1-1-2013

Position Domain Synchronization Control For Robotic Manipulators

Vangjel Pano
Ryerson University

POSITION DOMAIN SYNCHRONIZATION CONTROL FOR ROBOTIC MANIPULATORS

By

Vangjel Pano, B.Eng (Hons.)

Ryerson University

Toronto, Ontario, Canada, 2011

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the Program of

Aerospace Engineering

Toronto, Ontario, Canada, 2013

AUTHOR'S DECLARATION

AUTHORS DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis including the required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

Abstract

Position Domain Synchronization Control for Robotic Manipulators

Vangjel Pano

A thesis for the degree of

Master of Applied Science, 2013

Department of Aerospace Engineering, Ryerson University

Developed in this thesis is a new control law focusing on the improvement of contour tracking of robotic manipulators. The new control scheme is a hybrid controller based on position domain control (PDC) and position synchronization control (PSC). On PDC, the system's dynamics are transformed from time domain to position domain via a one-to-one mapping and the position of the master axis motion is used as reference instead of time. The elimination of the reference motion from the control input improves contouring performance relative to time domain controllers. Conversely, PSC seeks to reduce the error of the systems by diminishing the synchronization error between each agent of the system. The new control law utilizes the aforementioned techniques to maximize the contour performance. The Lyapunov method was used to prove the proposed controller's stability. The new control law was compared to existing control schemes via simulations of linear and nonlinear contours, and was shown to provide good tracking and contouring performances.

Acknowledgements

This thesis would not have been possible without the help, support and patience of my supervisor, Dr. Ouyang. His knowledge and dedication has been invaluable.

I also wish to thank my lab mates Mr. Richard P. Mohamed, Mr. Yu Lin and Mr. John Michael Acob for their advice and opinions.

Finally, I would like to sincerely thank my parents, my sister, and my relatives, Stefan, Vasilika, Grigor and Maria for their support, encouragement and understanding.

Dedication

To my niece, Lydia

Table of Contents

List of Tables

List of Figures

List of Symbols

CCC	*Cross-coupled control*
CNC	*Computer numerical control*
DOF	*Degrees of freedom*
ILC	*Iterative learning control*
MRAC	*Model-reference adaptive control*
P-D	*Proportional-derivative*
PD(C)	*Position domain (control)*
PD-CCC	*Position domain cross-coupled control*
PDcg	*Proportional-derivative control with gravity compensation*
PD-PD	*Position domain proportional derivative*
PDSC	*Position domain synchronization control*
PID	*Proportional-integral-derivative controller*
PSC	*Position synchronization control*
RC	*Repetitive control*
RRR	*Three revolute joint configuration*
SMC	*Sliding mode control*
STC	*Self-tuning control*
TD(C)	*Time domain (control)*
TD-PD	*Time domain proportional derivative*

Nomenclature

A_i	*Constraint matrix*
B	*Diagonal gain matrix (synchronization)*
B_i	*Offset of i^{th} agent*
$C(q, \dot{q})$	*Vector of coriolis and centrifugal forces*
$C(t)$	*Common vector (synchronization)*
C_y, C_x	*Cross-coupled control gains*
$F(t, q, \dot{q})$	*Vector of friction forces*
$G(q)$	*Vector of gravity*
$H(q, \dot{q})$	*Sum of force vectors on system*
I_i	*Inertia of i^{th} link*
$\mathcal{J}$	*Identity matrix*
J	*Jacobian matrix*
K	*User defined gain matrix*
K_D or K_d	*Derivative gain matrix*
K_I or K_i	*Inertial gain matrix*
K_P or K_p	*Proportional gain matrix*
$M(q)$	*Inertia Matrix*
P_d	*Desired trajectory*
Q	*User defined matrix*
R	*Radius*
S	*Schur complement*
$S(\varrho, t)$	*Time-varying desired shape*
T	*Synchronization matrix*
$V(e_s(q_m), e_s'(q_m))$	*Lyapunov candidate function*
dV/dq_m	*Lyapunov derivative w.r.t master motion*
b	*Viscous friction parameter*
c_i	*PSC constraint*
$c_i(t)$	*Coupling parameter of i^{th} agent*
e	*Tracking error*
$\dot{e}$	*Velocity error*
e_c	*Contour tracking error*
e'	*Relative position error*
e'^*	*Relative derivative coupled error*
e^*	*Coupled error*
$\dot{e^*}$	*Derivative of coupled error*
f_c	*Crossover frequency*
f_c	*Coulomb friction force*
f_s	*Static friction force*
f_t	*Magnitude of Stribeck friction force*
l_i	*Length of i^{th} link*
q	*Joint/Axis (angular) Position*
$\dot{q}$	*Joint/Axis (angular) Velocity*
$\ddot{q}$	*Joint/Axis (angular) Acceleration*

q_{ef}	*End-effector angular position*
$\dot{q}_{ef}$	*End-effector angular velocity*
q_i'	*Relative position velocity*
q_i''	*Relative position acceleration*
m	*Subscript indicating master motion*
m_i	*Mass of i^{th} link*
$r(t)$	*Trajectory polynomial*
r_i	*Length from centre of gravity of i^{th} link*
x_{ef}	*Horizontal end-effector position*
y_{ef}	*Vertical end-effector position*
$\dot{x}_{ef}$	*Horizontal end-effector velocity*
$\dot{y}_{ef}$	*Vertical end-effector velocity*
s_i	*Sliding surface*
s	*Subscript indicating slave motion*
t	*time*
u	*System input*
γ	*Constant gain*
δ	*Friction parameter*
ε	*Synchronization error*
$\theta(t)$	*Angular trajectory parameter*
λ	*Constant (SMC)*
λ_m	*Smallest eigenvalue of M*
λ_M	*Larger eigenvalue of M*
ϱ	*State vector*
τ	*Torque*
φ	*Boundary layer*
ϕ	*Angle of contour*

Chapter 1: Introduction

1.1 Background

The first industrial robotic manipulator, Unimate, was created by George Devol and Joseph Engelberger in 1959. The initial design of the robotic manipulator weighted close to two tons and used hydraulic actuators and was controlled by a program on a magnetic drum. Although the first Unimate cost US$ 65,000, it was sold for only US$18,000 (International Federation of Robotics, 2012). For the next years, robotic manipulators in the industry would not significantly change in size or capabilities. The first robots were used in the metal industry (i.e. metal pipe industry) and were mostly used due to their productivity, since they could operate for more hours than a human worker. However, in the 1970s, the emergence of micro-processors and the use of electrical motors lead to increased performance and versatility for the robotic manipulators. The more precise computer-controlled electrical motors could perform more elegant acts such as assembly, welding, cutting riveting etc. This was translated in an average yearly growth of 30% (Wallen, 2008).

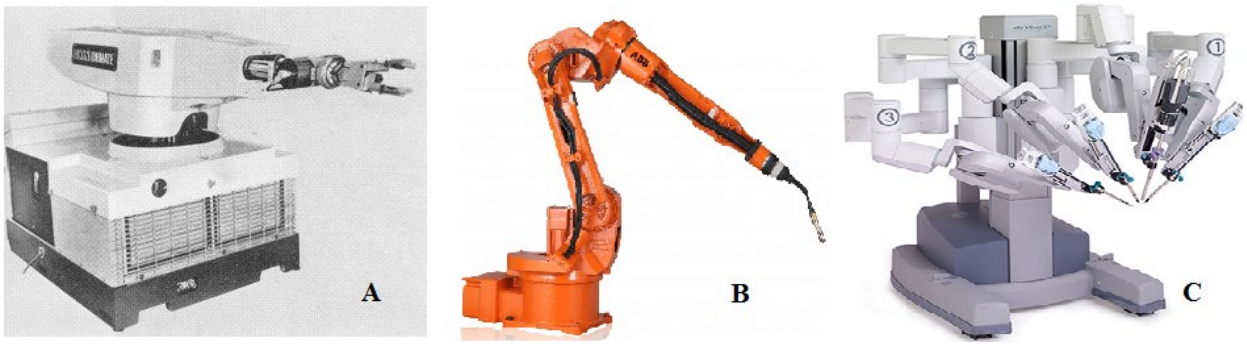

Figure 1-1: A: Unimate (Bayliss, Jones and Bayliss). B: ABB IRB 1520ID (Alliance Communications Inc.). C: DaVinci Surgical System (WINTHROP Inc.)

Nowadays, the great precision as well as the increased productivity and lower production costs associated with them has made industrial manipulators a staple in current industrial production, from electrical and electronics production to surgical procedures and space exploration. As an indication, Taiwan's Foxconn Technology Group in 2011 announced that was planning to increase its robot usage from 10,000 to one million by 2014 (Yee, 2011).

An important research area of robotics and mechatronics is control. Each control type or control law is introduced to compensate for the innate imprecisions that are present on a system as well as other errors and disturbances which arise depending on the use of the system and usually cannot be calculated or accounted for. The causes of these inaccuracies can be summarized as (Slotine J. W., 1988):

- Friction
- Vibration dynamics
- Mechanical hardware deficiencies
- Inaccurate system dynamics
- Process generated disturbances

Due to the aforementioned factors, many types of control modes have been employed depending on the application and the environment of use. Robot control can be categorized as Free Space or Constrained Space depending on the obstacles present in the path of a manipulator (Dombre E. K., 2007). Another categorization distinguishes between Force and Position control. Force control focuses on regulating the force of a robotic manipulator which comes in contact with objects in applications such as cutting, milling etc. On the other hand, Position control is

appropriate when the manipulator should follow a specified trajectory, or at least to reach a set of point in space, in applications such as welding, assembling etc (Craig, 2005).

The purpose of this thesis is to introduce position domain synchronization control as a valid and powerful control approach for the minimization of the errors in the dynamic response and the contour tracking of robotic manipulators. This is a critical area of research in the realm of control theory since a core aspect of industrial and technological evolution is the quest for faster and more accurate industrial processes.

1.2 Motivation and Objectives

The main goal of this research is to improve the contour tracking performance of a robotic manipulator. This thesis will propose a new control law based on position domain control (PDC) in order to minimize the tracking error and coordinate the motion of the joints of a serial robot. The use of this control law will result in the reduction of the contour error on the end-effector level and therefore, the ability to increase the speed and efficiency of industrial processes. The overall goals of the proposed controller can be summarized as follows:

1. To provide an alternative to time domain synchronization control laws.
2. To indicate the advantages of position domain control in nonlinear robotic systems.
3. To introduce position domain synchronization control for serial robotic manipulators.
4. To improve contouring performance for robotic manipulators.
5. To allow for faster industrial procedures without loss of accuracy.
6. To offer a simple, easy implementation of feedback control laws.

In this thesis, the proposed position domain synchronization control (PDSC) and the existent time domain position synchronization control (PSC) will be simulated for various types of contours. To exhibit the advantages of the synchronization approach, proportional-derivative control will also be simulated both in time domain (TD-PD) and position domain (PD-PD). The results of all simulations will be compared for the purpose of demonstrating the advantages of the PDSC. In summary, the following objectives will be fulfilled:

1. Model the dynamics of a simplified RRR, planar, serial manipulator.
2. Formulate a position domain synchronization controller.
3. Perform a stability analysis for the proposed control law via the Lyapunov method.
4. Compare the performance of the proposed controller and the existing time domain controller as well as well as a PD controller both in time and position domain.

1.3 Organization of Contents

The remaining thesis is organized as follows:

Chapter 2: provides a literature review on previous research and control approaches to improve contour tracking performance of robotic systems. The review covers both independent and coordinated control schemes as well as an overview of more advanced types of control.

Chapter 3: analyses the concept of position domain control (PDC) and the position synchronization control (PSC) principles. The dynamic model of a robotic system is transformed from time to position domain and a new control law is introduced which combines PDC and PSC. Lastly, a stability analysis is performed for the proposed control law using the Lyapunov method.

Chapter 4: includes a performance comparison of time domain and position domain controllers for various linear and nonlinear contours.

Chapter 5: concludes the thesis and examines the findings of the simulation results and offers suggestions for further work and development of the position domain control principle.

Chapter 2: Literature Review

Although usually assumed to be a single piece of machinery, robotic manipulators can be viewed as a group of devices that cooperate to achieve given tasks. Every robotic manipulator is nothing but a collection of actuators/motors and their respective linkage that are simultaneously used to perform motions that serve certain purposes.

In that sense, the various proposed and applied control methods for the control of robotic manipulators can be categorized under two different groups: decentralized controllers (Figure 2-1) and coordinated controllers (Figure 2-2). The former category includes control systems where each actuator (or any other part of the manipulator) is independently controlled by its own designated control loop (controller, sensors, etc). Of course, the constraints, desired trajectories, etc, are provided by the operator but each actuator acts parallel to the others, as an independent unit, without realizing the existence of the rest of the mechanism.

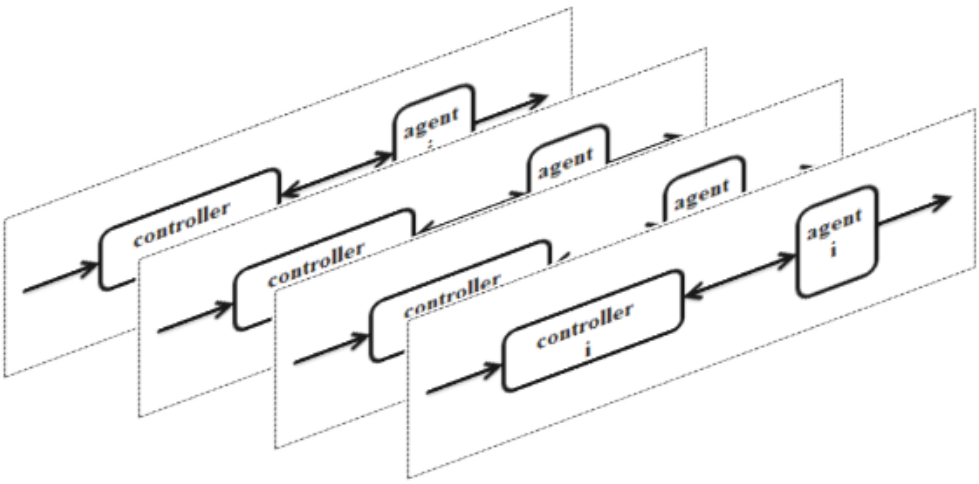

Figure 2-1: Decentralized Control

On the other hand, coordinated control systems try to interconnect the motion of each actuator and relate its behaviour with the behaviour of the rest of the system. This coordination can be a simple correlation of the various agents' trajectories or it can take the form of collective influence on each agent's control signal. Robotic manipulators under coordinate control systems are more "conscious" of their whole structure, a fact which affects their performance positively.

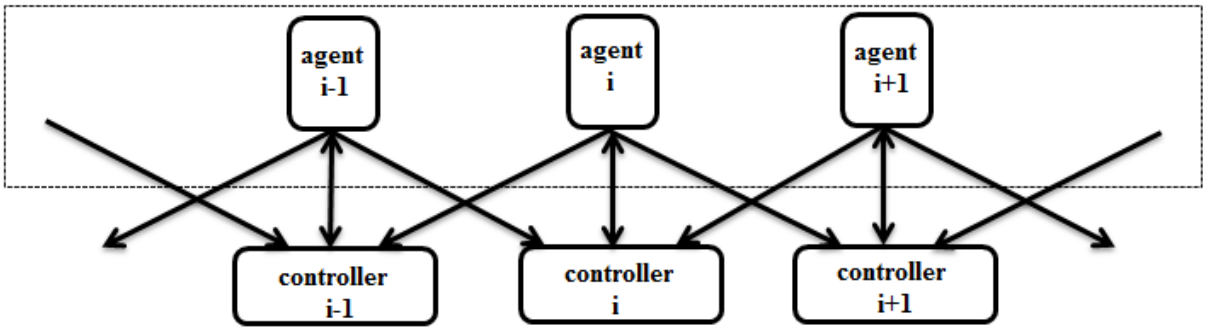

Figure 2-2: Coordinated Control

2.1 Decentralized Control

2.1.1 PID/PD Control

Inarguably, PID control and its other versions (PI and PD) have been the most popular control schemes for robotic manipulators. In fact with more than 90% of industrial applications incorporating some form of PID control, PID controllers dominate the world of feedback control (Astrom & Hagglund, 2001). For a multi-DOF robotic manipulator, PID control is used independently for each motion axis. For a robotic manipulator with n axes, the PID controller for axis i takes the following form:

$$\tau_i = K_{Pi}e_i + K_{Di}\dot{e}_i + K_{Ii}\int_0^t e_i dt \qquad (2\text{-}1)$$

where $e = q_{di} - q_i$ is the tracking error of the axis motion, and K_P, K_I, K_d are constant positive definite gain matrices as shown in Figure 2-3.

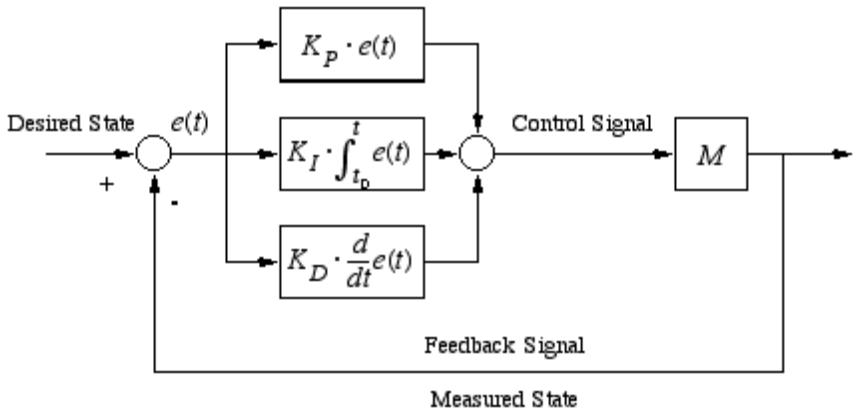

Figure 2-3: Ideal PID Control (Dean, 2002)

The main advantages of the above controller are its simple, linear nature and ease of implementation. In spite of its simplicity, PID control is powerful enough to deal with practical issues such as actuator saturation and integrator windup. Furthermore, a great number of tuning techniques have been developed for PID controllers (Bansal , Sharma, & Shreeraman, 2012). Despite its success, however, PID control is still inappropriate for certain cases due to its inability to adjust to the system's dynamics. The dynamic properties of a robotic manipulator vary with time due to wear, friction, disturbances, damage or simply changes in the conditions of use, i.e. different payloads, tasks or environment, creating a need for controller retuning with each new task.

2.1.2 PD with Desired Gravity Compensation

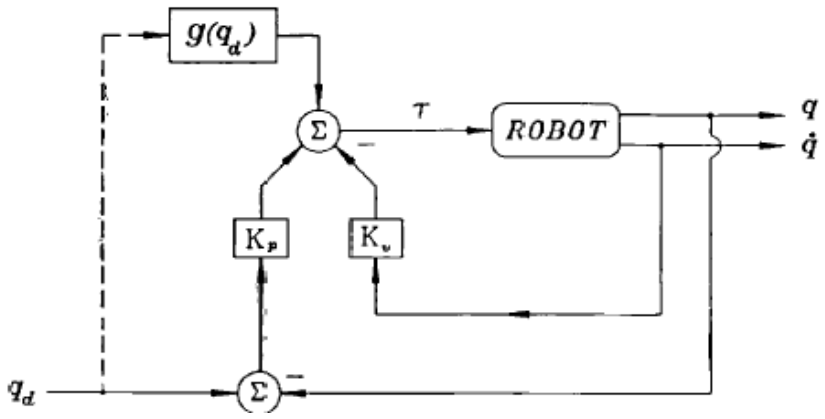

Figure 2-4: PD control with desired gravity compensation (Kelly, 1997)

To improve the stability and robustness of the control system, PD control with desired gravity compensation (PDcg) was developed (Takegaki & Arimoto, 1981). As shown in Figure 2-4, this type of control is based on PID control, specifically PD control, with the addition of a feedforward element called "desired gravity compensation". This new term $g(q_d)$ is nothing but the gravitational torque vector as it is evaluated at the desired joint position and allows the controller to drive the robot in such a way that its joints can be placed at asymptotically desired trajectories regardless of the dynamic systems initial conditions. Mathematically this control law can be expressed as:

$$\tau_i = K_{Pi}e_i + K_{Di}\dot{q}_i + g(q_{di}) \tag{2-2}$$

where e is the tracking error, q_d is the desired joint position.

Unlike PID control, PDgc is globally asymptotically stable. Nonetheless, this type of control features some of the weaknesses of the PID control. Calculating the gravity compensation requires exact knowledge of the dynamic system's parameters and the robotic manipulator's possible payload (Kelly, 1997).

2.1.3 Sliding Mode Control

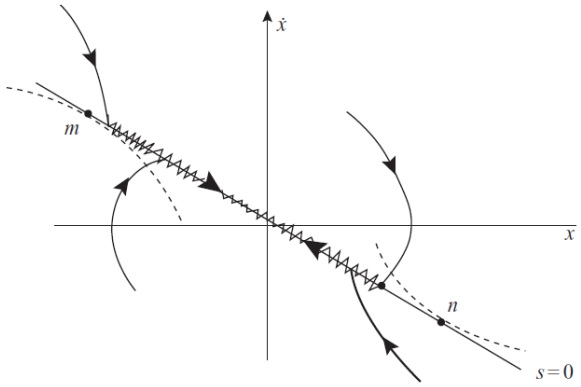

Figure 2-5: Sliding surface (Sabanovic, Fridman, & Spurgeon, 2004)

To counteract the uncertainties of the dynamic system, a sliding mode control (SMC) was proposed (Bartolini, Pisano, Punta, & Usai, 2003). The concept of a sliding mode surface is introduced, based on the tracking error of the system and defined as:

$$s_i = (\frac{d}{dt} + \lambda_i)e_i \tag{2-3}$$

where e is the tracking error for each joint and λ is a positive constant. As defined, the sliding surfaces represent the frequency response of a low-pass first order linear system with the cross over frequency of $f_c = \lambda_i/2\pi$. Therefore, the objective of this control scheme is to bring the controllable variables as close the sliding surface as possible, i.e. $s_i = 0$ (Dombre & Khalil, 2006). As seen in Figure 2-5, chattering can be present in the dynamics of the system. To minimize this phenomenon, a boundary layer is defined to smooth out the control discontinuity in the sliding surface (Slotine & Li, 1991). Including the estimated dynamic parameters of the system as feedforward terms, the control law of the SMC becomes:

$$\tau_i = \widehat{M}(\ddot{q}_{di} + \lambda_i\dot{e}_i) + \widehat{H} + \widehat{M}(\varphi_i|q_i| + \sigma_i|\dot{q}_i| + \gamma_i)\text{sign}(s_i) \tag{2-4}$$

where $\widehat{M}$ and $\widehat{H}$ are estimated dynamic parameters and φ is the boundary layer. This control law produces good results for varying payloads and unknown dynamics and is far more robust than the aforementioned controllers. Conversely, SMC controllers still experience high chatter and energy losses and are generally harder to realize than PID due to their complexity (Slotine & Li, 1991).

2.2 Coordinated Control

2.2.1 Master/Slave Control

The first attempts of coordinated control in robotics focused on master-slave control of two robotic manipulators based on pre-existing bilateral master-slave controls for machine teleoperation (Yokokohji & Yoshikawa, 1994). A master-slave control scheme was proposed in (Arimoto, Miyazaki, & Kawamura, 1987). This control scheme uses the error on the end-effector level between the two manipulators to specify the controller input and can be mathematically expressed as:

$$\begin{cases} V_m = K_{am}^{-1} J_m^T \{F_{mp}(x_1 - x_{l1}) + \mathrm{K}(x_1 - x_2 - x_{l2})\} + F_{mv} q_m \\ V_s = K_{as}^{-1} J_s^T \{-K(x_1 - x_2 - x_{l2})\} + F_{mv} q_m \end{cases} \quad (2\text{-}5)$$

where K_a and F_v are control gains, J and $K$are Jacobian and stiffness matrices respectively and subscripts $m, 1$ and $s, 2$ are used to indicate the master and slave agents. Despite its efficiency, this controller has difficulties in realizing complicated contours due to its use of the end-effector error which is hard to estimate efficiently while in motion.

On the joint level, a different master-slave controller was introduced (Alford & Belyeu, 1984) where a desired path can be defined for the master motion and a relative position is used

for the slave motion. To coordinate the motion, the controller uses the master motion's actual position and via an error prediction matrix defines the change required in the slave motion's command based on a hierarchical system. The coordination of the system is performed by a computer dedicated to that task, which is responsible for receiving the position data from the encoders.

A similar approach was followed by (Rodriguez-Angeles & Nijmeijer, 2001) who used a PD + feedforward type controller. Once again, the trajectory of the master motion is freely defined and the slave motion's control input is based on the master's actual position.

$$\tau_s = M_s(q_s)\hat{\ddot{q}}_m + C_s(q_s, \hat{\dot{q}}_s)\hat{\dot{q}}_m + g_s(q_s) - K_d\hat{\dot{e}}_s - K_p e_s \qquad (2\text{-}6)$$

where M, C and g are the manipulators dynamic parameters, K_d and K_p are controller gains with e being the tracking error, and subscripts m and s refer to master and slave motions.

One can see that the actual position of the master motion is incorporated in the feedforward term of the controller and the error of the slave system is used in the PD segment. In cases where the position and error cannot be measured directly, observers can be used to estimate their values (Rodriguez-Angeles & Nijmeijer, 2001).

The greatest drawback of the aforementioned controllers is their focus on two robot formations and their complete disregard of the coordination of the joint motions with each other. Additionally, the consideration of the slave motion as relative and proportional to the master motion creates difficulties when it comes to more complicate paths and tasks.

2.2.2 Position Domain Control

A different approach was used by (Ouyang, Huang, & Zhang, 2011) who proposed a position domain control (PDC) structure. In PDC, the dynamics describing the slave motion are transformed from time domain to position domain using the position of the master axis as reference. The trajectories and dynamics of the slave axes are therefore expressed as functions of the master axis trajectory which itself is controlled by an equivalent time domain controller. A PD controller in position domain for the slave axes has the following form:

$$\tau_{si} = K_{pi}\big(q_{id}(q_m) - q_i(q_m)\big) + K_{di}\,\big(q'_{id}(q_m) - q'_i(q_m)\big) \qquad (2\text{-}7)$$

where the prime superscript indicates the relative velocity with respect to the master axis position. It should be noted that the actual position of each slave motion, $q_i(q_m)$, is provided by the slave motion's encoder or any other type of sensor used as in the case of time domain.

PDC completely eliminates the error input of the master axis from the control signal. This way it achieves better contour performance than the controllers in time domain. Additionally, the resulting controller after the transformation is quite similar to its time domain counterpart, a fact which guarantees similar dynamic behaviour and stability for both controllers (Ouyang & Dam, 2011).

2.2.3 Cross-Coupled Control

Another family of coordinated control schemes was created in the form of cross-coupled control (CCC) (Koren, 1980). Although each axis of the CCC system has its own feedback loop which is capable of detecting and eliminating the axis' disturbance, the error that occurs on each axis will affect the control input of both axes. As seen in Figure 2-6, this controller cross-couples

the error in its dynamics and coordinate the motion of all axes with each other. The CCC controller uses the tracking error of each axis to derive the contour tracking error for the whole system. Thus, the CCC control loop is based on the task space error for its control input.

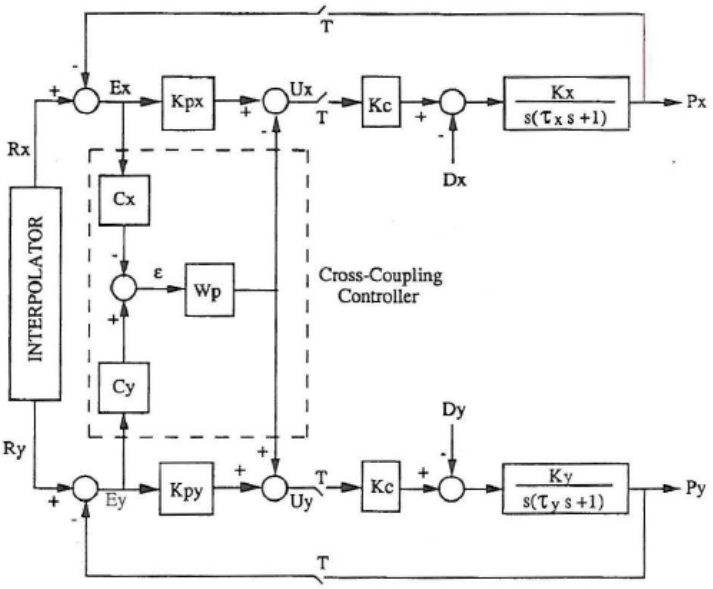

Figure 2-6: Biaxial Cross-coupled Control (Koren, 1980)

The CCC scheme was able to outperform more conventional controllers like PID, but it was not able to produce satisfactory results for non-linear contours. To overcome this drawback, Koren and Lo (1991) introduced a variable-gain CCC control structure. Variable gains were added to the pre-existing controller which had the ability to change in value depending on the desired contour. Mathematically the variable gains were expressed as:

$$\begin{cases} C_x = \sin\theta - \dfrac{e_x}{2R} \\ C_y = \cos\theta + \dfrac{e_y}{2R} \end{cases} \tag{2-8}$$

where e is the tracking erro, and θ and R are functions of the desired contour.This made the CCC controller capable of estimating the contour tracking error for nonlinear contour based on the radius of curvature with significant accuracy (Koren & Lo, 1991).

A position domain approach to the CCC structure (PD-CCC) was introduced by (Dam & Ouyang, 2012). Through a transformation of the system dynamisc from the time domain to position domain, this CCC controller can be defined as:

$$\tau_j(\mathrm{x}) = K_P^j e_j(\mathrm{x}) + K_D^j e'_j(\mathrm{x}) + \mathrm{K}_I^j \int_0^x e_j(s)ds - C_j(K_P^j e_c + K_D^j \dot{e}_c) \qquad (2\text{-}9)$$

where C_j is the contour gain defined as in the time domain CCC case and e_c is the contour error. Similar to PID, position domain the CCC in position domain is defined on a master-slave basis for the axes. The performed simulations showed that PD-CCC had a better contour tracking performance than the time domain CCC, even in cases where the CCC performed better in the tracking errors.

Despite its success, the design of the CCC controller limits it when it comes to the structure to be controlled. The more complicate the kinematics and dynamics of the system to be controlled the harder the estimation of the contour error, a fact which makes the implementation of CCC controllers in serial or parallel manipulators highly difficult or unwise.

2.2.4 Position Synchronization Control

To sidestep the problem of complicated contour estimation and bring the problem back to the joint space level, position synchronization control (PSC) was developed (Sun, 2003). This control concept introduced the notion of synchronization error, defined as the differential position error between the tracking errors of each axes pair, and coupled position error, defined

as a function of both position and synchronization errors. The coordination strategy is to bring each axis tracking error in a stable state and cause the position errors between the axes to converge to zero, i.e.

$$c_1 e_1(t) = c_2 e_2(t) = \cdots = c_n e_n(t) \tag{2-10}$$

Where e is the tracking error and c is the coupling parameter stemming from the constraints for each motion.

Although this controller had an adaptive structure with feedforward, a model-free scheme was later suggested with a structure similar to a proportional-derivative controller (Sun, Shao, & Feng, 2007). The position synchronization control loop has been proven to be quite versatile for various situation besides robotic manipulators and it has been used from CNC machines (Sun & Tong, 2009) to groups of synchronized mini helicopters (Shan, Liu, & Notwotny, 2005). Another great advantage of this control is that it allows for the cross-coupling of multiple axes, and therefore the improvement of contour tracking, without having to deal with the kinematics of the manipulator to be controlled. Since this thesis expands on the concept of position synchronization, a more detailed description of this controller follows in later sections.

2.2.5 Event-Based Control

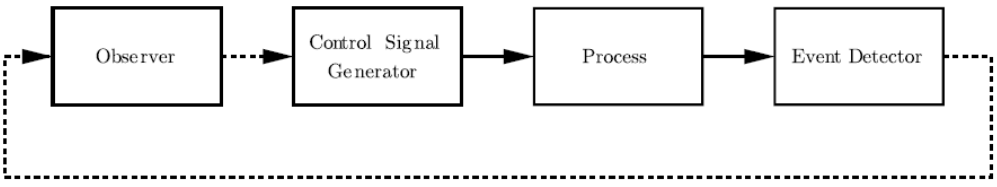

Figure 2-7: Event Based Control Scheme (Astrom, 2007)

Finally, an alternative to periodic sensor sampling for dynamic systems is event based control. Event based systems, instead of sampling the system state on a set time interval, sample the state of the system only when the measurement signal crosses a predefined threshold level. More specifically, during the operation of a system, the even detector generates a signal when an event occurs and sends it to the observer which is responsible for the estimations to be fed on the control signal generator (Astrom, 2007). It is obvious that if the measurement signal does not cross the threshold, the feedback does not materialize for that measurement.

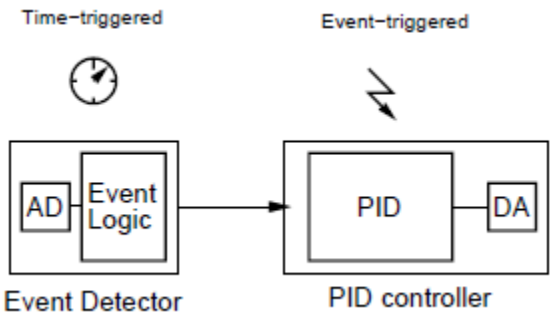

Figure 2-8: Event Based PID Structure (Arzen, 1999)

Due to its nature, event based control reacts faster to the changes of a system's change than conventional periodic sampling control. This occurs because even if a change is detected in periodic sampling, the controller will act after the sampling procedure has finished whereas the event based system acts exactly when the change occurs. For the same reason, event based systems require less CPU power than the periodic sampling system, making them quite attractive for situations where high computing powers are required. An event-based PID structure can be seen in Figure 2-8 (Arzen, 1999).

2.3 Other Control Schemes

Sections 2.1 and 2.2 dealt with the fundamental control concepts for robotic manipulators. Yet more complex concepts of robot control exist which can be applied on the aforementioned controllers in order to enhance their performance in different ways. This section will examine some of multipart control techniques and more specifically will cover the concepts of adaptive control, iterative learning control, repetitive control and event based control.

2.3.1 Adaptive Control

Adaptive control was developed to compensate for the time-varying parameters of the control system, i.e. system dynamics, payload, operating environment. The main idea behind adaptive control is the use of a method to adjust the parameters of a controller online, based on the signals in the system. The sum of adaptive control systems can be categorized as model-reference adaptive control laws (MRAC) or self-tuning controllers (STC), both schematically represented in Figure 2-9 (Slotine & Li, 1991). On the MRAC the model to be controlled (*plan*) is assumed to have a known structure but some of its parameters are unknown. A reference model is employed to indicate the ideal response of the adaptive control system to external signals and an adaptation law is used to adjust the controller based on the difference between the reference model output and the actual output. Clearly, MRAC is centered on the adaptation law and the objective of each adaptation control design effort is to create an adaptation law which would minimize the tracking error to zero and bring the system to stability.

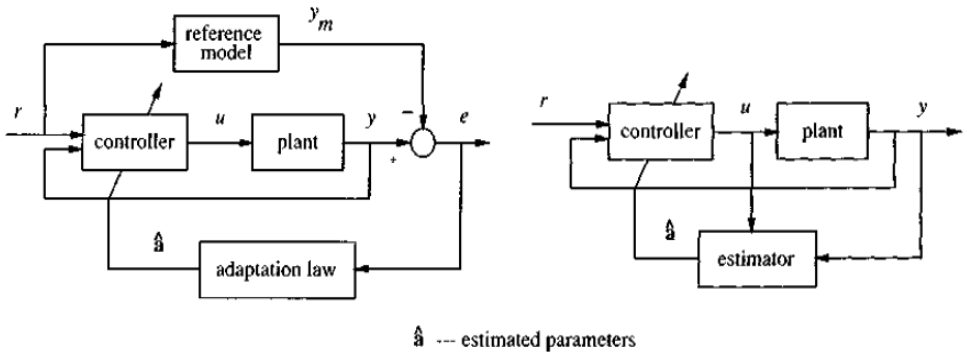

Figure 2-9: Adaptive Control: MARC on left and RTC on the right (Slotine & Li, 1991)

Self-tuning controllers, on the other hand, employ a simpler architecture. The estimator of the system produces a set of estimated parameters based on the past plant inputs and outputs which the controller uses in order to produce the current control input for the plant. Both types of adaptive control feature come with various pros and cons, however, MARC systems have been more successful in the control of robotic manipulators, mostly due to their usually guaranteed stability and tracking error convergence (Slotine & Li, 1991).

Adaptive control laws have been flexible enough to be employed in a various control structures, from PID loops (Kuc & Han, 2000) to SMC loops (Zhao, Li, Gao, & Zhu, 2008). Additionally, the time domain synchronization control was initially proposed as an adaptive control system (Sun, 2003).

2.3.2 Iterative Learning Control

While adaptive control laws try to produce more flexible and versatile robot behaviours, the iterative learning control (ILC) is to refine the performance of the robotic manipulator for a specific task through repetition. ILC stems from the fact that the majority of industrial systems are to repeatedly perform the same or similar tasks.

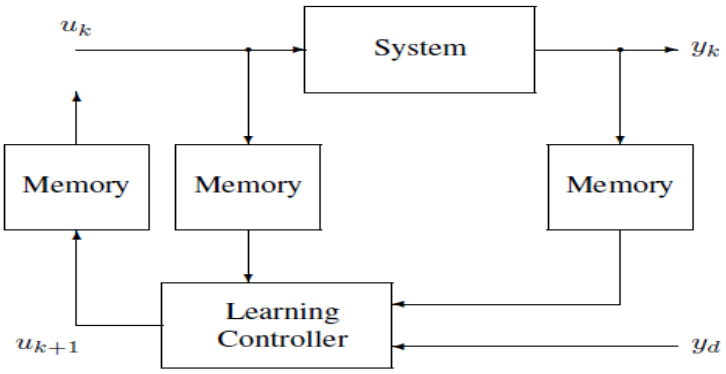

Figure 2-10: Iterative Learning Control (Moore, Chen, & Ahn, 2006)

During the process, the input and output signals of the system are being saved in the memory of the system. The error between the two signals is used to construct the input signal for the new cycle. In contrast to conventional controllers which try to decrease the error of the system as time increases, ILC decreases the system error as the number repetition increases. A basic ILC algorithm can be expressed as:

$$u_{k+1}(\mathrm{t}) = u_k(\mathrm{t}) + \gamma \frac{d}{dt} e_k(\mathrm{t}) \tag{2-11}$$

where $u(t)$ is the system input and $e(t)$ is the error of the system output. The above equation also shows the main advantage of ILC. The initial conditions of the system are reset with each cycle, but the new input signal is created based on the previous cycle's data. This allows for a "non-causal" processing on the errors and therefore ILC can anticipate and proactively respond to repeated disturbances (Moore, Chen, & Ahn, 2006). Due to its nature as a control algorithm and not a singular controller, ILC has been easily implemented on conventional controllers for the control of robotic manipulators used for various processes (Bristow, Tharayil, & Alleyne, 2006).

2.3.3 Repetitive Control

Quite similar to the ILC, repetitive control (RC) also uses repetition of the process to achieve the stabilization of the system's performance. Proposed by (Inoue, Nakano, Matsumoto, & Baba, 1981), RC can be mathematically expressed as:

$$u_{i+1}(\mathrm{k}) = u_i(\mathrm{k}) + \lambda e_i(k) \tag{2-12}$$

where $u(t)$ is the system input and $e(t)$ is the error of the system output. In contrast to ILC, RC is used in continuous, periodical processes for tracking or rejecting exogenous signals. For this reason, the RC system does not maintain the original initial conditions as the new cycle as the ILC does. Instead, the final values of the previous cycle are set as the new initial conditions for the new iteration (Wang, Gao, & Doyle, 2009).

2.4 Tracking and Contour Error

Two important parameters of the dynamic response, and therefore the industrial performance of an industrial manipulator are the axial tracking of each motor and the contouring performance of the end effector. In control theory, these parameters take the form of the tracking error and contour error, respectively. Both errors result from phenomena such as backlash, friction, difference in actuator dynamics, disturbance loads, the contour shape itself, etc. (Koren & Lo, 1991).

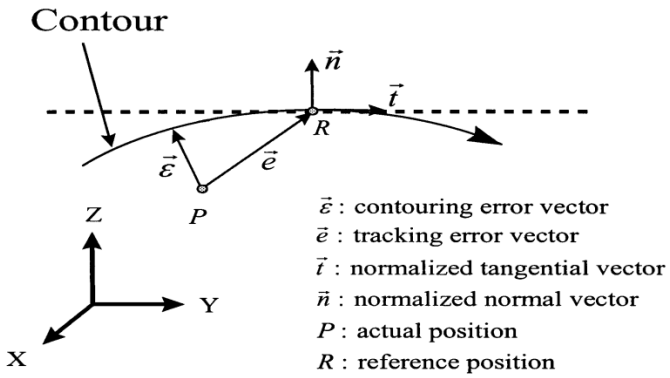

Figure 2-11: Tracking and Contour Error (Yeh, 2003)

The tracking error is defined as the difference between the desired trajectory of an axis of the robotic manipulator and the actual trajectory followed. In other words it is the measurement of the performance of the individual actuator and can be numerically defined as:

$$e = r - a \tag{2-13}$$

where r is the desired postion and a the actual position.

Alternatively, the contour error is the actual difference in distance between the desired path of the end effector and the actual path resulting from the actuated system (Ramesh, 2005). Obviously, for a robotic manipulator the contour error refers to the global coordinates of the en effector. Additionally, due to the nature of the contour error, different methods of must be employed for its calculation or estimation, depending on the path.

2.4.1 Contour Error Calculation for Linear Contours

For linear contours, the contour error can be calculated as (Yeh, 2003):

$$e_c = -e_x \sin\theta + e_y \cos\theta \tag{2-14}$$

Where e_x and e_y are the tracking errors for the actual point P in the x-axis and y-axis respectively, and θ is the angle formed by the desired point P^*, in the desired linear contour and the x-axis, as shown in Figure 2-12

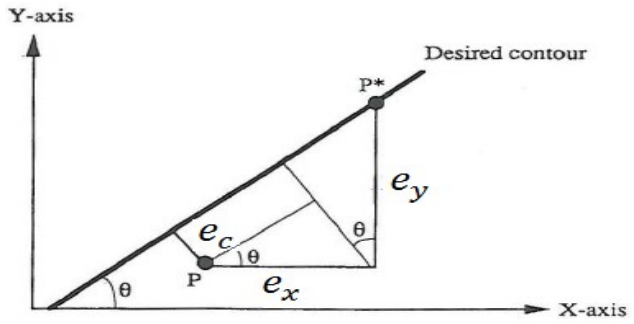

Figure 2-12: Linear Contour Error (Yeh, 2003)

2.4.2 Contour Error Calculation for Circular Contours

For a circular contour, the contour error can be easily calculated based on the Cartesian equation of a circle as (Yeh, 2003):

$$e_c = \sqrt{(x_P - x_o)^2 + (y_P - y_o)^2} - R \qquad (2\text{-}15)$$

where (x_o, y_o) are the coordinates of the circular contour, R is the radius of the circle, and (x_P, y_P) are the coordinates of the actual point P.

2.4.3 Free-Form Contour Error Estimation

In the case of a contour shape that cannot be classified as one of the above, the contour error cannot be easily calculated, in real-time or otherwise. In this report, the contour error estimation algorithm proposed by Cheng and Lee (2007). Although this algorithm is designed for real-time error estimation, it is also practical for off-line calculations.

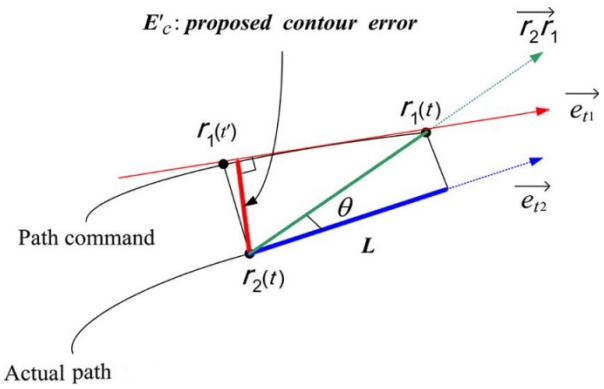

Figure 2-13: Free-Form Contour Error Estimation (Cheng, 2007)

Consider two position vectors $r_1(t)$ and $r_2(t)$ indicating the desired and actual positions of the end effector for each time instant$t$. The unit tangent vectors at points $r_1(t_k)$ and $r_2(t_k)$ are given as:

$$\overrightarrow{e_1(t_k)} = \frac{d\overrightarrow{r_1(t_k)}}{ds_1(t_k)} \tag{2-16}$$

$$\overrightarrow{e_2(t_k)} = \frac{d\overrightarrow{r_2(t_k)}}{ds_2(t_k)} \tag{2-17}$$

where $ds_i = \sqrt{dr_{x_i}^2 + dr_{y_i}^2}$, $i = 1,2$ is the magnitude of the position vectors. Hence at time instant t_k the distance between the two position vectors is given as:

$$\overrightarrow{r_2r_1(t_k)} = \overrightarrow{r_2} - \overrightarrow{r_1} \tag{2-18}$$

and the length of the projection of the distance (i.e. tracking error) can be calculated by:

$$L = \frac{\overrightarrow{r_2r_1(t_k)} \cdot \overrightarrow{e_2(t_k)}}{\left\|\overrightarrow{e_2(t_k)}\right\|} \tag{2-19}$$

Now assume that a point $r_1(t_k{}')$ exists on the desired contour and the distance between this point and $r_2(t_k)$ is the real contour error. Assuming that the end effector has a desired velocity $\overrightarrow{V}$ the time required to cover the distance L can be found to be:

$$\Delta t = L/\left|\vec{V}\right| \tag{2-20}$$

which is also the required time to travel from point $r_1{}'(t_k)$ and $r_1(t_k)$. Hence the previously unknown point $r_1{}'(t_k)$ can be found to be:

$$r_1(t_k{}') \;=\; r_1(t_k - \Delta t) \approx\; r_1(t_k) - V \cdot \Delta t \tag{2-21}$$

Therefore, the contour can be calculate in a similar manner to the linear contour,

$$e_c = \; -e_x \sin\phi + e_y \cos\phi \tag{2-22}$$

Where e_x and e_y are the tracking errors in the Cartesian axes and ϕ is the reference angle between the vector $\overrightarrow{r_1(t_k{}')\, r_1(t_k)}$ and the horizontal x-axis, calculated as:

$$\phi = \; \tan^{-1}\left(\frac{r_{1y}(t_k{}') \; - \; r_{1y}(t_k)}{r_{1x}(t_k{}') \; - \; r_{1x}(t_k)}\right) \tag{2-23}$$

Concerning velocity V, if the end effector maintains a constant velocity $V_1(t_k{}')$ is equal to $V_1(t_k)$. Usually through, the end effector of a robotic manipulator experiences significant changes in its velocity. Thus, an estimation can be used where the velocity is defined as the average of the desired ($V_1(t_k)$) and actual velocities ($V_2(t_k)$) for each time instantt_k. This is, in fact, the main reason that the resulting contour error value is considered estimation and not an exact calculation (Cheng, 2007).

2.5 Remarks

All of the above control schemes try to improve the contouring performance and dynamic response of a robotic manipulator based on different methodologies. Nevertheless, each control strategy features disadvantages which tend to overshadow its benefits, as is the case for the CCC controllers which are really hard to implement on other manipulators besides CNC. Yet, PSC

controllers have proven to quite adaptable when it comes to implementation without losing their effectiveness of characteristics (Sun, 2011) , and PDC controllers have shown superior contour tracking performance (Ouyang, Pano, & Dam, 2012).

To further progress the notion of position domain control, a new control law based on the position domain and position synchronization concepts will be introduced in the next chapter. The main incentive of this research it to combine the main advantages of PDC as a master/slave, event-driven control system and the PSC as a coordinated control scheme in order to improve the contour tracking performance of the system.

Chapter 3: Position Domain Synchronization Control

In this section the main principles of position domain control and position synchronization control are explored. The dynamic model of a robotic manipulator is transformed from time to position domain. Furthermore, the position domain synchronization controller is developed and some remarks are made. A stability analysis is performed to study the stability of the system.

3.1 Dynamic Model

The dynamic model of a robotic manipulator relates the torques and forces applied to the manipulator's actuators to the joint positions, velocities and accelerations (Dombre & Khalil, 2006). On mathematical terms, the dynamic model takes the following form:

$$M(q)\ddot{q}(t) + C(q,\dot{q})\dot{q}(t) + G(q) + F(t,q,\dot{q}) = \tau(t) \qquad (3\text{-}1)$$

where:

- $q(t), \dot{q}(t)$ and $\ddot{q}(t)$ are the joint position, velocity and acceleration vectors, defined as functions of time.
- $M(q)$ is the symmetric and positive-definite inertia matrix
- $C(q, \dot{q})$ is the vector of coriolis and centrifugal forces
- $G(q)$ is the vector of gravity terms
- $F(t, q, \dot{q})$ is the vector of friction forces, and
- $\tau(t)$ is the vector of joint torques/forces.

3.2 Position Synchronization Control

3.2.1 The Synchronization Concept

Assume a multiagent system with *n* agents given to perform a task that requires the coordination of each agent's motion, i.e., an n-DOF robotic manipulator following a defined contour. The goal of synchronization control is to regulate and synchronize the motion of all agents so that a certain kinematics relationship is maintained for the agents as is required by the coordination task. Not unlike a sliding surface, the regulation of the agents to maintain a kinematic relationship can be understood as guiding the agents along the boundary of multidimensional a compact set. A time-varying desired shape in such a compact set can be introduced as $S(\varrho, t)$ with ϱ being a state vector and t the time. Then the boundary of the shape can be parameterized by a curve, denoted as $\partial S(\varrho, t) = 0$.

Therefore, the synchronization of a system of multiple entities requires two main tasks to be performed. The first one is the classic control goal of bringing the state error of each agent as close to zero as possible as time increases, i.e., $e_i \to 0$ as $t \to \infty$. The second task requires

maintaining each agent on the desired curve so that $\partial S(\varrho, t) = 0$. Both tasks are equally important and should be achieved simultaneously in order to achieve system synchronization.

Although the constraints for the second synchronization task can take various forms, depending on the system's nature and purpose, the following mathematical expression can be used without loss of generality:

$$\partial S(x_i, t) = 0: \{x_i(t) = A_i(t)C(t) + B_i(t)\}\,, i = 1,2,..,n \qquad (3\text{-}2)$$

where $x_i(t)$ is the state of the i[th] agent, $A_i(t)$ is a constraint matrix based on the desired boundary, which on itself depends on the agent's characteristics ; $C(t)$ is a common vector of characteristics that are applicable on the whole system but not on individual agents , and $B_i(t)$ is an offset of the i[th] agent.

The above equation indicates that all agents of the system can be related through the common vector $C(t)$ which requires a linear mapping from $x_i(t)$ to $C(t)$ denoted as $\{x_i(t)\} \longmapsto C(t)$. For the existence of this unique linear mapping to exist, the constraint matrix $A_i(t)$ must be invertible. It should be noted that because $A_i(t)$ is based on the desired boundary, it is mainly determined by the topology of the given formation task.

Assuming the existence of the inverse of matrix $A_i(t)$, it follows that:

$$C(t) = A_i^{-1}(t)\big(x_i(t) - B_i(t)\big) \qquad (3\text{-}3)$$

However, $C(t)$ is a common vector for every agent. Hence for a system of *n* agents:

$$A_1^{-1}(t)\big(x_1(t) - B_1(t)\big) = \cdots = A_n^{-1}(t)\big(x_n(t) - B_n(t)\big) = C(t) \qquad (3\text{-}4)$$

Eq. (3-4) is expressed in terms of the actual state $x_i(t)$ of the i[th] agent, but it can also be expressed in terms of the desired state $x_i^d(t)$,

$$A_1^{-1}(t)\left(x_1^d(t) - B_1(t)\right) = \cdots = A_n^{-1}(t)\left(x_n^d(t) - B_n(t)\right) = C(t) \quad (3\text{-}5)$$

Subtracting Eq (3-4) from Eq (3-5),

$$A_1^{-1}(t)e_1(\mathrm{t}) = A_2^{-1}(t)e_2(\mathrm{t}) = \cdots = A_n^{-1}(t)e_n(\mathrm{t}) \quad (3\text{-}6)$$

where e is the tracking error.

Lastly, $c_i(t) = A_i^{-1}(t)$ can be defined as the coupling parameter of the i[th] agent, leading to:

$$c_1(t)e_1(\mathrm{t}) = c_2(t)e_2(\mathrm{t}) = \cdots = c_n(t)e_n(\mathrm{t}) \quad (3\text{-}7)$$

The above equation is ultimately the synchronization control goal which all position error must satisfy in order for system to meet the coordination requirements (Sun, 2011).

It should be noted that the coupling parameters do not always take the form of mathematical expressions, but they can simply be constants or identity matrices. Additionally, depending on the nature and purpose of the system, all the agents might have the same coupling parameter. In that case, the synchronization goal becomes:

$$e_1(\mathrm{t}) = e_2(\mathrm{t}) = \cdots = e_n(\mathrm{t}) \quad (3\text{-}8)$$

Obviously, such is the case for a serial robot described by Eq. (3-1).

3.2.2 Synchronization Errors

With its introduction in control terms, a measure of synchronicity should be also introduced and employed in the control of the system. This measure, taking the form of synchronization error should satisfy the following criteria:

1. The synchronization error should be defined according to the synchronization concept (i.e. Eq (3-8))

2. The differential state error concepts in cross-coupling controllers can be used for the definition of synchronization error.
3. The synchronization error should be independent of the agent population.
4. The definition of the synchronization error should also include factors that produce constraints to the states in synchronization.

With the above consideration, the synchronization error of a system can be defined as:

$$\varepsilon = \begin{bmatrix} c_1(t)e_1(\mathrm{t}) - c_2(t)e_2(\mathrm{t}) \\ c_2(t)e_2(\mathrm{t}) - c_3(t)e_3(\mathrm{t}) \\ \vdots \\ c_{n-1}(t)e_{n-1}(\mathrm{t}) - c_n(t)e_n(\mathrm{t}) \end{bmatrix} \tag{3-9}$$

$$\varepsilon = \begin{bmatrix} c_1(t) & -c_2(t) & \dots & 0 \\ 0 & c_2(t) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -c_1(t) & 0 & \dots & c_n(t) \end{bmatrix} \begin{Bmatrix} e_1(\mathrm{t}) \\ e_1(\mathrm{t}) \\ \vdots \\ e_n(\mathrm{t}) \end{Bmatrix} = T(t)e(t) \tag{3-10}$$

Furthermore, a coupled error is introduced in order to make the tracking error e and synchronization error ε converge to zero, defined as:

$$e^* = e(t) + B \cdot \varepsilon(t) = (\mathrm{I} + BT)e(t) \tag{3-11}$$

where B is a positive definite diagonal gain matrix and T is called the synchronization matrix. With the existence of $(\mathrm{I} + \mathrm{BT})^{-1}$, the convergence of the synchronization error to zero implies that the tracking error also converges to zero and vice versa. Therefore, the control gain B should be properly chosen so that the $(\mathrm{I} + \mathrm{BT})$ is positive definite with full rank.

3.2.3 Model-free Position Synchronization Control

With the theoretical background established and the required entities developed, a PD-type synchronized controller was proposed by (Sun, Shao, & Feng, 2007) as:

$$\tau = K_p e^*(t) + K_d \dot{e^*}(t) + \{(\mathrm{I} + BT)^{-1}\}\dot{e} \quad (3\text{-}12)$$

where K_p, K_d and B are diagonal positive gain matrices. It should be noted that the last term on the right-hand side was introduced for stability purposes and does not have significant contribution on the control input.

3.3 Position Domain Control

In position domain control, an n-DOF system is discretized between master and slaves agents. The motion of the master agent is sampled equidistantly and used as an independent reference, in the same manner as time is used in time domain control. The motions of the slave agents are defined as functions of master motion and the contouring requirements of the system. The formulation of these functions necessitates the development of the system's dynamic model on the position domain through a one-to-one transformation from the time domain as well as its representation as a master-slave system. The great advantage of position domain control is the elimination of the tracking error from the reference model, which minimizes the contribution of the master agent to the contour error of the system. Hence, in order for the position domain model to be effectively utilized, the master motion should be measured with as high precision as possible.

3.3.1 Relative Derivative and Position Domain Mapping

The first step for transforming a dynamic system to position domain is to develop a relation which relates the position domain to time domain. This is done by introducing the relative derivate of the ith slave agent's motion (q_i) with respect to the master agent's motion (q_m):

$$q_i' = \frac{dq_i}{dq_m} = \frac{\dot{q}_i}{\dot{q}_m} \quad (3\text{-}13)$$

From Eq. (3-13) it can be easily understood that q_i' is the speed ratio between the slave and master agent's motions and it describes a synchronized motional relationship between the two motions. This relative derivative is called relative position velocity of agent i with respect to the master agent.

In the same manner, the relative position acceleration can be defined as the second relative derivative,

$$q_i'' = \frac{dq'_i}{dq_m} \tag{3-14}$$

From Eq. (3-14), the velocity of axis *i* can be defined as:

$$\dot{q}_i = \dot{q}_m q_i' \tag{3-15}$$

Hence, Eq. (3-15) can be expressed as

$$\ddot{q}_i = q_i''(\dot{q}_m)^2 + \ddot{q}_m q_i' \tag{3-16}$$

The above equations show the relationship between absolute and relative motions. Eq. (3-15) relates the absolute velocity in the time domain with the relative derivative in the position domain, whereas Eq. (3-16) relates the absolute acceleration with the relative acceleration. Both equations are used to transform the dynamic model from time domain to position domain.

3.3.2 Dynamic Model in the Position Domain

The dynamic model of Eq (3-1) can be de expressed in a master-slave structure as:

$$\begin{bmatrix} m_{mm} & M_{ms} \\ M_{sm} & M_{ss} \end{bmatrix} \begin{bmatrix} \ddot{q}_m \\ \ddot{q}_s \end{bmatrix} + \begin{bmatrix} c_{mm} & C_{ms} \\ C_{sm} & C_{ss} \end{bmatrix} \begin{bmatrix} \dot{q}_m \\ \dot{q}_s \end{bmatrix} + \begin{bmatrix} G_m \\ G_s \end{bmatrix} + \begin{bmatrix} F_m \\ F_s \end{bmatrix} = \begin{bmatrix} \tau_m \\ \tau_s \end{bmatrix} \tag{3-17}$$

The dynamic model in Eq. (3-17) is a reinterpretation of Eq. (3-1) (3-17)as a master slave system in time domain. Subscript *m* refers to the master agent/motion and subscript *s* refers to the slave agents/motions. Therefore q_m is the reference position and is used as the reference for tracking a defined contour. Consequently, the dynamic model for the slave agents can be rewritten in position domain as a function of the reference q_m through a transformation from time domain (t) to position domain (q_m).

Substituting Eqs (3-15) and (3-16) to Eq. (3-17) for the slave motions, a dynamic model for the position domain slave motions is derived in the following form:

$$\dot{q}_m^2 M_{ss} q_s''(q_m) + \left(\ddot{q}_m M_{ss} + \dot{q}_m C_{ss}\right) q_s'(q_m) + \ddot{q}_m M_{sm} + \dot{q}_m M_{sm} + G_s + F_s = \tau_s(q_m) \quad (3\text{-}18)$$

Remark 1: The above equation represents the dynamic relationship between the master motion, indicated by subscript *m*, and the slave motions, indicated by subscript s, in the position domain via the transformation described by Eqs (3-15) and (3-16). Clearly the nonlinearity of the time domain model in Eq. (3-17) is maintained in the potion domain.

Remark 2: From the above dynamics equation it can be deduced that the position of the master motion has greater influence on the system's control than the master motion tracking error. Therefore, in order to achieve accurate contour performance, high precision measurement of the master motion's position is required. Nonetheless, a high tracking precision of the master motion is not required for the position domain control since the master motion's error is not to be included in the position domain controller. Therefore, a low cost actuator can be used for the master motion.

Remark 3: It is understood that the position domain control structure requires the master motion control to operate in the time domain. In that sense then PDC is the combination of two different controllers running in sequence as shown in Figure 3-1.

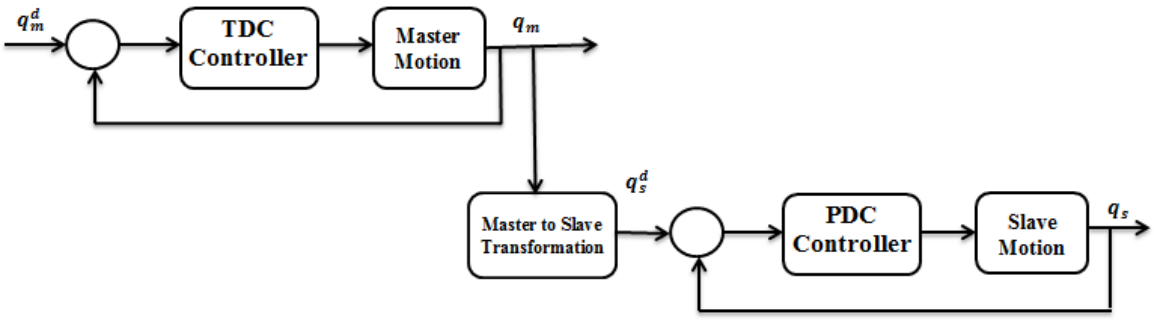

Figure 3-1: Ideal Position Domain Control

3.3.3 Position Domain PID Control law

Following the position domain mapping described above, the PID control law described in Eq. (2-1) can be used to control the slave motions of Eq. (3-18) .This position domain PID control law is expressed as:

$$\tau_i = K_{Pi} e_i(q_m) + K_{Di} e_i'(q_m) + K_{Ii} \int_0^{q_m} e_i(s) ds \qquad (3\text{-}19)$$

The stability of this controller was proven by (Ouyang, Pano, & Dam, 2012).

3.3.4 Position Domain Synchronization Control law

In order to take advantage of the synchronization control properties and the improved contour tracking performance of position domain control, a new control law is introduced for the slave motions of a multi-DOF robotic manipulator described in Eq. (3-18). First, the new error concepts have to be defined.

The tracking position error and relative derivative error in the position domain are defined as:

$$e_s(q_m) = \begin{bmatrix} e_{s,1} \\ \vdots \\ e_{s,n-1} \end{bmatrix} = \begin{bmatrix} q_{2d}(q_m) - q_2(q_m) \\ \vdots \\ q_{nd}(q_m) - q_n(q_m) \end{bmatrix}$$
$$e'_s(q_m) = \begin{bmatrix} e'_{s,1} \\ \vdots \\ e'_{s,n-1} \end{bmatrix} = \begin{bmatrix} q'_{2d}(q_m) - q'_2(q_m) \\ \vdots \\ q'_{nd}(q_m) - q'_n(q_m) \end{bmatrix} \quad (3\text{-}20)$$

Similarly, the synchronization error is expressed as:

$$\varepsilon_s(q_m) = \begin{bmatrix} e_{s,1} - e_{s,2} \\ e_{s,2} - e_{s,3} \\ \vdots \\ e_{s,n-1} - e_{s,1} \end{bmatrix} = \begin{bmatrix} 1 & -1 & \cdots & 0 \\ 0 & 1 & -1 & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ -1 & \cdots & 0 & 1 \end{bmatrix} e_s(q_m) \quad (3\text{-}21)$$
$$\varepsilon_s(q_m) = T e_s(q_m)$$

And the coupled error is:

$$\begin{cases} e_s^*(q_m) = e_s(q_m) + B\varepsilon_s(q_m) \\ e_s^{*\prime}(q_m) = e'_s(q_m) + B\varepsilon'_s(q_m) \end{cases} \quad (3\text{-}22)$$

It should be noted that the coupling terms have been reduced to unit values. This happened because the agents of the robotic manipulator are just the joints and are not restricted by anything more than just the predefined path their have to follow and their kinematics, which are expressed by parameter $C(t)$ in Eq. (3-7) which is not present in the synchronization control goal.

Having defined the errors in the position domain, the position domain synchronization control law can be defined as:

$$\tau_s(q_m) = K_{ps} e_s^*(q_m) + K_{ds} e_s^{*\prime}(q_m) \quad (3\text{-}23)$$

with K_{ps}, K_{ds} being constant diagonal gain matrices. The third term in Eq. (3-12) is not included in the new controller. This is because the third term was introduced to stabilize the system and

due to the presence of $(\mathrm{I} + BT)^{-1}$, which of a small vale, it does not have a significant contribution in the control input.

Taking into account the error definitions above, the control law in Eq. (3-23) can also be expressed in the following form:

$$\tau_s(q_m) = \overline{K}_{ps} e_s(q_m) + \overline{K}_{ds} e_s'(q_m) \tag{3-24}$$

where $\overline{K}_{ps} = K_{ps}(I + BT)$ and $\overline{K}_{ds} = K_{ds}(I + BT)$.

Remarks 4: The position domain control law in Eqs (3-23) and (3-24) is similar to its counterpart in time domain shown in Eq. (3-12) with three essential differences. Firstly, the two control laws function in different domains. Furthermore, due to the different natures of the derivative coupled errors for each system ($\dot{e}_i(t)$ and $e_s'(q_m)$), the derivative gains of each controller have different physical meanings. Lastly, the controller in Eq (3-24) lacks the third term of the right-hand side of Eq. (3-12). The inverse term is present for the stability of the system and does not affect its performance. As it will be seen in later sections, the new controller does not require that feature for the stability of the system. Hence the third term was removed from the controller altogether.

The new control law can be expressed in time domain terms by simply substituting Eqs (3-15) and (3-16) into (3-24):

$$\tau_s(q_m) = \overline{K}_{ps} e_s(q_m) + \frac{\overline{K}_{ds}}{\dot{q}_m} \dot{e}_s(t) \tag{3-25}$$

where $\overline{K}_{ps} = K_{ps}(\mathcal{I} + BT)$ and $\overline{K}_{ds} = K_{ds}(\mathcal{I} + BT)$.

Remark 5: Eq. (3-25) shows that the derivative term of the position domain synchronization controller is variable in nature with its value depending on the value of the master motions speed.

If the master motion has a constant speed and is sampled equidistantly, the position domain controller can be viewed as a constant gain controller similar to the time domain PSC controller with similar stability properties.

Remark 6: It should be noted that the PDSC controller is used for the synchronization of the slave motions and the master motion does not participate. This is because the master to slave synchronization is already achieved by the position domain structure itself. Consequently, the synchronization error for the master motion is eliminated and the master motion controller it reduced to a simple PD controller as visualized in Figure 3-2.

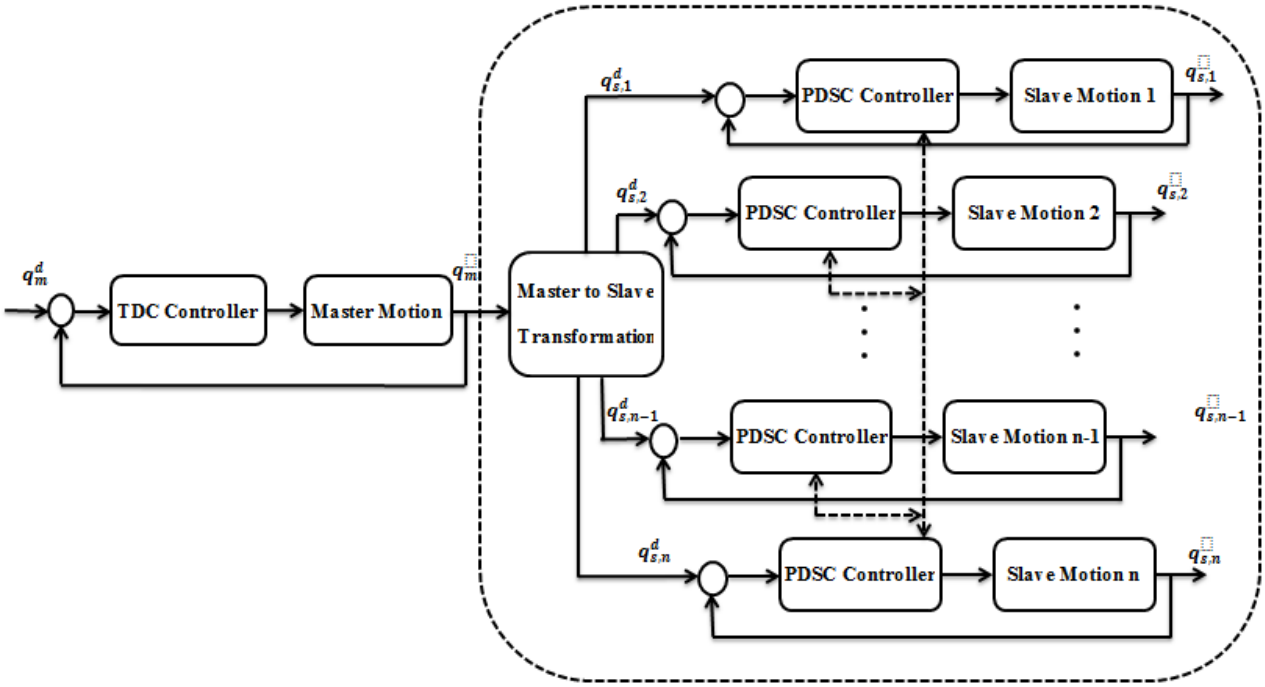

Figure 3-2: Position Domain synchronization control Schematic

Remark 7: Due the elimination of time from the control, Eqs (3-18) and (3-23) constitute an event-driven system.

3.3.5 Properties and Assumptions of the Dynamic Model

For the new control law, a list of properties of a rigid robotic manipulator described by Eqs (3-1) and (3-17) are used in the stability analysis (Kelly, 1997). These properties are described as follows:

P1: The inertia matrix $M(q)$ is symmetric and positive definite. M_{ss} can be also be proven to be symmetric and positive definite without great difficulty.

P2: The matrix $\dot{M}(q) - 2C(q,\dot{q})$ is skew symmetric and consequently, $\dot{M}_{ss}(q) - 2C_{ss}(q,\dot{q})$ is skew symmetric as well.

P3: The inertia and centrifugal-coriolis matrices must satisfy the following:

$$\begin{cases} \dot{M}(q) = C(q,\dot{q}) + C^T(q,\dot{q}) \\ \dot{M}_{ss}(q) = C_{ss}(q,\dot{q}) + {C_{ss}}^T(q,\dot{q}) \end{cases} \tag{3-26}$$

P4: $M(q), C(q,\dot{q}), G(q)$ and $F(t,q,\dot{q})$ are all bounded. From P4 it can also be deduced that:

i. M_{s1} is bounded with $\|M_{s1}\| \leq m_{s1}$
ii. C_{s1} is bounded with $\|C_{s1}\| \leq c_{s1}$
iii. G_s is bounded with $\|G_s\| \leq g_s$
iv. F_s is bounded with $\|F_s\| \leq f_s$ for a normal operation condition

Additionally, the following notation is introduced: $\lambda_m(M)$ and $\lambda_M(M)$ represent the smallest and largest eigenvalues of a positive definite matrix *M*. If a square *M* matrix is positive definite, then it is denoted as $M \succ 0$. If a square matrix $M - N$ is positive definite, then it is denoted as $M - N \succ 0$.

For positive definite matrices, the following properties will be used (Arendt & Schleuch, 2009):

P5: If $M \succ 0$, then $M^{-1} \succ 0$.

P6: If $M \geq N \succ 0$, then $M^{-1} \geq N^{-1} \succ 0$.

P7: If $M \succ 0$ and $\lambda > 0$ is a real number, then$\lambda M \succ 0$.

P8: If $M \succ 0$ and $N \succ 0$, then $M + N \succ 0$, $MNM \succ 0$, and $NMN \succ 0$

Finally, the following reasonable assumptions are used.

A1: The master motion q_m is a monotonically increasing function with the second order derivative for $q_m \in [q_{ms}, q_{mn}]$.

A2: The velocity $\dot{q}_m$ and acceleration $\ddot{q}_m$of the master motion are bounded in the desired trajectory region.

A3: The desired contour trajectory $q_{sd}(q_m)$ is second order continuous for $q_m \in [q_{ms}, q_{mn}]$.

3.4 Stability Analysis

3.4.1 Theorem

Theorem: For a rigid robotic manipulator described in the position domain by Eq. (3-18), if the position domain synchronization control law in Eqs (3-23) and (3-24) is applied to control a contour tracking of the robotic manipulator, and the following conditions in Eq. (3-27) are satisfied, then the controlled robotic manipulator is globally asymptotically stable for contour tracking.

$$\begin{cases} K \succ 0 \\ \overline{K}_{ps} \succ 0, \quad \overline{K}_{ds} \succ 0 \\ K + \dot{q}_m C_{ss}^T - \ddot{q}_m M_{ss} > 0 \\ \lambda_m(\overline{K}_{ps}) > \dot{q}_m^2 \lambda_M(M_{ss}) > 0 \\ \lambda_m(\overline{K}_{ps}) > \frac{1}{2}\lambda_M(K + \dot{q}_m C_{ss}^T - \ddot{q}_m M_{ss}) \\ \lambda_m(\overline{K}_{ds} + (\ddot{q}_m - \dot{q}_m^2) M_{ss}) > \frac{1}{2}\lambda_M(K + \dot{q}_m C_{ss}^T - \ddot{q}_m M_{ss}) \end{cases} \tag{3-27}$$

where K is a user defined constant matrix, $\overline{K}_{ps} = K_{ps}(\mathcal{J} + BT)$ and $\overline{K}_{ds} = K_{ds}(\mathcal{J} + BT)$.

3.4.2 Control Gains and Synchronization Matrix

The synchronization matrix can be proven to be positive definite as follows:

$$\begin{aligned} z^T T z &= [a_1\ a_2\ \dots\ a_n] \begin{bmatrix} 1 & -1 & \cdots & 0 \\ 0 & 1 & -1 & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ -1 & \cdots & 0 & 1 \end{bmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} \\ &= a_1(a_1 - a_n) + \dots a_n(a_{n-1} - a_1) \\ &= \frac{1}{2}[(a_1 - a_2)^2 + \cdots (a_1 - a_n)^2] > 0 \end{aligned} \tag{3-28}$$

where $a_1, a_2, \dots, a_n$ are positive nonzero elements.

Therefore, T is positive definite. By also choosing K_{ps}, K_{ds} and B to be positive definite, $\overline{K}_{ps}$ and $\overline{K}_{ps}$ are also positive definite as products of positive definite matrices.

3.4.3 Proposition

Assume a matrix Q is a symmetric matrix expressed as:

$$Q = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \tag{3-29}$$

Let S be the Schur complement of a matrix A in Q (Arendt & Schleuch, 2009), that is:

$$S = C - B^T A^{-1} B \tag{3-30}$$

Then the matrix Q is positive definite if and only if A and S are both positive definite, i.e. if $A \succ 0$ and $S \succ 0$, then $Q \succ 0$. The proof of this proposition can be found in (Arendt & Schleuch, 2009).

To prove the stability of the new control law, the following matrix, Q, has to be proven to be symmetric positive definite.

$$Q = \begin{bmatrix} \overline{K}_{ps} & \dot{q}_m^2 M_{ss} \\ \dot{q}_m^2 M_{ss} & \dot{q}_m^2 M_{ss} \end{bmatrix} \tag{3-31}$$

Proof: The synchronization control gains are symmetric diagonal matrices with positive constant elements. From Eq. (3-27), it is known that $\overline{K}_{ps}$ is a symmetric positive definite matrix, and $\overline{K}_{ps} \succ 0$. From P1, we know that M_{ss} is a symmetric positive definite, i.e. $M_{ss} = M_{ss}^T$ and $M_{ss} \succ 0$. Therefore, Q is a symmetric matrix.

From conditions (3-27), we have:

$$\lambda_m(\overline{K}_{ps}) > \dot{q}_m^2 \cdot \lambda_M(M_{ss}) > 0 \tag{3-32}$$

From Eq. (3-32) , it can be concluded:

$$\overline{K}_{ps} - \dot{q}_m^2 M_{ss} \succ 0 \tag{3-33}$$

As $K_{ps} \succ 0$ and $M_{ss} \succ 0$, according to P5-P7, we have:

$$M_{ss}^{-1} \succ \dot{q}_m^2 \overline{K}_{ps}^{-1} > 0 \tag{3-34}$$

According to the definition of a positive definite matrix, Eq. (3-34) can be rewritten as:

$$M_{ss}^{-1} - \dot{q}_m^2 \overline{K}_{ps}^{-1} \succ 0 \tag{3-35}$$

Furthermore, based on Eq. (3-35) and $M_{ss} \succ 0$, according to P8, we have:

$$\dot{q}_m^2(M_{ss}(M_{ss}^{-1} - \dot{q}_m^2\overline{K}_{ps}^{-1})M_{ss}) \succ 0 \tag{3-36}$$

Then, according to P1 and reorganizing Eq. (3-36) :

$$S = \dot{q}_m^2 M_{ss} - (\dot{q}_m^2 M_{ss}^T)\overline{K}_{ps}^{-1}\,(\dot{q}_m^2 M_{ss}) \tag{3-37}$$

Hence, according to the Proposition and Eq. (3-37), matrix Q is proved to be symmetric and positive definite.

3.4.4 Stability Analysis

Using the definition of the tracking error in Eq. (3-20), the dynamic model in Eq. (3-18) can be rewritten in an error function format as follows:

$$\begin{cases} \rho = \dot{q}_m^2 M_{ss} q_{sd}''(q_m) + \ddot{q}_m M_{ss} q_{sd}'(q_m) + M_{sm}\ddot{q}_m + C_{sm}\dot{q}_m + G_s + F_s \\ \dot{q}_m^2 M_{ss} e_s''(q_m) + (\overline{K}_{ds} + M_{ss}\ddot{q}_m + C_{ss}\dot{q}_m)e_s'(q_m) + \overline{K}_{ps} = \rho \end{cases} \tag{3-38}$$

According to the properties and the assumptions, we have:

$$\begin{aligned} \rho &\le \|\dot{q}_m^2 M_{ss} q_{sd}''(q_m) + \ddot{q}_m M_{ss} q_{sd}'(q_m) + M_{sm}\ddot{q}_m + C_{sm}\dot{q}_m + G_s + F_s\| \\ &\le m_{ss}\|\dot{q}_m^2 q_{sd}''\| + m_{ss}\|\ddot{q}_m q_{sd}'\| + m_{s1}\|\ddot{q}_m\| + c_{s1}\|\dot{q}_m\| + \|g_s\| + \|f_s\| \\ &= \|\rho\| \end{aligned} \tag{3-39}$$

Eq. (3-39) indicates that parameter ρ is bounded.

For the dynamic system defined in position domain in Eq.(3-18), we define the following Lyapunov function:

$$\begin{aligned} V(e_s(q_m), e_s'(q_m)) &= \tfrac{1}{2}\left(e_s^T\ e_s'^T\right)Q\begin{pmatrix} e_s \\ e_s' \end{pmatrix} + \tfrac{1}{2}e_s^T(K + \overline{K}_{ds})e_s \\ &= \tfrac{1}{2}(e_s^T\ e_s'^T)\begin{bmatrix} \overline{K}_{ps} & M_{ss}\dot{q}_m^2 \\ M_{ss}\dot{q}_1^2 & M_{ss}\dot{q}_m^2 \end{bmatrix}\begin{pmatrix} e_s \\ e_s' \end{pmatrix} + \tfrac{1}{2}e_s^T(K + \overline{K}_{ds})e_s \end{aligned} \tag{3-40}$$

From the above discussion, we know that Q is symmetric positive definite. Therefore the Lyapunov function in Eq. (3-40) is a positive definite function:

$$V(e_s(q_m), e'_s(q_m)) > 0 \tag{3-41}$$

In position domain control, the reference angular position is an independent variable that with the same characteristics and use as t in time domain (monotonically increasing or decreasing, etc). e_s and e'_s are functions of the independent variable. Therefore, the derivative of V is related to variable q_m in this stability analysis.

From Eq. (3-40), the derivative of V along the contour tracking errors of the system is given by:

$$\begin{aligned}\frac{dV}{dq_m} = & \ (e_s^T \ e'^T_s)\begin{bmatrix}\overline{K}_{ps} & M_{ss}\dot{q}_m^2 \\ M_{ss}\dot{q}_m^2 & M_{ss}\dot{q}_m^2\end{bmatrix}\begin{pmatrix}e'_s \\ e''_s\end{pmatrix} + e_s^T(K+\overline{K}_{ds})e'_s \\ & + \frac{1}{2\dot{q}_m}(e_s^T \ e'^T_s)\begin{bmatrix}0 & \dot{M}_{ss}\dot{q}_m^2 \\ \dot{M}_{ss}\dot{q}_m^2 & \dot{M}_{ss}\dot{q}_m^2\end{bmatrix}\begin{pmatrix}e_s \\ e'_s\end{pmatrix} \\ = & \ e_s^T\left(\overline{K}_{ps}+\overline{K}_{ds}+K+\dot{q}_m\dot{M}_{ss}\right)e'_s + e'^T_s\left(\dot{q}_m^2 M_{ss}+\tfrac{\dot{q}_1}{2}\dot{M}_{ss}\right)e'_s \\ & + (e_s^T + e'^T_s)\dot{q}_m^2 M_{ss}e''_s\end{aligned} \tag{3-42}$$

Considering P2 and P3, and applying Eq.(3-26), (3-38), (3-39) to (3-42):

$$\begin{aligned}\frac{dV}{dq_m} = & \ -e_s^T\overline{K}_{ps}e_s - e'^T_s(K_{ds}+(\ddot{q}_m-\dot{q}_m^2)M_{ss})e'_s \\ & + e_s^T(K+\dot{q}_1C_{ss}^T-\ddot{q}_1M_{ss})e'_s + (e_s^T + e'^T_s)\rho\end{aligned} \tag{3-43}$$

Since $K+\dot{q}_1C_{ss}^T-\ddot{q}_1M_{ss}$ is positive definite from Eq. (3-27), we have:

$$e_s^T(K+\dot{q}_1C_{ss}^T-\ddot{q}_1M_{ss})e'_s \le \frac{1}{2}\lambda_M(K+\dot{q}_1C_{ss}^T-\ddot{q}_1M_{ss})(e_s^Te_s + e'^T_s e'_s) \tag{3-44}$$

Applying Eq. (3-44) to Eq. (3-43):

$$\begin{aligned}\frac{dV}{dq_m} \le & \ (\|e_s\| + \|e'_s\|)\|\rho\| - e_s^T\left(\overline{K}_{ps} - \frac{1}{2}\lambda_M(K+\dot{q}_1C_{ss}^T-\ddot{q}_1M_{ss})I\right)e'_s \\ & -e_s^T\left(\overline{K}_{ds}+(\ddot{q}_m-\dot{q}_m^2)M_{ss}+\frac{1}{2}\lambda_M(K+\dot{q}_1C_{ss}^T-\ddot{q}_1M_{ss})I\right)e_s\end{aligned} \tag{3-45}$$

Now, it is assumed:

$$\begin{cases} \rho_e = \lambda_m \left(\overline{K}_{ps} - \frac{1}{2} \lambda_M (K + \dot{q}_1 C_{ss}^T - \ddot{q}_1 M_{ss}) \mathcal{I} \right) \\ \rho_{e\prime} = \lambda_m \left(\overline{K}_{ds} + (\ddot{q}_m - \dot{q}_m^2) M_{ss} + \frac{1}{2} \lambda_M (K + \dot{q}_1 C_{ss}^T - \ddot{q}_1 M_{ss}) \mathcal{I} \right) \end{cases} \tag{3-46}$$

The according to Eq. (3-27), we have $\rho_e > 0$ and $\rho_{e\prime} > 0$. Then Eq. (3-45) can be rewritten as:

$$\frac{dV}{dq_m} \leq -\rho_e \|e_s\|^2 + \|\rho\| \|e_s\| - \rho_{e\prime} \|e'_s\|^2 + \|\rho\| \|e'_s\| \tag{3-47}$$

Applying another inequality:

$$az - bz^2 \leq \frac{a^2}{b} - \frac{1}{4} bz^2 \text{ for } a > 0 \text{ and } b > 0 \tag{3-48}$$

we get:

$$\begin{cases} \|\rho\| \|e_s\| - \rho_e \|e_s\|^2 \leq -\frac{\rho_e}{4} \|e_s\|^2 + \frac{\|\rho\|^2}{\rho_e} \\ \|\rho\| \|e'_s\| - \rho_{e\prime} \|e'_s\|^2 \leq -\frac{\rho_{e\prime}}{4} \|e'_s\|^2 + \frac{\|\rho\|^2}{\rho_{e\prime}} \end{cases} \tag{3-49}$$

Applying Eq. (3-49) to Eq. (3-47) we finally derive:

$$\frac{dV}{dq_m} \leq -\frac{\rho_e}{4} \|e_s\|^2 - \frac{\rho_e}{4} \|e'_s\|^2 + \left(\frac{1}{\rho_e} + \frac{1}{\rho_{e\prime}}\right) \|\rho\|^2 \tag{3-50}$$

Based on the Lyapunov theorem, we conclude that the robotic system controlled by the position domain synchronization control law is globally ultimately bounded. The bounded errors for the tracking error and the relative derivative of the tracking error can be obtained as follows:

$$\begin{cases} \|e_s\| \leq 2 \sqrt{\frac{1}{\rho_e^2} + \frac{1}{\rho_e \rho_{e\prime}}} \|\rho\| \\ \|e'_s\| \leq 2 \sqrt{\frac{1}{\rho_{e\prime}^2} + \frac{1}{\rho_e \rho_{e\prime}}} \|\rho\| \end{cases} \tag{3-51}$$

From Eq. (3-51), it can be seen that the tracking errors can be controlled in a very small boundary layer through proper selection of the proportional and derivative gains. It is shown that the two eigenvalues in Eq. (3-46), associated with the two control gains, have significant contributions in the control of tracking errors. From Eq. (3-46), one can see that a large gain K_{ps} will increase the value of ρ_e, while a large control gain K_{ds} will increase the value of $\rho_{e\prime}$. From Eq. (3-51), we conclude that the increase in both gains will reduce the tracking errors. Such a conclusion is very similar with the synchronization control in the time domain.

3.5 Remarks

In this section the position domain concept and the synchronization principle were explored. A new control law was introduced as a position domain alternative to the time domain PSC control law. The new control law is also based on the synchronization and coupling of the errors of the agent of the system and was proven to be stable in Section 3.4. However, the performance of this control law should be studied and compared to existing control schemes.

Chapter 4: Simulation & Results

4.1 Simulation Setup

A virtual robotic manipulator composed of a planar 3 revolute joints is used for the simulation. The manipulator is assumed to be composed of links of different sizes which are actuated with same type of actuators. This type of configuration was chosen because it was complex enough to show the capabilities of the proposed control law. A spatial or more complex planar configuration was not considered since it would not provide significant indications of the proposed controller's efficiency. The structural parameters of the robotic manipulator are listed on Table 4-1.

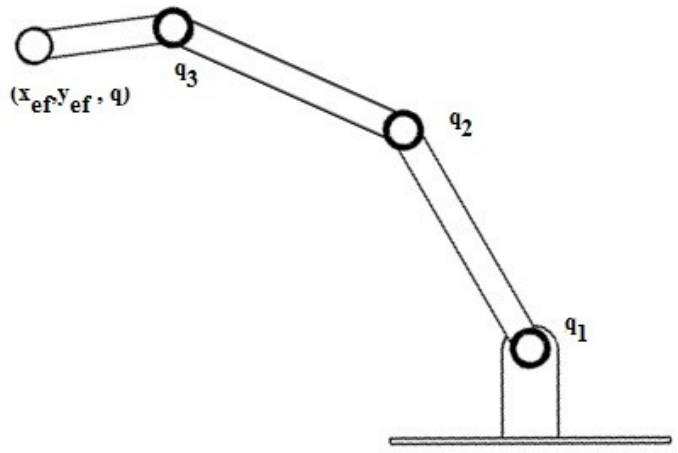

Figure 4-1: 3-DOF Planar Robotic Manipulator

Table 4-1: Structural Parameters of a Serial Robotic Manipulator

Link	Mass $m_i\,(kg)$	Length $l_i\,(m)$	Center $r_i\,(m)$	Inertia $I_i\,(kgm^2)$
1	1.00	0.50	0.25	0.10
2	1.00	0.50	0.25	0.10
3	0.50	0.30	0.15	0.05

For the given manipulator, the control scheme is defined as follows:

$$\begin{cases} \tau_1 = K_{P1} e_1(\mathrm{t}) + K_{d1} \dot{e}_1(\mathrm{t}) \\ \tau_s = K_{Ps} \mathrm{e}_s^*(q_m) + K_{ds} e_s^{*'}(q_m) \end{cases} \quad s = 2,3 \qquad (4\text{-}1)$$

Clearly, the first axis motor is used as the master motion, which is controlled with a conventional PD type controller as described in Eq (2-1) (with K_{I1} set to zero). The rest of the motors are considered the slave motions of the system for which the proposed controller is used as shown in Eq. (4-1). Also, the coupled and synchronization error of the slave motions were defined based on Eqs (3-21) and (3-22).

A number of different contours will be simulated, both linear and nonlinear. For the linear contour types, zigzag and diamond contour motions will be simulated, and for the nonlinear types, the simulation will consist of a circular and epitrochoidal contour. For all the contours, PID and PSC controllers will be used both in the time and position domains in order to compare their performances.

For the time domain controllers, a sampling frequency of $1000\,[Hz]$ was used for the simulations. Due to the complicated kinematics of the robotic manipulator the trajectories of the master and slave motions are nonlinear. Therefore, even with an equidistantly sampled master motion, the speed of the master motion still varies nonlinearly; making it very hard for the sampling frequency of the slave motion's to be constant. Hence, the position domain controllers use variable sampling frequencies based on the master motion's position.

4.2 Trajectory Planning

4.2.1 Fifth-Order Polynomial

To guarantee smooth trajectories for all the joints of the manipulator, a fifth-order polynomial is used to define each contour's position, velocity and acceleration (Dombre & Khalil, 2006). The fifth-order polynomial is defined with respect to time as:

$$r(t) = 10\left(\frac{t}{T}\right)^3 - 15\left(\frac{t}{T}\right)^4 + 6\left(\frac{t}{T}\right)^5 \tag{4-2}$$

where t is time and T is the total time duration of the motion. Then, the velocity and acceleration are given as:

$$\dot{r}(t) = \frac{1}{T}\left\{30\left(\frac{t}{T}\right)^2 - 60\left(\frac{t}{T}\right)^3 + 30\left(\frac{t}{T}\right)^4\right\} \tag{4-3}$$

$$\ddot{r}(t) = \frac{1}{T^2}\left\{60\left(\frac{t}{T}\right) - 180\left(\frac{t}{T}\right)^2 + 120\left(\frac{t}{T}\right)^3\right\} \tag{4-4}$$

4.2.2 Linear Contours

With Eqs (4-2)-(4-4), a linear contour on the end-effector level (task space coordinates) can be defined as follows:

$$P_d = (P_f - P_i)r(t) \tag{4-5}$$

where P_d, is the desired contour expressed in Cartesian coordinates (x_{ef}, y_{ef}) , and P_f and P_i are final and initial points in the contour.

Both zigzag and diamond contours are composed of four linear contours defined by equation (4-5). The transition from one segment to the other is a stop-and-go motion and not continuous as is usually the case for these types of contours.

The direction (q_{ef}) of the end effector is kept constant for each segment. The inverse kinematics equations of the robotic manipulator (found on the appendix) are used to calculate the joint position and velocity required to follow the defined contours. Figure 4-2 to Figure 4-4 show the zigzag contour along with the end-effector and joint level trajectories. Figure 4-5 to Figure 4-7 show the same information for the diamond contour.

Table 4-2 and

Table 4-3 show the final and initial positions on the end-effector and joint space level respectively. Table 4-4 and Table 4-5 indicate the maximum velocities for each segment and once again for the end-effector and joint space respectively. One can see that the end-effector values are similar because the segments were set to be equal in distance and time duration. On the other hand, the joint space values are quite different due to the kinematics of the robotic manipulator.

Table 4-2: Initial and Final End-Effector Positions for Linear Contours

	Segment 1		**Segment 2**		**Segment 3**		**Segment 4**	
	p_i	p_f	p_i	p_f	p_i	p_f	p_i	p_f
Zigzag Contour								
$\boldsymbol{x_{ef}}$ $\boldsymbol{(m)}$	0.50	0.60	0.60	0.70	0.70	0.80	0.80	0.90
$\boldsymbol{y_{ef}}$ $\boldsymbol{(m)}$	0.10	0.40	0.40	0.10	0.10	0.40	0.40	0.10
$\boldsymbol{q_{ef}}$ $\boldsymbol{(rad)}$	1.047	1.047	1.047	1.047	1.047	1.047	1.047	1.047
Diamond Contour								
$\boldsymbol{x_{ef}}$ $\boldsymbol{(m)}$	0.70	0.90	0.90	0.70	0.70	0.50	0.50	0.70
$\boldsymbol{y_{ef}}$ $\boldsymbol{(m)}$	0.70	0.50	0.50	0.30	0.30	0.50	0.50	0.70
$\boldsymbol{q_{ef}}$ $\boldsymbol{(rad)}$	1.047	1.047	1.047	1.047	1.047	1.047	1.047	1.047

Table 4-3: Initial and Final Joint Position for Linear Contours

	Segment 1		Segment 2		Segment 3		Segment 4	
	$q_i\ (rad)$	$q_f\ (rad)$	$q_i\ (rad)$	$q_f\ (rad)$	$q_i\ (rad)$	$q_f\ (rad)$	$q_i\ (rad)$	$q_f\ (rad)$
Zigzag Contour								
$\boldsymbol{q_1}$	-1.604	-0.778	-0.778	-1.244	-1.244	-0.631	-0.631	-0.907
$\boldsymbol{q_2}$	2.352	2.160	2.160	1.922	1.922	1.687	1.687	1.394
$\boldsymbol{q_3}$	0.299	-0.335	-0.335	0.369	0.369	-0.009	-0.009	0.560
Diamond Contour								
$\boldsymbol{q_1}$	-0.114	-0.354	-0.354	-0.914	-0.914	-0.531	-0.531	-0.114
$\boldsymbol{q_2}$	1.578	1.328	1.328	1.973	1.973	2.265	2.265	1.578
$\boldsymbol{q_3}$	-0.417	0.073	0.073	-0.012	-0.012	-0.687	-0.687	-0.417

Table 4-4: Task Space Maximum Velocities

	Segment 1	Segment 2	Segment 3	Segment 4
Zigzag Contour				
$\dot{\boldsymbol{x}}_{\boldsymbol{ef}}\ (\boldsymbol{m/s})$	0.187	-0.187	0.187	-0.187
$\dot{\boldsymbol{y}}_{\boldsymbol{ef}}\ (\boldsymbol{m/s})$	0.562	-0.562	0.562	-0.562
$\dot{\boldsymbol{q}}_{\boldsymbol{ef}}\ (\boldsymbol{rad/s})$	0	0	0	0
Diamond Contour				
$\dot{\boldsymbol{x}}_{\boldsymbol{ef}}\ (\boldsymbol{m/s})$	0.250	-0.250	-0.250	0.250
$\dot{\boldsymbol{y}}_{\boldsymbol{ef}}\ (\boldsymbol{m/s})$	-0.250	-0.250	0.250	0.250
$\dot{\boldsymbol{q}}_{\boldsymbol{ef}}\ (\boldsymbol{rad/s})$	0	0	0	0

Table 4-5: Joint Space Maximum Velocity (in rad/sec)

	Segment 1	Segment 2	Segment 3	Segment 4
Zigzag Contour				
$\dot{\boldsymbol{q}}_{\boldsymbol{1}}$	1.608	-0.961	1.166	-0.584
$\dot{\boldsymbol{q}}_{\boldsymbol{2}}$	-0.509	-0.555	-0.538	-0.632
$\dot{\boldsymbol{q}}_{\boldsymbol{3}}$	-1.312	1.347	-0.774	1.075
Diamond Contour				
$\dot{\boldsymbol{q}}_{\boldsymbol{1}}$	-0.332	-686	0.537	0.510
$\dot{\boldsymbol{q}}_{\boldsymbol{2}}$	-0.356	0.812	0.410	-0.854

$\dot{q}_3$	0.617	−0.173	−0.854	0.349

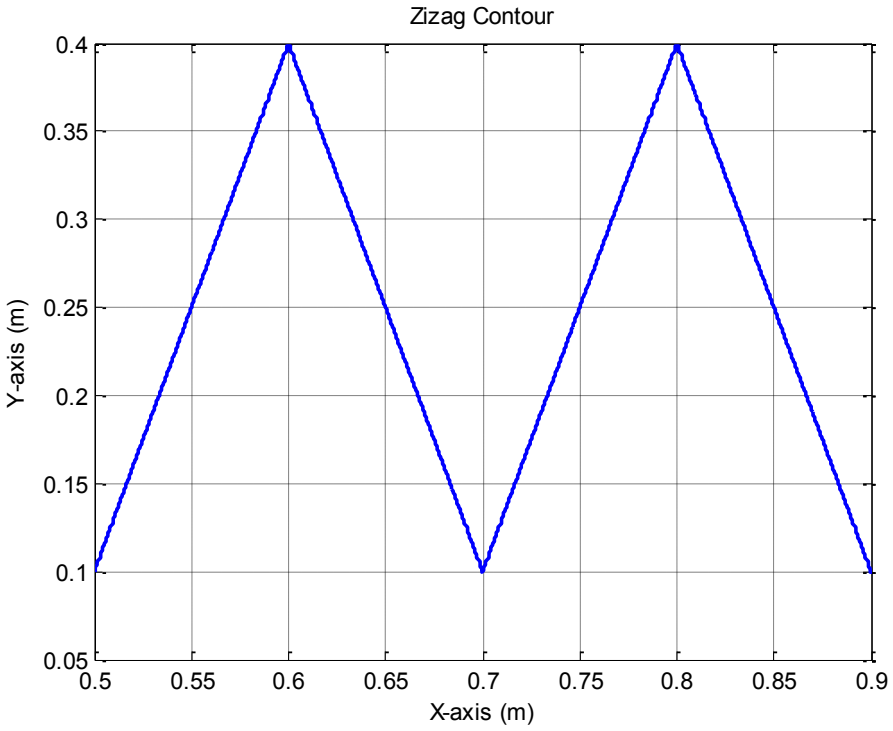

Figure 4-2: Zigzag Contour

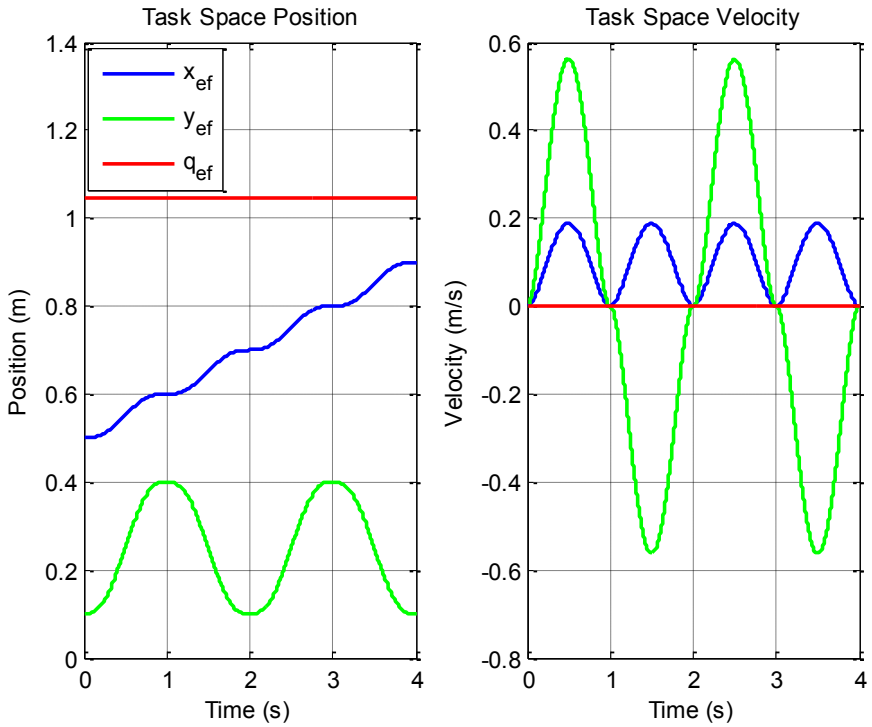

Figure 4-3: Position and Velocity on Task Space (Zigzag Contour)

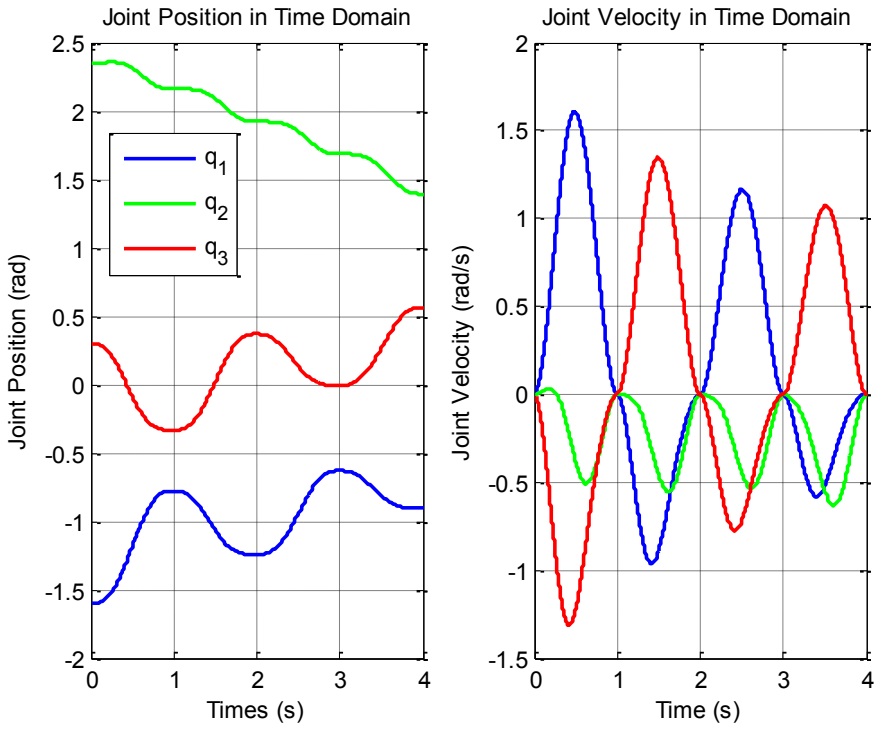

Figure 4-4: Joint Position and Velocity (Zigzag Contour)

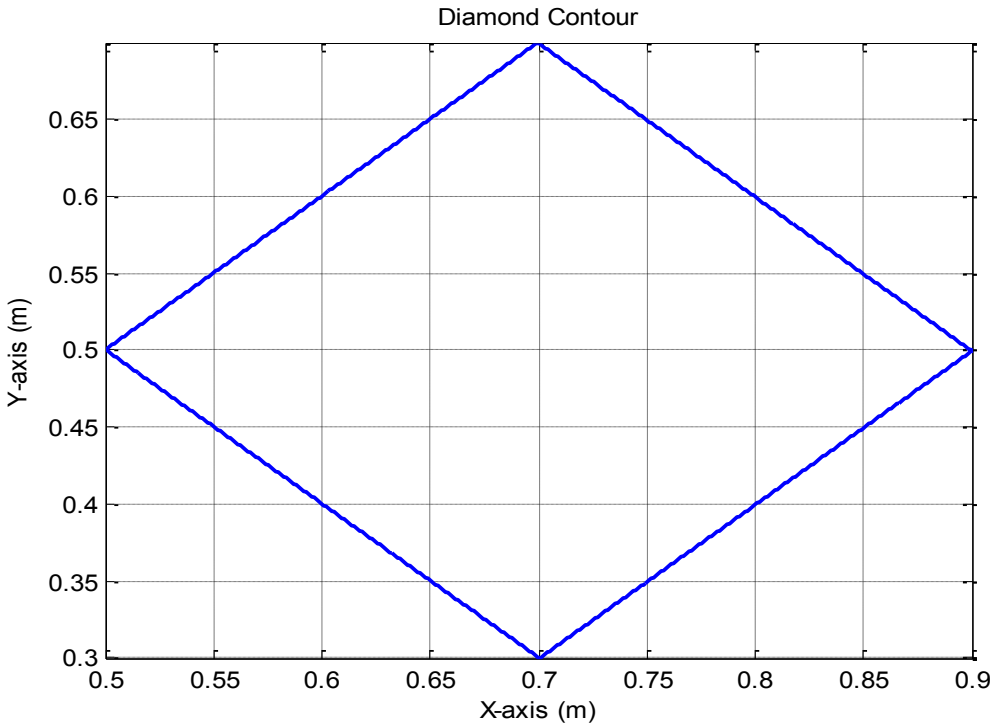

Figure 4-5: Diamond Contour

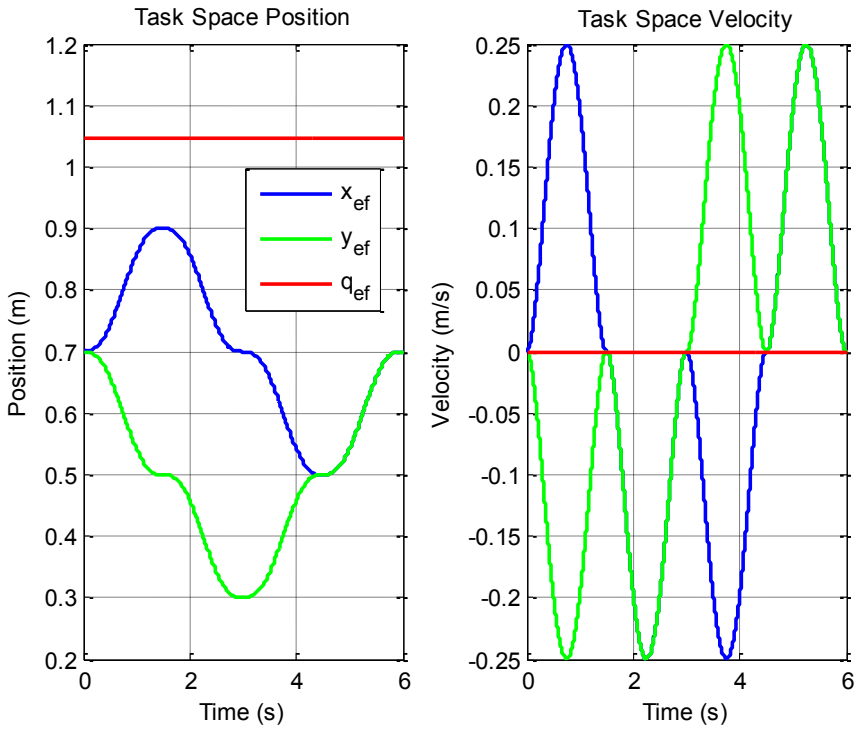

Figure 4-6: Task Space Position and Velocity (Diamond Contour)

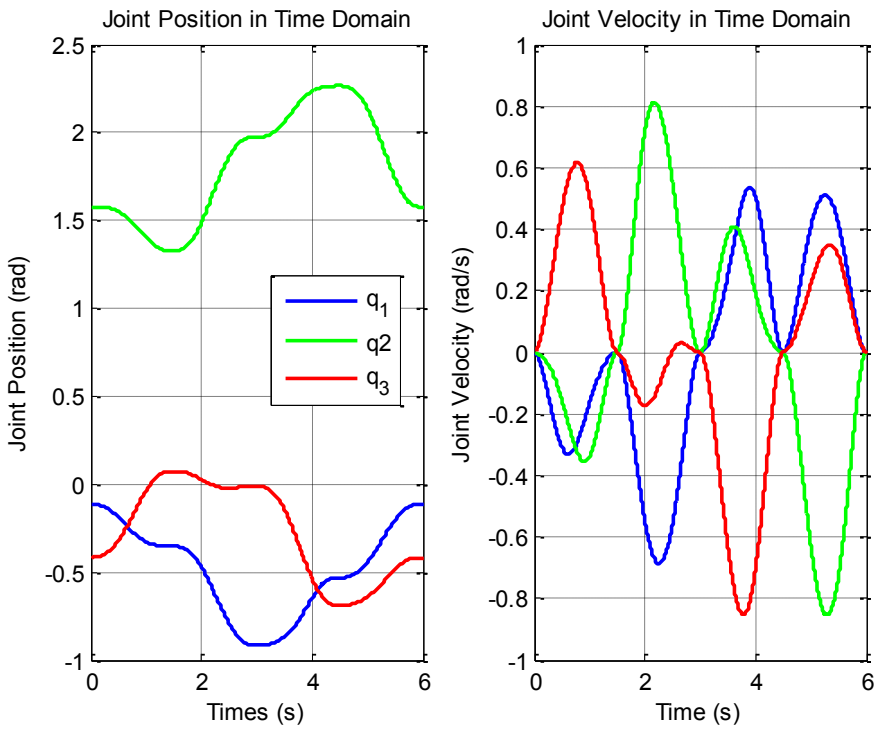

Figure 4-7: Joint Position and Velocity (Diamond Contour)

4.2.3 Nonlinear Contour

To define nonlinear contour, an angular parameter has to be introduced:

$$\theta(t) = r(t)(\theta_f - \theta_i) \tag{4-6}$$

where θ_f and θ_i are the final and initial values for the parameter. This parameter is used to define the nonlinear contours which are expressed by different equations based on the contour.

The circular contour is defined as follows:

$$\begin{cases} x_{ef}(t) = x_c + \mathrm{R} \cdot \cos(\theta(t)) \\ y_{ef}(t) = y_c + \mathrm{R} \cdot \sin(\theta(t)) \end{cases} \tag{4-7}$$

with (x_c, y_c) are the centre of the circle and ρ is radius.

Table 4-6: Circular Contour Parameters

x_c	y_c	R
$\mathbf{0.0}\ [m]$	$0.0\ [m]$	$0.6\ [m]$

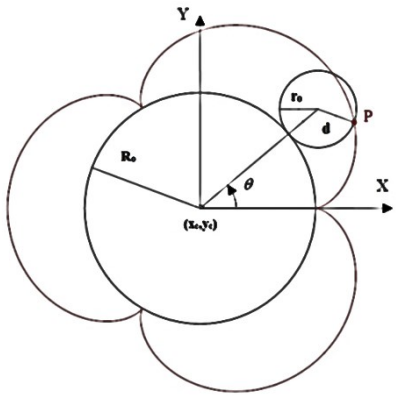

Figure 4-8: Epitrochoid Motion (Hsu, Yan, Liu, & Hsieh, 2008)

On the other hand, the epitrochoidal contour is defined based on the epitrochoidal parametric function:

$$\begin{cases} x_{ef}(t) = x_c + (R_o + r_o)\cos\big(\theta(t)\big) - d \cdot cos\left(\dfrac{R_o + r_o}{r_o}\theta(t)\right) \\ y_{ef}(t) = y_c + (R_o + r_o)\sin\big(\theta(t)\big) - d \cdot sin\left(\dfrac{R_o + r_o}{r_o}\theta(t)\right) \end{cases} \tag{4-8}$$

where (x_c, y_c) is the origin of symmetry, R_o is the radius of the large circle, r_o is the radius of the small circle and d is the distance of the trajectory's points from the centre of the smaller circle as shown in Figure 4-8. The smaller circle moves around the larger circle, translating rod d in the process which creates the resulting contour. The parameters used to form the desired contour are listed in the following table.

Table 4-7: Epitrochoidal Contour Parameters

x_c	y_c	R_o	r_o	d

$\mathbf{0.3}\ [\boldsymbol{m}]$	$0.3\ [m]$	$0.3\ [m]$	$0.1\ [m]$	$0.08\ [m]$

Both nonlinear contours are performed on their whole and are not segments. For the circular contour, the direction (q_{ef}) of the end-effector was kept constant, but for the epitrochoidal contour Eq.(4-6) was used to define the end effector direction. As in the linear contour case, the joint trajectories are calculated based on the manipulators inverse kinematics equations which can be found in the appendix.

Figure 4-9 to Figure 4-14 display the contours and trajectories for both circular and epitrochoidal contour. Table 4-8 and Table 4-9 show the initial and final positions and the maximum velocity for each contour on the end-effector level and joint space level, respectively. Once more the two tables feature significantly different values due to the robotic manipulator's kinematics.

Table 4-8: End-Effector Trajectory for Nonlinear Contours

	Initial Position	**Final Position**	**Max velocity**
Circular contour			
$\boldsymbol{x_{ef}}$	$0.60\ [m]$	$0.60\ [m]$	$0.768\ [m/s]$
$\boldsymbol{y_{ef}}$	$0.0\ [m]$	$0.0\ [m]$	$-0.884\ [m/s]$
$\boldsymbol{q_{ef}}$	$1.047\ [rad]$	$1.047\ [rad]$	$0\ [rad/s]$
Epitrochoidal contour			
$\boldsymbol{x_{ef}}$	$0.62\ [m]$	$0.62\ [m]$	$1.561[m/s]$
$\boldsymbol{y_{ef}}$	$0.30\ [m]$	$0.30\ [m]$	$-2.120\ [m/s]$
$\boldsymbol{q_{ef}}$	$0.524\ [rad]$	$0.524\ [rad]$	$0.0\ [rad/s]$

Table 4-9: Joint Space Trajectory for Nonlinear Contours

	$\boldsymbol{q_i\ (rad)}$	$\boldsymbol{q_f\ (rad)}$	**Max Velocity (rad/s)**
Circular contour			
$\boldsymbol{q_1}$	-1.548	4.735	2.289
$\boldsymbol{q_2}$	2.049	2.049	-1.048
$\boldsymbol{q_3}$	0.546	-5.737	-2.104
Epitrochoidal contour			

q_1	-0.775	5.508	5.184
q_2	2.339	2.339	2.287
q_3	-1.041	-7.324	-5.420

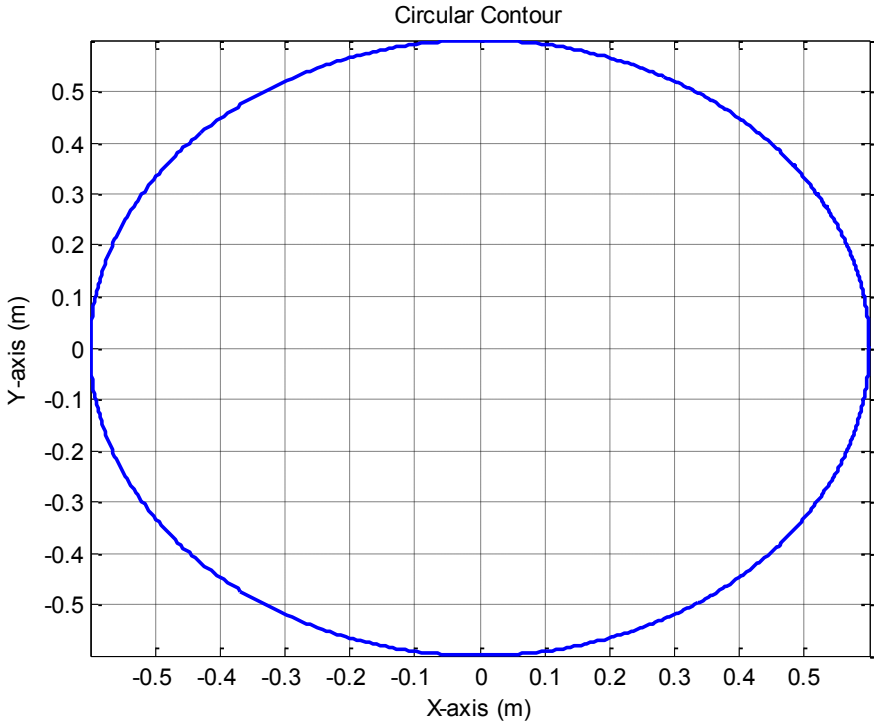

Figure 4-9: Circular Contour

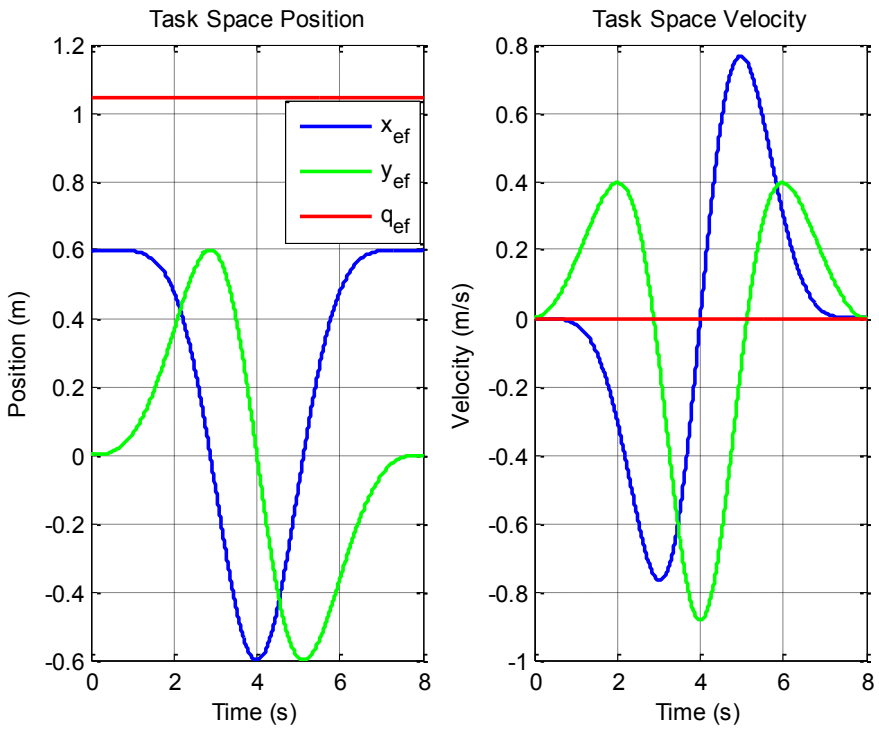

Figure 4-10: Task Space Position and Velocity (Circular Contour)

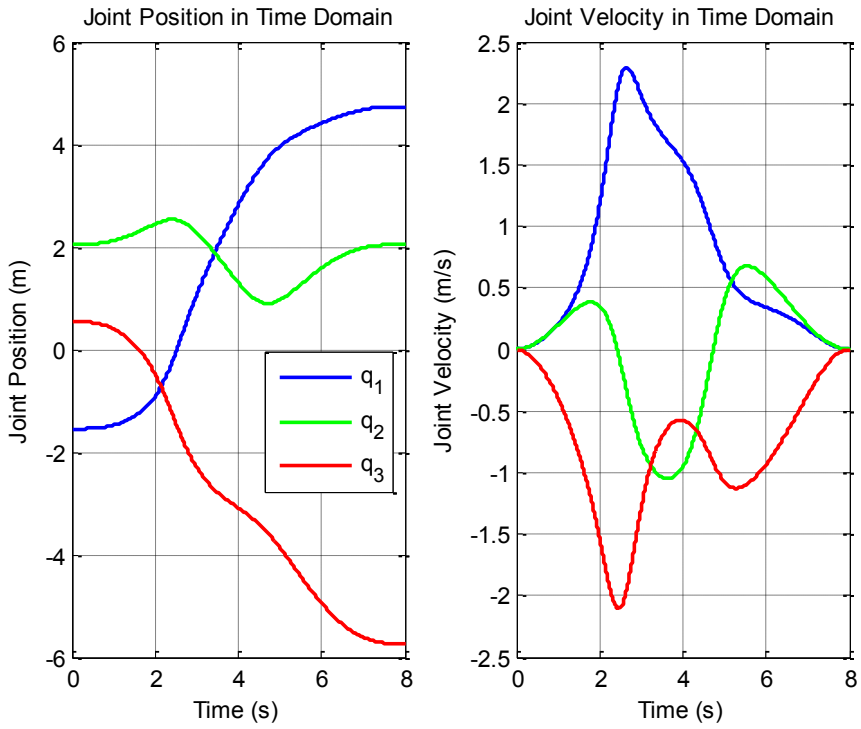

Figure 4-11: Joint Position and Velocity (Circular Contour)

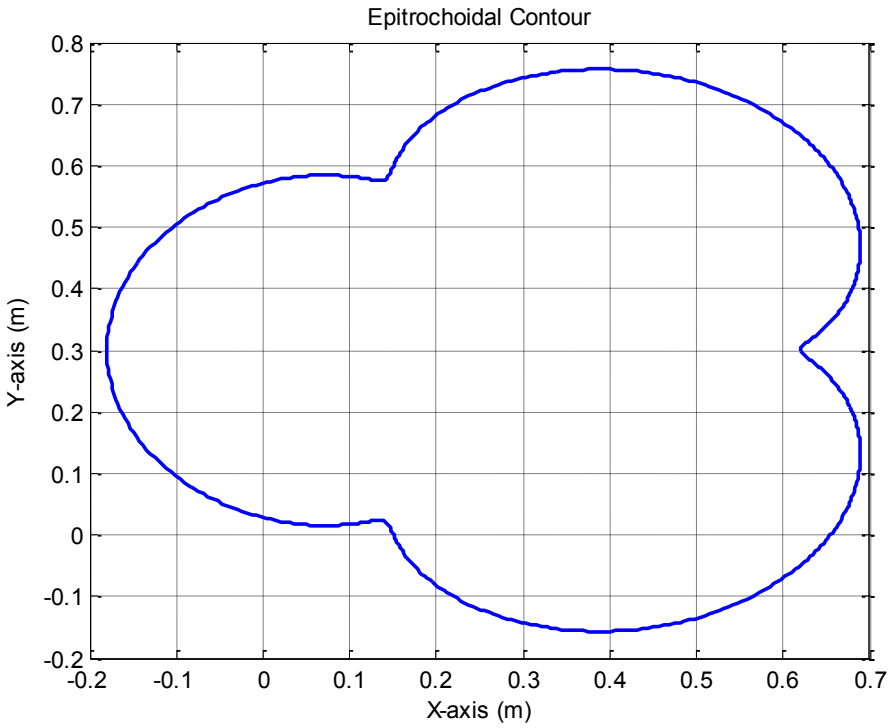

Figure 4-12: Epitrochoidal Contour

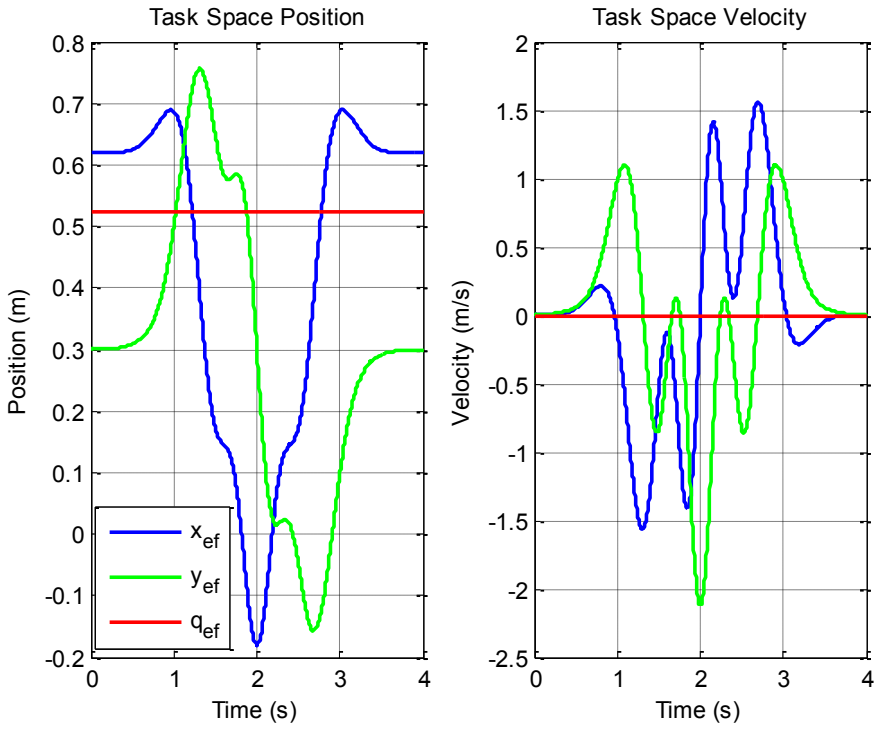

Figure 4-13: Task Space Position and Velocity (Epitrochoidal Contour)

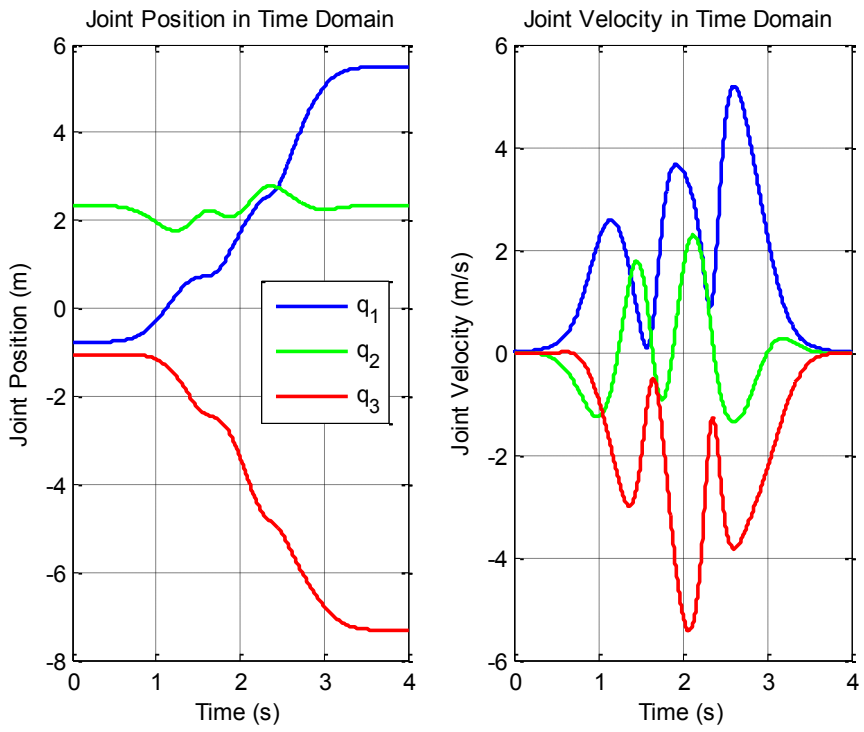

Figure 4-14: Joint Position and Velocity (Epitrochoidal Contour)

4.3 Trajectory Transformation

As mentioned in Section 3.3, in position domain control the slave motion has to be transformed from the time domain to position domain. This means that the desired trajectory of the slave axes has to be also transformed from functions of time to functions of the master axis position.

For the purposes of the following simulations, lookup tables are used for the transformation of the desired axis trajectories. More specifically, the desired trajectory of each axis is initially defined as a function of time as explained in Section 4.2. Afterwards, the resulting trajectories are linearly interpolated with respect to the master axis' real trajectory, as shown in Figure 4-15. This way the desired slave axes position is found for the equivalent master

axis real trajectory. This procedure was easily performed in the simulations by using the *intep1* in MATLAB.

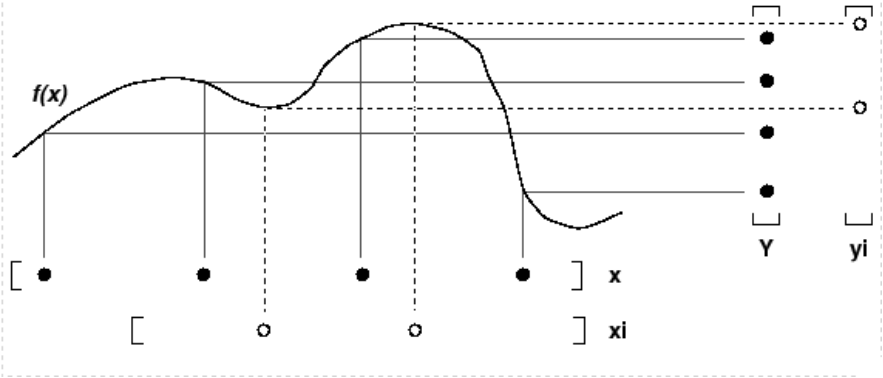

Figure 4-15: Lookup Table (interp1) (The MathWorks, Inc)

The resulting contours derived from the above procedure can be seen in the following figures. Figure 4-16 to Figure 4-19 contain in sequence the resulting trajectories for the zigzag, diamond, circular and epitrochoidal contours respectively.

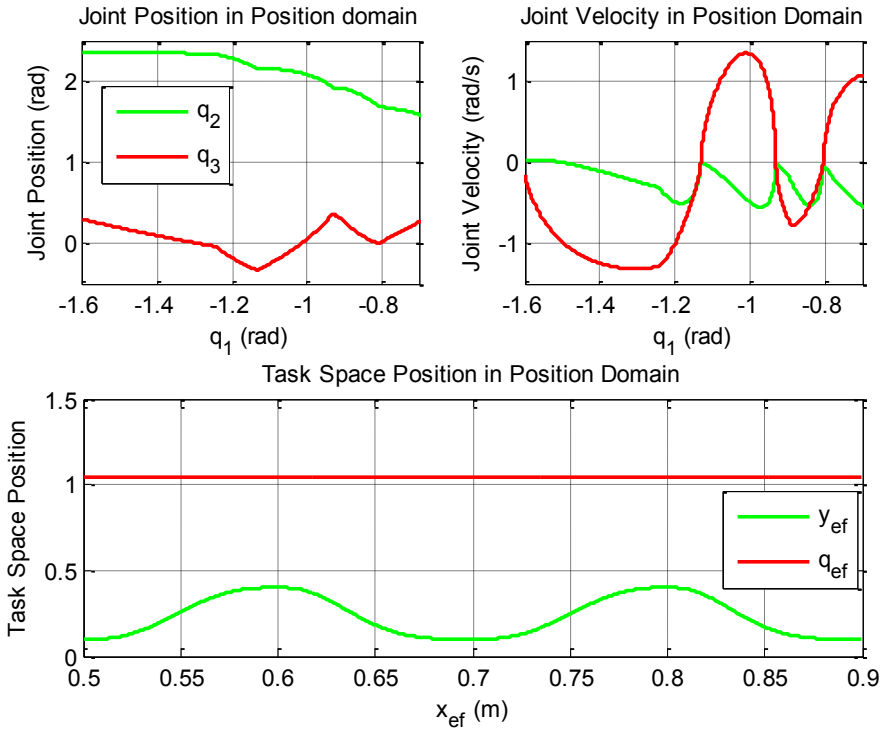

Figure 4-16: Position Domain Slave Trajectories for Zigzag Contour

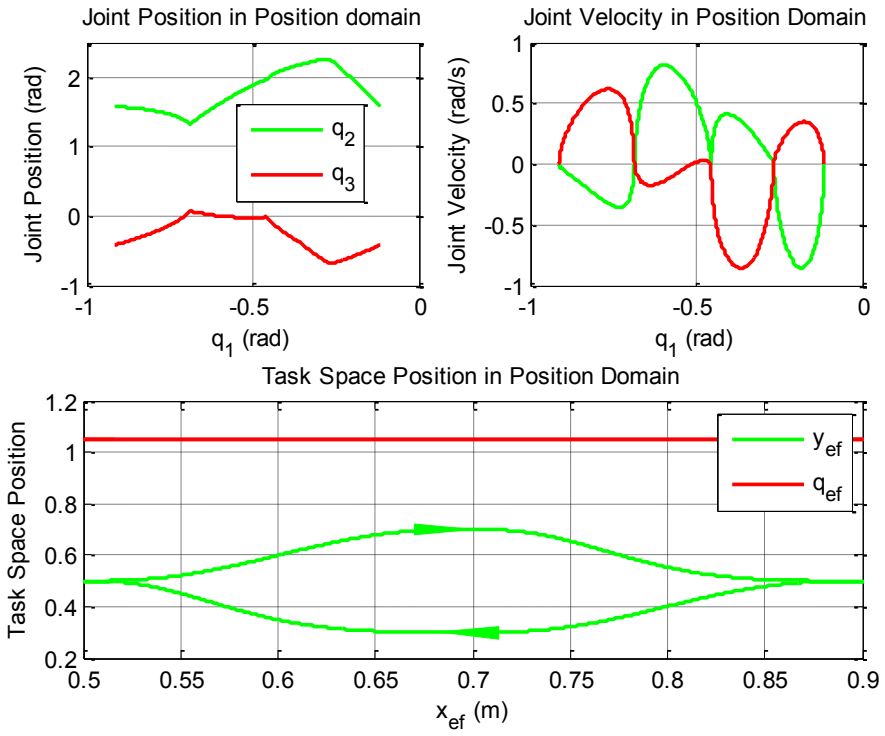

Figure 4-17: Position Domain Slave Trajectories for Diamond Contour

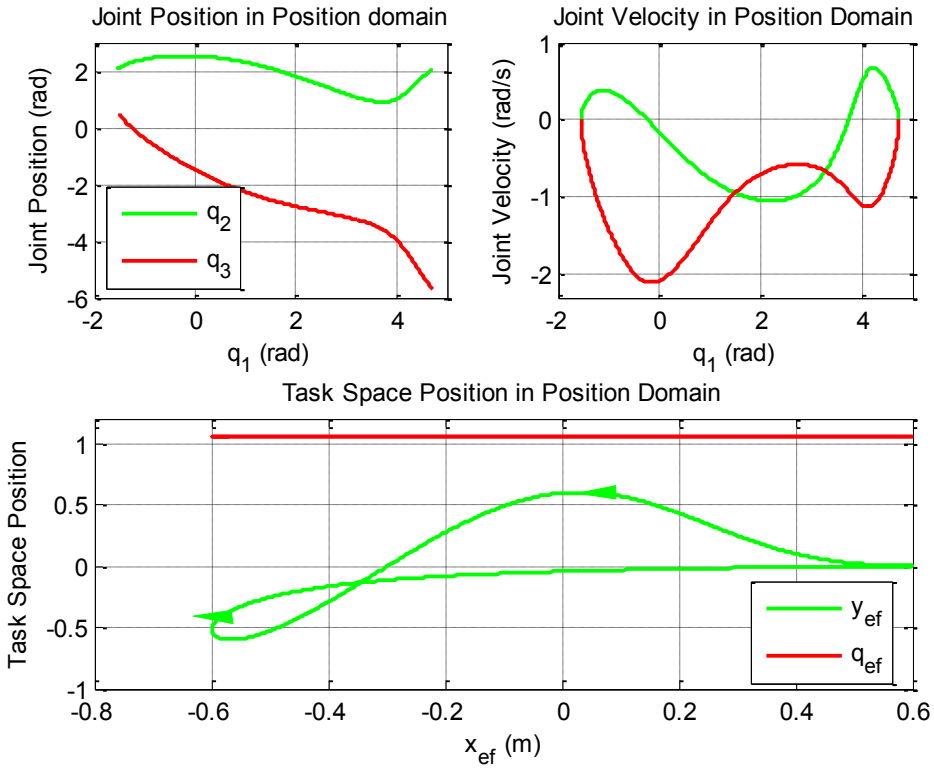

Figure 4-18: Position Domain Slave Trajectories for Circular Contour

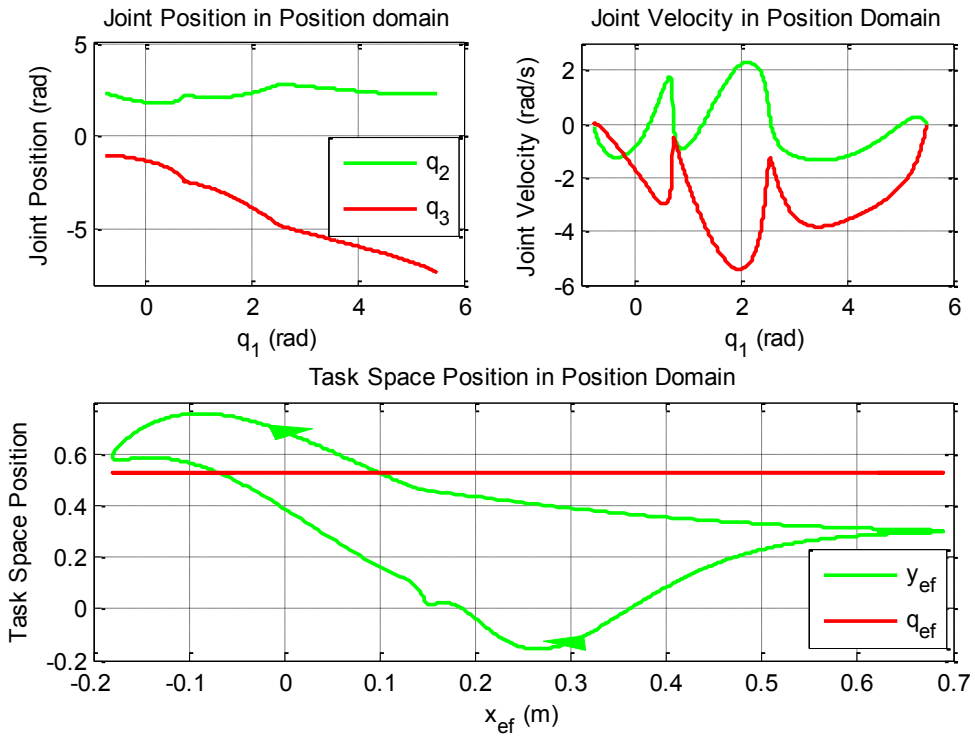

Figure 4-19: Position Domain Slave Trajectories for Epitrochoidal Contour

4.4 Friction Model

In an effort to make a realistic simulation and demonstrate the robustness of the new controller, the friction of the robotic manipulators joints is added to the simulated model. The friction was modeled based on the following equation:

$$\mathrm{F}(\dot{q}_i) = \left\{ f_c + (f_s - f_c) e^{-\left|\frac{\dot{q}_i}{f_t}\right|^{\delta}} \right\} \mathrm{sgn}(\dot{q}_i) + b\dot{q}_i \qquad (4\text{-}9)$$

where f_c is the Coulomb friction force, f_t is the magnitude of the Stribeck friction, b is the viscous friction parameter, f_s is the static friction coefficient and δ is an empirically determined parameter.

The model of Eq (4-9) predicts the static, viscous and break-away friction values for the joint actuators of the robotic manipulators (Armstrong-Helouvry, Dupont, & Canudas De Wit, 1994). Table 4-10 shows the friction parameters used for the simulations. Since the joint actuators are assumed to be of the same type, the same parameters are used for each joint.

Table 4-10: Friction Parameters

Parameter	Value
Coulomb friction (f_c)	3 $[N]$
Stribeck friction (f_t)	100 $[N]$
Static friction (f_s)	5 $[N]$
Viscous friction (b)	1.5 $[N\ s/rad]$
δ	2

4.5 Gain Effect

As with every controller, the gains of the proposed control law in Eq (3-23) affect the behaviour of the system in different ways. Simulations for a linear contour were performed with the values of the gains changing with each one. The gain matrices are diagonal and for simplicity all the non-zero entries have the same value for each gain. Three sets of simulations are performed with the value of only one gain varying on each set while the other ones remain the same. The mean of the absolute of each error is used to compare the performance of each gain value. The range of values of the gains can be seen in Table 4-11.

Table 4-11: Gain Range

Gain	Min. Value	Max Value	Increment
K_p	1000	10000	1000
K_d	1000	10000	1000
β	1	20	2

Figure 4-20 shows the effects of K_p on the systems performance. Increasing the proportional gain matrices value led to decrease in the contour error (e_c) , a trend which is also

followed by the synchronization (ε) and the coupled error(e^*). However, the tracking error of the second axis reached a minimum with $K_{pi} \approx 6000$, while the tracking error for the third axis decreases only slightly.

On the other hand, Figure 4-21 shows that an increase in the derivative gain K_{di}results in increasing contour, synchronization and coupled errors. Conversely, the second axis tracking error decreases as the derivative gain grows in value.

An increasing β_i value leads to decreases in the value of the contour error but increases the synchronization and coupled errors. The tracking error of the second axis decreases with the first changes in the value of β_i but then it increases again while the third axis maintains a slight decrease. The high increase of the coupled error indicates that the value of β_i should be maintained low enough in order for the error to be maintained low in value as well.

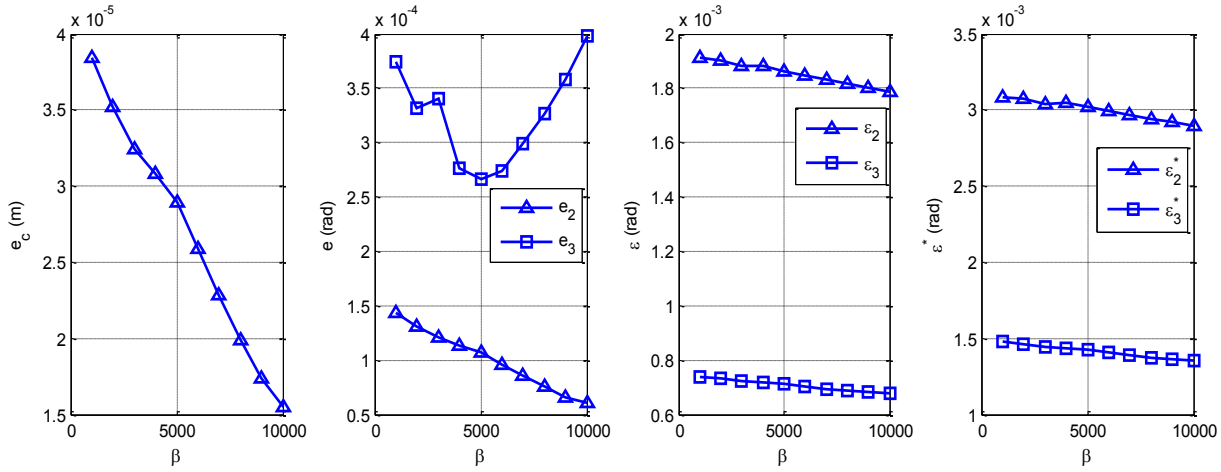

Figure 4-20: Variable K_p

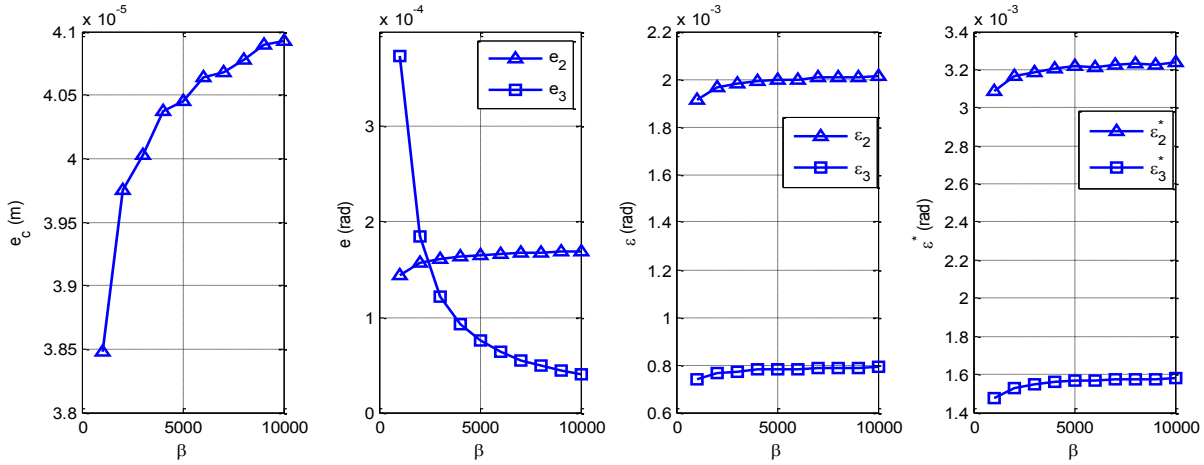

Figure 4-21: Variable K_d

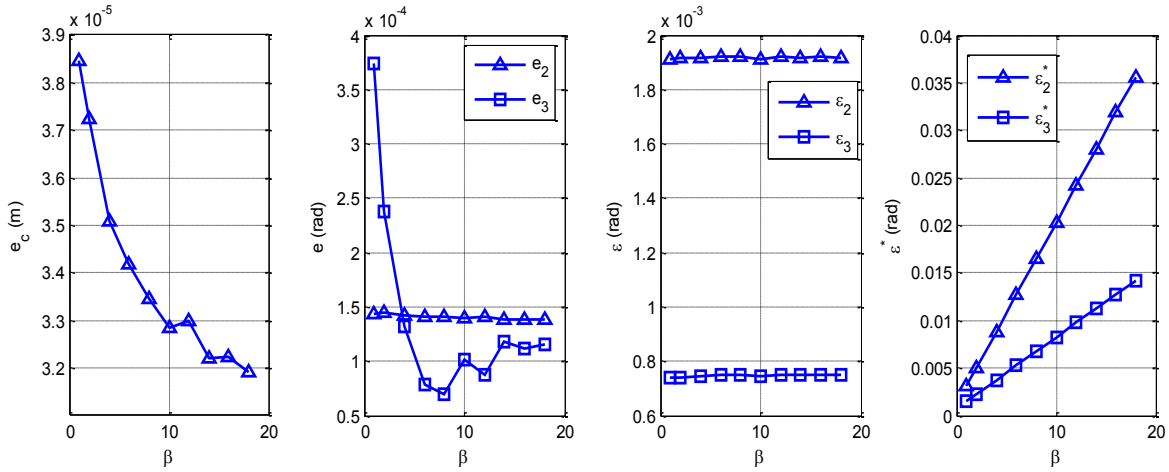

Figure 4-22: Variable β

4.6 Gain Selection

Based on the observations in Section 4.5, a simple algorithm was used to select the gains of the controller. The algorithm is composed of the following steps:

1. Set $\beta = 0$, making the PDSC controller similar to a proportional-derivative controller.
2. Tune K_{ps} and K_{ds} via a trial-and-error until a desired performance is achieved.
3. Increase the value of β until desired performance due to synchronization is achieved.
4. Further adjust K_{ps} and K_{ds} if needed.

The above algorithm is quite similar to the one used by (Sun, 2011) to tune the PSC controller, an algorithm which was also used to determine the K_s gain. The gain used for the simulations in the later sections can be seen in Table 4-12.

Table 4-12: Simulation Control Gains

Controller Type	*Linear Contour*	*Nonlinear Contour*
Time Domain PD Controller	$K_p = diag\{[9090, 8300, 6720]\}$ $K_d = diag\{[9870, 7900, 7475]\}$	$K_p = diag\{[6895, 7500, 7530]\}$ $K_d = diag\{[4850, 7350, 8580]\}$
Position Synchronization Controller	$K_p = diag\{[9090, 8300, 6720]\}$ $K_d = diag\{[9870, 7900, 7475]\}$ $K_s = diag\{[9.6, 4.9, 1.2]\}$ $B = diag\{[6.5, 6.7, 8.5]\}$	$K_p = diag\{[6895, 7500, 7530]\}$ $K_d = diag\{[4850, 7350, 8580]\}$ $K_s = diag\{[580, 765, 690]\}$ $B = diag\{[13.1, 13.4, 12.9]\}$
Position Domain PD Controller	$K_p = diag\{[9090, 8300, 6720]\}$ $K_d = diag\{[9870, 2900, 7475]\}$	$K_p = diag\{[6895, 7500, 7530]\}$ $K_d = diag\{[4850, 7350, 8580]\}$
Position Domain Synchronization Controller	$K_p = diag\{[9090, 8300, 6720]\}$ $K_d = diag\{[9870, 7900, 7475]\}$ $B = diag\{[6.5, 6.7, 8.5]\}$	$K_p = diag\{[6895, 7500, 7530]\}$ $K_d = diag\{[4850, 7350, 8580]\}$ $B = diag\{[13.1, 13.4, 12.9]\}$

4.7 Linear Contour Tracking Results

4.7.1 Zigzag Contour

All the simulated control schemes, TD-PD, PDSC, PD-PD and PDSC achieved good axial tracking performance for a zigzag motion with the control gains of Section 4.6. As seen in Table 4-13 the position domain controllers were able to produce less tracking error than the time domain controllers with the PDSC controller being approximately 30 % more exact that then PD-PD controller. All the tracking errors are also displayed in Figure 4-23. Concerning synchronization and coupled errors, PDSC outperformed the PSC controller by achieving

12.5% (axis 2) and 50%(axis 3) lower synchronization error with equally lower standard deviations. Similar results were produced for the coupled error, where the PSC exhibited higher that the PDSC values by 20% (axis 2) and 47% (axis 3), as seen also in Table 15, Figure 4-24 and Figure 4-25.

It should be noted that for the position domain controllers (PD-PD and PDSC), the first axis is the master axis and its position acts as the reference for the motion of the rest axis. Hence the first axis yields zero tracking, synchronization and coupled error in the case of the position domain controllers.

Table 4-13: Mean and Standard Deviation for the Tracking Error for Zigzag Motion

	Axis 1		**Axis 2**		**Axis 3**	
	Mean (rad)	S.D. (rad)	Mean (rad)	S.D. (rad)	Mean (rad)	S.D. (rad)
TD-PD	0.00032	0.00027	0.00030	0.00019	0.00020	$9.849\ 10^{-5}$
PSC	0.00024	0.00015	0.00023	0.00014	0.00022	0.00013
PD-PD	N/A	N/A	$9.977\ 10^{-5}$	$8.210\ 10^{-5}$	0.00015	0.00013
PDSC	N/A	N/A	$7.063\ 10^{-5}$	$7.674\ 10^{-5}$	0.00010	$8.545\ 10^{-5}$

Table 4-14: Mean and Standard Deviation of the Synchronization and Coupled Errors for Zigzag Motion

	Axis 1		**Axis 2**		**Axis 3**	
	Mean (rad)	S.D. (rad)	Mean (rad)	S.D. (rad)	Mean (rad)	S.D. (rad)
Synchronization Error (ε)						
PSC	0.00030	0.00030	0.00032	0.00039	0.00058	0.00040
PDSC	N/A	N/A	0.00028	$8.962\ 10^{-5}$	0.00029	$9.290\ 10^{-5}$
Coupled Error (e^*)						
PSC	0.00217	0.00208	0.00232	0.00206	0.00519	0.00362
PDSC	N/A	N/A	0.00184	0.00062	0.00272	0.00088

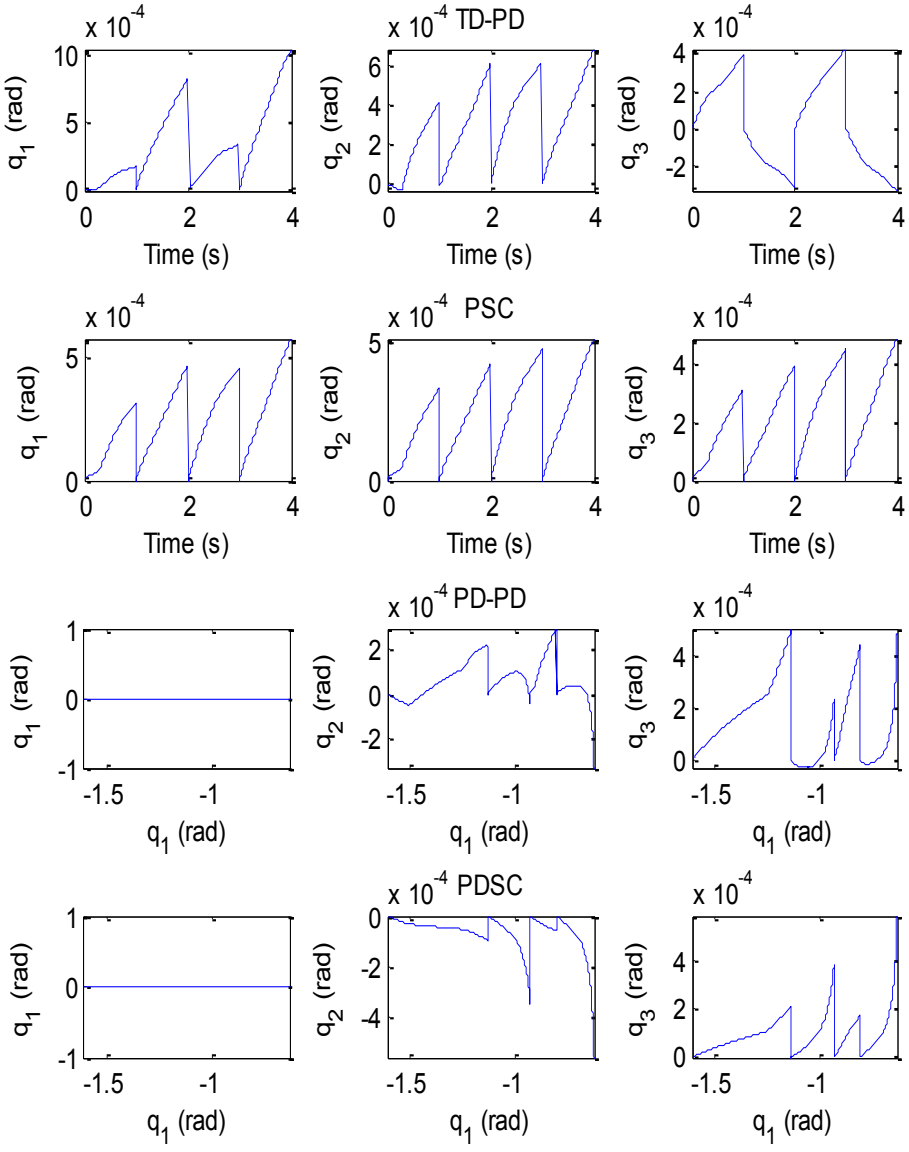

Figure 4-23: Zigzag Motion Tracking Error for TD-PD, PSC, PD-PD and PDSC

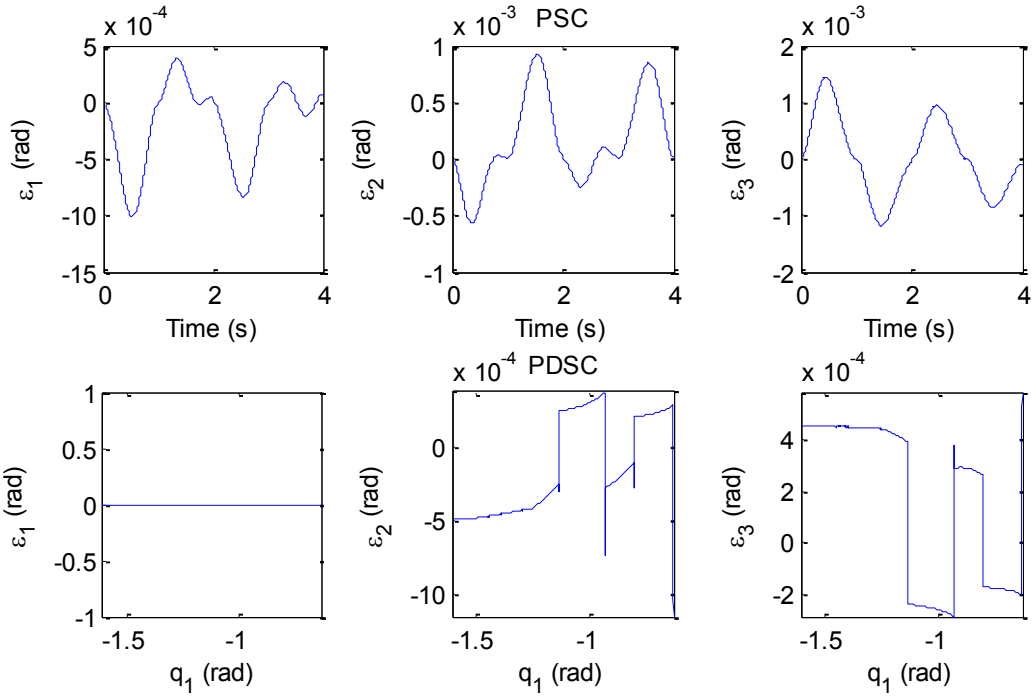

Figure 4-24: Zigzag Motion Synchronization Error for PSC and PDSC

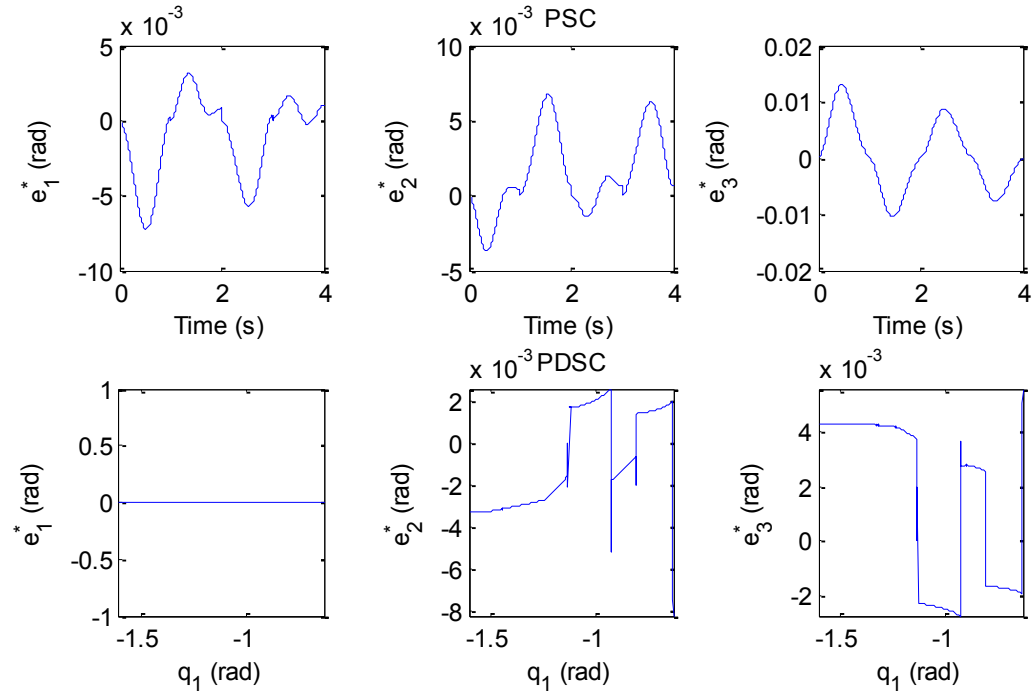

Figure 4-25: Zigzag Motion Coupled Error for PSC and PDSC

Regarding contour tacking accuracy, the position domain controllers were able to perform better than the time domain controllers, as previous research had also demonstrated (Ouyang, Dam, Huang, & Zhang, 2012). The PD-PD controller produced 47% and 52% lower contour error than the TD-PD and PSC controllers, respectively, but the PDSC controller outperformed the PD-PD by 88.4% as seen in Table 4-15. Hence, it is clear that the PDSC controller achieved the best contour performance with the least deviation from the desired path, something also supported evident by Figure 4-26.

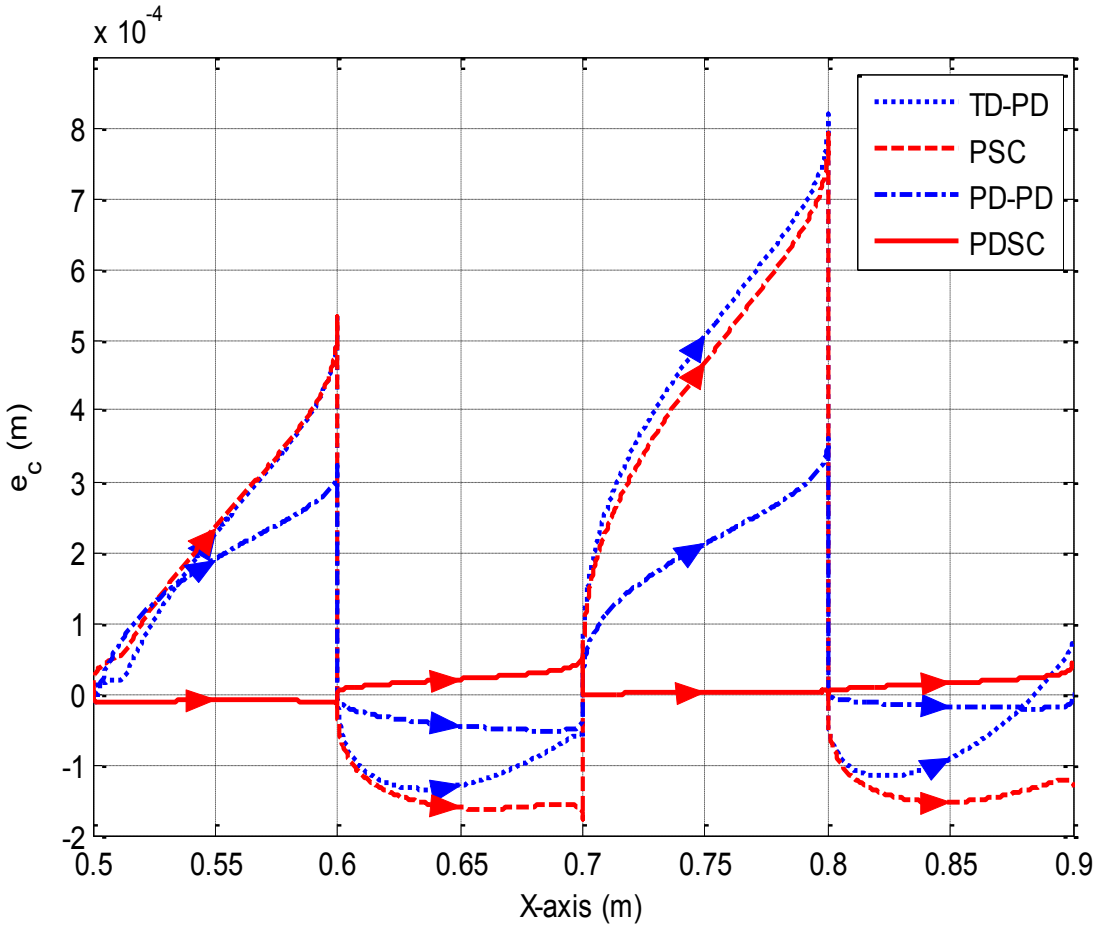

Figure 4-26: Zigzag Motion Contour Tracking Performance for TD-PD, PSC, PD-PD and PDSC

Table 4-15: Mean and Standard Deviation of Contour Tracking for Zigzag Motion[1]

	Mean (m)	S.D. (m)
TD-PD	0.00021	0.00022
PSC	0.00023	0.00020
PD-PD	0.00011	0.00011
PDSC	$1.273\ 10^{-5}$	$1.199\ 10^{-5}$

As shown in Table 4-16, the position domain controllers produced higher maximum values for the second axis while the maximum torque for the third axis remained essentially the same. This was expected as the trajectories on the position domain are rather different than the time domain ones and create different values in the required torques. The torques produced by the PDSC controller can be seen in Figure 4-27.

Table 4-16: Maximum Torque for Zigzag Contour (in Nm)

	q_1	q_2	q_3
TD-PD	17.250	10.292	5.386
PSC	17.253	10.291	5.391
PD-PD	17.250	12.634	5.363
PDSC	17.250	12.632	5.363

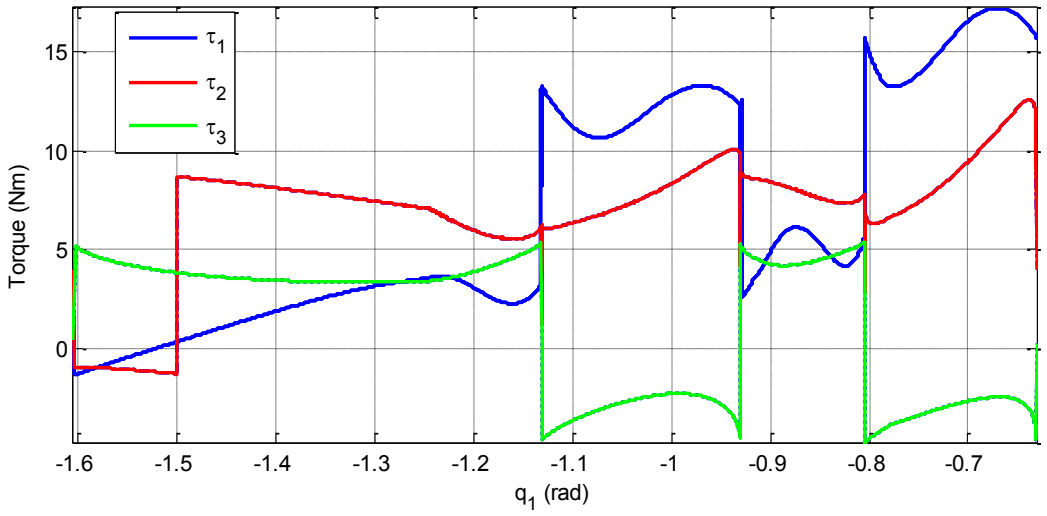

Figure 4-27: Zigzag Contour Torques for PDSC Controller

[1] Contour Error calculated with Eq (2-14)

4.7.2 Diamond Contour

Similar to the zigzag motion, the diamond contour simulations produced good axial tracking results for all the controllers. Still, the PDSC controller yielded better results than the rest of the controllers as it can be seen in Figure 4-29. More specifically, Table 4-17 shows that the PDSC controller produced approximately 70% less error than the PSC controller and approximately 32 % error from the PD-PD controller. The standard deviation of the mean error followed in the same trend, with the standard deviation of the PDSC controller being an order of magnitude lower than the rest of the controllers.

On the other hand, the PSC controller outperformed the PDSC controller on the synchronization error of the second axis by 27.6% and produced only 18% higher error than the PDSC on the third axis with the standard deviation following the same trend. Likewise, PSC produced by 15% less coupled error for the second axis but 8% greater error for the third axis as shown in Table 4-18. The standard deviations of the error had similar terms with the PDSC standard deviations being $1-2\%$ higher for the second axis but $22-40\%$ lower than the PSC standard deviation for the third axis.

Table 4-17: Mean and Standard Deviation of Axial Tracking Error for Diamond Motion

	Axis 1		**Axis 2**		**Axis 3**	
	Mean (rad)	S.D. (rad)	Mean (rad)	S.D. (rad)	Mean (rad)	S.D. (rad)
TD-PD	0.00044	0.00034	0.00021	0.00017	0.00027	0.00013
PSC	0.00035	0.00041	0.00027	0.00031	0.00030	0.00036
PD-PD	N/A	N/A	0.00011	0.00011	0.00015	0.00013
PDSC	N/A	N/A	$7.607\ 10^{-5}$	$8.054\ 10^{-5}$	$8.647\ 10^{-5}$	$8.743\ 10^{-5}$

Table 4-18: Mean and Standard Deviation for Synchronization and Coupled Error for Diamond Motion

	Axis 1		Axis 2		Axis 3	
	Mean (rad)	S.D. (rad)	Mean (rad)	S.D. (rad)	Mean (rad)	S.D. (rad)
Synchronization Error (ε)						
PSC	0.00019	0.00020	0.00029	0.00020	0.00021	0.00020
PDSC	N/A	N/A	0.00037	0.00022	0.00018	0.00015
Coupled Error (e^*)						
PSC	0.00143	0.00132	0.00223	0.00142	0.00182	0.00188
PDSC	N/A	N/A	0.00264	0.00158	0.00167	0.00145

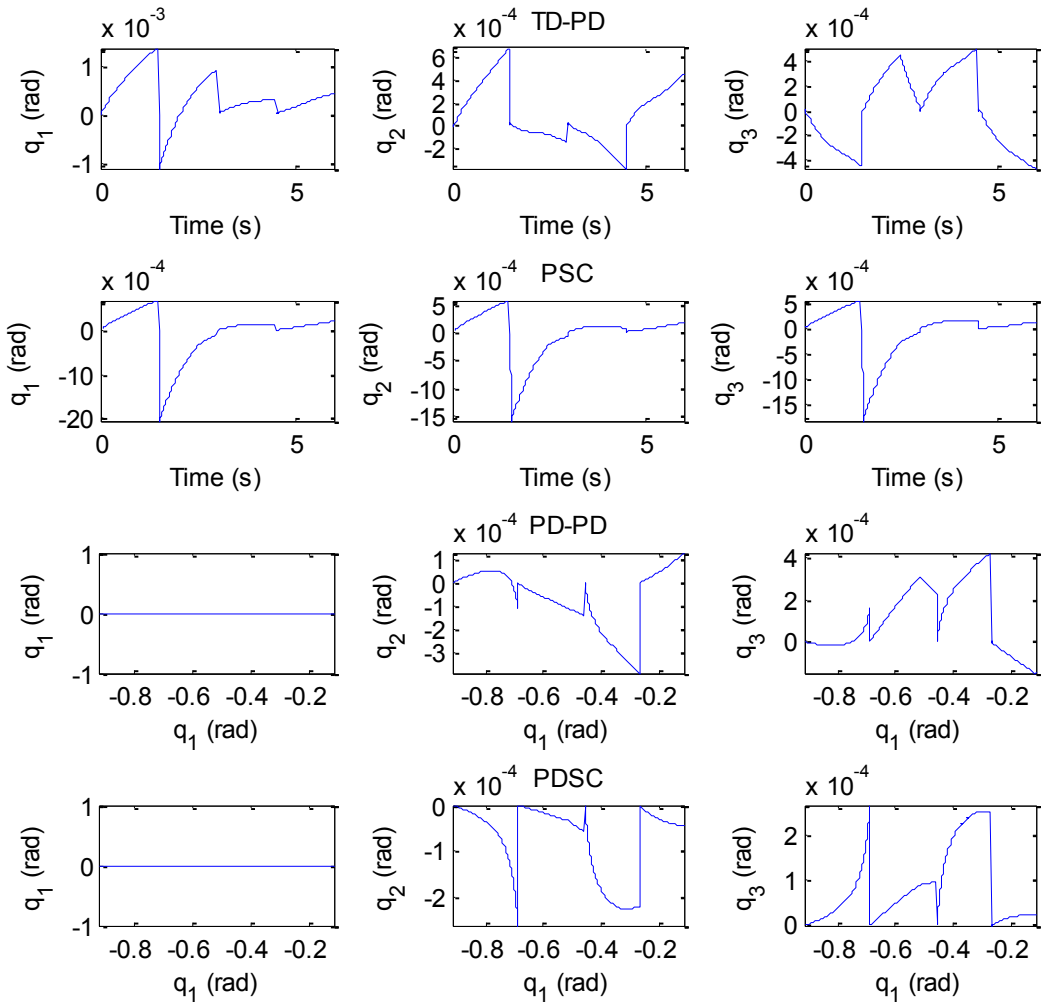

Figure 4-28: Diamond Motion Axial Tracking Error for TD-PD, PSC, PD-PD, PDSC

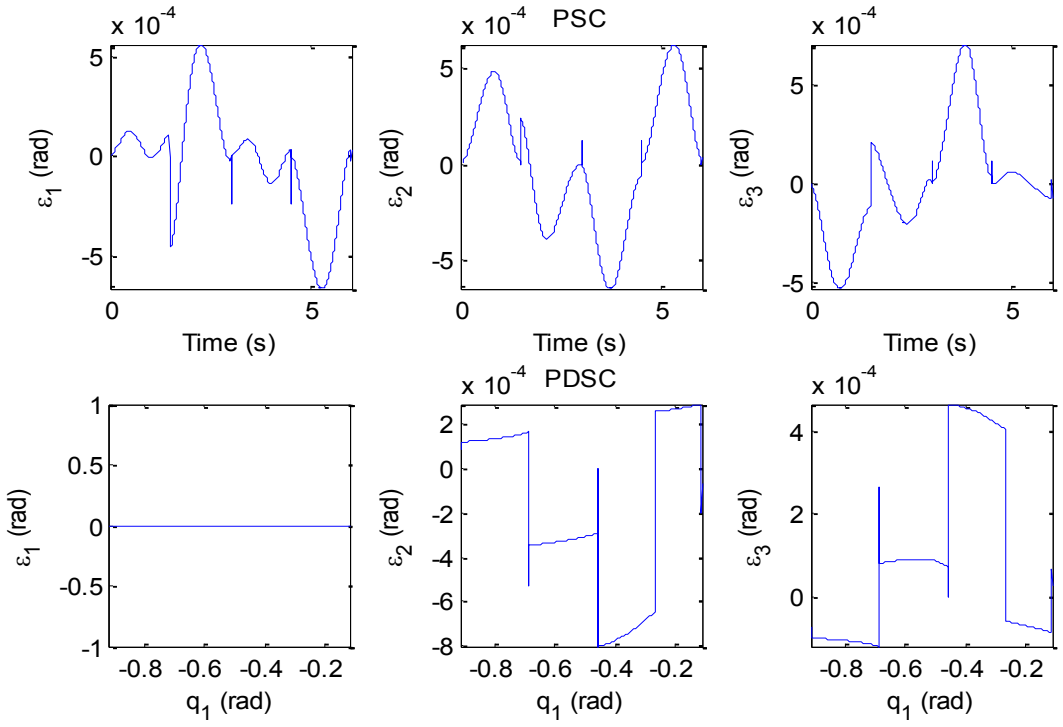

Figure 4-29: Diamond Motion Synchronization Error for PSC and PDSC

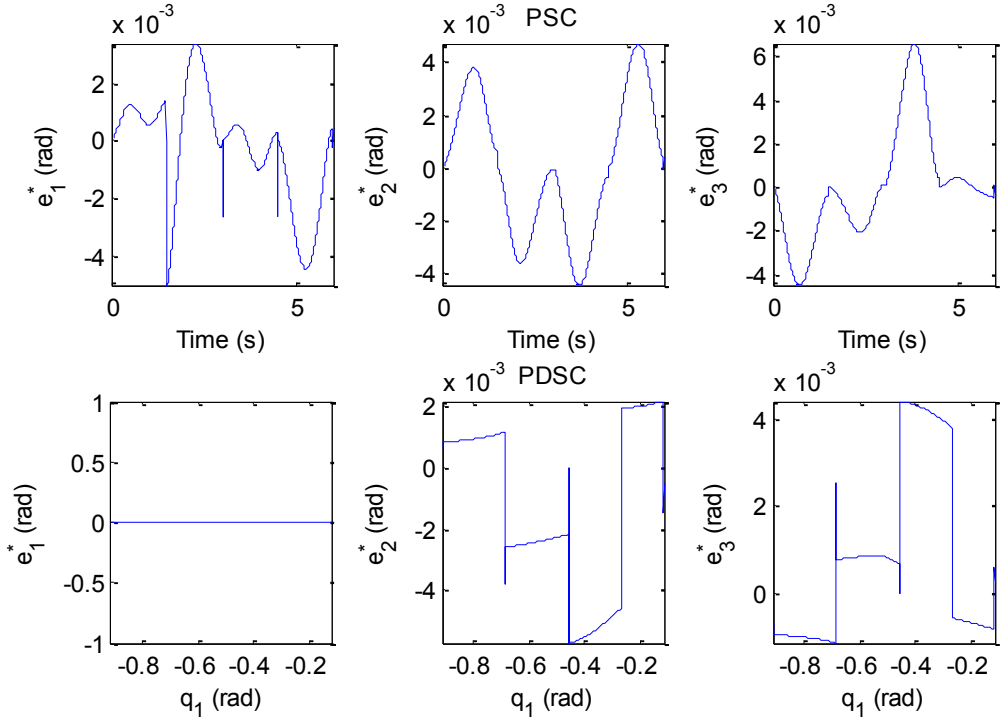

Figure 4-30: Diamond Motion Coupled Error for PSC and PDSC

Once again the position domain controllers performed better than their time domain counterparts, featuring mean errors and standard deviations an order of magnitude lower than the time domain as shown in Figure 4-31. Furthermore, the PDSC controller produced 25.6 % lower mean error than the PD-PD controller with the standard deviation being also lower by 33.4% as it is indicated in Table 4-19. The contour error results of the position domain controllers can be seen more clearly in Figure 4-32.

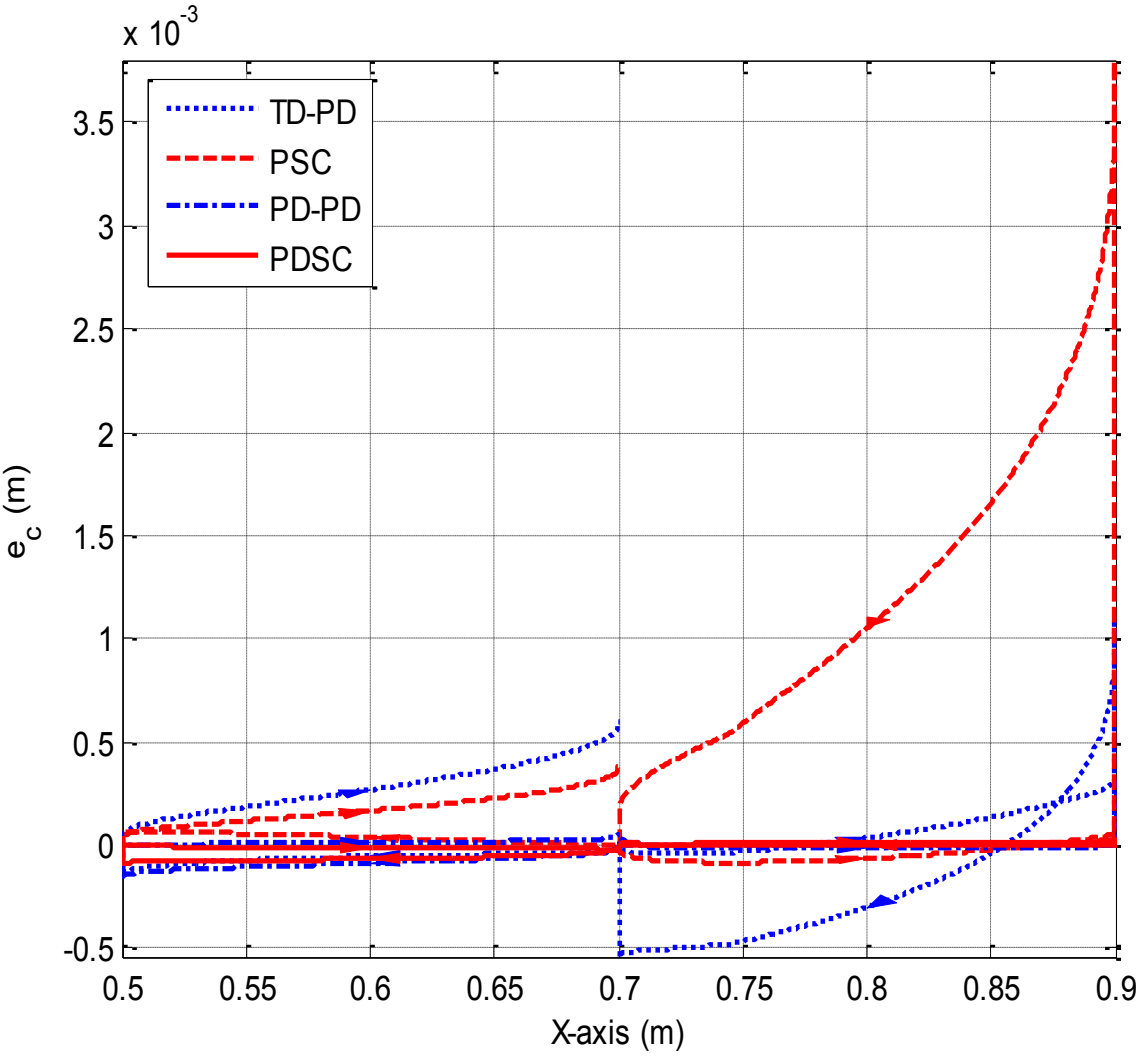

Figure 4-31: Diamond Motion Contour Tracking Performance for TD-PD, PSC, PD-PD and PDSC

Table 4-19: Mean and Standard Deviation for Diamond Contour[2]

	Mean (m)	S.D. (m)
TD-PD	0.00022	0.00021
PSC	0.00041	0.00078
PD-PD	$2.919\ 10^{-5}$	$3.974\ 10^{-5}$
PDSC	$2.172\ 10^{-5}$	$2.646\ 10^{-5}$

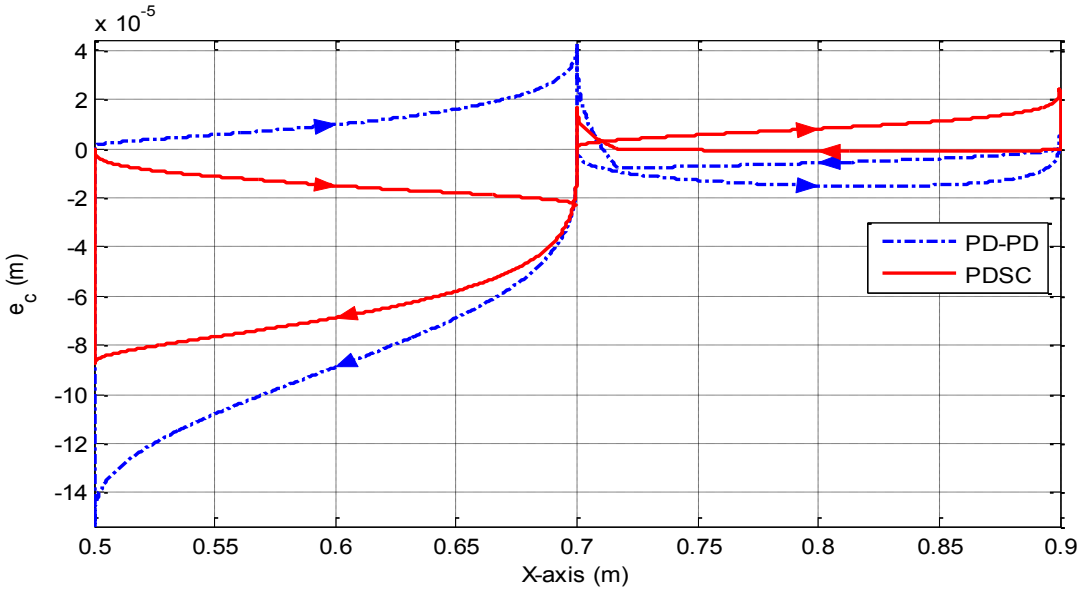

Figure 4-32: Contour Tracking Error for PD-PD and PDSC

As exhibited in Table 4-20, the torques of the position domain controllers were once again different than the ones of the time domain. This time the position domain controllers produced lower torque from the time domain by approximately 50% for the second axis while the difference between the third axis torques is approximately 40%.

Table 4-20: Maximum Torque for Diamond Contour (in Nm)

	q_1	q_2	q_3
TD-PD	22.868	18.048	9.197
PSC	22.953	18.53	9.187
PD-PD	22.868	9.198	5.445
PDSC	22.868	9.197	5.445

[2] Contour error calculated with Eq. (2-14)

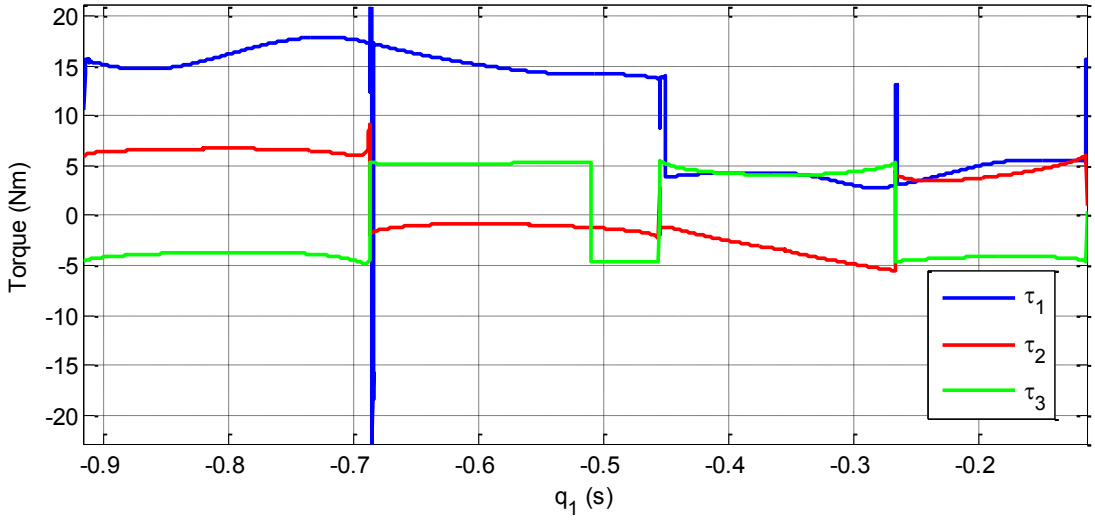

Figure 4-33: Diamond Contour Torques for PDSC Controller

4.8 Circular Contour Tracking Control

4.8.1 Circular Contour

The circular contour simulations also yielded good tracking error results. On the second axis TD-PD produced 49% lower mean error than PDSC, the second lowest and on the third axis PSC had the lowest error with PDSC coming second. Similar comments can be made for the standard deviation of the tracking errors of the controllers which are included in Table 4-21. The PSC controller was proven to produce the more stable tracking error having10.8% and 32% lower standard deviation for the second and third axis when compared with the TD-PD controller which was second best.

On the contrary, the PDSC controller proves more efficient in the synchronization errors since its mean error and their standard deviations are lower than the PSC's by 4% and 47% for the second and third axis respectively. This trend continued to the coupled error results in Table

4-21 and Figure 4-36 where the mean error and standard deviation of PDSC for the third axis is lower than the ones of the PSC controller by 46.5% but for the second axis the PSC controller featured10% less error deviation.

Table 4-21: Mean and Standard Deviation of Axial Tracking Error for Circular Contour

	Axis 1		Axis 2		Axis 3	
	Mean (rad)	S.D. (rad)	Mean (rad)	S.D. (rad)	Mean (rad)	S.D. (rad)
TD-PD	0.00059	0.00053	$8.133\ 10^{-5}$	$8.163\ 10^{-5}$	0.00046	0.00012
PSC	0.00018	0.00010	0.00016	$7.281\ 10^{-5}$	0.00017	$8.117\ 10^{-5}$
PD-PD	N/A	N/A	0.00030	0.00018	0.00043	0.00016
PDSC	N/A	N/A	0.00016	0.00015	0.00018	0.00014

Table 4-22: Mean and Standard Deviation of Synchronization and Coupled Error for Circular contour

	Axis 1		Axis 2		Axis 3	
	Mean (rad)	S.D. (rad)	Mean (rad)	S.D. (rad)	Mean (rad)	S.D. (rad)
Synchronization Error (ε)						
PSC	0.00044	0.00055	0.00044	0.00033	0.00082	0.00059
PDSC	N/A	N/A	0.00042	0.00037	0.00042	0.00018
Coupled Error (e^*)						
PSC	0.00605	0.00764	0.00594	0.00454	0.01102	0.00785
PDSC	N/A	N/A	0.00573	0.00506	0.00590	0.00249

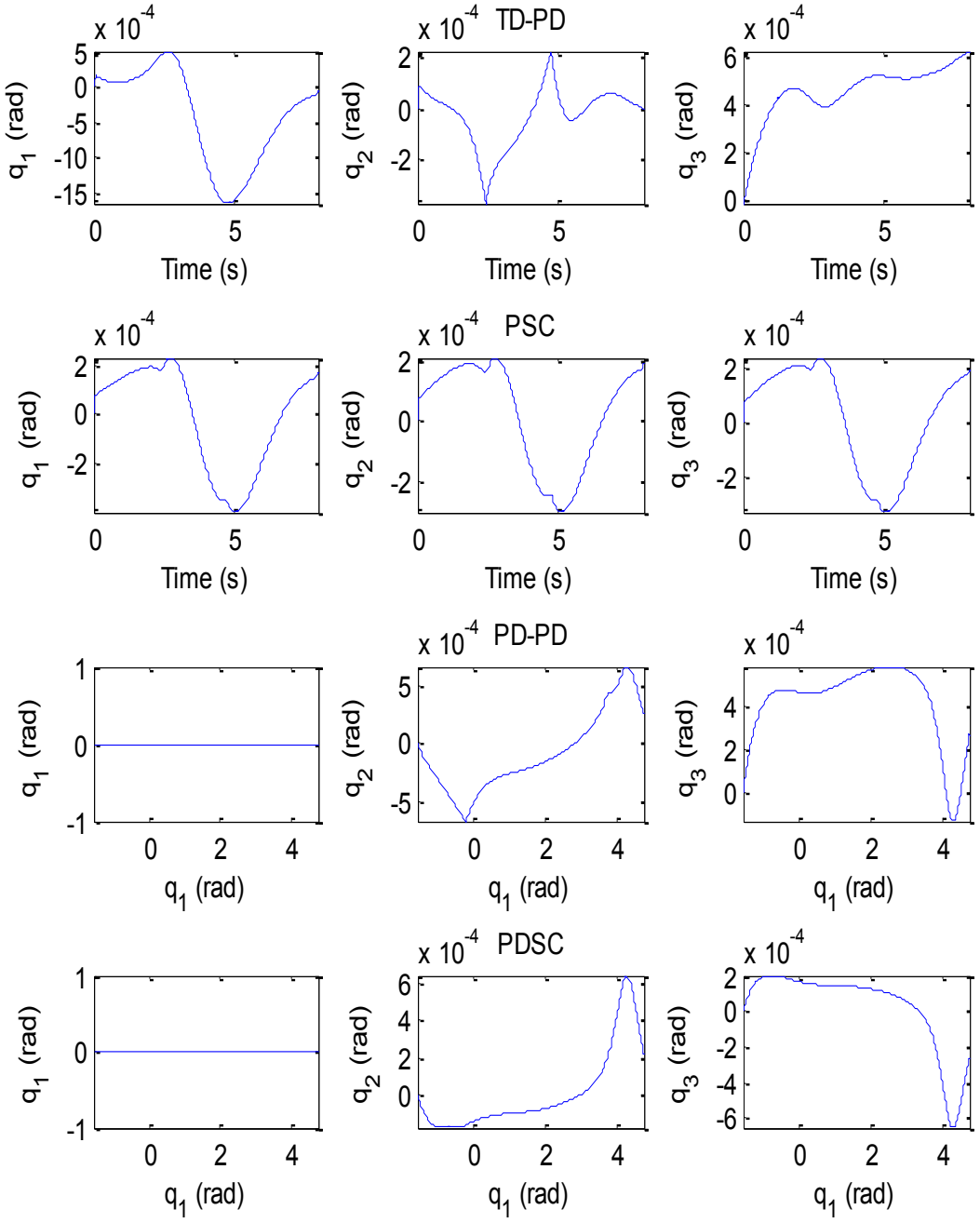

Figure 4-34: Axial Tracking Error of Circular Contour for TD-PD, PSC, PD-PD, PDSC

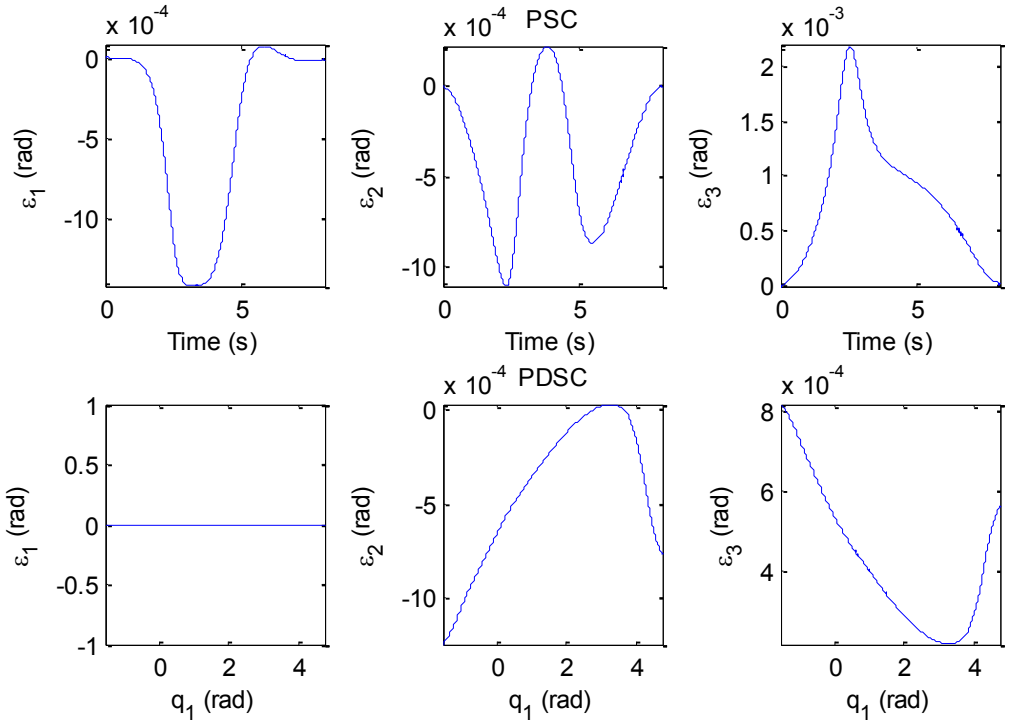

Figure 4-35: Circular Contour Synchronization Error for PSC and and PDSC

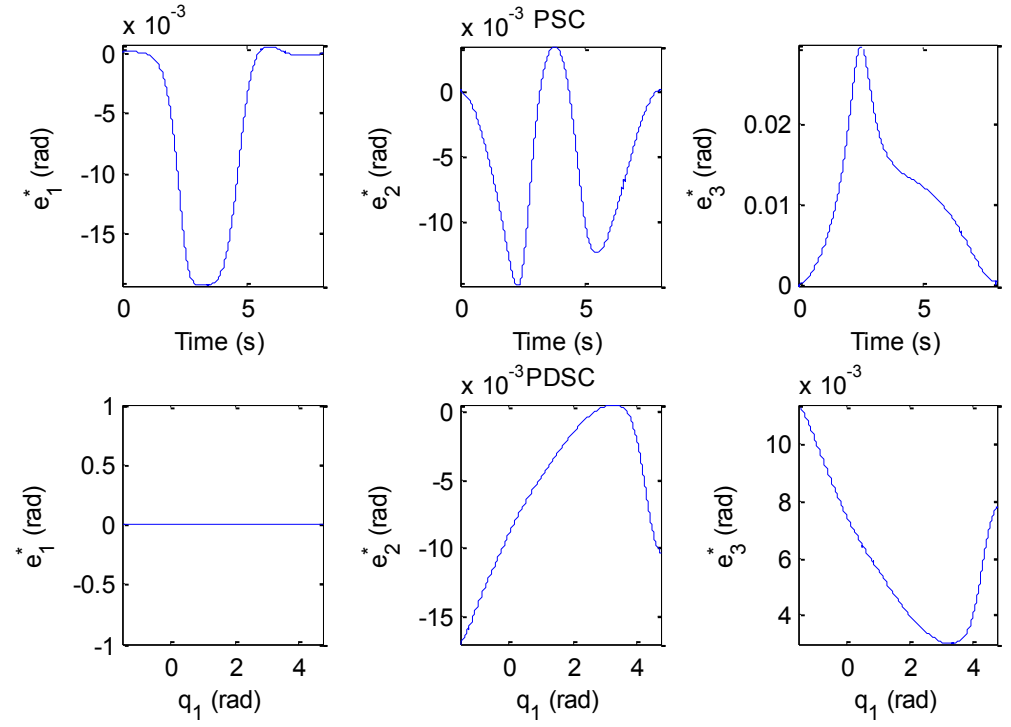

Figure 4-36: Circular Contour Coupled Error for PSC and PDSC

As presented in Figure 4-37, the PSC and PDSC controllers generated lower mean contour error than the rest of the controllers. Nonetheless, PDSC yielded 32% lower mean contour error than PSC while having a 68.5% difference from the mean error of PD-PD which was the lowest performer. Additionally, the standard deviation of the error was equally small for all the controllers as shown in Table 4-23, but the PDSC controller yielded 52.9% lower deviation than the TD-TD controller which was the second best in that category.

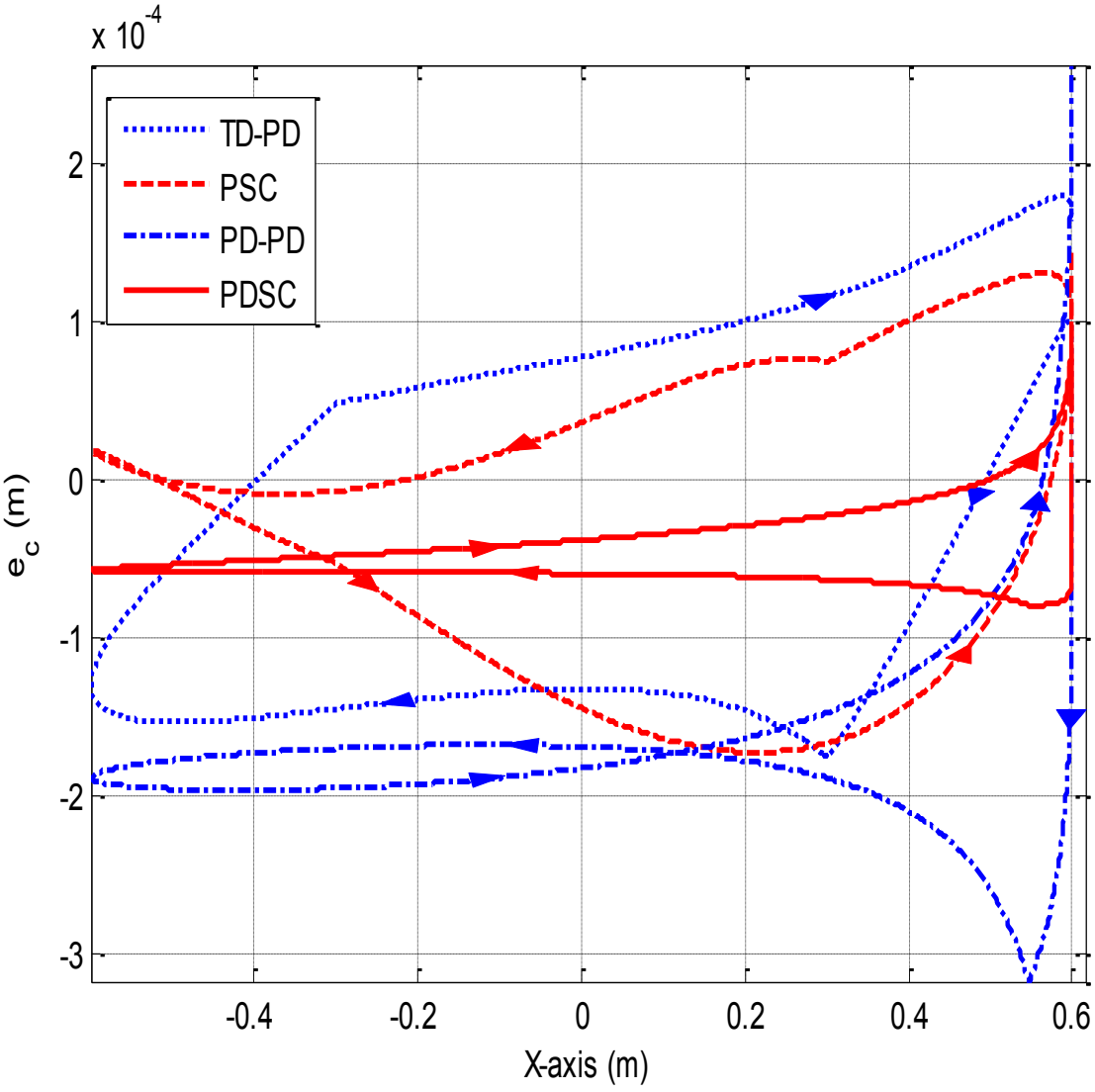

Figure 4-37: Circular Motion Contour Tracking Performance for TD-PD, PSC, PD-PD and PDSC

Table 4-23: Mean and Standard Deviation of Contour Tracking Error or Circular Contour[3]

	Mean (m)	S.D. (m)
TD-PD	0.00012	$4.861\ 10^{-5}$
PSC	$7.90\ 10^{-5}$	$5.010\ 10^{-5}$
PD-PD	0.00017	$7.212\ 10^{-5}$
PDSC	$5.363\ 10^{-5}$	$2.290\ 10^{-5}$

As with the other cases, the torques of the slave motions on the position domain were different from the ones in the time domain. As presented in Table 4-24 this time the PD-PD and PDSC controllers produced 15.8% and 23% lower torques in the second axis than the time domain controllers while the differences in the third axis torque are negligible.

Table 4-24: Maximum Torque for Circular Contour (in Nm)

	q_1	q_2	q_3
TD-PD	14.560	9.791	5.399
PSC	14.563	9.678	5.401
PD-PD	14.560	8.241	5.365
PDSC	14.560	7.541	5.365

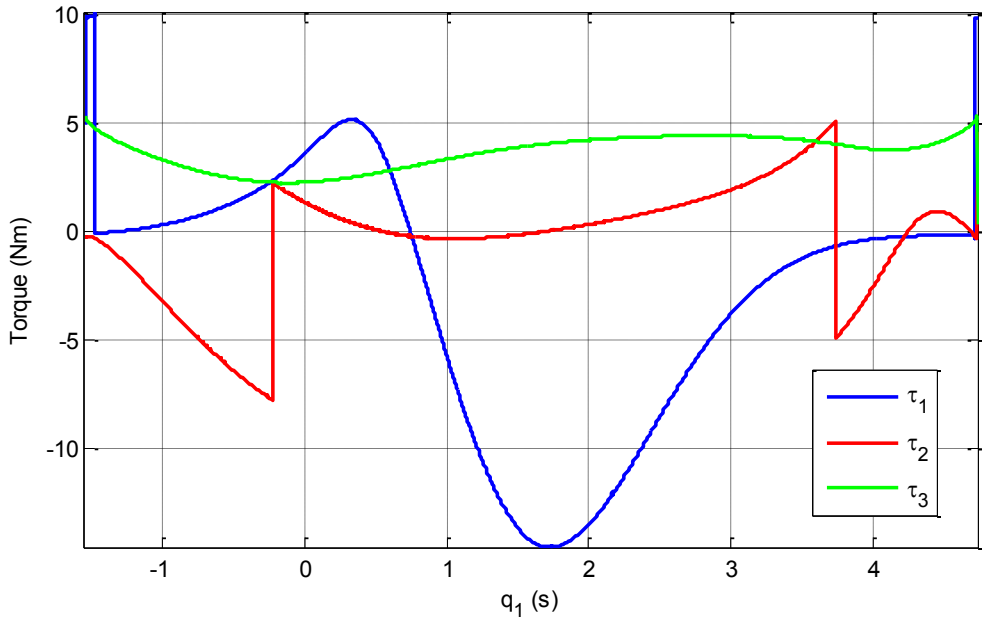

Figure 4-38: Circular Contour Torques for PDSC Controller

[3] Contour error calculated with Eq (2-15)

4.8.2 Epitrochoidal Contour

Interestingly, the time domain controllers, TD-PD and PSC, performed better than the position domain controllers. Particularly, the TD-PD controller featured 18% lower tracking error than the PDSC for the second axis, and the PSC controller achieved 14% lower error than the PDSC for the third axis. Similarly, the standard deviation of TD-PD for the second axis was 53% lower than the standard deviation of PDSC with the equivalent standard deviation for the PSC being 63% lower. The rest of the tracking performance can be seen in Table 4-25 and Figure 4-39.

In terms of the synchronization and coupled errors, the PDSC controller outperformed the PSC controller. For the second axis, the mean synchronization and coupled error were approximately 27% lower for the PDSC than the PSC controllers. Similarly, the third axis synchronization and coupled error for PDSC were approximately 56% lower than the errors of the PSC. As presented in Table 4-26, the standard deviations of the errors followed the same trend, with the PDSC providing a more stable error performance in synchronization. As an indication, the standard deviation of the synchronization error for the third axis was 89% lower for the PDSC controller than the PSC.

Table 4-25: Mean and Standard Deviation of Axial Tracking Error for Epitrochoidal Contour

	Axis 1		**Axis 2**		**Axis 3**	
	Mean (rad)	S.D. (rad)	Mean (rad)	S.D. (rad)	Mean (rad)	S.D. (rad)
TD-PD	0.00031	0.00022	0.00023	0.00017	0.00026	$0.8087\ 10^{-5}$
PSC	0.00025	0.00013	0.00024	0.00013	0.0024	0.00013
PD-PD	N/A	N/A	0.00064	0.00052	0.00030	0.00033
PDSC	N/A	N/A	0.00028	0.00036	0.00028	0.00037

Table 4-26: Mean and Standard Deviation of Synchronization and Coupled Error for Epitrochoidal contour

	Axis 1		Axis 2		Axis 3	
	Mean (rad)	S.D. (rad)	Mean (rad)	S.D. (rad)	Mean (rad)	S.D. (rad)
Synchronization Error (ε)						
PSC	0.00083	0.00092	0.00084	0.00097	0.00156	0.00148
PDSC	N/A	N/A	0.00061	0.00036	0.00069	0.00016
Coupled Error (e^*)						
PSC	0.01149	0.01272	0.01133	0.01315	0.02121	0.01996
PDSC	N/A	N/A	0.00812	0.00498	0.00962	0.00219

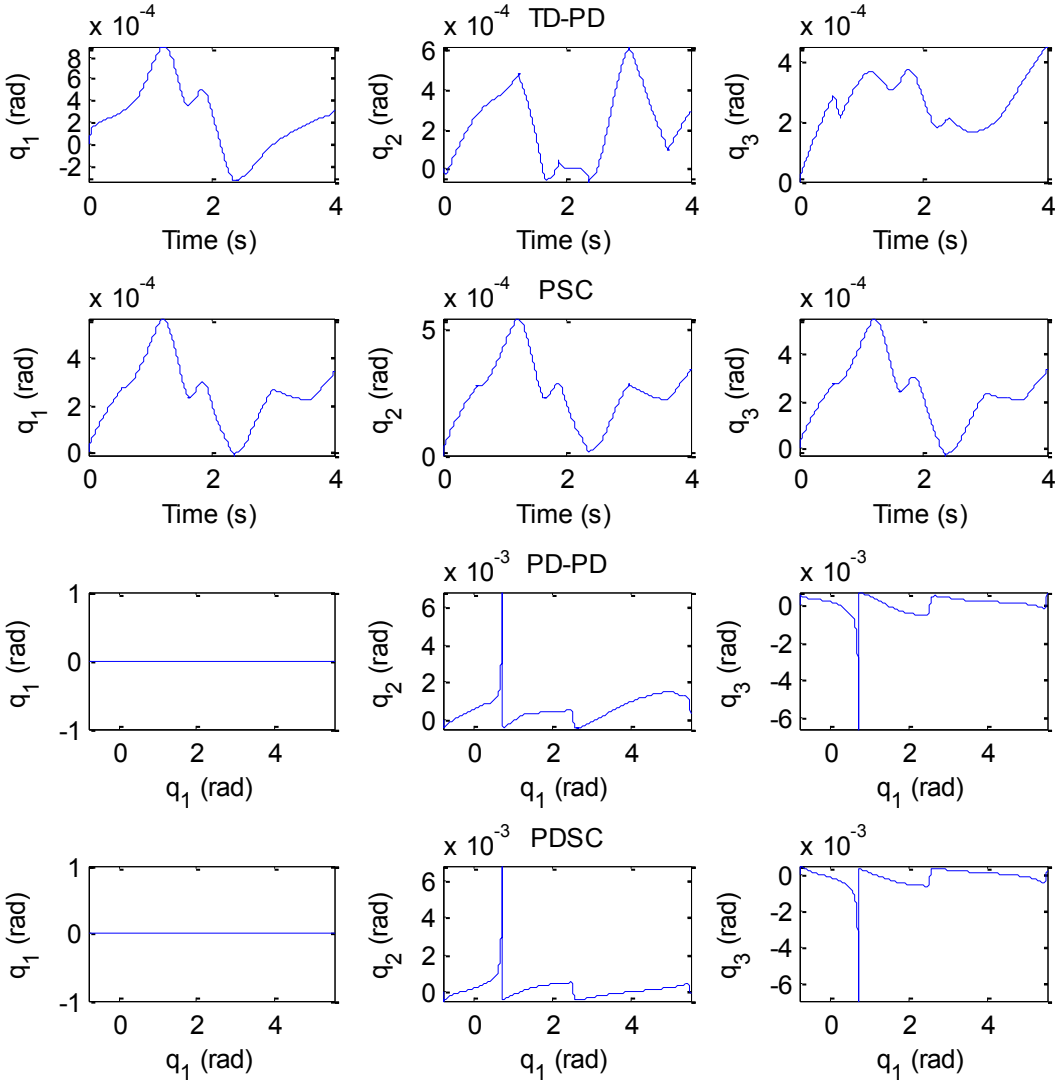

Figure 4-39: Axial Tracking Error of Epitrochoidal Contour for TD-PD, PSC, PD-PD, PDSC

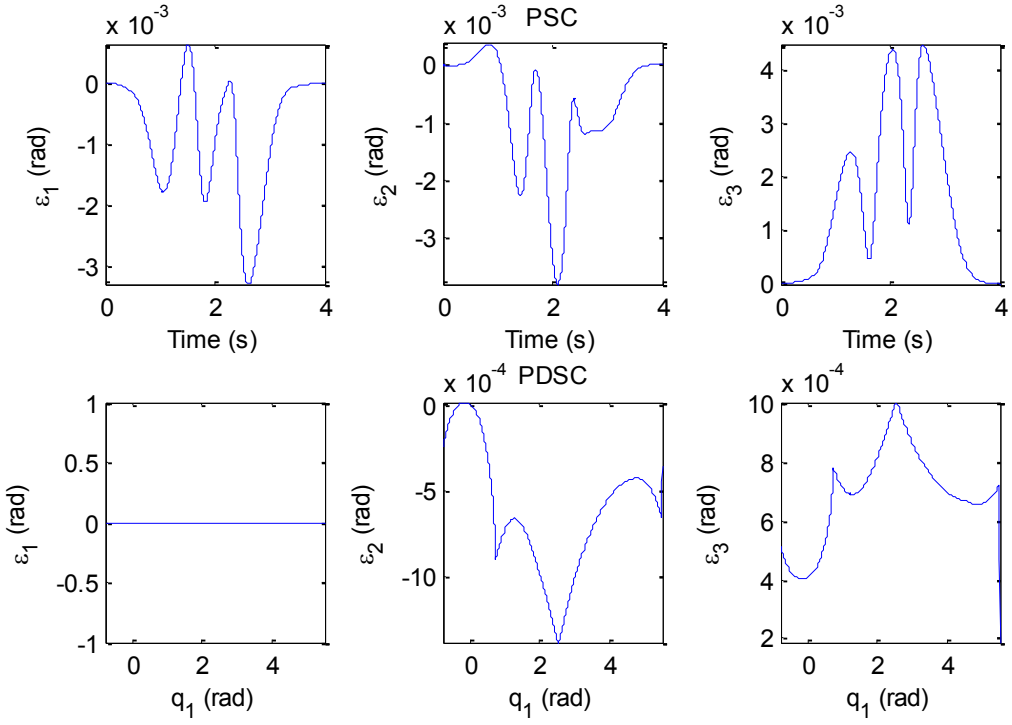

Figure 4-40: Epitrochoidal Contour Synchronization Error for PSC and and PDSC

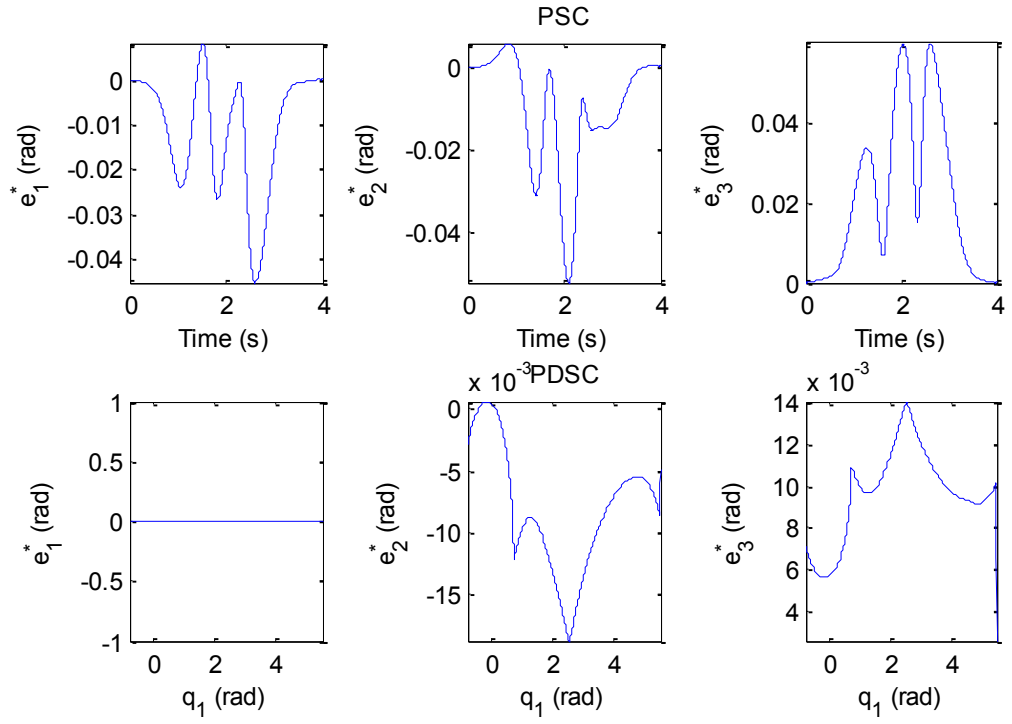

Figure 4-41: Epitrochoidal Contour Coupled Error for PSC and PDSC

Despite the PDSC controller being overtaken by the other controllers on the tracking error, Figure 4-42 shows that it still ouperformed the other controllers in terms of contour tracking. In particular the contour error of PDSC was 52.4% lower than the error of the second best controller, PSC. Comparing it with the lower contour error performer, the PDSC produced 70.6% lower mean error than the PD controller. The standard deviation follows idential trends as it can be seen in Table 4-27. The contour error results of the position domain controllers more clearly presented in Figure 4-43.

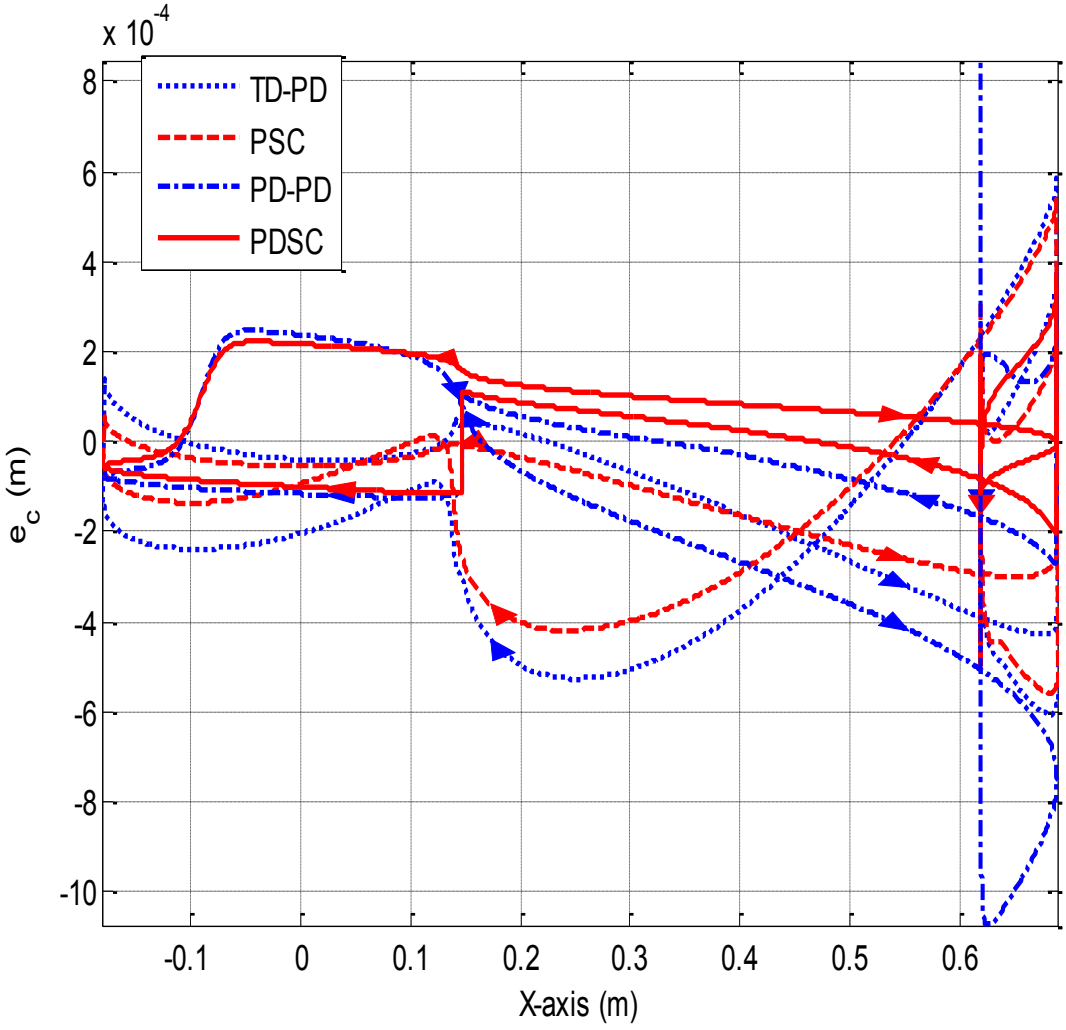

Figure 4-42: Epitrochoidal Motion Contour Tracking Performance for TD-PD, PSC, PD-PD and PDSC

Table 4-27: Mean and Standard Deviation of Contour Tracking Error of Epitrochoidal Contour[4]

	Mean (m)	S.D. (m)
TD-PD	0.00024	0.00018
PSC	0.00021	0.00017
PD-PD	0.00034	0.00034
PDSC	$9.984\ 10^{-5}$	$6.243\ 10^{-5}$

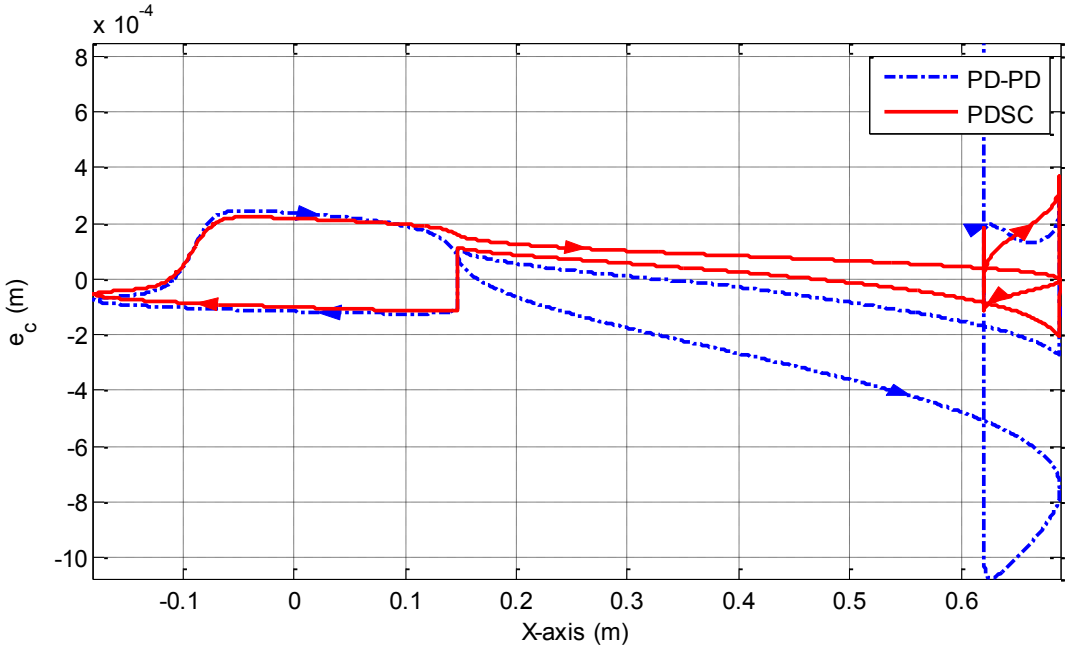

Figure 4-43: Contour Tracking Error for PD-PD and PDSC

In this case, the position domain trajectories led to similar second axis for all the controllers simulated. Still, Table 4-28 shows that the third axis maximum torque was increased by 70% for the position domain.

Table 4-28: Maximum Torque for Epitrochoidal Contour (in Nm)

	q_1	q_2	q_3
TD-PD	12.863	13.289	5.870
PSC	12.683	13.293	5.870
PD-PD	12.863	13.607	9.841
PDSC	12.863	13.588	9.839

[4] Contour Calculated with method explained in Section 2.4.3

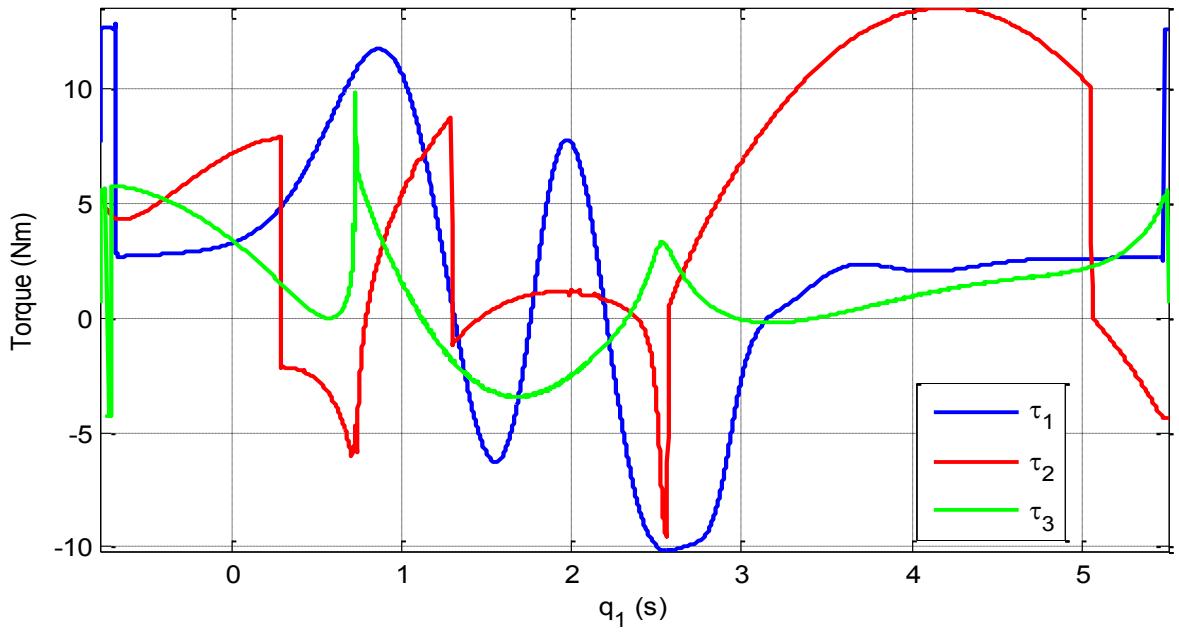

Figure 4-44: Epitrochoidal Contour Torques (in Nm)

4.9 Remarks

The simulations results demonstrate a general improvement of performance for the tracking and contour tracking error of the proposed PDSC control law. For linear contours, the position domain controllers yielded 50% to 90% lower mean contour error when compared to the time domain controllers. For the same contours the standard deviation of the error was improved by the position domain control in an amount ranging from 40% to 80%. Figure 4-45 indicates also that the PDSC controller produced 35.8% and 25.6% lower mean error than the PD-PD controller for the zigzag and diamond contours, respectively. The standard deviation was also improved by approximately the same percentages. The lower contour performance of the controller in the zigzag contour, when compared with the diamond motion, can be attributed to the lower time allowed for the motion to be performed, which led to higher trajectories and therefore higher required torque (also see Table 4-16 and Table 4-20).

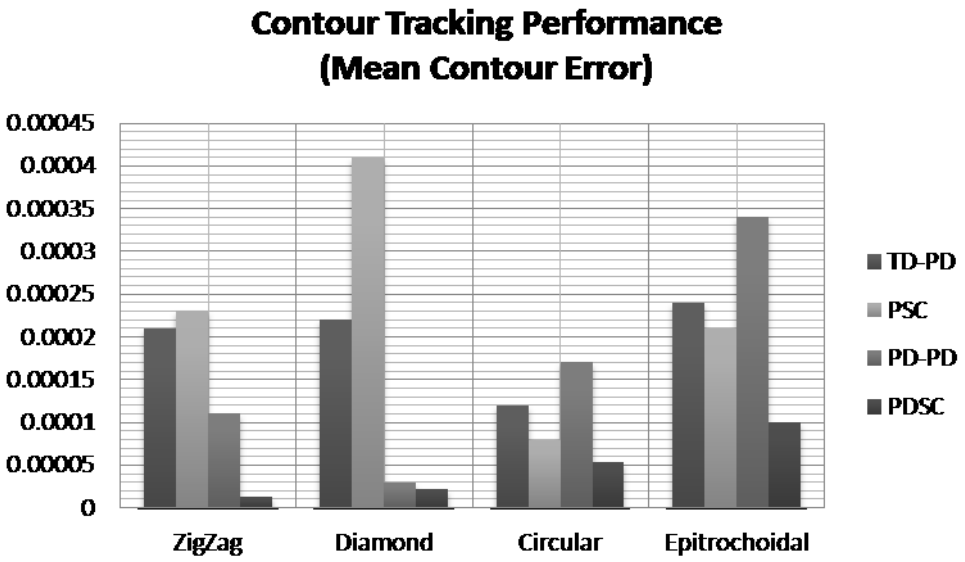

Figure 4-45: Contour Tracking Performance Summary

For the circular contours, the controllers employing the synchronization principle achieved higher contour tracking performance than the independent PD controllers. With synchronization the contour error was improved by 32% to 70% with the standard deviation following similar improvement. The superior performance of the synchronization controllers can be explained by considering the nonlinearity of the contours and the robotic manipulator which also led to varying sampling distance for the position domain controllers. The innate coordination of the agents that is achieved with the synchronization controllers compensates the nonlinearity of the motion in a way which the independent controllers cannot do. Between the two synchronization controllers, PDSC yielded 32.1% and 52.5% lower mean contour error than the PSC for the circular and epitrochoidal contours with the standard deviation being improved by 54.3% and 63.3% correspondingly.

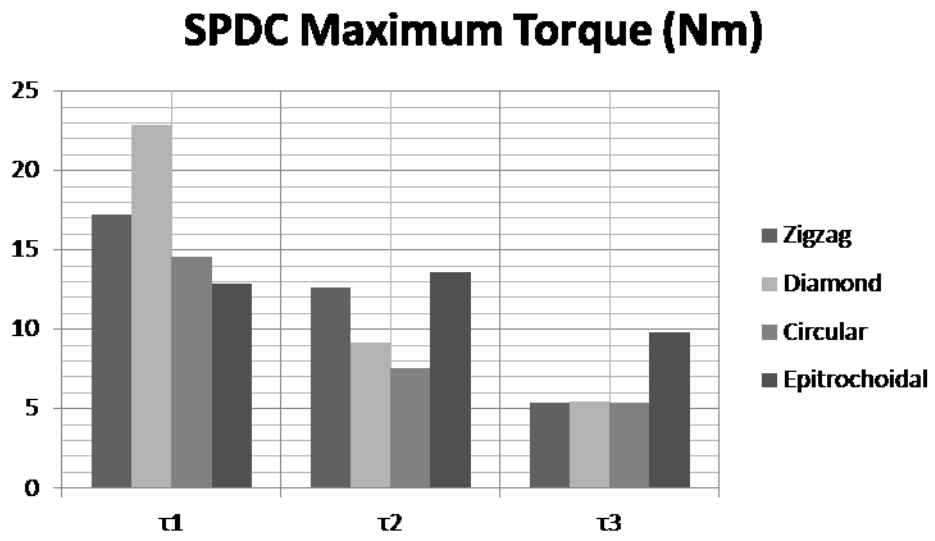

Figure 4-46: SPDC Maximum Torque

Figure 4-46 presents the maximum torque values of the robotic manipulators axes, as produced by the PDSC controller for all the simulated contours. Since the maximum torques of the slave axes (τ_2 and τ_3) does not exceed the maximum values of τ_1 it can be easily deduced that the proposed controller comes with the similar power requirements as conventional time domain controllers.

Chapter 5: Conclusions and Future Work

5.1 General Review

Robotic manipulators have many applications from electrical and electronics production to surgical procedures and space explorations. An important research area for robotics is the development of advanced control systems for the purpose of improving tracking or contour performance, and that is the main goal of this research presented in this thesis.

In this thesis, a new model-free position domain synchronization control law was proposed for the control of multi-DOF robotic manipulators. The new control law uses the principles of the position domain and synchronization to improve the contour tracking performance in the end-effector level. In position domain, the dynamics of the system are transformed from time domain to position domain via a one–to-one mapping and one the system's axis is used as the reference instead of time for the slave axes. The addition of the synchronization principles incorporates the synchronization of each slave axis' error in order to further coordinate the motion of the whole system. The stability of the proposed controller was established with the Lyapunov method where the error was proven to be bounded.

The new controller was used for a planar RRR robotic manipulator and tested via linear and nonlinear contour simulations. The new controller's performance was also compared with the performances of time domain PD, time domain PSC, and position domain PD controllers. In terms of tracking error, the newly introduced controller was shown to be on par with the other controllers and in the case of linear contours it produced mean errors and standard deviation of one magnitude lower. Similarly, the mean and standard deviation of the synchronization and

coupled errors of the new controller were frequently lower than the ones of its time domain counterpart.

Regardless of the tracking error results, the contour tracking performance of the new control law was consistently superior to the other controllers with the mean contour error and standard deviation of the PDSC controller steadily an order of magnitude lower than the contour errors of the rest of the controllers for all the contour cases.

5.2 Main Contributions

A number of the contributions are made in this thesis:

- Position domain was introduced as a control alternative for nonlinear robotic systems
- The principle of synchronization was introduced to position domain controllers allowing for better coordination of the slave systems.
- A new and simple control law for nonlinear system was introduced which is able to provide high contour tracking performance. The new control is not limited to robotic manipulators or CNC machines but can also be used for robotic swarms, multi-agent systems and any other configuration that required high coordination.
- Better contour tracking performance can be obtained by using the developed position domain synchronization control.

5.3 Future Work

Although this thesis has successfully introduced the notion of position domain synchronization control, more work can be done the further improve and expand the new control

concept. First of all, experiments should be performed to validate the simulations and study any unforeseen difficulties. The controller should also be used in different robot configurations (parallel, spatial, etc) as well as robot swarms and coordinated robotic groups. Lastly, position domain controllers use different trajectories than their time domain counterparts. Therefore, a path planning optimization technique should be developed in order to define position domain trajectories that would yield the best performances for the position domain controller.

Bibliography

Alford, C., & Belyeu, S. (1984). Coordinated control of two robot arms. *Proceedings of IEEE International Conference in Robotics and Automation*, 468-473.

Alliance Communications Inc. (n.d.). *Lean Arc Welding Lowers Cost per Weld*. Retrieved March 16, 2013, from Fabricating and Metal Working: http://www.fabricatingandmetalworking.com/2012/11/lean-arc-welding-lowers-cost-per-weld/

Arendt, W., & Schleuch, W. P. (2009). *Mathematical Analysis of Evolution, Information and Complexity*. Wiley-VCH.

Arimoto, S., Miyazaki, F., & Kawamura, S. (1987). Cooperative Motion Control of Multiple Robot Arms or Fingers. *Proceedings of IEEE International Conference on Robotics and Automation*, 1407-1412.

Armstrong-Helouvry, B., Dupont, P., & Canudas De Wit, C. (1994). A Survey of Models, Analysis Tools and Compensation Methods for the Control of Machines with Friction. *Automatica*, 1083-1138.

Arzen, K. E. (1999). A Simple Event-Based PID Controller. *Procceedings of 14th IFAC World Congress*, 423-428.

Astrom, K. J. (2007). Event Based Control. *Analysis and Design of Nonlinear Control Systems: In Honor of Alberto Isidori*, 127-147.

Astrom, K. J., & Hagglund, T. (2001). The Future of PID Control. *Control Engineering Practice*, 1163-1175.

Bansal , H. O., Sharma, R., & Shreeraman, P. R. (2012). PID Controller Tuning Techniques: A Review. *Journal of Control Engineering and Technology*, 168-176.

Bartolini, G., Pisano, A., Punta, E., & Usai, E. (2003). A Survey of Applications of Second-Order Sliding Mode Control to Mechanical Systems. *International Journal of Control*, 875-892.

Bayliss, Jones and Bayliss. (n.d.). Retrieved March 16, 2012, from History Website: http://www.historywebsite.co.uk/Museum/OtherTrades/BCN/BJB.htm

Bristow, D. A., Tharayil, M., & Alleyne, A. G. (2006). A Survey o Iterative Learning Control. *IEEE Control Systems*, 96-114.

Cheng, M. L. (2007). Motion Controller Design for Contour-Following Tasks Based on Real-Time Contour Error Estimation. *IEEE Transactions on Industrial Electronics*, 1686-1695.

Cominos, P., & Munro, N. (2002). PID controllers: Recent Tuning Methods and Design to Specification. *Control Theory and Applications, IEEE Proceedings* .

Craig, J. (2005). *Introduction to Robotis: Mechanics and Control.* Upper Saddle River: Peason Prentice Hall.

Dam, T., & Ouyang, P. R. (2012). Position Domain Controur Tracking with Cross-Coupled Control. *21st IEEE International Symposium on Industrial Electronics*, 1303-1308.

Dean, T. (2002, May). *Sensors and Sensing.* Retrieved February 22, 2012, from http://cs.brown.edu: http://cs.brown.edu/~tld/courses/cs148/02/sensors.html

Dombre, E. K. (2007). *Robot Manipulators: Modeling, Performance Analysis and Control.* Newport Beach: ISTE USA.

Dombre, E., & Khalil, W. (2006). *Robot Manipulators: Modeling, Performance Analysis and Control.* Newport Beach: Wiley Publishing.

Hsu, M. H., Yan, H. S., Liu, J. Y., & Hsieh, L. C. (2008). Epicycloid (Hypocycloid) Mechanisms Design. *Proceedings of the International MultiConference of Engineers and Computer Scientics, 2*.

Inoue, T., Nakano, M., Matsumoto, S., & Baba, H. (1981). High Accuracy Contour Control of Proton Synchroton Magnet Power Supply. *Proceeding of the 8th World Conference of IFAC*, 216-221.

International Federation of Robotics. (2012). *History of Industrial Robots.* Retrieved February 9, 2012, from International Federation of Robotics: http://www.ifr.org/fileadmin/user_upload/downloads/forms___info/History_of_Industrial_Robots_online_brochure_by_IFR_2012.pdf

Kelly, R. (1997). PD Control with Desired Gravity Compensation of Robotic Manipulators: A Review. *The International Journal of Robotics Research*, 660-672.

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. *Proceedings of IEEE International Conference in Neural Networks*, 1942-1948.

Koren, Y. (1980). Cross-Coupled Biaxial Computer Control for Manufacturing Systems . *ASME Journal of Dynamic Systems, Measurement, and Control*, 265-272.

Koren, Y., & Lo, C.-C. (1991). Variable-Gain Cross-Coupling Controller for Contouring. *ASME Journal of Dynamic Systems, Measurement, and Control, 40*(1), 265-272.

Kuc, T. Y., & Han, W. G. (2000). An Adaptive PID Learning Control of Robot Manipulators . *Automatica*, 717-725.

Moore, K. L., Chen, Y., & Ahn, H. S. (2006). Iterative Learning Control: A Tutorial and Big Picture View. *45th IEEE Converence on Decision and Control*, 2352-2357.

Ouyang, P. R., & Dam, T. (2011). Position Domain PD Control: Stability and Comparison. *The 6th IEEE Conference on Industrial Electronics and Applications*, 962-976.

Ouyang, P. R., Dam, T., Huang, J., & Zhang, W. J. (2012). Contour Tracking Control In Position Domain. *Mechatronics*, 934-944.

Ouyang, P. R., Huang, J., & Zhang, W. J. (2011). Position Domain Contour Control for Robotic System. *6th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, 962-967.

Ouyang, P. R., Pano, V., & Dam, T. (2012). PID Contour Tracking Control in Position Domain. *21st IEEE International Symposium on Industrial Electronics*, 1297-1302.

Ramesh, R. M. (2005). Tracking and Contour Error Control in CNC Servo Systems . *International Journal of Machine Tools and Manufacture*, 301-326.

Rao, R. S. (2009). *Engineering Optimization: Theory and Practice.* New Jersey: John Wiley & Sons, Inc.

Rocco, P. (1996). Stability o PID Control for Industrial Robot Arms. *IEEE Transactions on Robotics and Automation* , 606-614.

Rodriguez-Angeles, A., & Nijmeijer, H. (2001). Coordination of two robot manipulators based on position measurements only. *International Journal of Control*, 1311-1323.

Sabanovic, A., Fridman, M., & Spurgeon, S. (2004). *Variable Structure Systems, from principles to implementation.* London, UK: The Institution of Engineering and Technology.

Shan, J., Liu, H. T., & Notwotny, S. (2005). Synchronized Trajectory-Tracking Control of Multipled 3-DOF Experimental Helicopters. *IEEE Proccedings in Control Theory and Applications*, 683-692.

Slotine, J. E., & Li, W. (1991). *Applied Nonlinear Control.* New Jersey: Prentice Hall.

Slotine, J. W. (1988). Adaptive Manipulator Control: A Case Study. *IEEE Transactions on Automatic Control* , 995-1003.

Sun, D. (2003). Position Synchronization of Multiple Motion Axes with Adaptive Coupling Control. *Automatica, 15*(5), 997-1005.

Sun, D. (2011). *Synchronization and Control of Multiagent Systems.* Boca Raton, FL: CRC Press.

Sun, D., & Tong, M. C. (2009). A Synchronization Approach for the Minimization of Contouring Error of CNC Machine Tools. *IEEE Transactions on Automation Science and Engineering*, 720-729.

Sun, D., Shao, X., & Feng, G. (2007). A Model-Free Cross-Coupled Control for Position Synchronization of Multi-Axis Motions: Theory and Experiment. *IEEE Transactions on Control Systems Technology*, 306-314.

Takegaki, M., & Arimoto, S. (1981). A new Feedback Method for Dynamic Control of Manipulators. *ASME Journal of Dynamical Systems, Measurements and Control*, 119-125.

The MathWorks, Inc. (n.d.). *MathWorks Documentation Centre.* Retrieved March 13, 2013, from www.mathworks.com: http://www.mathworks.com/help/matlab/ref/interp1.html?searchHighlight=interp1

Wallen, J. (2008). *The History of the Industrial Robot.* Linkoping, Sweden: Linkopings Universitet: Department of Electrical Engineering.

Wang, Y., Gao, F., & Doyle, F. J. (2009). Survey on Iterative Learning Control, Repetitive control, and Run-to-Run Control. *Journal of Process Control*, 1589-1600.

WINTHROP Inc. (n.d.). *The DaVinci Si HD Surgical System at Winthrop-University Hospital*. Retrieved March 16, 2013, from Winthrop University Hospital: https://www.winthrop.org/departments/institutes/family/ob-gyn/robotic-surgery/daVinci-Si-HD-Surgical-System/

Yee, L. L. (2011, August 1). *Foxconn to rely more on robots; could use 1 million in 3 years*. Retrieved February 9, 2012, from www.reuters.com: http://www.reuters.com/article/2011/08/01/us-foxconn-robots-idUSTRE77016B20110801

Yeh, S. H. (2003, May). Yeh, S. S., & Hsu, P. L. (2003). Analysis and design of integrated control for multi-axis motion systems. Control Systems Technology, IEEE Transactions on, 11(3), 375-382. *IEEE Transactions on Control Systems Technology*, 375-382.

Yokokohji, Y., & Yoshikawa, T. (1994). Bilaterl Control of Master-Slave Manipulators for Ideal Kinesthetic Coupling-Formulation and Experiment. *IEEE Transactions on Robotics and Automation*, 605-620.

Zhao, D., Li, S., Gao, F., & Zhu, Q. (2008). Robust Adaptive Terminal Sliding-Based Synchronised Position Control For Multiple Motion Axes Systems. *IET Control Theory and Applications*, 136-150.

Appendix

3R Dynamics and Kinematics

Forward Kinematics

$$x_{ef} = l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) + l_3 \cos(q_1 + q_2 + q_3) \tag{1}$$

$$y_{ef} = l_1 \sin(q_1) + l_2 \sin(q_1 + q_2) + l_3 \sin(q_1 + q_2 + q_3) \tag{2}$$

$$q_{ef}(t) = q_1(t) + q_2(t) + q_3(t) \tag{3}$$

Inverse Kinematics

Position:

$$x_2 = x_{ef} - l_3 \cos(q_{ef}) \tag{4}$$

$$y_2 = y_{ef} - l_3 \sin(q_{ef}) \tag{5}$$

$$\cos(q_2) = \frac{x_2^2 + y_2^2 - l_1^2 - l_2^2}{2 l_1 l_2} \tag{6}$$

$$\sin(q_2) = \sqrt{1 - (\cos(q_2))^2} \tag{7}$$

Then,

$$q_2 = atan2(\cos(q_2), \sin(q_2)) \tag{8}$$

$$q_1 = atan2(y_2, x_2) - atan2(l_1 \sin(q_2), l_1 + l_2 \cos(q_2)) \tag{9}$$

$$q_3 = q_{ef} - q_1 - q_2 \tag{10}$$

Velocity:

$$\dot{q}_1 = \frac{\cos(q_1+q_2)}{l_1\sin(q_2)}\dot{x}_{ef} + \frac{\sin(q_1+q_2)}{l_1\sin(q_2)}\dot{y}_{ef} + \frac{l_3\sin(q_3)}{l_1\sin(q_2)}\dot{q}_{ef} \tag{11}$$

$$\dot{q}_2 = -\frac{l_2\cos(q_1+q_2)+l_1\cos(q_1)}{l_1 l_2\sin(q_2)}\dot{x}_{ef} - \frac{l_2\sin(q_1+q_2)+l_1\sin(q_1)}{l_1 l_2\sin(q_2)}\dot{y}_{ef} - \frac{l_3\sin(q_2+q_3)+l_3\cos(q_3)}{l_1 l_2\sin(q_2)}\dot{q}_{ef} \tag{12}$$

$$\dot{q}_2 = \frac{\cos(q_1)}{l_2\sin(q_2)}\dot{x}_{ef} + \frac{\sin(q_1)}{l_2\sin(q_2)}\dot{y}_{ef} + \left(\frac{l_3\sin(q_2+q_3)}{l_2\sin(q_2)}+1\right)\dot{q}_{ef} \tag{13}$$

Dynamics

Inertia Matrix

$$M_{11} = I_1 + I_2 + I_3 + m_1 r_1^2 + m_2(l_1^2 + r_2^2 + 2l_1 r_2\cos(q_2)) + m_3(l_1^2 + l_2^2 + r_3^2 + 2l_1 l_2\cos(q_2) + 2l_2 r_3\cos(q_3) + 2l_1 r_3\cos(q_2+q_3)) \tag{14}$$

$$M_{12} = M_{21} = I_2 + I_3 + m_2(r_2^2 + l_1 r_2\cos(q_2)) + m_3(l_2^2 + r_3^2 + l_1 l_2\cos(q_2) + 2l_2 r_3\cos(q_3) + l_1 r_3\cos(q_2+q_3)) \tag{15}$$

$$M_{13} = M_{31} = I_3 + m_3(r_3^2 + l_2 r_3\cos(q_3) + l_1 r_3\cos(q_2+q_3)) \tag{16}$$

$$M_{22} = I_2 + I_3 + m_2 r_2^2 + m_3(l_2^2 + r_3^2 + 2l_2 r_3\cos(q_3)) \tag{17}$$

$$M_{23} = M_{32} = I_3 + m_3(l_2^2 + r_3^2 + 2l_2 r_3\cos(q_3)) \tag{18}$$

$$M_{33} = I_3 + m_3 r_3^2 \tag{19}$$

Coriolis-Centrifugal Forces

Define,

$$C_{11} = \begin{bmatrix} 0 \\ m_2 l_1 r_2 \sin(q_2) - m_3(l_1 r_2 \sin(q_2) + l_1 r_3 \sin(q_2 + q_3)) \\ -m_3(l_1 r_2 \sin(q_2) + l_1 r_3 \sin(q_2 + q_3) \end{bmatrix}^T \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} \tag{20}$$

$$C_{12} = \begin{bmatrix} m_2 l_1 r_2 \sin(q_2) - m_3(l_1 r_2 \sin(q_2) + l_1 r_3 \sin(q_2 + q_3)) \\ -m_2 l_1 r_2 \sin(q_2) - m_3(l_1 l_2 \sin(q_2) + l_1 r_3 \sin(q_2 + q_3)) \\ m_3(l_2 r_3 \sin(q_3) + l_1 r_3 \sin(q_2 + q_3)) \end{bmatrix}^T \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} \tag{21}$$

$$C_{13} = \begin{bmatrix} -m_3(l_1 r_2 \sin(q_2) + l_1 r_3 \sin(q_2 + q_3) \\ m_3(l_2 r_3 \sin(q_3) + l_1 r_3 \sin(q_2 + q_3) \\ -m_3(l_2 r_3 \sin(q_3) + l_1 r_3 \sin(q_2 + q_3)) \end{bmatrix}^T \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} \tag{22}$$

$$C_{21} = \begin{bmatrix} m_3(l_2 r_3 \sin(q_3) + l_1 r_3 \sin(q_2 + q_3)) + m_2 l_1 r_2 \sin(q_2) \\ 0 \\ m_3 l_2 r_3 \sin(q_3) \end{bmatrix}^T \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} \tag{23}$$

$$C_{22} = \begin{bmatrix} 0 \\ 0 \\ -m_3 l_2 r_3 \sin(q_3) \end{bmatrix}^T \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} \tag{24}$$

$$C_{23} = \begin{bmatrix} m_3 l_2 r_3 \sin(q_3) \\ -m_3 l_2 r_3 \sin(q_3) \\ -m_3 l_2 r_3 \sin(q_3) \end{bmatrix}^T \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} \tag{25}$$

$$C_{31} = \begin{bmatrix} m_3(l_2 r_3 \sin(q_3) + l_1 r_3 \sin(q_2 + q_3)) \\ m_3 l_2 r_3 \sin(q_3) \\ 0 \end{bmatrix}^T \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} \tag{26}$$

$$C_{32} = \begin{bmatrix} m_3 l_2 r_3 \sin(q_3) \\ m_3 l_2 r_3 \sin(q_3) \\ 0 \end{bmatrix}^T \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} \tag{27}$$

$$C_{32} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}^T \begin{Bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{Bmatrix} \tag{28}$$

Gravity Vector

$$G_{11} = g\{m_1 r_1 \cos(q_1) + m_2(l_1 \cos(q_1) + r_2 \cos(q_1 + q_2)) + m_3(l_1 \cos(q_1) + l_2 \cos(q_1 + q_2) + r_3 \cos(q_1 + q_2 + q_2))\} \tag{29}$$

$$G_{12} = g\{m_2 r_2 \cos(q_1 + q_2) + m_3(l_2 \cos(q_1 + q_2) + r_3 \cos(q_1 + q_2 + q_2))\} \tag{30}$$

$$G_{12} = g m_3 r_3 \cos(q_1 + q_2 + q_2) \tag{31}$$

MATLAB Functions

Master File

This is the Master controlling the secondary functions and defining the variables of the system.

In this case a linear contour is executed.

Declaration of variables

```
global mass I l r tspan dt
global Pi Pf tf  Pd dPd ddQd dQd Qd
global Kp Kd Ki
global kp kd ks beta  N
global dp pq vq aq rvq raq
global alpha b fc fs ft delta dxs
global eta deta star dstar peta pdeta pstar pdstar
```

Friction Parameters

```
b =1.5;
fc = 3;
fs =5;
ft =100;
delta = 2;
```

Gain Definition

```
kp = [9.088240799499298;     8.304024394997082;   6.717464827016674]*1e3;
kd = [9.871988565005141;     7.900803640707129;   7.476418888180112]*1e3;
ks = [0.009597076721277;     0.004867545399102;   0.001182615747053]*1e3;
beta = [0.006550083466772;     0.006691256417746;   0.008477182028822]*1e3;

% TD-PD and PD-PD gains
Kp = kp;
Kd = kd;
```

```
Ki = [0;0;0];

% PDSC and PSC gains
kp = diag(kp); kd = diag(kd);ks = diag(ks);beta=diag(beta);

% Robotic Manipulator Parameters
mass = [1 1 0.5];
l = [0.5 0.5 0.3];
r = l/2;
I = [0.1 0.1 0.05];
```

Trajectory Planning

```
% Sampling and time definition
N =1500;
tf = 1.5;
dt = tf/(N-1);
tspan = 0:dt:tf;

%Definition of trajectory function
tr   = 10*(tspan/tf).^3-15*(tspan/tf).^4+6*(tspan/tf).^5;
dtr  = (30*(tspan/tf).^2-60*(tspan/tf).^3+30*(tspan/tf).^4)/tf;
ddtr = (60*(tspan/tf)-180*(tspan/tf).^2+120*(tspan/tf).^3)/tf^2;

% Definition of end-effector trajectory
Pi = [0.5;0.5;pi/3]; Pf = [0.7;0.7;pi/3];
th = atan2(Pf(2)-Pi(2),Pf(1)-Pi(1));
Pd  = bsxfun(@plus,bsxfun(@times,(Pf-Pi),tr),Pi)';
dPd = bsxfun(@times,(Pf-Pi),dtr)';

% Inverse kinematics for the joint trajectory
[Qd,dQd] = inverse_3dof_matrix(Pd,dPd);
ddQd = [ 0 0 0; diff(dQd)/dt];
```

TD-PID Controller

```
p0 = [Qd(1,1) dQd(1,1) Qd(1,2) dQd(1,2) Qd(1,3) dQd(1,3)];
opts = odeset('RelTol',1e-5,'AbsTol', 1e-5*ones(6,1));
[t,x] = ode15s('system_time_PID',tspan,p0,opts);
error = [Qd(:,1) dQd(:,1) Qd(:,2) dQd(:,2) Qd(:,3) dQd(:,3)] - x;
```

TD-PSC Controller

```
p0 = [Qd(1,1) dQd(1,1) Qd(1,2) dQd(1,2) Qd(1,3) dQd(1,3)];
opts = odeset('RelTol',1e-5,'AbsTol', 1e-5*ones(6,1));
[ts,xs] = ode15s('system_time_synchro2',tspan,p0,opts);
errors = [Qd(:,1) dQd(:,1) Qd(:,2) dQd(:,2) Qd(:,3) dQd(:,3)] - xs;
```

Position Domain Trajectory definition

```
pq = zeros(N,3);vq = zeros(N,3); aq = zeros(N,3);
dp = (x(end,1)-x(1,1))/(N-1);
pq(1,:) = Qd(1,:); vq(1,:) = dQd(1,:);
accel = [0 0 0; diff(x(:,2)) diff(x(:,4)) diff(x(:,6))]/dt;

% Trajectory transformation from time to position domain
% as a function of master motion
pq(:,1) = x(1,1):dp:(x(end,1));
pq(:,2) = interp1(Qd(:,1),Qd(:,2),pq(:,1));
pq(:,3) = interp1(Qd(:,1),Qd(:,3),pq(:,1));

vq(:,1) = interp1(Qd(:,1),x(:,2),pq(:,1));
vq(:,2) = interp1(Qd(:,1),dQd(:,2),pq(:,1));
vq(:,3) = interp1(Qd(:,1),dQd(:,3),pq(:,1));

aq(:,1) = interp1(Qd(:,1),accel(:,1),pq(:,1));
aq(:,2) = interp1(Qd(:,1),accel(:,2),pq(:,1));
aq(:,3) = interp1(Qd(:,1),accel(:,3),pq(:,1));


pq = [smooth(pq(:,1)) smooth(pq(:,2)) smooth(pq(:,3))];
vq = [smooth(vq(:,1)) smooth(vq(:,2)) smooth(vq(:,3))];
aq = [smooth(aq(:,1)) smooth(aq(:,2)) smooth(aq(:,3))];
rvq = [ 0 0;  bsxfun(@rdivide,diff(pq(:,2:3)),diff(pq(:,1)))];
rvq(1,:) = rvq(2,:);
raq = bsxfun(@rdivide,bsxfun(@minus,aq(:,2:3),bsxfun(@times,aq(:,1),rvq)),rvq(:,1).^2);
Pdq = forward_3dof(pq);
vq(end,:) = [ 0 0 0];
```

PD-PID Controller

```
pp0 = [pq(1,2) rvq(1,1) pq(1,3) rvq(1,2)];
popts = odeset('RelTol',1e-5,'AbsTol', 1e-5*ones(4,1));
[px, xp] = ode15s('system_position_PID',pq(:,1),pp0,popts);
errorp = [pq(:,2) rvq(:,1) pq(:,3) rvq(:,2)]-xp;
```

PD-PDSC Controller

```
pp0 = [pq(1,2) rvq(1,1) pq(1,3) rvq(1,2)];
popts = odeset('RelTol',1e-5,'AbsTol', 1e-5*ones(4,1));
[pxs, xps] = ode15s('system_position_synchro2',pq(:,1),pp0,popts);
errorps = [pq(:,2) rvq(:,1) pq(:,3) rvq(:,2)]-xps;
```

Sum and calculations Error

```
% Resulting trjectories
c = forward_3dof([x(:,1) x(:,3) x(:,5)]);
cs = forward_3dof([xs(:,1) xs(:,3) xs(:,5)]);
cp = forward_3dof([pq(:,1) xp(:,1) xp(:,3)]);
cps = forward_3dof([pq(:,1) xps(:,1) xps(:,3)]);

% Resulting contour error
ER = Pd-c; ERp = Pdq-cp;ERs = Pd-cs; ERps = Pdq-cps;
E = -ER(:,1)*sin(th)+ER(:,2)*cos(th);
Es = -ERs(:,1)*sin(th)+ERs(:,2)*cos(th);
Ep = -ERp(:,1)*sin(th)+ERp(:,2)*cos(th);
Eps = -ERps(:,1)*sin(th)+ERps(:,2)*cos(th);

% Synchronization and coupled errors
synchro_error=eta;synchro_derror=deta;coupled_error=star;coupled_derror=dstar;
synchro_perror=peta;synchro_pderror=pdeta;coupled_perror=pstar;coupled_pderror=pdstar;
time = tspan;pos = sort(px);
```

Published with MATLAB® R2012b

Secondary Functions

System Dynamics and PDSC Controller

This MATLAB function calculates the system's dynamics and control input. Then it calculates the state derivative

Input: x - vector of state variables t – time

Output: xdot – derivative of vector of state variables

```
%Output: xdot - derivative of state variable

function xdot = system_position_synchro2(t,x)
```

Declaration of variables

```
global mass I l r
global kp kd ks beta T ksii2
global pq vq aq rvq raq
global b fc fs ft friction
```

```
global peta pdeta pstar pdstar


m1=mass(1); m2=mass(2); m3=mass(3);          % Masses for each link
r1=r(1);  r2=r(2); r3=r(3);                  % Centres of gravity
I1=I(1); I2=I(2); I3=I(3);                   % Inertia of each link
l1 = l(1); l2=l(2); l3=l(3);                 % Length of each link
```

Identification of desired and actual state

```
dd = abs(bsxfun(@minus,pq(:,1),t));
j = find(dd == min(dd));

qd = pq(j,:)';                               % Desired position
dqd = [vq(j,1) rvq(j,:)]';                   % Desired velocity
ddqd = aq(j,:)';                             % Desired acceleration
dqp = rvq(j,:)';                             % Desired position relative
                                             % velocity
ddqp = raq(j,:)';                            % Desired position relative
                                             % acceleration

q  = [pq(j,1); x(1); x(3)];                  % Actual position
dq = vq(j,1)*[1; x(2); x(4)];                % Actual velocity
```

Cosines & Sines

```
c1=cos(q(1)); s1=sin(q(1));
c2=cos(q(2)); s2=sin(q(2));
c3=cos(q(3)); s3=sin(q(3));

c12=cos(q(1)+q(2));
c123=cos(q(1)+q(2)+q(3));

c23=cos(q(2)+q(3)); s23=sin(q(2)+q(3));
```

Matrix M

```
M(1,1)=I1+I2+I3+m1*r1^2+m2*(l1^2+r2^2+2*l1*r2*c2);
M(1,1)=M(1,1)+m3*(l1^2+l2^2+r3^2+2*l1*l2*c2+2*l2*r3*c3+2*l1*r3*c23);

M(1,2)=I2+I3+m2*(r2^2+l1*r2*c2)+m3*(l2^2+r3^2+l1*l2*c2+2*l2*r3*c3+l1*r3*c23);
M(2,1)=M(1,2);

M(1,3)=I3+m3*(r3^2+l2*r3*c3+2*l1*r3*c23);
M(3,1)=M(1,3);

M(2,2)=I2+I3+m2*r2^2+m3*(l2^2+r3^2+2*l2*r3*c3);
M(2,3)=I3+m3*(r3^2+l2*r3*c3);
M(3,2)=M(2,3);
```

```
M(3,3)=I3+m3*r3^2;
```

Matrix C

```
t11(1)=0; t11(2)=-m2*l1*r2*s2-m3*(l1*l2*s2+l1*r3*s23);t12(1)=t11(2);

t11(3)=-m3*(l2*r3*s3+l1*r3*s23);t13(1)=t11(3);

t12(2)=-m2*l1*r2*s2-m3*(l1*l2*s2+l1*r3*s23);t12(3)=-m3*(l2*r3*s3+l1*r3*s23);

t13(2)=t12(3);t13(3)=-m3*(l2*r3*s3+l1*r3*s23);

t21(1)=m3*(l1*r2*s2+l1*r3*s23)+m2*l1*r2*s2;t21(2)=0;t22(1)=0;t21(3)=-m3*l2*r3*s3;

t23(1)=t21(3);t22(2)=0;t22(3)=-m3*l2*r3*s3;t23(2)=t22(3);t23(3)=t23(2);

t31(1)=m3*(l2*r3*s3+l1*r3*s23);t31(2)=m3*l2*r3*s3;t32(1)=t31(2);t31(3)=0;

t33(1)=0;t32(2)=m3*l2*r3*s3;t32(3)=0;t33(2)=0;t33(3)=0;
C(1,1)=t11*dq;C(1,2)=t12*dq;C(1,3)=t13*dq;
C(2,1)=t21*dq;C(2,2)=t22*dq;C(2,3)=t23*dq;C(3,1)=t31*dq;C(3,2)=t32*dq;C(3,3)=t33*dq;
```

Matrix G

```
G(1,1)=9.8*(m1*r1*c1+m2*(l1*c1+r2*c12)+m3*(l1*c1+l2*c12+r3*c123));
G(2,1)=9.8*(m2*r2*c12+m3*(l2*c12+r3*c123));
G(3,1)=9.8*m3*r3*c123;
```

Friction Matrix F

```
f = -(fc+(fs-fc)*exp(-(dq/ft).^2)).*sign(dq)+b*dq;
```

Matrices for PDC

```
A = [ M(2,2) M(2,3); M(3,2), M(3,3)];
B = [M(2,2)*ddqd(1)+C(2,2)*dq(1), M(2,3)*ddqd(1)+C(2,3)*dq(1);M(3,2)*ddqd(1)+C(3,2)*dq(1),
M(3,3)*ddqd(1)+C(3,3)*dq(1)];
D = [M(2,1) C(2,1); M(3,1) C(3,1)]*[ddqd(1); dq(1)];
E = [G(2); G(3)];
F = [f(2);f(3)];
```

Controller

```
% Gains
pkp = kp(2:3,2:3);
```

```
pkd= kd(2:3,2:3);
pks= ks(2:3,2:3);

% Tracking Error
e = qd-q;                                   % Position error
de = [0 ;dqp-[x(2);x(4)]];                  % Derivative of position error

% Sychronization Error
Tau = [1 -1 0; 0 1 -1; -1 0 1];             % Synchronization Matrix
epsilon = Tau*e;                            % Synchronization error
depsilon = Tau*de;                          % Derivative of synchronization
                                            % error
peta(j,:) = epsilon(2:3)';
pdeta(j,:) = depsilon(2:3)';
ksii2(:,j) =Tau'*epsilon;

% Coupled position error
es = e+beta*epsilon;                        % Coupled error
des = de+beta*depsilon;                     % Derivative coupled error
pstar(j,:) = es(2:3)';
pdstar(j,:) = des(2:3)';
```

Control Input

```
if vq(j,1) ~=0                              % A logic function to signify
    s = sign(vq(j,1));                      % direction
elseif vq(j,1)==0
    s = 1;
end

tau = pkp*es(2:3)+pkd*s*des(2:3);           % PDSC controller
```

Derivative of State

```
xx = [0;0];
if vq(j,1)~=0
xx = A\(tau-B*[x(2);x(4)]-D-E-F)/vq(j,1)^2;
end

T(j,2:3) = vq(j,1)^2 *A*xx+B*[x(2);x(4)]+D+E+F;

xdot = [x(2);xx(1);x(4);xx(2)];

end
```

Published with MATLAB® R2012b

Inverse Kinematics of Robotic Manipulator

This MATLAB function calculates the joint angular position and velocities based on the end effector task space coordinates and velocities

Inputs: pos - End-effector task space position

vel - End-effector task space velocity

Outputs: angle- Joint angular position

dangle - Joing angular velocity

```
function [angle,dangle]=inverse_3dof_matrix(pos,vel)
```

Variable declaration and definition

```
global l N                                          % Apprehending link lengths and
population size from master file

x = pos(:,1);y=pos(:,2); q = pos(:,3);
dx = vel(:,1); dy=vel(:,2); dq = vel(:,3);

l1=l(1); l2=l(2);l3=l(3);                           % Renaming length links for use on this
function
```

Angular position calculations

```
x2=x-l3*cos(q);
y2=y-l3*sin(q);

cq2=(x2.^2+y2.^2-l1^2-l2^2)/(2*l1*l2);
sq2=sqrt(1-cq2.^2);

angle(:,2)=atan2(sq2,cq2);
angle(:,1)=atan2(y2,x2)-atan2(l2*sq2,l1+l2*cq2);
angle(:,3)=q-angle(:,1)-angle(:,2);

% Small logic algorithm for eliminating trajectory discontinuities
A = diff(angle);
if max(abs(diff(angle(:,1))))>=pi
    k = find(abs(A(:,1))==max(abs(A(:,1))))+1;
    nk = round(max(abs(A))/pi).*( A(k,:)./abs(A(k,:)));
    angle(k:end,:) = angle(k:end,:) + [ones(N-k+1,1)*nk(1)*pi ones(N-k+1,1)*nk(2)*pi ones(N-
k+1,1)*nk(3)*pi];
end
```

Calculating joint velocities

```
q1=angle(:,1); q2=angle(:,2); q3=angle(:,3);

s1=sin(q1); s2 = sin(q2); s3 =cos(q3);s12=sin(q1+q2); s123=sin(q1+q2+q3); s23 = sin(q2+q3);
c1=cos(q1); c2 = cos(q2); c3 = cos(q3);c12=cos(q1+q2); c123=cos(q1+q2+q3); c23 = cos(q2+q3);

dangle(:,1) = (c12./(l1*s2)).*dx+(s12./(l1*s2)).*dy + ((l3*s3)./(l1*s2)).*dq;
dangle(:,2) = (-(l2*c12 + l1*c1)./(l1*l2*s2)).*dx + (-(l2*s12 + l1*s1)./(l1*l2*s2)).*dy + (-
(l3*(l1*s23 + l2*s3))./(l1*l2*s2)).*dq;
dangle(:,3) = (c1./(l2*s2)).*dx+(s1./(l2*s2)).*dy+((l3*s23)./(l2*s2) + ones(N,1)).*dq;

end
```

Published with MATLAB® R2012b

Forward Kinematics of Robotic Manipulator

This MATLAB function calculates the end-effector task space position based on the joint angular positions

Inputs: q - Joint angular position

Outputs: pos - End-effector task space position

```
function pos=forward_3dof(q)
```

Variable declaration and definition

```
global l N                      % Apprehending link lengths and population size from master file

l1=l(1); l2=l(2);l3=l(3);       % Renaming length links for use on this function
```

End-effector position calculations

```
pos(:,1)=l1*cos(q(:,1))+l2*cos(q(:,1)+q(:,2))+l3*cos(q(:,1)+q(:,2)+q(:,3));
pos(:,2)=l1*sin(q(:,1))+l2*sin(q(:,1)+q(:,2))+l3*sin(q(:,1)+q(:,2)+q(:,3));
pos(:,3)=q(:,1)+q(:,2)+q(:,3);
```

Published with MATLAB® R2012b

Nonlinear Contour Error Estimation

This function calculates the error for nonlinear contours

Inputs: C - desired contour

C2 - actual contour

t-time

Output: Contour error

```
function Ec = contour_error(t,C,C2)

C = C(:,1:2); C2 = C2(:,1:2); t=t(:);
V = [0 0;bsxfun(@rdivide,diff(C(:,1:2)),diff(t))];
V2 = [0 0;bsxfun(@rdivide,diff(C2(:,1:2)),diff(t))];
E = C-C2;
dC = [0 0 ;diff(C)];
dC2 = [ 0 0;diff(C2)];

ds1 = sum(dC.^2,2);
ds2 = sum(dC2.^2,2);

L = (dC2(:,1)./ds2).*E(:,1)+(dC2(:,2)./ds2).*E(:,2);
dT = L./sum(V.^2,2);

Cr = C - bsxfun(@times,0.5*(V+V2),dT);
phi = atan((Cr(:,2)-C(:,2))./(Cr(:,1)-C(:,1)));
j = find(isnan(phi));

phi(j) = pi/2;
Ec = E(:,1).*sin(phi)-E(:,2).*cos(phi);
end
```

Published with MATLAB® R2012b