\) is the time taken to complete the path. Note that the constraints are enforced for all \(\tau \in [t, \infty)\).

### 3.2 Receding horizon formulation

A closed-loop receding horizon approach is proposed with the intention of achieving minimum time control. The idea is to advance along the path as far as possible within a finite horizon of length \(N\). Consider the following finite horizon optimisation problem:

\[\mathcal{P}_N(t, \theta_t, \xi_t) : \text{ Minimise } - \int_t^{t+N} \theta_\tau \ d\tau,\]

Subject to \[\dot{\theta}_\tau = v_\tau,\]
\[v_\tau \geq 0,\]
\[\theta_\tau \leq \theta^f,\]
\[\dot{\xi}_\tau = f_\xi(\xi_\tau, u_\tau),\]
\[u_\tau \in \mathcal{U}, \xi_\tau \in \mathcal{X},\]
\[f_z(\xi_\tau) = z^d(\theta_\tau),\]
\[f_\xi(\xi_{t+N}, u_{t+N}) = 0.\]
\[(3.5)\]

The finite horizon optimisation (3.5) is posed over a finite horizon and may include only a portion of the path, in contrast to (3.4) which is solved over an infinite horizon which includes the entire path. The terminal constraint \(f_\xi(\xi_{t+N}, u_{t+N}) = 0\) forces the system to steady state at the end of the horizon.
The optimisation (3.5) is solved repeatedly at intervals of length $\Delta t$ in a receding hori-
zon fashion, as per the following proposed Control Algorithm:

**Control Algorithm 3.1. RH-TOC Algorithm**

1. Measure the current state, $\xi_t$.
2. Solve the finite horizon optimisation (3.5) to obtain an optimal finite horizon input trajectory $\{u^{*\tau}_t(\theta_t, \xi_t), v^{*\tau}_t(\theta_t, \xi_t)\}$, $\tau \in [t, t + N]$.
3. Over the interval $\tau \in [t, t + \Delta t]$, integrate $\dot{\theta}_\tau = v^{*\tau}_t$ to calculate $\theta_{t+\Delta t}$ and apply the input trajectory $u^{*\tau}_t$ to the plant.
4. Increment $t$ by $\Delta t$ and return to Step 1.

Control Algorithm 3.1 allows for feedback to be taken into account at intervals of length $\Delta t$, and is therefore a closed-loop control strategy. As the optimisation problem is posed over a finite horizon, it is not necessary for the entire path function to be known at ini-
tialisation; it is sufficient to maintain a look-ahead buffer of the path function during execution. In the remainder, denote the input trajectory produced by Control Algo-
algorithm 3.1 as $\{u^{RH}_t, v^{RH}_t\}$.

### 3.3 Properties of receding horizon time optimal control

In this section, the feasibility and optimality properties of Control Algorithm 3.1 are explored.

#### 3.3.1 Recursive feasibility

Since the optimisation (3.5) includes output and state constraints, it is important to ensure that recursive feasibility is maintained throughout the trajectory. Define the set of all feasible input trajectories for $\mathcal{P}_\infty(t, \theta_t, \xi_t)$ and $\mathcal{P}_N(t, \theta_t, \xi_t)$ as $\mathcal{X}_\infty(\theta_t, \xi_t)$ and $\mathcal{X}_N(\theta_t, \xi_t)$ respectively. It is assumed throughout that at $t = 0$ there exists at least one feasible solution for (3.4) and (3.5).
Assumption 3.1. $X_{\infty}(\theta_0, \xi_0)$ and $X_N(\theta_0, \xi_0)$ are non-empty.

The terminal constraint $f_{\xi}(\xi_{t+N}, u_{t+N}) = 0$ ensures that in each optimisation, the state at end of the horizon lies in a control invariant set. Hence, provided that (3.5) is feasible at initialisation, recursive feasibility of RH-TOC is guaranteed [51].

3.3.2 Time optimality

The finite horizon optimisation (3.5) occurs over only a portion of the path. As a result, the question arises of whether or not Control Algorithm 3.1 produces time-optimal trajectories. In this section, properties of differentially flat systems are used to show that for a sufficiently long horizon, $\{u_{RH}^t, v_{RH}^t\}$ is time-optimal.

It is assumed that starting from rest at any point on the path, it is possible to complete the path in finite time. This is stated formally in Assumption 3.2.

Assumption 3.2. For all $\theta_0 \in [\theta^s, \theta^f]$, if $z_0 = z^d(\theta_0)$ and $f_{\xi}(\xi_0, u_0) = 0$, then $X_{\infty}(\theta_0, \xi_0)$ is non-empty.

Differential flatness is a useful property when considering path-following problems. In particular, time optimality of RH-TOC will be analysed for second order differentially flat systems, as described in the following definition.

Definition 3.1. The system (3.1) is second order differentially flat [46, 72] with flat output $z \in \mathbb{R}^m$ if:

1. $\xi$ can be expressed as a function of $z$ and $\dot{z}$,

$$\xi = \Phi(z, \dot{z}), \quad (3.6)$$

2. $u$ can be expressed as a function of $z$, $\dot{z}$ and $\ddot{z}$,

$$u = \Psi(z, \dot{z}, \ddot{z}), \quad (3.7)$$
3. the elements of $z$ are differentially independent; they do not satisfy any differential equation.

The analysis for the remainder of this Chapter is restricted to systems which are second order differentially flat.

**Assumption 3.3.** The system (3.1) is second order differentially flat.

**Remark 3.1.** It can be readily shown that systems describing the dynamics of rigid fully actuated robotic manipulators with invertible forward kinematics, such as those considered in [12, 89, 103–105, 109], are second order differentially flat.

It is shown in [46] that for second order differentially flat systems, the minimum time path following problem (3.4) is equivalent to

$$
\hat{P}_\infty(t, \theta_t, v_t) : \begin{aligned}
\text{Minimise} & \int_t^{t_f} 1 \ d\tau, \\
\text{Subject to} & \dot{\theta}_r = v_r, \\
& v_r \geq 0, \\
& \theta_{t_f} = \theta_f, \ v_{t_f} = 0, \\
& u_r = \hat{\Psi}(\theta_r, \dot{\theta}_r, \ddot{\theta}_r) \in U, \\
& \xi_r = \hat{\Phi}(\theta_r, \dot{\theta}_r) \in X,
\end{aligned}
$$

(3.8)

where

$$
\hat{\Psi}(\theta, \dot{\theta}, \ddot{\theta}) = \Psi \left( z^d(\theta), \dot{z}^d(\theta, \dot{\theta}), \ddot{z}^d(\theta, \dot{\theta}, \ddot{\theta}) \right), \\
\hat{\Phi}(\theta, \dot{\theta}) = \Phi \left( z^d(\theta), \dot{z}^d(\theta, \dot{\theta}) \right).
$$

(3.9)

In a similar manner, the finite horizon problem (3.5) can be equivalently expressed as:

$$
\hat{P}_N(t, \theta_t, v_t) : \begin{aligned}
& - \int_t^{t+N} \dot{\theta}_r \ d\tau, \\
\text{Subject to} & \dot{\theta}_r = v_r, \\
& v_r \geq 0,
\end{aligned}
$$

(3.9)
\[ \theta_x \leq \theta_f, \quad v_{t+N} = 0, \]
\[ u_\tau = \hat{\Psi}(\theta_\tau, \dot{\theta}_\tau, \ddot{\theta}_\tau) \in \mathcal{U}, \]
\[ \xi_\tau = \hat{\Phi}(\theta_\tau, \dot{\theta}_\tau) \in \mathcal{X}. \]  

(3.10)

The optimisation problems (3.8) and (3.10) are completely described by two states, \( \theta \) and \( v = \dot{\theta} \), and one input \( \ddot{\theta} \). As a result, the solutions to (3.8) and (3.10) can be analysed in the \( \theta-v \) phase plane. This approach was utilised in \([12, 104]\) to construct open loop solutions to the minimum time problem (3.4). The same approach is employed to investigate time optimality of RH-TOC.

The following definition will be useful for the analysis.

**Definition 3.2.** Given a trajectory \( \{\theta_t, v_t\} \), define \( \tau(\theta) \) and \( v(\theta) \) as follows:

\[ \tau(\theta) = \min \{t : \theta_t = \theta\}, \]
\[ v(\theta) = v_{\tau(\theta)}. \]  

(3.11)

Denote the set of all feasible trajectories for \( \hat{\mathcal{P}}_\infty(t, \theta_t, \xi_t) \) and \( \hat{\mathcal{P}}_N(t, \theta_t, v_t) \) as \( \hat{\mathcal{X}}_\infty(\theta_t, v_t) \) and \( \hat{\mathcal{X}}_N(\theta_t, v_t) \) respectively. The following two Lemmas will be used later to prove time optimality of RH-TOC.

**Lemma 3.1.** Let Assumption 3.3 hold and for any \( t, \theta_t, \xi_t \), let \( \mathcal{X}_\infty(\theta_t, \xi_t) \) be non-empty. Then the optimal trajectory for \( \mathcal{P}_\infty(t, \theta_t, \xi_t) \), denoted \( \{u^*_\tau, v^*_\tau\} \) is unique and for all \( \{u_\tau, v_\tau\} \in \mathcal{X}_\infty(\theta_t, \xi_t) \),

\[ v(\theta) \leq v^*(\theta), \quad \forall \theta \in [\theta_t, \theta_f]. \]  

(3.12)

**Proof.** Since (3.11) is second order differentially flat, \( \mathcal{P}_\infty(t, \theta_t, \xi_t) \) is equivalent to \( \hat{\mathcal{P}}_\infty(t, \theta_t, v_t) \) and hence \( \hat{\mathcal{X}}_\infty(\theta_t, v_t) \) is non-empty. Assume to the contrary that there exists \( \{\tilde{\theta}_\tau, \tilde{v}_\tau\} \in \hat{\mathcal{X}}_\infty(\theta_t, v_t) \) and \( \theta_c \in [\theta_t, \theta_f] \) such that

\[ \tilde{v}(\theta_c) > v^*(\theta_c). \]  

(3.13)
Construct a path velocity trajectory $\tilde{v}_\tau$ as follows:

$$\tilde{v}(\theta) = \max(v^*(\theta), \tilde{v}(\theta)). \quad (3.14)$$

As the optimisation problem (3.8) is completely described by two states $\theta$ and $v$, $\tilde{v}(\theta)$ is a feasible trajectory, as illustrated in Figure 3.1.

![Figure 3.1: Illustration of proof of Lemma 3.1](image)

Clearly, $\tilde{\tau}(\theta_f) < \tau^*(\theta_f)$, and hence $\tilde{v}(\theta)$ contradicts optimality of $\{\theta^*_\tau, v^*_\tau\}$. The uniqueness of $\{\theta^*_\tau, v^*_\tau\}$ follows from a similar argument - see [34]. \qed

**Lemma 3.2.** Let Assumption 3.3 hold and for any $t, \theta_t, \xi_t$, let $X_N(\theta_t, \xi_t)$ be non-empty. If the system (3.1) is second order differentially flat, the optimal finite horizon trajectory for $P_N(t, \theta_t, \xi_t)$, denoted $\{u^*_\tau, v^*_\tau\}$ is unique and for all $\{u_\tau, v_\tau\} \in X_N(\theta_t, \xi_t)$,

$$v(\theta) \leq v^*(\theta), \quad \forall \theta \in [\theta_t, \min(\theta_{t+N}, \theta_{t^*_N}^{**})]. \quad (3.15)$$

**Proof.** The proof is divided into two parts. First, it is established that there exists $\{u^*_\tau, v^*_\tau\} \in X_N(\theta_t, \xi_t)$ such that

$$v(\theta) \leq v^*(\theta), \quad \forall \theta \in [\theta_t, \min(\theta_{t+N}, \theta_{t^*_N}^*)]. \quad (3.16)$$

Subsequently, it is shown that $\{u^*_\tau, v^*_\tau\}$ is the unique solution to $P_N(t, \theta_t, \xi_t)$. 
To establish the existence of \( \{u^*_\tau, v^*_\tau\} \), consider \( X_N(\theta_t, \xi_t) \), the set of all feasible solutions to \( P_N(t, \theta_t, \xi_t) \). Since (3.11) is second order differentially flat, \( P_N(t, \theta_t, \xi_t) \) is equivalent to \( \hat{P}_N(t, \theta_t, \xi_t) \), and any solution \( \{u_\tau, v_\tau\} \in X_N(\theta_t, \xi_t) \) is uniquely defined by the function \( v(\theta) \). Therefore, to construct \( \{u^*_\tau, v^*_\tau\} \), it is sufficient to construct \( v^*(\theta) \), which is achieved as follows:

\[
v^*(\theta) = \max_{\{\theta_\tau, v_\tau\} \in X_N(\theta_t, \xi_t)} v(\theta).
\]

As \( \hat{X}_N(\theta_t, v_t) \) is non-empty (since \( X_N(\theta_t, \xi_t) \) is non-empty), \( v^*(\theta) \) is guaranteed to exist. Using the same argument as shown in Figure 3.1, it is clear that \( v^*(\theta) \) is a feasible solution as long as the terminal constraint \( v^*_t + N = 0 \) is satisfied.

Let

\[
\theta_e = \max_{\{\theta_\tau, v_\tau\} \in X_N(\theta_t, \xi_t)} \theta_{t+N}.
\]

Since \( v^*(\theta) \) is the maximum of all feasible \( v(\theta) \), \( v^*(\theta_e) = 0 \) and \( \tau^*(\theta_e) \leq t + N \). Therefore, \( v^*_t = 0 \) and \( v^*(\theta) \) produces a feasible trajectory. In addition, \( \{\theta^*_\tau, v^*_\tau\} \) is clearly unique.

It now remains to show that \( \{u^*_\tau, v^*_\tau\} \) is the unique optimal solution to \( P_N(t, \theta_t, \xi_t) \), i.e. \( v^{**}(\theta) = v^*(\theta) \) for all \( \theta \in [\theta_t, \min(\theta^{**}_t, \theta^*_t)] \).

Assume to the contrary that there exists \( \theta_a \in [\theta_t, \min(\theta^{**}_t, \theta^*_t)] \) such that

\[
v^{**}(\theta_a) < v^*(\theta_a).
\]

Since (3.16) holds, it follows that for some \( t_a \in [t, t + N] \),

\[
\theta^{**}_{t_a} < \theta^*_t.
\]

From optimality of \( \{u^{**}_\tau, v^{**}_\tau\} \),

\[
- \int_t^{t+N} \theta^{**}_\tau \,d\tau \leq - \int_t^{t+N} \theta^*_\tau \,d\tau.
\]
From (3.21) and (3.21), there must exist \( t_b \in [t, t + N] \) such that

\[
\theta^*_{t_b} < \theta^{**}_{t_b},
\]  

(3.22)

and therefore

\[
\tau^{**}(\theta^*_{t_b}) < \tau^*(\theta^*_{t_b}).
\]  

(3.23)

Rewriting (3.23) as

\[
\int_t^{\tau^{**}(\theta^*_{t_b})} 1 \, dt < \int_t^{\tau^*(\theta^*_{t_b})} 1 \, dt.
\]  

(3.24)

and applying a change of variables yields

\[
\int_{\theta^*_{t_b}}^{\theta^{**}_{t_b}} \frac{1}{v^{**}(\theta)} \, d\theta < \int_{\theta^*_{t_b}}^{\theta^*_{t_b}} \frac{1}{v^*(\theta)} \, d\theta.
\]  

(3.25)

It follows that there exists \( \theta_c \in [\theta_t, \theta^*_{t_b}] \) such that

\[
v^{**}(\theta_c) > v^*(\theta_c),
\]  

(3.26)

which contradicts (3.10). Therefore, \( v^{**}(\theta) = v^*(\theta) \) for all \( \theta \in [\theta_t, \theta^*_{t_b + N}] \).

\[ \square \]

Lemmas 3.1 and 3.2 will be used to show that RH-TOC is time-optimal, provided the horizon length is sufficiently long. A condition on the minimum horizon length required to guarantee time optimality is derived in the following.

Let \( \{\theta^*_r, \xi^*_r\} \) denote the optimal state trajectory for \( \mathcal{R}_\infty(0, \theta_0, \xi_0) \). For any \( t \), define the following optimisation which describes the problem of stopping along the path in minimum time:

\[
\text{Minimise} \quad \int_t^{t+t_s} 1 \, d\tau,
\]

Subject to \( \theta_t = \theta^*_t, \xi_t = \xi^*_t \).
\[ \dot{\theta}_\tau = v_\tau, \]
\[ v_\tau \geq 0, \theta_\tau \leq 0, \]
\[ \dot{\xi}_\tau = f_\xi(\xi_\tau, u_\tau), \]
\[ u_\tau \in \mathcal{U}, \xi_\tau \in \mathcal{X}, \]
\[ f_\eta(\xi_\tau) = q^d(\theta_\tau), \]
\[ f_\xi(\xi_{t_e}, u_{t_e}) = 0. \]  
(3.27)

Denote optimal input trajectories for (3.27) as \( \{u_0^\tau(t), v_0^\tau(t)\} \) and the optimal time as \( t_{s}^\tau(t) \), which is the minimum time required to come to rest along the path from initial state \( \{\theta^*_t, \xi^*_t\} \).

**Condition 3.1.** The horizon length \( N \) satisfies

\[ N - \Delta t \geq \max_t t_{s}^0(t). \]  
(3.28)

As shown in the following, Condition 3.1 ensures that starting from an appropriate initial condition, there exists a feasible trajectory for the finite horizon problem (3.5) that coincides with the minimum time trajectory over an initial interval of length \( \Delta t \).

**Lemma 3.3.** Let Condition 3.1 hold. Then for any \( t \) there exists \( \{\tilde{u}_\tau, \tilde{v}_\tau\} \in \mathcal{X}_N(\theta^*_t, \xi^*_t) \) such that

\[ \tilde{u}_\tau = u^*_\tau, \]
\[ \tilde{v}_\tau = v^*_\tau, \forall \tau \in [t, t + \Delta t]. \]  
(3.29)

**Proof.** Construct \( \{\tilde{u}_\tau, \tilde{v}_\tau\} \) as follows:

\[ \{\tilde{u}_\tau, \tilde{v}_\tau\} = \begin{cases} 
\{u^*_\tau, v^*_\tau\}, & \tau \in [t, t + \Delta t], \\
\{u_0^\tau(t + \Delta t), v_0^\tau(t + \Delta t)\}, & \tau \in (t + \Delta t, t_{e}], \\
\{u_{t_e}^0(t + \Delta t), v_{t_e}^0(t + \Delta t)\}, & \tau \in (t_{e}, t + N], 
\end{cases} \]  
(3.30)

where \( t_{e} = t + \Delta t + t_{s}^0(t + \Delta t) \). Note that since (5.28) holds, \( t_{s}^0(t + \Delta t) \leq N - \Delta t \). By
the constraints of (3.27),

\[ f_\xi(\xi_{t+N}, u_{t+N}) = 0. \]  \hspace{1cm} (3.31)

Therefore, \( \{ \tilde{\theta}_\tau, \tilde{v}_\tau \} \in X_N(\tilde{\theta}_t, \tilde{v}_t) \).

The following Lemma states that the initial segment of length \( \Delta t \) of \( \{ v^*_\tau, u^*_\tau \} \) is equal to the time-optimal trajectory, and will be used to show that under Condition 3.1, \( \{ u^{\text{RH}}_t, v^{\text{RH}}_t \} \) is time-optimal.

**Lemma 3.4.** Let Assumptions 3.1-3.3 and Condition 3.1 hold. For any \( t \), let \( \{ v^*_\tau, u^*_\tau \} \) be the optimal input trajectory for \( \mathcal{P}_N(t, \theta^*_t, \xi^*_t) \). Then the following holds:

\[ u^*_\tau = u^*_{t+\tau}, \quad v^*_\tau = v^*_{t+\tau}, \quad \forall \tau \in [t, t+\Delta t]. \]  \hspace{1cm} (3.32)

**Proof.** Since (3.1) is second order differentially flat, (3.32) holds if for all \( \theta \in [\theta^*_t, \theta^{**}_{t+\Delta t}] \),

\[ v^*(\theta) = v^{**}(\theta). \]

Assume to the contrary that there exists \( \theta_c \in [\theta^*_t, \theta^{**}_{t+\Delta t}] \) such that \( v^*(\theta_c) \neq v^{**}(\theta_c) \). Therefore, either \( v^*(\theta_c) < v^{**}(\theta_c) \) or \( v^*(\theta_c) > v^{**}(\theta_c) \).

Assume first that \( v^*(\theta_c) < v^{**}(\theta_c) \). It is straightforward to show that if Assumption 3.2 holds, there exists \( \{ \bar{u}_\tau, \bar{v}_\tau \} \in X_\infty(\theta^*_t, \xi^*_t) \) such that

\[ u^*_\tau = \bar{u}_\tau, \quad v^*_\tau = \bar{v}_\tau, \quad \forall \tau \in [t, t+\Delta t]. \]  \hspace{1cm} (3.33)

However, (3.33) contradicts optimality of \( \{ u^*_\tau, v^*_\tau \} \) by Lemma 3.1, since \( v^*(\theta_c) < v^{**}(\theta_c) \).

Now assume that \( v^*(\theta_c) > v^{**}(\theta_c) \). By Lemma 3.3, there exists \( \{ \tilde{u}_\tau, \tilde{v}_\tau \} \in X_N(\theta^*_t, \xi^*_t) \) such that

\[ u^*_\tau = \tilde{u}_\tau, \quad v^*_\tau = \tilde{v}_\tau, \quad \forall \tau \in [t, t+\Delta t]. \]  \hspace{1cm} (3.34)

However, (3.34) contradicts optimality of \( \{ u^{**}_\tau, v^{**}_\tau \} \) by Lemma 3.2, since \( v^*(\theta_c) > v^{**}(\theta_c) \).

It has been shown that \( v^*(\theta_c) \neq v^{**}(\theta_c) \) leads to a contradiction, and therefore (3.32).
must hold.

Time optimality of RH-TOC immediately follows from Lemma 3.4, as shown in the following main result.

**Theorem 3.1.** Let Assumptions 3.1-3.3 and Condition 3.1 hold. Then \( \{u_{t}^{RH}, v_{t}^{RH}\} \) is the time-optimal solution for \( \mathcal{P}_{\infty}(0, \theta_{0}, \xi_{0}) \).

**Proof.** Let \( \{\theta^{*}, \xi^{*}\} \) denote the optimal solution for \( \mathcal{P}_{\infty}(0, \theta_{0}, v_{0}) \) and assume that at time \( t, \theta_{t} = \theta^{*}_{t} \) and \( v_{t} = v^{*}_{t} \). By Lemma 3.4,

\[
\begin{align*}
    u_{\tau}^{RH} &= u_{\tau}^{*}, \quad v_{\tau}^{RH} = v_{\tau}^{*}, \quad \forall \tau \in [t, t + \Delta t].
\end{align*}
\]

Consider the initialisation of the Control Algorithm 3.1, where \( t = 0 \). Clearly, \( \theta_{0} = \theta^{*}_{0} \) and \( v_{0} = v^{*}_{0} \). It can therefore be concluded using induction that the input trajectory produced by Control Algorithm 3.1 is the time-optimal solution for \( \mathcal{P}_{\infty}(t, \theta_{0}, \xi_{0}) \).

Theorem 3.1 implies that if \( N \) satisfies Condition 3.1, RH-TOC is a closed-loop minimum-time path-following control strategy for second order differentially flat systems. Furthermore, Control Algorithm 3.1 is guaranteed to be recursively feasible regardless of horizon length. Therefore, even if the horizon is not long enough to produce time-optimal trajectories, it is ensured that in the absence of disturbances the system will never deviate from the path.

The receding horizon strategy can easily incorporate additional constraints, such as rate of change constraints on the inputs, by including them in the finite horizon optimisation (3.5). Time optimality of the receding horizon control scheme with additional constraints remains an open question.

### 3.4 Practical considerations

Practical considerations regarding online implementation of the control scheme are discussed in the following.
3.4.1 Horizon selection

Time optimality of the receding horizon approach requires \( N \) to be selected sufficiently large. It is straightforward to show that if the system is initialised at rest, there exists a suitable \( N \) which is less than the total time required to traverse the path. If the plant model and path function are known \textit{a priori}, a numerical search based on the offline approaches proposed in [12, 103, 104, 109] can be used to determine an upper bound on the minimum \( N \) required to satisfy Condition 3.1.

However, without \textit{a priori} knowledge of the path, it is difficult to determine a useful value of \( N \) which will guarantee satisfaction of Condition 3.1. However, it is suggested to choose \( N - \Delta t \) to be the maximum time required to decelerate an individual joint from its maximum speed to rest, assuming that the velocity of each joint is bounded. While it is not guaranteed that this choice of \( N \) is sufficient to guarantee time optimality, from a practical perspective the suggested method provides a reasonable approximation. The effect of the horizon length on traversal time is investigated in Section 5.5.

3.4.2 Disturbance handling

As the receding horizon approach utilises feedback, it can be used to implement online time optimal control strategies with potential disturbance rejection properties. In practice, feasibility issues are likely to arise as a disturbance may cause the system to deviate from the path. One approach to address this limitation is to include a penalty on path following accuracy in the cost function instead of imposing a hard constraint. The resulting optimisation problem is then

\[
\mathcal{P}^*_N(t, \theta_t, \xi_t) :
\]

Minimise \( J^*_t, N \),

Subject to \( \dot{\theta}_t = v_t \),

\( v_t \geq 0 \),

\( \theta_t \leq \theta^f \),

\( \theta^f \),
\[ \dot{\xi}_\tau = f_\xi(\xi_\tau, u_\tau), \]
\[ u_\tau \in \mathcal{U}, \, \xi_\tau \in \mathcal{X}, \]
\[ v_{t+N} = 0, \]  
(3.36)

where
\[ J_{t,N}^* = \int_t^{t+N} \left( -\theta_\tau + (f_q(\xi_\tau) - q^d(\theta_\tau))^T Q (f_q(\xi_\tau) - q^d(\theta_\tau)) \right) d\tau \]  
(3.37)

and \( Q \in \mathbb{R}^{n \times n} \) is positive definite. Near time-optimal path-following trajectories can be produced by setting \( Q \) sufficiently large, while avoiding feasibility issues when disturbances arise.

### 3.5 Simulation results

To demonstrate the theoretical results derived in Section 3.3.2, simulations were conducted where RH-TOC was implemented on a rigid X-Y table model with the following equation of motion:

\[
\begin{bmatrix}
J_x & 0 \\
0 & J_y
\end{bmatrix} \ddot{z} + \begin{bmatrix}
B_x & 0 \\
0 & B_y
\end{bmatrix} \dot{z} = \begin{bmatrix}
u_x \\
u_y
\end{bmatrix},
\]  
(3.38)

where:
- \( z \) is the vector containing the \( x \) and \( y \) axis positions,
- \( u_x, u_y \) are the input torques,
- \( J_x, J_y \) are rotational inertias,
- \( B_x, B_y \) are coefficients of viscous friction, and
- \( P_x, P_y \) are the values of lead screw pitch

for the \( x \) and \( y \) axes respectively. The rigid X-Y table model parameters are given in Table 3.1. The control inputs are subject to the constraints \(|u_x|, |u_y| \leq 1 \text{ N m}\). It can easily be shown that the model (3.38) is a second order differentially flat system.
Table 3.1: Rigid X-Y table model parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Unit</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inertia $J$</td>
<td>kg m$^2$</td>
<td>$3.83 \times 10^{-5}$ $4.15 \times 10^{-5}$</td>
</tr>
<tr>
<td>Viscous friction $B$</td>
<td>N m s rad</td>
<td>$2.88 \times 10^{-5}$ $2.74 \times 10^{-4}$</td>
</tr>
<tr>
<td>Pitch $P$</td>
<td>mm rad</td>
<td>0.7958</td>
</tr>
</tbody>
</table>

Figure 3.3 shows plots of the input torques $u_x$, $u_y$ and virtual input $v$ produced by the receding horizon controller for the elliptical path shown in Figure 3.2. The horizon length is specified as $N = 0.04$ s, which is sufficient for time optimality.

To verify that the system traverses the path in minimum-time, the offline method proposed in [12] was used to calculate the minimum-time solution for the entire path. It can be shown that the bound constraints on $u_x$ and $u_y$ correspond to bounds on the time derivative of the virtual input $\dot{v}$ which depend on $\theta$ and $v$ (see Section 2.1). According to [12], in the time-optimal solution, $\dot{v}$ switches between the minimum and maximum bound at certain points along the path, and a procedure is provided for finding the switching points. These switching points can be observed in Figure 3.3, and were shown to match those obtained using the offline method. It should also be noted
3.5 Simulation results

Figure 3.3: Input trajectories for elliptical path

from Figure 3.3 that at least one of $u_x$ and $u_y$ is saturated as the path is being traversed, as would be expected of a minimum time trajectory. The small oscillations in the plots of $u_x$ and $u_y$ are caused by numerical issues in the nonlinear solver.

In order to investigate the effect of horizon length on traversal time, simulations were also conducted with the more complicated flower-shaped path shown in Figure 3.4 with a range of values of $N$. Figure 3.5 shows plots of path speed and joint velocities versus $\theta$ for four different horizon lengths. It can be observed that using shorter horizons limits the joint velocities as the system traverses the path.

The relationship between traversal time and horizon length is shown in Figure 3.6. The results indicate that the trajectories approach the minimum time trajectory as $N$ increases. Beyond $N=0.036$ s, Condition 3.1 is satisfied, and hence there is no further improvement in traversal time. The selection of the tuning parameter $N$ can be seen
Figure 3.4: Flower-shaped path

as a tradeoff between traversal time and computational complexity. It should be noted that the method suggested in Section 3.4.1 for estimating the minimum horizon length, using the maximum joint velocities recorded in Figure 3.5, also yields a minimum horizon length of $N = 0.036$ s.
3.5 Simulation results

Figure 3.5: Plots of $\dot{x}$, $\dot{y}$ and $v$ versus $\theta$ with varying horizon lengths for flower-shaped path

Figure 3.6: Traversal time versus horizon length for flower shaped path
3.6 Conclusion

A receding horizon time-optimal path-following controller has been developed for non-linear systems. The proposed approach is shown to achieve minimum-time path following for a class of differentially flat systems if the horizon is selected to be sufficiently long. Simulation results demonstrate how selecting a shorter horizon increases the traversal time. A soft-constrained approach is proposed in order to maintain feasibility of the optimisation problem in the presence of disturbances.

This Chapter has focussed on the problem of traversing along a desired path exactly in minimum-time subject to the system constraints, and establishing conditions under which the proposed receding horizon approach yields the minimum-time solution. However, as discussed in Chapter 1, there exists a tradeoff between competing control objectives of accuracy and productivity.

Therefore, it is more practical to address the contouring problem as defined in Chapter 1, where the constraint on exact path-following is relaxed and a trade-off between contouring accuracy and productivity is considered. With this motivation, a similar control scheme, entitled model predictive contouring control, is developed in the remainder of this thesis. The approach is first developed for biaxial contouring systems in Chapter 4, and is subsequently generalised to multi-axis systems in Chapter 6.
Chapter 4
Biaxial Model Predictive Contouring Control

A receding horizon approach to the contouring problem is developed for biaxial systems with linear dynamics. The control scheme, referred to as model predictive contouring control (MPCC), is based on the minimisation of a cost function representing competing objectives of minimising contouring error while maximising path speed.

In the receding horizon time-optimal scheme developed in the previous chapter, the cost function was selected to minimise traversal time and the contouring error was constrained to zero. In contrast, the MPCC scheme allows the system to deviate from the path, but with a penalty applied to contouring error in the cost function.

In order to facilitate online implementation, a linear time-varying (LTV) approximation is proposed which reduces the computational burden of the approach. Path completion and convergence to the endpoint are guaranteed by imposing an additional contractive constraint on the path parameter and switching to a stable regulator at the end of the path.

A substantial proportion of this Chapter and the next have been published in [66–68].
4.1 The biaxial contouring control problem

The contouring problem introduced in Chapter II is stated more precisely for biaxial systems in the following. Consider a linear discrete-time system which describes the dynamics of a biaxial contouring system:

\[
\xi_{k+1} = A \xi_k + B u_k, \quad \begin{bmatrix} x_k \\ y_k \end{bmatrix} = C \xi_k,
\]

(4.1)

where \( \xi_k \in \mathbb{R}^n \) and \( u_k \in \mathbb{R}^{n_u} \) denote the system state and input vectors respectively, and \( x_k \in \mathbb{R} \) and \( y_k \in \mathbb{R} \) denote the \( x \) and \( y \) axis displacements at time \( k \). The system is subject to input and state constraints \( u_k \in \mathcal{U}, \xi_k \in \mathcal{X} \), where \( \mathcal{U} \) and \( \mathcal{X} \) are polytopes containing the origin.

Remark 4.1. The minimum-time path-following problem (3.4) defined in Chapter III was nonlinear, and formulated in continuous time. While a continuous-time formulation is suitable for theoretical analysis of the receding horizon control scheme, a linear discrete-time version is more suitable for practical implementation. As the focus of this Chapter is to develop a control approach suitable for implementation on a real system, the biaxial contouring control problem is posed for discrete-time linear systems.

The objective is to steer \((x_k, y_k)\) along a continuously differentiable and bounded two-dimensional geometric path \((x^d(\theta), y^d(\theta))\):

\[
x^d : [\theta^s, 0] \rightarrow \mathbb{R}; \quad y^d : [\theta^s, 0] \rightarrow \mathbb{R}; \quad \theta^s < 0.
\]

(4.2)

As discussed in Chapter III, contouring error is defined as the minimum deviation from the desired path. In biaxial contouring, this is equivalent to the normal deviation from the path [63]. The contouring error \( \epsilon_k^c \) can be expressed as

\[
\epsilon_k^c = \sin \phi(\theta^r) \left(x_k - x^d(\theta^r)\right) - \cos \phi(\theta^r) \left(y_k - y^d(\theta^r)\right),
\]

\[
\phi(\theta^r) = \arctan \left(\frac{\nabla y^d(\theta^r)}{\nabla x^d(\theta^r)}\right), \quad \nabla f(\theta^r) = \frac{df(\theta)}{d\theta} \bigg|_{\theta=\theta^r},
\]

(4.3)
where $\theta^r(x, y)$ is the value of the path parameter at which the distance between the point $(x^d(\theta), y^d(\theta))$ and $(x, y)$ is minimal. An illustration of contouring error is shown in Figure 4.1.

![Figure 4.1: Illustration of contouring error measurement $\epsilon^c_k$ at time $k$](image)

The multi-objective control problem involves selecting the control input $u$ such that the solutions of (4.1) traverse near the desired geometric path, in a manner that minimises contouring error while maximising path speed. This can be expressed as the following optimisation over the entire path,

$$\min_u \sum_{k=0}^{\infty} q_c (\epsilon^c_k)^2 + q_t T,$$

subject to $\xi_{k+1} = A\xi_k + B u_k$, $u_k \in \mathcal{U}$, $\xi_k \in \mathcal{X}$, \hspace{1cm} (4.4)

where $T$ is the time taken to traverse the path and $q_c$, $q_t$ are weights corresponding to the relative importance of contouring accuracy and speed.

### 4.2 Receding horizon formulation

The contouring control problem as expressed in (4.4) is not suitable for online implementation. Consequently, it is proposed to reformulate (4.4) using a model predictive framework, so that a cost function is minimised over a finite horizon.
It is assumed that the desired path \((x^d(\theta), y^d(\theta))\) is parameterised by arc length:

\[
\frac{ds}{d\theta} = 1,
\]

where \(s\) denotes the distance travelled along the path. Arc length parameterisation of general curves is nontrivial; however techniques exist in the literature for approximate arc length parameterisation of spline curves, see for example [42]. The system (4.1) is augmented with the following dynamics

\[
\theta_{k+1} = \theta_k + v_k,
\]

where \(v_k\) is a virtual input to be determined by the controller and \(\theta_k\) denotes the value of the path parameter at time \(k\). The following constraints are imposed on \(v_k\):

\[
v_k \in [0, v_{\text{max}}], \ v_{\text{max}} > 0.
\]

Since the path is parameterised by arc length, \(v\) is directly proportional to the path speed. Also, non-reversal of the path is guaranteed, since \(v_k \geq 0\).

It is proposed to use \(\theta_k\), whose evolution is governed by (4.6), as an approximation to \(\theta^r(x_k, y_k)\).

**Assumption 4.1.** \(\theta_k\) is sufficiently close to \(\theta^r(x_k, y_k)\).

**Remark 4.2.** Let \(\epsilon^l_k\) denote the path distance that \((x^d(\theta^r), y^d(\theta^r))\) lags \((x^d(\theta_k), y^d(\theta_k))\) and approximate \(\epsilon^l_k\) as

\[
\hat{\epsilon}^l(\xi_k, \theta_k) = -\cos \phi(\theta_k)(x_k - x^d(\theta_k)) - \sin \phi(\theta_k)(y_k - y^d(\theta_k))
\]

Refer to Fig. 4.2 for a graphical interpretation of \(\epsilon^l\), \(\epsilon^l\) and their approximations. It can be observed that for most paths, \(\theta_k \to \theta^r(x_k, y_k)\) as \(\hat{\epsilon}^l(\xi_k, \theta_k) \to 0\). Therefore for practical purposes, Assumption 4.1 can be enforced by including an appropriate penalty on \(\hat{\epsilon}^l\) in the cost function (4.10).
By Assumption 4.1, the contouring error can be approximated by

\[
\tilde{\epsilon}(\xi_k, \theta_k) = \sin(\phi(\theta_k))(x_k - x^d(\theta_k)) - \cos(\phi(\theta_k))(y_k - y^d(\theta_k)),
\]

(4.9)

and \( \theta_k \) can be used as an approximation of how far along the path the system has travelled. Note that while \( \theta^r \) in Figure 4.1 is not necessarily unique, the smooth evolution of \( \theta_k \) enforced by the constraint on \( v_k \) ensures that the system follows the path smoothly, provided \( v_{\text{max}} \) is chosen to be sufficiently small.

The model predictive cost function \( J_k \) is an approximation of the contouring control cost function in (4.4) over a horizon of \( N \) time steps, and represents the relative importance of contouring accuracy and path speed. Control input deviations are also penalised in order to achieve smooth control inputs.

\[
J_k = \sum_{i=1}^{N} \begin{bmatrix} \tilde{\epsilon}(\xi_{k+i,k}, \theta_{k+i,k}) \\ \tilde{\epsilon}(\xi_{k+i,k}, \theta_{k+i,k}) \end{bmatrix}^T Q \begin{bmatrix} \tilde{\epsilon}(\xi_{k+i,k}, \theta_{k+i,k}) \\ \tilde{\epsilon}(\xi_{k+i,k}, \theta_{k+i,k}) \end{bmatrix} - q_\theta \theta_{k+i,k} + \begin{bmatrix} \Delta u_{k+i-1,k} \\ \Delta v_{k+i-1,k} \end{bmatrix}^T R \begin{bmatrix} \Delta u_{k+i-1,k} \\ \Delta v_{k+i-1,k} \end{bmatrix},
\]

(4.10)
where

\[
\Delta u_{k+i-1,k} = u_{k+i-1,k} - u_{k+i-2,k}, \\
\Delta v_{k+i-1,k} = v_{k+i-1,k} - v_{k+i-2,k}, \\
Q = \begin{bmatrix} q_c & 0 \\ 0 & q_l \end{bmatrix}, \quad q_c, q_l, q_\theta > 0, \quad R \in \mathbb{R}^{(n_u+1)^2}, \tag{4.11}
\]

The notation \( \xi_{k+i,k} \) denotes the prediction of \( \xi \) at time \( k+i \), predicted at time \( k \). The predictions \( \xi_{k+i,k} \) and \( \theta_{k+i,k} \) are formed by recursively applying the model equations (4.1) and (4.6)

\[
\xi_{k+i,k} = A\xi_{k+i-1,k} + Bu_{k+i-1,k}, \\
\theta_{k+i,k} = \theta_{k+i-1,k} + v_{k+i-1,k}, \quad i \in \{1, \ldots, N\} \tag{4.12}
\]

where at each time step, \( \xi_{k+i,k} \) is initialised with the measured state \( \xi_k \) and \( \theta_{k+i,k} \) is initialised with the value of the path parameter \( \theta_k \):

\[
\xi_{k,k} = \xi_k, \\
\theta_{k,k} = \theta_k. \tag{4.13}
\]

The penalty weights \( q_c, q_\theta \) and \( R \) are tuning parameters to be decided based on the relative importance of contouring accuracy, path speed, and control deviations, and \( q_l \) is chosen to be sufficiently large to satisfy Assumption 4.1.

The cost function (4.10) can easily be modified to incorporate other control objectives, such as the minimisation of control effort. Similarly, additional constraints specific to certain applications (e.g. jerk constraints, as considered in [35]) can also be included.

The cost function (4.10) leads to the following optimisation problem being posed at
Minimise $J_k$

Subject to

\[
\xi_{k+i,k} = A\xi_{k+i-1,k} + Bu_{k+i-1,k},
\]

\[
\xi_{k,k} = \xi_k,
\]

\[
\theta_{k+i,k} = \theta_{k+i-1,k} + v_{k+i-1,k},
\]

\[
\theta_{k,k} = \theta_k,
\]

\[
u_{k+i-1,k} \in \mathcal{U}, \quad v_{k+i-1,k} \in [0, v_{max}],
\]

\[
\xi_{k+i,k} \in \mathcal{X}, \quad \theta_{k+i,k} \in [\theta^s, 0], \quad i = 1, ..., N.
\] (4.14)

The following Control Algorithm summarises the implementation of biaxial model predictive contouring control.

**Control Algorithm 4.1 (Nonlinear Biaxial MPCC Algorithm).**

1. Initialise $k = 0$, $\theta_0 = \theta^s$.
2. Solve the constrained optimisation (4.14) to obtain $u^*_{k}$ and $v^*_{k}$
3. Apply the first element of $u^*_{k}$ to the plant and update $\theta_k$ with the first element of $v^*_{k}$
4. Increment $k$ and return to Step 2

### 4.3 Linear time-varying model predictive contouring control

In general, solving the optimisation (4.14) in real time is computationally difficult. Even though the system (4.1) is linear, the nonlinearity of the path function $(x^d(\theta), y^d(\theta))$ results in a nonlinear optimisation problem. As a result, Control Algorithm 4.1 is not suitable for real-time implementation.

To allow for practical implementation of MPCC, a linear time-varying (LTV) approach is proposed which approximates the optimisation problem with a convex quadratic program (QP). A number of LTV MPC schemes have been proposed in the literature (e.g. [81, 99]), where a nonlinear plant model is linearised around one or more operating
points at each time step.

In order to approximate the cost function with a quadratic function of the decision variables, the path functions are linearised around an estimate of the path state trajectory. The estimated path state trajectory is computed using information from the previous time step. This is described in the following sections.

### 4.3.1 Cost function approximation

Assume that an estimate of the path state trajectory \( \hat{\Theta}^*_k = \{ \hat{\theta}^*_k, \ldots, \hat{\theta}^*_{k+N-1,k} \} \) is known. The desired path functions \( x^d(\theta), y^d(\theta) \) are linearised across the prediction horizon using a Taylor series expansion around \( \hat{\Theta}^*_k \) and neglecting higher order terms

\[
\begin{align*}
    x^a_d(k, \hat{\Theta}^*_k) &= x^d(\hat{\theta}^*_{k+i,k}) + \nabla x^d(\hat{\theta}^*_{k+i,k})(\theta - \hat{\theta}^*_{k+i,k}), \\
    y^a_d(k, \hat{\Theta}^*_k) &= y^d(\hat{\theta}^*_{k+i,k}) + \nabla y^d(\hat{\theta}^*_{k+i,k})(\theta - \hat{\theta}^*_{k+i,k}).
\end{align*}
\]  

(4.15)

Approximations for the contouring error and \( \hat{\epsilon} \) can then be formed using (4.15),

\[
\begin{align*}
    \hat{\epsilon}^{a,c}_{k+i}(\xi, \theta, \hat{\Theta}^*_k) &= \sin(\hat{\theta}^*_{k+i,k})(x - x^a_d(k, \hat{\Theta}^*_k)) + \cos(\hat{\theta}^*_{k+i,k})(y - y^a_d(k, \hat{\Theta}^*_k)), \\
    \hat{\epsilon}^{a,l}_{k+i}(\xi, \theta, \hat{\Theta}^*_k) &= -\cos(\hat{\theta}^*_{k+i,k})(x - x^a_d(k, \hat{\Theta}^*_k)) + \sin(\hat{\theta}^*_{k+i,k})(y - y^a_d(k, \hat{\Theta}^*_k)).
\end{align*}
\]  

(4.16)

The linearised predictions \( \hat{\epsilon}^{a,c} \) and \( \hat{\epsilon}^{a,l} \) are used to approximate the cost function. The estimated cost is

\[
J^a(k, \hat{\Theta}^*_k) = \sum_{i=1}^{N} \left( \begin{array}{c}
    \hat{\epsilon}^{a,c}_{k+i,k} \\
    \hat{\epsilon}^{a,l}_{k+i,k}
\end{array} \right)^T Q \left( \begin{array}{c}
    \hat{\epsilon}^{a,c}_{k+i,k} \\
    \hat{\epsilon}^{a,l}_{k+i,k}
\end{array} \right) - q_\theta \hat{\theta}^*_{k+i,k} + \left( \begin{array}{c}
    \Delta u^T_{k+i-1,k} \\
    \Delta v^T_{k+i-1,k}
\end{array} \right) R \left( \begin{array}{c}
    \Delta u^T_{k+i-1,k} \\
    \Delta v^T_{k+i-1,k}
\end{array} \right),
\]  

(4.17)

where

\[
\hat{\epsilon}^{a,c}_{k+i,k} = \hat{\epsilon}^{a,c}_{k+i,k}(\xi_{k+i,k}, \theta_{k+i,k}, \hat{\Theta}^*_k),
\]
The LTV model predictive contouring controller is implemented by minimising the approximate cost function (4.17) subject to the constraints:

\[
\begin{align*}
\text{Minimise} & \quad J_{a,k}^p(\hat{\Theta}_k^*), \\
\text{Subject to} & \quad \xi_{k+i,k} = A\xi_{k+i-1,k} + Bu_{k+i-1,k}, \\
& \quad \theta_{k+i,k} = \theta_{k+i-1,k} + v_{k+i-1,k}, \\
& \quad \xi_{k,k} = \xi_k, \quad \theta_{k,k} = \theta_k, \\
& \quad u_{k+i-1,k} \in \mathcal{U}, \quad v_{k+i-1,k} \in [0,v_{max}], \\
& \quad \xi_{k+i,k} \in \mathcal{X}, \quad \theta_{k+i,k} \in [\theta_s,0].
\end{align*}
\]

The optimisation (4.19) can be formulated as a convex quadratic program (QP) of the following form

\[
\begin{align*}
\text{Minimise} & \quad \frac{1}{2} w_k^T H_k w_k + w_k^T G_k, \\
\text{Subject to} & \quad E_k w_k \leq F_k,
\end{align*}
\]

where \( w_k = [v_k, \ldots v_{k+N-1}, \Delta u_k, \ldots \Delta u_{k+N-1}]^T \). \( H_k \) and \( G_k \) are computed by combining the approximate cost function (4.17) with the plant model (4.1), and \( E_k \) and \( F_k \) are derived from the inequality constraints of (4.19). Refer to [79] for a detailed description. The quadratic program (4.20) can be solved using conventional optimisation techniques.

Calculation of the linearised path functions (4.15) requires an estimate of the path state trajectory \( \hat{\Theta}_k^* \). In a similar manner to [81], the state trajectory is estimated using the optimal virtual input trajectory from the previous time step, as described in the following.

### 4.3.2 Trajectory approximation

Let \( \hat{v}_{k-1} \) denote the optimal virtual input trajectory obtained by solving (4.14) at time \( k - 1 \). An approximate input trajectory for the current time step \( \hat{v}_k \) can be obtained by
truncating $v^*_{k-1}$ and appending a feasible input $\hat{\theta}^*_{k+N-1}$ (typically $\hat{\theta}^*_{k+N-1} = 0$), so that $\hat{v}^*_k = \{u^*_{k-1}, \ldots, u^*_{k+N-2, k-1}, \hat{\theta}^*_{k+N-1}\}$. The approximate state trajectory $\hat{\Theta}^*_k$ is computed by applying $\hat{v}^*_k$ to (4.6) with initial condition $\hat{\theta}^*_{k,k} = \theta_k$.

At the initial time step $k = 0$, $\hat{\Theta}^*_0$ is computed via the following iterative procedure.

**Procedure 4.1 (Initial trajectory estimation).**

1. Initialise $\hat{\Theta}^*_0$ to $\hat{\Theta}^*_{0} = \{\theta_0, \theta_0, \ldots, \theta_0\}$ and set $j = 0$.
2. Compute the LTV model (4.16) based on $\hat{\Theta}^*_j$.
3. Solve the optimisation (4.19) to obtain $v^*_j$. Compute $\hat{\Theta}^*_j$ by applying $v^*_j$ to (4.6).
4. Increment $j$ and set $\hat{\Theta}^*_j = \Theta^*_j$.
5. Repeat steps 2-4 until $||\Theta^*_j - \Theta^*_{j-1}|| \leq \epsilon$, for some $\epsilon > 0$.

Procedure 4.1 is essentially a Newton-like approach to solving the nonlinear optimisation (4.14). Convergence results for such approaches are described in [90]. Since the optimal trajectories are not expected to change much from one time step to the next, $\hat{\Theta}^*_k$ is a good approximation of the (unknown) optimal state trajectory $\Theta^*_k$, which can be used to calculate a linear time-varying approximation to the cost function.

The LTV MPCC scheme is summarised below.

**Control Algorithm 4.2 (LTV Biaxial MPCC Algorithm).**

1. Set $k = 0$ and calculate $\hat{\Theta}^*_0$ using Procedure 4.1.
2. Linearise the predictions of $\hat{e}^c$ and $\hat{e}^l$ using (4.16).
3. Solve the constrained optimisation (4.19) to obtain $u^*_k$ and $v^*_k$.
4. Apply the first element of $u^*_k$ to the plant and update $\theta_k$ with the first element of $v^*_k$.
5. Calculate $\hat{\Theta}^*_k$ from $v^*_k$.
6. Increment $k$ and return to Step 2.
4.4 Guaranteed path completion

It is important to ensure that the system completes the path in finite time and that the
tracking error converges to zero as $k \to \infty$. This can be achieved by modifying Control
Algorithm \ref{algorithm:control} slightly, as described in the following.

Path completion in finite time is achieved by forcing $\theta_k$ to contract to zero. The contractive
MPC framework proposed by De Oliveira et al. \cite{30} is adopted, where a contraction
constraint is incorporated into the optimisation problem. The contractive MPC framework allows for guaranteed contraction of the path state without restricting the state to contract at every time step.

When contractive MPC is employed in discrete time, the contraction constraint remains
constant for groups of $N$ time steps \cite{80}. It is therefore useful to express the time step $k$
in terms of indices $l, m$ and the horizon length $N$ as follows:

$$ k = lN + m $$

(4.21)

where $l \in \{0, 1, ..., \infty\}$ and $m \in \{0, 1, ..., N - 1\}$. Table 4.1 shows how $l$ and $m$
evolve with $k$.

<table>
<thead>
<tr>
<th>$k$</th>
<th>0</th>
<th>1</th>
<th>$N - 1$</th>
<th>$N$</th>
<th>$N + 1$</th>
<th>$N_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$m$</td>
<td>0</td>
<td>1</td>
<td>$N - 1$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 4.1: Values of $l$ and $m$ for increasing $k$

Let $\alpha \in (0, N v_{max}]$ and constrain $|\theta_k|$ to contract according to

$$ |\theta_{(l+1)N,k}| \leq \begin{cases} 
|\theta_{lN}| - \alpha, & |\theta_{lN}| > \alpha, \\
0, & |\theta_{lN}| \leq \alpha,
\end{cases} $$

(4.22)

An example trajectory with contraction constraint is shown in Figure 4.3.

Note that the current time step $k = lN + m$ can be between 1 and $N$ time steps away
from time $(l + 1)N$, which is when the constraint is enforced. Hence the contraction
occurs at the end of a series of subhorizons whose length varies between 1 and $N$ time steps [30].

The contraction constraint (4.22) is incorporated into the LTV optimisation problem (4.19) as follows

$$\begin{align*}
\text{Minimise} & \quad J_k^a(\hat{\xi}_k), \\
\text{Subject to} & \quad \xi_{k+i,k} = A\xi_{k+i-1,k} + Bu_{k+i-1,k}, \\
& \quad \theta_{k+i,k} = \theta_{k+i-1,k} + v_{k+i-1,k}, \\
& \quad \xi_{k,k} = \xi_k, \quad \theta_{k,k} = \theta_k, \\
& \quad u_{k+i-1,k} \in \mathcal{U}, \quad v_{k+i-1,k} \in [0, v_{max}], \\
& \quad \xi_{k+i,k} \in \mathcal{X}, \quad \theta_{k+i,k} \in [\theta^*, 0], \\
& \quad |\theta_{(t+1)N,k}| \leq \begin{cases} 
|\theta_{tN}| - \alpha, & |\theta_{tN}| > \alpha, \\
0, & |\theta_{tN}| \leq \alpha.
\end{cases}
\end{align*}$$

(4.23)

The optimisation with contraction constraint (4.23) may still be formulated as a convex QP with only one additional linear constraint. Hence, the contraction constraint adds
very little computational effort to the control algorithm.

Another advantage of using the contractive MPC framework is that \( \theta_k \) is only required to contract every \( N \) time steps, as illustrated in Figure 4.3. In addition, \( \alpha \) can be chosen to be arbitrarily small. As a result, the effect of the contraction constraint on the behaviour of the controller is minimal. Alternatively, \( \alpha \) can be used to impose a minimum average path velocity over each interval of \( N \) time steps.

The modified optimisation (4.23) ensures that \( \theta_k \) reaches zero in finite time. It then remains to ensure that once \( \theta_k = 0 \), the system converges to \((x^d(0), y^d(0))\). It is assumed that there is an equilibrium point of (4.1) corresponding to \((x^d(0), y^d(0))\).

**Assumption 4.2.** There exists \( \xi^f \in \mathcal{X} \) such that \( C\xi^f = \begin{bmatrix} x^d(0) & y^d(0) \end{bmatrix}^T \) and \( A\xi^f = \xi^f \).

Observe that once \( \theta_k = 0 \), the control problem becomes one of regulation of the system states to \( \xi^f \) rather than contouring. Therefore, it is proposed to switch to an asymptotically stable regulating controller when \( \theta_k = 0 \), which is guaranteed to drive \( \xi_k \) to \( \xi^f \).

Conventional model predictive control is applied to regulate the states to \( \xi^f \). An error state \( \tilde{\xi} \) is defined as the difference between \( \xi \) and \( \xi^f \),

\[
\tilde{\xi} = \xi - \xi^f.
\] (4.24)

The cost function takes the form of a finite horizon linear quadratic regulator (LQR), penalising the error state \( \tilde{\xi} \) and control effort with weighting matrices \( Q^{LQR} \) and \( R^{LQR} \):

\[
J_k^{LQR} = \sum_{i=0}^{N-1} \left( \tilde{\xi}_{k+i,k}^T Q^{LQR} \tilde{\xi}_{k+i,k} + u_{k+i,k}^T R^{LQR} u_{k+i,k} \right) + J_f(\tilde{\xi}_{k+N,k}).
\] (4.25)

The last term \( J_f(\tilde{\xi}_{k+N,k}) \) is a terminal cost which forms one of the well-known “ingredients” for stabilising MPC, along with a terminal region \( X_f \) and a local controller \( \kappa_f(\tilde{\xi}) \) [82]. Asymptotic stability is guaranteed if \( X_f, \kappa_f() \) and \( J_f() \) satisfy certain conditions stated in [82], and if the predicted state at the end of the horizon lies in \( X_f \).

Following the approach of Scokaert and Rawlings [97], the local controller and terminal
cost are selected to be

\[ \kappa_f(\tilde{\xi}) = -K_f \tilde{\xi}, \]
\[ J_f(\tilde{\xi}_{k+N,k}) = \tilde{\xi}_{k+N,k}^T P_f \tilde{\xi}_{k+N,k}, \] (4.26)

where \( K_f \) is the state feedback gain matrix corresponding to the unconstrained LQR controller with weighting matrices \( Q^{LQR} \) and \( R^{LQR} \) and \( P_f \) is the solution of the Lyapunov equation

\[ P_f = Q^{LQR} + K_f^T R^{LQR} K_f + (A - BK_f)^T P_f (A - BK_f). \] (4.27)

Note that this choice of terminal cost corresponds to the unconstrained infinite horizon cost. The terminal region \( X_f \) is then the set of error states \( \tilde{\xi} \) where the control law \( u_k = \kappa_f(\tilde{\xi}_k) \) satisfies the constraints of (4.21). It is shown in [82, 97] that this selection of \( \kappa_f(\cdot), J_f(\cdot) \) and \( X_f \) satisfy the conditions required for asymptotic stability.

The optimisation problem solved at each time step for the LQR controller is then

Minimise \( J_k^{LQR} \),
Subject to \( \xi_{k+i+1,k} = A \xi_{k+i,k} + Bu_{k+i,k}, \)
\[ \xi_{k,k} = \xi_k, \quad u_{k+i,k} \in \mathcal{U}, \quad \xi_{k+i+1,k} \in \mathcal{X}, \] (4.28)

The LQR optimisation (4.28) can also be formulated as a convex QP, does not require any linearisation and involves less computation than the MPCC optimisation (4.23).

Stability of the constrained linear quadratic regulator is guaranteed only if the optimal predicted error state at the end of the horizon lies in \( X_f \). This is automatically satisfied if \( N \) is sufficiently large. Scokaert and Rawlings [97] propose an iterative scheme where \( N \) is increased until \( \tilde{\xi}_{k+N,k} \in X_f \) is satisfied, thereby guaranteeing stability of the control scheme. While the same approach could be adopted here, in the interests of fast computation it is assumed that \( \tilde{\xi}_{k+N,k} \in X_f \) with fixed \( N \).

**Assumption 4.3.** For all \( k \) where \( \theta_k = 0, \tilde{\xi}_{k+N,k}^* \in X_f \), where \( \tilde{\xi}_{k+N,k}^* \) denotes the error state prediction at the end of the horizon associated with the optimal solution to (4.28).
4.4 Guaranteed path completion

Under Assumption 4.3, the constrained linear quadratic regulator is guaranteed to drive $\xi_k$ to $\xi^f$ as $k \to \infty$. While this assumption may appear to be restrictive, in practice it can be easily satisfied with an appropriate choice of weights in the MPCC cost function (4.17).

A Control Algorithm for the modified LTV MPCC algorithm is given below. The controller switches to the constrained linear quadratic regulator when $\theta_k = 0$.

**Control Algorithm 4.3** (LTV Biaxial MPCC with guaranteed path completion).

1. Initialise $l = 0, m = 0, \theta = \theta_0$ and calculate $\hat{\Theta}_0^*$ using Procedure 4.1.
2. Linearise the predictions of $\hat{\epsilon}^c$ and $\hat{\epsilon}^l$ using (4.16) and $\hat{\Theta}_k^*$ where $k = lN_m$.
3. (a) If $\theta_k < 0$, solve the MPCC optimisation with contraction constraint (4.23) to obtain $u_k^*$ and $v_k^*$.
   (b) If $\theta_k = 0$, solve the constrained LQR optimisation (4.28) to obtain $u_k^*$ and set $v_k^* = 0$.
4. Apply the first element of $u_k^*$ to the plant and update $\theta_k$ with the first element of $v_k^*$.
5. Calculate $\hat{\Theta}_{k+1}^*$ from $v_k^*$.
6. If $m = N - 1$, set $m = 0$ and increment $l$. Otherwise, increment $m$. Return to Step 2.

By incorporating the contraction constraint into the optimisation and switching to a stable regulator at the end of the path, completion of the path and convergence of the states to $\xi^f$ are guaranteed.

**Theorem 4.1.** Let Assumptions 4.2 and 4.3 hold. Under Control Algorithm 4.3, the following hold:

1. There exists $k_f > 0$ such that $\theta_{k_f} = 0$.
2. $\epsilon^c_k, \epsilon^l_k \to 0$ as $k \to \infty$.

**Proof.** From the contraction constraint (4.22),

$$|\theta_{lN}| \leq \max(|\theta_0| - l\alpha, 0).$$  

(4.29)
Therefore, selecting \( k_f = \left\lceil \frac{|\theta_0|}{\alpha} \right\rceil N \) satisfies the first property.

Under Assumption 4.3, the well-known asymptotic stability results of [52, 77] apply and hence \( \xi_k \to \xi_f \) as \( k \to \infty \). By Assumption 4.2, this also implies that \( \epsilon^c, \epsilon^l \to 0 \) as \( k \to \infty \).

Control Algorithm 4.3 offers guaranteed path completion while preserving computational efficiency of LTV MPCC. In addition, since \( \alpha \) can be selected to be arbitrarily small, the desired behaviour of the controller is minimally effected. By switching to a stable regulator at the end of the path, convergence to the desired equilibrium point is also guaranteed.

### 4.5 Conclusion

A model predictive contouring control scheme has been developed for biaxial systems. The controller is based on minimisation of a cost function representing the competing objectives of maximising productivity while minimising contouring error. Available feedback is taken into account at each time step within a receding horizon framework.

A computationally efficient approximation is proposed where the controller implementation requires the solution of a single QP at each time step. Imposing a contraction constraint on \( \theta_k \) and switching to a stable regulator at \( \theta_k = 0 \) enforces path completion in finite time.

In Chapter 5, an application for biaxial MPCC is presented along with experimental results.
Chapter 5

Application of Biaxial MPCC to an X-Y Table

The biaxial model predictive contouring control scheme developed in Chapter 4 is applied to an X-Y table test rig. A first principles dynamic model of the X-Y table is developed and used to implement the controller on the test rig. Experimental results show the effect of varying the penalty weights, and a comparison with conventional tracking controllers demonstrates the effectiveness of the proposed approach.

A substantial proportion of this Chapter and the previous Chapter have been published in [66–68].

5.1 Test rig description

The X-Y table test rig is shown in Fig. 5.1. Each axis consists of a Baldor BSM50A-375AA brushless AC servomotor, flexible coupling, and Daedal 404100XR precision linear table with lead screw, as shown in Figure 5.2. The motors are controlled using two Baldor DBSC102 servo drives which receive $q$-axis (torque producing) current commands $i_{c,x}$ and $i_{c,y}$ at 1 ms intervals from the Target PC.

The servo drives employ proportional-integral (PI) controllers to control the $d$-axis currents to zero and the $q$-axis currents $i_x$, $i_y$ to the commands $i_{c,x}$, $i_{c,y}$ by applying the appropriate phase voltages to the motors. The current commands $i_{c,x}$, $i_{c,y}$ are subject to the constraints $|i_{c,x}|$, $|i_{c,y}| \leq 0.5$ A. Position feedback is available at 1ms intervals via
Application of biaxial MPCC to an X-Y Table

Figure 5.1: X-Y table test rig

Figure 5.2: X-Y table
5.2 X-Y table modelling

Implementation of MPCC requires a dynamic model of the X-Y table. A model in the form of (4.1) is derived from first principles, neglecting nonlinear friction. The model is then modified to include a disturbance model to account for disturbances and modelling errors along with a computational delay, to be used in the real-time implementation of MPCC on the X-Y table.
Each axis of the X-Y table is modelled as two rotational inertias connected by a flexible coupling, as shown in Fig. 5.4 for the X-axis.

\[ \ddot{\psi}_x = \frac{1}{J_m} \left( T_{m,x} + k_{c,x}(\varphi_x - \psi_x) + c_x(\dot{\varphi}_x - \dot{\psi}_x) - F_{m,x} \right), \]
\[ \ddot{\varphi}_x = \frac{1}{J_{l,x}} \left( k_{c,x}(\psi_x - \varphi_x) + c_x(\dot{\psi}_x - \dot{\varphi}_x) - F_{l,x} \right), \]
\[ \ddot{\psi}_y = \frac{1}{J_m} \left( T_{m,y} + k_{c,y}(\varphi_y - \psi_y) + c_y(\dot{\varphi}_y - \dot{\psi}_y) - F_{m,y} \right), \]
\[ \ddot{\varphi}_y = \frac{1}{J_{l,y}} \left( k_{c,y}(\psi_y - \varphi_y) + c_y(\dot{\psi}_y - \dot{\varphi}_y) - F_{l,y} \right), \] (5.1)

where for each axis, \( T_m \) is the motor torque, \( F_m \) and \( F_l \) are the motor and load friction torques respectively, \( J_m \) and \( J_l \) are the motor and load inertias respectively, and \( \psi \) and \( \varphi \) are the angular positions of the motor and load respectively. Note that the motor and coupling characteristics are identical in both axes, while the load properties differ. The linear displacements \( x \) and \( y \) are related to the angular displacements by

\[ x = \tau \varphi_x, \quad y = \tau \varphi_y, \] (5.2)

where \( \tau \) is the lead screw pitch. The motor torque is modelled as a linear function of motor current, so that

\[ T_{m,x} = K_t i_x, \quad T_{m,y} = K_t i_y, \] (5.3)

where \( K_t \) is the motor torque constant as specified in the motor datasheet. The PI current controller is assumed to be sufficiently fast so that \( i_x = i_{c,x} \) and \( i_y = i_{c,y} \).
The friction torques are given by

\[ F_{m,x} = b_{m,x} \dot{\psi}_x, \quad F_{l,x} = b_{l,x} \dot{\phi}_x, \]
\[ F_{m,y} = b_{m,y} \dot{\psi}_y, \quad F_{l,y} = b_{l,y} \dot{\phi}_y, \]

(5.4)

where \( b_m, b_l \) are the co-efficients of viscous friction for the motor and load respectively. It is known from system-identification experiments (see Appendix A) that the rig exhibits significant nonlinear friction which, along with other disturbances and modelling errors, is compensated by the inclusion of a constant output disturbance model in the prediction model, as discussed later in Section 5.2.

The continuous time equations (5.1) are discretised by applying a zero-order hold with sampling period \( h \). The linear discrete-time dynamic model can be expressed in state-space form

\[
\begin{bmatrix}
\xi_{k+1} \\
x_k \\
y_k
\end{bmatrix} = A_m \xi_k + B_m u_k, \\
\begin{bmatrix}
x_k \\
y_k
\end{bmatrix} = C_m \xi_k,
\]

\[ u_k \in [u_{\text{min}}, u_{\text{max}}], \quad (5.5) \]

where \( \xi = [\phi_x \ \dot{\phi}_x \ \psi_x \ \dot{\psi}_x \ \phi_y \ \dot{\phi}_y \ \psi_y \ \dot{\psi}_y]^T, \quad u = [i_{c,x} \ i_{c,y}]^T, \quad u_{\text{min}}, u_{\text{max}} \) are the constraints on the command currents, and \( A_m, B_m, \) and \( C_m \) are the discrete-time state matrices which are obtained from (5.1)-(5.4) and \( h \). The model parameters were identified using conventional system-identification methods and are summarised in Table 5.1.

Model validation experiments show that the model, with the addition of nonlinear friction, matches the behaviour of the X-Y table closely. Appendix A contains the numerical values of the state space matrices in (5.5), along with details regarding the X-Y table system identification and model validation.

Model predictive contouring control requires the solution of a constrained optimisation.
at each time step, which requires some time to compute. As a result, a 1-sample computational delay is introduced, which must be accounted for in the prediction model in order to maintain performance. Therefore, the X-Y table model (5.5) is modified to include a unit delay in the input.

Furthermore, due to parameter error and unmodelled dynamics such as nonlinear friction, it is expected that the X-Y table model (5.5) will exhibit significant modelling errors. These errors, if not compensated, will cause deterioration in performance. Hence, it is necessary to modify (5.5) by incorporating a disturbance model. There are a variety of techniques available for incorporating disturbance modelling in MPC [86]. The method chosen for the X-Y table is to convert (5.5) to a velocity form state-space model, where the new state and input represent the change in the original state and input, and augmenting the state vector with the system outputs $x$ and $y$. The new model incorporating both the unit delay and disturbance model is expressed as

$$
\begin{bmatrix}
\Delta x_{k+1} \\
\Delta y_{k+1} \\
\end{bmatrix}
= \begin{bmatrix}
A_m & B_m & 0 \\
0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\Delta x_k \\
\Delta y_{k-1} \\
\end{bmatrix}
+ \begin{bmatrix}
\Delta u_{k-1} \\
\Delta u_k \\
\end{bmatrix}
+ \begin{bmatrix}
0 \\
I \\
0
\end{bmatrix}
\Delta u_k.
$$

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Unit</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_m$ Motor inertia</td>
<td>kg $\cdot$ m$^2$</td>
<td>$2.3 \times 10^{-5}$</td>
</tr>
<tr>
<td>$J_l$ Table inertia</td>
<td>kg $\cdot$ m$^2$</td>
<td>$2.10 \times 10^{-5}$</td>
</tr>
<tr>
<td>$b_m$ Motor viscous friction</td>
<td>Nm $\cdot$ s/rad</td>
<td>$7.75 \times 10^{-6}$</td>
</tr>
<tr>
<td>$b_l$ Table viscous friction</td>
<td>Nm $\cdot$ s/rad</td>
<td>$9.44 \times 10^{-4}$</td>
</tr>
<tr>
<td>$K_t$ Motor torque constant</td>
<td>Nm/A</td>
<td>0.4364</td>
</tr>
<tr>
<td>$k_c$ Coupling stiffness</td>
<td>Nm/rad</td>
<td>2.67</td>
</tr>
<tr>
<td>$c$ Coupling damping</td>
<td>Nm $\cdot$ s/rad</td>
<td>5.00 $\times 10^{-4}$</td>
</tr>
<tr>
<td>$\tau$ Lead screw pitch</td>
<td>mm/rad</td>
<td>0.7958</td>
</tr>
</tbody>
</table>

Table 5.1: X-Y table model parameters
5.3 MPCC implementation

The model predictive contouring controller was implemented on the X-Y table test rig using MATLAB and xPC Target. The controller was programmed using Simulink and then compiled and downloaded to the Target PC for real-time execution.

Control Algorithm 4.3 was implemented with a sample period of $h = 4$ ms and horizon length $N = 7$. The controller operated with a 4 ms computational delay. This was incorporated into the model (5.6) which was used to compute state predictions, along with a constant output disturbance model, as described in Section 5.2. In order to guarantee path completion, the contraction constraint (5.22) was included with $\alpha = 10^{-3}$. As dis-
discussed in Section 4.4, at $\theta_k = 0$, the controller switched to a constrained LQ regulator with $Q^{LQR}$ and $R^{LQR}$ selected to be identity matrices.

Hildreth’s active set algorithm [126], implemented as an embedded MATLAB function, was used to solve the constrained QP at each time step, with input constraints $|i_{c,x}|, |i_{c,y}| \leq 0.5$ and $0 \leq v \leq 10^{-3}$, corresponding to current command saturation at $\pm 0.5$ A and a maximum path velocity of $0.25$ m/s. Hildreth’s algorithm involves calculating the Lagrange multipliers iteratively. In order to ensure that a solution to the QP is found within one sample period, the number of iterations used to find the Lagrange multipliers is limited to 10. It is possible that the constraints may be violated if the Lagrange multipliers do not converge within 10 iterations. Therefore, the input constraints are also applied to the optimal control inputs determined by the MPCC algorithm to prevent constraint violation occurring at the plant. It should be noted that more advanced optimisation algorithms than the implemented active set method could yield further reductions in computation time.

The path shape used for the experiments is the flower-shaped path used in Chapter 3, repeated in Figure 5.5, and was selected to demonstrate the behaviour of MPCC as its curvature varies along its length. It is expected that the MPCC controller will slow down as the tight curves are approached to maintain accuracy along the path. Tests were conducted for various values of the path speed weighting $q_\theta$, while keeping the other weightings constant with $q_l = 200$, $q_c = 100$ and $R = \text{diag}(60, 60, 200)$.

### 5.4 Benchmarking controllers

To establish a basis for comparison, existing tracking control schemes were also tested on the X-Y table. These include the current industry standard cascaded PI control and a model predictive control scheme as described in the literature (see Section 2.2.4). The desired path $(x^d(\theta), y^d(\theta))$ was converted to a reference trajectory $(x^d_k, y^d_k)$ by applying a constant path velocity. This is in contrast to MPCC, where the controller sets the path velocity automatically. The constant path velocity used to generate the reference
5.4 Benchmarking controllers

The trajectory was varied from 0.055 m/s to 0.140 m/s.

The cascaded PI and model predictive tracking control schemes are described briefly in the following sections.

5.4.1 Cascaded PI control

The current industry standard control scheme consists of an inner PI velocity loop nested inside an outer proportional position loop, as shown in Fig. 5.6.

Figure 5.5: Desired contour for X-Y table experiments

Figure 5.6: Cascaded PI tracking control scheme
The integrator state is restricted to the saturation limits to prevent integral windup. The PI gains were tuned manually such that the best possible performance was achieved at a path speed of 0.055 m/s, and then kept constant over the range of path speeds. The cascaded PI controller was implemented with sample periods of both 1 ms and 4 ms.

5.4.2 Model predictive tracking control

Model predictive control for tracking is discussed in Section 2.2. Model predictive tracking control is similar to MPCC in that it also involves minimisation of a cost function over a prediction horizon, and has the same constraint handling capabilities. However, MPTC seeks to minimise the axis tracking errors \((x_k - x_k^d, y_k - y_k^d)\) while the MPCC cost function focuses on contouring error. In addition, MPTC requires the path speed to be determined \textit{a priori} while MPCC automatically adjusts the path speed as part of the optimisation. The MPTC cost function is as follows:

\[
J_k^T = \sum_{i=1}^{N} \left[ \begin{array}{c} x_{k+i,k} - x_{k+i}^d \\ y_{k+i,k} - y_{k+i}^d \end{array} \right] ^T Q_t \left[ \begin{array}{c} x_{k+i,k} - x_{k+i}^d \\ y_{k+i,k} - y_{k+i}^d \end{array} \right] + \Delta u_{k+i-1,k}^T R_t \Delta u_{k+i-1,k} 
\] (5.8)

where \(Q_t\) and \(R_t\) are weighting matrices representing the relative importance of tracking accuracy and control deviations. The MPTC controller was implemented by minimising the cost function (5.8) subject to current saturation constraints of \(\pm 0.5\) A in a receding horizon fashion.

Similarly to the PI gains, weighting matrices \(Q_t\) and \(R_t\) were tuned manually to achieve the best possible performance at a path speed of 0.055 m/s. The horizon length for MPTC was chosen to be \(N = 7\), the same as was used for MPCC. Like MPCC, the model predictive tracking controller was implemented with a 4 ms sample period and 1-sample computational delay.
5.5 Experimental results

Figures 5.7, 5.8, 5.9, 5.10 and 5.11 show plots of the contour, command currents, path speed, and contouring error respectively for MPCC with $q_\theta = 0.03$ and $q_\theta = 3$. The contour plot in Figure 5.7 highlights the difference in contouring accuracy in various parts of the path for the two values of $q_\theta$. Figures 5.8 and 5.9 show that the controller is more aggressive for $q_\theta = 3$ compared to $q_\theta = 0.03$, and that the path is completed much more quickly when $q_\theta = 3$. It can be observed in Figure 5.10 that the controller reduces the path speed around the tight curves in the contour in order to maintain accuracy along the path. The effect of the cost function weights can be observed from Figures 5.10 and 5.11. For $q_\theta = 0.03$, the contouring error is much lower, but with a lower path speed compared to $q_\theta = 3$.

The traversal time and RMS contouring error for a range of values of $q_\theta$ between 0.03 and 3 is shown in Figure 5.12. It is observed that as the weighting $q_\theta$ is increased, the traversal time decreases and the contouring error increases.

Figure 5.13(a) shows the root mean square (RMS) contouring error versus traversal time for the three controllers, with contouring error expressed as a percentage of the maximum diameter of the contour shape. As expected, for all controllers the contouring accuracy improves for longer traversal times. Figure 5.13(b) shows the percentage of traversal time where one or more of the actuators was saturated. It can be observed that the sharp increase in contouring error for the cascaded PI and MPTC controllers occurs as the amount of saturation begins to increase.

Figure 5.13(a) shows that MPCC outperforms the cascaded PI controller operating at 4 ms and the MPTC controller over the complete range of traversal times. The performance of the cascaded PI controller operating at 1 ms is similar to MPCC at slow speeds, but deteriorates for shorter traversal times. Model predictive contouring control is shown to achieve superior performance at high speeds compared to the other three controllers, even one which is operating at a much faster sample rate. Unlike the other controllers, the accuracy of MPCC does not deteriorate rapidly as the amount
of saturation increases. The experimental results demonstrate the potential benefits of MPCC over the industry standard control scheme and more advanced methods such as MPTC.

Figure 5.7: MPCC contour for $q_0 = 0.03$ and $q_0 = 3$
5.5 Experimental results

Figure 5.8: Current commands vs. path parameter for $q_\theta = 0.03$ and $q_\theta = 3$

Figure 5.9: Current commands vs. time for $q_\theta = 0.03$ and $q_\theta = 3$
Figure 5.10: Path speed vs. path parameter for $q_\theta = 0.03$ and $q_\theta = 3$

Figure 5.11: Contour error vs. path parameter for $q_\theta = 0.03$ and $q_\theta = 3$
Figure 5.12: Effect of travel time weighting for MPCC on traversal time and contouring error
Figure 5.13: RMS contouring error (a) and percentage of time saturated (b) versus traversal time for PI, MPTC and MPCC
5.6 Conclusion

The biaxial model predictive contouring control scheme developed in Chapter 4 has been applied to an X-Y table. A dynamic model of the X-Y table was derived from first principles, incorporating a disturbance model and computation delay as required by the controller implementation.

Linear time-varying MPCC with guaranteed path completion was implemented in real-time on an X-Y table test rig with significant performance improvements over cascaded PI and model predictive tracking control schemes operating with constant path velocity. The results also demonstrate how the selection of penalty weights in the MPCC cost function is reflected in the behaviour of the controller.

Biaxial MPCC can be applied to a variety of biaxial contouring applications, such as laser profiling. However, many contouring applications have a more complicated structure, with a large number of axes. In the following Chapter, a multi-axis model predictive contouring control scheme is developed by generalising the concepts introduced in Chapter 4. Additionally, the control architecture for MPCC is modified to make it more suitable for industrial systems.
Chapter 6

Multi-axis Model Predictive Contouring Control with Industrial System Architecture

In this chapter, the model predictive contouring control scheme developed for biaxial systems in Chapter 4 is generalised to multi-axis systems. Multi-axis contouring involves additional complications as it requires consideration of manipulator forward kinematics. In addition, the definition of contouring error must be extended to include not only the position but also the orientation of the end effector.

The multi-axis model predictive contouring control scheme is developed with an emphasis on industrial contouring systems. Therefore, a multi-rate control architecture is proposed where the MPCC algorithm generates position command signals for each joint, subject to joint acceleration and jerk constraints. The position commands are then tracked by low level feedback controllers operating at a faster sample rate compared with the MPCC controller. In a further extension to biaxial contouring, state-dependent path speed constraints are considered, so that the bounds on path speed are allowed to vary along the path.

6.1 Industrial system architecture

The biaxial contouring control scheme proposed in Chapter 4 was successfully implemented on a laboratory test rig where MPCC was used to determine the motor current
commands in Chapter 5. Due to the computational requirements of MPCC, the current command signal could only be updated every 4 ms. In industrial systems, the current command signal is updated much more frequently, typically around every 500 µs. This is achieved by implementing low level joint controllers on the servo drives, and using the motion co-ordinator to determine a joint reference trajectory (see Figure 1.1 from Chapter 1). The servo drives, operating at a faster sample rate, accept position command signals from the motion co-ordinator and use feedback controllers to track the reference. This allows for improved control performance at the drive level, as feedback can be taken into account more frequently without placing a high computational burden on the motion co-ordinator.

It is intended to develop a multi-axis model predictive contouring control scheme suitable for implementation on industrial systems. Therefore, the MPCC architecture is modified to include low level joint controllers as would be present on industrial servo drives, as shown in Figure 6.1.

Figure 6.1: Multi-axis MPCC architecture

The model predictive contouring control algorithm is implemented on the motion co-ordinator, and generates the joint command acceleration $u_k \in \mathbb{R}^n$. The double integrator converts the acceleration $u_k$ into a joint position reference $q^d_k$ which is tracked by the joint controllers, operating at a faster sample rate compared to the model pre-
dictive contouring controller. The subscript \( t \) indicates that \( q^d_t \) is updated at a faster sample rate. This multi-rate architecture allows for position control to be implemented at sample rates comparable to those used in industrial systems without increasing the computational burden of the model predictive contouring control scheme. The model predictive contouring controller receives measurements of the joint position \( q_k \in \mathbb{R}^n \).

The joint command acceleration and jerk are subject to bound constraints, which may be reflected as constraints on \( u_k \) and \( \Delta u_k := u_k - u_{k-1} \):

\[
\begin{align*}
    u_{\text{min}} &\leq u_k \leq u_{\text{max}}, \\
    \Delta u_{\text{min}} &\leq \Delta u_k \leq \Delta u_{\text{max}}.
\end{align*}
\]  

(6.1)

Remark 6.1. In this Chapter, bound constraints on joint acceleration and jerk are considered, as these arise frequently in industrial applications. Alternative constraints may also be considered by replacing the double integrator in Figure 6.1 with a different model, and/or modifying (6.1).

6.2 The multi-axis contouring control problem

In order to develop a multi-axis version of model predictive contouring control, it is necessary to generalise the contouring control problem defined in Chapter 4 to multi-axis systems. The multi-axis contouring control problem is considerably more complicated than the biaxial problem. Both the position and orientation of the end effector must be considered when defining multi-axis contouring error, and in addition the joint positions \( q \) are related to end effector position and orientation via a nonlinear forward kinematics function.

Multi-axis contouring machines may be modelled as serial robotic manipulators, consisting of a series of links connected by joints [111]. The joint positions combined with the manipulator forward kinematics determine the position and orientation, or pose, of the end effector. The desired path is then expressed in terms of the end effector pose,
in contrast to biaxial contouring, where the desired path may be expressed directly in joint co-ordinates. The multi-axis contouring error is also defined with respect to the end effector.

It is well-known that serial robotic manipulators are governed by nonlinear equations of motion. However, as shown in Figure 6.1, multi-axis MPCC is implemented in conjunction with low level joint controllers. Provided the joint controllers do not saturate, the combined dynamics of the double integrator, joint controllers and plant may be approximately represented by a linear time-invariant model

\[ \xi_{k+1} = A\xi_k + Bu_k, \]
\[ q_k = C\xi_k, \]

(6.2)

where \( \xi \in \mathbb{R}^{n_s} \) denotes the system states. It is assumed that the acceleration and jerk constraints (6.1) prevent the joint controllers from saturating, thereby ensuring that the linear approximation of the joint dynamics remains reasonably accurate.

The plant model (6.2) is similar to the biaxial system model (4.1); however unlike (4.1), the joint vector \( q \) is not equal to the end effector co-ordinates. The machine’s kinematics, described in the following Section, relates the joint positions \( q \) to the position and orientation, or pose, of the end effector.

### 6.2.1 Forward kinematics

The pose of the machine’s end effector is determined by the forward kinematics, and is a nonlinear function of the joint positions:

\[
\begin{bmatrix}
R & o \\
0 & 1
\end{bmatrix}
= \hat{F}_K(q),
\]

(6.3)

where \( R \in \mathbb{R}^{3 \times 3} \) is the rotation matrix defining the orientation of the end effector and \( o \in \mathbb{R}^3 \) defines the position of the end effector in Cartesian space. It is convenient for
6.2 The multi-axis contouring control problem

control applications to express the end effector pose in a single vector \( z \in \mathbb{R}^{12} \):

\[
    z = \begin{bmatrix}
        o \\
        r_1 \\
        r_2 \\
        r_3
    \end{bmatrix},
\]

(6.4)

where \( r_1, r_2, r_3 \in \mathbb{R}^3 \) denote the columns of the rotation matrix \( \mathcal{R} \). Thus, a nonlinear function \( f_{FK} : \mathbb{R}^n \rightarrow \mathbb{R}^{12} \) is defined such that

\[
    z = f_{FK}(q).
\]

(6.5)

Refer to [111] for details regarding how to derive the forward kinematics for a particular manipulator configuration.

6.2.2 Multi-axis path function

The path function describes the desired position of the end effector pose vector \( z_k \), parameterised by \( \theta \):

\[
    z^d(\theta) : \mathbb{R} \rightarrow \mathbb{R}^{12}, \, \theta \in [\theta^s, 0], \, \theta^s < 0,
\]

\[
    z^d(\theta) = \begin{bmatrix}
        o^d(\theta) \\
        r^d_1(\theta) \\
        r^d_2(\theta) \\
        r^d_3(\theta)
    \end{bmatrix},
\]

(6.6)

where \( z^d(\theta) \) denotes the desired end effector pose along the path. Note that \( z^d(\theta) \) in (6.6) includes the desired position and orientation in three-dimensional space, in contrast to \( (x^d(\theta), y^d(\theta)) \) in (4.2) which includes only the desired position co-ordinates in the \( XY \)-plane.
In Chapter 4, the path function was assumed to be arc-length-parameterised, so that
\[
\frac{ds}{d\theta} = 1, \quad (6.7)
\]
where \( s \) is the distance travelled along the path, i.e.
\[
ds = \|dx^d, dy^d\| \quad (6.8)
\]
where \( \|.\| \) denotes the Euclidean norm. In multi-axis contouring, where the desired path includes both position and orientation of the end effector, the concept of arc length is somewhat more complicated, and depends on the specific application. For example, \( s \) could be defined as the linear distance travelled by the end effector:
\[
ds = \|d\alpha^d\|. \quad (6.9)
\]
However, \( s \) may also be defined differently. For example, in profile cutting, \( s \) may be defined as the distance travelled by the point where the cutting beam and the workpiece intersect (this corresponds to the cutting distance). This is in contrast to biaxial MPCC where \( s \) was defined using (6.8) for all applications. It is assumed that (6.7) holds for an appropriate definition of \( s \).

The path speed \( \dot{s} \) is usually subject to bound constraints. For example, in cutting applications, the path speed constraints arise due to surface finish requirements. In Chapter 4, the path speed constraint was assumed to be constant. However in industrial applications, the minimum and maximum path speed often changes as the path is traversed, as surface finish requirements may vary in particular segments of the path. To improve the applicability of multi-axis MPCC to industrial machines, the path speed constraint is generalised to a function of \( \theta \):
\[
0 \leq \dot{s}_{\min}(\theta) \leq \dot{s} \leq \dot{s}_{\max}(\theta). \quad (6.10)
\]
Note that (6.10) implies that path reversal is not allowed. Since \( \theta \in [\theta^\alpha, 0] \), it is necessary
that \( \hat{s}_{min}(0) = 0 \), to ensure feasibility at \( \theta = 0 \).

### 6.2.3 Multi-axis contouring error

The definition of contouring error introduced in Chapter 4 is generalised to multi-axis contouring. Multi-axis contouring error is defined as the minimum distance from the desired path with respect to the norm defined by a positive semi-definite matrix \( \tilde{Q}_c \in \mathbb{R}^{12 \times 12} \):

\[
\epsilon^c_k = \min_{\theta} \sqrt{(z_k - z^d(\theta))^T \tilde{Q}_c (z_k - z^d(\theta))}.
\] (6.11)

The selection of \( \tilde{Q}_c \) determines the relative weighting applied to the error in each element of \( z \). This allows the control designer to place more emphasis on position error relative to orientation error, or vice versa. As \( \tilde{Q}_c \) is only positive semi-definite, it is also possible to ignore position or orientation error in a particular direction. This is useful for profile cutting machines where rotation of the cutting beam about its axis does not affect cutting accuracy.

The magnitude of biaxial contouring error as defined in Chapter 4 can be expressed as the minimum Euclidean distance from the desired path:

\[
|\epsilon^c_k| = \min_{\theta} \sqrt{\begin{bmatrix} x_k - x^d(\theta) \\ y_k - y^d(\theta) \end{bmatrix}^T \begin{bmatrix} x_k - x^d(\theta) \\ y_k - y^d(\theta) \end{bmatrix}}.
\] (6.12)

It is clear that (6.12) is a special case of (6.11) with \( \tilde{Q}_c \) selected to be the identity matrix. Note that the multi-axis contouring error \( \epsilon^c_k \) is always positive, in contrast to the biaxial contouring error (4.3) defined in Chapter 4 which is positive on one side of the path and negative on the other. Since the cost function always penalises a quadratic function of contouring error, the sign of the contouring error does not affect the controller.
It should be noted that (6.11) implies that the orientation error vector $\epsilon^R$ is defined as

$$
\epsilon^R = \begin{bmatrix}
  r_1 - r_1^d \\
  r_2 - r_2^d \\
  r_3 - r_3^d
\end{bmatrix}.
$$

This is a naive approach, as $\epsilon^R$ does not have any physical meaning. A more meaningful way of defining the orientation error vector is to express the orientation using Euler angles \(^{17}\) instead of rotation matrices. However, using an Euler angle representation may be problematic, as it is well-known that for some rotations there is no uniquely defined set of Euler angles. Even though (6.13) lacks physical meaning, it is clear that small $\epsilon^R$ implies small physical difference in orientation, and therefore is considered an acceptable error definition for use in the cost function. In cutting applications, the control designer should select the matrix $\tilde{Q}_c$ so that the contouring error as defined in (6.11) corresponds well to the error measured at the workpiece. This is demonstrated in a simulation example presented in Chapter 7.

**Remark 6.2.** It may be possible to use the machine inverse kinematics to express the desired path, and therefore contouring error, in joint co-ordinates. However, the forward kinematics function $f_{FK}()$ is not invertible in general. In addition, as the position and orientation of the end effector is of more practical interest than the joint positions, it is more intuitive to express contouring error in terms of the end effector pose.

The multi-axis contouring control task is to select the joint inputs such that the machine’s end effector traverses near the desired path, minimising contouring error (6.11) while maximising path speed.

### 6.3 Multi-axis model predictive contouring control formulation

Using the multi-axis path function and contouring error derived in Section 6.2, the model predictive contouring controller developed in Chapter 4 is extended to multi-axis contouring as follows.
The evolution of the path parameter $\theta$ is governed by the dynamic equation (6.14) from Chapter 4, repeated here:

$$\theta_{k+1} = \theta_k + v_k,$$

(6.14)

where $v_k$ is a virtual input to be determined by the controller. It is clear from (6.14) and (6.15) that $v_k$ is proportional to a discrete-time approximation of the path speed $\dot{s}$. From (6.11), $v_k$ is subject to the following constraints,

$$v_{\min}(\theta_k) \leq v_k \leq v_{\max}(\theta_k),$$

(6.15)

where

$$v_{\min}(\theta) = h\dot{s}_{\min}(\theta), \quad v_{\max}(\theta) = h\dot{s}_{\max}(\theta).$$

(6.16)

Observe that $v_{\min}()$ and $v_{\max}()$ depend on $\theta$, allowing different path speed constraints to be specified for different sections of the path. This is in contrast to (4.7), where only a constant $v_{\max}$ was imposed.

In a similar manner to biaxial contouring, $\theta_k$, whose evolution is governed by (6.14), is used to develop an approximation to the contouring error (6.11).

**Assumption 6.1.**

$$\theta_k \approx \arg\min_{\theta} \sqrt{(z_k - z^d(\theta))^T \tilde{Q}_c (z_k - z^d(\theta))}.$$  

By Assumption 6.1, $\epsilon^c$ can be approximated by

$$\hat{\epsilon}_k^c = \sqrt{(z_k - z^d(\theta_k))^T \tilde{Q}_c (z_k - z^d(\theta_k))},$$

(6.17)
The multi-axis version of the cost function (6.10) is

\[
J_k = \sum_{i=1}^{N} \left( q_c (\hat{\epsilon}_{k+i,k})^2 - q_\theta \theta_{k+i,k} + \begin{bmatrix} \Delta v_{k+i-1,k} \\ \Delta u_{k+i-1,k} \end{bmatrix}^T R \begin{bmatrix} \Delta v_{k+i-1,k} \\ \Delta u_{k+i-1,k} \end{bmatrix} \right) \quad (6.18)
\]

where \( R = \text{diag}(r_v, r_u) \) and \( r_u \in \mathbb{R}^{n_u}, r_v, q_c, q_\theta > 0 \). The penalty weightings \( q_c, q_\theta, r_u \) and \( r_v \) specify the relative importance of contouring accuracy, productivity and smoothness of the path speed and joint acceleration.

**Remark 6.3.** Assumption 6.1 is enforced by setting the contouring error weighting \( q_c \) sufficiently large in the cost function.

In Chapter 4, it was convenient to separate the normal (contouring error) and tangential (lag) components of the tracking error using (4.8) and (4.9). Then, Assumption 4.1, which is equivalent to Assumption 6.1, could be enforced by setting the penalty weighting on the tangential component sufficiently large.

In multi-axis contouring, it is not straightforward to separate the normal and tangential components, so Assumption 6.1 is enforced by setting the penalty weighting on both normal and tangential components (combined together in \( \hat{\epsilon} \)) sufficiently large.

Let \( Q_c = q_c \tilde{Q}_c \) and

\[
\tilde{z}_k = z_k - z^d(\theta_k). \tag{6.19}
\]

Then, combining (6.17) and (6.19), the cost function (6.18) becomes

\[
J_k = \sum_{i=1}^{N} \left( \tilde{z}_{k+i,k}^T Q_c \tilde{z}_{k+i,k} \right) - q_\theta \theta_{k+i,k} + \begin{bmatrix} \Delta v_{k+i-1,k} \\ \Delta u_{k+i-1,k} \end{bmatrix}^T R \begin{bmatrix} \Delta v_{k+i-1,k} \\ \Delta u_{k+i-1,k} \end{bmatrix} \quad (6.20)
\]

The nonlinear optimisation problem (4.14) extended to multi-axis contouring is stated below:

**Minimise** \( J_k \),
Subject to  
\[ \xi_{k+i,k} = A\xi_{k+i-1,k} + Bu_{k+i-1,k}, \]
\[ \theta_{k+i,k} = \theta_{k+i-1,k} + v_{k+i-1,k}, \]
\[ \xi_{k,k} = \xi_k, \quad \theta_{k,k} = \theta_k, \]
\[ u_{k+i-1,k} \in [u_{\text{min}}, u_{\text{max}}], \quad \Delta u_{k+i-1,k} \in [\Delta u_{\text{min}}, \Delta u_{\text{max}}], \]
\[ v_{k+i-1,k} \in [v_{\text{min}}(\theta_{k+i-1}), v_{\text{max}}(\theta_{k+i-1})], \]
\[ \theta_{k+i,k} \in [\theta^*, 0], \quad i = 1, ..., N. \quad (6.21) \]

The optimisation (6.21) may be modified to suit particular applications. For example, other control objectives, such as the minimisation of control effort, may easily be incorporated into the cost function, and additional constraints on the system states may be imposed.

As discussed in Chapter 4, the nonlinear optimisation (6.21) involves too much computation to be practical for real-time implementation. In the following section, the linear time-varying approximation presented in Section 4.3 is used to reduce the computational burden of multi-axis MPCC.

### 6.4 Linear time-varying multi-axis MPCC

The linear time-varying MPCC formulation developed in Section 4.3 can also be extended to multi-axis MPCC, where the optimisation problem (6.21) is approximated with a convex quadratic program (QP).

In order to approximate (6.21) with a QP, it is necessary to approximate the cost function (6.20) with a quadratic function of the control input trajectories, and also replace the state-dependent path speed constraint (6.15) with an approximate time-dependent constraint.
6.4.1 Cost function approximation

An approximate cost function is developed by linearising both the path function \( z^d() \) and forward kinematics function \( f_{FK}() \) about estimated state trajectories. Given estimates of the state trajectories \( \hat{\Xi}_k^* = \{ \hat{\xi}_{k,k}^*, \ldots, \hat{\xi}_{k+N,k}^* \}, \hat{\Theta}_k^* = \{ \hat{\theta}_{k,k}^*, \ldots, \hat{\theta}_{k+N,k}^* \} \), where the notation \( \hat{\theta}_{k+i,k}^* \) represents the estimated optimal state \( \hat{\theta}^* \) predicted at time \( k+i \) using information available at time \( k \), define linear approximations of \( z \) and \( z^d \) as follows:

\[
\begin{align*}
z^a_{k+i}(\xi_{k+i,k}, \hat{\Xi}_k^*) &= f_{FK}(C\hat{\xi}_{k+i,k}^*) + \nabla f_{FK}(C\hat{\xi}_{k+i,k}^*)(C(\xi_{k+i,k} - \hat{\xi}_{k+i,k}^*)), & (6.22) \\
z^{a,d}_{k+i}(\theta_{k+i,k}, \hat{\Theta}_k^*) &= z^d(\hat{\theta}_{k+i,k}^*) + \nabla z^d(\hat{\theta}_{k+i,k}^*)(\theta_{k+i,k} - \hat{\theta}_{k+i,k}^*), & (6.23)
\end{align*}
\]

where for multi-axis MPCC it is necessary to linearise the forward kinematics function \( f_{FK}() \) as well as the path function \( z^d() \). This is in contrast to the linearisation used in Chapter 4, where it is sufficient to linearise only the path function (see (4.15)).

As stated in [12], the derivative of the forward kinematics function \( \nabla f_{FK}(q) \) can be computed from

\[
\nabla f_{FK}(q) = - \begin{bmatrix} S(r_1(q)) \\ S(r_2(q)) \\ S(r_3(q)) \end{bmatrix} J_{FK}(q),
\]

where \( J_{FK}(q) \) is the geometric Jacobian for the manipulator and \( S() \) is defined such that given a vector \( a = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}^T \),

\[
S(a) = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}.
\]

(6.25)

Equations (6.22) and (6.23) are combined to form an approximation to \( \tilde{z} \),

\[
\tilde{z}^a_{k+i,k}(\hat{\Xi}_k^*, \hat{\Theta}_k^*) = z^a_{k+i}(\xi_{k+i,k}, \hat{\Xi}_k^*) - z^{a,d}_{k+i}(\theta_{k+i,k}, \hat{\Theta}_k^*). 
\]

(6.26)
where the dependence of $\tilde{z}^a$ on the approximate state trajectories $\hat{\Xi}_k^*, \hat{\Theta}_k^*$ is emphasised in the notation.

An approximate cost function can then be expressed using (6.26):

$$J_k^a(\hat{\Xi}_k^*, \hat{\Theta}_k^*) = \sum_{i=1}^N (\tilde{z}^a_{k+i,k}(\hat{\Xi}_k^*, \hat{\Theta}_k^*)^T Q \tilde{z}^a_{k+i,k}(\hat{\Xi}_k^*, \hat{\Theta}_k^*) - q_\theta \theta_{k+i,k} + \begin{bmatrix} \Delta u_{k+i-1,k} \\ \Delta v_{k+i-1,k} \end{bmatrix}^T R \begin{bmatrix} \Delta u_{k+i-1,k} \\ \Delta v_{k+i-1,k} \end{bmatrix}).$$  (6.27)

The approximate cost function (6.27) is quadratic and convex in the input trajectories $u_k$ and $v_k$. Estimates of both the joint and path state trajectories $\hat{\Xi}_k^*, \hat{\Theta}_k^*$, are required to form the approximate cost function (6.27), in contrast to the biaxial LTV formulation from Section 4.3, where only $\hat{\Theta}_k^*$, is required.

Clearly, the approximation of the cost function depends on the accuracy of the state trajectory estimates $\hat{\Xi}_k^*$ and $\hat{\Theta}_k^*$. A methodology for computing the state trajectory estimates will be discussed in Section 6.4.3.

### 6.4.2 Constraint approximation

In order to simplify the optimisation problem (6.21), the state-dependent constraint (6.15) is approximated with a time-dependent constraint, so that over the horizon,

$$v_{k+i-1,k} \in [\hat{v}_{\text{min}}(k+i-1,k), \hat{v}_{\text{max}}(k+i-1,k)].$$  (6.28)

The approximate bounds $\hat{v}_{\text{min}}(k+i-1,k), \hat{v}_{\text{max}}(k+i-1,k)$ are computed using the estimate of the optimal path state trajectory, $\hat{\Theta}_k^*$:

$$\hat{v}_{\text{min}}(k+i-1,k, \hat{\Theta}_k^*) = v_{\text{min}}(\hat{\theta}_{k+i-1,k}),$$

$$\hat{v}_{\text{max}}(k+i-1,k, \hat{\Theta}_k^*) = v_{\text{max}}(\hat{\theta}_{k+i-1,k}),$$

$$i \in \{1, \ldots, N\}.  $$ (6.29)
As the time-dependent constraint (6.28) varies across the horizon, the controller can anticipate changes to the minimum and maximum path speed and act accordingly. In addition, the estimated trajectory $\hat{\Theta}^*_k$ is already used to obtain the approximate cost function (6.27), so there is little additional computation required.

Using (6.27), (6.28) and (6.29), a simplified optimisation problem can be posed:

Minimise $J^u_k(\hat{\Xi}^*_k, \hat{\Theta}^*_k)$,

Subject to

\[
\begin{align*}
\xi_{k+i,k} &= A\xi_{k+i-1,k} + Bu_{k+i-1,k}, \\
\theta_{k+i,k} &= \theta_{k+i-1,k} + v_{k+i-1,k}, \\
\xi_{k,k} &= \xi_k, \quad \theta_{k,k} = \theta_k, \\
u_{k+i-1,k} &\in [u_{\text{min}}, u_{\text{max}}], \\
\Delta u_{k+i-1,k} &\in [\Delta u_{\text{min}}, \Delta u_{\text{max}}], \\
v_{k+i-1,k} &\in [\hat{v}_{\text{min}}(k+i-1, \hat{\Theta}^*_k), \hat{v}_{\text{max}}(k+i-1, \hat{\Theta}^*_k)], \\
\theta_{k+i,k} &\in [\theta^a, 0], \quad i = 1, \ldots, N.
\end{align*}
\] (6.30)

6.4.3 Trajectory approximation

Computing the linearised predictions (6.22) and (6.23) as well as the approximate path speed constraints (6.29) requires estimates of the state trajectories $\hat{\Xi}^*_k$ and $\hat{\Theta}^*_k$. Using the same approach as in Section 4.3, state trajectories are estimated using the optimal input trajectories from the previous time step, as described in the following.

Let $v^*_k$ and $u^*_k$ denote the optimal input trajectories obtained by solving (6.31) at time $k - 1$. An approximate virtual input trajectory for the current time step can be obtained by truncating $v^*_k$ and appending a feasible input $\hat{v}^*_k \equiv \min(v^*_{k+N-2,k-1}, \theta^*_k|v^*_{k+N-2,k-1})$ so that $\hat{v}^*_k = \{v^*_k, \ldots, v^*_{k+N-2,k-1}, \hat{v}^*_k, \ldots\}$. Similarly, an approximate plant input trajectory is computed by truncating $u^*_k$ and appending $\hat{u}^*_k \equiv u^*_{k+N-2,k-1}$, yielding $\hat{u}^*_k = \{u^*_k, \ldots, u^*_{k+N-2,k-1}, \hat{u}^*_k, \ldots\}$. The approximate state trajectories $\hat{\Theta}^*_k$ and $\hat{\Xi}^*_k$ are computed by applying $\hat{v}^*_k$ and $\hat{u}^*_k$ to (6.14) and (6.2) with initial conditions $\hat{\theta}^*_k = \theta_k$ and $\hat{\xi}_k = \xi_k$. 
At the initial time step \( k = 0 \), \( \hat{\Theta}_0^* \) and \( \hat{\Xi}_0^* \) can be computed via Procedure 6.1, which is analogous to Procedure 4.1.

**Procedure 6.1 (Initial trajectory estimation).**

1. Initialise \( \hat{\Theta}_0^* \) to \( \hat{\Theta}_0^0 = \{\theta_0, \theta_0, \ldots, \theta_0\} \), \( \hat{\Xi}_0^* \) to \( \hat{\Xi}_0^0 = \{\xi_0, \xi_0, \ldots, \xi_0\} \) and set \( j = 0 \).

2. Compute the linearised error function \( \tilde{z}_{a,k+i,k}^a(\hat{\Theta}_0^* j, \hat{\Xi}_0^* j) \) using (6.22)-(6.26).

3. Calculate the approximate path speed constraints \( \hat{v}_{\min}, \hat{v}_{\max} \) from (6.29) using \( \hat{\Theta}_0^* j \).

4. Solve the optimisation (6.30) to obtain \( u_0^{*j} \) and \( v_0^{*j} \). Compute \( \Theta_0^{*j} \) and \( \Xi_0^{*j} \) by applying \( v_0^{*j}, u_0^{*j} \) to (6.14), (6.2).

5. Set \( \hat{\Theta}_0^{*j+1} = \Theta_0^{*j}, \hat{\Xi}_0^{*j+1} = \Xi_0^{*j} \) and increment \( j \).

6. Repeat steps 2-5 until \( ||\hat{\Theta}_0^{*j} - \hat{\Theta}_0^{*j-1}|| + ||\hat{\Xi}_0^{*j} - \hat{\Xi}_0^{*j-1}|| \leq \epsilon \), for some \( \epsilon > 0 \), or until some iteration limit is exceeded.

The estimated trajectories \( \hat{\Theta}_k^* \) and \( \hat{\Xi}_k^* \) are used to calculate linear time-varying approximations to the cost function and path speed constraints. The accuracy of the linear time-varying approximation depends on the extent that solutions to the optimisation (6.31) vary from one time step to the next. If the horizon is sufficiently long, it can be expected that the LTV approximation is reasonably accurate.

As the state-dependent constraint (6.15) is replaced by the approximate time-dependent constraint (6.28), it is important to ensure that the solution to the simplified optimisation problem (6.31) does not violate the original constraint (6.15). Recall that \( \hat{\theta}_{k,k}^* = \theta_k \).

It follows that

\[
\begin{align*}
\hat{v}_{\min}(k; \hat{\Theta}_k^*) &= v_{\min}(\hat{\theta}_{k,k}^*) = v_{\min}(\theta_k), \\
\hat{v}_{\max}(k; \hat{\Theta}_k^*) &= v_{\max}(\hat{\theta}_{k,k}^*) = v_{\max}(\theta_k),
\end{align*}
\]

which satisfies (6.15). Since only the first element of \( v_k \) is ever applied, (6.15) is satisfied for all \( k \).

**Remark 6.4.** In order to satisfy (6.15), it is only necessary to enforce (6.28) for \( i = 1 \). Imposing the estimated constraint across the horizon is important only from a perfor-
6.4.4 Guaranteed path completion

In Chapter 4, path completion was enforced by introducing a contraction constraint on $\theta$ and switching to a stable regulator when $\theta$ reached zero. The same approach can be applied directly to multi-axis contouring. The following assumptions, analogous to Assumptions 4.2 and 4.3, are required.

**Assumption 6.2.** There exists $\xi^f \in \mathbb{R}^{n_x}$ such that $f_{FK}(C\xi^f) = z^d(0)$ and $A\xi^f = \xi^f$.

**Assumption 6.3.** For all $k$ where $\theta_k = 0$, $\hat{\xi}_{k+N,k}^* \in X_f$, where $\hat{\xi}_{k+N,k}^* = \xi_{k+N,k}^* - \xi^f$ denotes the error state prediction at the end of the horizon associated with the optimal solution to (6.33) and $X_f$ is defined as per Section 4.4.

The contraction constraint (4.22) is incorporated into the multi-axis optimisation problem

Minimise $J^a_k(\hat{\xi}^*, \hat{\Theta}^*)$,

Subject to

$\xi_{k+i,k} = A\xi_{k+i-1,k} + B_{k+i-1,k}$,

$\theta_{k+i,k} = \theta_{k+i-1,k} + v_{k+i-1,k}$,

$\xi_k, \theta_k, \theta_{k,k} = \theta_k$,

$u_{k+i-1,k} \in [u_{\min}, u_{\max}]$, $\Delta u_{k+i-1,k} \in [\Delta u_{\min}, \Delta u_{\max}]$,

$v_{k+i-1,k} \in [\hat{\nu}_{\min}(k+i-1, \hat{\Theta}_k^*), \hat{\nu}_{\max}(k+i-1, \hat{\Theta}_k^*)]$,

$\theta_{k+i,k} \in [\theta^*, 0]$,

$|\theta_{l+1,N,k}| \leq \begin{cases} |\theta_{l+1,N}^*| - \alpha, & |\theta_{l+1,N}^*| > \alpha, \\ \alpha, & |\theta_{l+1,N}^*| \leq \alpha, \end{cases}$

$i = 1, \ldots, N,$

(6.32)

where $\alpha \in (0, N \min_\theta v_{\max}(\theta)]$. When $\theta$ reaches zero, control is switched to a classical model predictive regulator where the following optimisation problem, analogous to
6.4 Linear time-varying multi-axis MPCC

(6.28), is solved at each time step in a receding horizon fashion:

\[
\begin{align*}
\text{Minimise} & \quad J_{k}^{LQR} = \sum_{i=0}^{N-1} \left( \tilde{\xi}_{k+i,k}^T Q_{LQR} \tilde{\xi}_{k+i,k} + u_{k+i,k}^T R_{LQR} u_{k+i,k} \right) + \tilde{\xi}_{k+N,k}^T P_f \tilde{\xi}_{k+N,k}, \\
\text{Subject to} & \quad \xi_{k+i+1,k} = A \xi_{k+i,k} + B u_{k+i,k}, \\
& \quad \xi_{k,k}, u_{k+i,k} \in [u_{min}, u_{max}], \Delta u_{k+i,k} \in [\Delta u_{min}, \Delta u_{max}],
\end{align*}
\]

(6.33)

where \( Q_{LQR} \), \( R_{LQR} \) and \( P_f \) are defined as per Section 4.4.

The complete proposed multi-axis LTV MPCC algorithm is as follows.

**Control Algorithm 6.1** (LTV multi-axis MPCC with guaranteed path completion).

1. Initialise \( l = 0, m = 0, \theta = \theta_0 \) and calculate \( \hat{\Theta}_0^*, \hat{\Xi}_0 \) using Procedure 6.1.
2. Compute the linearised error function \( \tilde{z}_k^a, \hat{\Theta}_0^*, \hat{\Xi}_0^j \) using (6.22)-(6.26).
3. Calculate the approximate path speed constraints \( \hat{\theta}_{min}, \hat{\theta}_{max} \) from (6.24) using \( \hat{\Theta}_k^* \) where \( k = lN + m \).
4. (a) If \( \theta_k < 0 \), solve the approximate MPCC optimisation (6.31) to obtain optimal input trajectories \( u_k^*, v_k^* \).
   (b) If \( \theta_k = 0 \), solve the constrained LQR optimisation (6.33) to obtain \( u_k^* \) and set \( v_k^* = 0 \)
5. Apply the first element of \( u_k^* \) to the plant and use the first element of \( v_k^* \) to update \( \theta_{k+1} \) via (6.13).
6. Calculate \( \hat{u}_{k+1}, \hat{v}_{k+1} \) by truncating \( u_k^* \), \( v_k^* \) and compute \( \hat{\Theta}_{k+1}^* \) and \( \hat{\Xi}_{k+1}^* \) by applying \( \hat{v}_{k+1}^* \), \( \hat{u}_{k+1}^* \) to (6.2), (6.14).
7. If \( m = N - 1 \), set \( m = 0 \) and increment \( l \). Otherwise, increment \( m \). Return to step 2.

Control Algorithm 6.1 guarantees path completion in finite time, and convergence to \( \xi_f \) as \( k \to \infty \).

**Theorem 6.1.** Let Assumptions 6.2 and 6.3 hold. Under Control Algorithm 6.1, the following hold:

1. There exists \( k_f > 0 \) such that \( \theta_{k_f} = 0 \).
2. $||z_k - z^d(\theta_k)|| \to 0$ as $k \to \infty$.

\textit{Proof.} The proof follows the same steps as the proof of Theorem 4.1.

\section{Conclusion}

The MPCC scheme developed in Chapter 4 has been extended to multi-axis systems with a view towards industrial implementation. The control architecture has been modified to include low level joint controllers which are typically used in industrial machines. In the proposed architecture, the MPCC algorithm generates a joint command acceleration signal which is converted into a joint position reference. This is then tracked by the joint controllers operating at a faster sample rate compared to MPCC.

In order to adapt the biaxial model predictive contouring control formulation to the more complicated problem of multi-axis contouring, the concept of contouring error was generalised to include both position and orientation error, and is expressed in terms of the end effector pose. The end effector pose is related to the joint positions via the forward kinematics. A limitation of the multi-axis definition of contouring error is that the orientation error does not have a physical meaning. However, it will be shown in a simulation example in Chapter 5 that the matrix $\tilde{Q}$ may be selected such that the contouring error corresponds well to the error measured at the workpiece.

State-dependent path speed constraints were also introduced, allowing for a different maximum path speed to be assigned to different segments of the path.

In the following Chapter, the linear time-varying multi-axis model predictive contouring control algorithm is applied to a simulation model of an industrial profile cutting machine.
Chapter 7

Application of Multi-axis MPCC to an Industrial Machine Model

HAVING developed the multi-axis model predictive contouring control algorithm, in this chapter the approach is applied to an industrial machine model. The simulation model is based on a 5-axis profile cutting machine with a combination of translational and rotational axes.

Applying the multi-rate architecture proposed in Chapter 6, low level joint controllers are implemented using cascaded PI control, as would most likely be used in industrial servo drives. A model of the closed-loop dynamics is identified and used as the prediction model for MPCC. Measurements of joint position are received as feedback by the MPCC controller.

Simulations are carried out over a range of penalty weights. Results demonstrate that the selection of the cost function weights has the expected effect on contouring behaviour.

7.1 Machine description

Control Algorithm 6.1 is implemented on a simulation model of an industrial profile cutting machine. A beam is released from the end effector which cuts into a workpiece lying in the X-Y plane. The machine features three translational axes as well as two rotational axes. This facilitates bevelled cutting, where the cutting beam is not perpen-
dicular to the workpiece. The three translational axes allow motion in the left/right, fore/aft and up/down directions respectively. The two rotational axes are referred to as the swivel and tilt joints, which are shown in Figure 7.1.

![Figure 7.1: Cutting head with two rotational axes](image)

The profiling machine may be treated as a serial robotic manipulator, comprised of three prismatic joints and two revolute joints, connected in a kinematic chain. The Denavit-Hartenberg (D-H) parameters corresponding to the configuration of the cutting machine are given in Table 7.1.

<table>
<thead>
<tr>
<th>Link #</th>
<th>Description</th>
<th>$\theta$</th>
<th>$d$</th>
<th>$a$</th>
<th>$\alpha$</th>
<th>Joint type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Left/Right (X)</td>
<td>$\pi$</td>
<td>0</td>
<td>0</td>
<td>$\pi$/2</td>
<td>Prismatic</td>
</tr>
<tr>
<td>2</td>
<td>Fore/Aft (Y)</td>
<td>$\pi$/2</td>
<td>0</td>
<td>0</td>
<td>$\pi$/2</td>
<td>Prismatic</td>
</tr>
<tr>
<td>3</td>
<td>Up/Down (Z)</td>
<td>0</td>
<td>$q_3^*$</td>
<td>0</td>
<td>0</td>
<td>Prismatic</td>
</tr>
<tr>
<td>4</td>
<td>Swivel</td>
<td>$q_4^*$</td>
<td>0</td>
<td>0</td>
<td>$-\pi$/4</td>
<td>Revolute</td>
</tr>
<tr>
<td>5</td>
<td>Tilt</td>
<td>$q_5^*$</td>
<td>0</td>
<td>0</td>
<td>$\pi$/4</td>
<td>Revolute</td>
</tr>
</tbody>
</table>

Table 7.1: D-H parameters for profile cutting machine

A diagram showing the co-ordinate frame assignment corresponding to the D-H parameters given in Table 7.1 is given in Figure 7.2.
Figure 7.2: Symbolic diagram of profile cutting machine with Denavit-Hartenberg (D-H) co-ordinate systems
7.1.1 Contouring error

The desired path function \( z_d() \) completely defines the position and orientation of the end effector. However, for profile cutting, rotation about the cutting beam is unimportant. In fact, as the profile cutting machine only possesses five degrees of freedom, it is not possible to achieve all orientations.

For this application, it is only necessary to consider orientation errors in the third column of the rotation matrix. This is achieved by setting some elements of the matrix \( \tilde{Q}_c \) in (6.11) to be zero. For the simulations conducted, \( \tilde{Q}_c \) was defined as

\[
\tilde{Q}_c = \text{diag}(1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0),
\]

where the end effector position, corresponding to the first three elements of \( z \), is expressed in millimetres. This choice of \( \tilde{Q}_c \) penalises orientation errors only in the last column of the rotation matrix, so that any rotation about the cutting beam in the desired path function is ignored.

7.2 Simulation model

A simulation model of the profile cutting machine was developed using the Robotics Toolbox for MATLAB [25]. The Left/Right and Fore/Aft joints are modelled using two inertias connected by a flexible coupling, as it is known that these joints exhibit significant amounts of compliance, while the other three joints may be reasonably modelled as rigid. This is illustrated in Figure 7.3.

The motor equation of motion for the first and second joints may be expressed as

\[
\frac{J^m_1}{N_1^G} \ddot{q}_1^m = T_1^m - \frac{b_1^m}{N_1^G} \dot{q}_1^m - N_1^G \left( k_c^1 (q_1^m - q_1) + c_1 (\dot{q}_1^m - \dot{q}_1) \right),
\]

\[
\frac{J^m_2}{N_2^G} \ddot{q}_2^m = T_2^m - \frac{b_2^m}{N_2^G} \dot{q}_2^m - N_2^G \left( k_c^2 (q_2^m - q_2) + c_2 (\dot{q}_2^m - \dot{q}_2) \right),
\]

where for the \( i \)-th joint
7.2 Simulation model

Figure 7.3: Two-inertia joint model for profile cutting machine

- \(q^m_i\) is the motor position, expressed in (m) for prismatic joints or (rad) for revolute joints,
- \(q_i\) is the joint position, (m) or (rad),
- \(J^m_i\) is the motor inertia (kg \(m^2\)),
- \(T^m_i\) is the motor torque (Nm),
- \(b^m_i\) is the motor coefficient of viscous friction (Nm s/rad),
- \(k^c_i\) is the joint referred coupling stiffness (N/m) or (Nm/rad),
- \(c_i\) is the joint referred coupling damping co-efficient (N s/m) or (Nm s/rad),
- \(N^G_i\) is the equivalent gear ratio (m/rad) or (rad/rad).

As the last three joints are modelled as rigid, their motor dynamics are neglected and the motor positions \(q^m_3, q^m_4\) and \(q^m_5\) are equal to the joint positions \(q_3, q_4\) and \(q_5\):

\[
\begin{align*}
q_3 &= q^m_3, \\
q_4 &= q^m_4, \\
q_5 &= q^m_5. \\
\end{align*}
\] (7.3)

The vector of torques \(T^r := \begin{bmatrix} T^r_1 & T^r_2 & T^r_3 & T^r_4 & T^r_5 \end{bmatrix}^T\) applied to the joints is then

\[
\begin{align*}
T^r_1 &= k^c_1(q^m_1 - q_1) + c_1(\dot{q}^m_1 - \dot{q}_1), \\
T^r_2 &= k^c_2(q^m_2 - q_2) + c_2(\dot{q}^m_2 - \dot{q}_2),
\end{align*}
\]
The Robotics Toolbox [25] is used to simulate the dynamics of the serial link robot with $T^r$ as the applied torque, and uses the recursive Newton-Euler algorithm to compute the forward dynamics. The equation of motion for the serial link robot may be written as

$$ T^r = M^r(q)\ddot{q} + C^r(q, \dot{q})\dot{q} + F^r(\dot{q}) + G^r(q), $$

(7.5)

where

- $M^r(q)$ is the joint space inertia matrix,
- $C^r(q, \dot{q})$ is the Coriolis and centripetal coupling matrix,
- $F^r(\dot{q})$ is the viscous friction torque and
- $G^r(q)$ is the load due to gravity.

Refer to Appendix B for further details regarding the simulation model.

### 7.3 Joint controllers

As shown in Figure 6.1, the proposed control architecture for multi-axis MPCC includes low level joint controllers which are used to track the joint position command. For the simulations presented in this Chapter, the joint controllers are implemented as the industry standard cascaded PI controllers, as shown in Figure 7.4 for the $i$-th joint. For simulation purposes, the motor electrical dynamics are neglected, so that the joint controllers produce the applied motor torque $T^m$ directly.

Measurements of motor position $q^m$ are provided as feedback to the joint controllers. Therefore, the joint controllers are unable to compensate for any discrepancy between
motor position and joint position caused by the compliance in joints 1 and 2.

\[ q_i^d \rightarrow \text{P} \rightarrow q_i^d \rightarrow \text{PI} \rightarrow T_i^m \rightarrow \text{Motor} \rightarrow \text{Joint} \rightarrow q_i \]

Figure 7.4: Cascaded PI joint controller

For the simulations conducted, the joint controllers operate at a sample period of 500 µs, which is several times faster than the 8 ms sample period used for MPCC. This allows for feedback to be taken into account at a high sample rate. The PI gains used for joint control are provided in Appendix C.

7.4 MPCC implementation with joint feedback

Control Algorithm 6.1 is implemented on the profile cutting machine model as shown in Figure 7.5. Measurements of the joint position vector \( q \) are received by the MPCC controller, which generates the joint acceleration command \( u_k \). Operating with a sample period of 500 µs, the double integrator converts the acceleration command to a position command \( q_i^d \), which is then tracked by the joint controllers.

For the purposes of obtaining state feedback, the model predictive contouring controller uses \( u_k \) to update an internal double integrator model which operates with a sample period of 8 ms, with states \( q_k^d \) and \( \dot{q}_k^d \). It is expected that the actual joint position \( q_k \) will approximately follow \( q_k^d \) with some lag. Therefore, the dynamics governing \( q_k \) are modelled with a simple first order discrete-time system and unit delay with \( q_k^d \) as the input, as illustrated in Figure 7.6. The first-order model of the closed-loop plant was obtained from system-identification experiments with the simulation model, as described in Appendix C.
Figure 7.5: MPCC with joint feedback

Figure 7.6: Prediction model for MPCC with joint feedback

The state vector corresponding to the model shown in Figure 7.6 is

$$\bar{\xi}_k^J = \begin{bmatrix} q^d_k \\ \dot{q}^d_k \\ q^d_{k-1} \\ q_k \end{bmatrix}.$$  \hspace{1cm} (7.6)

In addition, a constant output disturbance model is incorporated into the prediction model, following the same technique as described in Section 5.2. The state vector for
The addition of the disturbance model adds “integral action” to the MPCC controller. The model equations can be found in Appendix C. At each sample, the measured joint position \( q_k \) along with \( q_k^d \) from the double integrator model are used to calculate the current state \( \xi_k^J \). This is then used in conjunction with the MPCC model to form predictions of \( q_{k+i,k} \) across the horizon for use in the cost function.

7.5 Simulations

Control Algorithm 6.1 was simulated with joint feedback for a range of values of the traversal time weighting \( q_\theta \). The simulation and controller parameter values are given in Table 7.2.

The desired path used for the simulations is shown in Figure 7.7. A three-dimensional plot of the desired path of the end effector is shown in Figure 7.7(a), and Figure 7.7(b) shows the resulting flower-shaped path of the intersection between the cutting beam and the workpiece. Note that this path corresponds to the flower-shaped path used in Chapters 3 and 5, and is selected to test the MPCC algorithm due to the varying degrees of curvature along its length. The cutting beam is inclined at an angle of 22.5° perpendicular to the direction of cut, and the standoff distance of the cutting head is 20 mm.

The path distance \( s \) for this application is defined as the distance travelled by the point where the cutting beam intersects the workpiece in the X-Y plane. As discussed in Chapter 6, it is assumed that \( z^d(\theta) \) is parameterised such that \( ds/d\theta = 1 \).
### Simulation parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow sample period $h$</td>
<td>8 ms</td>
</tr>
<tr>
<td>Fast sample period</td>
<td>500 $\mu$s</td>
</tr>
</tbody>
</table>

### Input constraints

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Limits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prismatic joint acceleration limits</td>
<td>-300 to 300 mm/s$^2$</td>
</tr>
<tr>
<td>Revolute joint acceleration limits</td>
<td>-30 to 30 rad/s$^2$</td>
</tr>
<tr>
<td>Prismatic joint jerk limits</td>
<td>-3500 to 3500 mm/s$^3$</td>
</tr>
<tr>
<td>Revolute joint jerk limits</td>
<td>-600 to 600 rad/s$^3$</td>
</tr>
</tbody>
</table>

### MPCC parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horizon length $N$</td>
<td>24</td>
</tr>
<tr>
<td>Contour error weighting $q_c$</td>
<td>1000</td>
</tr>
<tr>
<td>Joint input deviation weighting $r_u$</td>
<td>$10^{-3} \times \mathbf{I}$</td>
</tr>
<tr>
<td>Path speed deviation weighting $r_v$</td>
<td>500</td>
</tr>
<tr>
<td>Traversal time weighting $q_\theta$</td>
<td>0.05 to 2</td>
</tr>
</tbody>
</table>

### Path completion parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contraction parameter $\alpha$</td>
<td>0.032</td>
</tr>
<tr>
<td>LQR weightings $Q^{LQR}, R^{LQR}$</td>
<td>$\mathbf{I}, \mathbf{I}$</td>
</tr>
</tbody>
</table>

Table 7.2: Parameters for multi-axis MPCC simulation
Figure 7.7: Desired path for multi-axis MPCC simulations
Figures 7.8 - 7.11 show plots of the joint command acceleration and jerk, path speed and contouring error respectively for MPCC with joint feedback with $q_\theta = 0.05$ and $q_\theta = 2$. It can be observed that the joint command acceleration and jerk remains within the prescribed limits.

(a) Joint command acceleration  
(b) Joint command jerk

Figure 7.8: Joint command acceleration and jerk for joint feedback MPCC for $q_\theta = 0.05$ and $q_\theta = 2$

The maximum path speed was specified at 45 mm/s for the initial part of the path and then reduced to 30 mm/s. Figure 7.9 shows how the controller anticipates the reduction in the maximum path speed and reduces $v$ accordingly. It can also be seen that the path
Figure 7.9: Path speed for joint feedback MPCC for $q_\theta = 0.05$ and $q_\theta = 2$

Figure 7.10: Contouring error for joint feedback MPCC for $q_\theta = 0.05$ and $q_\theta = 2$
speed is reduced around the tight curves on the path to maintain accuracy.

Figure 7.10(a) shows how the contouring error varies with path parameter for $q_\theta = 0.05$ and $q_\theta = 2$, while the box plots in Figure 7.10(b) illustrate the statistical distribution of contouring error for each simulation. It can be observed that for both $q_\theta = 0.05$ and $q_\theta = 2$, the top 25% of error measurements are spread out over more than half of the range, and the average contouring error lies in the lower half.

From a practical perspective, the error measured on the workpiece to be cut is of more interest than the contouring error. Figure 7.11 shows a plot of the error between the intersection of the cutting beam and the $X$-$Y$ plane and the desired shape shown in Figure 7.7(b) for the same simulations. It can be seen that the plots in Figures 7.10 and 7.11 are quite similar. This indicates that with appropriate choice of $\tilde{Q}_c$, the contouring error definition (6.11) used in the MPCC cost function corresponds well to the error measured at the workpiece.

![Figure 7.10(a): Workpiece error vs $\theta$](image1)

![Figure 7.10(b): Workpiece error box plots](image2)

Figure 7.11: Workpiece error for joint feedback MPCC for $q_\theta = 0.05$ and $q_\theta = 2$

The effect of varying $q_\theta$ is reflected in Figures 7.10 and 7.11. The smaller value of $q_\theta$
leads to the path being traversed more slowly, but with smaller contouring error. The
cost function weights therefore have the desired effect on contouring behaviour. The
difference in traversal time, contouring error and workpiece error between \( q_\theta = 0.05 \)
and \( q_\theta = 2 \) is illustrated in Table 7.3.

<table>
<thead>
<tr>
<th>Travel time weighting ( q_\theta )</th>
<th>0.05</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traversal time (s)</td>
<td>20.945</td>
<td>14.793</td>
</tr>
<tr>
<td>RMS contouring error</td>
<td>0.00314</td>
<td>0.0106</td>
</tr>
<tr>
<td>RMS workpiece error (mm)</td>
<td>0.00325</td>
<td>0.0108</td>
</tr>
</tbody>
</table>

Table 7.3: Contouring error and traversal time for \( q_\theta = 0.05 \) and \( q_\theta = 2 \) for joint feedback MPCC

The plot of traversal time and RMS contouring error versus \( q_\theta \) shown in Figure 7.12
demonstrates that this trend continues over the range of values of \( q_\theta \) between 0.05 and 2.
7.6 Conclusion

The multi-axis model predictive contouring control scheme was applied to an industrial profile cutting machine model. The joint position $q$ is provided as feedback to the MPCC controller. Simulations conducted over a range of cost function weights show that, as expected, using a low traversal time weighting results in the system taking a longer time to traverse the path, but with a smaller contouring error.

The implementation of MPCC presented in this Chapter assumes that measurements of the joint position $q$ are available as feedback. As joint position is not directly measured with the existing hardware present in the profile cutting machine, such an implementation would require additional sensors to measure the joint position, which may be costly to install. In Chapter 8, alternative implementations of model predictive contouring control are proposed which require less modification of the existing hardware. These alternative approaches can therefore more readily be applied to industrial machines.
Chapter 8

Alternative MPCC Implementations

In Chapter 7, model predictive contouring control was implemented on an industrial machine model under the assumption that joint feedback is available to the contouring controller. While such an implementation is expected to achieve a high level of contouring performance, appropriate sensors and communication capabilities are required, which may involve a significant change to existing industrial hardware. Therefore, from an industrial implementation perspective, there is motivation to consider alternative implementation approaches which require little or no modification of the current hardware.

In this Chapter, two alternative MPCC implementations are presented where direct joint feedback is not required. First, MPCC is implemented under the assumption that only motor feedback is available, and subsequently an open loop implementation is presented, which does not require any communication of feedback to the MPCC controller. Simulation results demonstrate that while MPCC may be adapted to operate with only motor feedback or in an open loop fashion, these simpler implementation approaches come at the cost of reduced contouring performance.

8.1 MPCC implementation with motor feedback

Model predictive contouring control may be implemented using measurements of motor feedback, where the measured motor position $q^{m}$ is used in place of $q$. With this
approach, direct measurements of joint position are not required. The control architecture using motor feedback is shown in Figure 8.1.

![Figure 8.1: MPCC with motor feedback](image)

The measured joint position is also replaced by measured motor position \( q^m \) in the MPCC internal model, so that the state vector is

\[
\xi_k^M = \begin{bmatrix}
\Delta q_k^d \\
\Delta i_k^d \\
\Delta q_{k-1}^d \\
\Delta q_k \\
q_k^m
\end{bmatrix}.
\]  

(8.1)

The prediction model for MPCC with motor feedback is identified using the same conventional techniques as was used for obtaining the prediction model with joint feedback, shown in Figure 7.6.

Simulation results from multi-axis MPCC implementation with motor feedback are shown in Figures 8.2-8.6 and Table 8.1. The joint command acceleration and jerk shown in Figure 8.2 remain within the prescribed limits, with the control inputs for \( q_\theta = 2 \) more aggressive compared to \( q_\theta = 0.05 \).
Figure 8.2: Joint command acceleration and jerk for motor feedback MPCC for $q_0 = 0.05$ and $q_0 = 2$
Figure 8.3: Path speed for motor feedback MPCC for $q_\theta = 0.05$ and $q_\theta = 2$

Figure 8.4: Contouring error for motor feedback MPCC for $q_\theta = 0.05$ and $q_\theta = 2$
Figure 8.3 shows how the path speed is adjusted in order to satisfy path speed constraints as well as maintain accuracy along the path. Plots of contouring and workpiece error for $q_\theta = 0.05$ and 2 are shown in Figures 8.4 and 8.5.

It can be observed that for $q_\theta = 0.05$, the path is traversed more slowly than for $q_\theta = 2$, but with greater contouring accuracy. A comparison of traversal time, contouring error and workpiece error is shown in Table 8.1. Figure 8.6 demonstrates the effect of the cost function weights on contouring error and traversal time, which is similar to that achieved for joint feedback MPCC. A comparison of the contouring performance achieved by motor feedback MPCC and joint feedback MPCC is presented in Section 8.3.

![Workpiece error vs $\theta$](image1.png)

![Workpiece error box plots](image2.png)

Figure 8.5: Workpiece error for motor feedback MPCC for $q_\theta = 0.05$ and $q_\theta = 2$

<table>
<thead>
<tr>
<th>Travel time weighting $q_\theta$</th>
<th>0.05</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traversal time (s)</td>
<td>21.038</td>
<td>14.801</td>
</tr>
<tr>
<td>RMS contouring error</td>
<td>0.00721</td>
<td>0.0191</td>
</tr>
<tr>
<td>RMS workpiece error (mm)</td>
<td>0.00734</td>
<td>0.0195</td>
</tr>
</tbody>
</table>

Table 8.1: Contouring error and traversal time for $q_\theta = 0.05$ and $q_\theta = 2$ for motor feedback MPCC
Figure 8.6: Traversal time and RMS contouring error vs. traversal time weighting for MPCC with motor feedback
The current architecture of the profile cutting machine does not include the communication of up-to-date motor position measurements from the servo drives to the motion co-ordinator, which is required for the implementation of MPCC with motor feedback. In the following section, MPCC is implemented in an open loop fashion. This approach requires the least modification to the existing industrial architecture.

### 8.2 Open loop MPCC implementation

The architecture for open loop MPCC is shown in Figure 8.7. Multi-axis model predictive contouring control is used to generate a joint reference trajectory in an open loop fashion. As the model predictive contouring controller does not receive feedback from the plant, there is no need for a closed-loop system model, and the double integrator is used as the prediction model, with state vector

\[ \xi_{OL}^k = \begin{bmatrix} q^d_k \\ \dot{q}^d_k \end{bmatrix}. \]  

\[ (8.2) \]
In contrast to existing trajectory planning schemes such as those discussed in Chapter 2, the MPCC trajectory planner explicitly addresses the trade-off between accuracy and productivity, but neglects any contouring error introduced by the joint controllers. The receding horizon formulation also allows for trajectory planning to occur without complete knowledge of the path: only a look-ahead horizon of the path function is required.

An advantage of the open loop approach is that the MPCC algorithm can operate in a quasi-real-time fashion, reducing the requirement for fast computation. In addition, real-time communication of joint feedback to the motion co-ordinator is not required, and there is no need to model the closed-loop dynamics. However, without consideration of feedback, the full advantages of MPCC are not realised.

Simulation results for open loop MPCC are given in Figures 8.8-8.12 and Table 8.2. The command acceleration and jerk shown in Figure 8.8 is similar to that obtained for the other two implementation approaches. The path speed and contouring error plots shown in Figures 8.9 and 8.10 as well as the traversal time and RMS errors shown in Table 8.2 demonstrate how a small traversal time weighting corresponds to slower, more accurate contouring. Figure 8.12 shows that the cost function weights have the intended effect on contouring accuracy and traversal time over the range of values of \( q_\theta \).

Open loop MPCC behaves in much the same way as the other implementation approaches. However, different levels of contouring performance are achieved by implementing MPCC with motor or joint feedback. A comparison of the three implementation approaches is provided in the following Section, which highlights the trade-off between contouring performance and modification of existing hardware.

<table>
<thead>
<tr>
<th>Travel time weighting ( q_\theta )</th>
<th>0.05</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traversal time (s)</td>
<td>21.329</td>
<td>14.793</td>
</tr>
<tr>
<td>RMS contouring error</td>
<td>0.0297</td>
<td>0.0469</td>
</tr>
<tr>
<td>RMS workpiece error (mm)</td>
<td>0.0313</td>
<td>0.0497</td>
</tr>
</tbody>
</table>

Table 8.2: Contouring error and traversal time for \( q_\theta = 0.05 \) and \( q_\theta = 2 \) for open loop MPCC
8.2 Open loop MPCC implementation

Figure 8.8: Joint command acceleration and jerk for open loop MPCC for \( q_\theta = 0.05 \) and \( q_\theta = 2 \)
Figure 8.9: Path speed for open loop MPCC for $q_\theta = 0.05$ and $q_\theta = 2$

Figure 8.10: Contouring error for open loop MPCC for $q_\theta = 0.05$ and $q_\theta = 2$
8.2 Open loop MPCC implementation

(a) Workpiece error vs θ

(b) Workpiece error box plots

Figure 8.11: Workpiece error for open loop MPCC for \( q_θ = 0.05 \) and \( q_θ = 2 \)

Figure 8.12: Traversal time and RMS contouring error vs. traversal time weighting for open loop MPCC
8.3 Comparison of MPCC implementations

In order to compare the performance of the alternative implementations from Sections 8.1 and 8.2 with the implementation presented in Section 7.4, the RMS contouring error versus traversal time is plotted for MPCC with joint feedback, motor feedback, and open loop MPCC over a range of penalty weightings in Figure 8.13. Figure 8.14 shows a similar plot of RMS workpiece error versus traversal time.

As expected, for all three control implementations the contouring and workpiece errors decrease as the traversal time increases. It can be observed that MPCC with joint feedback exhibits the best performance of the three approaches. Implementing MPCC with motor feedback reduces contouring accuracy slightly, as without joint measurements, the controller cannot compensate for the effects of the flexible coupling. It can be observed that the improvement in average contouring accuracy achieved by using joint feedback increases with shorter traversal times. Open loop MPCC, where the controller does not have access to any feedback from the plant, achieves the worst contouring per-
The results indicate that a significant improvement in contouring accuracy is achieved by using closed-loop MPCC, while a smaller improvement in accuracy results from using direct joint feedback instead of motor feedback. It should be noted that the difference in performance between the implementation approaches depends on the amount of compliance present in the joints. If the stiffness and damping constants of the flexible couplings in the simulation model were decreased, the performance improvement achieved by implementing MPCC with joint feedback would increase. In addition, the performance of open loop MPCC depends on the tuning of the joint controllers, as illustrated in the following Section.
8.3.1 Axis mismatch

In contouring control architectures involving open loop trajectory planning, it is well-known that if the joint controllers are well-matched [112], good contouring performance is achieved. However, if the closed-loop dynamics of one joint differs significantly from the others, the axis mismatch leads to poor contouring. Therefore, it is expected that the performance of open loop MPCC is highly dependent on the tuning of the joint controllers.

Figures 8.15 and 8.16 show the RMS contouring and workpiece error plots where axis mismatch was introduced by reducing the proportional gain in the position loop for joint 1 by 50%. It can be observed that the performance of the open loop MPCC becomes significantly worse when the proportional gain is reduced. In contrast, the performance of the two closed-loop MPCC approaches remains approximately the same. Note that the MPCC internal models used for the closed-loop approaches were re-identified for the modified gain. Figures 8.13-8.16 illustrate how the performance of open loop MPCC deteriorates if axis mismatch is present. In contrast, MPCC implemented with joint or motor feedback is able to achieve the same performance with some axis mismatch.

8.4 Conclusion

Alternative implementations of multi-axis model predictive contouring control have been presented where the controller operates either with motor position feedback or in an open loop fashion. These alternative implementations require less modification to the existing machine hardware than the implementation with joint feedback proposed in Chapter 6. However, this comes at the cost of reduced contouring performance.

Simulation results show that for the alternative implementations, the cost function weights have a similar effect on contouring behaviour. Implementing MPCC with motor feedback achieves a significant improvement in contouring performance over open loop MPCC. A smaller performance improvement is achieved when direct joint feed-
Figure 8.15: RMS contouring error versus traversal time with reduced PI gain

Figure 8.16: RMS workpiece error versus traversal time with reduced PI gain
back is used instead of motor feedback. In addition, if MPCC is implemented without feedback, controller performance is highly dependent on appropriate tuning of the joint controllers. In contrast, the performance of MPCC implemented with joint or motor feedback is largely unaffected by axis mismatch. These results demonstrate the trade-off between minimising modification of existing hardware and maximising contouring performance, and provide some insight into the performance improvement to be gained with modification of the existing industrial hardware and architecture.

Current state-of-the-art trajectory planners (see Section 2.1) select path speed by minimising an objective function aimed at minimising traversal time, in some cases with additional terms penalising control effort. However, as these trajectory planners typically assume that the reference trajectory remains exactly on the desired path, contouring accuracy is not explicitly included in the objective function. In contrast, in the proposed MPCC approach, the cost function to be minimised reflects the relevant control objectives, including accuracy. In addition, the receding horizon structure allows the MPCC scheme to respond to available feedback.
Chapter 9
Contributions and Further Work

THE contributions of this thesis include the development of closed-loop control schemes addressing the minimum-time path-following and contouring problems. In addition, several opportunities for further research have been revealed, including extending the theoretical results relating to time-optimal receding horizon control and implementation of MPCC on industrial hardware.

9.1 Summary of contributions

This thesis has investigated the application of a receding horizon framework to path-following control, with the intention of reflecting specific control objectives in the finite horizon cost function. The aims of this research, established in Chapter 2, have been addressed as follows.

9.1.1 Receding horizon formulation for time-optimal control

A closed-loop, time-optimal controller based on a receding horizon framework was developed in Chapter 3. In contrast to [45], the cost function and constraint are selected with the intention of achieving minimum-time path-following. At each sample, a finite horizon optimisation was posed where the objective is to advance along the path as far as possible while honouring the system constraints. A terminal constraint was used to guarantee recursive feasibility of the controller. The receding horizon formulation
allows for the consideration of feedback, unlike offline time-optimal trajectory planning approaches.

9.1.2 Conditions for minimum-time path-following under receding horizon control

In Chapter \[3\], it was rigorously shown that for sufficiently long horizons, minimum-time path-following is achieved when the receding horizon scheme is applied to second order differentially flat systems. In addition, a condition on the minimum required horizon length was derived. Simulations verified the theoretical results and demonstrated that selecting shorter horizons leads to longer traversal times, revealing a trade-off between traversal time and computational burden.

9.1.3 Receding horizon formulation for biaxial contouring control

In Chapter \[4\], the constraint on path-following accuracy was replaced with a contouring error penalty in the cost function, so that the trade-off between productivity and accuracy was explicitly addressed. The cost function also incorporated a penalty on control deviations to encourage smooth control inputs. By representing competing control objectives of minimising traversal time and contouring error in the cost function, the behaviour of the controller could be modified in an intuitive fashion by adjusting the penalty weights.

A linear time-varying approximation was proposed to reduce the computational burden and facilitate real-time implementation. In addition, a contraction constraint was introduced to ensure path completion in finite time with minimal effect on the behaviour of the controller and computation time.
9.1.4 Real-time implementation of model predictive contouring control

In the first real-time implementation of a receding horizon path-following control scheme, the model predictive contouring controller was implemented on an X-Y table test rig using MATLAB/Simulink and xPC Target. The linear time-varying approximation was used to reduce the computational burden to a level acceptable for practical implementation. A first principles model of the X-Y table was used as the controller internal model, incorporating a constant output disturbance model and unit delay. Experimental results demonstrated the effect of the penalty weights on contouring accuracy and traversal time. The proposed control scheme was shown to outperform cascaded PI and model predictive tracking control schemes with constant velocity reference trajectories.

9.1.5 Multi-axis model predictive contouring control

In Chapter 6, the model predictive contouring control approach was generalised to the more complicated problem of multi-axis contouring. This required the consideration of both the position and orientation of the end effector, along with the machine forward kinematics. The definition of contouring error was generalised to include position and orientation error and penalised in the finite horizon cost function. The linear time-varying approximation was extended to accommodate nonlinear forward kinematics as well as state-dependent path speed constraints.

With a view towards industrial implementation, a multi-rate architecture was proposed where the model predictive contouring controller generates joint position commands to be tracked by low level position controllers. This allows for the servo controllers to operate at sample rates comparable to those used in industry without increasing the computational burden of MPCC, thereby increasing the viability of multi-axis MPCC for industrial machines.
9.1.6 Multi-axis model predictive contouring control simulations

The multi-axis model predictive contouring control scheme with industrial architecture was implemented in simulation on an industrial profile cutting machine model in Chapter 7. Measurements of joint position are provided as feedback to the MPCC controller. Results demonstrate that the cost function weights have the desired effect on contouring behaviour.

In Chapter 8, alternative implementations of MPCC were proposed requiring less modification to the existing machine hardware. These alternative implementations may then be more readily applied to industrial machines. Model predictive contouring control may be implemented with measurements of motor position, eliminating the need to install additional joint position sensors. Alternatively, MPCC may be used as an open loop trajectory planner, so that real-time feedback to the MPCC controller is not required. Results show that while these alternative implementations behave similarly to MPCC implemented with joint feedback, a lower level of contouring accuracy is achieved. The trade-off between achieving the best contouring performance and minimising system modification was highlighted in a comparison of the simulation results obtained from the three approaches.

9.2 Publications

The research carried out for this thesis has been, or is intended to be, published in the following.

2. Lam, D., Manzie, C., Good, M. C. and Bitmead, R. R. “Receding horizon time-optimal control for a class of differentially flat systems.” Under review.
3. Lam, D., Manzie, C. and Good, M. C. “Model predictive contouring control for biaxial systems.” IEEE Transactions on Control Systems Technology. To be pub-
9.3 Further work

Opportunities for further research stemming from the contributions of this thesis have been identified, and are discussed in the following.

9.3.1 Receding horizon time-optimal control extensions

The receding horizon time-optimal control scheme developed in Chapter 3 may be extended in a number of ways. It was mentioned in Section 3.3.2 that constraints on the rate of change of the control input may be incorporated into the control scheme. An investigation if the time optimality properties of the controller are preserved under these new constraints would be beneficial, in order to determine if the receding horizon scheme may be used to produce smooth time-optimal solutions.

In addition, Theorem 3.1 is only applicable to second order differentially flat systems, as stated in Assumption 3.3. Perhaps the class of systems to which Theorem 3.1 may be applied can be enlarged. The two-dimensional phase plane approach used for the proofs in Chapter 3 may only be used for second order differentially flat systems. Therefore, in order to extend the result to other systems, an alternative proof methodology would need to be adopted.

Receding horizon time-optimal control requires the solution of a nonlinear optimisation problem with equality constraints. Unfortunately, finding such a solution in real-time


is computationally challenging. The application of fast optimisation approaches for the implementation of receding horizon time-optimal control is another potential area of investigation.

Moreover, while the receding horizon control scheme is able to take advantage of feedback to reject disturbances, the formulation presented in Chapter 3 does not explicitly address the issue of robustness. A robust MPC technique (see [8]) could be applied in order to provide a guarantee of exact path-following in the presence of bounded disturbances. However, incorporating robust control would come at the cost of increased traversal times.

9.3.2 Further reduction in computational burden of MPCC

The linear time-varying approximation proposed in Chapter 4 allowed for biaxial MPCC to be implemented in real-time with a sample period of 4 ms using a basic QP algorithm. However, as multi-axis MPCC typically involves a larger number of control inputs to be determined, the multi-axis LTV MPCC algorithm takes significantly longer to execute. Therefore, to facilitate real-time implementation of multi-axis MPCC, further reduction in computation time is necessary.

The linearisation of the forward kinematics required for the linear time-varying approximation uses the Jacobian of the serial manipulator, which itself can be expensive to compute. Therefore, alternative, derivative-free linearisation techniques could be investigated to reduce the computational burden. Applying more advanced QP solvers may also assist in facilitating real-time implementation.

9.3.3 Implementation of multi-axis MPCC on industrial hardware

Multi-axis model predictive control was implemented in simulation only. Implementation of multi-axis MPCC on a real industrial machine would demonstrate the benefits of the proposed approach and allow for its adoption in a commercial setting. It is likely
that the industrial implementation of MPCC will rely on further reduction in its computational burden as discussed earlier.

If joint position sensors are installed as a trial on a particular machine, the performance of model predictive contouring control with joint feedback, motor feedback, or as an open loop trajectory planner could be assessed experimentally. This would allow for a more detailed analysis of the benefits and costs of each approach to be conducted, which could then guide the decision making process for the machine manufacturer as to which implementation approach is most suitable.
References


REFERENCES


Appendix A

X-Y Table System Identification

System identification and model validation were carried out for the X-Y table test rig in order to facilitate implementation of MPCC as described in Chapter 5. A simulation model including nonlinear friction was developed and the model parameters identified from experimental data. Further experiments were then conducted to validate the model against the X-Y table test rig. Acknowledgement is due to Maria Weiß for her work in conducting the system-identification and validation experiments.

A.1 Friction model

The friction model used for the nonlinear simulation model includes Coulomb and Stribeck friction, in contrast to the linear model used to implement the controller (5.5). The motor and load friction torques for one axis may be expressed as

\[
\begin{align*}
\bar{F}_m(\dot{\phi}) &= b_m \dot{\phi} + F_{m}^{NL}(\dot{\phi}), \\
\bar{F}_l(\dot{\psi}) &= b_l \dot{\psi} + F_{l}^{NL}(\dot{\psi})
\end{align*}
\]  

(A.1)

where \( \varphi, \psi, b_m, \) and \( b_l \) are defined as per Chapter 5 and \( F_{m}^{NL}(\dot{\phi}), F_{l}^{NL}(\dot{\psi}) \), are nonlinear functions representing the combined Coulomb and Stribeck friction torques. The functions
$F_{NL}^m$ and $F_{NL}^l$ are based on the Gaussian and Karnopp friction models [52, 88],

$$
F_{NL}^m = \begin{cases} 
(F_m^C + (F_m^S - F_m^C)e^{(-\frac{\dot{\phi}}{v_m})^2}) \text{sgn}(\dot{\phi}), & \text{if } |\dot{\phi}| > \epsilon_f, \\
F_m^E, & \text{if } |\dot{\phi}| \leq \epsilon_f \text{ and } |F_m^E| \geq F_m^s, \\
F_m^S \text{sgn}(F_m^E), & \text{otherwise},
\end{cases}
$$

$$
F_{NL}^l = \begin{cases} 
(F_l^C + (F_l^S - F_l^C)e^{(-\frac{\dot{\psi}}{v_l})^2}) \text{sgn}(\dot{\psi}), & \text{if } |\dot{\psi}| > \epsilon_f, \\
F_l^E, & \text{if } |\dot{\psi}| \leq \epsilon_f \text{ and } |F_l^E| \geq F_l^s, \\
F_l^S \text{sgn}(F_l^E), & \text{otherwise},
\end{cases}
$$

(A.2)

where $F_m^E, F_l^E$ are the combined torques acting on the motor and load respectively, $F_m^C$, $F_m^S, v_m, F_l^C, F_l^S$ and $v_l^S$ are parameters to be identified and $\epsilon_f$ is a small positive scalar used to avoid problems arising in simulation around zero velocity.

The motor friction model was identified first by disconnecting the motor from the load and running the motors through a variety of constant velocity sweeps. The average of the measured $q$-axis current for each sweep was then used to form an estimate of the motor current required to compensate friction. The model parameters were calculated to fit the experimental data using nonlinear least squares.

Once the motor friction model was identified, the motor was reconnected to the load and the experiment repeated in order to identify the load friction parameters. Figure A.1 shows the experimental data and identified friction models for the $X$-axis. It can be observed that the model fits the data quite well.

It was observed that the measured friction compensating current shown in Figure A.1 is not perfectly symmetrical about the zero velocity axis. Therefore, separate friction model parameters were identified for positive and negative motor and load velocities. The viscous friction parameters identified for negative velocity were used in the controller model (5.3). Tables A.1 and A.2 show the identified friction parameter values for the $X$ and $Y$ axes respectively. Note that $K_t$ is the motor torque constant, used to related the current required to compensate for friction to the actual friction torque.
Figure A.1: Friction models and measured data for the X-axis

<table>
<thead>
<tr>
<th>Velocity sign</th>
<th>$F_{mc} / K_t$</th>
<th>$F_{ms} / K_t$</th>
<th>$b_m / K_t$</th>
<th>$v^v_m / K_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>positive</td>
<td>0.02451</td>
<td>0.01928</td>
<td>0.0003170</td>
<td>0.9306</td>
</tr>
<tr>
<td>negative</td>
<td>0.03089</td>
<td>0.02390</td>
<td>0.0001773</td>
<td>1.198</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Velocity sign</th>
<th>$F_{cl} / K_t$</th>
<th>$F_{cs} / K_t$</th>
<th>$b_l / K_t$</th>
<th>$v^v_l / K_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>positive</td>
<td>0.05833</td>
<td>0.07942</td>
<td>0.002289</td>
<td>0.6739</td>
</tr>
<tr>
<td>negative</td>
<td>0.06415</td>
<td>0.06415</td>
<td>0.002160</td>
<td>-1.631</td>
</tr>
</tbody>
</table>

Table A.1: X-axis friction parameters
Motor friction

<table>
<thead>
<tr>
<th>Velocity sign</th>
<th>( F_{m,x} / K_i )</th>
<th>( F_{m,y} / K_i )</th>
<th>( b_{m} / K_i )</th>
<th>( v_{m} / K_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>positive</td>
<td>0.02811</td>
<td>0.02637</td>
<td>0.0002682</td>
<td>0.1459</td>
</tr>
<tr>
<td>negative</td>
<td>0.02826</td>
<td>0.02414</td>
<td>0.0002958</td>
<td>0.3426</td>
</tr>
</tbody>
</table>

Load friction

<table>
<thead>
<tr>
<th>Velocity sign</th>
<th>( F_{l,x} / K_i )</th>
<th>( F_{l,y} / K_i )</th>
<th>( b_{l} / K_i )</th>
<th>( v_{l} / K_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>positive</td>
<td>0.07788</td>
<td>0.1027</td>
<td>0.001663</td>
<td>-0.4243</td>
</tr>
<tr>
<td>negative</td>
<td>0.09023</td>
<td>0.06630</td>
<td>0.001451</td>
<td>0.5265</td>
</tr>
</tbody>
</table>

Table A.2: Y-axis friction parameters

A.2 Two-inertia model

The continuous-time equations of motion for the nonlinear X-Y table simulation model are identical to (5.1), except that the friction torque includes the nonlinear friction model identified in the previous section:

\[
\begin{align*}
\ddot{\psi}_x &= \frac{1}{J_m} \left( T_{m,x} + k_{c,x}(\varphi_x - \psi_x) + c_x(\dot{\varphi}_x - \dot{\psi}_x) - \bar{F}_{m,x} \right), \\
\ddot{\varphi}_x &= \frac{1}{J_{l,x}} \left( k_{c,x}(\psi_x - \varphi_x) + c_x(\dot{\psi}_x - \dot{\varphi}_x) - \bar{F}_{l,x} \right), \\
\ddot{\psi}_y &= \frac{1}{J_m} \left( T_{m,y} + k_{c,y}(\varphi_y - \psi_y) + c_y(\dot{\varphi}_y - \dot{\psi}_y) - \bar{F}_{m,y} \right), \\
\ddot{\varphi}_y &= \frac{1}{J_{l,y}} \left( k_{c,y}(\psi_y - \varphi_y) + c_y(\dot{\psi}_y - \dot{\varphi}_y) - \bar{F}_{l,y} \right). 
\end{align*}
\]

(A.3)

The system-identification experiment used to identify the remaining model parameters is shown in Figure A.2. Each axis is commanded to run at a constant average velocity with a chirp signal injected into the velocity command. The chirp signal excites the necessary range of frequencies for system-identification purposes.

For the identification of the linear model parameters, the model (A.3) was linearised around the operating point neglecting nonlinear friction. The linear state space model...
A.2 Two-inertia model

Figure A.2: X-Y table system-identification experiment

for one axis is given as

\[
\begin{bmatrix}
\dot{\phi} \\
\dot{\psi} \\
\ddot{\phi} \\
\ddot{\psi}
\end{bmatrix} =
\begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
-k_c J_m & k_c J_m & -\frac{c}{J_l} & \frac{b}{J_l} \\
\frac{k_c}{J_l} & -\frac{k_c}{J_l} & -\frac{c}{J_l} & -\frac{b}{J_l}
\end{bmatrix}
\begin{bmatrix}
\phi \\
\psi \\
\dot{\phi} \\
\dot{\psi}
\end{bmatrix} +
\begin{bmatrix}
0 \\
0 \\
K_t \\
0
\end{bmatrix} i,
\]

where the \( q \)-axis current \( i \) is the system input and the motor velocity \( \dot{\phi} \) is the output. The prediction error method was used to fit the linearised model to the experimental data. Table A.3 shows the identified parameter values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>X-axis value</th>
<th>Y-axis value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( J_m ) (kg m(^2))</td>
<td>2.300 \times 10^{-5}</td>
<td>2.000 \times 10^{-5}</td>
</tr>
<tr>
<td>( J_l ) (kg m(^2))</td>
<td>2.1041 \times 10^{-5}</td>
<td>2.3269 \times 10^{-5}</td>
</tr>
<tr>
<td>( k_c ) (Nm/rad)</td>
<td>2.6682</td>
<td>2.3017</td>
</tr>
<tr>
<td>( c ) (N ms/rad)</td>
<td>5.000 \times 10^{-4}</td>
<td>1.0894 \times 10^{-3}</td>
</tr>
<tr>
<td>( K_t ) (Nm/A)</td>
<td>0.4364</td>
<td></td>
</tr>
</tbody>
</table>

Table A.3: Identified X-Y table model parameters
A.3 Model validation

A simulation model based on the nonlinear continuous time equations was developed and validated against data from the X-Y table test rig. For the first validation experiment, the simulation model and test rig were subjected to a sinusoidal position command on each axis. Plots of the command current, motor position and velocity, and load position and velocity for the X-axis are shown in Figure A.3. It can be seen that the signals from the simulation model and test rig match quite well.

Figure A.3: Closed-loop X-axis model validation
A.3 Model validation

A more challenging validation experiment was conducted where the measured current command signal from Figure A.3 was applied to the simulation model. Plots comparing the simulation model and test rig results from this open loop validation test for the X-axis are shown in Figure A.4. It can be observed that the model agrees well with the measured data.

![Figure A.4: Open loop X-axis model validation](image)

The simulation model used for validation includes nonlinear friction, which is not present in the model used for the controller. Therefore, it is recognised that there is significant modelling error, which is compensated in the controller with a disturbance
model, as discussed in Section 5.2.

A.4 MPCC prediction model

The X-Y table parameters specified in Table 5.1 for the prediction model used by the controller are taken from Tables A.1–A.3. It should be noted that the nonlinear friction parameters $F_{c m}, F_{s m}, v_{s m}, F_{c l}, F_{s l}$ and $v_{s l}$ are not used in the controller prediction model (5.5). The discrete-time system matrices $A_m, B_m$ and $C_m$ in (5.5) are given below:

$$A_m = \begin{bmatrix} A_{m,x} & 0 \\ 0 & A_{m,y} \end{bmatrix}, \quad B_m = \begin{bmatrix} B_{m,x} & 0 \\ 0 & B_{m,y} \end{bmatrix}, \quad C_m = \begin{bmatrix} C_{m,x} & 0 \\ 0 & C_{m,y} \end{bmatrix},$$

$$A_{m,x} = \begin{bmatrix} 0.3694 & 0.0029 & 0.6306 & 0.0010 \\ -202.2258 & 0.3218 & 202.2258 & 0.6217 \\ 0.6470 & 0.0011 & 0.3530 & 0.0026 \\ 194.3546 & 0.6796 & -194.3546 & 0.2011 \end{bmatrix},$$

$$A_{m,y} = \begin{bmatrix} 0.4577 & 0.0029 & 0.5423 & 0.0011 \\ -183.2905 & 0.3548 & 183.2905 & 0.6004 \\ 0.5188 & 0.0010 & 0.4812 & 0.0028 \\ 170.0795 & 0.5935 & -170.0795 & 0.3253 \end{bmatrix},$$

$$B_{m,x} = \begin{bmatrix} 0.1278 & 54.6872 & 0.0246 & 21.6112 \end{bmatrix}^T,$$

$$B_{m,y} = \begin{bmatrix} 0.1264 & 54.5327 & 0.0235 & 19.7790 \end{bmatrix}^T,$$

$$C_{m,x} = C_{m,y} = \begin{bmatrix} 0 & 0 & 0.7958 & 0 \end{bmatrix}. \quad (A.5)$$
Appendix B

Profile Cutting Machine Simulation Model

A simplified simulation model was developed for an industrial profile cutting machine. Corke’s Robotics Toolbox [25] was used to calculate the forward dynamics from the Denavit-Hartenberg parameters in Table 7.2 and the dynamic parameters of each link. The geometry of the links corresponding to the revolute joints was approximated with an assembly consisting of cylinders and rectangular prisms. The simplified geometry was then used to calculate the dynamic parameters of each link.

B.1 Simplified geometry

The simplified geometry for the cutting head is shown in Figure B.1. The geometry was assumed to be made up of solid cylinders and rectangular prisms, and it was assumed that the distribution of mass throughout the volume of Figure B.1 was constant. The dimensions used for the cutting head geometry are given in Table B.1.
Figure B.1: Simplified geometry used to calculate rotational inertia matrices
<table>
<thead>
<tr>
<th>Dimension name</th>
<th>Value (mm)</th>
<th>Remark</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>$d_2$</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>$d_3$</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>$d_4$</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$c_1$</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>$l_1$</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>$l_2$</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>$l_3$</td>
<td>41.8259</td>
<td>Constrained by geometry</td>
</tr>
<tr>
<td>$l_4$</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>$l_5$</td>
<td>21.8701</td>
<td>Constrained by geometry</td>
</tr>
<tr>
<td>$l_6$</td>
<td>32.2721</td>
<td>Constrained by geometry</td>
</tr>
<tr>
<td>$l_7$</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>$t_1$</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>$t_2$</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>$w_1$</td>
<td>16</td>
<td>Same as $d_3$</td>
</tr>
<tr>
<td>$w_2$</td>
<td>16</td>
<td>Same as $d_3$</td>
</tr>
</tbody>
</table>

Table B.1: Dimension values for simplified geometry

### B.2 Dynamic parameters

The centre of gravity vectors and rotational inertia matrices for links 4 and 5 were calculated from first principles based on the simplified geometry from Figure B.1 and are given below:

\[
\begin{align*}
    \mathbf{r}_4^G &= \begin{bmatrix} 0 & -0.0223 & 0.0600 \end{bmatrix}^T, \\
    \mathbf{I}_4 &= \begin{bmatrix} 0.0052 & 0 & 0 \\ 0 & 0.002408 & -0.0008 \\ 0 & -0.0008 & 0.0036 \end{bmatrix}, \\
    \mathbf{r}_5^G &= \begin{bmatrix} 0 & -0.0128 & 0.0354 \end{bmatrix}^T, \\
    \mathbf{I}_5 &= \begin{bmatrix} 0.0038 & 0 & 0 \\ 0 & 0.0019 & -0.0003 \\ 0 & -0.0003 & 0.0022 \end{bmatrix}.
\end{align*}
\]

The parameters $\mathbf{r}_4^G$, $\mathbf{I}_4$, $\mathbf{r}_5^G$ and $\mathbf{I}_5$ are expressed in metres and are defined relative to reference frames 4 and 5 respectively as shown in Figure B.2. A scale diagram of links 4 and 5 showing the location of the centre of gravity vectors is shown in Figure B.2.
The motor, coupling and gear parameters used in (7.2) and (7.4) are given in Table B.2, while the serial link parameters used with the Robotics Toolbox [25] in the profile cutting machine model are given in Table B.3.

<table>
<thead>
<tr>
<th>Joint #</th>
<th>Motor inertia $J^M$ (kg m$^2$)</th>
<th>Motor viscous friction co-eff $b^M$ (Nm s/rad)</th>
<th>Eq. gear ratio $N^G$ (m/rad) or (rad/rad)</th>
<th>Coupling stiffness $k^c$ (N/m) or (N/rad)</th>
<th>Coupling damping co-eff $c$ (Ns/m) or (N s/rad)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$2.2608 \times 10^{-4}$</td>
<td>0.2</td>
<td>0.0012</td>
<td>$8 \times 10^9$</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>$1.0222 \times 10^{-5}$</td>
<td>0.2</td>
<td>$6.6667 \times 10^{-4}$</td>
<td>$8 \times 10^5$</td>
<td>500</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>$3.1831 \times 10^{-4}$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table B.2: Motor, coupling and gear parameters for profile cutting machine model

The parameters in Tables B.2 and B.3 were specified using empirically realistic values. The stiffness and damping constants $k^c$ and $c$ for joints 1 and 2 were selected so that
B.2 Dynamic parameters

<table>
<thead>
<tr>
<th>Link #</th>
<th>Mass (kg)</th>
<th>Centre of gravity (m)</th>
<th>Rotational inertia (kg m$^2$)</th>
<th>Viscous friction co-eff.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>157</td>
<td>-</td>
<td>-</td>
<td>1 (N s/m)</td>
</tr>
<tr>
<td>2</td>
<td>23</td>
<td>-</td>
<td>-</td>
<td>1 (N s/m)</td>
</tr>
<tr>
<td>3</td>
<td>20.97</td>
<td>-</td>
<td>-</td>
<td>1 (N s/m)</td>
</tr>
<tr>
<td>4</td>
<td>12.79</td>
<td>$r^G_4$</td>
<td>$I_4$</td>
<td>0.05 (Nm s/rad)</td>
</tr>
<tr>
<td>5</td>
<td>11.24</td>
<td>$r^G_5$</td>
<td>$I_5$</td>
<td>0.05 (Nm s/rad)</td>
</tr>
</tbody>
</table>

Table B.3: Serial link parameters for profile cutting machine model

the frequency response from motor torque $T^m$ to joint velocity $\dot{q}$ showed a resonance at around 20-30 Hz, as shown in Figures B.3 and B.4.

Figure B.3: Frequency response from $T^m_1$ to $\dot{q}_1$
Figure B.4: Frequency response from $T_2^{in}$ to $q_2$

Note that the purpose of the simulation model is not to represent the dynamics of the real machine, but rather to provide a reasonable example which can be used to demonstrate the behaviour of multi-axis MPCC. The development of a high-fidelity model of the profile cutting machine is beyond the scope of this thesis.
Appendix C

Multi-axis Model Predictive Contouring Control Implementation

This appendix provides further details on the implementation of multi-axis model predictive contouring control on a profile cutting machine model, as discussed in Chapters 7 and 8.

C.1 Joint controller gains

The cascaded PI joint controller gains are given in Table C.1.

<table>
<thead>
<tr>
<th>Joint #</th>
<th>Position loop P' gain</th>
<th>Velocity loop P' gain</th>
<th>I' gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>500</td>
<td>300</td>
<td>500</td>
</tr>
<tr>
<td>2</td>
<td>500</td>
<td>300</td>
<td>500</td>
</tr>
<tr>
<td>3</td>
<td>500</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>4</td>
<td>500</td>
<td>20</td>
<td>200</td>
</tr>
<tr>
<td>5</td>
<td>500</td>
<td>20</td>
<td>200</td>
</tr>
</tbody>
</table>

Table C.1: Cascaded PI gains for profile cutting machine joint controllers

C.2 Controller models for profile cutting machine simulations

The internal model matrices used to simulate multi-axis model predictive contouring control on the profile cutting machine model are given in the following sections.
C.2.1 Double integrator model

As mentioned in Chapter 7, a double integrator model at a sample period of 8 ms is used to generate state feedback. The double integrator model equations in state-space form are:

\[
\begin{align*}
\xi_{k+1}^{OL} &= A^{OL} \xi_k^{OL} + B^{OL} u_k, \\
z_k &= C^{OL} \xi_k^{OL},
\end{align*}
\]  

**(C.1)**

where

\[
A^{OL} = \begin{bmatrix}
a^{OL} & 0 & 0 & 0 & 0 \\
0 & a^{OL} & 0 & 0 & 0 \\
0 & 0 & a^{OL} & 0 & 0 \\
0 & 0 & 0 & a^{OL} & 0 \\
0 & 0 & 0 & 0 & a^{OL}
\end{bmatrix}, \\
B^{OL} = \begin{bmatrix}
b^{OL} & 0 & 0 & 0 & 0 \\
0 & b^{OL} & 0 & 0 & 0 \\
0 & 0 & b^{OL} & 0 & 0 \\
0 & 0 & 0 & b^{OL} & 0 \\
0 & 0 & 0 & 0 & b^{OL}
\end{bmatrix}, \\
C^{OL} = \begin{bmatrix}
c^{OL} & 0 & 0 & 0 & 0 \\
0 & c^{OL} & 0 & 0 & 0 \\
0 & 0 & c^{OL} & 0 & 0 \\
0 & 0 & 0 & c^{OL} & 0 \\
0 & 0 & 0 & 0 & c^{OL}
\end{bmatrix}, \\
a^{OL} = \begin{bmatrix}
1 & 0 & 0.008 \\
0 & 1 & 0
\end{bmatrix}, \\
b^{OL} = \begin{bmatrix}
0 \\
0.008
\end{bmatrix}, \\
c^{OL} = \begin{bmatrix}
1 & 0
\end{bmatrix}.
\]  

**(C.2)**  

**(C.3)**

The double integrator model is used as the prediction model for open loop MPCC, described in Section 8.2.
C.2 Controller models for profile cutting machine simulations

C.2.2 Prediction model for MPCC with joint feedback

The internal model includes a model of the closed-loop system incorporating the cascaded PI joint controllers and a unit delay. The closed-loop model was identified by stimulating each joint with a chirp signal (in simulation) with frequency range of 0.001-0.5 Hz. Logged input-output data from the simulation was then put through MATLAB’s \texttt{pem} function to obtain closed-loop models for each joint.

The equations for the model shown in Figure 7.6 are

\[
\bar{\xi}_{k+1}^J = \bar{A}^J \bar{\xi}_k^J + \bar{B} w_k,
\]
\[
z_k = \bar{C} \bar{\xi}_k^J,
\]  

where

\[
\bar{A}^J = \begin{bmatrix}
A^{OL} & 0 & 0 & 0 & 0 \\
B^{CL} C^{OL} & A^{CLJ} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
B^{CL} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{bmatrix},
\]

\[
\bar{B} = \begin{bmatrix}
B^{OL} \\
B^{CL} \\
0 \\
0 \\
0 \\
\end{bmatrix},
\]

\[
\bar{C} = \begin{bmatrix}
0 & C^{OL} \\
0 & 0 \\
\end{bmatrix},
\]

\[
A^{CLJ} = \begin{bmatrix}
a_1^{CLJ} & 0 & 0 & 0 & 0 \\
0 & a_2^{CLJ} & 0 & 0 & 0 \\
0 & 0 & a_3^{CLJ} & 0 & 0 \\
0 & 0 & 0 & a_4^{CLJ} & 0 \\
0 & 0 & 0 & 0 & a_5^{CLJ} \\
\end{bmatrix},
\]

\[
B^{CL} = \begin{bmatrix}
b_1^{CL} & 0 & 0 & 0 & 0 \\
0 & b_2^{CL} & 0 & 0 & 0 \\
0 & 0 & b_3^{CL} & 0 & 0 \\
0 & 0 & 0 & b_4^{CL} & 0 \\
0 & 0 & 0 & 0 & b_5^{CL} \\
\end{bmatrix},
\]

\[
a_1^{CLJ} = \begin{bmatrix}
0.2547 & 0.7460 \\
0 & 0 \\
\end{bmatrix},
a_2^{CLJ} = \begin{bmatrix}
0.2943 & 0.7071 \\
0 & 0 \\
\end{bmatrix},
a_3^{CLJ} = \begin{bmatrix}
0.2000 & 0.8001 \\
0 & 0 \\
\end{bmatrix},
a_4^{CLJ} = \begin{bmatrix}
0.2002 & 0.7999 \\
0 & 0 \\
\end{bmatrix},
a_5^{CLJ} = \begin{bmatrix}
0.1988 & 0.8013 \\
0 & 0 \\
\end{bmatrix},
b^{CL} = \begin{bmatrix}
0 \\
1 \\
\end{bmatrix}.
\]  

(C.4)

(C.5)

(C.6)

(C.7)
Note that the internal model does not capture any interaction between joints; these are treated as disturbances and compensated with a constant output disturbance model as described in Chapter 7. The internal model equations used to implement closed-loop MPCC, including the disturbance model, are as follows:

\[
\begin{align*}
\xi_{J_{k+1}} &= \begin{bmatrix} \tilde{A}^J & 0 \\ \tilde{C} \tilde{A}^J & I \end{bmatrix} \xi_{J_k} + \begin{bmatrix} \tilde{B} \\ \tilde{C} \tilde{B} \end{bmatrix} u_k, \\
\end{align*}
\]

\[
z_k = \begin{bmatrix} 0 & I \end{bmatrix} \xi_{J_k}.
\]

\[\text{(C.8)}\]

C.2.3 Prediction model for MPCC with motor feedback

The model used for MPCC with motor feedback was identified using the same method as was used to identify the prediction model with joint feedback. The model equations are as follows:

\[
\begin{align*}
\tilde{\xi}_{M_{k+1}} &= A^M \tilde{\xi}_{M_k} + B u_k, \\
\tilde{z}_k &= \tilde{C} \tilde{\xi}_{J_k}, \\
\end{align*}
\]

where

\[
\tilde{A}^M = \begin{bmatrix} A^{OL} & 0 \\ B^{CL} C^{OL} & A^{CLM} \end{bmatrix},
\]

\[\text{(C.9)}\]

\[
A^{CLM} = \begin{bmatrix} a_1^{CLM} & 0 & 0 & 0 & 0 \\ 0 & a_2^{CLM} & 0 & 0 & 0 \\ 0 & 0 & a_3^{CLM} & 0 & 0 \\ 0 & 0 & 0 & a_4^{CLM} & 0 \\ 0 & 0 & 0 & 0 & a_5^{CLM} \end{bmatrix},
\]

\[\text{(C.10)}\]
C.2 Controller models for profile cutting machine simulations

\[ a_{CLM}^1 = \begin{bmatrix} 0.2542 & 0.7465 \\ 0 & 0 \end{bmatrix}, \quad a_{CLM}^2 = \begin{bmatrix} 0.2926 & 0.7085 \\ 0 & 0 \end{bmatrix}, \quad a_{CLM}^3 = \begin{bmatrix} 0.2000 & 0.8001 \\ 0 & 0 \end{bmatrix}, \quad a_{CLM}^4 = \begin{bmatrix} 0.2002 & 0.7999 \\ 0 & 0 \end{bmatrix}, \quad a_{CLM}^5 = \begin{bmatrix} 0.1988 & 0.8013 \\ 0 & 0 \end{bmatrix}, \quad b_{CL} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \]  

(C.12)

C.2.4 Prediction models for reduced PI gain

For the simulations discussed in Section 8.3 where the proportional gain in the position loop of joint 1 was reduced by 50%, the closed-loop system models for joint 1 were re-identified as follows:

\[ a_{I}^1 = \begin{bmatrix} 0.4058 & 0.5953 \\ 0 & 0 \end{bmatrix}, \quad a_{M}^1 = \begin{bmatrix} 0.4055 & 0.5956 \\ 0 & 0 \end{bmatrix}. \]  

(C.13)

All the other system matrices are unchanged by the reduced PI gain.
Author/s:
LAM, DENISE

Title:
A model predictive approach to optimal path-following and contouring control

Date:
2012

Citation:

Persistent Link:
http://hdl.handle.net/11343/37555

File Description:
A model predictive approach to optimal path-following and contouring control

Terms and Conditions:
Terms and Conditions: Copyright in works deposited in Minerva Access is retained by the copyright owner. The work may not be altered without permission from the copyright owner. Readers may only download, print and save electronic copies of whole works for their own personal non-commercial use. Any use that exceeds these limits requires permission from the copyright owner. Attribution is essential when quoting or paraphrasing from these works.