Multi-axis model predictive contouring control

Denise Lam, Chris Manzie* and Malcolm C. Good

Department of Mechanical Engineering, The University of Melbourne, Melbourne, Australia

(Received 8 July 2012; final version received 20 January 2013)

Contouring systems involve competing control objectives of maximising accuracy while minimising traversal time. A previously developed model predictive contouring controller for biaxial systems is extended to multi-axis systems subject to joint acceleration and jerk constraints. This requires consideration of manipulator forward kinematics and both position and orientation of the end effector. The control design is based on minimising a cost function which reflects the trade-off between the control objectives. A new architecture is proposed where the joint position controllers operate at a sample rate comparable to industrial machines, while the contouring control scheme operates at a slower rate. The proposed approach is applied to a simulation model of an industrial profile cutting machine. A number of implementations are presented requiring varying degrees of modification to the existing machine hardware and sensing capability. Results demonstrate the effect of the cost function weights on contouring accuracy and traversal time, as well as the trade-off between achieving the best contouring performance and minimising modification of the existing system.

Keywords: model predictive control; contouring; path-following; motion control

1. Introduction

Control of multi-axis contouring systems involves accurate, high speed tracking of a predetermined geometric path. Industrial applications include machine tools and laser profiling. Such systems are often subject to actuator constraints which limit the acceleration capabilities of the machine. Contouring control differs from tracking control in that the path to be followed is not time-dependent; as a result there is an extra degree of freedom in selecting the time parameterisation of the path.

In current industrial contouring systems, a trajectory planning routine is used to convert the desired path to a time-dependent reference trajectory, which is then tracked using feedback controllers. It is often desired to traverse the path at high speed to maximise productivity. However, due to the constraints and dynamics of the system, this may lead to reduced accuracy. As a result, time optimal planning of the reference trajectory is of significant interest.

A number of approaches have been proposed in order to optimise the time parameterisation of the reference trajectory such that the contour follows a desired geometric path exactly in minimum time. These include dynamic programming methods (Pfeiffer & Johanni, 1987; Shin & McKay, 1986) and phase plane techniques (Bobrow, Dubowsky, & Gibson, 1985; Shiller & Lu, 1992; Shin & McKay, 1985; Slotine & Yang, 1989). In the latter, a numerical search is employed to find the switching points between minimum and maximum path acceleration. Two-pass algorithms based on velocity and acceleration limits were proposed by Renton and Elbestawi (2000), Dong and Stori (2007), and later Dong, Ferreira, and Stori (2007) with the addition of jerk constraints. Verscheure, Demeulenaere, Swevers, Schutter, and Diehl (2009) reformulate the minimum-time path-following problem as a convex optimisation problem, resulting in efficient calculation of the optimal solution. All of these approaches generate open loop solutions which in practice are used as reference trajectories for feedback controllers.

In contouring applications, it is often desirable to sacrifice contouring accuracy to allow the path to be traversed faster. Imamura and Kaufman (1991) optimise the reference trajectory and the tuning parameters of the feedback controller together such that the tracking accuracy is below a specified tolerance, for a given contour. Naturally, as the tolerance is increased, the time to complete the trajectory decreases. An offline two-stage iterative optimisation is used, first optimising the reference trajectory and then the controller tuning parameters. Erkorkmaz, Yeung, and Altintas (2006) use spline fitting techniques to smooth out sharp corners in the contouring path. The cornering feed rate is then tuned manually using a simulation model until a desired level of accuracy is achieved. As these offline trajectory planning approaches do not take feedback into account, disturbances or modelling errors may cause performance deterioration or constraint violation.
Several researchers have attempted to address this problem by modifying the reference trajectory online using available feedback. In the reference governor approach (Bemporad, Casavola, & Mosca, 1997; Gilbert & Kolmanovsky, 2002), the reference trajectory is modified online in order to avoid constraint violation. The position reference is selected to be as close as possible to the original reference trajectory while ensuring that the system constraints are satisfied. The time parameterisation of the trajectory remains unchanged, and as a result, poor tracking performance with respect to the original path can occur. In the machine tool literature, Susanu and Dumur (2006) 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.

Another approach which adjusts the speed of the reference trajectory online is trajectory time scaling, proposed by Dahl and Nielsen (1990). The path acceleration and velocity constraints are calculated, and if violation of these constraints is detected, a time scaling factor is applied to slow the reference down. A similar scheme is proposed by Kieffer, Cahill, and James (1997), where the path acceleration is governed by a feedback control law which is designed to compensate for worst case disturbances. These techniques all involve modification of a pre-determined reference trajectory.

The path governor approach (Bemporad, Tam, & Xi, 1999) generates a time parameterisation of the path online in a receding horizon fashion, and hence does not require a pre-computed reference trajectory. Using a model of the closed loop system, a constrained optimisation is solved at each sample with the intention of minimising traversal time while honouring state and input constraints, as well as maintaining a prescribed level of accuracy. However, the path governor does not allow the reference position for each axis to deviate from the path. Meanwhile, the reference governor (Bemporad et al., 1997; Gilbert & Kolmanovsky, 2002) discussed earlier can affect the reference positions, but not the path speed. In contrast, the approach proposed in this paper selects the path speed and the reference positions simultaneously.

In previous work, a model predictive contouring control (MPCC) scheme was developed for biaxial systems (Lam, Manzie, & Good, 2013). Based on the receding horizon framework proposed by Faulwasser, Kern, and Findeisen (2009), a constrained optimisation is solved at each time step in order to determine the path speed and control inputs. Contouring control objectives of accuracy and productivity are explicitly addressed using the finite horizon cost function, allowing the system to deviate from the desired path in order to increase productivity. A linear time-varying formulation is proposed to reduce computational complexity, allowing the control scheme to be successfully implemented in real-time on an X-Y table.

In this paper, MPCC is extended to multi-axis systems subject to joint acceleration and jerk constraints with a view towards industrial implementation. In comparison to biaxial contouring, 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. A multi-rate architecture is proposed where the MPCC scheme is used to generate position command signals for each joint, to be tracked by low level feedback controllers operating at a faster sample rate. In this way, position control may occur at sample rates comparable to those used in industry while keeping the computational burden of MPCC within a reasonable level.

The proposed multi-axis MPCC scheme is implemented in simulation on an industrial profile cutting machine model. Three distinct implementations of MPCC are proposed, corresponding to different levels of feedback and modification to the current hardware. Simulation 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 successive modification of the existing industrial hardware.

2. System architecture

The proposed machine architecture including the model predictive contouring controller is shown in Figure 1. The MPCC algorithm generates a joint command acceleration $u_k \in \mathbb{R}^n$, where $k$ denotes the current time step. The double integrator converts the acceleration $u_k$ into a joint position reference $q^d$ which is tracked by the joint controllers, operating at a faster sample rate compared to the model predictive contouring controller. The subscript $t$ indicates that $q^d$ 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 MPCC 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

```
\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure1.png}
\caption{Multi-axis MPCC architecture.}
\end{figure}
```
where $o$ denotes the system states and $q \in \mathbb{R}^n$ is the vector of positions of each joint. Note that even if the plant is nonlinear, the combined dynamics of the plant and joint controller may often be reasonably approximated with a linear system.

The pose of the machine’s end effector is determined by the forward kinematics (see, for example Spong, Hutchinson, & Vidyasagar, 2006), and is a nonlinear function of the joint positions:

$$\vec{r}_k = A\vec{\theta}_k + Bu_k,$$

$$q_k = C\vec{\theta}_k,$$  \hspace{1cm} (2)

where $\vec{r} \in \mathbb{R}^n$ denotes the end effector pose in Cartesian space. It is convenient for control applications to express the end effector pose in a single vector $z \in \mathbb{R}^{12}$:

$$z = \begin{bmatrix} \begin{bmatrix} o \\ 0 \\ 1 \end{bmatrix} \end{bmatrix} = f_{FK}(q),$$

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

The objective is to steer the end effector pose $z$ along a continuously differentiable and bounded geometric path, parameterised by a scalar $\theta$:

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

$$z^d(\theta) = \begin{bmatrix} o^d(\theta) \\ r_1^d(\theta) \\ r_2^d(\theta) \\ r_3^d(\theta) \end{bmatrix}.$$  \hspace{1cm} (6)

It is assumed that the parameterisation of the path satisfies

$$\frac{ds}{d\theta} = 1,$$  \hspace{1cm} (7)

where $s$ is the path distance as defined by the application. For example, in a profile cutting machine, $s$ would be defined as the distance travelled by the cutting beam on the workpiece. The path speed $\dot{s}$ is subject to bound constraints which vary along the length of the path:

$$0 \leq \dot{s}_{\text{min}}(\theta) \leq \dot{s} \leq \dot{s}_{\text{max}}(\theta).$$  \hspace{1cm} (8)

Note that Equation (8) implies that path reversal is not allowed. Since $\theta \in [\theta^s, 0]$, it is necessary that $\dot{s}_{\text{min}}(0) = 0$, to ensure feasibility at $\theta = 0$.

Contouring accuracy is measured by the contouring error, defined here as the minimum distance from the desired path with respect to the norm defined by a positive semidefinite matrix $Q_c \in \mathbb{R}^{12 \times 12}$:

$$e_k^c = \min_{\theta} \sqrt{(z_k^d(\theta))^T Q_c (z_k^d(\theta))}.$$  \hspace{1cm} (9)

The multi-objective control problem involves selecting the joint command acceleration $u$ such that the solutions of Equation (2) traverse near the desired geometric path, in a manner that minimises contouring error (9) while maximising path speed.

4. **Multi-axis model predictive contouring control**

As in Lam et al. (2013), a finite horizon cost function is proposed which reflects competing control objectives of minimising contouring error and maximising path speed. The following dynamics are introduced for the path parameter $\theta$:

$$\theta_{k+1} = \theta_k + \nu_k,$$  \hspace{1cm} (10)

where $\nu_k$ is a virtual input to be determined by the controller and $\theta_k$ denotes the value of the path parameter at time $k$. Since the path is parameterised according to Equation (7), $\nu$ is directly proportional to the path speed $\dot{s}$. Also, non-reversal of the path is guaranteed, since $\nu_k \geq 0$.

From Equation (8), $\nu_k$ is subject to the following constraints:

$$v_{\text{min}}(\theta_k) \leq \nu_k \leq v_{\text{max}}(\theta_k),$$  \hspace{1cm} (11)

where

$$v_{\text{min}}(\theta) = h\dot{s}_{\text{min}}(\theta), \quad v_{\text{max}}(\theta) = h\dot{s}_{\text{max}}(\theta),$$  \hspace{1cm} (12)

and $h$ denotes the MPCC sample period. The bounds $v_{\text{min}}(\theta)$ and $v_{\text{max}}(\theta)$ depend on $\theta$, allowing different path speed.
constraints to be specified for different sections of the path. This is a further extension to Lam et al. (2013), where only a constant $v_{\text{max}}$ was imposed.

It is proposed to use $\theta_k$, whose evolution is governed by Equation (10), to develop an approximation to the contouring error (9).

**Assumption 4.1:**

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

By Assumption 4.1, $\epsilon^c$ can be approximated by

$$\tilde{\epsilon}_k^c = \sqrt{(z_k - z^d(\theta_k))^T \dot{Q}_c(z_k - z^d(\theta_k))}. \quad (13)$$

The model predictive cost function $J_k$ represents the relative importance of contouring accuracy and path speed over a horizon of $N$ time steps. The $\theta_{k+i, k}$ term is negated as it is desired to advance along the path as far as possible, which corresponds to maximising $\theta$ across the horizon. Control input deviations are also penalised in order to achieve smooth control inputs.

$$J_k = \sum_{i=1}^N \left( q_c (\tilde{\epsilon}_{k+i, k}^c)^2 - q_\theta \theta_{k+i, k} \right) + \frac{[\Delta v_{k+i-1, k}]^T R \Delta v_{k+i-1, k}}{\Delta u_{k+i-1, k}} \right), \quad (14)$$

where $R = \text{diag}(r_v, r_u)$ and $r_u \in \mathbb{R}_{>0}$, $r_v$, $q_c$, $q_\theta > 0$. The notation $\xi_{k+i, k}$ denotes the prediction of $\xi$ at time $k+i$, predicted at time $k$. The penalty weightings $q_c$, $q_\theta$, $r_u$ and $r_v$ are tuning parameters to be decided based on the relative importance of contouring accuracy, productivity and smoothness of the control inputs.

**Remark 1:** Assumption 4.1 is enforced by setting the contouring error weighting $q_c$ sufficiently large in the cost function (14).

Let $Q_c = q_c \dot{Q}_c$ and

$$\tilde{z}_k = z_k - z^d(\theta_k). \quad (15)$$

Then, combining Equations (13) and (15), the cost function (14) becomes

$$J_k = \sum_{i=1}^N \left( \tilde{z}_{k+i, k}^T Q_c \tilde{z}_{k+i, k} - q_\theta \theta_{k+i, k} \right) + \frac{[\Delta v_{k+i-1, k}]^T R \Delta v_{k+i-1, k}}{\Delta u_{k+i-1, k}} \right), \quad (16)$$

The cost function (16) leads to the following optimisation problem being posed at each time interval, $k$:

Minimise $J_k$,  

Subject to

\begin{equation}
\begin{aligned}
\dot{\xi}_{k+i, k} &= A \xi_{k+i-1, k} + B u_{k+i-1, k}, \\
\theta_{k+i, k} &= \theta_{k+i-1, k} + v_{k+i-1, k}, \\
\xi_k &= \xi_k, \quad \theta_k = \theta_k, \\
u_{k+i-1, k} &\in [u_{\min}, u_{\max}], \quad \Delta u_{k+i-1, k} \in [\Delta u_{\min}, \Delta u_{\max}], \\
v_{k+i-1, k} &\in [v_{\min}(\theta_{k+i-1}), v_{\max}(\theta_{k+i-1})], \\
\theta_{k+i, k} &\in [\theta^u, 0], \quad i = 1, \ldots, N.
\end{aligned}
\end{equation}

The multi-axis optimisation problem (17) is significantly more complex than an equivalent optimisation problem for a biaxial system. Assuming each joint has a single input, the optimisation problem would involve twice the number of decision variables for a 5-joint multi-axis system compared to a biaxial system.

The optimisation (17) 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.

### 4.1 Linear time-varying (LTV) MPCC

The optimisation (17) is nonlinear and therefore computationally difficult to solve online. Following the technique described in Lam et al. (2013), a linear time-varying approach is employed to approximate the optimisation problem with a quadratic programme (QP). In order to approximate Equation (17) with a QP, it is necessary to approximate the cost function (16) with a quadratic function of the control input trajectories, and also replace the state-dependent path speed constraint (11) with an approximate time-dependent constraint.

#### 4.1.1 Cost function approximation

An approximate cost function is developed by linearising both the path function $z^d(\theta)$ and forward kinematics function $f_{FK}(q)$ 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 $\hat{\Theta}_k^*$ predicted at time $k+i$ using information available at time $k$, define linear approximations of $z$ and $z^d$ as follows:

$$z^d_{k+i}(\hat{\xi}_{k+i, k}, \hat{\Theta}_k^*) = f_{FK}(C\hat{\xi}_{k+i, k}) + \nabla f_{FK}(C\hat{\xi}_{k+i, k}) \times (C(\xi_{k+i, k} - \hat{\xi}_{k+i, k})), \quad (18)$$

$$z_{k+i}^d(\theta_{k+i, k}, \hat{\Theta}_k^*) = z^d(\hat{\Theta}_{k+i, k}) \times (\theta_{k+i, k} - \hat{\theta}_{k+i, k}). \quad (19)$$

The multi-axis optimisation problem (17) is significantly more complex than an equivalent optimisation problem for a biaxial system. Assuming each joint has a single input, the optimisation problem would involve twice the number of decision variables for a 5-joint multi-axis system compared to a biaxial system.

The optimisation (17) 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.

The multi-axis optimisation problem (17) is significantly more complex than an equivalent optimisation problem for a biaxial system. Assuming each joint has a single input, the optimisation problem would involve twice the number of decision variables for a 5-joint multi-axis system compared to a biaxial system.

The optimisation (17) 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.
The linearised Equations (18) and (19) are based on Taylor series expansions around the estimated state trajectories. As stated by Campa and de la Torre (2009), 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 \( S(\cdot) \) is defined such that given a vector \( a = [a_1, a_2, a_3]^T \),

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

Equations (18) and (19) are combined to form an approximation to \( \hat{z}_k \),

\[
\hat{z}_{k+i;k}^a(\tilde{\xi}_k^*, \tilde{\Theta}_k^*) = z_{k+i;k}(\hat{\xi}_k, \hat{\Theta}_k) - z_{k+i;k}^a(\hat{\xi}_k - \xi_k, \hat{\Theta}_k).
\]

where the dependence of \( z_{k+i;k}^a \) on the approximate state trajectories \( \tilde{\xi}_k^*, \tilde{\Theta}_k^* \) is emphasised in the notation.

An approximate cost function can then be expressed using Equation (22):

\[
J_k^a(\tilde{\xi}_k^*, \tilde{\Theta}_k^*) = \sum_{i=1}^{N} \left( z_{k+i;k}^a(\tilde{\xi}_k^*, \tilde{\Theta}_k^*)^T Q z_{k+i;k}^a(\tilde{\xi}_k^*, \tilde{\Theta}_k^*) - q \partial_{\Theta_k} \right) R \left( \frac{\Delta u_{k+i-1;k}}{\Delta u_{k+i-1;k}} \right) ^T \left( \frac{\Delta u_{k+i-1;k}}{\Delta u_{k+i-1;k}} \right),
\]

(23)

The approximate cost function (23) is quadratic and convex in the input trajectories \( u_k \) and \( v_k \). Clearly, the approximation of the cost function depends on the accuracy of the state trajectory estimates \( \tilde{\xi}_k^* \) and \( \tilde{\Theta}_k^* \). A methodology for computing the state trajectory estimates will be discussed in Section 4.1.3.

4.1.2 Constraint approximation

In order to simplify the optimisation problem (17), the state-dependent constraint (11) is approximated with a time-dependent constraint, so that over the horizon,

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

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

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

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

\( i \in [1, \ldots, N] \).

(24)

As the time-dependent constraint (24) varies across the horizon, the controller can anticipate changes to the minimum and maximum path speed and act accordingly. In addition, the estimated trajectory \( \tilde{\Theta}_k^* \) is already used to obtain the approximate cost function (23), so there is little additional computation required.

Using Equations (23) and (24), a simplified optimisation problem can be posed:

Minimise \( J_k^a(\tilde{\xi}_k^*, \tilde{\Theta}_k^*) \), Subject to

\[
\xi_{k+i;k} = A\xi_k + Bu_{k+i-1;k} + \theta_{k+i;k} = \theta_{k+i-1;k} + \xi_k + v_{k+i-1;k},
\]

\[
\theta_{k+i-1;k} = \theta_k, u_{k+i-1;k} \in [u_{min}, u_{max}], \Delta u_{k+i-1;k} \in [\Delta u_{min}, \Delta u_{max}], \n\]

\[
\hat{v}_{min}(k + i - 1, \tilde{\Theta}_k^*), \hat{v}_{max}(k + i - 1, \tilde{\Theta}_k^*), \theta_{k+i;k} \in [\theta^*, 0], i = 1, \ldots, N.
\]

4.1.3 Trajectory approximation

Computing the linearised predictions (18) and (19) as well as the approximate path speed constraints (25) requires estimates of the state trajectories \( \tilde{\xi}_k^* \) and \( \tilde{\Theta}_k^* \). State trajectories are estimated using the optimal input trajectories from the previous time step, as described in the following.

Let \( \tilde{u}_{k-1}^* \) and \( \tilde{u}_{k-1}^* \) denote the optimal input trajectories obtained by solving Equation (26) at time \( k - 1 \). An approximate virtual input trajectory for the current time step can be obtained by truncating \( \tilde{v}_{k-1}^* \) and appending a feasible input \( \tilde{v}_{k+N-1}^* = \min(v_{k+N-2,k-1}^*, v_{k+N-2,k-1}) \) so that \( \tilde{v}_k^* = \{\tilde{v}_{k-1}^*, \ldots, \tilde{v}_{k+N-2,k-1}^*, \tilde{v}_{k+N-1}^*\} \). Similarly, an approximate plant input trajectory is computed by truncating \( \tilde{u}_{k-1}^* \) and appending \( \tilde{u}_{k+N-1}^* = u_{k+N-2,k-1}^*, \tilde{u}_{k+N-1}^* \), yielding \( \tilde{u}_k^* = \{u_{k-1}^*, \ldots, u_{k+N-2,k-1}^*, u_{k+N-1}^*\} \). The approximate state trajectories \( \tilde{\Theta}_k^* \) and \( \tilde{\xi}_k^* \) are computed by applying \( \tilde{v}_k^* \) and \( \tilde{u}_k^* \) to Equations (10) and (2) with initial conditions \( \tilde{\xi}_0^* = \xi_0, \tilde{\Theta}_0^* = \theta_0 \) and \( \hat{\xi}_k^* = \xi_k \).

At the initial time step \( k = 0, \tilde{\Theta}_0^* \) and \( \tilde{\xi}_0^* \) can be computed via the following iterative procedure.

Procedure 4.2: Initial trajectory estimation

(1) Initialise \( \tilde{\Theta}_0^* \) to \( \tilde{\Theta}_0^* = \{\theta_0, \theta_0, \ldots, \theta_0\} \), \( \tilde{\xi}_0^* \) to \( \tilde{\xi}_0^* = \{\xi_0, \xi_0, \ldots, \xi_0\} \) and set \( j = 0 \).
The linear time-varying MPCC algorithm is summarised as follows.

Algorithm 1: LTV multi-axis MPCC

1. Initialise $k = 0$, $\theta = \theta_0$ and calculate $\mathbf{\Theta}_0$, $\mathbf{\hat{\Theta}}_0$ using Procedure 4.2.
2. Compute the linearised error function $z_{k+1,k}^* (\mathbf{\Theta}_0, \mathbf{\hat{\Theta}}_0)$ using Equations (18)–(22).
3. Calculate the approximate path speed constraints $\dot{v}_{min}, \dot{v}_{max}$ from Equation (25) using $\mathbf{\hat{\Theta}}_0$.
4. Solve the optimisation (26) to obtain $\mathbf{u}_0^j$ and $\mathbf{v}_0^j$.
5. Set $\mathbf{\hat{\Theta}}_0^{j+1} = \mathbf{\Theta}_0^j$, $\mathbf{\hat{\Xi}}_0^j = \mathbf{\Xi}_0^j$ and increment $j$.
6. Repeat steps 2–5 until $||\mathbf{\hat{\Theta}}_0^j - \mathbf{\Theta}_0^j|| + ||\mathbf{\hat{\Xi}}_0^j - \mathbf{\Xi}_0^j|| \leq \epsilon$, for some $\epsilon > 0$, or until some iteration limit is exceeded.

The estimated trajectories $\mathbf{\hat{\Theta}}_k^*$ and $\mathbf{\hat{\Xi}}_k^*$ 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 (26) 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 (11) is replaced by the approximate time-dependent constraint (24), it is important to ensure that the solution to the simplified optimisation problem (26) does not violate the original constraint (11). Recall that $\hat{\theta}_{k,k}^* = \hat{\theta}_k$. It follows that

$$\dot{v}_{min}(k, \mathbf{\hat{\Theta}}_k^*) = v_{min}(\hat{\theta}_{k,k}^*) = v_{min}(\hat{\theta}_k),$$

$$\dot{v}_{max}(k, \mathbf{\hat{\Theta}}_k^*) = v_{max}(\hat{\theta}_{k,k}^*) = v_{max}(\hat{\theta}_k),$$

which satisfies Equation (11). Since only the first element of $\mathbf{v}_k$ is ever applied, Equation (11) is satisfied for all $k$.

Remark 2: In order to satisfy Equation (11), it is only necessary to enforce Equation (24) for $i = 1$. Imposing the estimated constraint across the horizon is important only from a performance point of view.

The multi-axis MPCC algorithm with linear time-varying approximation may be summarised as follows.

Algorithm 1: LTV multi-axis MPCC

1. Initialise $k = 0$, $\theta = \theta_0$ and calculate $\mathbf{\Theta}_0^*$, $\mathbf{\hat{\Theta}}_0$ using Procedure 4.2.
2. Compute the linearised error function $z_{k+1,k}^* (\mathbf{\Theta}_0^*, \mathbf{\hat{\Theta}}_0)$ using Equations (18)–(22).
3. Calculate the approximate path speed constraints $\dot{v}_{min}, \dot{v}_{max}$ from Equation (25) using $\mathbf{\hat{\Theta}}_0$.
4. Solve the approximate MPCC optimisation (26) to obtain optimal input trajectories $\mathbf{u}_k^*$ and $\mathbf{v}_k^*$.
5. Apply the first element of $\mathbf{u}_k^*$ to the plant and use the first element of $\mathbf{v}_k^*$ to update $\theta_{k+1}$ via Equation (10).
6. Calculate $\mathbf{\hat{u}}_{k+1}^*$, $\mathbf{\hat{v}}_{k+1}^*$ by truncating $\mathbf{u}_k^*$, $\mathbf{v}_k^*$ and compute $\mathbf{\hat{\Theta}}_{k+1}$ and $\mathbf{\hat{\Xi}}_{k+1}$ by applying $\mathbf{\hat{v}}_{k+1}^*$ and $\mathbf{\hat{u}}_{k+1}^*$ to Equations (2) and (10).
7. Increment $k$ and return to step (2).

The linear time-varying MPCC algorithm is simulated on a profile cutting machine model, described in the following.

5. Simulation model

The simulation model is based on an industrial profile cutting machine, whose configuration is shown in Figure 2. 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 perpendicular to the workpiece. Figure 2(a) shows the translation axes which allow motion in the Left/Right, Fore/Aft and Up/Down directions. The two rotational axes are referred to as the swivel and tilt joints. The axis of rotation of the tilt joint is oriented at an angle of 45°, as shown in Figure 2(b).

The machine configuration is defined using standard Denavit–Hartenberg parameters (see Spong et al., 2006) given in Table 1.

The simulation model of the profile cutting machine was developed using the Robotics Toolbox for MATLAB (Corke, 1996). 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.

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

$$J_i^m \dot{q}_i^m = T_i^m - b_i^m \dot{q}_i^m - N_i^G (k_i^m (q_i^m - q_i) + c_1 (\dot{q}_i^m - \dot{q}_i)), $$

$$J_2^m \dot{q}_2^m = T_2^m - b_2^m \dot{q}_2^m - N_2^G (k_2^m (q_2^m - q_2) + c_2 (\dot{q}_2^m - \dot{q}_2)), $$

where for the $i$th joint

- $q_i^m$ is the motor position, expressed in (m) for prismatic joints or (rad) for revolute joints,
- $q_i$ is the joint position, (m) or (rad),
- $J_i^m$ is the motor inertia (kg m$^2$),
- $T_i^m$ is the motor torque (Nm),
- $b_i^m$ 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),
\( N_i^G \) 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 directly related to the joint positions \( q_3, q_4 \) and \( q_5 \):
\[
q_3 = q_m^3, \\
q_4 = q_m^4, \\
q_5 = q_m^5.
\] (29)

The vector of torques \( \mathbf{T}^r \) applied to the joints is then
\[
T_1^r = k_1^c(q_1^m - q_1) + c_1(\dot{q}_1^m - \dot{q}_1), \\
T_2^r = k_2^c(q_2^m - q_2) + c_2(\dot{q}_2^m - \dot{q}_2), \\
T_3^r = \frac{1}{N_3^G} T_3^m.
\] (30)

The Robotics Toolbox (Corke, 1996) 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
\[
T^r = M'(q)\ddot{q} + C'(q, \dot{q})\dot{q} + F'(\dot{q}) + G'(q),
\] (31)
where
- \( M'(q) \) is the joint space inertia matrix,
- \( C'(q, \dot{q}) \) is the Coriolis and centripetal coupling matrix,
- \( F'(\dot{q}) \) is the viscous friction torque and
- \( G'(q) \) is the load due to gravity.

While Equation (31) is nonlinear, the dynamics of the closed loop system including the joint controllers may be reasonably approximated by a linear model in the form of Equation (2).

### 5.1 Joint controllers

The joint controllers shown in Figure 1 are implemented using the industry standard cascaded PI control approach, as shown in Figure 3 for the \( i \)th joint. For simulation purposes, the motor electrical dynamics are neglected, so that the joint controllers apply torque directly to the motors.
Measurements of motor position $q^m_i$ are provided as feedback to the joint controllers at a sample period of 500 $\mu$s.

5.2 Contouring error
The desired path function $z^w(\theta)$ 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 Equation (9) to be zero. For the simulations conducted, $\tilde{Q}_c$ is defined as

$$\tilde{Q}_c = \text{diag}(1, 1, 1, 0, 0, 0, 0, 0, 1000, 1000, 1000)$$

where the end effector position, corresponding to the first three elements of $z$, is expressed in millimetres.

6. MPCC implementation
The MPCC approach developed in Section 4 is applied in simulation to the profile cutting machine model. The linear time-varying MPCC algorithm (Algorithm 1) is first implemented under the assumption that direct measurements of the joint position $q$ are available to the model predictive 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.

To address this potential shortcoming, alternative implementations using Algorithm 1 but with other levels of feedback are considered. In the first alternative, motor feedback is used to estimate $\xi_0$ in place of actual measurement. A further reduction of complexity of the Algorithm 1 through an open loop implementation is then proposed and investigated.

These stages represent a pathway from an ideal sensor feedback situation, to a more practical sensor implementation and finally no sensor feedback.

6.1 MPCC with joint feedback
The architecture of the model predictive contouring controller with joint feedback is shown in Figure 4, where $j = 1$.

The LTV multi-axis MPCC algorithm generates the joint acceleration command $u_k$ which is converted into a position command $q^d_k$ to be tracked by the joint controllers. The subscript $t$ indicates that $q^d_k$ is updated at a faster sample rate. The measured joint position $q_k$ is used as feedback, so that

$$q^{FB,1}_k = q_k$$

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. It is expected that the actual joint position $q_k$ will approximately follow $q^{d}_k$ with some lag. Therefore, the dynamics governing $q_k$ are modelled with a simple first-order discrete time system and unit delay with $q^{d}_k$ as the input. The state vector $\xi$ for MPCC with joint feedback is

$$\xi^f_k = \begin{bmatrix} \Delta q^d_k \\ \Delta q^{d}_k \\ \Delta q^{d}_{k-1} \\ q_k \end{bmatrix}.$$ 

The closed loop model was identified by applying a chirp signal to each joint and using measurements of $q_k$ to fit the model parameters.

In addition, ‘integral action’ is incorporated by converting the combined model into a velocity-form state-space model, as shown in Wang (2009). At each sample, the measured joint position $q_k$ along with $q^{d}_k$ from the double integrator model are used to calculate the current state $\xi^f_k$. This is then used in conjunction with the MPCC model to form predictions of $q_{k+1,k}$ across the horizon for use in the cost function.
Implementing MPCC with joint feedback requires the installation of sensors on the joints, rather than relying on motor feedback alone which is the case for most industrial machines. While it is expected that using joint feedback will result in superior performance, it may be preferred to use motor feedback for the implementation of MPCC, thereby reducing the amount of modification to the existing hardware.

### 6.2 MPCC with motor feedback

When implementing MPCC with motor feedback, the measured motor position \( q^m \) is used in place of \( q \). The control architecture using motor feedback may also be represented by Figure 4 with \( j = 2 \), where

\[
q^{FB,2}_k = q^m_k. \tag{35}
\]

The combined model (2) is derived in the same manner as for joint feedback MPCC, with the exception that the closed loop system is identified using \( q^m \) as the system output. Therefore, the state vector \( \xi \) for MPCC with motor feedback is

\[
\xi^M_k = \begin{bmatrix}
\Delta q^d_k \\
\Delta q^d_k \\
\Delta q^d_k \\
\Delta q^d_k \\
q^m_k
\end{bmatrix}. \tag{36}
\]

Implementing MPCC with motor feedback eliminates the requirement for additional joint position sensors. However, the achievable contouring performance compared to using direct joint feedback may be reduced.

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. Furthermore, due to computational limitations of the current hardware, it may be more practical to implement the MPCC algorithm in a quasi-real-time fashion. In the following section, MPCC is implemented in an open loop fashion. This approach requires the least modification to the existing industrial architecture.

### 6.3 Open loop MPCC

The architecture for open loop MPCC is shown in Figure 4 with \( j = 3 \). Multi-axis MPCC is used to generate a joint reference trajectory in an open loop fashion. No feedback is received by the MPCC controller from the plant, and it is assumed that the joint position is equal to the command \( q^d \). Therefore,

\[
q^{FB,3}_k = q^d_k. \tag{37}
\]

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. The state vector \( \xi \) for open loop MPCC is then

\[
\xi^{OL}_k = \begin{bmatrix} q^d_k \\ q^d_k \end{bmatrix}. \tag{38}
\]

An advantage of the open loop approach is that the MPCC algorithm need not run in real-time, 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, and contouring performance depends heavily on the performance of the joint controllers.

### 7. Simulations

The control algorithm 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 2.

<table>
<thead>
<tr>
<th>Simulation parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slow sample period</td>
<td>8 ms</td>
</tr>
<tr>
<td>Fast sample period</td>
<td>500 ( \mu )s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input constraints</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prismatic joint</td>
<td>-300 to 300 mm/s^2</td>
</tr>
<tr>
<td>Revolute joint</td>
<td>-30 to 30 rad/s^2</td>
</tr>
<tr>
<td>Prismatic joint</td>
<td>-3500 to 3500 mm/s^3</td>
</tr>
<tr>
<td>Revolute joint</td>
<td>-600 to 600 rad/s^3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPCC parameters</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 1 )</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>

The joint controllers operate at a sample period of 500 \( \mu \)s, 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 desired path used for the simulations is shown in Figure 5. A three-dimensional plot of the desired path of the end effector is shown in Figure 5(a), and Figure 5(b) shows the resulting flower-shaped path of the intersection between the cutting beam and the workpiece. The cutting
Figure 5. Desired path for multi-axis MPCC simulations. (a) Three-dimensional plot of end effector path. (b) Intersection between the cutting beam and the workpiece.

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. The path was parameterised accordingly using the technique presented in Erkorkmaz and Altintas (2005).

As discussed in Section 3, it is assumed that $z^d(\theta)$ is parameterised such that $ds/d\theta = 1$.

Figures 6–8 show plots of the joint command acceleration and jerk, path speed and contouring error respec-
Figure 6. Joint command acceleration and jerk for joint feedback MPCC for $q_\theta = 0.05$ and $q_\theta = 2$. (a) Joint command acceleration. (b) Joint command jerk.

Figure 7 shows how the controller anticipates the reduction in the maximum path speed and reduces $v$ accordingly. It can also be seen that the path speed is reduced around the tight curves on the path to maintain accuracy.

Figure 8(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 8(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 9 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 5(b) for the same simulations. It can be seen that the plots in Figures 8 and 9 are quite similar. This indicates that with appropriate choice of $\tilde{Q}_c$, the contouring error definition (9) used in the MPCC
cost function corresponds well to the error measured at the workpiece.

The difference in contouring behaviour for \( q_0 = 0.05 \) and \( q_0 = 2 \) is reflected in Figures 7 and 8. The smaller value of \( q_0 \) 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 plot of traversal time and root mean square (RMS) contouring error versus \( q_0 \) shown in Figure 10 demonstrates that this trend continues over the range of values of \( q_0 \) between 0.05 and 2.

Similar results were obtained for MPCC with motor feedback and open loop MPCC, as summarised in Figure 10.

It was observed that the three implementation approaches achieved different levels of contouring performance. This is illustrated in the following section, which compares the RMS error and traversal time for all three controllers.

### 7.1 Comparison of MPCC implementations

Figure 11 shows plots of the RMS contouring error and workpiece error versus traversal time for MPCC implementations with joint feedback, motor feedback and in open loop. 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, while open loop MPCC performs the worst. In addition, the improvement in contouring accuracy achieved by using joint feedback increases with shorter traversal times. 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.

#### 7.1.1 Axis mismatch

In contouring control architectures involving open loop trajectory planning, it is well-known that if the joint controllers are well-matched (Srinivasan & Kulkarni, 1990),
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.

Figure 12 shows 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 11 and 12 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. Consequently, while the open loop MPCC approach offers computational advantages relative to the closed loop case, this comes at the cost of a loss of robustness to errors such as those introduced by axis mismatch.

8. Conclusion
An MPCC scheme previously developed for biaxial systems has been extended to multi-axis contouring. A multirate architecture is proposed which is well-suited to industrial machines, and allows feedback to be taken into account at sample periods of 500 μs. The multi-axis MPCC algorithm is applied to a profile cutting machine model using three different strategies, each requiring different amounts of modification to the existing hardware.

Simulation results demonstrate the trade-off between minimising modification of the existing machine and maximising contouring performance.
While the sample rates in the multi-rate system were fixed here, there remains the opportunity to optimally balance the sampling rates to allow the best combination of both computation and performance levels. This investigation is left as an open question for future research.

Acknowledgement
The authors would like to thank ANCA Motion Pty Ltd for its generous support of this work.

References


