An adaptive system for enhancing vehicle bodies assembly using range sensing.

Yasser Mohamed. Eldeeb
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
https://scholar.uwindsor.ca/etd/943
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films
the text directly from the original or copy submitted. Thus, some thesis and
dissertation copies are in typewriter face, while others may be from any type of
computer printer.

The quality of this reproduction is dependent upon the quality of the
copy submitted. Broken or indistinct print, colored or poor quality illustrations
and photographs, print bleedthrough, substandard margins, and improper
alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript
and there are missing pages, these will be noted. Also, if unauthorized
copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by
sectioning the original, beginning at the upper left-hand corner and continuing
from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced
xerographically in this copy. Higher quality 6" x 9" black and white
photographic prints are available for any photographs or illustrations appearing
in this copy for an additional charge. Contact UMI directly to order.
AN ADAPTIVE SYSTEM FOR ENHANCING VEHICLE BODIES ASSEMBLY USING RANGE SENSING

by

Yasser M. Eldeeb

A Dissertation Submitted to the College of Graduate Studies and Research through the Industrial and Manufacturing Systems Engineering Program in Partial Fulfillment of the Requirements for the degree of Doctor of Philosophy at the University of Windsor

Windsor, Ontario, Canada

1999

© Yasser M. ElDeeb 1999
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-52420-5
ABSTRACT

In many of today's industrial robot applications, motions of the manipulator are planned using a prototype workpiece and played back for the execution of the actual manufacturing processes. Thus, for a successful task, a workpiece must be placed precisely at the pose (position and orientation) where the prototype workpiece was placed during the robot teaching process. Very often this precise positioning of the workpiece is difficult and costly. The motivation behind the research presented here is to overcome this difficulty. To this end, laser range sensors have been utilized to feedback the pose of the actual part to the robot controller, which corrects the robot pre-programmed path accordingly. In addition, a robot base and tool calibration system has been designed and implemented. The purpose of this system is to correct for pose discrepancies between the robot base, the tool coordinate frames in the simulation environment, and the corresponding poses in the actual workcell. The workcell simulation, motion planning and programming are performed off-line. The program is then downloaded to the robotic controller for execution.

This thesis addresses the problem of programming and controlling of industrial robots with the aim of responding to the demands for increased reliability and productivity with the desired quality and adaptability to environmental changes. For more efficient robotic systems, the control system must be able to adaptively react to parameters variation and unmodelled dynamics as it deals with moving and mating of parts. The challenge in this work is to develop a mechanism for integration of off-line programming with low level system control and sensing to achieve an adaptive assembly system. Thus enhancing the assembly process accuracy and minimizing the production cost. The developed system has been applied to the assembly of automotive Body-In-White parts, namely, the hinge and fender.
Robotic systems consist primarily of a manipulator and actuators that drive the manipulator links. How a real robot-actuator system behaves under various advanced control techniques is of great importance in practical robot applications, and is the motivation for work pursued in chapters 4, 5 and 6 of this thesis. Currently available robot controllers require exact knowledge of both robot and motor dynamics and acceleration feedback. Chapters 4, 5 and 6 presents a new approach that does not require exact knowledge of robot-actuator system. In chapter 4, This is achieved by adopting adaptive feedback linearization techniques. Enhancing those techniques, to be applied in an optimal sense by exploiting the Hamilton-Jacobi equation, minimizes the required control effort. The results demonstrate an asymptotic tracking response for the output; this is achieved using minimum control effort. By implementing an observer that guarantees asymptotic stability for the estimation errors, the acceleration feedback constraint is removed, which has a significant impact from a practical point of view since acceleration feedback is not available for most industrial controllers. Moreover, applying this technique removes linear growth constraints on the nonlinearities inherent in the system in order to guarantee global stability. Chapter 5 presents a different approach to the same problem by adopting a robust adaptive motion controller that requires only position measurements. The global stability of the proposed controller tracking performance is proved in the Lyapunov sense. In addition, simulation results are presented to demonstrate the asymptotic tracking performance of the closed-loop system. The significance of the above mentioned technique is that it does not require that a general expression and bound for the control input signal be found. In addition, it does not assume the availability of velocity measurements, which, for practical purposes may not be readily available. In Chapter 6, an adaptive controller ensuring zero force and tracking errors has been proposed. The stability of the proposed controller tracking performance is proven in the Lyapunov sense. The presented method establishes global as opposed to local convergence, in the presence of unmodelled dynamics.
ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor Professor Waguih H. ElMaraghy for his advice and effort on my behalf throughout the course of this work, which will be beneficial to my career for the rest of my life. I wish to thank my other committee members, Dr. N. Zamani, Dr. F. Salustri, and Prof. Hoda ElMaraghy for their valuable discussions and suggestions, on this work. I would like to appreciate the assistance I received from Dr. T. Pryor and Mr. A. Gray from Sensor Adaptive Machines Inc. (SAMI) for this work. I wish to thank all my colleagues who have provided help to this work at the Intelligent Manufacturing Systems Centre (IMS), specially Mrs. Ana Djuric and Dr. T. Lahdiri and Mr. Ram Barakat from the technicians.

Finally, I am sincerely grateful to my wife, kids and my parents for their optimism, support and understanding during several years of my graduate studies in Windsor.
TABLE OF CONTENTS

TABLE OF CONTENTS ........................................................................................................ vi
LIST OF TABLES ........................................................................................................... viii
LIST OF FIGURES .......................................................................................................... ix
NOMENCLATURE ........................................................................................................... x

- CHAPTER 1 -
  Introduction .................................................................................................................. 1
    1.1 Background ........................................................................................................... 2
    1.2 Research objective ............................................................................................. 3
    1.3 Contributions ...................................................................................................... 4
    1.4 Thesis Organization ............................................................................................ 5

- CHAPTER 2 -
  Preliminaries .............................................................................................................. 7
    2.1 Introduction ........................................................................................................... 7
    2.2 Industrial robots .................................................................................................. 7
    2.3 Literature survey of current approaches .............................................................. 10
    2.4 Calibration review .............................................................................................. 12
    2.5 Simulation review ............................................................................................... 15

- CHAPTER 3 -
  Assembly Automation ............................................................................................... 18
    3.1 Introduction ........................................................................................................... 18
    3.2 Sensors in assembly ............................................................................................ 20
    3.3 Sensor integration for assembly ......................................................................... 21
    3.4 Proposed industrial open architecture robot control (OARC) system ................. 24
    3.5 Major components of the proposed adaptive assembly system ......................... 26
       3.5.1 Trellis open architecture controller .............................................................. 26
       3.5.2 Sensor adaptive Machines Inc. (SmartProx) sensors ........................................ 27
       3.5.3 Off-line simulation software (TELEGRIP) ...................................................... 27
    3.6 Communication with sensors and actuators ....................................................... 30
    3.7 Supervisory system ............................................................................................ 31
    3.8 Part calibration module ...................................................................................... 32
    3.9 Workcell calibration system ............................................................................. 37
    3.10 Kinematic model for Robot Calibration ............................................................ 40
    3.11 Dynamic robot calibration ............................................................................... 41
    3.12 Errors in robot kinematic model ....................................................................... 43
    3.13 A modified regressor and controller expressions .......................................... 44
       3.13.1 Introduction ................................................................................................ 44
       3.13.2 Modified Model and Control Law Expressions ......................................... 46
    3.14 Workcell simulation ......................................................................................... 51
    3.15 Implementation of robot base and tool calibration ............................................ 51
    3.16 Part calibration (setup) ..................................................................................... 53
    3.17 Implementation of part calibration (setup) ...................................................... 54
## 3.18 System operation .............................................................. 54
## 3.19 Results ........................................................................... 56
## 3.20 Conclusion .................................................................... 58

- **CHAPTER 4 -**

  Adaptive control of a single-link manipulator using feedback linearization ................. 61
  4.1 Introduction ....................................................................... 61
  4.2 Exact linearization techniques ............................................ 64
  4.3 Adaptive control of linearizable systems ............................ 66
  4.4 Stability of closed-loop system ......................................... 68
  4.5 Optimal control of linearizable systems ............................. 71
  4.6 Case study ....................................................................... 73
  4.7 Simulation results ............................................................ 77
  4.8 Conclusion ....................................................................... 78

- **CHAPTER 5 -**

  Robust Adaptive Control of a Robotic Manipulator Including Motor Dynamics .......... 83
  5.1 Introduction ....................................................................... 83
  5.2 Problem statement ............................................................ 85
  5.3 Comparison with other designs ......................................... 92
  5.4 Stability analysis ............................................................... 93
  5.5 Simulation results ............................................................. 96
  5.6 Conclusion ....................................................................... 99

- **CHAPTER 6 -**

  Adaptive Control of a Constrained Robotic Manipulator Including Motor Dynamics .. 102
  6.1 Introduction ....................................................................... 102
  6.2 Problem statement ............................................................ 104
  6.3 Stability analysis ............................................................... 108
  6.4 Simulation ........................................................................ 113
  6.5 Conclusion ....................................................................... 115

- **CHAPTER 7 -**

  Conclusion and Future Work .................................................. 119
  7.1 Conclusion ....................................................................... 119
  7.2 Future work ..................................................................... 123

REFERENCES .......................................................................... 124
VITA AUCTORIS ................................................................... 135
LIST OF TABLES

Table 1: Physical Parameters ................................................................. 80
Table 2: Nominal System Parameters ...................................................... 80
Table 3: Parameters and States Initial Conditions .................................... 80
| Figure (3.1): | Working principle for the Smartprox sensor ........................................ 21 |
| Figure (3.2): | Workcell model in the simulation environment ........................................ 29 |
| Figure (3.3): | Actual workcell layout ........................................................................... 29 |
| Figure (3.4): | Robot calibration system (setup) .............................................................. 30 |
| Figure (3.5): | Schematic of adaptive assembly system ................................................... 56 |
| Figure (3.6): | Schematics of robot calibration system ..................................................... 59 |
| Figure (3.7): | Probes used for calibration in the actual workcell ...................................... 59 |
| Figure (3.8): | Probes used for calibration in the simulation environment ........................ 60 |
| Figure (3.9): | Verification of assembly operation ......................................................... 60 |
| Figure (4.1): | Schematic of a single link robot arm ........................................................ 80 |
| Figure (4.2): | Tracking response in case of typical feedback linearization .................... 81 |
| Figure (4.3): | Tracking response in case of optimal feedback linearization .................... 81 |
| Figure (4.4): | Comparison of output tracking error ......................................................... 82 |
| Figure (4.5): | Comparison of control input .................................................................... 82 |
| Figure (5.1): | Schematic of two-link planar robot .......................................................... 96 |
| Figure (5.2): | Tracking response for first joint ............................................................... 100 |
| Figure (5.3): | Tracking response for second joint ........................................................... 100 |
| Figure (5.4): | Tracking error for first joint .................................................................... 101 |
| Figure (5.5): | Tracking error for second joint ................................................................ 101 |
| Figure (5.6): | Parameter errors of the proposed controller ............................................. 101 |
| Figure (6.1): | Tracking error for first joint .................................................................... 117 |
| Figure (6.2): | Constraint force error ............................................................................... 117 |
| Figure (6.3): | Control torque for first joint .................................................................... 118 |
| Figure (6.4): | Control torque for second joint ................................................................ 118 |
NOMENCLATURE

\( p \) is an array representing the ideal part coordinates

\( q \) is an array representing the actual part coordinates

\( A \) direction cosine matrix

\( \alpha \) rotation angle about \( z \) axis

\( \beta \) rotation angle about new \( y \) axis

\( \Gamma \) rotation angle about new \( x \) axis

\( e_m \) error between corresponding ideal and measured points in the normal direction

\( H \) array representing part offset

\( J_i \) jacobian of the error at each measuring point \( i \)

\( A_n \) kinematic model transformation matrix

\( \theta_n \) joint angle of the \( n \)th link

\( dd_n \) common normal offset of the \( n \)th link

\( a_n \) link length of the \( n \)th link

\( \alpha_n \) twist angle of the \( n \)th link

\( \beta_n \) rotation angle around \( Y \) of the \( n \)th link

\( q, \dot{q}, \ddot{q} \in \mathbb{R}^n \) denotes joint position, velocity and acceleration respectively

\( \tau \in \mathbb{R}^n \) input torque

\( M(q), C(q, \dot{q}) \in \mathbb{R}^{n \times n} \) inertia, centrifugal and coriolis forces respectively

\( g(q) \in \mathbb{R}^n \) vector representing gravitational forces

\( Y(q, \dot{q}, \ddot{q}) \in \mathbb{R}^{n \times m} \) regressor represented by a matrix of known time functions

\( J_{hi} \) denote the jacobian with respect to the center of the \( i \)th link

\( m_i \) is the mass of the \( i \)th link

\( v_{hi} \) is the linear velocity of the center of the \( i \)th link

\( \omega_i \) is the angular velocity of the \( i \)th link

\( I_{hi} \) is the inertia matrix of the \( i \)th link with respect to the center of mass of the \( i \)th link

\( P \) vector representing robot parameters
\((f(x), g(x), h(x))\) are smooth functions where \(x \in \mathbb{R}^n\)

\(L_f h\) is the Lie derivative of \(h(x)\) w.r.t \(f\)

\(L_g h\) is the Lie derivative of \(h(x)\) w.r.t \(g\)

\(y\) system output

\(u\) control input

\(v\) new control input

\(z\) normal form transformation

\(\theta_i\) system unknown parameters for the \(f_i(x)\) known function

\(\theta_j\) system unknown parameters for the \(g_j(x)\) known function

\(\hat{\theta}_i\) is the estimate of \(\theta_i\)

\(O(\dot{x}(t))\) is the observability matrix

\(K_{obs}\) is the observer gain

\(e_t = y - y_m\) is the output tracking error

\(e_o\) is the observer error

\(\eta(t)\) is the closed loop error

\(V\) is a Lyapunov function candidate

\(\lambda_m\) is the maximum eigen value of \(M(s)\)

\(\gamma_M\) is the maximum eigen value of \(\Xi(s)\)

\(PI\) is the performance index

\(S\) is a smooth, positive function

\(u_{opt}\) is the optimal control input \((V)\)

\(\lambda\) is a positive constant scalar

\(\Theta(t)\) is the link angle \((rad)\)
$\omega(t)$ angular velocity of the link \((rad/s)\)

\(J\) is the sum of both link and motor rotational inertia \((Kg \cdot m^2)\)

\(M\) is the link mass \((Kg)\)

\(b\) is the damping coefficient of the motor \(\left(\frac{Kg \cdot m^2}{s}\right)\)

\(R_a\) is the armature resistance \(\Omega\)

\(L_a\) is the armature inductance \((\Omega \cdot s)\)

\(i_a\) is the armature current \((A)\)

\(K_m\) is the electromechanical coefficient (back EMF) \((V \cdot s/rad)\)

\(K_T\) is the motor torque coefficient \((N \cdot m/A)\)

\(l\) is the distance from the joint axis to link center of mass \((m)\)

\(\varepsilon\) is the torque error

\(\tau_f\) feed forward torque signal

\(K_{cd}\) controller derivative gain matrix

\(K_{cp}\) controller proportional gain matrix

\(e\) is the joint error

\(\phi_p\) error in robot parameters

\(\Phi_h, \Phi_r, \Phi_k\) errors in motor parameters

\(\Phi(q)\) geometric constraints in joint space

\(J(q)\) task space jacobian

\(s \in \mathbb{R}^{n-m}\) position control signal

\(\mu \in \mathbb{R}^n\) force control signal

\(\sigma\) combined control variable

\(\hat{\theta}\) is the estimate of the robot parameter vector \(p\)
\( \hat{M}(q) \), \( \hat{C}(q, \dot{q}) \), \( \hat{g}(q) \) are computed using the estimated parameter vector \( \hat{\Theta} \)

\( \tilde{f} \) force error

\( \kappa \) positive constant

\( G \) positive definite gain matrix

\( \Lambda \) positive definite gain matrix

\( \Sigma \) positive definite gain matrix
- CHAPTER 1 -

Introduction

Great strides are being made towards the automation of various manufacturing processes and systems with demonstrated productivity improvements and economical benefits. The application of robots in industry has been somewhat slower than previously anticipated. This may be attributed to a combination of factors including cost, performance, design and programming methods. Significant progress in robot technology in the next decade is envisioned in the areas of improved environment perception using sensors, more effective adaptation to tasks and environment uncertainties using advanced adaptive control techniques. Implementation of robotic applications in industry can be accelerated by providing quantitative proofs of the feasibility of such applications. Therefore, simulation models which can accurately represent proposed robotic cells and allow testing and evaluation of various alternatives before committing capital resources are valuable aids for decision makers. Robot simulation using computer graphics can be effective in many areas such as robotic workcell layout, program verification, task planning, off-line programming and performance evaluation. Cost saving, safety and user interaction are just a few advantages that make graphic simulations a valuable tool for such applications.

The rest of this introductory chapter gives the scope of the problem, the objectives behind this work, its contributions towards the solution, and applications of the results. The chapter concludes with an overview of the remaining chapters in the present work.
1.1 Background

Currently, growing numbers of robots are being employed in manufacturing environments to increase productivity and efficiency. One of the areas in which robots are utilized is the automotive industry. Performing complex robotic assembly tasks with prescribed accuracy is a challenging task. Some of the factors involved include signal communication, cell layout, robot speed and acceleration, trajectory planning and collision detection. The research described here is focused on the development of an adaptive system for automotive body-in-white assembly. The main goal of this system is to provide a physical mechanism for real-time geometric compensation at the assembly stage. In the initial phase, the parts to be mated are at some initial distance apart, and a contact-less measurement process is used to obtain the relative position of the main assembly part with respect to a master part nominal position. The results of the measurements are then used to control the robotic manipulator actuators to place the sub-assembly part in the desired position relative to the main assembly part. The proposed approach depends on the performance, \textit{a priori}, of a robot base and tool calibration procedure.

In this research, an industrial assembly workcell has been utilized. Research issues involved are computer simulation, sensing, flexible fixture, off-line programming, workcell calibration and control system software. The purpose of this study is to increase manufacturing industries’ flexibility and accuracy by introducing state-of-the-art technologies into manufacturing environments and by developing new workcell calibration and control approaches. Computer simulation software has been used in developing and programming an assembly cell. In addition, control strategies designed to handle modelling
and environment uncertainties, like adaptive and robust control have been developed. Examples for verifying the proposed strategies, utilizing single and two-link manipulators, are provided. Modelling and environmental uncertainties are represented by motor dynamics and payload variation, respectively.

Despite many advanced control algorithms that have been developed, Zaki et al. [1], Brady et al. [2], Koivo [3], Nagy, and Siegler [4], Spong, and Vidyasagar [5] and Massoud, and ElMaraghy [6], only relatively simple solutions have been successfully used in real industrial applications because of problems in tailoring text-book algorithms for practical use, Astrom, and Wittenmark [7], Nilsson, and Nielsen [8]. However, less attention has been paid to the means of incorporating the industrial aspects in real-time implementation of the system, specifically open controller architecture and sensor integration with the controller.

1.2 Research objective

The objective of this study is to develop an adaptive assembly system with the aim of improving the accuracy of automotive Body-In-White assembly parts. Very few research results have been reported as applied in industry, because of the complexity of the problem, especially in the presence of the practical industrial constraints. The development of workcell calibration and control techniques towards improved performance will be pursued further in this thesis.

To achieve the main goals, a number of related problems concerning design and implementation of a workcell calibration system had to be addressed. In order to enhance
the assembly operation accuracy, an intelligent robot controller architecture has been utilized to provide on-line response to sensors’ feedback.

As with many other dynamic systems, robustness and performance can be improved by introducing advanced control techniques. This approach has been adopted in designing controllers to handle parameters variation, input unmodelled dynamics and unmodelled uncertainties together with constrained motion.

1.3 Contributions

This thesis takes a problem-oriented approach, i.e. it includes a discussion of real industrial problems. The solution, as well as formulation of these problems, are its major contributions towards the development of an adaptive assembly system that handles process variations from one production cycle to another.

1- An adaptive system for enhancing automotive assembly accuracy, using sensor feedback has been designed and implemented.

2- A workcell calibration system has been designed and implemented in an industrial setup.

3- Workcell and sensor have been modelled in the simulation environment. Off-line programming was performed in the simulation environment and then downloaded for verification on the actual workcell.

4- In the proposed system, the adaptive process assessment is achieved through on-
line sensing. A method for integration of the data acquired by multiple sensors is proposed. In this approach, geometric workpiece data are transformed into the information which identifies necessary process correction(s). This transformation is accomplished by performing the Least Squares fit of the process data to the nominal workpiece data.

5- The achieved transformation is used to modify the nominal assembly path, designed originally for the CAD data.

6- An adaptive feedback linearization controller has been designed and implemented to handle parameters variation and input unmodelled dynamics. The performance of the proposed controller has been enhanced in an optimal sense.

7- An adaptive controller was designed and implemented to ensure zero position and force tracking errors in the presence of parameter variation and unmodelled disturbances.

1.4 Thesis Organization

This dissertation is organized in seven chapters. Chapter 1 states the background and motivation for this research, a discussion of the research objective, a list of contributions and the thesis organization.

Chapter 2 briefly introduces manufacturing systems and puts the aim of the research into a broader perspective. A literature review for current state-of-the-art related technologies is also presented.
Chapter 3 presents the structure of the proposed system and the idea of sensor integration and how within an industrial perspective, development towards more advanced industrial equipment should start from cost efficient, open control systems. In addition, this chapter presents the design and implementation of the workcell calibration system. Robot kinematic and dynamic calibration are addressed. It concludes with the system's operational sequence.

In chapter 4, an adaptive feedback linearization controller for a single link manipulator is developed and implemented. The developed controller is then revisited in an optimal control context and the optimal version is then implemented.

In chapter 5, a robust adaptive controller for a two-link manipulator including motor dynamics is developed and implemented.

In chapter 6, an adaptive controller for a constrained two-link manipulator including motor dynamics is developed and implemented.

Chapter 7 states some conclusions, discussion and proposals for future research areas.
- CHAPTER 2 -

Preliminaries

2.1 Introduction

Making machines programmable has been very beneficial in industrial production systems. The programmability is normally achieved by controlling the equipment from a computer which also provides a user interface for operation, configuration and programming. Typical examples are NC machines and industrial robots. The use of computer control to achieve flexibility implies that software issues for embedded control systems are central for the applicability and utilization of the equipment. This thesis focuses on problems related to simulation, calibration, programming and control of an industrial workcell. It takes a problem-oriented approach, i.e. it includes a discussion of a real industrial problem.

2.2 Industrial robots

The Robotics Industries Association (RIA) defines an industrial robot as follows, Singh [119]:

*An industrial robot is a programmable, multi-functional manipulator designed to move materials, parts, tools, or special devices through variable programmed motions for the
performance of a variety of tasks.

However, from an industrial point of view, the individual joints are commanded and controlled as any other multi-axis servo-controlled machine. This is easily demonstrated as motion commands specified numerically are interrupted, and thus, result in calls to the move-procedures. These procedures in turn control the physical system via sensors and actuators. To enhance the robots performance, more sophisticated motion specifications have been developed in the last 20 years. This has been accomplished on two fronts: the first is how motions and computations are specified, Nilsson, and Nielsen [8], while the other is in terms of the tools used for programming, Blume, and Jakob [12]. In spite of these efforts, robot programming is still manipulator-oriented rather than task-oriented. In this thesis we attempt to add more knowledge about the physical system by utilizing the dynamic model to improve performance. The idea presented above is to model the physical environment as we would a reference signal to a control loop to be tracked as the controlled output.

In Computer Integrated Manufacturing (CIM), a powerful host computer is utilized in both programming and supervision capacities. This is achieved through a computer network that connects on one end to the centralized engineering facility and on the other to the machines and workcells. This is one approach in which planning and control of the robot takes place on the centralized level.

Another approach is the decomposition of a large manufacturing system into smaller sub-systems. In that case the robot would be used as a stand-alone system that is programmed and operated directly on the shop floor.

The first approach has the advantage of decreasing the buffers for materials and components, and at the same time sharing different machines to manufacture a product.
The second approach has the advantage of enabling the operator to adjust the equipment to obtain the required production result.

In this work, a new approach is developed whereby the planning takes place on the central system using an off-line simulation environment, while some of the decision-making capabilities are delegated to the workcell, utilizing the intelligent sensors located in the cell. With this approach, advanced planning and sensor-based control systems are integrated into the supervisor (task-level programming layer) without taking control away from the operator.

Simulation and monitoring are integral functions of the supervisory software. Robot tasks are simulated before they are performed and the simulation system's collision detection functions determine whether the task can be performed safely. System operators who have proper control access can override safety systems if they determine that the safety constraints are too conservative for an experienced operator.

While the tasks are being performed the supervisor (end-user programmer) slaves the simulation system to the robot's motion sensors and monitors the robot to verify that the task was performed as simulated. In the proposed adaptive assembly system, the effect of unplanned events (e.g. workpiece pose variation) are represented in the world model, and thus, can be quickly handled by the operator. The end-user's ability to modify tasks without rewriting the supervisor program is a real asset, since the section that needs to be modified to reflect fundamental changes is the only portion to be written to extend the supervisor's ability (task-level programming) to the new situation.

To implement the above mentioned approach, a dedicated real-time operating system has to be used for low level robot control, and it has to be linked to the supervisory
computer with standard communication interfaces.

2.3 Literature survey of current approaches

Robot programs consist of commands in the form of statements for motion and information processing. Those commands are either built into the language or they are achieved by the use of robot software libraries. In the former case we have a special manipulator language, e.g AML, Taylor et al. [9] and PDL2, Comau S.p.A [10].

Recently, the fact that a robot program has a lot to do with information processing has led to the trend for using an existing language with a supplement of a robot library or a new generic programming language with special robot programming support. Examples of the first approach are RCCL, Hayward, and Paul [11], PASRO, Blume, and Jakob [12], whereas Karel, GMF [13] and RADIP, ABB Flexible Automation [14] are examples of the second approach.

Robot programming techniques today are either off-line, Chatila, and Harmon [15], or on-line, Craig [16]. The off-line programming has the advantage of savings in time (money) spent on teaching the robot various positions and via-points that the robot has to go through to perform a complex task. In case of using on-line programming, those robots have to be taken out of production during the teaching process, which results in production time loss, and expenses due to the involvement of experienced personal in the teaching task.

In off-line programming the production cell can be designed, programmed and its operation simulated before the cell is actually built. The outcome of this process usually
reveals how the equipment in the cell should be arranged and which robots should be used. Another advantage, besides having an advanced modelling and 3D visualization capabilities, is that most of the off-line programming software provides a code generator for a variety of robot controllers. One major problem with off-line programming software is that the representation of the real world is "approximate" and thus there is a bit of discrepancy between the cell model and the real cell.

On-line programming has a few more disadvantages in addition to those mentioned earlier. These may be summarized as follows:

1- The programming language/tools for on-line programming of advanced applications have been limited (e.g. teach pendant programming)

2- The availability of computing and control tools in embedded systems of on-line programming systems has been limited

On the other hand, the advantage of on-line programming is that it is tangible to the real world and does not suffer from the discrepancies associated with building the cell model in the simulation system.

It can be easily deduced from the above discussion that there is a trade-off between simplicity and programmability, represented in the on-line and off-line programming techniques, respectively. At the same time, both on-line and off-line programming are needed in various situations; that is why this thesis attempts to combine the advantages of both systems.

This is achieved by implementing a model based control approach coupled with animated graphics, facilitated by the use of a commercial state-of-the-art, off-line programming package, enabling the user through a graphical user interface to employ
advanced visualization software. In addition, a feature in this software which allows communication with the operator and/or sensors, has been exploited to decrease the programming time for complex operations and to monitor the robot's motion to verify task compliance. Moreover, the ability to feedback on-line variations in the robot program to the engineering workstation, has been developed.

Software support for demanding robot applications, requiring on-line tuning to deal with the manufacturing process and its uncertainties are the focus of this thesis. Appropriate choices of programming tools are probably the most crucial in such applications. This concept motivated the work in the following directions:

1- A dedicated program package for a specific application has been developed in close interaction with an industrial supplier, and

2- Design of robot control systems that support application specific programs

In summary, a review of different robot programming methods shows that a variety of approaches are beneficial in different programming situations. In order to support various manufacturing practices, it should be possible to provide the end-user (programmer) with proper programming tools, while bearing in mind that from an industrial perspective, development towards more advanced industrial equipment has to be cost effective and able to interface with different vendors sensors.

2.4 Calibration review

High precision is usually a requirement for successful realization of robotic applications with a Computer Integrated Manufacturing (CIM) environment. Particularly
important to this precision is high robot pose (position and orientation) accuracy to further enable implementation of off-line programs on the shop floor. Absolute pose accuracy depends on the quality of the manufactured robot and the accuracy of the robot model used for motion control. To ensure an accurate robot model, advanced measuring procedures and model-based identification methods are required. These procedures and methods are referred to as "robot calibration". Calibration results in a set of identified robot model parameters which can be used by the robot manufacturer as a check on the quality of robot production and by the robot user to improve the robot’s absolute end-effector (or tool) pose accuracy. Research into static calibration of robots has resulted in various methods for identifying the actual internal features of a robot, such as joint-axis geometries, which are not accurately accounted for in the robot's nominal controller model, Bennett et al. [17], Chen et al. [18], Driels, and Pathre [19], Everett, and Hsu [20], Hayati, and Mirmirani [21], Judd, and Knasinski [22], Mooring, and Pack [23], Mooring et al. [24], Schroer [25], Stone, and Sanderson [26], Veitschegger, and Wu [27]. These investigations include the development of various modelling techniques for estimating kinematic features of robots.

A key element in the calibration process is pose measurement. This involves workspace sensing of the end-effector or tool of the robot. Sensing the pose, therefore requires measurements that would be equivalent to imposing six conditions (3 translational and 3 rotational degrees of freedom) on the body pose. Constraining the TCP (Tool Center Point) of the tool hemisphere to remain in a given identical hemisphere in the workspace (i.e. fixture) corresponds to three conditions (3 translational degrees of freedom). Different approaches to the problem of 3-D position determination were introduced. One offers static measurement based on the use of automated theodolites and the triangulation principle, Riemensperger, and Gottwald [28]. The other takes a dynamic approach which utilizes a
laser tracking interferometer, and uses distance and direction to determine position, Decker [29]. Stereo triangulation uses two cameras located a fixed distance from each other detecting an illuminated target. Cameras require pre-calibration, and the limit of accuracy is in the order of 1% of the range, Zhang [30]. These systems are very expensive to use on the shop floor. Coordinate measuring machines (CMM) are instrumented mechanisms with multiple degrees of freedom (DOFs). When the tip contacts a target on the end effector, the robot and CMM form a closed kinematic chain by connecting the end effector to the ground, Duele, and Schroer [31]. The disadvantage of using the CMM for pose accuracy measurements is its limited ability to span the robot workspace.

Identification of robot kinematic parameters is a problem that has been addressed by a number of researchers, Everett, and Hsu [20], Mooring, et al.[24], Stone, and Sanderson [26]. The transformation between joint and world coordinate frames depends on the robot's model. This model employs a certain number of parameters, such as joint-axis offset, link length, angular axis tilt and link offset. These parameters geometrically describe the robot. The model is used to calculate the forward and inverse kinematics and, therefore, influences the robot's accuracy. Parameters identification means that with end-effector pose information and the corresponding robot encoder readings, the parameters of the robot model can be identified. This procedure generally optimizes the parameters over the set of measuring points. It has been shown that the overall accuracy of the robot can be substantially improved, Judd, and Knasinski [22], Duele, and Schroer [31]. Initial approaches to calibration tried to identify the parameters used in robots' controllers. Since explicitly invertible kinematic equations were used in control algorithms, only link length and joint zero position errors were identified. Small deviations of relative joint orientation were not identified in this way. To solve this problem, the four-parameter description of transformation was introduced by Denavit and Hatrenberg [46] and used by many authors
Caenen, and Angue [32], Schroer [33]. However, it became obvious that successive joints, which are nearly parallel, violated the model proportionality requirement when using DH parameterization. To overcome this problem, a different parameterization was introduced by Hayati and Mirmirani [21]. In general, parameters identification demands that the following three requirements be met.

1- Completeness: It implies that the robot kinematic model has enough coefficients to express any variation of the actual robot structure from its nominal design.

2- Proportionality: It implies that small changes in the robot structure should be reflected by small changes in the parameters of the kinematic model. To this end a modified kinematic model (Hayati model) is used in this work, where the proportionality problem is addressed by adding a twist angle variable to the DH model.

3- Minimality: This means that model parameter redundancy must be avoided. In this work nine parameters are identified for the robot base and tool coordinate frames.

2.5 Simulation review

A manipulator kinematic simulation is the solution of the equation of motion of the manipulator system that yields the position, velocity and acceleration of the system elements, as functions of time. The dynamic simulation also yields the drive force and torques required to produce this motion. Simulations are typically done off-line. In most cases, two levels of simulation are used to study a manipulator design and performance. The first is a kinematic simulation. In this type of simulation, the dynamics of the manipulator and its control systems are neglected. The motion of the manipulator arm is
assumed to be determined by the commanded joint displacement. The positions and velocities of the manipulator and associated equipment are calculated using standard kinematic methods, such as homogenous transformations, Denavit, and Hatrenberg [46]. The second and more complex form of simulation, uses a rigid link dynamic model of the arm. At this level of simulation the dynamic characteristics of the control system, drive actuators and transmission are considered. Here, the equations of motion are coupled sets of nonlinear, algebraic, differential equations. Their solution requires the use of numerical forward-integration techniques (fourth order Runge-Kutta). This form of simulation is most useful in designing and evaluating the manipulator’s control system. The basic structure of these levels of simulation is described as follows. The data describing the system parameters and commands are input to the simulation program and all calculations that are not time-dependant are performed. In advanced simulation packages, these data are obtained from computer-aided design (CAD) models of the manipulator’s elements, Pugh [34], Derby [36]. Then, when the user defines the initial conditions, the equations of motion of the system are evaluated. These equations are usually written in the well-known state variables form, typically joint-position, and those states associated with the control system. Using the simulation, the analyst is able to evaluate the performance of the manipulator as a function of its design.

Graphical simulation is a significant tool in providing better visualization of robot interactions with the workcell environment. A surface modeler is used to represent objects in ROBOCELL [124] simulation package while Leu et al. [125] and TELEGRIP [40] have based their simulators on solid models as opposed to wire-frame. Surface modelling is adequate for modelling objects if the purpose is visualization, motion simulation, task validation or collision avoidance. On the other hand, solid modelling is necessary if dynamic behavior of workcell components is a primary concern (trajectory tracking,
painting, complicated assembly, etc.).

The major disadvantage of simulations is that if they tend to become very large, and their cost grows rapidly. Simulation costs can be reduced if the analyst uses simulation cases that investigate a particular aspect of the design. For example, a simulation of the manipulator’s kinematic might be appropriate to plan and validate the task execution, especially if this task is pick-and-place operation. However, the cost of simulation is far less than the cost of evaluating various designs experimentally, where design or task modifications can only be studied by the expensive and time-consuming method of manufacturing new hardware.
Assembly Automation

3.1 Introduction

One of the most pressing needs in today's manufacturing is the continuous demand to increase productivity, and thus minimize the time from concept to the final product. A true increase in productivity can be achieved by reducing the burden of costs and processes that do not add value to the product and by eliminating manufacturing components that are dedicated to a single product. These factors, together with new technological developments, have led to efforts to develop automated assembly systems. Car body assembly can be considered as a process in which stamped sheet metal parts are brought together then permanently joined. Mechanical fixturing and tooling is traditionally employed in manual and automated assembly to position and constrain workpieces. The jigs and fixtures are dedicated devices for specific workpieces. Therefore a significant issue in flexible assembly is workpiece positioning. The major function of the hard tooling is to locate a workpiece accurately. These tooling systems are usually designed to be insensitive to variations in the geometry of the workpiece upon which they operate. That desired insensitivity is not always achieved due to wear and part variation. Traditional assembly systems suffer from numerous sources of uncertainty, such as for example

- Parts variation;
- Robotic base and tool frames errors;

- Positioning errors; and

- Fixture location with respect to other devices used in the process.

In recent years, the volume of research work and applications related to the development of flexible assembly systems has increased significantly. Most of the published results, however, are from the electronic and computer industries, Gordon, and Seering [37], Moinet [38]. In the aerospace industry, hard automation of the assembly is possible, but not cost effective due to the small batch sizes. To overcome this limitation the Flexible Assembly System for the assembly of airframe components was developed as a flexible assembly cell that could replace dedicated tooling and be able to quickly reconfigure itself for new types of subassemblies. The most significant contribution towards adaptive automotive assembly was introduced by Pasek [39] at the University of Michigan. However, the proposed adaptive assembly system in this thesis differs from the previously mentioned study in several important aspects:

1- Robot calibration is addressed in the wider context of workcell calibration (robot, fixture, sensor, workpiece etc.).

2- The proposed approach distinguishes between manufacturing and fixturing errors.

3- The proposed adaptive assembly system is built around a commercial distributed control system, sensors and simulation software.

4- Position and force controllers adopting adaptive and robust control techniques
have been proposed and their performance demonstrated in simulation.

3.2 Sensors in assembly

As a result of recent advances in sensing technology many on-line sensors for industrial applications have been developed. These can be classified by their function as proximity, tactile, and imaging. In the automotive assembly application the sensing system should meet the following requirements

- A high measurement rate,

- High measurement accuracy and repeatability, and

- Insensitivity to electromagnetic noise and mechanical vibration.

Optical sensors used in the proposed adaptive assembly system, Sensor Adaptive Machines Inc. [44] make use of the basic principle of triangulation. Each sensor employs a combination of laser diodes to create a structured light pattern, and a CCD camera to capture the image of the laser light reflected from the surface being measured. Both the laser and camera are mounted in a common housing for accurate alignment. The laser is equipped with optics to project a pattern of structured light. During the manufacturing process, each sensor is calibrated to determine its internal geometry and optical characteristics and establish the sensor coordinate frame. Since the sensors perform only the image acquisition, further image processing is provided by the system controller. The system controller is a multi-functional unit, combining functions of an A/D converter, sensor multiplexer, image processing, as well as an operator's interface.
3.3 Sensor Integration for Assembly

The integration of sensors for process and robotic control makes it possible to design a manufacturing cell capable of dealing with inconsistencies in the workpiece, as well as in the process itself. By interfacing the sensors' controller with the robot's controller, the process can be adapted to changes in the environment.

The basic elements of interaction within the assembly process are the subassembly (held by the robot) and the workpiece (main assembly). In a typical assembly cell, those two elements are tied by an off-line program that defines the assembly process. This means
that the process is likely to fail when there is a change in the environment. Changes in the environment are mainly caused by parts, robot base and tool mislocations. This factor needs to be measured to get a true representation of the workcell. The deviation of the main assembly workpiece from its nominal position is measured by the laser range sensors (SmartProx™). This information is fed to the supervisory controller which determines the transformation representing this mislocation and feeds this information to the robot controller (using the high level C function calls provided by the industrial robot controller software) to correct the robot’s tool path respectively. As for the pose discrepancy between the nominal (CAD model in simulation) and real poses of the robot base and tool frame, a calibration system is designed, built and implemented. This system identifies the pose error and corrects for it by moving the robot base and tool frame in the simulation environment to match the real pose of these frames in the workcell, respectively.

In order to achieve the goal of high precision in performing robotic tasks, especially assembly tasks, workcell positioning accuracy has to be addressed. Workcell positioning accuracy refers to the workcell robotic devices’ ability to move the end effector to a desired pose (position and orientation) that is specified in the task space. By desired pose we mean, a commanded pose that has been programmed using the robot programming language. Accuracy, in this sense, is different from repeatability since the robot has to attain the required pose without being taught.

At present, the repeatability of industrial robots is at least an order of magnitude better than their accuracy, Mooring [24]. This fact has been exploited for on-line programming. On-line programming, Chatila, and Harmon [15], refers to the use of teach pendants to jog the robot through the entire assembly task, achieving operator-defined
poses, and saving the corresponding joint angles. These saved taught poses are played back for task execution. However, to exploit the full power of off-line programming, i.e., time savings, collision detection, cycle time calculation, and virtual manufacturing environment simulation, the positioning accuracy has to be addressed.

The sources of error considered in this work are summarized according to Mooring et al. [24] as follows:

1. Static errors in the part and robot pose relative to one another.

2. Kinematic errors in the robot kinematic model (TCP coordinate frame relative to robot base coordinate frame).

3. Dynamic errors in the pose of the robot tool as it moves along a path in the workcell.

Little formalism exists in the literature of robot performance standards due to the newness of the subject. Manufacturers generally provide sparse information about their products beyond a single valued quote of repeatability. A fundamental enhancement in this work is the ability to model high frequency disturbances due to unmodelled actuator dynamics and parameters variation due to payload variations which affects tracking and positioning accuracy of industrial robots. To this end, several control techniques were proposed in this thesis based on feedback linearization and Lyapunov design methods.

For industrial robots, the motor, power electronics and controller are proprietary to the manufacturer, therefore, the ability to test the proposed control techniques depends on the availability of an industrially open controller architecture. In such a controller, the user
must have the ability to model the dynamics of the robotic system and to interface sensors in the workcell to it.

One approach towards minimizing kinematic errors, involves building more accurate robotic devices with tighter tolerances, and more precise manufacturing processes. This approach is expensive and time consuming. Moreover, even with this achieved, the components of the robotic workcell must be precisely located relative to one another on the shop floor, which is also an expensive and time-consuming task.

An alternative approach, adopted in this work, is to use calibration. In which the workcell components are simulated in the TELEGRIP™, Deneb Robotics [40] environment in nominal locations. The measurements from the actual workcell defining the workcell components pose are gathered and uploaded to the simulation environment. The simulation environment is then updated to reflect the real location of the workcell components, and the assembly task is programmed off-line in the simulation environment and downloaded to the workcell robotic devices.

3.4 Intelligence in proposed adaptive assembly system

It is rather difficult to find common grounds for the definition of intelligent control, generally employed in the robotics industry. It is proposed that an intelligent control system should allow the user (system manager) to change and/or add certain internal components of the system. Fanuc's definition, Brooke [41] on the other hand, is the ability to communicate with other vendors' equipment through the ethernet or remote I/O (Input/Output), and to modify the programmable locators' positions using off-line simulation. At
the same time, the user is denied access to the trajectory generator or controller for both the robots and the programmable locators. The Nissan (IBAS), Abe et al. [42] system shares some of the activities with the previously mentioned system (namely, the ability to communicate with other vendors' equipment through the ethernet or remote I/O), except that the latter has a structured hierarchy in delegating the control commands at different levels (e.g. station sequencer, sub controllers, unit controllers and servo controllers).

The proposed adaptive assembly system shares the advantages of the above-mentioned systems, basically possessing off-line simulation capabilities for graphical representation, trajectory planning, collision detection, sensor integration, and an interface to different vendor equipment through ethernet or I/O external connections. The proposed controller provides the user with the ability to tune the controller gains. However, it does not provide the user with the facility of incorporating a model based controller. In addition, the system was built on top of a Trellis controller allowing access to the SERCO drives, which are digital drivers communicating on a fiber-optic ring. This servo control layer represents a lower layer in the hierarchy topped by another layer where the application is defined in the robot manufacturer's language or the Trellis controller language. This layer, in turn, is topped by another layer where the off-line programming takes place. The lowest level in this hierarchy, is the hardware controls for the drivers themselves. On the sensing side, the highest level is a user interface which represents the client that slaves a server, represented by the SAMI monitoring system to gather measurements. From here, commands are sent to the lowest level of the interface, which is the sensors' controller and SmartProx™ sensors. This structure has the following benefits:

1- It provides the end user with hardware transparency for both the sensors and the
actutators.

2- It provides multitasking to concurrently support a multitude of sensing, actuation and control tasks.

3- It assures the ability to generate sensor monitoring processes on the fly, based on the world state.

3.5 Major components of the proposed adaptive assembly system

3.5.1 Trellis controller

The main module is the Trellis Motion Controller operating system (TMOS), Trellis Software & Controls [43], which is a family of products for programming & controlling motion of robots, machine tools, and other more general purpose multi-axis machines. It is compromised of off-the-self software (Lynx which is a UNIX based real-time operating system) and hardware components (PC computer and SERCO drives), and a motion control system that provides a tunable gain Proportional-Integral-Derivative controller (PID). The robot application (C-works) contains the code for motion and I/O control. The TMOS module is linked with the Servo Interface and the Kinematics library which defines the servo hardware and robot kinematics configuration. The Control Panel is primarily an interface to the Workstation Manager. For the C-Works application, the Control Panel provides basic motion control and start-up functions, such as Hold/Pause, Start, Enable Drives, etc.
The Message Logger is responsible for displaying messages received from any Trellis Motion Controller module or application. The three types of messages include progress messages, error messages and debugging messages. Hence, the Message Logger provides a unified interface to monitor the system operation.

3.5.2 Sensor adaptive Machines Inc. SmartProx\textsuperscript{TM} sensors

Sensors are used for in-process monitoring in assembly operations. These operate via non-contact laser/electro-optical sensors mounted directly in the assembly weld tools. Actual surface location data, relative to the desired location, is provided. The SmartProx\textsuperscript{TM} sensors, Sensor Adaptive Machines Inc. [44] are based on optical measurement techniques, using digital image sensors for drift-free operation on the plant floor. All sensors for measuring surface, edge and hole locations feature a measurement range of +/- 5mm from nominal, with sensor accuracy of 0.1 mm. Typical sensor data acquisition time is 15 msec per point, which allows data to be acquired without adding significantly to the cycle time.

3.5.3 Off-line simulation software TeleGrip\textsuperscript{TM}

The proposed adaptive assembly system uses 3D visualization and simulation software with intuitive operator interfaces for the programming and control of complex robotic systems. Graphical programming software modules allow an operator to command and simulate complex tasks in a graphic preview mode. When the task performed in
simulation environment is acceptable, motion commands are issued for the actual robots and their motion is monitored within the simulation graphic environment. At the same time, it reports the changes within the real world and allows the robot to perform tasks that cannot be accurately represented with models alone using a combination of model and sensor-based control.

What distinguishes this software from conventional off-line simulation is that it is not only a tool to verify robot programs before execution, but it also reports the changes within the real world through built in sensor integration facilities which enhances the ability to update the model on-line. The workcell model in the simulation environment is shown in Figure (3.2). The real workcell layout is shown in Figure (3.3). A schematic of the proposed adaptive assembly system is shown in Figure (3.4).
Figure 3.2: Workcell model in the simulation environment

Figure 3.3: Actual Workcell layout
3.6 Communication with sensors and actuators

Reduced cost for cables, reduced noise level and increased reliability are advantages obtained by placing sensors, actuators and necessary electronics close to the motors on the mechanical robot. This was experienced a long time ago and led to the use of optical connections. This is used in retro-fitting the Comau S2 with SERCO drives. The advantages of using SERCO drives in motion control are as follows:

1- More flexibility is achieved by moving processing power to the drives.

2- It is suitable for distributed control because it pushes axis-dependant control functions (e.g. loop closures, interpolation) into the drives letting controllers concentrate
1- The robotic system communicates via synchronous serial communication. A bit transfer rate of 1.5 Mbit/s makes it possible to transfer 32 bits of data with a sampling rate of 8 KHz for each joint, if six joints are used.

2- The serial communication signals between the Trellis controller and the drives hardware interface, as well as between the drives hardware interface and the robot are connected via IEEE-422 line drivers.

3- The laser range sensors controller communicates with the supervisory computer system (End-user Interface) through Transmission Control Protocol (TCP) / Internet Protocol (IP) connection on the Personal Computer Interface (PCI) bus.

3.7 Supervisory system

The supervisory software running on the front end Personal Computer (PC), is connected to the trellis controller through TCP/IP. This software downloads application programs written in the K2 programming language of the Trellis controller to the Trellis controller. Additionally, it reports back to the user any communication problems between the front end PC and the controller. On top of this interface there is a software layer that provides transparency of the low level control to the actuators and sensor setup information to the operator. This layer provides the functionality that the operator typically needs on the shop floor, including calibrating the cell, enabling and disabling the drives, starting the assembly operation, halting the operation and connecting to the sensors for feedback. The industrial significance for the proposed adaptive assembly system includes the following:
1- It achieves greater flexibility in production (assembly, etc.) systems. This is due to the fact that it does not require dedicated tooling.

2- It performs the assembly operations more accurately due to the fact that part positioning errors are eliminated through laser range sensors feedback. In addition, the relative workcell components position accuracy is increased significantly through robot base, tool and workpiece calibration.

3- Potential benefits due to the fact that it is more software than hardware oriented. For example, the measuring equipment setup can be rearranged (e.g for reuse of sensors) to perform new assembly functions.

4- The system is easy to use from the operator point of view, since the low level actuator and sensor control are transparent to the user.

3.8 Part calibration module

Introduction

The major function of the ordinary tooling equipment is to locate a workpiece accurately. Most of today's assembly systems use a process in which the workpiece is pushed against hard locators, while sensing is used mainly to monitor the presence or absence of parts. The solution proposed here aims at the eliminating the part positioning errors during assembly. The errors are due to the lack of positioning accuracy of the workcell components relative to each other and hence to the world coordinates frame. The proposed approach comprises error detection using laser range sensors (SP1) and
correction using a parametric calibration to identify the transformation parameters needed to determine the part pose relative to the fixture and therefore world coordinates frame.

Part calibration refers to determining the part pose relative to the fixture, and hence to the world coordinates frame. The process has an extra complexity added to it, which is the manufacturing variations between different parts due to tolerances. For operation on the shop floor, the variations between parts to be assembled is a problem which must be addressed continuously. In this work the part error is identified by differentiating between errors in part pose and errors in the manufacturing process. The first error is corrected by locating the laser range sensors used (SmartProx\textsuperscript{TM}) relative to the world coordinates frame, using the designed calibration system (Section 3.9). The sensors' pose is updated in the simulation accordingly. The manufacturing errors are identified, and compared with the allowable tolerance levels defined in the sensor set-up for the assembly operation at each measuring point, respectively.

**Problem Statement**

The object is to find a variable number of degrees of freedom, up to 6 \((x, y, z, \text{pitch}, \text{yaw}, \text{roll})\)that define the part coordinates frame relative to a another coordinate frame describing the ideal (master) part pose. After this transformation is defined, the sensor readings of the actual workpiece at the measured points are multiplied by the transformation matrix, generating their corresponding values on the master workpiece coordinate frame. The residuals obtained by subtracting the transformed values and the corresponding master readings at the same points, along the normal to the surface,
are taken as the manufacturing errors.

Algorithm for part calibration

It is assumed that an ideal part pose is a 6-tuple, namely: \( p = x, y, z, a, b, c \) where, \( x, y, z \), is the Cartesian coordinates frame and, \( a, b, c \), are the unit normals in the direction of measurement. The 6-tuple is required only for defining the ideal part pose. The measurement coordinates of the actual part consist of, \( q = (x, y, z) \) only, since the measurements are acquired along the normal direction of the sensors. For \( m \) degrees of freedom, at least, \( m \), ideal part pose coordinates \( (p_i \text{ where } 1 \leq i \leq m) \) are needed that span the space and their corresponding measurement coordinates \( (q_i \text{ where } 1 \leq i \leq m) \) based on the degrees of freedom sought. In the proposed adaptive assembly system setup the three degrees of freedom sought are, \( x, y \), and \( roll \). The objective of the part calibration algorithm is to fetch translation and rotation parameters that, when applied to the measurement coordinates, will minimize the error between the transformed measurement coordinates and the ideal part coordinates, along the direction normal to the sensors. The transformation is assumed to be a rigid body transformation.

Procedure

Let, \( p \), represent the ideal part coordinate and, \( q \), be the corresponding measured (actual) point of, \( p \). Then, the ideal part coordinates frame, \( p \), can be transformed to the measurement coordinate frame, \( q \), as follows:

\[
q = Ap + H
\]  

(3.1)

where, \( A \), is a \( 3 \times 3 \) direction cosine matrix defined as follows
The direction cosine matrix, $A$, is based on a $ZYX$ rotating sequence where:

- $\alpha$, is about the $Z$-axis,  
- $\beta$, is about the new $Y$-axis,  
- $\Gamma$, is about the new $X$-axis.

$H$, is the part offset $(h_x, h_y, h_z)$.

By projecting the error vector, $q - p$, onto the normal, $n$, a measurement of the residual error in the critical direction of measurement is obtained. Therefore, there is a need to find parameters such that for all measured, $k$, data points, we minimize the residual error squared. This is mathematically represented as follows:

$$e_m^2 = \sum_{i=1}^{k} [n_i (q_i - p_i)]^2$$  \hspace{1cm} (3.2)

This problem is nonlinear and the Levenberg-Marquardt, Scales [45], method is used to formulate its solution. The algorithm has the advantage of both the traditional Steepest Descent Methods, Scales [45], and the Inverse-Hessian method. Steepest Descent methods are based on the simple Newton method. This means that convergence form one point to the other is always in the direction of the steepest negative gradient. Although these methods are very robust to poor initial conditions, they can take a long time to converge. 

These algorithms are classified as first order methods since they require the calculation of first order partial derivatives. The Inverse-Hessian method involves the calculation of second order derivatives. By using the information from the second derivatives the algorithm will tend to converge faster than in the Steepest Descent methods. However, this
method requires very good initial conditions. If they are lacking, the method becomes very computationaly expensive. The ideal situation would be to find a compromise between the two algorithms. The Levenberg-Marquardt algorithm provides this compromise. This method uses steepest descent parameter estimation when the search is far from the minimum of the objective function (error function in this work), and continuously switches to an inverse-Hessian method when it gets close to a minimum. The error function is represented at each measurement point as follows:

\[
e_{m_i} = (a_{11} n_x + a_{12} n_y + a_{13} n_z) \times (q_x - p_x) +
(a_{21} n_x + a_{22} n_y + a_{23} n_z) \times (q_y - p_y) +
(a_{31} n_x + a_{32} n_y + a_{33} n_z) \times (q_z - p_z)\]

(3.3)

The formulation is as follows:

\[
\begin{bmatrix} J_i^T J_i + \mu_i I \end{bmatrix} C_i = -J_i^T e_{m_i} \quad (3.4)
\]

where, \(\mu_i\), is a positive scalar coefficient, set at the beginning of each iteration step. This coefficient starts with a large value and, decreases as the variables approach their optimal values. \(I\), is the, \(n \times n\), identity matrix, in our case \(n = 6\). \(C_i\), is a search vector and in our case it has a descent direction.

The Jacobian of the error at each measurement point, used in the Levenberg-Marquardt method, is as follows:

\[
J_i = \begin{bmatrix}
\frac{\partial e_{m_i}}{\partial p_x} & \frac{\partial e_{m_i}}{\partial p_y} & \frac{\partial e_{m_i}}{\partial p_z} & \frac{\partial e_{m_i}}{\partial \alpha} & \frac{\partial e_{m_i}}{\partial \beta} & \frac{\partial e_{m_i}}{\partial \Gamma}
\end{bmatrix}
\]

(3.5)
If the error Jacobian matrix $J$ for the measurement points $k$ is of full rank, this guarantees the uniqueness of the solution. This means that the workpiece position is deterministic, which, in turn, means that there exists only one transformation that defines the workpiece pose relative to its ideal (master) coordinate frame. In order to achieve the proposed workpiece calibration a C program has been used to implement the “part calibration” technique discussed earlier. The program is integrated in the simulation software as a shared library using the C function hooks facility provided by the simulation software. The developed function “part calibration” is called from the GSL program written for the calibration process.

3.9 Workcell Calibration System

Introduction

One approach to improving accuracy in robotic workcells is to build more accurate robots with stiff joints and links and tight tolerances. In addition, the layout could be developed very precisely using 3D metrology, Riemensperger, and Gottwald [28]. The idea is to match the real workcell on the shop floor with the simulation. This approach is both expensive and time consuming.

The alternative approach is to use calibration, where the idea is to match the simulated workcell with the real workcell, since the robotic workcell components are known to be repeatable. The calibration process allows the user to identify sources of position inaccuracy and modify the simulated workcell accordingly to correct for this inaccuracy.
Modelling

The first step in the calibration process is to produce a model that relates between the joint space and corresponding end effector pose in the task space. The model used for the calibration process depends on the level of calibration required. The joint-level calibration compensates for errors in zero positions of each joint in the robotic device. Robot manufacturers, including COMAU, often include calibration functions to account for such errors. Thus, if a robot is calibrated at the joint level, the major source of error (about 80%) [40] is derived from the location of the robot in the workcell, its tool pose relative to the robot end plate, and its base pose relative to the other workcell components.

Measurement

The goal in measurement is to precisely determine the end effector pose (or a subset of it) for a set of known (given) robot joint displacements. Different examples in the literature, Stone and Sanderson [26], Veitschegger and Wu [27], Riemensperger and Gottwald [28], have demonstrated measurement with laser tracking interferometers, phototracking devices, acoustic distance ranging, and surveying theodolites. The calibration functions make no assumption about the method of gathering the experimental information. This is the reason for choosing a simple, efficient, and automatic method of collecting data using the robot and fixture equipped with a special tool as shown in Figure (3.7).

Identification

The purpose of identification is to choose a specific translation and/or rotation for the robot base coordinate frame to minimize the difference between the nominal and actual
robot base coordinate frames. In the same manner, the translation and rotation of the tool coordinate frame is achieved. The robot workcell parameters may be identified from complete or partial measurements of the tool pose in the workcell coordinates. The calibration software identifies the workcell relative pose parameters using non-linear least squares model fitting with the Levenberg-Marquardt method. The measurements in the robot workcell are made using the robot tool tip position.

Correction

Correction entails using the information achieved through the previous steps of the calibration process to improve the workcell model in the simulation environment to match the real workcell and hence, enhance the positioning accuracy of workcell components relative to each other. The details of the actual correction tend to be machine and application specific. In this workcell, two steps of calibration have been addressed. The first step was to establish the workcell reference frame by calibrating the robot with respect to the calibration fixture. The other devices have been calibrated relative to the workcell reference frame, and their poses have been modified accordingly in the simulation environment. The second step is calibrating the workpiece relative to the robot coordinate frame and repositioning it accordingly in the simulation environment.

Robot base calibration

The base pose is the six degrees of freedom (DOF) transformation representing the robot base position and orientation relative to the workcell coordinate frame. This transformation allows six parameters (X, Y, Z, Yaw, Pitch, Roll) to be identified in the base calibration process. This calibration procedure defines the transformation, $R_W^B$, as shown
in Figure (3.6). Since the robot base coordinate frame is at an imaginary location inside the base, these parameters can not be directly measured and must be identified using indirect measurement techniques.

**Tool calibration**

The tool pose is a six degrees of freedom (DOF) transformation from the robot mount plate to the tip of the tool. The tool parameters to be identified are \([X, Y, Z]\) which determine the tool tip position in the mount plate coordinate frame, \(R^T_{EE}\), (as shown in Figure (3.6)), since the measurements are made for the tool tip position. The experimental probe used in this procedure is shown in Figure (3.7) and in the simulation environment in Figure (3.8).

**3.10 Kinematic Model for Robot Calibration**

Broad interest has been paid to kinematic calibration and its effect on improving robot accuracy through software rather than changing the mechanical design of the robot. The kinematic model of the robot is generally invariant and may be represented with a fixed number of parameters. The fundamental tool used to describe the spatial relationship between various objects and locations in the manipulator workspace is the Denavit-Hartenberg method [46], with modifications proposed by Hayati [21] and Mooring [24] to account for disproportional models when two consecutive joint axes are nominally parallel. The kinematic model is defined by the transformations required to change from one frame to another. The transformation takes the form:
\[ A_n = R(x, \alpha_n)T(x, a_n)R(z, \theta_n)R(y, \beta_n)T(z, dd_n) \]  

When the coordinate frames are placed in accordance with the modified Denavit-Hartenberg method, the transformations given in the above equation will apply to all transforms from one frame to the next. These may be written in a generic matrix form where the elements of the matrix are functions of the kinematic parameters. These parameters are simply: the joint angle, \( \theta_n \), the common normal offset, \( dd_n \), the link length, \( a_n \), and the angle of twist, \( \alpha_n \), and the angle, \( \beta_n \). Without loss of generality, \( C \) and \( S \) represent, Cosine and Sine functions in the following matrix:

\[
A_n = \begin{bmatrix}
C\theta_n & C\beta_n & -S\theta_n & C\theta_n S\beta_n & C\theta_n S\beta_n dd_n + a_n \\
C\alpha_n S\theta_n & C\alpha_n & C\alpha_n S\theta_n & C\alpha_n S\theta_n S\beta_n & C\alpha_n S\theta_n S\beta_n dd_n - S\alpha_n C\beta_n dd_n \\
S\alpha_n S\theta_n & C\alpha_n & S\alpha_n S\theta_n & S\alpha_n S\theta_n S\beta_n + C\alpha_n C\beta_n & S\alpha_n S\theta_n S\beta_n + C\alpha_n C\beta_n dd_n \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
\]

### 3.11 Dynamic Robot Calibration

The dynamic parameters of a robot include its mass, three components of its center of mass vector and six components of its inertia tensor. A great deal of work has been done regarding the identification of dynamic parameters. Mukerjee, and Ballard [47] presented an identification procedure using sophisticated multi-force and torque sensors. Olsen, and Bekey [48] presented another approach in which they considered noise and excitation effects. Neuman et al. [49] and [50] describe a noise free joint/link/load parameter estimation procedure using high resolution resolvers and tachometers. Atkenso et al. [51] used a direct drive robot to increase the accuracy by eliminating the effects of transmission losses due to gear backlash.
However, their study did not account for joint friction. Direct experimental measurement of the mass and inertial parameters of a 6-link robot is reported in the work of Armstrong et al. [52]. Armstrong [53] continued his work on modelling joint friction and the effect of joint excitation on parameter estimation. Megahed et al. [54] have conducted a preliminary study for the identification of the inertial parameters of a 5-axis ASEA robot arm using Newton-Euler formulation. Ozaki et al. [55] presented an identification method for dynamic models of robots using sequential estimation which reduces the parameter error propagation and also the necessary calculations. Canudas de Wit and Aubin [56] dealt with the problem of improving the parameter identifiability properties of robot models through the exploitation of an information matrix using a sequential hybrid estimation algorithm. The algorithm facilitated the selection of a high excited identification sequence, which improved parameter identifiability. Zomaya [57] presented an algorithm for identifying the dynamic parameters of a robot arm using Lagrange-Euler formulation and a recursive least-square technique. Nicosia, and Tomambe [58] focused on the estimation of inertial parameters for robots having elastic joints by asymptotic observers. ElMaraghy, and Johns [120] developed a model for inherent compliance in a SCARA robot used for peg-in-hole insertion to improve assembly performance. Experimental procedures and numerical solutions methods for the SCARA compliance model were presented by ElMaraghy and Johns [121]. Raucet et al. [59] presented a new approach for estimating the barycentric parameters of a robot arm which required only the measurements of reactions, positions, velocities and accelerations at the bedplate. Gautier, and Khalil [60] presented a method to generate exciting identification trajectories to minimize the effect of noise and error modeling on estimated parameters. Goldenberg et al. [61] presented a method to identify
the inertial parameters of a closed kinematic chain robotic arm. Lin [62] introduced a novel method for estimating the inertial parameters using an extended Kalman filter.

In this work, our approach deals with the kinematic as well as the dynamic variations in the robot parameters whereby a new method of calculating the regressor is presented. Unlike the previously mentioned approaches, it does not require explicit identification of the robot parameters, instead it updates the regressor and robot parameters as a lump variable. Moreover, in this work, an alternative controller is designed to compensate for the varying parameters and thus eliminate the need for the correction module in robot off-line programming software.

3.12 Errors in Robot Kinematic Model

Azadivar [63] and Paul [64] have demonstrated that the location of a manipulator is affected by kinematic parameters. However, the actual location is often somewhat different from the desired location because of errors in the kinematic parameters. Errors in the geometric link parameters are due to the variability in the machining of the links. Joint error and positional error are defined as the deviations from the desired (nominal) joint value and the corresponding position, respectively. They are represented in terms of the differential joint variable vector, $\Delta q$, and differential positional vector, $\Delta d$. The relationship between these two vectors is described by the Jacobian matrix, $J$, which is assumed to be non-singular Paul [64] as follows:

$$\Delta d = J \Delta q$$ (3.7)
For an, \( n \), DOF manipulator, \( \Delta q \), is an, \( n \times 1 \), vector which consists of the, \( n \), differential joint variables. The differential vector, \( \Delta d \), is always a, \( 6 \times 1 \), vector. The jacobian matrix is then a, \( 6 \times n \), matrix and it represents the infinitesimal displacement of the end effector due to infinitesimal change in a joint variable. If the positional error due to the variation in kinematic parameters is known as a range then the joint error representing these variations may be given by:

\[
\Delta q = J^{-1} \Delta d
\]  
(3.8)

3.13 A Modified Regressor and Controller Expressions

3.13.1 Introduction

According to Atkeson et al. [51] the robot regressor can be expressed as:

\[
M(q) \ddot{q} + C(q, \dot{q}) + g(q) = Y(\dot{q}, \ddot{q}, q) p = \tau
\]  
(3.9)

where, \( q, \dot{q}, \ddot{q} \in R^n \), denote, respectively, the joint position, velocity and acceleration, while, \( M(q), C(q, \dot{q}) \in R^{n \times n} \), are the inertia and centrifugal matrices; \( g(q) \in R^n \), is the gravitation force. The left hand side of the above equation represents the rigid robot dynamics while the right hand side is the input torque, \( \tau \in R^n \). The robot dynamics are linearized with respect to an \( m \) dimensional vector of parameters, while \( Y(q, \dot{q}, \ddot{q}) \in R^{n \times m} \).

The first regressor-based adaptive controller was developed by Craig et al. [65] and Craig [66]; however, it required acceleration measurement to compute the regressor.
Slotine, and Li [73] remarkably came up with a regressor that does not need such measurement but relies only on the velocity and acceleration of the reference trajectory, \( Y(q, \dot{q}, v, a) \). They also established the stability of their adaptive controller.

Despite its popularity, the Slotine-Li regressor is not easy to compute, the major difficulty is the multiple forms the, \( C(q, \dot{q}) \), may have. While all possible forms of, \( C(q, \dot{q}) \), satisfy:

\[
\begin{align*}
\mathbf{s}^T \left[ \mathbf{M} - 2C(q, \dot{q}) \right] \mathbf{s} &= 0 \\
\end{align*}
\tag{3.10}
\]

when, \( s = \dot{q} \), only one, \( C(q, \dot{q}) \), for an arbitrary, \( s \neq 0 \), this fact may be demonstrated according to the two-link planar robot example in [27] where:

\[
M(q) = \begin{bmatrix}
(l_1 c_2 + l_2) l_2 m_2 + l_1^2 (m_1 + m_2) & l_1^2 m_2 + l_1 l_2 c_2 m_2 \\
l_2^2 m_2 + l_1 l_2 c_2 m_2 & l_2^2 m_2 \\
\end{bmatrix}
\]

\[
C_1(q, \dot{q}) = l_1 l_2 m_2 \begin{bmatrix}
-s_2 \dot{q}_2 & -s_2 \left( \dot{q}_1 + \dot{q}_2 \right) \\
s_2 \dot{q}_1 & 0 \\
\end{bmatrix}
\text{and}
\]

\[
C_2(q, \dot{q}) = l_1 l_2 m_2 \begin{bmatrix}
-2s_2 \dot{q}_2 & -s_2 \ddot{q}_2 \\
s_2 \dot{q}_1 & 0 \\
\end{bmatrix}
\tag{3.11}
\]

In this case, \( C_1(q, \dot{q}) \), and, \( C_2(q, \dot{q}) \), are two possible forms of, \( C(q, \dot{q}) \), but while, \( C_1(q, \dot{q}) \), satisfies equation (3.10) for an arbitrary, \( s \neq 0 \), \( C_2(q, \dot{q}) \), satisfies equation (3.10) only for, \( s = \dot{q} \).
Generally,

\[
c_{kj} = \sum_{i=1}^{n} \left[ \left( \frac{1}{2} \right) \left( \frac{\partial m_{kj}}{\partial q_i} + \frac{\partial m_{ki}}{\partial q_j} - \frac{\partial m_{ij}}{\partial q_k} \right) \right]^2 \tag{3.12}
\]

where, \(m_{ij}\) is the \(ij\)th element of \(M(q)\), otherwise Slotine-Li stability proof is not valid for the resulting controller. That is why there is no reported algorithm to compute the Slotine-Li regressor for a general, \(n\), link manipulator.

Following Spong, and Vidysagar [5] an alternative robot dynamic expression is adopted, the final form for the model and the novel expression of the Slotine-Li control law is shown in the following section.

**Modified Model and Control Law Expressions**

An alternative model, according to Spong, and Vidysagar [5], may be expressed as follows:

\[
\sum_{i=1}^{n} \left( J_{h_i}^T(q) \right) \left( \begin{bmatrix} m_i \left( \dot{v}_{h_i} + g \right) \\ I_{h_i} \times \omega_i + \omega_i \times I_{h_i} \omega_i \end{bmatrix} \right) = \tau \tag{3.13}
\]

where, \(J_{h_i}^T(q)\), denotes the jacobian with respect to the center of the \(i\)th link, \(g\), is the gravitational acceleration vector, \(m_i\), is the mass of the \(i\)th link, \(v_{h_i}\), is the linear velocity of the center of the \(i\)th link, \(\omega_i\), is the angular velocity of the \(i\)th link, and, \(I_{h_i}\), is the inertia matrix of the \(i\)th link with respect to the center of mass of the \(i\)th link. The two velocity vectors, \(v_{h_i}\), and, \(\omega_i\), are related to the joint velocity, \(\dot{q}\), by:

\[
\begin{bmatrix}
v_{h_i} \\
\omega_i
\end{bmatrix} = J_{h_i}^T(q) \dot{q} \quad 1 \leq i \leq n \tag{3.14}
\]
For a general $n$-link robot with all rotational joints, the $j$th column of, $J_{h_i}^T(q)$, is given by [28]:

\[
\begin{bmatrix}
    z_{j-1} \times ((o_i - o_{i-1}) + h_i) \\
    z_{j-1}
\end{bmatrix}
\text{ for } j < i \quad \text{and} \quad \begin{bmatrix}
    0 \\
    0
\end{bmatrix}
\text{ for } j \geq i
\]

(3.15)

where, $z_{j-1}$, is unit vector representing the pivot axis of the $(j-1)$th joint, $o_i$, is the origin of the $i$th joint frame, $h_i$, is a vector pointing from the origin of the $i$th frame to the center of mass of the $i$th link. Throughout the rest of this work, the modified Denavit-Hartenberg [46] convention is used to establish the $n$ coordinate frames. A more convenient approach is to define the jacobian matrices with respect to the origins of the $i$th link coordinate frame system instead of the center of mass of the $i$th link, Spong, and Vidyasagar [5]. As a result the modified version becomes:

\[
\sum_{i=1}^{n} \left( J_{h_i}^T(q) \begin{bmatrix}
    m(v_{h_i} + g) \\
    I_{h_i} \times \omega_i + \omega_i \times I_{h_i} \omega_i + m_i [h_i \times (v_{h_i} + g)]
\end{bmatrix} \right) = \tau
\]

(3.16)

According to Spong, and Vidyasagar [5] the linear acceleration at the center of mass (pointed to by the vector $h_i$) can be expressed as follows:

\[
v_{h_i} = v_i + \omega_i \times h_i + \omega_i \times (\omega_i \times h_i)
\]

(3.17)

It then follows that:

\[
m(v_{h_i} + g) = m_i(v_i + g) + S^T(d_i) \omega_i + \omega_i \times (\omega_i \times d_i)
\]

(3.18)

where, $d_i = m_i h_i \in R^3$, $S(x) \in R^{3 \times 3}$, is a skew-symmetric matrix such that, $S(x)y = S^T(y)x = x \times y$, for any $x, y \in R^3$. 
A similar principle enables one to write:

\[
I_h \dot{\omega}_i + \omega_i \times I_h \omega_i + m_i \left[ h_i \times \left( v_i + g \right) \right] - I_i \dot{\omega}_i + S(d_i) \left[ v_i + g \right] + I_i \omega_i
\]  

(3.19)

As a result, the alternative robot dynamic equation can be easily achieved by substituting equation (3.18) and equation (3.19) into equation (3.13). Then the alternative dynamic model can be shown to be:

\[
\sum_{i=1}^{n} \left( J_i^T(q) \right) \begin{bmatrix}
\left( \dot{v}_i + g \right) \\
\dot{\omega}_i \\
\omega_i \times I_i \omega_i
\end{bmatrix} + \Gamma_i \begin{bmatrix}
\omega_i \times (\omega_i \times d_i)
\end{bmatrix} = \tau
\]  

(3.20)

where

\[
\Gamma_i = \begin{bmatrix}
m_i I_{3 \times 3} & S^T(d_i) \\
S(d_i) & I_i
\end{bmatrix} \in R^{6 \times 6}
\]  

(3.21)

According to Slotine, and Li [73], a control law can be synthesized as follows:

\[
M(q) \ddot{q}_r + C(q, \dot{q}) q_r + g(q) - K_v s = \tau
\]  

(3.22)

where, \( s = \dot{q} - \dot{q}_r \).

The closed loop dynamic equation can be derived by substituting equation (3.22) into equation (3.9), which results in:

\[
M(q) \ddot{s} + C(q, \dot{q}) s + K_v s = 0
\]  

(3.23)
Therefore, an alternative form of equation (3.20) is suggested to be:

\[ \sum_{i=1}^{n} \left( J_i^T(q) \right) \left[ \Gamma_i \left[ \begin{array}{c} \dot{v}_{i}^{\text{ref}} + g \\ \dot{\omega}_{i}^{\text{ref}} \\ \omega_i^{\text{ref}} \end{array} \right] + \left[ \begin{array}{c} \omega_i^{\text{ref}} \times (\omega_i \times d_i) \\ S(\omega_i) I_i \omega_i^{\text{ref}} \end{array} \right] \right] - K_s = \tau \] (3.24)

where, \( \omega_i^{\text{ref}}, \dot{\omega}_i^{\text{ref}}, \) and, \( \dot{v}_i^{\text{ref}}, \) are variables obtained when \( \dot{q}_{\text{ref}}, \ddot{q}_{\text{ref}}, \) are substituted for, \( \dot{q}, \ddot{q}, \) in the following definitions:

\[
\begin{bmatrix}
\dot{v}_i^{\text{ref}} \\
\dot{\omega}_i^{\text{ref}} \\
\omega_i^{\text{ref}}
\end{bmatrix} = J_i(q) \dot{q}_{\text{ref}} \quad \text{and} \quad \begin{bmatrix}
\ddot{v}_i^{\text{ref}} \\
\ddot{\omega}_i^{\text{ref}} \\
\omega_i^{\text{ref}}
\end{bmatrix} = J_i(q) \ddot{q}_{\text{ref}} + \frac{dJ_i(q)}{dt} \dot{q}_{\text{ref}}
\] (3.25)

In the case of unknown parameters we have:

\[ M(q) \ddot{q}_r + C(q, \dot{q}) \dot{q}_r + g(q) = D(\ddot{q}_r, q_r, \dot{q}_r, q_r) + Y(\dddot{q}_r, q_r, \ddot{q}_r, \dot{q}_r) p \] (3.26)

where,

\[ D(\ddot{q}_r, q_r, \dot{q}_r, q_r) = \sum_{i=1}^{n} \left( J_i^T(q) \right) \left[ \Gamma_i \left[ \begin{array}{c} \dot{v}_{i}^{\text{ref}} + g \\ \dot{\omega}_{i}^{\text{ref}} \\ \omega_i^{\text{ref}} \end{array} \right] + \left[ \begin{array}{c} \omega_i^{\text{ref}} \times (\omega_i \times d_i) \\ S(\omega_i) I_i \omega_i^{\text{ref}} \end{array} \right] \right] \] (3.27)

is computed for the known part of the first (n-1) links, while

\[ Y(\dddot{q}_r, q_r, \ddot{q}_r, \dot{q}_r) p = \left( J_n^T(q) \right) \left[ \Gamma_n \left[ \begin{array}{c} \dot{v}_{n}^{\text{ref}} + g \\ \dot{\omega}_{n}^{\text{ref}} \\ \omega_n^{\text{ref}} \end{array} \right] + \left[ \begin{array}{c} \omega_n^{\text{ref}} \times (\omega_n \times d_n) \\ S(\omega_n) I_n \omega_n^{\text{ref}} \end{array} \right] \right] \] (3.28)

is the uncertain part where the linearity in dynamic parameters is exploited. Since the product, \( I\omega_n^{\text{ref}}, \) may be expressed as:
\[ I \omega_{\text{ref}}^r = \begin{bmatrix} i_{x\text{ref}} & i_{y\text{ref}} & i_{z\text{ref}} \\ i_{x\text{ref}} & i_{y\text{ref}} & i_{z\text{ref}} \\ i_{x\text{ref}} & i_{y\text{ref}} & i_{z\text{ref}} \end{bmatrix} \begin{bmatrix} \omega_x^r \\ \omega_y^r \\ \omega_z^r \end{bmatrix} \] 

(3.29)

then,

\[ I \omega_{\text{ref}}^r = \begin{bmatrix} \omega_{x\text{ref}} & 0 & 0 & \omega_{y\text{ref}} & \omega_{z\text{ref}} & 0 \\ 0 & \omega_{x\text{ref}} & 0 & \omega_{y\text{ref}} & 0 & \omega_{z\text{ref}} \\ 0 & 0 & \omega_{x\text{ref}} & \omega_{y\text{ref}} & \omega_{z\text{ref}} & 0 \\ 0 & i_{x\text{ref}} & i_{y\text{ref}} & i_{z\text{ref}} & 0 & 0 \end{bmatrix} - \Phi (\omega_{\text{ref}}^r) \Phi \] 

(3.30)

Substituting, \( I_n \omega_n^r = \Phi_n \), and, \( I_n \omega_n^r = \Phi_n \), in equation (3.28) and using some algebraic manipulations it may be written as:

\[ Y_P = J_n^T(q) \begin{bmatrix} A \\ B \end{bmatrix} \] 

(3.31)

where

\[ \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} m_n (\dot{\omega}_n^r + g) + [S(\dot{\omega}_n^r) + S(\omega_n^r)S(\omega_n^r)] R_n d_n \\ S^T(\dot{\omega}_n^r + g)(R_n d_n) + [R_n Q(\dot{\omega}_n^r) + S(\omega_n^r)R_n Q(\omega_n^r)] \Phi_n \end{bmatrix} \] 

(3.32)

The derivation is complete and this way instead of computing the regressor itself we could compute, \( Y_P \), and it is valid for any \( n \)-link robot with open chain kinematic structure.
3.14 Workcell simulation

The workcell components were modelled completely in the simulation environment from CAD primitives, since CAD models for the robot, fixture or workpiece were not available a priori. This was an extensive and time-consuming effort. The kinematics of these devices were built to match the kinematic models of the robot and fixture supplied by Trellis, and embedded in the controller configuration files. The kinematic representation used is the DH parameters, Denavit, and Hatrenberg [46]. The joint, speed and acceleration limits were matched with their corresponding values form the robot controller as well as the motion interpolation routines. The SmartProx TM sensors were modelled physically in the simulation environment with a button to represent their functionality. This could be called in the assembly program to get their readings and correct the assembly process. The assembly process simulation was programmed using Deneb’s graphical simulation language (GSL) which is generic and takes advantage of the considerable functionality built into the software.

3.15 Implementation of robot base and tool calibration

The purpose of the robot base and tool calibration is to identify the best fit values for base \([X,Y,Z,Yaw,Pitch,Roll]\) and tool \([X,Y,Z]\) parameters based on an experimental procedure using the real robot and tool. The calibration technique relies on selecting a robot base and calibration fixture, and two paths of tag points in the workcell and simulation environment, respectively. The fixture represents a well-known coordinate frame of reference in the workcell, and the robot base is adjusted relative to it.
The first set of tag points represents the robot mount plate position, as seen by the robot controller. This path is generated at the robot mount plate while moving the tool tip to a set of measurement locations in space at different orientations of the robot wrist. This path is uploaded in the simulation environment. A corresponding set of tag points is generated in the simulation environment by moving the robot mount plate to the same pose as the previous tag points. This path is attached to the robot base plate.

The second set of tag points (at the two poles attached to the fixture representing the calibration fixture) represents the tool tip positions in the measurement locations that have been described in the previous paragraph. These tag points are attached to the real fixture as an external coordinate measuring system. The coordinates of the points in the fixture frame are determined by the transformation between the calibration fixture (two poles coordinate frame) and the external coordinate measuring system.

The calibration requires an initial estimate of the robot base transform parameters in the simulation environment with respect to the fixture coordinate system. Also, an estimate for the tool tip transformation parameters, with respect to the robot mount plate is needed. Based on the estimated parameters the robot base \([X,Y,Z,Yaw,Pitch,Roll]\) and tool \([X,Y,Z]\) transformation parameters are adjusted to obtain the best fit values of the parameters. The algorithm works by minimizing the mean squared positional error between the corresponding points on the two paths. The technique used in the algorithm is the same as part calibration discussed earlier (Section 3.8) in equation 3.4. The robot base and tool calibration set up in the simulation environment is shown in Figure (3.4).

Upon convergence between the real and estimated parameters, the results are:
1- Number of iterations to convergence = 4

2- Number of fitting tag points = 7

3- Root mean square fitting error before calibration = 6.17371 mm

4- Root mean square fitting error after calibration = 0.2 mm

5- Maximum uncertainty in the base Yaw, Pitch, Roll = 0.0075 deg

6- Robot base coordinate frame before calibration =

   [20.0980, 18.9998, 985.680, -1.0352, 1.59989, -177.909]

7- Robot base coordinate frame after calibration =

   [17.6639, 36.7471, 983.975, -0.73246, 1.91, -177.909]

8- Initial estimate of tool coordinate frame position [-70.0, -10.0, 250]

9- Final tool coordinate frame position [-68.4223, -16.2461, 245.499].

3.16 Part calibration (setup)

Part calibration is intended to adjust the pose of a workpiece in the simulation environment based on measurements from the real workcell. The purpose is to match the part pose [X, Y, Z, Yaw, Pitch, Roll] with respect to the robot, with the corresponding pose in the real workcell. After this calibration is achieved the robot programs for the assembly task, are updated accordingly, and downloaded to the actual robot in the real workcell.
3.17 Implementation of part calibration

This technique is based on adjusting the pose of the part based on the measurement of multiple points. The points are input as tag points on two paths to the simulation environment. The first path represents the CAD data pose information of these points, while the other represents the corresponding experimental robot tool pose information at these points. The algorithm used is Levenberg-Marquardt [45] nonlinear least squares method used earlier to determine the transformation that minimizes the error between the set of points on two previously specified paths. The least squares routine incorporated requires at least three non-collinear points on each path. After calibration, the part in the simulation environment is transformed to match the actual part (tag points) in the real workcell.

Upon convergence between the real and initial estimated parameters, the results are summarized as follows:

1- Number of iterations to converge = 3
2- Number of fitting tag points = 4
3- Root mean square fitting error = 1.44 mm
4- Maximum uncertainty in the part Yaw, Pitch, Roll = 0.025 deg.

3.18 System operation

The robot is commanded to place an ideal part on the fixture. An ideal part is a part known previously to be within the specified manufacturing tolerances. This part is then gaged
(measurements of specific points on the surface of interest are recorded using SmartProx\textsuperscript{TM} sensors). The robot is commanded to pick the same part but from a different location on the nest where the part is lying. This is typical of a shop floor environment since the sensors are not used to locate the part relative to the nest, and hence, to the world coordinates frame. Another gage cycle is performed, and the measurements are recorded. The idea behind the correction algorithm is to find translation and rotation parameters, depending on the number of degrees of freedom (up to six) that, when applied to the measurement coordinate, will minimize the error between the transformed measurement coordinates and the ideal part coordinates in the direction of the normal to the surface being measured at the measurement points. Since the transformation of the actual part with respect to the ideal part is now known, the actual part coordinates frame relative to the world coordinates frame can be obtained. The simulation environment is then updated with the modified pose transform. The simulation environment now matches the real workcell and off-line programming for the assembly task is pursued. The program is downloaded to the real cell for verification. For on-line operational mode the robot must be able to accommodate the changes in the assembly operation due to variation in the part location relative to the fixture. This is the driving force behind developing a communication protocol using the trellis motion controller Application Program Interface (API) libraries to pass the actual part pose parameters to the robot. This is achieved through a supervisory module that slaves the robot to the sensors' (\textit{SmartProx}\textsuperscript{TM}) feedback. The interface program communicates with the sensors' operational software for utilizing its data structures, sensors setup, laser initialization and accessing sensor readings. The communication between the supervisory module and Trellis motion controller is through the TCP/IP networking protocol using sockets, Stewart \textit{et al.} [68]. A schematic of the integrated system is shown in Figure (3.5).
3.19 Results

The proposed adaptive assembly system was implemented on an automotive Body-In-White subassembly operation, namely mounting a hinge on a fender. During the measurements, the fender (main assembly) is located on the fixture used in the experimental set up as shown in Figure (3.3). The measurements were performed using three SmartProx™ (SP1) laser range sensors and a set of two holes on both the hinge and the fender. Only the X-Y coordinates related to the hole axis location are measured. The axis location is measured using a vision system composed of a CCD camera, a frame
grabber and vision software, Zghal et al. [67]. The vision system was used to test the assembly operation before and after the fender was mislocated, respectively. Meanwhile, the holes on the fender are equipped with retro-reflector material from the sensors’ side enabling the sensors to identify the, distance between the fender and the sensors at the specified measuring points, along the sensors normal axis. The robot was programmed to mount the hinge on the fender in the desired pose. The fender was then deliberately mislocated in the fixture to simulate errors which occur during the part loading process. The supervisory controller commanded the robot to mount the hinge. In each cycle, the supervisory controller slaved the sensors to get part actual position, which was then fed back to the controller and compared to the desired position. The variation between the desired and actual positions was sent to the robot to modify the end point pose in such a way as to maintain the position of the hinge relative to the fender. The experimental setup, shown in Figure (3.3), was used to test the performance of the proposed part calibration module in the assembly operation. The assembly operation was achieved with an accuracy of +/- 0.25 mm, as calculated from pictures taken by the vision system. This represented the worst case scenario since the fender was mislocated on the fixture to the limits of the sensor field of view. For clarity, this accuracy represents the difference in the hole centre locations on both the fender and hinge, as shown by the plus signs in Figure (3.9). The typical accuracy in the automotive body-in-white assembly processes are in the range from 0.8 to 1.0 mm, Yamauchi [116]. From the results discussed above, it is clear that the proposed adaptive assembly system is quite promising in improving assembly process performance.
3.20 Conclusion

An adaptive assembly system using sensor feedback was designed and implemented. The integration of sensors for process and robotic control makes it possible to design a robotic-assembly cell capable of dealing with inconsistencies in the process itself.

The supervisory control system offered many advantages to standard solutions where no sensors were utilized. The following summarizes the main advantages:

1- Increased flexibility because the system was capable of adapting to the main assembly pose discrepancies.

2- The integrated system, where the sensors and the robot controller communicated, created the condition needed for more consistent assembly quality.

3- The ability to calibrate the cell guaranteed a better match between the off-line programming environment and the real cell, which, in turn, resulted in better assembly process accuracy.

4- The ability to simulate the sensor enhanced the possibility of updating the simulation environment to match the real workcell. The purpose was the adjustment of off-line programs to compensate for the real workcell environment inaccuracies.
Figure 3.6: Schematics of robot calibration system

Figure 3.7: Probes used for calibration in the actual workcell
Figure 3.8: Probes used for calibration as represented in the simulation environment

Figure 3.9: Verification of assembly operation
Adaptive control of a single-link manipulator using feedback linearization

4.1 Introduction

Geometric modelling systems are rapidly replacing manual drafting techniques for defining the geometry of mechanical parts and assemblies during the design process. This makes possible the development of software tools to aid manufacturing engineers in setting up assembly workcells, thus, integrating design and manufacturing. Robotic assembly can be automatically programmed based on geometric models of assembly. However, when programming robotic assembly workcells automatically, the accuracy and repeatability of robots’ motion must be extremely high, so that parts are placed within thousands of an inch of the desired position and orientation. For this reason, it is critical that the entire robotic assembly workcell be precisely calibrated. This type of calibration is known as kinematic calibration, it deals with static errors and has been extensively investigated in the literature, Mooring et al. [24]. The errors, which this thesis address, are the dynamic errors. They affect the tracking performance as well as robotic tool end position. Unmodelled manipulator dynamics and parameter variations are a major source of dynamic errors. In chapters 4 and 5 adaptive and robust control techniques are proposed to minimize the tracking errors due to parameter variations and unmodelled dynamics. Meanwhile in chapter 6, adaptive control techniques are proposed to ensure both tracking and force errors, due to parameter variations and unmodelled dynamics, asymptotically converge to
zero. Simulation results are presented to demonstrate the performance of the proposed controllers.

Control of rigid-body robot manipulators is a well-studied subject about which there have been numerous contributions in the literature. An overwhelming majority of the available controllers are based on the assumption that the actuator dynamics are negligible [69-73]. This assumption reduces the dynamic model of the robot and facilitates the design of controllers. As a result of this simplification, unmodelled disturbances exist in the robot control system which affect the tracking and positioning of the robot. Although there are several methods of making a controller robust with respect to the unmodelled dynamics, the performance of the controller is not as expected, meaning that the tracking errors are bounded but do not converge to zero, Reed, and Ioannou [74].

The negligence of the motor dynamics is a trade-off between model accuracy and controller simplicity. The motivation for and evaluation of such a trade-off are, in a sense, technology dependent. In the early days of robotics, a linearized model was deemed acceptable in the design of controllers and performance in a specific bandwidth, Dubowsky, and DesForges [70]. As a further step, a nonlinear rigid-body dynamic model has been adopted, Luh et al. [69] and Lim, and Eslami [71]. The rapid development of computer engineering allowed the use of a linearized model in the dynamic parameters to be used to estimate the robot inertia parameters and the design of model based adaptive controllers, Craig [72], Slotine, and Li [73], Garceia, and D’Attellis [80] and Krstic et al. [81]. Recently, the actuator dynamics, joint flexibility and link elasticity have been included in the robot dynamic models, Spong, and Vidyasagar [5]. In the work by Jankowski and ElMaraghy [118] and [123], an inverse dynamics approach has been proposed to address the tracking problem for flexible joint robots. The main challenge in using such method is
that analytical analysis was not presented for the closed loop stability, or how it is affected by the observer design, respectively.

Since many industrial robots are driven by electric motors, the motor dynamics are the major source of unmodelled high-frequency disturbance [74]. The production demand for shorter cycle times requires the robot to move faster, and thus, the high frequency disturbances due to unmodelled dynamics can no longer be neglected since it operates in the controller bandwidth [74]. The importance of including the motor dynamics in the robot dynamic model to improve the accuracy of robot tracking and positioning was studied by Goor [75]. Inclusion of the actuators in the robot dynamic model complicates the controller structure, because the system is described by third-order differential equations. In the early studies of rigid robots control including actuator dynamics, Tarn et al. [76] and Beekmann, and Lee, [77], full knowledge of the actuator parameters was assumed. Ge and Postlethwaite [78] proposed an adaptive controller for robots including motor dynamics. It required acceleration feedback to guarantee stable tracking. Recent work by Lahdhiri and ElMaraghy [117] has dealt with optimal control of a 2 link flexible joints robot, using LQG/LTR techniques. Although, the flexible joint robots initiate frequencies at relatively low frequency (i.e. 14 Hz), the controller bandwidth of 1 Khz includes a typical motor dynamics frequency of 200-500 Hz. In this work, however the proposed technique addresses the problems of parameter variations and unmodelled actuator dynamics using an adaptive feedback linearization technique. The new optimal adaptive controller developed here is based on feedback linearization. It does not need acceleration feedback, nor does it assume full state is available for measurement, but it does require an observer. Naturally, it does not assume exact knowledge of either robot or actuator parameters. The optimality is based on the minimization of a performance index which would be possible if we could find a solution to the Hamilton-Jacobi equation.
4.2 Exact Linearization Techniques

In this section the input-output linearization of single-input nonlinear systems is discussed. By applying input-output linearization techniques, a linear differential relation between the output, \( y \), and a new input, \( \nu \), is obtained, by differentiating the output repeatedly until the input, \( u \), appears and then designing \( u \) to cancel the nonlinearity. This is mathematically presented for a single-input, single-output system in, Sastry, and Isidori [79] as follows:

\[
\dot{x} = f(x) + g(x) u \\
y = h(x) \tag{4.1}
\]

with, \( x \in \mathbb{R}^n; (f, g, h) \), smooth functions. Differentiating, \( y \), with respect to time, we get:

\[
\dot{y} = L_f h + L_g h u \tag{4.2}
\]

where, \( L_f h \), and, \( L_g h \), are the Lie derivatives of, \( h(x) \), w.r.t, \( f, g \), respectively and may be represented by \( L_f h = \nabla h f = \frac{\partial h}{\partial x} f \). If, \( L_g h \neq 0 \forall x \in \mathbb{R}^n \), then the control law of the form, \( \alpha(x) + \beta(x) \nu \), specifically:

\[
u = \frac{(-L_f h + \nu)}{L_g h} \tag{4.3}
\]
yields the linear system:

\[
\dot{y} = \nu \tag{4.4}
\]

If, \( L_g h = 0 \), then we differentiate further until we find the smallest integer, \( \gamma \), for which, \( L_g L_f^{\gamma-1} h(x) \neq 0 \forall x \in \mathbb{R}^n \), then the control law becomes:
\[ u = \frac{(-L_f \gamma h + \nu)}{L g L_f \gamma^{\gamma-1} h} \]  

(4.5)

where, \( \gamma \), is the system relative degree. If the control law in, Sastry, and Isidori [79] is compared with the form, \( u(x) = \alpha(x) + \beta(x) \nu \), we deduce that:

\[ \alpha(x) = \left[ \frac{-L_f \gamma h}{L g L_f \gamma^{\gamma-1} h} \right], \beta(x) = \left[ \frac{1}{L g L_f \gamma^{\gamma-1} h} \right] \]  

(4.6)

The above control law, when applied to the system equation (4.1) yields:

\[ y^\gamma = \nu \]  

(4.7)

The above-mentioned methodology is applied to transform the nonlinear systems into an input-output linearizable system, bearing in mind that the system is feedback linearizable. This, in turn, means that the nonlinear system has to satisfy the following conditions.

a) The vector fields \( \{ g, ad_f g, \ldots, ad_f^{n-1} g \} \) are linearly independent in \( \Omega \)

b) The set \( \{ g, ad_f g, \ldots, ad_f^{n-2} g \} \) is involutive in \( \Omega \)

\( \Omega \) is the region where the above conditions hold. While, \( ad_f^i g = [f, ad_f^{i-1} g] \), \( [f, g] = \nabla g f - \nabla f g \), and, \( ad_f^0 g = g \). This means that if the Lie bracket of \( f \) and \( g \), \( [f, g] \), can be expressed as a linear combination of \( f \) and \( g \), this condition is called the involutivity condition on the vector fields \( \{ f, g \} \).

where the normal form is represented by applying the transformation, \( z = \phi(x) \), to the system equations (4.1) to get a linear system in the new coordinates represented by:

\[ z = Az + bv \]
\[ y = Cz \]  

Where

\[
\begin{bmatrix}
\phi_1(x) \\
\phi_2(x) \\
\vdots \\
\phi_\gamma(x)
\end{bmatrix} = \begin{bmatrix}
h(x) \\
L_fh(x) \\
\vdots \\
L_f^{\gamma-1}h(x)
\end{bmatrix}
\]

and,

\[
A = \begin{bmatrix}
0 & 1 & 0 & \cdots & 0 \\
0 & 0 & 1 & \cdots & \cdots \\
\vdots & \vdots & \vdots & \ddots & \cdots \\
0 & 0 & 0 & \cdots & 1 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}, \quad b = \begin{bmatrix}
0 \\
0 \\
\vdots \\
0 \\
1
\end{bmatrix}, \quad C = \begin{bmatrix}
1 & 0 & \cdots & 0
\end{bmatrix}
\]

4.3 Adaptive control of Linearizable systems

The drawback of exact feedback linearization is that, in order to cancel the nonlinear terms, the system parameters that constitute the, \( f \), and, \( g \), nonlinear functions should be available. This assumption is always difficult to fulfill in practical implementations. Hence, the motivation for adopting adaptive control techniques to achieve asymptotic cancellation.

According to Sastry and Isidori [79] for SISO systems we may represent the single-link robot system as follows:

\[
f(x) = \sum_{i=1}^{n_1} \theta_i^1 f_i(x) \quad (4.10)
\]

\[
g(x) = \sum_{j=1}^{n_2} \theta_j^2 g_j(x) \quad (4.11)
\]
where, $\theta_i$ and $\theta_j$, represent the system unknown parameters for the $f_i(x)$, and $g_j(x)$, known functions. The parameters used for the single-link system are shown in Table 2. The estimates at time $t$ have the form:

$$\hat{f}(x) = \sum_{i=1}^{n_1} \hat{\theta}_i^1(t)f_i(x)$$

$$\hat{g}(x) = \sum_{j=1}^{n_2} \hat{\theta}_j^2(t)g_j(x)$$

(4.12) (4.13)

where $\hat{\theta}_i$, and $\hat{\theta}_j$, represent the estimates of the parameters, $\theta_i$, and $\theta_j$, respectively. The control law then becomes:

$$u = \frac{-\hat{L}_f h + v}{\hat{L}_g h}$$

(4.14)

where $\hat{L}_g h$, $\hat{L}_f h$, are the estimates of $L_g h$, and $L_f h$, respectively.

In case of adaptive tracking for SISO system the control law becomes:

$$u = \frac{-\hat{L}_f h + \dot{y}_{\text{ref}} + k (y_{\text{ref}} - y)}{\hat{L}_g h}$$

(4.15)

While for a MIMO system the control law becomes:

$$u = \hat{A}(x)^{-1} - \left[ \begin{array}{c} \hat{\gamma}_1 h_1 \\ \hat{\gamma}_2 h_2 \\ \vdots \\ \hat{\gamma}_p h_p \end{array} \right] + v$$

$$u = \hat{A}(x)^{-1} - \left[ \begin{array}{c} \hat{\gamma}_1 h_1 \\ \hat{\gamma}_2 h_2 \\ \vdots \\ \hat{\gamma}_p h_p \end{array} \right] + v$$

(4.16)

where $\hat{A}(x)$ is a $p \times p$ matrix defined as follows:
\[
\hat{A}(x) = \begin{bmatrix}
L_g(L_f^{-1}h_1) & \cdots & L_g(L_f^{-1}h_1) \\
\vdots & & \vdots \\
L_g(L_f^{-1}h_p) & \cdots & L_g(L_f^{-1}h_p)
\end{bmatrix}
\]

To estimate, \(\hat{x}\), the observer of Garceia, and D'Attellis [80] was adopted. The observer is represented by:

\[
\dot{\hat{x}} = f(\hat{x}) + g(\hat{x})u(t) + O^{-1}(\hat{x})K_{obs}(y(t) - h(\hat{x}))
\]

(4.16)

where, \(O(\hat{x}(t))\), is defined as:

\[
O(x(t)) = \frac{\partial}{\partial t} \begin{bmatrix}
h(x(t)) \\
L_fh(x(t)) \\
L_f^2h(x(t)) \\
\vdots \\
L_f^{n-1}h(x(t))
\end{bmatrix}
\]

The observer gain, \(K_{obs}\), is chosen to guarantee asymptotic stability of the estimation errors which should typically be 10 times faster than the tracking errors.

### 4.4 Stability of closed-loop system

As we have established in the introduction, the non-linear observer and adaptive controller will be used in a closed loop for trajectory tracking. The idea behind using the adaptive controller is that the system parameters are not exactly known in practice, and thus,
exact cancellation of the nonlinear terms is not possible. At the same time, we need state feedback, but since not all the states are available or feasible to measure, we suggest the use of a nonlinear observer. The output of the system composed of the adaptive controller, nonlinear observer and the plant should track the output trajectory of a reference predefined model.

Substituting equation (4.14) into equation (4.2) we get:

\[ \dot{y} = L_f h + L_g h \frac{L_f h + \nu}{L_g h} \]  

(4.17)

Which may be written as follows:

\[ \dot{y} = \begin{bmatrix} \theta_1^1 & \cdots & \theta_n^1 \\ \vdots & \ddots & \vdots \\ \theta_1^n & \cdots & \theta_n^n \end{bmatrix} \begin{bmatrix} L_f h \\ \vdots \\ L_f h \end{bmatrix} + \begin{bmatrix} \theta_1^2 & \cdots & \theta_n^2 \\ \vdots & \ddots & \vdots \\ \theta_1^n & \cdots & \theta_n^n \end{bmatrix} \begin{bmatrix} L_g h \\ \vdots \\ L_g h \end{bmatrix} \frac{L_f h + \nu}{L_g h} + \nu - \nu \]  

(4.18)

from equation (4.14), \(-\nu\), may be expressed as:

\[ -\nu = -\begin{bmatrix} \hat{\theta}_1^1 & \cdots & \hat{\theta}_n^1 \\ \vdots & \ddots & \vdots \\ \hat{\theta}_1^n & \cdots & \hat{\theta}_n^n \end{bmatrix} \begin{bmatrix} L_f h \\ \vdots \\ L_f h \end{bmatrix} - \begin{bmatrix} \theta_1^2 & \cdots & \theta_n^2 \\ \vdots & \ddots & \vdots \\ \theta_1^n & \cdots & \theta_n^n \end{bmatrix} \begin{bmatrix} L_g h \\ \vdots \\ L_g h \end{bmatrix} \frac{L_f h + \nu}{L_g h} \]  

(4.19)

When equation (4.19) is substituted in equation (4.18) it yields:

\[ \dot{y} = \nu + \Theta^T W \]  

(4.20)

where \( \Theta^T = (\theta - \hat{\theta})^T \), \( W = \begin{bmatrix} w_1 \\ \vdots \\ w_2 \end{bmatrix} \) and \( w_2 = \begin{bmatrix} L_f h \\ \vdots \\ L_f h \end{bmatrix} \frac{L_f h + \nu}{L_g h} \).
Assuming noise-free measurements an appropriate choice for a control law used for tracking is:

\[ v = \dot{y}_m + \partial (y_m - y) \]

which yields the following error equation, where, \( e_t \), is defined by, \( e_t = y - y_m \), and, \( \partial \), is a positive gain:

\[ \dot{e}_t + \partial e_t = \Theta^T W \]  
(4.21)

The difficulty in constructing this signal comes from the fact that, \( \dot{e}_t \), is not available for measurement since it depends on, \( L_f h \), which, in turn, is not available as not all the states are measurable. A nonlinear observer has been used to overcome this difficulty. The observer error equation is as follows:

\[ \dot{\varepsilon} = -K_{obs} \varepsilon \]  
(4.22)

where, \( \varepsilon_o \), is represented by, \( \varepsilon_o = \hat{\beta}(t) - z(t) \), \( z(t) \), was defined earlier as, \( z(t) = \Phi(x(t)) \).

The differential equation for the closed-loop error is given by:

\[ \dot{\eta}(t) = A\eta(t) + B\left( \Theta^T W \right) \]  
(4.23)

where \( \eta(t) = \begin{bmatrix} \varepsilon_o \\ e_t \end{bmatrix} \), \( A = \begin{bmatrix} -K_{obs} & 0 \\ 0 & -\alpha \end{bmatrix} \) and, \( B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \).

A Lyapunov function candidate is

\[ V = \frac{1}{2} \eta^T \eta \]  
(4.24)

Hence, \( V = \frac{1}{2} \eta^T \varepsilon_o + e_t^T e_t \), and is positive definite.
\[
\dot{V} = \frac{1}{2} \varepsilon_o T \varepsilon_o + \frac{1}{2} e_t^T \varepsilon_o + \frac{1}{2} e_t^T \varepsilon_t + \frac{1}{2} e_t^T T
\]

(4.25)

By substituting equations (4.21) and (4.22) in equation (4.25) we get:

\[
\dot{V} = -\left[ K_{obs} e_o ^T e_o \right] - \left[ \alpha e_t ^T e_t \right] + W^T \Theta M (s) \Theta^T \xi
\]

(4.26)

If we substitute, \( \xi = \Theta^T W \), in equation (4.26) we get:

\[
\dot{V} = -\left[ K_{obs} e_o ^T e_o \right] - \left[ \alpha e_t ^T e_t \right] + \xi^T M (s) \xi
\]

(4.27)

Hence,

\[
\dot{V} \leq - K_{obs} \| e_o \|^2 - \alpha \| e_t \|^2 + \lambda_m \| \xi \|^2
\]

(4.28)

Motivated by the linear case, where the augmented error scheme is adopted by Sastry and Bodson [122] we define, \( e_t = M (s) \xi \), then, \( \| \xi \| \leq \text{Sup} \frac{1}{M (s)} \| e_t \| \), \( \| \xi \|^2 \leq \gamma_M^2 \| e_t \|^2 \), and

\[
\dot{V} \leq - K_{obs} \| e_o \|^2 - \alpha \| e_t \|^2 + \lambda_m \gamma_M^2 \| e_t \|^2
\]

(4.29)

bearing in mind \( \eta (t) = \begin{bmatrix} e_o \\ e_t \end{bmatrix} \), then equation (4.29) yields:

\[
\dot{V} \leq - \left( K_{obs} + \alpha - \lambda_m \gamma_M^2 \right) \| \eta \|^2
\]

(4.30)

which may be written as

\[
\dot{V} \leq - \Lambda \| \eta \|^2
\]

(4.31)

The norms used in the previous definitions are Euclidean norms. A sufficient condition for, \( \dot{V} \leq 0 \), is, \( \Lambda > 0 \), or equivalently that, \( \left( K_{obs} + \alpha - \lambda_m \gamma_M^2 \right) > 0 \), which can be guaranteed through an appropriate choice of \( K_{obs} \), \( \alpha \), and, \( M (s) \).
4.5 Optimal control of Linearizable systems

Another drawback of feedback linearization is that a considerable control effort is required to achieve fast and accurate responses. In this section, an optimal control law is derived and implemented for the case of unknown system parameters and the system relative degree, $\gamma$, is equal to the system order, $n$. The performance index chosen in the pursuit to derive the optimal control law is of the form

$$ PI = \int_{0}^{\infty} S[x(t), u(t)] \, dt $$

(4.32)

where, $S$, is a smooth, positive function satisfying $S(0, 0) = 0$. Such a problem has a solution if minimization of the following Hamilton-Jacobi equation has a smooth solution $V(x)$ as follows:

$$ \min \left[ \frac{\partial}{\partial x} V(x) \left(f(x) + g(x)u + S(x, u)\right) \right] = 0, \quad V(0) = 0 $$

(4.33)

If we choose the function, $S(x, u)$, to be:

$$ PI = \frac{1}{2} \int_{0}^{\infty} \left( \phi^T(x) C^T C \phi(x) + \frac{1}{\beta^2(\phi(x))} \left[u - \alpha(\phi(x))\right]^2 \right) \, dt $$

(4.34)

and the observer for estimating, $\hat{x}$, to be represented by equation (4.16). The minimization of equation (4.38) leads to:

$$ \frac{\partial}{\partial x} V(x) \left[ g(x) \right] + \frac{1}{\beta^2(\phi(x))} \left[u - \alpha(\phi(x))\right] = 0 $$

(4.35)

which in turn leads to:

$$ u = \alpha(\phi(x)) - \beta^2(\phi(x)) g(x) T \frac{\partial}{\partial x} V(x) $$

(4.36)

If we substitute equation (4.41) in the Hamilton-Jacobi equation in equation (4.38) we get:
\[
\frac{\partial}{\partial x} V(x) \left[ f(x) + g(x) \alpha(\phi(x)) \right] - \frac{1}{2} \beta^2(\phi(x)) \frac{\partial}{\partial x} V(x) g(x) g(x)^T \frac{\partial}{\partial x} V(x) + \frac{1}{2} \phi^T(x) C^T C \phi(x) = 0, \quad V(0) = 0 \quad (4.37)
\]

One of the fundamental results from dissipative system theory as given by, van der Schaft [86] is the following:

\[
\frac{\partial}{\partial x} V(x) f(x) + \frac{1}{2} \frac{\partial}{\partial x} V(x) g(x) g(x)^T \frac{\partial}{\partial x} V(x)^T + \frac{1}{2} \phi^T(x) C^T C \phi(x) = 0 \quad (4.38)
\]

Where, \( \lambda > 0 \). By equating equations (4.42) and (4.43) we get:

\[
g(x)^T \frac{\partial}{\partial x} V(x)^T = \frac{\alpha(\phi(x))}{\left( \frac{1}{2\lambda^2} + \frac{\beta^2(\phi(x))}{2} \right)} = \frac{2\alpha(\phi(x))}{\left( \frac{1}{\lambda^2} + \frac{\beta^2(\phi(x))}{2} \right)} \quad (4.39)
\]

A possible choice of \( V(x) \), is, \( V(x) = \frac{1}{2} \phi^T(x) K \phi(x) \), where, \( K \), is the unique symmetric, positive-definite solution of the following algebraic Riccati equation:

\[
KA + A^T K - Kbb^T K + C^T C = 0 \quad (4.40)
\]

\[
A = \begin{bmatrix}
0 & 1 & 0 & \ldots & 0 \\
0 & 0 & 1 & \ldots & \ldots \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\vdots & \vdots & \vdots & \ldots & \ldots \\
0 & 0 & 0 & \ldots & 1 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}, \quad b = \begin{bmatrix}
0 \\
0 \\
\ldots \\
\ldots \\
0 \\
1
\end{bmatrix} \quad \text{and,} \quad C = \begin{bmatrix}
1 & 0 & \ldots & 0
\end{bmatrix}
\]

According to Sastry and Isidori [79], using the change of coordinates \( \phi(x) \) we can write:

\[
\left[ \frac{\partial}{\partial x} \phi(x) \left[ f(x) + g(x) \alpha(\phi(x)) \right] \right]_{x=\phi(z)^{-1}} = Az
\]
\[ \left[ \frac{\partial}{\partial x} \phi (x) g (x) \beta (\phi (x)) \right] x = \phi (z)^{-1} = b \]

Which in turn simplifies equation (4.42) to the following form:

\[ z^T K A z - \frac{1}{2} z^T K b b^T K z + \frac{1}{2} z^T C^T C z = 0 \quad (4.41) \]

By substituting equation (4.44) in equation (4.41) we get:

\[ u_{opt} = \alpha (\phi (x)) - \beta^2 (\phi (x)) \frac{2\alpha (\phi (x))}{\left( \frac{1}{\lambda^2} + \beta^2 (\phi (x)) \right)} \quad (4.42) \]

Since the states are unknown, they are replaced by the observed states from equation (4.16) in equation (4.47) for calculating the optimal control input.

### 4.6 Case Study

The case we present is of a single link manipulator arm consisting of a rigid link coupled with the shaft of an armature controlled DC motor. The model parameters are listed in Table 1. The state variables are chosen as follows, \( x_1 (t) = \theta (t) \), \( x_2 (t) = \omega (t) \), and \( x_3 (t) = i_a (t) \).

where, \( \theta (t) \), is the link angle (rad), \( \omega (t) \), is the link angular velocity (rad/s), \( i_a (t) \), is the armature current (A) and, \( u (t) \), is the control input (voltage).

The state equations are:

\[ x_1' = x_2 \quad (4.43) \]

\[ x_2' = -\frac{b}{J} x_2 + \frac{K_T}{J} x_3 + \frac{M g l}{J} \sin (x_1) \quad (4.44) \]

\[ x_3' = \frac{-K_m}{L_a} x_2 + \frac{-R_a}{L_a} x_3 + \frac{1}{L_a} u \quad (4.45) \]
The output equation is given by:

\[ y = h(x) = \theta(t) = x_1(t) \]  \hspace{1cm} (4.46)

It is clear from the state equations that:

\[
\begin{bmatrix}
    x_2 \\
    -\frac{b}{J}x_2 + \frac{K_T}{J}x_3 + \frac{M gl}{J}\sin(x_1) \\
    -\frac{K_m}{L_a}x_2 + \frac{-R_a}{L_a}x_3 \\
\end{bmatrix}
\]

\[ f(x) = \begin{bmatrix}
    0 \\
    0 \\
    \frac{1}{L_a}
\end{bmatrix} \]  \hspace{1cm} (4.48)

The relative degree of this system is 3. If we define the system parameters as follows:

\[
\begin{align*}
    \theta_1^1 &= 1, \quad \theta_2^1 = \frac{b}{J}, \quad \theta_3^1 = \frac{K_T}{J}, \quad \theta_4^1 = \frac{M gl}{J}, \quad \theta_5^1 = \frac{R_a}{L_a}, \quad \theta_6^1 = \frac{K_m}{L_a}, \quad \theta_1^2 = \frac{1}{L_a}
\end{align*}
\]

The estimated system equations may be written in the following form:

\[
\begin{align*}
    \hat{x}_1 &= \begin{bmatrix} \dot{x}_2 \\ 0 \\ 0 \end{bmatrix}, \quad \hat{x}_2 &= \begin{bmatrix} 0 \\ -\dot{x}_2 \\ \ddot{x}_3 \end{bmatrix}, \quad \hat{x}_3 &= \begin{bmatrix} \dot{x}_3 \\ \ddot{x}_3 \\ 0 \end{bmatrix}, \quad \hat{x}_4 &= \begin{bmatrix} 0 \\ \sin(\hat{x}_1) \\ 0 \end{bmatrix}, \quad \hat{x}_5 &= \begin{bmatrix} 0 \\ 0 \\ -\dot{x}_3 \end{bmatrix}, \\
    \hat{x}_6 &= \begin{bmatrix} 0 \\ 0 \\ -\dot{x}_2 \end{bmatrix} \quad \text{and} \quad g_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\end{align*}
\]

The estimated system states are given by the observer equation (4.16), choosing \( K_{obs} = 100 \) results in the following equations for \( \hat{x}_1, \hat{x}_2, \hat{x}_3 \):
\[ \dot{x}_1 = 300(x_1 - \hat{x}_1) + \dot{x}_2 \]  
(4.49)

\[ \dot{x}_2 = -\left(\theta_2 \dot{x}_2\right) + 3000(x_1 - \hat{x}_1) + \theta_3 \dot{x}_3 + \theta_4 \sin(\hat{x}_1) \]  
(4.50)

\[ \dot{x}_3 = \theta_1 u - \theta_6 \dot{x}_2 + (x_1 - \hat{x}_1) \left[ \frac{10^6}{\theta_3} + 30000 \left( \frac{\theta_2}{\theta_3} \right) - 300 \left( \frac{\theta_4}{\theta_3} \right) \cos(\hat{x}_1) \right] - \theta_5 \dot{x}_3 \]  
(4.51)

The stability of the zero dynamics is demonstrated as follows, chose the vector representing the new system states \( z = \begin{bmatrix} \mu & \psi_1 & \psi_2 \end{bmatrix}^T \), where \( x_1 = \psi_2, \ x_2 = -\psi_1 \) and \( x_3 = \mu \).

\[ \frac{\partial z}{\partial x} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \]  
(4.52)

Therefore the matrix is invertible.

The system dynamics can be simply written in normal form as:

\[ \dot{\mu} = 0 \]  
(4.53)

\[ \psi_1 = -\dot{x}_2 = -\left( \frac{b}{J} \psi_1 + \frac{Mg}{J} \sin(\psi_2) \right) \]  
(4.54)

\[ \psi_2 = \dot{x}_1 = x_2 = -\psi_1 \]  
(4.55)

Consider now the Lyapunov function candidate:

\[ V(\psi_1, \psi_2) = \frac{1}{2} (\psi_1^2 + \psi_2^2) \]  
(4.56)

which is positive definite. Now if we differentiate, \( V \), along the system trajectory given by equations (4.59) and (4.60) then we get:

\[ \dot{V} = \psi_1 \dot{\psi}_1 + \psi_2 \dot{\psi}_2 = -\psi_1 \left[ \frac{b}{J} \psi_1 + \frac{Mg}{J} \sin(\psi_2) \right] - \psi_2 \psi_1 < 0 \]  
(4.57)
Proof:

\[
\frac{Mgl}{J} \sin(\psi_2) \leq \frac{Mgl}{J} \psi_2
\]  

(4.58)

Therefore:

\[
\dot{V} = -\frac{b}{J} \psi_1^2 - \frac{Mgl}{J} \psi_2 \psi_1 - \psi_2 \psi_1
\]  

(4.59)

\[
\dot{V} = -\frac{b}{J} \psi_1^2 - \left(\frac{Mgl}{J} + 1\right) (\psi_2 \psi_1) = -A \psi_1^2 - B (\psi_2 \psi_1)
\]  

(4.60)

\[
\dot{V} = -A \psi_1^2 - B (\psi_2 \psi_1) = -A \left( \psi_1^2 + \frac{B}{A} (\psi_2 \psi_1) \right)
\]  

(4.61)

\[
\dot{V} = -A \left( \psi_1 + \frac{B}{2A} \psi_2 \right)^2 - A \left( \frac{B}{2A} \psi_2 \right)^2 < 0
\]  

(4.62)

It could be concluded from equations (4.62) and (4.67) that the zero dynamics for this system are stable. From equation (4.21) it is clear that, \( \psi \), is bounded which implies that, \( x_1 \), is bounded and in turn, \( \psi_2 \), is bounded. This result and equation (4.67) imply that, \( \psi_1 \), is bounded.

The optimal control output, \( u_{opt} \), is then given by:

\[
u_{opt} = K_m \dot{x}_2 + R_a \dot{x}_3 + 1000 \frac{JL_a}{K_T} (-x_1 + y_{ref}) - \frac{MglL_a}{K_T} \dot{x}_2 \left( \cos (x_1 - y_{ref}) \right)
\]

\[
+ \frac{bL_a}{J} \dot{x}_3 - \frac{b^2 L_a}{JK_T} \dot{x}_2 + \frac{bL_a Mgl}{JK_T} \sin (x_1 - y_{ref})
\]

\[
u_{opt} = -0.8 (x_1 - y_{ref}) + 9.99995 \dot{x}_2 + 1.0005 \dot{x}_3
\]

\[-0.004 \left( \dot{x}_2 - y_{ref} \right) \cos (x_1 - y_{ref}) + 0.001 \sin (x_1 - y_{ref})
\]  

(4.63)
4.7 Simulation Results

The following simulation demonstrates the performance of the proposed controller. The controller is applied to a single-link rigid robot arm (Figure 4.1). The simulation assumed nominal values for the physical parameters, given on Table (1). These values correspond to system parameters, $\theta_1^1, \theta_2^1, \theta_3^1, \theta_4^1, \theta_5^1, \theta_6^1, \theta_1^2$, as shown in Table (2). The initial conditions used are presented in Table (3). Figures (4.2) and (4.3) show the response of the reference signal $y_{ref}$ and the output signal $x_1$ in case of applying typical and optimal feedback linearization, respectively. It is clear from Figure (4.2) that although the output shows a very rapid and accurate tracking for the reference signal, it suffers from a high overshoot. The high overshoot is attributed to two reasons. First, different initial conditions for the reference tracking signals were used for both the reference (nominal) and actual systems. Second, the mismatch in system parameters between the reference (nominal) and actual systems. The proposed optimal feedback linearization controller does not suffer from this drawback as clear from Figure (4.3). In order to clarify this fact, the corresponding output tracking error applying both controllers is plotted in Figure (4.4).

The main purpose for applying the proposed optimal feedback linearization controller is to minimize the required control effort to achieve the same response (in this case better) as opposed to applying typical feedback linearization. This fact is illustrated in Figure (4.5). The motor torque required to track the desired trajectory in the case of typical feedback linearization is 350 Nm while in case of optimal feedback linearization is 150 Nm. This is calculated from the relation $\tau = K_T i$.

The above mentioned simulations were pursued using Matlab, Mathworks Inc. [87].
4.8 Conclusion

The increasing requirement on robot tracking performance prompts the study of including motor dynamics in controller design, an approach made possible by today's ever advancing computer technology. The currently available robot controllers in this class require exact knowledge of both robot and motor dynamics and acceleration feedback. In this chapter we presented a new approach addressing this problem, we achieved the first objective by adopting an adaptive feedback linearization techniques, Sastry, and Isidori [79] and enhancing those techniques to be applied in an optimal sense by exploiting the Hamilton-Jacobi equation and choosing an appropriate cost function. We achieved the second objective by implementing an observer that guarantees asymptotic stability for the estimation errors, the acceleration feedback constraint is removed, which has a significant impact from the a practical point of view since acceleration feedback is not available for most industrial controllers. Moreover, applying this technique removes linear growth constraints on the nonlinearities inherent in the system in order to guarantee global stability. The results demonstrate an asymptotic tracking response for the output, this is achieved using minimum control effort.
### Table 1: Physical Parameters

<table>
<thead>
<tr>
<th>$R_a$</th>
<th>$L_a$</th>
<th>$K_T$</th>
<th>$K_m$</th>
<th>$J$</th>
<th>$b$</th>
<th>$l$</th>
<th>$Mg$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.002</td>
<td>1</td>
<td>10</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>20</td>
</tr>
</tbody>
</table>

### Table 2: Nominal System Parameters

<table>
<thead>
<tr>
<th>$\theta_1^1$</th>
<th>$\theta_2^1$</th>
<th>$\theta_3^1$</th>
<th>$\theta_4^1$</th>
<th>$\theta_5^1$</th>
<th>$\theta_6^1$</th>
<th>$\theta_1^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.25</td>
<td>2.5</td>
<td>5</td>
<td>500</td>
<td>5000</td>
<td>500</td>
</tr>
</tbody>
</table>

### Table 3: Parameters and States Initial Conditions

<table>
<thead>
<tr>
<th>$x_1(0)$</th>
<th>$x_2(0)$</th>
<th>$x_3(0)$</th>
<th>$\dot{\theta}_5^1$</th>
<th>$\dot{\theta}_6^1$</th>
<th>$\dot{\theta}_1^2$</th>
<th>$y_{ref}$</th>
<th>$\dot{y}_{ref}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>0.0</td>
<td>550</td>
<td>5500</td>
<td>550</td>
<td>2</td>
<td>-1</td>
</tr>
</tbody>
</table>

![Figure 4.1: Schematic of Single link robot arm](image-url)
Figure 4.2: Tracking response in case of typical Feedback Linearization

Figure 4.3: Tracking response in case of optimal Feedback Linearization
Figure 4.4: Comparison of output tracking error

Figure 4.5: Comparison of Control input
Robust Adaptive Control of a Robotic Manipulator
Including Motor Dynamics

5.1 Introduction

Robustness is an important issue in robotic controller design. There are several control strategies which provide robust control for robotic manipulators, namely, variable structure control, non-linear decoupling control and adaptive control. Variable structure controllers estimate the worst possible effect of the non-linear dynamics and use excessive chattering force to control the robot tracking, Grimm, and Frank [88]. Non-linear decoupling (using feedback linearization) uses a complete model of the robot dynamics. The model-plant mismatch effect of non-linear decoupling controllers has been studied by Craig et al. [89], where the mismatch effect is shown to be norm bounded. Adaptive controllers try to estimate and compensate the non-linear dynamics of a robotic manipulator. The control torques or forces generated by an adaptive controller are generally smoother than variable structure controller and hence more suitable for practical implementation. This is the motivation for adopting robust adaptive control for a robotic manipulator in this study. Adaptive controllers may be classified according to whether or not a regressor matrix is used to linearize the non-linear dynamics with respect to a constant parameter vector. Craig et al. [90] were perhaps the first to use the regressor to design an asymptotically stable adaptive controller for robotic manipulators. In order to avoid acceleration feedback, Hsu [91] used a low pass filtered regressor to adapt a parameter
vector. An essential progress is due to Middleton, and Goodwin [92]. This design avoids acceleration measurement and the inversion of the estimated inertia matrix. An exponentially stable adaptive controller was proposed by Slotine, and Li [93]. A group of adaptive controllers were designed by, Slotine, and Li [94] which use off-line-computed regressor matrices to adapt a parameter vector. A composite adaptive controller was proposed by Sadegh, and Horowitz [95] which uses two kinds of regressors to speed up the convergence of tracking and parameter errors.

Adaptive controllers designed without adopting the regressor matrix approach were proposed by many researchers. Dubowsky, and DesForges [70] proposed an adaptive controller that used the steepest descent technique to adjust the dynamic coefficient matrices. The work by Bayard, and Wen [96] utilizes the hyper-stability theory to derive an adaptive law for linearizing and decoupling the nonlinear dynamics. Experimental evaluation of this approach was presented in, Reed, and Ioannou [97] and Slotine, and Li [98]. These adaptive controllers have to compensate for the varying dynamic coefficient matrices. Their stability analysis is based on an assumption that the nonlinear coefficient matrices in the robotic dynamic model change so slowly, that they can be approximated by constants.

A major reason for the incorporation of the regressor matrices is to remove this conservative assumption, especially if the robotic manipulator is expected to move with high speed. The nonlinear dynamics are expressed as a product of a regressor matrix and a constant unknown parameter vector. This is achieved by making use of a physical property of robotic manipulators, namely, by the dynamics being linear with respect to the parameter vector. The time varying effects are all attributed to the regressor which is supposed to be available for the controller. In the Lyapunov sense, a properly designed adaptive law will
cancel out the regressor, creating a semi-negative definite time derivative of the Lyapunov function.

The new adaptive controller developed here requires neither velocity nor acceleration feedback. Naturally, it does not assume exact knowledge of either robot or actuator parameters.

5.2 Problem Statement

The equation of motion of a rigid body robot manipulator is given by Spong, and Vidyasagar [5] as follows:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = \tau$$  \hspace{1cm} (5.1)

where, $q, \dot{q}, \ddot{q} \in \mathbb{R}^n$, denote, respectively, the joint position, velocity and acceleration, while $M(q), C(q, \dot{q}) \in \mathbb{R}^{n \times n}$, are the inertia and centrifugal matrices, and $g(q) \in \mathbb{R}^n$, is the gravitation force. The left-hand side of the above equation represents the rigid robot dynamics while the right-hand side is the input torque, $\tau \in \mathbb{R}^n$. Many robot controllers are based on this model with the assumption that the actuator dynamics are negligible. Recently, the effects of actuator dynamics, specifically motor dynamics, have attracted considerable research interest. For a robot driven by DC motors, the actuator dynamics are represented by:

$$H\ddot{\tau} + R\dot{\tau} + K\dot{\tau} = u$$  \hspace{1cm} (5.2)

where $H, R, K$, are constant diagonal matrices representing the motors (armature) inductance, resistance and electromechanical conversion coefficient respectively. The real control input according to this analogy is the voltage signal, $u$, not the torque, $\tau$, which is
clear from equation (5.2). The dynamic relation described by equation (5.2) is actually equation (1) in Tarn et al.[76] with, \( \tau \), proportional to, \( i \), the motor current, and, \( \dot{q} \), proportional to, \( \dot{\omega} \), the angular velocity of the motor shaft. The motor dynamics will inevitably affect the tracking performance of the robot, this is the reason why they should be accounted for if high accuracy tracking is required.

We may introduce a diagonal matrix, \( D(x) : R^n \rightarrow R^{n \times n} \), whose diagonal elements are the components of, \( x \in R^n \). Then one can write, \( H = D(h) \), \( R = D(r) \). and, \( K = D(k) \). Where, \( h \), \( r \), \( k \in R^n \), are vectors whose components are the diagonal elements of matrices, \( H \), \( R \), and, \( K \), respectively. The use of, \( D(x) \), reduces the uncertainties of diagonal matrices, \( H \), \( R \), and, \( K \), to the uncertainties of vectors, \( h \), \( r \), and, \( k \). Hence, equation (5.2) may be expressed as:

\[
H \dot{\tau} + D(\tau) r + D(\dot{q}) k = u
\]  

(5.3)

In order to control, \( q \), one must control, \( \tau \), first such that it converges to a desired torque, \( \tau_d \), which will be defined later.

The objective of this study is to synthesize an adaptive control signal, \( u \), that ensures the convergence of robot joint position vector, \( q \), to the desired position vector, \( q_d \), when the exact robot model is not known a priori. The uncertainty of, \( M(q) \), \( C(q, \dot{q}) \), and, \( g(q) \), is due to the uncertain robot inertia parameters that vary with different payloads. A well known approach to dealing with such uncertainty is to exploit the fact that robot dynamics are linear with respect to a properly defined inertia parameter vector, \( \rho_r \in R^m \), via the computable regressor, \( Y(\ddot{q}, \dot{q}, q) \in R^{n \times m} \), Slotine, and Li [73] and, Yuan and Stepanenko [102]. The robot dynamic equation could then be expressed as follows:

\[
M(q) \ddot{q}_d + C(q, \dot{q}_d) \dot{q}_d + g(q) = Y(\ddot{q}_d, \dot{q}_d, q) \rho_r
\]  

(5.4)
where, \( Y(\dot{q}_d, \dot{\dot{q}}_d, q) \), is the regressor assumed available for the controller and, \( p_r \), is the actual (unknown) inertia parameter vector.

If we choose, \( \varepsilon \), to be:

\[
\varepsilon = \tau - \tau_d \tag{5.5}
\]

as a representative of the torque error which is the difference between the desired torque and the actual torque. The desired torque is represented as follows:

\[
\tau_d = \tau_f - K_{cd} \dot{e} - K_{cp} \dot{\hat{e}} \tag{5.6}
\]

where, \( \tau_f = Y(\dot{q}_d, \dot{\dot{q}}_d, q) \), \( p \in R^n \), is a feed-forward signal intended to cancel the nonlinear dynamic forces, \( p \in R^m \), is an estimate of, \( p_r \), \( K_{cd} = K_{cd}^T \), is the controller derivative (positive definite) gain matrix, \( K_{cp} = K_{cp}^T \), is the controller proportional (positive definite) gain matrix and, \( \dot{e} = \dot{q} - q_d \), representing the error between the estimated joint position vector and the desired joint position vector.

Consider the linear observer equations given by:

\[
\dot{\hat{e}} = w + K_{obd}(e - \hat{e}) \tag{5.7}
\]

\[
\dot{\hat{w}} = K_{obp}(e - \hat{e}) \tag{5.8}
\]

where, \( e = q - q_d \), is the error and represents the difference between the actual and the desired joint position vectors, respectively. Meanwhile, \( K_{obd} = K_{obd}^T \), \( k_{obp} \), \( k_{obp} > 0 \), is the observer derivative (positive definite) gain matrix and, \( K_{obp} = K_{obp}^T \), is the observer proportional (positive definite) gain matrix. We need to define, \( \hat{q} = q - \hat{q} \), as the difference between the actual and estimated joint position vectors, respectively.
Substituting equations (5.5), (5.6) and (5.7) in equation (5.1) we get:

\[ M(q) \ddot{q} + C(q, \dot{q}) \dot{q} + g(q) = \tau_f - K_{cd} (w + K_{obd} (e - \dot{e})) - K_{cp} \dot{e} + \varepsilon \]  

(5.9)

if we choose \( K_{cp} = \gamma K_{cd} \), where \( \gamma \) is a positive constant and substitute in equation (5.9) we get:

\[ M(q) \ddot{q} + C(q, \dot{q}) \dot{q} + g(q) = \tau_f - K_{cd} w - K_{cd} K_{obd} (e - \dot{e}) - \gamma K_{cd} \dot{e} + \varepsilon \]  

(5.10)

we may define, \( \phi_p \), as the difference between the estimated robot inertia parameters and the actual parameters as follows:

\[ \phi_p = p - p_r \]  

(5.11)

if we subtract equation (5.4) from equation (5.10) we get:

\[ M(q) \ddot{q} + C(q, \dot{q}) \dot{q} + g(q) - M(q) \ddot{q}_d - C(q, \dot{q}_d) \dot{q}_d - g(q) \]

= \[ \tau_f - K_{cd} w - K_{cd} K_{obd} (e - \dot{e}) - \gamma K_{cd} \dot{e} + \varepsilon - Y(\dot{\theta}_d, \dot{\phi}_d, \dot{q} \dot{p}) \]

(5.12)

which may be written as follows after substituting equation (5.11)

\[ M(q) \ddot{q} = - K_{cd} w - K_{cd} K_{obd} (e - \dot{e}) - \gamma K_{cd} \dot{e} + \varepsilon + Y(\dot{\theta}_d, \dot{\phi}_d, \dot{q} \dot{p}) \phi_p - C(q, \dot{q}) \dot{e} - C_q \dot{e} \]

(5.13)

where according to Su and Stepanenko [115] is given by \( C_q \dot{e} \equiv C(q, \dot{q}) \dot{q} - C(q, \dot{q}_d) \dot{q}_d = C(q, \dot{e}) \dot{q}_d \), and, \( \|C_q \dot{e}\| \leq \|\dot{q}\| \).

It can easily be shown that:

\[ e - \dot{e} = q - q_d - \dot{q} + q_d = \ddot{q} \]  

(5.14)

if we substitute equation (5.14) in both equations (5.7) and (5.8) we get:
\[ \dot{\mathbf{e}} = w + K_{obd} \ddot{q} \]  
\[ \dot{w} = K_{obp} \ddot{q} \]  
(5.15) 
(5.16)

from equation (5.14) it is easily shown that:
\[ \ddot{q} = \dot{\mathbf{e}} - \dot{\mathbf{e}} \]  
(5.17)

we may now substitute equation (5.15) in equation (5.18) to get:
\[ \ddot{q} = \dot{\mathbf{e}} - w - K_{obd} \ddot{q} \]  
(5.18)

we may substitute equation (5.14) in equation (5.13) and rewrite equation (5.13) in the following form:
\[ \ddot{\mathbf{e}} = -M^{-1} C_{cd} K_{obd} \ddot{q} - M^{-1} C_{cd} w - M^{-1} \gamma K_{cd} \dot{\mathbf{e}} + M^{-1} \gamma K_{cd} \ddot{q} - M^{-1} C(q, \dot{q}) \dot{\mathbf{e}} - M^{-1} C_d \dot{\mathbf{e}} + M^{-1} \gamma \Phi_p + M^{-1} \varepsilon \]  
(5.19)

If we introduce a state vector, \( \mathbf{x}^T = [\dot{\mathbf{e}}, w, e, \ddot{q}]^T \), we may then represent the closed loop robot dynamics in state space form as follows:
\[
\dot{\mathbf{x}} = A \mathbf{x} + B \left( Y \Phi_p - C(q, \dot{q}) \dot{\mathbf{e}} - C_d \dot{\mathbf{e}} + \varepsilon \right) 
\]  
(5.20)

where \( A = \begin{bmatrix} 0 & -M^{-1} K_{cd} & -M^{-1} \gamma K_{cd} & -M^{-1} K_{obd} + M^{-1} \gamma K_{cd} \\ 0 & 0 & 0 & K_{obp} \\ I & 0 & 0 & 0 \\ I & -I & 0 & K_{obd} \end{bmatrix} \), \( B = \begin{bmatrix} M^{-1} \\ 0 \\ 0 \\ 0 \end{bmatrix} \)  
(5.20 a)

An important procedure in the design is the choice of a pair of positive definite matrices \( P \) and \( Q \) such that \( \frac{1}{2} \left( PA + A^T P \right) = Q \), to this effect one possible choice is given by:
\[
P = \begin{bmatrix}
M\left(\frac{1}{\gamma}\right) & M\left(-\frac{1}{\gamma}\right) & M & MK_{obd}\left(-\frac{1}{\gamma}\right) \\
M\left(-\frac{1}{\gamma}\right) & 2IK_{obp}^{-1} & 0 & 0 \\
M & 0 & \gamma K_{cd} & 0 \\
MK_{obd}\left(-\frac{1}{\gamma}\right) & 0 & 0 & K_{cd}
\end{bmatrix}
\]

(5.20 b)

The eigenvalues of \(M(q)\), are uniformly bounded for all \(q\), as shown by Spong, and Vidyasagar [5]. Hence, by choosing a sufficiently large constant \(\gamma\), one can guarantee that both matrices \(P\), and \(Q\), are positive definite. The eigenvalues of \(P\), and \(Q\), are bounded by:

\[
\lambda_p = \inf_{x} \inf_{q} \left( \frac{x^T Px}{\|x\|^2} \right) \text{ and } \gamma \lambda_q = \inf_{x} \inf_{q} \left( \frac{x^T Qx}{\|x\|^2} \right) \text{ which means}
\]

\[
\lambda_p \|x\|^2 \leq x^T Px \text{ and } \gamma \lambda_q \|x\|^2 \leq x^T Qx
\]

(5.21)

From equation (5.3), a possible control law for the robot-actuator system is of the following form:

\[
u = D\left(\dot{\tau}_d\right)h_a + D(\tau)r_a + D(\dot{\theta})k_a - \beta \varepsilon - z
\]

(5.22)

where, \(z\), is given by \(z = B^T Px = \frac{1}{\gamma} \dot{e} - \frac{1}{\gamma} w + e - \frac{1}{\gamma} K_{obd} \ddot{q}\), and \(\beta > 0\), is a design constant. \(h_a\), \(r_a\), and \(k_a\), are adaptive vectors intended to compensate the effect of unknown vectors \(h\), \(r\), and \(k\), in equation (5.3). The adaptive laws for \(h_a\), \(r_a\), and \(k_a\), are
given by:

\[ h' \boldsymbol{a} = \Phi \Phi h = -D(\tau_d)\varepsilon, \quad r' \boldsymbol{a} = \Phi \Phi r = -D(\tau)\varepsilon \quad \text{and} \]

\[ k' \boldsymbol{a} = \Phi \Phi k = -D(\dot{q})\varepsilon \]

(5.23)

Where, \( \Phi \Phi h = h' - h \), \( \Phi \Phi r = r' - r \), and, \( \Phi \Phi k = k' - k \), are the compensation error vectors. The unknown vectors, \( h \), \( r \) and \( k \), are assumed to be constant.

**Lemma 1:** The dynamic system given by equations (5.22) and (5.23) is globally stable.

Proof: At this point, the closed-loop motor dynamics can be easily obtained by substituting equation (5.22) into equation (5.2), and, \( D(\tau_d)h \), for, \( H\tau_d \), which results in:

\[ H\dot{\varepsilon} = -\beta\varepsilon - z + D(\tau_d)\Phi h + D(\tau)\Phi r + D(\dot{q})\Phi k \]

(5.24)

Consider the following Lyapunov function candidate:

\[ V_m = \frac{1}{2}(\varepsilon^T H\varepsilon + \Phi^T h\Phi h + \Phi^T r\Phi r + \Phi^T k\Phi k) \geq \frac{1}{2}(h_{min} \varepsilon^2 + \Phi h^2 + \Phi r^2 + \Phi k^2) \]

Where, \( h_{min} > 0 \), is the smallest element of the positive definite diagonal matrix, \( H \). The time derivative of, \( V_m \), evaluated along the trajectory of equation (5.24) is given by:

\[ \dot{V}_m = -\beta\varepsilon^T \varepsilon - z^T \varepsilon + \left[ \varepsilon^T D(\tau_d) + \Phi^T h \right] \Phi h + \left[ \varepsilon^T D(\tau) + \Phi^T r \right] \Phi r + \left[ \varepsilon^T D(\dot{q}) + \Phi^T k \right] \Phi k \]

The adaptation laws, given by equation (5.23), eliminate all terms involving the compensation error vectors in the above equation. As a result, \( \dot{V}_m = -\beta\varepsilon^T \varepsilon - z^T \varepsilon \leq 0 \).

The scalar, \( z^T \varepsilon \), is designed such that it first forces the convergence of, \( \varepsilon \rightarrow 0 \), thus enabling the desired torque to dominate the system dynamics as given by equation (5.10).

In the same time, this scalar does not affect the overall closed loop system stability (represented by \( \dot{V} \)) since it is cancelled out by adding \( \dot{V} = \dot{V}_r + \dot{V}_m \). The monotonous decrease in, \( V_m \), implies the uniform boundedness of, \( \tau, \varepsilon, \Phi h, \Phi r \), and, \( \Phi k \). It can then
be concluded that applying the Barbalat's lemma, Spong, and Vidyasagar[5] enables one to conclude that, $\varepsilon$, will converge to zero. This completes the proof.

In the next section, it will be shown that equation (5.20) is able to guarantee stable tracking of, $\varepsilon = \tau - \tau_d \rightarrow 0$. The adaptive parameter vector, $p$, is adjusted by the following adaptive law:

$$\dot{p} = \phi_p = -Y^T(T \ddot{q}, \dot{q}, q, z)$$  \hspace{1cm} (5.25)

The desired torque, $\tau_d$, and its time derivative, $\dot{\tau}_d$, are now synthesized by four equations (5.6),(5.7),(5.8) and (5.22) which in turn satisfy the design constraint to avoid velocity and acceleration measurements by designing a linear observer and to synthesize $\omega, \dot{\omega}$, instead of $\dot{\theta}, \ddot{\theta}$.

Although the realization of, $\dot{\tau}_d$, ensuring the convergence of, $\tau \rightarrow \tau_d$, implies the computation of, $\dot{\theta}$, which is an increase in computational burden this step is also required in the case of third order models. Moreover this method does not require torque (or current) feedback since the torque is synthesized by the linear controller.

5.3 Comparison with Other Designs

In the early studies of the control of rigid robots including actuator dynamics, Tarn et al. [76], Beekmann, and Lee [77] full knowledge of the actuator parameters was assumed. Ge, and Postlethwaite [78] proposed an adaptive controller for robots including motor dynamics. It required acceleration feedback to guarantee stable tracking. Aldhaheri, and Khalil [113] used singular perturbation to confirm that if the actuator dynamics were sufficiently fast, the controller stabilized the origin of the closed-loop system. The focus of
their work, however, is the effect of unmodelled fast actuator dynamics on the output feedback stabilization and not the tracking response. This is also the focus of work by Taylor et al. [114], on adaptive regulation. Work similar to that described here is presented by Su, and Stepanenko [115]. However, this technique suffers from the following restrictions in terms of implementation: First, it requires that a general expression and bound for the control input signal be found. Second, it assumes the availability of velocity measurements, which, for practical purposes are not readily available. Third, the tracking performance depends crucially on the choice of the controller parameters which could lead to chattering. The new adaptive tracking controller developed here requires neither the velocity nor acceleration feedback. Naturally, it does not assume exact knowledge of either robot or actuator parameters. The choice and implementation of the proposed control law are less restrictive than the previously-mentioned design method.

5.4 Stability Analysis

The stability proof will be based on a Lyapunov function candidate

\[ V = V_r + V_m \]

where:

\[ V_r = \frac{1}{2} \left( x^T P x + \phi_p^T \phi_p \right) \]  \hspace{1cm} (5.26)

and:

\[ V_m = \frac{1}{2} \left( \varepsilon^T H \varepsilon + \Phi_h^T \Phi_h + \Phi_r^T \Phi_r + \Phi_k^T \Phi_k \right) \]  \hspace{1cm} (5.27)

Theorem 1: For the adaptive system described by equations (5.20), (5.22) and (5.25), there exists a constant, \( V_{\text{max}} > 0 \), such that, \( \dot{V} \leq 0 \), \( V < V_{\text{max}} \). The tracking system is stable if and only if \( \gamma > K_{\text{obd}} > 1 \), and, \( \gamma \), is sufficiently large. The region of
attraction is defined by the following radius:

\[ V_{\text{max}} = \frac{\lambda_p \gamma^2}{8 \xi_m} \left( \gamma \lambda_q - 2 \left( \frac{\xi_m}{\gamma} \xi_q - 3 \xi_d \right) \right)^2 \]  

(5.28)

where \( \xi_d \), was given earlier and \( \xi_q = \sup \| \dot{q} \| \), is the upper bound of the desired velocity and given that the system satisfies the following condition:

\[ \xi_m \| \dot{q} \| \| x \| \geq \left\| [M - C(q, q)] \dot{e} \right\| \]  

(5.29)

**Proof:**

Since \( V = V_r + V_m \), \( \dot{V} = \dot{V}_r + \dot{V}_m \) and, \( \dot{V}_m = -\beta e^T e - z^T e \leq 0 \) has been previously proven (Lemma 1).

Evaluating \( \dot{V}_r \), along the trajectory of equation (5.20) we get:

\[ \dot{V}_r = -x^T Q x - z^T \left( C(q, \dot{q}) \dot{e} + C_d \dot{e} \right) + \frac{1}{2} x^T \dot{P} x + z^T e + \left( z^T Y + \phi_p^T \right) \phi_p \]

From equation (5.25) we can write \( z^T Y + \phi_p^T = 0^T \), then \( \dot{V}_r \) can be written as follows:

\[ \dot{V}_r = -x^T Q x - z^T \left( C(q, \dot{q}) \dot{e} + C_d \dot{e} \right) + \frac{1}{2} x^T \dot{P} x + z^T e \]  

(5.30)

Using lemma 1, we may write, \( \dot{V} \) as follows:

\[ \dot{V} = -x^T Q x - z^T \left( C(q, \dot{q}) \dot{e} + C_d \dot{e} \right) + \frac{1}{2} x^T \dot{P} x - \beta e^T e \]  

(5.31)

A closer look at equation (5.31) indicates a tendency of, \( \dot{V} \), to decrease because of the negative definiteness of the first and last terms. The middle two terms, however, have some uncertain effects, which should be addressed before drawing any conclusions on the negativity of \( \dot{V} \). Applying the skew-symmetry property, \( \dot{e}^T \left[ \frac{M(q)}{2} - C \right] \dot{e} \), the term, \( \frac{1}{2} x^T \dot{P} x - z^T C \dot{e} \), may be expressed as follows:
\[
\frac{1}{2} x^T P x - z^T C \dot{e} = \frac{1}{2\gamma} (e - w - K_{obd} \tilde{q})^T [\dot{M} - C (\dot{q}, q)] \dot{e}
\]

(5.32)

Applying the bound from equation (5.29), the following bound is achieved:

\[
\frac{1}{\gamma} (e - w - K_{obd} \tilde{q})^T [\dot{M} - C (\dot{q}, q)] \dot{e} \leq 2 \left( \frac{\xi_m}{\gamma} \right) ||\dot{q}|| ||x||^2
\]

(5.33)

Applying the bound for the term \( ||C_d \dot{e}|| \leq \xi_d ||\dot{e}|| \), the following bound is achieved:

\[
-z^T C_d \dot{e} = - \left( \frac{1}{\gamma} \dot{e} - \frac{1}{\gamma} w + \frac{1}{\gamma} K_{obd} \tilde{q} \right)^T C_d \dot{e} \leq 3 \xi_d ||x||^2
\]

(5.34)

Substituting equations (5.21), (5.33) and (5.34) in equation (5.31), the following expression is obtained for, \( \dot{V} \):

\[
\dot{V} \leq - \left( \gamma \lambda_q - 2 \left( \frac{\xi_m}{\gamma} \right) ||\dot{q}|| - 3 \xi_d \right) ||x||^2 - \beta ||\epsilon||^2
\]

(5.35)

If we substitute \( ||\dot{q}|| \leq \dot{q}_{d} || + ||\dot{\epsilon}|| \leq \sup ||\dot{q}_{d}|| + ||x|| = \xi_q + ||x|| \) in equation (5.35) we get the following expression for the inequality in equation (5.35):

\[
\dot{V} \leq - \left( \gamma \lambda_q - 2 \left( \frac{\xi_m}{\gamma} \right) (\xi_q + ||x||) - 3 \xi_d \right) ||x||^2 - \beta ||\epsilon||^2
\]

(5.36)

Since, \( V(t) \geq V_r(t) \geq \frac{1}{2} x^T P x \), and recalling equation (5.21), it can be observed that:

\[
V (t) \geq \frac{\lambda_p}{2} ||x||^2
\]

hence it is clear that an upper bound on, \( ||x|| \), will have the form of:

\[
||x|| \leq \frac{2V}{\lambda_p}
\]

(5.37)

Substituting equation (5.37) in equation(5.36), the upper bound on, \( \dot{V} \), may be represented by:

\[
\dot{V} \leq - \left( \gamma \lambda_q - 2 \left( \frac{\xi_m}{\gamma} \right) (\xi_q + \frac{2V}{\lambda_p}) - 3 \xi_d \right) ||x||^2 - \beta ||\epsilon||^2
\]

(5.38)
A sufficient condition for $\dot{V} \leq 0$, is that:

$$
\gamma \lambda_q - 2 \left( \frac{\xi_m}{\gamma} \right) \xi_q - 3 \xi_d \geq 2 \left( \frac{\xi_m}{\gamma} \right) \frac{2 \dot{V}}{\lambda_p} \tag{5.39}
$$

By squaring equation (5.39) we get the following expression for $V_{max}$:

$$
V_{max} = \frac{\lambda_p \gamma^2}{8 \xi_m^2} \left( \gamma \lambda_q - 2 \left( \frac{\xi_m}{\gamma} \right) \xi_q - 3 \xi_d \right)^2 \tag{5.40}
$$

From equations (5.26) and (5.39) and using Barbalat's lemma, Spong, and Vidyasagar [5], the convergence of the parameter error $\phi_p$, to zero can be shown, similar to the lemma 1 proof for the tracking error $\varepsilon$.

### 5.5 Simulation Results

The following simulation demonstrates the performance of the proposed controller.

The controller is applied to a two-link rigid robot (Figure 5.1), whose dynamic coefficient matrices are as follows:

![Figure 5.1: Schematic of Two-link planar robot](image-url)
Two-Link Planar robot parameters

<table>
<thead>
<tr>
<th></th>
<th>First Link</th>
<th>Second link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length (m)</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Actual Mass (kg)</td>
<td>5.0</td>
<td>5.0</td>
</tr>
<tr>
<td>Estimated Mass (Kg)</td>
<td>5.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Motor Parameters
First Motor
(ohm.second)              | Inductance 0.0029 |
(ohm)                     | Resistance 0.96  |
(volt.second/radians)     | EM constant 11.88 |
Second Motor
(ohm.second)              | Inductance 0.0017 |
(ohm)                     | Resistance 0.55  |
(volt.second/radians)     | EM constant 20.48 |

\[
M(q) = \begin{bmatrix}
m_2 l_2^2 (2 l_1 \cos(q_2) + l_2) + l_1^2 (m_1 + m_2) l_2 m_2 + l_1 l_2 \cos(q_2) m_2 \\
l_2^2 m_2 + l_1 l_2 \cos(q_2) m_2 \\
l_2^2 m_2 + l_1 l_2 \cos(q_2) m_2
\end{bmatrix}
\]

\[
C(q, \dot{q}) = \begin{bmatrix}
-l_1 l_2 m_2 \sin(q_2) \dot{q}_2 \\
l_1 l_2 m_2 \sin(q_2) \dot{q}_1
\end{bmatrix}
\]

\[
g(q) = \begin{bmatrix}
9.8 (m_2 l_2 \cos(q_1 + q_2) + (m_1 + m_2) l_1 \cos(q_1)) \\
9.8 m_2 l_2 \cos(q_1 + q_2)
\end{bmatrix}
\]

The regressor, \(Y\left(\ddot{q}, \dot{q}, q\right)\), is a 2 x 3 matrix, the entries of which, \(y_{ij}\), are derived as
follows:

\[ y_{11} = y_{21} = q_1^{\dd} + q_2^{\dd} \]
\[ y_{12} = \lambda_2 \left[ \left( 2q_1^{\dd} + q_2^{\dd} \right) \cos(q_2) - \left( 2q_1^{\dd} + q_2^{\dd} \right) q_2^{\dd} \sin(q_2) \right] + 9.8 \cos(q_1 + q_2) \]
\[ y_{13} = \lambda_1 q_1^{\dd} + 9.8 \cos(q_1) \]
\[ y_{22} = \lambda_1 \left[ q_1^{\dd} \cos(q_2) + q_2^{\dd} \sin(q_2) \right] + 9.8 \cos(q_1 + q_2) \]
\[ y_{23} = 0 \]

The components of, \( p_r \), are, \( p_{r1} = \lambda_1 (m_1 + m_2) \), \( p_{r2} = \lambda_2 m_2 \), and \( p_{r3} = \lambda_2^2 m_2 \). The actual robot parameters are, \( \lambda_1 = \lambda_2 = 0.5m \), \( m_1 = m_2 = 5Kg \). A limited knowledge is assumed about those parameters and the nominal parameters utilized by the controller are initially set as follows: \( p_{r1} = \lambda_1 (\hat{m}_1 + \hat{m}_2) \), \( p_{r2} = \lambda_2 \hat{m}_2 \), and \( p_{r3} = \lambda_2^2 \hat{m}_2 \). The initial estimate of the two masses are, \( \hat{m}_1 = m_1 \), and, \( \hat{m}_2 = 1Kg \). This large difference for the mass, \( m_2 \), simulates the uncertainty in the payload. The motor dynamics coefficient matrices are given by:

\[
H = \begin{bmatrix}
0.0029 & 0 \\
0 & 0.0017
\end{bmatrix},
\]
\[
R = \begin{bmatrix}
0.96 & 0 \\
0 & 0.55
\end{bmatrix},
\]
\[
P = \begin{bmatrix}
11.88 & 0 \\
0 & 20.48
\end{bmatrix}
\]

The desired joint position vector is given by \( q_{1d} = \pi (1 - \cos(\pi t)) \) and \( q_{2d} = \pi \left( 1 - \cos \left( \frac{\pi t}{2} \right) \right) \) (rad).

The controller and observer gains used are, \( \gamma = 20 \), \( K_{cd} = 10I \), and \( K_{obd} = 1000I \), respectively, where, \( I \), is the identity matrix. The tracking response for the first and second joints are shown in Figures (5.2) and (5.3), respectively.

Good tracking performance is demonstrated through plotting the tracking error as shown
in Figures (5.3) and (5.4), and parameter error convergence is shown in Figure (5.5).

5.6 Conclusion

Currently available robot controllers require exact knowledge of both robot and motor dynamics and acceleration feedback. In this chapter we presented a new approach addressing this problem by adopting a robust adaptive motion controller that requires only position measurements. We achieved this by designing a linear feedback controller, which utilizes an estimate of the error states. The observer guarantees asymptotic stability for the estimation errors. An adaptive controller is designed to handle the dynamic parameters variations and to synthesize the desired torque for the closed loop control without the need for neither velocity nor acceleration measurements. The global stability of the proposed controller tracking performance is proved in the Lyapunov sense. In addition, simulation results are presented to demonstrate the accurate tracking performance of the closed-loop system.
Figure 5.2: Tracking response for first joint

Figure 5.3: Tracking response for second joint
Figure 5.4: Tracking error for first joint

Figure 5.5: Tracking error for second joint

Figure 5.6: Parameter errors of the proposed controller
Adaptive Control of a Constrained Robotic Manipulator
Including Motor Dynamics

6.1 Introduction

Currently, growing numbers of robots are being employed in manufacturing environments to increase productivity and efficiency. For tasks performed by robotic manipulators such as moving payloads or painting objects, position controllers are deemed adequate because the task specifications require the robot end effector to follow a desired trajectory. However, during an assembly task the robot end effector comes in contact with the environment, hence, interaction forces develop between the robot end effector and the environment. One of the areas in which robots are utilized in assembly tasks is the automotive industry.

Force control is also significant in dealing with geometric uncertainties (errors) by establishing contact (fender and hinge assembly). The presented work in this chapter, deals with these errors in force feedback framework in order to control them which is essential for complicated assembly operations.

Accumulated research effort has been able to solve, the motion control part of the problem relatively well, however, the force control part remains a challenging problem. Assuming perfect knowledge of a robotic dynamic model, Raibert, and Craig [103] proposed a hybrid position and force controller in joint space. Jankowski, and ElMaraghy
[112], [118] and [123] proposed a task space decomposition for position and force. In the latter, a general method was presented that assures an exact feedback linearization for both cases of rigid and flexible-joint robots. The descriptor method was originally introduced by Mills, and Goldenberg [104]. MaClamroch, and Wang [105] studied the application of computed-torque controller for constrained robots. All of these methods depend on the exact cancellation of the robot dynamics to achieve the control objectives. In real applications, however, exact cancellation is seldom possible because of payload variations and actuator dynamics effect. The principal limitations of published models for industrial robots is the assumption that the dynamic model of a robot may be represented by interconnected rigid bodies driven by motors modeled as pure torque sources, Luh et al. [106], Dubowsky, and DesForges [107], Niemeyer, and Slotine [108]. In addition, an assumption common in the literature is that coupling terms due to Coriolis and centrifugal effects may be neglected, or that the model may be linearized. These assumptions simplify the dynamic model of the robot and facilitate the design of controllers. As a result of this simplification, unmodelled disturbances remain present in the actual robot system and affect its tracking and the positioning performance.

The design procedures presented by Tarn et al. [76], Beekmann and Lee, [77] and Ge and Postlethwaite [78] are based on full knowledge of the complex dynamics of robotic systems. If there are uncertainties in the system dynamics, the performance of these controllers may deteriorate and even lead to instability. The schemes given by Tarn et al. [76], Beekmann and Lee, [77] and Ge and Postlethwaite [78] deal with uncertainties in the manipulator alone and require full knowledge of the actuator parameters. To deal with uncertainties in the combined dynamics, some promising schemes were proposed by Krstic et al. [85], Grimm and Frank [88], Craig et al. [89] and Slotine and Li [98].
In this study, an adaptive controller is designed and implemented for constrained rigid robots including motor dynamics and parameter variations that guarantees zero force and tracking errors. The new adaptive controller developed does not require acceleration feedback.

6.2 Problem Statement

The equation of motion of a rigid body robot manipulator is given by Spong and Vidyasagar [5] as follows:

\[ M(q) \ddot{q} + C(q, \dot{q}) \dot{q} + g(q) = \tau + f \] (6.1)

where, \( q, \dot{q}, \ddot{q} \in R^n \), denote, respectively, the joint position, velocity and acceleration, while, \( M(q), C(q, \dot{q}) \in R^{n \times n} \), are the inertia and centrifugal matrices; \( g(q) \in R^n \), is the gravitation force. The left hand side of the above equation represents the rigid robot dynamics while the right hand side is the input torque, \( \tau \in R^n \), and the constraint force, \( f \in R^n \). This force is measured by a force sensor mounted on the end-effector and then converted to the joint space, where it is expressed as follows:

\[ f = J(q)^T \lambda \] (6.2)

where, \( J(q) = \frac{\partial \Phi}{\partial q} \), and, \( \Phi(q) = 0 \in R^m \), represents the set of, \( m \), nonlinear geometric constraint equations expressed in joint space. \( \lambda \in R^m \) is a generalized Lagrangian multiplier. Expressed in the joint space, the constraint equation may be written as follows:

\[ \frac{\partial \Phi}{\partial q} \dot{q} = J(q) \dot{q} = 0 \] (6.3)
The, $m$, differential constraints, described by, $J(q) \dot{q} = 0$, are linearly independent, as suggested by MaClamroch and Wang [105], it is possible to properly partition, $q^1 \in \mathbb{R}^{n-m}$, and, $q^2 \in \mathbb{R}^m$, such that $J(q) \dot{q} = J_1(q) \dot{q}^1 + J_2(q) \dot{q}^2 = 0$ where,

$$J_1(q) = \frac{\partial \Phi}{\partial q^1}, J_2(q) = \frac{\partial \dot{\Phi}}{\partial q^2}, \det[J_2(q)], \text{ and, } \det[J_2(q)] \neq 0 \forall q,$$

one may write

$$\dot{q} = L(q) \dot{q}^1, L(q) = \begin{bmatrix} I_{n-m} \\ -J_1^T(q)J_2^{-T}(q) \end{bmatrix} \quad (6.4)$$

where, $I_{n-m} \in \mathbb{R}^{(n-m) \times (n-m)}$, is an identity matrix. The above equation implies that $L^T(q) L(q) = I_{n-m} + J_1^T(q)J_2^{-T}(q)J_2^{-1}J_1(q) \neq 0$, therefore the matrix, $L^T(q) L(q)$, is uniformly continuous. It is clear from equation (6.4) the robot loses, $m$, degrees of freedom because of the constraints and thus it is not difficult to see that:

$$L^T(q)J^T(q) = 0 \quad (6.5)$$

Many robot controllers are based on this model with the assumption that the actuator dynamics are negligible. Recently, the effects of actuator dynamics, specifically motor dynamics, have attracted considerable research interest. For a robot driven by DC motors, the actuator dynamics are represented by:

$$H \ddot{\tau} + R \tau + K \dot{\tau} = u \quad (6.6)$$

where, $H, R, K$ are constant diagonal matrices representing the motors' armature inductance, resistance and electromechanical conversion coefficient, respectively. The real control input according to this analogy is the voltage signal, $u$, not the torque, $\tau$. The motor dynamics will inevitably affect the tracking performance of the robot, hence they should be
accounted for if high accuracy tracking is required.

The objective of this study is to synthesize an adaptive control signal, $u$, that ensures the convergence of robot joint position vector, $q$, to the desired position vector, $q_d$, and the constraint force, $f$, to the desired force to be exerted on the environment, $f_d$, when the exact robot model is not known \textit{a priori}. The uncertainty of $M(q), C(q, \dot{q})$ and, $g(q)$, is due to the uncertain robot inertia parameters that vary with different payloads. The robot dynamic equation could then be expressed as follows by substituting equation (6.4) in equation (6.1):

$$M(q)L(q)\ddot{q} + M(q)\dot{L}(q, \dot{q})\dot{q} + C(q, \dot{q})L(q)\dot{q} + g(q) = \tau + f$$  \hspace{1cm} (6.7)

In accordance with equation (6.4), a reference velocity may be represented by $\dot{q}_r = L(q)\dot{q}^1_r$, where, $\dot{q}^1_r = \dot{q}^1_d - \Lambda \ddot{q}^1$, and, $\Lambda \in \mathbb{R}^{(n-m) \times (n-m)}$, is a positive definite gain matrix. Throughout the rest of the chapter, and without loss of generality, functions, $F(q)$, and, $C(q, \dot{q})$, will be represented as, $F$, and, $C$, respectively. The reference acceleration is given by $\ddot{q}_r = L\ddot{q}_r^1 + \dot{L}\dot{q}_r^1$. Based on Slotine and Li [93] adaptive approach, a surface, $s = \dot{q}^1 - \dot{q}^1_r = \dot{q}^1 + \Lambda \ddot{q}^1$, may be created and, $\dot{s} = \dot{q}^1 - \dot{q}^1_r = \ddot{q}^1 + \Lambda \ddot{q}^1$, the dimension of $s$, $\dot{s}$, is reduced to, $n-m$. A new variable is developed for the force control part of the problem as follows:

$$\dot{\mu} + \kappa \mu = -\kappa \tilde{f}$$  \hspace{1cm} (6.8)

where, $\tilde{f} = f - f_d$, and, $\dot{q}^1 = q^1 - q^1_d$

The position control signal, $s \in \mathbb{R}^{n-m}$, and force control signal, $\mu \in \mathbb{R}^n$, are combined to form yet another control variable, $\sigma$, where:

$$\sigma = Ls + \mu \text{ and } \dot{\sigma} = L\dot{s} + \dot{L}s + \dot{\mu} \text{ or } \dot{\sigma} = L\dot{s} + \dot{L}s - \kappa (\tilde{f} + \mu)$$  \hspace{1cm} (6.9)
Since the acceleration measurement is not usually available form industrial robot controllers, the definition of the intermediate variable, \( w \), paves the way for the proposed adaptive controller to make use of, \( \dot{q}^1_r, \ddot{q}^1_r \) which can be easily calculated: Let

\[
\begin{align*}
\dot{w} &= \dot{q}^1_r - \mu = L\ddot{q}^1_r - \mu \quad \text{and} \\
\dddot{w} &= \dddot{q}^1_r - \mu = L\dddot{q}^1_r + L\dddot{q}^1_r + \kappa (\dddot{f} + \mu)
\end{align*}
\]  

(6.10)

Substituting, \( s = \dot{q}^1 - \dot{q}^1_r \), and, \( \mu = L\dddot{q}^1_r - w \), in equation (6.9) it is clear that:

\[
\begin{align*}
\sigma &= L\dot{q}^1 - w \quad \text{and} \quad \ddot{\sigma} = L\dddot{q}^1 + \dddot{L}q^1 - \dddot{w} \\
\tau &= -\Sigma \sigma - f_d + Y(\dot{w}, w, \dot{q}, q) \dot{\theta} \\
\dot{\theta} &= -\Gamma Y^T(\dot{w}, w, \dot{q}, q) \sigma
\end{align*}
\]  

(6.11) \hspace{1cm} (6.12) \hspace{1cm} (6.13)

where, \( \dot{\theta} \), is the estimate of the robot parameter vector. \( \Gamma \) is a diagonal, positive definite \( n \times n \) matrix. The regressor matrix is defined as:

\[
Y(\dot{w}, w, \dot{q}, q) \dot{\theta} = \dot{M}(q) \dot{w} + \dot{C}(q, q) w + \dot{g}(q)
\]  

(6.14)

where, \( \dot{M}, \dot{C} \) and, \( \dot{g}(q) \), are computed using the estimated parameter vector \( \dot{\theta} \). They are expressed as functions of \( q, \dot{q} \) because the full order model in equation (6.7) will be used.

**Property 1:** Matrices, \( M(q), L(q), g(q) \), and, \( J(q) \), are uniformly bounded and uniformly continuous if, \( q \), is uniformly bounded and continuous, respectively. Matrix, \( C(q, q) \), is uniformly bounded and continuous if, \( \dot{q} \), is uniformly bounded and continuous respectively. Similarly, The estimated, \( \dot{M}(q) \), and, \( \dot{g}(q) \), are uniformly bounded and continuous if the parameter estimate, \( \dot{\theta} \), is uniformly and continuous. The estimated regressor matrix, \( Y(\dot{w}, w, \dot{q}, q) \), is uniformly bounded and continuous if, \( w, \dot{w}, q \), and, \( \dot{q} \), are uniformly bounded and continuous.
Property 2: $\dot{L}$ is uniformly bounded and continuous provided that $\dot{q}$, is uniformly bounded and continuous.

Proof: Since, $\Phi(q)$, is twice continuously differentiable by the hypothesis found in Slotine and Li [97] and Beekmann and Lee [77], then, $\frac{\partial^2 J(q)}{\partial q^2}$, exists and is continuous $\forall q$.

Therefore, $\dot{J} = \begin{bmatrix} \dot{j}_1 & \dot{j}_2 \end{bmatrix}$, is uniformly continuous under the same conditions. The same conclusion could be reached for the uniform continuity of $\dot{L}$. Where, $\dot{L}$, is represented by:

$$\dot{L} = \begin{bmatrix} 0 \\ -J^{-1}_2 j_1 - J^{-1}_2 j_1 \end{bmatrix}$$

Similarly the boundedness condition could be satisfied.

6.3 Stability Analysis

Substituting equations (6.11) and (6.12) into equation (6.7), one obtains:

$$ML\ddot{q} + M\dot{L}\dot{q} + CL\dot{q} + g = -\Sigma\sigma - f_d + \dot{M}\dot{w} + \dot{C}w + \dot{g} + f$$ (6.15)

By adding and subtracting $M\dot{w} + Cw + g$, to the RHS of equation (6.15) it is clear that:

$$ML\ddot{q} + M\dot{L}\dot{q} + CL\dot{q} = -\Sigma\sigma + \tilde{f} + Y(\dot{w}, w, \dot{q}, q) \tilde{\theta} + M\dot{w} + Cw$$

where, $\tilde{f} = f - f_d$, is the force error and, $\tilde{\theta} = \hat{\theta} - \theta$, is the parameter vector error. Hence:

$$ML\ddot{q} + M\dot{L}\dot{q} - M\dot{w} + CL\dot{q} - Cw = -\Sigma\sigma + \tilde{f} + Y(\dot{w}, w, \dot{q}, q) \tilde{\theta}$$

which may be rewritten using equation (6.11) as follows:
\[ M\dot{\sigma} + C\sigma = -\Sigma\sigma + \ddot{f} + Y(\dot{w}, w, \dot{q}, q) \ddot{\theta} \]  \hspace{1cm} (6.16)

It is clear that the geometric and force constraints are embedded in equation (6.16) and will consequently affect the control law. The closed loop robot system, subject to the geometrical constraints is governed by equations (6.13) and (6.16).

**Lemma 1:** For the closed-loop system, represented by equations (6.13) and (6.16), the signals \( \sigma, \mu, \ddot{\theta}, \dot{q}, \) and \( \ddot{q} \) are uniformly bounded.

**Proof:** Consider the Lyapunov function candidate, Slotine and Li [102]:

\[ V = \frac{1}{2}\left[ \sigma^T M\sigma + \kappa^{-1} \mu^T \mu + \ddot{\theta}^T \Gamma^{-1} \ddot{\theta} \right] \]

Hence,

\[ \dot{V} = \frac{1}{2}\left[ \sigma^T M\sigma + \sigma^T M\dot{\sigma} + \sigma^T \dot{M}\sigma + \kappa^{-1} \mu^T \mu + \kappa^{-1} \mu^T \mu + \ddot{\theta}^T \Gamma^{-1} \ddot{\theta} + \ddot{\theta}^T \Gamma^{-1} \ddot{\theta} \right] \]  \hspace{1cm} (6.17)

substituting equation (6.16) in (6.17) for \( M\dot{\sigma} \), equation (6.8) for \( \dot{\mu} \), and equation (6.13) for \( \ddot{\theta} \), and since \( \dot{\hat{\theta}} = \hat{\dot{\theta}} \), then the following is deduced:

\[ \dot{V} = -\sigma^T \Sigma \dot{\sigma} + \dot{\sigma}^T \mu + \ddot{\theta}^T Y \sigma - \sigma^T \sigma + \sigma^T \mu - \mu^T \mu - \sigma^T C\sigma \]

\[ -\kappa^{-1} \mu^T \mu - \kappa^{-1} \kappa^T \mu - \kappa^{-1} \kappa^T \mu - \kappa^{-1} \kappa^T \mu - \sigma^T \sigma \]

\[ \frac{1}{2} \]

\[ \dot{V} = -2\sigma^T \Sigma \dot{\sigma} + \dot{\sigma}^T \mu - \mu^T \mu - \sigma^T \mu - \sigma^T \dot{M}\sigma \]

\[ -2\sigma^T \Sigma \dot{\sigma} + \dot{\sigma}^T \mu - \mu^T \mu - \sigma^T \mu - \sigma^T \dot{M}\sigma \]

\[ \frac{1}{2} \]

substituting \( \sigma^T \dot{f} = (Ls + \mu) \dot{f} = \mu \dot{f} = -\mu \left( \dot{\mu}^T + 1 + \mu \right) \) where equation (6.5) has been used to cancel out, the following is deduced:

\[ \dot{V} = -2\sigma^T \Sigma \dot{\sigma} - 2\sigma^T \mu^T \mu + \sigma^T \dot{M}\sigma \]

Using the skew-symmetric property \( \dot{M} - 2C = 0 \), it is finally concluded that:
\[ \dot{V} = -\sigma^T \Sigma \sigma - \mu^T \mu \leq 0 \quad (6.18) \]

along the trajectory of equations (6.13) and (6.16). Since, \( \dot{V} \), in equation (6.18) is negative semi-definite, it can be stated that, \( V \), is upper bounded. Using the facts that, \( V \), is upper bounded and, \( M(q) \), is a positive definite matrix, it can be concluded that \( \sigma \), \( \mu \) and \( \tilde{\theta} \) are uniformly bounded. According to equation (6.9), \( s = \left( L^T L \right)^{-1} L^T (\sigma - \mu) \), hence the uniform boundedness of, \( s \), becomes clear, since the first part of the Lemma proved, \( \sigma \), and, \( \mu \), to be uniformly bounded and similarly, \( L \), by property 1. Hence, the relation, \( s = \dot{q} + \Lambda \ddot{q} \), implies the uniform boundedness of, \( \dot{q} \), and, \( \ddot{q} \).

**Lemma 2:** For the closed loop system, represented by equations (6.13) and (6.16), \( \dot{\tilde{q}} \), \( \ddot{\tilde{q}} \), \( \tilde{\lambda} \) and, \( \tilde{f} \), are bounded.

**Proof:** This Lemma may be proven by multiplying equation (6.7) by \( J(q) M^{-1}(q) \). The effect of, \( L \ddot{q} \), is cancelled out using equation (6.5). The result may be arranged as:

\[ \lambda = \left[ J M^{-1} J^T \right]^{-1} J \left[ L \dot{q} + M^{-1} \left( C L \dot{q} + g - \tau \right) \right] \quad (6.19) \]

The matrices, \( M(q), L(q), g(q), C(\dot{q}, q), \) and, \( J(q) \), are uniformly bounded as shown in property 1. \( \dot{L}(q) \) is uniformly bounded by property 2. The variables, \( \sigma \), \( s \), \( \mu \) and, \( \tilde{\theta} \), are bounded by Lemma 1. The uniform boundedness of, \( Y(\dot{\tilde{w}}, w, \dot{\tilde{q}}, q) \), is guaranteed by the given uniform boundedness of, \( \dot{q}, \dot{\tilde{q}} \), by assumption. In addition, \( w, \dot{\tilde{w}} \), are uniformly bounded according to Lemma 1, property 2, equation (6.10) and the relations, \( \dot{\tilde{q}}_r = \dot{q}_d - \Lambda \ddot{q} \), and, \( \ddot{\tilde{q}}_r = \ddot{q}_d - \Lambda \dddot{q} \). Hence, it is implied that the control torque, synthesized by equation (6.12), is uniformly bounded. Therefore, \( \lambda, \lambda, \) and, \( \tilde{f} \), are uniformly bounded.

**Lemma 3:** For the closed loop represented by equations (6.13) and (6.16), the signals, \( \sigma \), and, \( \mu \), will converge to zero.
**Proof:** According to Barbalat’s Lemma, Tarn et al. [76], both sides of equation (6.18) converge to zero if \( \dot{V} \), is uniformly continuous or, \( \ddot{V} \), is uniformly bounded. Taking the time derivative of equation (6.18), one obtains:

\[
\dot{V} = -2 \left[ \sigma^T \Sigma \dot{\sigma} - \kappa \mu^T (\ddot{f} + \mu) \right] \tag{6.20}
\]

where equation (6.8) has been substituted for \( \dot{\mu} \). The uniform boundedness of \( \ddot{\sigma} = \ddot{q} - \dot{\omega} \), can be easily deduced using Lemma 2 and equation (6.10). Hence, \( \dot{V} \), is uniformly bounded as well. The Barbalat’s Lemma may now be applied to show, \( \dot{V} \rightarrow 0 \), which implies, \( \mu \rightarrow 0 \), and, \( \sigma \rightarrow 0 \).

The convergence of, \( s \rightarrow 0 \), \( \dot{s} \rightarrow 0 \), and, \( q^1 \rightarrow 0 \), can be easily deduced from the final argument in Lemma 1. Therefore, the motion control part of the problem is solved. The remaining task is to prove that \( \ddot{f} \rightarrow 0 \). As implied by equation (6.8), one may write, \( \ddot{f} = -(\dot{\mu} \kappa^{-1} + \mu) \). Lemma 3 has already established that, \( \mu \rightarrow 0 \). The convergence of, \( \ddot{f} \), is now related to the convergence of \( \dot{\mu} \). Clearly, \( \dot{\mu} = -\kappa (\mu + \dot{f}) \), is uniformly bounded. It follows that, \( \dot{\mu} = -\kappa (\mu + \dot{f}) \), is also uniformly bounded because both, \( \dot{\mu} \), and, \( \dot{f} \), are uniformly bounded. Barbalat’s Lemma is once again used to establish that \( \dot{\mu} \rightarrow 0 \). For the closed loop system represented by equations (6.13) and (6.16), the tracking error, \( \tilde{q} \), and the force error, \( \tilde{f} \), are asymptotically stable. The focus of the rest of this chapter is to study the effects of the motor dynamics on the stability of the closed loop robot system. The dynamic equation is given by equation (6.6) whereby a possible control law is given by:

\[
u = H\dot{\tau}_d + R\tau_d + K\dot{q} + G\tilde{\tau} \tag{6.21}
\]

where, \( \tilde{\tau} = \tau - \tau_d \), the error between the actual and the desired torques, \( G \), is a positive definite gain matrix. The closed-loop motor dynamics is obtained by substituting the
control law in equation (6.21) into equation (6.6) which yields:

\[ H\ddot{\tau} = -G\ddot{\tau} \]  \hspace{1cm} (6.22)

The stability of \( \ddot{\tau} \), can be established via a Lyapunov function candidate, \( V_m = \frac{1}{2} \ddot{\tau}^T H\ddot{\tau} \).

Using equation (6.22) and taking the derivative of \( V_m \), it is clear that:

\[ \dot{V}_m = -\ddot{\tau}^T G\ddot{\tau} \leq 0 \]  \hspace{1cm} (6.23)

In this study, \( \tau_d \), is synthesized by:

\[ \tau_d = -\Sigma \sigma_d - f_d + Y(\ddot{q}_d, \dot{q}_d, q) \dot{\theta} \]  \hspace{1cm} (6.24)

where, \( \sigma_d = Lq_d \cdot 1 - \dot{w} \), and \( \dot{\sigma}_d = Lq_d \cdot 1 + Lq_d \cdot 1 - \dot{w} \), where \( q_d \cdot 1 = L(q) \cdot 1 q_d \), and \( q_d \cdot 1 = L(q) \cdot 1 q_d \).

The stability analysis conducted so far has focused on robot and motor dynamics as two separate systems. In order to examine the stability of the overall system, however, the coupling effects of the two systems must be addressed. When motor dynamics are taken into effect, equation (6.16) is no longer valid to describe the closed loop dynamics. Instead, it should be written as:

\[ M\ddot{\theta} + C\dot{\theta} = -\Sigma \sigma + \ddot{f} + Y(\dot{w}, w, \dot{q}, q) \dot{\theta} + \ddot{\tau} \]  \hspace{1cm} (6.25)

**Lemma 4:** For the closed loop system described by equations (6.13), (6.22) and (6.25), the signals, \( \sigma, \mu, \dot{\theta} \) and, \( \ddot{\tau} \), are uniformly bounded.

**Proof:** Consider a Lyapunov function candidate, \( \psi = V + V_m \), where:

\[ V = \frac{1}{2} \left[ \sigma^T M\sigma + \kappa^{-1} \mu^T \mu + \dot{\theta}^T \Gamma^{-1} \dot{\theta} \right] \]  and \( V_m = \frac{1}{2} \ddot{\tau}^T H\ddot{\tau} \). Evaluating, \( \psi \), along the trajectory of equations (6.13), (6.22) and (6.25), equation (6.26), is easily deduced that:
\[ \psi \leq -\sigma^T \Sigma \sigma - \mu^T \mu - \tilde{r}^T G \tilde{r} \]  

(6.26)

From Lemma (1.4) one can deduce that for the overall closed-loop system, the tracking error, \( \tilde{q} \), and the force error, \( \tilde{f} \), are asymptotically stable.

### 6.4 Simulation

The following simulation demonstrates the performance of the proposed controller. The controller is applied to a two-link rigid robot where the constrained path to be followed is represented by a circle of radius, \( r = \sqrt{l_1^2 + l_2^2 + 2l_1l_2 \cos(q_2)} \), where, \( l_1 \), and, \( l_2 \), are the lengths of the two robot links. It is clear that: 
\[
\Phi(q) = l_1^2 + l_2^2 + 2l_1l_2 \cos(q_2) - r^2 = 0 \text{ then } J(q) = \begin{bmatrix} 0 & -2l_1l_2 \sin(q_2) \end{bmatrix}.
\]

The desired joint trajectory is given by:

\[
q_{1d} = \pi \left( 1 - \cos(\pi t) \right) \text{ and } q_{2d} = \frac{\pi}{2}
\]

The desired constraint force is always along the radius of the circular path, pointing outwards, and its magnitude in Newtons is given by:

\[
f_d = \begin{cases} 
20 & \text{for } 0 \leq t \leq 5 \\
30 - 10\cos\left(\frac{\pi}{2}(t - 5.0)\right) & \text{for } 5 \leq t \leq 15 \text{ N.} \\
20 & \text{for } 15 < t
\end{cases}
\]

The robot dynamic coefficient matrices are as follows:

\[
M(q) = \begin{bmatrix} 
m_2l_2(2l_1 \cos(q_2) + l_2) + l_1^2(m_1 + m_2) & l_1^2m_2 + l_1l_2 \cos(q_2)m_2 \\
l_2^2m_2 + l_1l_2 \cos(q_2)m_2 & l_2^2m_2
\end{bmatrix}
\]
\[ C(q, \dot{q}) = \begin{bmatrix} -l_1 l_2 m_2 \sin(q_2) q_2 \dot{q}_2 & -l_1 l_2 m_2 \sin(q_2)(q_1 + \dot{q}_2) \\ l_1 l_2 m_2 \sin(q_2) q_1 \dot{q}_1 & 0 \end{bmatrix} \]

\[ g(q) = \begin{bmatrix} 9.8 (m_2 l_2 \cos(q_1 + q_2) + (m_1 + m_2) l_1 \cos(q_1)) \\ 9.8 m_2 l_2 \cos(q_1 + q_2) \end{bmatrix} \]

The regressor \( Y(q_d, \dot{q}_d, \ddot{q}) \) is a 2 by 3 matrix, the entries of which \( y_{ij} \) are derived as follows:

\[ y_{11} = y_{21} = q_{1d} \ddot{q} + q_{2d} \ddot{q} \]
\[ y_{12} = l_1 \left[ (2q_{1d} \ddot{q} + q_{2d}) \cos(q_2) - (2q_{1d} \dot{q} + \dot{q}_{2d}) q_{2d} \sin(q_2) \right] + 9.8 \cos(q_1 + q_2) \]
\[ y_{13} = l_1 q_{1d} \ddot{q} + 9.8 \cos(q_1) \]
\[ y_{22} = l_1 \left[ q_{1d} \ddot{q} \cos(q_2) + q_{1d} \dot{q} \ddot{q} \sin(q_2) \right] + 9.8 \cos(q_1 + q_2) \]
\[ y_{23} = 0 \]

The components of \( p_r \) are \( p_{r1} = l_1 (m_1 + m_2) \), \( p_{r2} = l_2 m_2 \), and \( p_{r3} = l_2^2 m_2 \). The actual robot parameters are: \( l_1 = 0.5 m \), and \( m_1 = m_2 = 5 Kg \). A limited knowledge is assumed about those parameters and the nominal parameters utilized by the controller are initially set as follows: \( p_{r1} = l_1 (\hat{m}_1 + \hat{m}_2) \), \( p_{r2} = l_2 \hat{m}_2 \), and \( p_{r3} = l_2^2 \hat{m}_2 \). The initial estimate of the two masses are: \( \hat{m}_1 = m_1 \) and \( \hat{m}_2 = 1 Kg \). This large difference between the actual and estimated value of the mass \( m_2 \) represents the uncertainty due to the payload. The motor dynamics coefficient matrices are given by:
\[ H = \begin{bmatrix} 0.0029 & 0 \\ 0 & 0.0017 \end{bmatrix}, R = \begin{bmatrix} 0.96 & 0 \\ 0 & 0.55 \end{bmatrix}, K = \begin{bmatrix} 11.88 & 0 \\ 0 & 20.48 \end{bmatrix} \]

The controller parameters are chosen to be, \( \Lambda = 10, \Gamma = I, \Sigma = 100I, \) and, \( \kappa = 1, \) respectively, where, \( I, \) is the identity matrix. Good tracking performance is demonstrated through plotting the tracking error as shown in Figure (6.1). The zero force error is illustrated in Figure (6.2). The control torques are plotted in Figures (6.3) and (6.4) for the first and second joint, respectively. The transit in the tracking error response in Figure (6.1) is due to the difference between the payload mass \( m_2 = 5Kg \) and \( \hat{m}_2 = 1Kg. \) The high transit in the constraint error response in Figure (6.2) is due to the transition from free to constrained motion. In Figure (6.3), the high torque of the first joint compared to the lower torque of the second joint in Figure (6.4) is because, the first joint has to move the first link, second motor and second link which are considered as load on the first joint motor. Secondly the first joint motor is responsible for compensating the coupling dynamics of the second link. The second joint motor is only responsible for exerting a desired radial contact force pointing outwards. The non-symmetry about the \( x \) axis in Figure (6.4) is due to the combined effect of actuator dynamics, payload variation and environment interaction which are characteristics of the second link (motor).

### 6.5 Conclusion

In this chapter, an adaptive controller which ensures zero force and tracking errors has been proposed. The control objectives are achieved in the presence of uncertainty. The currently available robot controllers in this class require acceleration feedback and exact
knowledge of both the robot and motor dynamics. The regressor can be computed with algorithms developed by Stepanenko, and Yuan [102], Niemeyer, and Slotine [108], Arimoto et al. [110] and Hu et al. [111]. The presented method departs from previously mentioned approaches in that it establishes global as opposed to local convergence, in the presence of parameters variation and unmodelled dynamics. In this chapter, a new approach addressing this problem by adopting an adaptive motion controller that does not require acceleration measurements has been presented and simulated. The stability of the proposed controller tracking performance is proven in the Lyapunov sense.
Figure 6.1: Tracking error for first joint

Figure 6.2: Constraint force error
Figure 6.3: Control torque for first joint

Figure 6.4: Control torque for second joint
Conclusion and Future Work

7.1 Conclusion

The main objective of this study was to develop an adaptive assembly system aimed at enhancing automotive Body-In-White assembly. The objective was achieved in two major steps. The first was the integration of sensor feedback with the industrial robot controller. The proposed method of sensor integration allowed positional adjustment to accommodate workpiece pose variation. Workcell and sensor modelling were conducted in the simulation environment. Off-line programming was performed in the simulation environment and then downloaded for verification in the actual workcell.

The second step was through the design of robust and adaptive controllers. These controllers minimize the dynamic errors due to parameter variations, in conjunction with payload variation for various subassemblies, unmodelled dynamics resulting from the actuator dynamics.

The proposed adaptive assembly system is cost effective and does not require any dedicated equipment. A workcell calibration system was designed and implemented in an industrial setup. After calibration was achieved, the off-line program performed in the simulation environment proved to be very accurate when downloaded to the actual workcell. In addition, the assembly operation required parts mating which entailed force
control. This motivated the design and implementation of adaptive control strategies ensuring zero force and tracking errors.

In Chapter 1, the background, motivation and objectives for this research have been introduced.

In Chapter 2, a literature review for current state-of-the-art related technologies is presented.

In Chapter 3, issues related to static errors in the part, fixture and robot pose, are addressed and an approach to minimize these errors is developed and implemented. A modified regressor and controller are presented to adapt to dynamic parameter variations without the need to explicitly calculate the varying dynamic parameters. An approach to integrate the sensors’ feedback is introduced. The method of nonlinear optimization is used to best fit the measured part geometry to the predefined master part geometry. As a result, a transformation matrix representing the rigid body motion is computed and used to modify the robot end effector path, originally designed for the master part geometry. In addition, the adaptive assembly system components, and the technique by which they are integrated, are explained. The design and implementation issues of a robot base and tool calibration systems are also presented with results.

Chapter 4, 5 and 6 discuss how a real robot-actuator system behaves under various advanced control techniques, which is of great importance in practical robot applications, and is the motivation for work pursued in these chapters. The objective of the work presented in the above mentioned chapters was to minimize the dynamic errors due to parameters variations and unmodelled dynamics. Currently available robot controllers require exact knowledge of both robot and motor dynamics, and acceleration feedback.
Chapters 4, 5 and 6 present a new approach that does not require exact knowledge of the robot-actuator system. In chapter 4, this is achieved by using adaptive feedback linearization techniques. The results demonstrate an asymptotic tracking response for the output. This is achieved using minimum control through the application of optimal control techniques which exploit the Hamilton-Jacobi equation.

Chapter 5, presents a different approach to the same problem by adopting a robust adaptive motion controller that requires only position measurements. The global stability of the proposed controller tracking performance is proved in the Lyapunov sense. In addition, simulation results are presented to demonstrate the asymptotic tracking performance of the closed-loop system. Moreover, it does not assume the availability of velocity measurements which, for practical purposes, may not be readily available.

In Chapter 6, an adaptive controller ensuring zero force and tracking errors, has been proposed. The stability of the proposed controller tracking performance is proven in the Lyapunov sense. The presented method establishes global, rather than local, convergence in the presence of unmodelled dynamics. In addition, simulation results are presented to demonstrate the asymptotic tracking performance for position and force trajectories of the closed-loop system.

To implement the control algorithms developed in this thesis on the industrial workcell at the IMS centre, the robot controller must be retrofitted to allow the researcher full access to the motion control system and path planning functions. The motion control system provides a set of robot functions. These functions could form a software library where the internal implementation is hidden. There are proprietary and safety reasons for having a closed system, which is also easier to implement than an open system. This explains why several of the currently available robot control systems seem to have such a closed control architecture. This is a major reason for the difficulty in slightly modifying the motion control or including new sensors, etc.
The retrofitting should be approached with the goal of maintaining the mechanics, power electronics, and as much as possible of the safety system intact. A sensor interface had to be built to feed sensor measurements back to the robot controller. The Comau S2 robot would be controlled from VME-based embedded computers. Digital signal processors (DSP) would be used for low-level control and filtering of sensor signals. An additional DSP board should be added to accommodate force-torque sensor. The control signal (torque reference) is then passed to the actuator object (C,C++ code) describing appropriate control routines. The output of these routines is placed in an array for the output interrupt routine.

The built-in control of the robot arm supplies the motion control layer with:

- Forward and inverse kinematics for the robot arm,
- Dynamic parameters for the robot arm,
- Initial conditions for the robot arm and controller parameters,
- Desired trajectories,
- Torque feed-forward reference signals.
- Acquiring of control data used in the robot arm control such as:

  1- Actual trajectories,
  2- Sensor signals,
  3- Control error signals for position, velocity and force.

From the results of this study, the concept of adaptive assembly using range sensing and robotic manipulation, which includes adaptive and robust control, appears to be quite promising in improving assembly process performance through the minimization of static
and dynamic errors.

7.2 Future Work

The scope of this work can be extended in several directions, such as the following:

1- The application of adaptive feedback linearization for industrial applications that require on-line trajectory (position, velocity and acceleration) adjustment. So far, very few research results about applying such advanced control techniques have been reported in industry.

2- The proposed workpiece calibration procedure included only three measurement directions. The need for extension to all six cartesian directions, is obvious, especially if the main assembly and / or subassemblies are not assumed to be rigid.

3- The work could be extended to include the dynamics and control of the robot and of the assembly process, whereby the latter is augmented by a description of the contact process for mating parts, including impact, friction and stiction forces.

4- Software enhancement for code optimization, robustness and portability should be conducted before software deployment on the shop floor. However, the problem of adaptability has been partly solved through the structuring of all functional units of the supervisory controller into plant independent (task planning, off-line programming, workcell calibration and communication between functional units) and plant dependant (sensing and control).
REFERENCES


[43] Trellis Software & Controls, Rochester Hills, Michigan, USA.


[58] Nicosia, S. and Tornambe, A., "Use of asymptotic observers in the parameter estimation of robotic manipulators having elastic joints", *2nd Univ. Degli Studi di Ro-


Craig, J. Adaptive Control of Mechanical Manipulators, Addison-Wesley, 1986.


Systems Conference (IROS), Raleigh, NC, August, 1992.


1983.


[94] Slotine, J. E. and Li, W., "Adaptive Manipulator Control: A Case Study", *IEEE


[118] Jankowski, K. P. and ElMaraghy, H. A. “Inverse Dynamics and Feedforward con-


VITA AUCTORIS

NAME: Yasser Mohamed Anwar Eldeeb
PLACE OF BIRTH: Cairo, Egypt
YEAR OF BIRTH: 1963
POST-SECONDARY EDUCATION AND DEGREES:
Alexandria University,
Alexandria, Egypt.
1980 - 1985 B.Sc. (Hon) Mechanical Engineering

The University of Western Ontario
London, Ontario, Canada

RELATED WORK EXPERIENCE:
Research Assistant
The National Research Center, Cairo
1987-1992
Teaching Assistant
Mechanical Engineering Department,
The University of Western Ontario
1992 - 1994
Research Assistant
Advanced Manufacturing Lab.
The University of Western Ontario
1992 - 1994
Teaching Assistant,
Industrial and Manufacturing Systems Engineering Program,
The University of Windsor
1995 to date
Research Assistant
Intelligent Manufacturing Systems Centre,
Advanced Manufacturing Lab.,
The University of Windsor
1995 to date
PUBLICATIONS:

Journal Publications


Conference Publications


