Convergence Analysis of

ILC Algorithms with application to

Compass Gait Bipedal Walking Robot

A thesis submitted to The University of Manchester for the degree of

Doctor of Philosophy

in the Faculty of Engineering and Physical Sciences

2013

Inam Ul Hasan Shaikh

School of Electrical and Electronic Engineering
# TABLE OF CONTENTS

Table of contents ........................................................................................................ 2

List of Figures ............................................................................................................. 7

List of tables .............................................................................................................. 11

Abbreviations and Acronyms .................................................................................. 12

Abstract ...................................................................................................................... 13

Declaration ................................................................................................................. 16

Copyright Statement ................................................................................................. 17

DEDICATION ........................................................................................................... 19

Acknowledgements ................................................................................................... 20

List of publications related to this thesis ................................................................. 22

1 Introduction ............................................................................................................ 23

1.1 Motivation ........................................................................................................... 24

1.2 Problem Formulation ......................................................................................... 26

1.3 Thesis Contributions and Achievements ......................................................... 27

1.4 Thesis Outline ................................................................................................... 29

2 ILC and Bipedal Walking - Literature Review ............................................... 31

2.1 Bipedal Humanoid Robots .............................................................................. 31

2.2 Passive Dynamics for Walking Robots ............................................................ 33
3 Modelling the Dynamics of Bipedal Walking Robot

3.1 Two-Legged Human Locomotion

3.1.1 Compass Gait Model for Biped Robot

3.1.2 Assumptions for Compass Gait Model

3.2 Dynamics of Compass Gait Robot

3.2.1 Swing Phase Dynamics in State Space Form

3.2.2 Impact Reset During Stance Phase
4.7 Case Study: ILC Algorithm for a Damped Pendulum..........................110
4.7.2 Controllability via D-ILC ..............................................................122
4.7.3 Effect of Offset State Errors.......................................................122
4.8 Summary .........................................................................................123

5 Convergence Analysis for Cyclic ILC.................................................125

5.1 Control Input & Initial State Errors for Cyclic D-ILC...............125
5.1.1 Problem Formulation .................................................................126
5.1.2 Modelling Cyclic ILC.................................................................127
5.1.3 Evolution of Control Input and Initial State Errors...............128
5.1.4 Comments on the Structure of Error Evolution Matrix .........128
5.1.5 Convergence of Error Evolution Matrix ..................................129
5.2 State Errors for Cyclic D-ILC .........................................................135
5.3 Convergence of Cyclic PD–ILC ......................................................136
5.4 Evolution of State Errors for Cyclic PD-ILC ............................140
5.5 Damped Pendulum with Cyclic ILC- Simulations.....................141
5.5.1 CASE 1: Cyclic D-ILC .............................................................142
5.5.2 CASE 2: Cyclic PD-ILC ............................................................150
5.5.3 CASE 3: Cyclic PD-ILC Using P Gain Less Than D Gain ......153
5.6 Parameters for Stable Cyclic PD-ILC Learning .......................159
5.7 Summary .......................................................................................160

6 ILC for Robotics .................................................................................161

6.1 Cyclic D-ILC for Actuated Compass Gait Robot....................161
6.1.1 Simulations of CG Robot Walking on Flat Surface Actuated
with Cyclic D-ILC Generated Torques..............................................162
6.1.2 Simulations of CG Walking Downhill with Cyclic D-ILC ............ 164

6.2 Modified Cyclic PD-ILC for Bipedal Walk................................. 166

6.3 Simulations for CG Walk Using Modified Cyclic PD-ILC .......... 166

6.4 Summary ......................................................................................... 174

7 Conclusions .......................................................................................... 175

APPENDIX A .............................................................................................. 177

APPENDIX B............................................................................................... 181

REFERENCES ............................................................................................. 182

Word Count: 25,426
LIST OF FIGURES

Figure 2.1: Basic ILC scheme ............................................................... 40
Figure 2.2: Feedback versus learning .................................................. 41
Figure 3.1 The 2-link compass gait robot ........................................... 54
Figure 3.2 Pre-impact and post-impact during double support
phase .................................................................................................. 60
Figure 3.3 Stick diagram of CG robot walking on flat surface .............. 62
Figure 3.4 Phase portrait of a selected leg of CG robot for 10 steps ....... 63
Figure 3.5 Compass gait robot walking down a slope ......................... 64
Figure 3.6 Stick diagram of CG robot walking down hill ..................... 66
Figure 3.7 Stable limit cycle for CG walking on slope ......................... 66
Figure 3.8 Comparison between nonlinear and linear models of CG...... 73
Figure 3.9 Effects of +/- 20 % change in hip mass of CG robot .......... 81
Figure 3.10 Effects of +/- 20 % change in leg mass of CG robot ........... 82
Figure 4.1: Sequential convergence of samples .................................... 94
Figure 4.2: Singular values and eigenvalues of matrix $T$ ..................... 105
Figure 4.3: Evolution of components of control input errors ............... 106
Figure 4.4: Short range evolution of control input errors ................. 106
Figure 4.5: Medium range evolution of control input errors ............... 107
Figure 4.6: Long range evolution of control input errors ................. 108
Figure 4.7: Convergence rates of control input components............. 108
Figure 4.8: Bounds for rate of convergence ................................. 109
Figure 4.9: A damped pendulum .................................................. 111
Figure 4.10: Tracking control input with ILC algorithm .................. 113
Figure 4.11: Tracking desired angle and velocity using ILC .......... 114
Figure 4.12: RMS control input, angle and velocity errors .......... 114
Figure 4.13: Evolution of control input at selected indices .......... 115
Figure 4.14: Convergence rates for control input errors ............... 116
Figure 4.15: Short-term rates for control input errors ................. 117
Figure 4.16: Medium-term rates for control input errors .......... 117
Figure 4.17: Long-term rates for control input errors ................. 118
Figure 4.18: RMS errors in angle, velocity and control input .... 119
Figure 4.19: Evolution of input error at selected time indices .... 119
Figure 4.20: Long-term rates for control input errors ............... 120
Figure 4.21: RMS values for D-ILC with zero initial conditions .... 121
Figure 4.22: Convergence rate for input error ......................... 121
Figure 4.23: Convergence of velocity errors with offset state errors .. 123
Figure 5.1: Root mean squared initial state and input errors ........ 145
Figure 5.2: Tracking of desired control input .......................... 146
Figure 5.3: Tracking of desired angle and velocity .................. 146
Figure 5.4: Evolution of input error at selected time indices .... 147
Figure 5.5: Convergence rate of input error at selected indices .... 147
Figure 5.6: Evolution of angle error at selected time indices .............. 148
Figure 5.7: Evolution of velocity error at selected time indices .......... 148
Figure 5.8: Convergence rate for velocity error ................................. 149
Figure 5.9: RMS initial state and input errors using Cyclic PD-ILC .......................................................... 150
Figure 5.10: Tracking of control input using Cyclic PD-ILC ............... 151
Figure 5.11: Tracking of angle and velocity using Cyclic PD-ILC ...... 151
Figure 5.12: Evolution of input error using Cyclic PD-ILC ............... 152
Figure 5.13: Convergence rate of control input errors using Cyclic PD-ILC .......................................................... 152
Figure 5.14: RMS initial state and input errors ................................. 153
Figure 5.15: Tracking for different initial states with Cyclic PD-ILC .......................................................... 154
Figure 5.16: Tracking input for different initial states using Cyclic PD-ILC .......................................................... 154
Figure 5.17: Input errors evolution with Cyclic PD-ILC ...................... 155
Figure 5.18: Long-term rate of input error with Cyclic PD-ILC ........... 157
Figure 5.19: Evolution of angle error using Cyclic PD-ILC ............... 157
Figure 5.20: Evolution of velocity error using Cyclic PD-ILC ........... 158
Figure 5.21: Stability boundary for Cyclic PD-ILC ............................ 159
Figure 6.1: Torques generated using Cyclic D-ILC on flat surface ...... 163
Figure 6.2: Phase portrait on flat surface using Cyclic D-ILC .......... 163
Figure 6.3: Torques using Cyclic D-ILC for CG robot walking downhill .................................................................................................................. 164

Figure 6.4: Limit cycle - CG robot walking on slope ......................... 165

Figure 6.5: Cyclic D-ILC torques for CG using reduced $K_d$ .............. 165

Figure 6.6: Tracking ankle joint torque.............................................. 167

Figure 6.7: Tracking hip joint torque............................................... 168

Figure 6.8: Tracking stance foot angle ............................................ 168

Figure 6.9: Tracking hip joint angle............................................... 169

Figure 6.10: Evolution of ankle input torque error ......................... 170

Figure 6.11: Slopes showing rate of ankle input torque error ........... 170

Figure 6.12: Evolution of Hip Joint torque ........................................ 171

Figure 6.13: Slopes showing rate of hip joint error ......................... 171

Figure 6.14: Log-log RMS values of state and input errors............... 172

Figure 6.15: Phase portrait of ankle joint at last step....................... 172

Figure 6.16: Phase portrait of hip joint at last step.......................... 173

Figure 6.17: RMS values with modified Cyclic PD-ILC....................... 173
LIST OF TABLES

Table 3-1 Compass gait robot parameters .............................................. 62
Table 4-1: Pendulum parameters .......................................................... 111
Table 4-2: Final RMS values ............................................................... 113
# ABBREVIATIONS AND ACRONYMS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILC</td>
<td>Iterative Learning Control</td>
</tr>
<tr>
<td>RC</td>
<td>Repetitive Control</td>
</tr>
<tr>
<td>NRILC</td>
<td>No-reset ILC</td>
</tr>
<tr>
<td>RMS</td>
<td>Root Mean Square</td>
</tr>
<tr>
<td>P</td>
<td>Proportional</td>
</tr>
<tr>
<td>PD</td>
<td>Proportional – Derivative</td>
</tr>
<tr>
<td>PID</td>
<td>Proportional – Integral – Derivative</td>
</tr>
<tr>
<td>SISO</td>
<td>Single Input Single Output</td>
</tr>
<tr>
<td>MIMO</td>
<td>Multiple Input Multiple Output</td>
</tr>
<tr>
<td>LTI</td>
<td>Linear Time Invariant</td>
</tr>
<tr>
<td>CITE</td>
<td>Current Iteration Tracking Error</td>
</tr>
<tr>
<td>DOF</td>
<td>Degrees Of Freedom</td>
</tr>
<tr>
<td>AIST</td>
<td>Advanced Industrial Science and Technology</td>
</tr>
<tr>
<td>KAIST</td>
<td>Korea Advanced Institute of Science and Technology</td>
</tr>
<tr>
<td>ASIMO</td>
<td>Advanced Step in Innovative MObility</td>
</tr>
<tr>
<td>HRP</td>
<td>Humanoid Robotics Project</td>
</tr>
<tr>
<td>WABIAN</td>
<td>WAseda BIpedal humANoid</td>
</tr>
<tr>
<td>IIT</td>
<td>Italian Institute of Technology</td>
</tr>
<tr>
<td>COP</td>
<td>Centre of Pressure</td>
</tr>
<tr>
<td>ZMP</td>
<td>Zero Moment Point</td>
</tr>
<tr>
<td>LQR</td>
<td>Linear Quadratic Regulator</td>
</tr>
<tr>
<td>CG</td>
<td>Compass Gait</td>
</tr>
<tr>
<td>ASP</td>
<td>Average Speed of Progression</td>
</tr>
<tr>
<td>IFT</td>
<td>Iterative Feedback Tuning</td>
</tr>
</tbody>
</table>
ABSTRACT

At an early age, i.e., up to about 1-2 years the humans learn to walk and subsequently develop a robust and flexible gait. This is learned by repetitively taking similar steps and the experience is stored in the muscle/reflexive memory. Over the last 30 years, a variety of humanoid bipedal robots have been developed to copy the human gait. However, walking/locomotion is still a relatively difficult control problem due to its complex hybrid nature because of non-smooth dynamics. Although, simple walking comprises of single support in which one leg swings forward, then it impacts with ground for a brief double support phase and further transition of the other support leg to start a new swing. The steps are repeated again and again in a similar manner for walking over an even surface. As the swinging leg strikes the ground, it is a non-linear impact which poses a challenge since it causes non-zero initial state errors for each step which depend on the error in the gait at last moment for previous step. The usual bipedal control relies on complex techniques based on inverse kinematics, ZMP (Zero-Moment Pole) and COP (Centre Of Pressure) to generate the required control inputs for the joints. However, a basic cognitive assumption is that walking is a relatively simple task which can be learned and the biological systems have achieved it by simple repetitions. This has been over-looked in these control techniques.

In the past, ILC has been proposed to solve the repetitive learning problems. The Iterative Learning Controller learns to generate the desired set of input signals to compensate for the output tracking errors in a
sequential manner such that in the initial iterations, the signals values at earlier time indices have faster rate than the later ones. So, at the last time index the convergence is achieved after all the earlier ones. ILC learns/adapts the joint control for repetitive gaits. In this thesis it has been proposed to be used as a muscle memory where control signals are learnt for a repetitive batch. Thus, ILC equates to “learning a sequence of action by muscles”. Due to the transfer of state error in a cyclic manner from the end of a previous step/repetition to the recent step/repetition, the convergence has to be established in joint control and state space. Similar is the case of continuous walking where the ground impacts transfer part of the error in the gait to the start of a new step representing an impacting Cyclic ILC scenario. Hence, the ILC problem is changed from finite to an infinite horizon. The second problem occurs with the non-constant length of the iteration due to change in step size.

The two scenarios have been considered: Firstly, when the control input is updated using ILC with identical initial conditions at the start of each repetition. Secondly, control input update under varying initial conditions leading to Cyclic ILC. The batch to batch evolution of control inputs at each sample time within a batch is formulated. The sequential convergence of control input generated by ILC algorithms has been investigated. The exact relationship for the rate of convergence of the control input has been formulated down to the sample-time level. This provides deeper insight about the ILC algorithms and hence exact factors affecting the convergence could be established. Limits of the learning process have been clearly demonstrated as well. Although, simpler D-ILC converges for zero initial error but for cyclic non-zero initial errors, it has
offset error which corresponds to the initial state error. With proportional part, the PD-ILC algorithm has eliminated the offset error which has been illustrated for a damped pendulum and further implemented to bipedal locomotion. For reasons of energy efficiency, passive dynamics has been chosen for the compass gait model of the biped. The walking problem for the compass gait robot has been solved using the modified PD-ILC which utilizes the acceleration error term as well. The steady gait has been achieved for the compass gait robot on flat surface which has been verified by the phase portraits.
DECLARATION

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.
COPYRIGHT STATEMENT

[i] The author of this thesis (including any appendices and/or schedules to this thesis) owns any copyright in it (the “Copyright”) and he has given The University of Manchester the right to use such Copyright for any administrative, promotional, educational and/or teaching purposes.

[ii] Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, must be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with the licensing agreements which the University has from time to time. This page must form part of any such copies made.

[iii] The ownership of any patents, designs, trade marks and any and all other intellectual property rights except for the Copyright (the “Intellectual Property Rights”) and any reproductions of copyright works, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property Rights and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property Rights and/or Reproductions.
Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property Rights and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/display.aspx?DocID=487), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see http://www.library.manchester.ac.uk/aboutus/regulations/) and in The University’s Policy on Presentation of Theses.
DEDICATION

To

My Parents

Father:

Late Azimuddin Ahmad (Rahmatullah alaihe)

and

Mother:

Tasneem Sultana.
ACKNOWLEDGEMENTS

First and foremost, I am thankful to the Almighty Allah for He gave me the strength to pursue and complete this research. It gives me great pleasure to offer gratitude to my supervisor Dr. Martin Brown for his untiring support and guidance throughout my stay at Manchester. He groomed me up from roots and enabled to put up this novel work. Further, I would like to express my appreciation for the help and motivation from Prof. Hong Wang, Dr. Ahmad Hussain Al-Bayati, Dr. Onder Tutsoy and many other colleagues at Control Systems Centre. Some other kind friends whose moral support shall be remembered are Shafiq Qureshi, Saad Ahmad, Muhammad Kibria, Dr. Waseem Aziz, Dr. Zahid Mahmood, Dr. Muhammad Ali, Dr. Sarmad Sohaib, Dr. Junaid Ahmed, Dr. Laraib, Dr. Rashid Saleem, Dr. Muhammad Yaqub, Shabbir Majeed Choudhry, Dr. Muhammad ObaidUllah and Dr. Ahsan Ali.

I am also thankful for the financial support from University of Engineering and Technology, Taxila, Pakistan. Dr. Muhammad Abbas Choudhry, The Vice Chancellor of UET Taxila, Ex-VC Prof. Dr. Habibullah Jamal along with Prof. Dr. Ahmed Khalil Khan, Prof. Dr. Zafrullah, Prof. Dr. Tahir Nadeem Malik, Prof. Dr. Muhammad Ahmad Choudhry and Prof. Dr. Muhammad Iram Baig, all have continued to support my endeavour.
I am obliged to following for their useful comments: Dr. Fida Muhammad Khan, Dr. Sohail Akhtar and Dr. Syed Zafar Ali from Air University, Islamabad. Dr. Amir Iqbal Bhatti from MAJU, Islamabad, Pakistan gave deep insight about my research.

Lastly, but most importantly, the patience and motivation from my family during whole of my study has been the catalyst.
LIST OF PUBLICATIONS RELATED TO THIS

THESIS


Chapter 1

Introduction

To design a humanoid robot which can successfully demonstrate bipedal locomotion is a complex, hybrid control problem [1]. Most of the previous approaches use conventional feedback control based on inverse kinematics with some form of trajectory generation based on ZMP (Zero Moment Point). However, these are typically complex, require a lot of design effort and even then usually end up producing un-natural walking gaits and high energy consumption. But in real life humans learn to walk at a relatively early age. Aspects of balance are learnt using aids such as frames or furniture, then repetitive trials/steps are performed by infants and unsuccessful attempts/inefficient gaits are continuously improved. It is hypothesized that equipping a humanoid robot with such cognitive/learning abilities would produce robust walking gaits with a minimal amount of prior design as well as enabling the study of cognitive/learning algorithms in robots, the so-called embodiment principle [2-4]. There has been dire need to search of learning control algorithms for bipedal walking robot which do not require complex inverse kinematics and are closer to the human learning [5]. Iterative Learning Control (ILC) appears to be suitable to use for a bipedal walking robot due to its repetitive/batch learning
formulation and ILC techniques have previously been proposed for
bipedal robot locomotion, but no results demonstrated [6, 7].
ILC is a *Learning Control System* which evolved in 1970-80’s. It
improves the tracking performance of systems which operate in a
repetitive manner. These repetitions occur after fixed intervals of time.
ILC has helped to improve the performance of repetitive control
systems especially when dealing with uncertain systems [8].
This chapter presents an overview of the research, i.e., background
motivation, problem formulation, contributions, achievements and the
brief outline of the thesis.

1.1 Motivation

The human brain has the capability to acquire new knowledge or
modify existing knowledge on the basis of recent experience. This is
termed as cognitive learning [9-11]. In other words learning involves the
transformation of information in the environment into knowledge that
is stored in the mind. This has opened the door for the development of
*Learning Control Systems* which learn from their past experience, in a
manner similar to the human brain. The attractive features such as
learning capability, model-free control design, simple architecture, etc.,
have led to the work described in this thesis which applies and analyses
ILC algorithms to the problem of designing bipedal gaits in humanoid
robots.
The aim is to replicate the reflexive/muscle memory feature of humans which allows certain physical actions to become easier so that eventually the actions are performed without conscious thought. This learning process occurs through repetition of a particular action over and over again as walking, running, athletics and other physical games. Whilst, this repetition or practice of movement is carried out, new neural pathways within the brain are likely created that allow the action to be performed with less and less conscious effort on the part of a person. After sufficient time, the long-term muscle memory develops and future actions can be performed easily and accurately without conscious effort. The biological systems thus achieve maximum efficiency within the motor and memory systems.

Designing a bipedal walking robot is a complex, hybrid control problem due to non-linear effects such as stiction, the difficulty in estimating important parameters and the problem with modelling impacts, amongst others. Similarly, trajectories designed using ZMP (Zero Moment Point) often have an unnatural gait and high energy consumption. There is, therefore, a need to investigate how learning control algorithms can be applied to bipedal walking robots in order to reduce the design effort and also to inform ideas about cognitive learning [5]. In this research, simple ILC techniques have been successfully applied to compass gait model of the bipedal robot locomotion [12].
1.2 Problem Formulation

The learning control for bipedal robots can not be directly implemented as a conventional ILC problem due to non-zero initial state errors at the start of each step. Each step is considered as one batch. For continuous walking the error in the gait at last moment just before the swinging leg strikes the ground affects the initial state at the start of next step so the ILC horizon becomes infinite, unlike conventional Cyclic ILC which assumes a finite, horizon. Moreover, the fixed length assumption for each step can not be fulfilled in many cases such as when the foot impacts with ground occur earlier or later in consecutive steps due to variations in speed or step length. Moreover, the learning problem should not just be analysed in terms of its asymptotic behaviour. Humans learn to walk in hundreds or thousands of trials. Efficient learning, especially during the initial stages of learning is required, it is not enough to simply establish asymptotic convergence results. So, in this research, the rate of convergence of conventional ILC systems across the batch (i.e., for each time index) has been analysed. It has been further extended to non-zero initial condition Cyclic ILC so as to handle the impacting systems.

ILC is typically used for controlling stable systems or it may be incorporated along with a stabilizing controller which keeps the system with in the required stability region. The ILC iteratively improves the tracking performance and in the long run would be able to compensate for parameter variations in the plant. However, although establishing
convergence of the ILC algorithm is model-free, the rates of convergence depend on the system dynamics. Hence, to address the convergence rate problem, the general statement is given as follows:

“How to analyse the convergence rate of ILC generated control input up to component level and apply for the impacting systems having nonzero initial errors from previous iterations such as bipedal walking robots?”

The general problem stated above covers mathematical formulation, verification and simulation analysis of LTI and nonlinear impulsive hybrid dynamic systems such as bipedal walking robot.

1.3 Thesis Contributions and Achievements

The casting of walking problem in terms of Cyclic ILC required mathematical formulation which has been achieved by relaxing the conventional fixed length, zero initial error assumptions. For this the effect of initial state error on each of the time index inside a batch has to be formulated until the end so as to analyse how does it effect the initial state for the next consecutive iteration. Following is the brief about thesis contributions during this research:

- Development of a homogeneous two-dimensional input error evolution relationship at component level instead of conventional two dimensional models [13-15]. (Section 4.5)
- Derived the relationship for component-wise input error convergence rate and proved that long-term rate at all time
indices equals the eigenvalue of the matrix which relates the evolution from one iteration to the other in line with conventional asymptotic analysis. (Theorem 4.1)

- Extension of homogeneous two-dimensional input error evolution at component level and the convergence analysis for non-zero initial error ILC and impacting Cyclic ILC (Section 5.1).

- While analysing the simple D-ILC algorithm for impacting Cyclic ILC it was observed that initial error caused constant offset error which pointed towards existence of a dominant eigenvector associated with unit magnitude eigenvalue. It has been stated and proved as Theorem 5.1.

- The input error reduced to zero on adding proportional error term to D-ILC, i.e., using PD-ILC algorithm for Cyclic ILC. Hence stated and proved that for the input error evolution matrix that there does not exist eigenvector associated with unit magnitude eigenvalue (Theorem 5.2)

- Implementation of simple D-ILC algorithm to generate impulse-type torques for both the ankle and hip joints at the start of each step for compass gait robot to achieve stable walking gait of the robot (Section 6.1).

- Implementation of a modified PD-ILC algorithm under Cyclic ILC scheme to generate torques for both the ankle and hip joints at each sample time during the steps for stable biped walk (Sections 6.2 and 6.3).
1.4 Thesis Outline

This introductory chapter is followed by literature review of ILC in chapter 2. Its properties and features which distinguish it form conventional feedback (classical as well as modern) control systems have been elaborated. Various types of ILC update rules have been mentioned. Bipedal walking robot has been discussed including the efforts to use ILC algorithms to generate a stable walking gait.

In chapter 3, passive compass gait robot dynamics have been modelled for stable walking over a flat surface as well as on a slope. The passive dynamic compass gait robot has been chosen due to reasons of energy efficiency [16-21]. From the un-actuated passive walking downhill under the action of gravity considered as reference, the model for actuated compass gait has been achieved. Linearised version of the biped model has been presented along with analysis for uncertain parameters.

Convergence issues for the D-ILC algorithm have been covered in 4th chapter. A linear discrete-time system is considered with no initial state error. For control input update, D-ILC algorithm has been employed. Component-wise input error evolution from iteration to iteration has been formulated. Convergence analysis has been based on the operator matrix which relates the input error evolution.

In chapter 5, the case for non-zero initial states at the start of each repetition has been presented. The Cyclic ILC case where state errors are transferred from the end of one iteration to the start of next
iteration has been investigated. D-ILC has been unable to provide a satisfactory solution. It has been mathematically proven in Theorem 5.1. However, PD-ILC algorithm has achieved convergence for the variable initial states. Component-wise input error evolution for PD-ILC has been developed which has been implemented to generate the control inputs for damped pendulum.

In chapter 6, the ILC techniques have been applied to bi-pedal locomotion. The D-ILC algorithm has been applied to generate input torques for the hip and ankle joints at the start of each step for bipedal walk on a flat surface. These torques are applied at the start of each step. During the whole step, the robot legs move passively. This has led to a stable walking pattern learned by the compass gait robot while employing the minimum amount of torques. Secondly, using a reference trajectory from a fine-tuned feedback type PD controller, a modified Cyclic ILC algorithm for biped locomotion has been formulated to generate the set of control inputs for both the ankle and hip joints for stable gait of robot.

In chapter 7 the conclusions have been drawn about this research reflecting the overall summary of the thesis along with directions for future research.
Chapter 2
ILC and Bipedal Walking -
Literature Review

In this chapter, efforts to achieve stable walking gait for biped robot and the importance of learning for the robots have been elaborated. This has led to use of ILC algorithms to solve the biped walking problem. What is ILC, the difference from traditional feedback control and how has ILC evolved, the brief history of the ILC and the various commonly employed ILC update rules have been discussed. The Cyclic ILC and the corresponding research have been reviewed which has been employed in this research to solve the walking problem of the biped robot.

2.1 Bipedal Humanoid Robots

The interest in 2-legged (bipedal) humanoid robots as autonomous walking machines has been quite old [22]. The ability of walking over difficult uneven terrains, running, climbing up stairs and most important the ability for compliance in a human environment to work as servants or developing the prosthetic limbs for handicapped people are the main objectives. Recent trend focuses on the optimal
performance in which a robot can extract knowledge from the surroundings and learn to interact in an optimal manner.

Few notable bipedal humanoid robots are ASIMO (Advanced Step in Innovative MObility) developed by Honda, Japan. It has 34 DOFs with the ability to walk and run as well [23]. The Humanoid Research Group of Japan’s National Institute of Advanced Industrial Science and Technology (AIST) has come up with a series of human-sized robots as HRP-2, HRP-3, HRP-4C and HRP-4, the human co-operative robot. These robots have impressive capabilities of running, walking and even dancing [24]. WABIAN (WAseda BIpedal humANoid) and WABIAN-2R with 41 and 43 degrees of freedom (DOFs) respectively, have been developed by Biped Humanoid Robot Group at University of Waseda, Japan [25]. Humanoid Robot Research Centre at Korea Advanced Institute of Technology (KAIST) have developed HUBO (41 DOFs) and later HUBO2 (40 DOFs) which consumes lesser energy due to stretched-leg walking instead of traditional ZMP [26]. Under the European Commission funded RobotCub project, cognitive robots iCub and C-Cub with 53 DOFs have been developed to implement and explore the learning behaviour in humanoid walking robots similar to a human child of about 3 years of age. It is an open system platform licensed under Free Software Foundation GNU licences to allow its free use for research. First iCub prototype was developed at The Italian Institute of Technology (IIT) in Genoa, Italy [27]. The Humanoid
Robotics Research Group at The University of Manchester is also actively involved in the bipedal walking research [28].

2.2 Passive Dynamics for Walking Robots

Dynamics of the un-actuated passive walking robots were pioneered by Ted McGeer who showed that for gentle slopes, stable bipedal walking can be achieved where the necessary force is applied by the gravity [16]. Later on by many others for energy efficient steady walking patterns on slopes under the action of gravity only [29]. The passive dynamics have been extended for active energy inputs to create stable human-like walking [17, 30]. Limit Cycle Walking patterns called gaits have been developed which ensure stability of the passive walking [31]. Effects such as bifurcations and chaos were investigated which occur because of the parameter variations [24, 25]. For compass-like biped robot, passivity mimicking control laws were formulated for the hip and ankle joints to obtain a robust and efficient walking pattern [32]. Foot placement and velocity control of the biped robot have been further employed to eliminate problem of walking on rough terrains [33]. ILC has been employed to generate the stable walking gait for the bipedal humanoid robot using variational symmetry [34]. Instead of decentralized PID controllers, a centralized LQR multivariable controller has been successfully implemented for the stable and robust humanoid walking using iterative scheme [35]. The research is still under way to develop a biped robot with capability of learning to walk.
properly having same Degrees of Freedom (DOF), weight and length as a human. Research also focuses on the locomotion of the biped robots on unseen terrains along with energy efficiency during walking.

2.3 Importance of Learning for Bipedal Walking Robots

There has been a lot of progress in terms of development of hardware for bipedal robots and advanced control techniques for walking. But still there are unsolved issues such as robustness, energy efficiency, complete autonomy, safety and user-friendliness for the bipedal robots. Bipedal robots cannot cope with large movements, variable speeds and many other constraints. Due to these discrepancies, the current bipedal robots lack the level of robustness, versatility and adaptability that biological systems have and use them for efficient walking.

To address the stability and periodic walking issues, centre of pressure (COP) and zero moment point (ZMP) concepts have been previously implemented. However, biped robot needs to act like a human, so that they must be capable of learning new gaits in the case of moving in unknown terrains. Therefore, these force control techniques should be combined with learning strategies, because learning ability allows the bipedal robot to modify its dynamic walking pattern to the changing conditions that is necessary for autonomous walking. The control techniques for bipedal robot have been based on kinematic and dynamic modelling of the mechanism which requires complete state
measurement and interaction between feet and ground. This requires heavy computation and optimization. Intelligent control techniques are capable to overcome these constraints [29, 30]. Cognitive Robotics includes the representation of experience and reasoning problems handled by an autonomous robot in a dynamic and incompletely known environment [36].

2.4 What is ILC

There are many real life applications where same sequence of actions is repeated again and again such as assembly lines, robot manipulators, rolling mills, chemical batch processes, etc. The conventional feedback controllers cannot utilize the information from successive repetitions. Learning Control techniques such as ILC incorporate the intelligence to modify the performance on the basis of previous repetition. Iterative Learning Controllers provide an adaptive solution. The controller utilizes the error information of each batch and updates the control input accordingly for the next batch or batches. The control input signal using ILC converges to the desired value of the control input which is the inverse solution. Owing to this property, it is also termed as Iteration Inversion Process [37]. The error information at each time index can be used in a variety of ways to generate the update for control input. Hence, there are D, PD, PI, PID, etc many types of ILC controllers in which the output error, its derivative, integral or some combination of these is added to the current control input to generate
input for the same time instant in next batch. Learning occurs through pre-determined repetitions at hardware level [8]. These repetitions provide experience to the mental level [38, 39]. The experience is stored as data [34].

2.5 History of ILC

ILC evolved in 1970-80s, as one of the learning control systems. However, earliest use of the term learning control was reported in a 1967 US patent No. 3,555,252 by Murray Garden [40] as presented and compared with conventional ILC in [41]. The contributions of Cryer [42], Uchiyama [43] and Arimoto [34] are considered to be the initial works of ILC [44]. ILC improves the transient performance of systems which operate in a repetitive manner. These repetitions occur after fixed intervals of time. ILC has helped to achieve better performance of control systems especially when dealing with uncertain/stochastic systems [8]. The concept of learning through repeated trials evolved for improving the motion control of mechanical arms [34, 45]. The D-ILC algorithm based on the derivative of the output error for linear time-varying systems with application to robotic manipulators were developed [34, 46]. However, the D-ILC suffered with the problem of differentiation of high frequency noise. Later, the P-ILC improved upon by using only the error instead of its derivative [47]. The Current Iteration Tracking Error (CITE) was then introduced to formulate ILC in line with feedback control paradigm and helped to overcome large
overshoot thus convergence could be accelerated [48]. The discrete-time version of D-ILC was formulated for MIMO linear systems which possessed global robustness against state disturbances, measurement noise and re-initialisation error at the beginning of each iteration [49]. The ILC law has been formulated for non-linear time varying systems having affine input and linear output. Uniform convergence of input and state was achieved when there were no disturbances [8]. Robot manipulators have replaced the humans at many places in industrial environment such as manufacturing, assembly lines, tooling, palleting, machining and painting etc. The use of ILC has improved the motion control in time-invariant robot manipulators [24, 25]. For such robotics applications, D and PD type ILC were initially proposed and gave successful results [50]. But these could not be directly applied to bipedal walking due to zero-initial error constrain.

2.6 ILC for Bipedal Robots

The humans learn many physical actions by repetition such as walking at their early age. The experience is stored as muscle memory so that after few repetitions the humans are able to do these actions without conscious effort. It is achieved due to cognitive learning ability of the humans. So there are two important aspects that learning is achieved via a simple and repetitive process. Since, ILC has both the features hence it has been considered as a suitable candidate for learning the bipedal locomotion [6, 7].
Walking with two feet consists of swinging motion by one leg while the other rotates around the ankle but its foot remains fixed. Then, impact of swing leg occurs when it touches the ground associated with the change over to the other leg which again swings and impacts. Each cycle/repetition constitutes a step for the biped. Thus it is a special case for Cyclic ILC in which the state at start of new step/repetition is a function of the state at the end of the previous step/repetition.

Use of ILC to solve walking problem of the robot is relatively a new application area. ILC was initially employed to generate optimal passive gait trajectories for a one-legged hopping robot which required zero input for passive running [35]. ILC along with virtual constraints has produced optimal gait which resulted in constant speed walking patterns achieved for compass gait robot modelled as a Hamiltonian system [51]. Satoh et al. has used ILC to generate optimal gaits for one-legged hopping robot and extended to biped on the basis of variational symmetry of the Hamiltonian systems using virtual constraints [52, 53]. Iterative Feedback Tuning (IFT) has been used in conjunction with ILC using virtual constraint to generate stable gait for compass gait robot [54]. Zhang et al. have proposed impulsive toe-off push generated with ILC which is applied to the biped just before the heel strikes (ground impact of swing leg). However, there is no actuation during the entire swing phase and the swing leg moves passively [6, 55]. Keeping in view the impact based dynamics, a control strategy based on Receding Horizon Control was also suggested to stabilize the compass gait robot.
against initial states in [56]. But even here the swing phase has been left un-actuated.

However, the bipedal walk using actuation for both the ankle and hip-joint together has not been solved using ILC/RC techniques until now [16, 29, 57-60]. So, in this research, a mathematical frame work for the two-dimensional evolution of input and state errors for Cyclic ILC has been developed for the evolution of initial state errors for D- and PD-ILC on the basis of the rate of convergence analysis for zero initial errors [41-44]. This allowed casting the walking problem as a continuous process from one step to the other. The ground impacts could be modelled as well as the non-uniform step length has been accommodated. Further, simple PD-ILC algorithm modified with the acceleration error term has been used to generate input torques at ankle and hip joint for the steady gait generation of compass gait model of the biped. The hybrid dynamics of compass gait biped locomotion have been handled with the proposed scheme [61]. Compass gait model for biped robot based on Lagrangian dynamics has been used for the analysis in this thesis [9, 21, 23, 40, 41, 43].

2.7 Iterative Learning Control - Architecture

In ‘Iterative Learning Control’ Iterative points to the repetitive or recursive operation of the plant/system. Starting from identical initial conditions for each batch, it was observed that such systems exhibit similar errors in the output response. This motivated to record these
errors and modify/update the control input signal for the subsequent repetitions, i.e. learning from past experience [34]. Learning Systems can adapt and change their behaviour on the basis of input-output observations.

ILC can be defined as follows. “It’s a learning control technique where the controller learns to generate the desired set of control inputs over the iterations so as to minimize the tracking error between the output and the reference signal in a sequential manner” [38, 46, 62-65].

The Figure 2.1 below shows the basic ILC scheme in which current input \( u(i,k) \) is applied to the system to generate output \( y(i,k) \) at the \( i^{th} \) time index during the \( k^{th} \) batch. These values are stored in memory and used off-line to calculate values of control input \( u(i,k+1) \) for the next \( (k+1)^{th} \) iteration such that the desired output \( y^*(i) \) is tracked over the iterations.

![Figure 2.1: Basic ILC scheme](image-url)
ILC differs from conventional feedback control where only the current error is used to correct or update the control input for the next time instant as shown by yellow block arrow on top row in Figure 2.2. Since there is no mechanism to utilize the results of one batch to improve the output of the other batches, the same errors are repeated over different iterations [38]. However, in case of ILC the tracking error information of each iteration is utilized to improve the output for next consecutive iteration/s as shown by blue block arrow in Figure 2.2.

![Feedback versus learning](image)

**Figure 2.2: Feedback versus learning**

The system response to control input at each instant of time within the batch is memorized and the error information is used to correct/update
the control input for the next iteration. This updated input minimizes
the performance error during the next iteration [39]. In ILC the
parameters of the controller are not changed so it can be distinguished
from Adaptive Control in which the parameters of the controller itself
are changed for performance improvement. Likewise, ILC is different
from Optimal Control since exact model parameters are not required to
be known [66].

2.8 ILC Notation and Assumptions

ILC has been established for continuous-time as well as discrete-time
systems. It is being utilized for both linear and non-linear systems
successfully. In this research ILC for discrete-time systems has been
considered.

2.8.1 Assumptions

Following assumptions usually hold for conventional zero-initial error
ILC algorithms [67-70].

• Initial conditions are reset at the start of each batch.

• The error converges after every iteration. Although there may be
  uncertainties/un-modelled dynamics, i.e., minimal knowledge
  about the exact system parameters.

• The length of a batch is fixed.

The ILC learns to produce the best possible control signal without
changing its own configuration or parameters. This distinguishes it
from Adaptive Controller. The control signal update for the next iteration is computed off-line, i.e., at the end of the current iteration. However, in this research some of the assumptions have been relaxed to accommodate the walking problem of the bipedal robot such as zero initial error. It is required because the initial state error for each iteration/step is inherited from the end of the previous iteration/step. It thus represents a specific class of non-zero initial error ILC called Cyclic ILC. It still needs further enhancement because ground impacts introduce a specific relation for the state transformation. The next assumption relaxed is the batch size which may not remain same for each iteration/step due to variations in speed or step size.

2.8.2 Discrete-time System and Input-output Relationship

The discrete-time state space description of the system is used with two indices $i \in [1,M]$ and $k$ which stand for the time index inside a batch and the batch number respectively as follows.

\[
x(i + 1, k) = A x(i, k) + B u(i, k) \\
y(i, k) = C x(i, k)
\]

where state vector is $x \in \mathbb{R}^n$, input $u(i, k) \in \mathbb{R}^p$ and output of the system is $y(i, k) \in \mathbb{R}^m$. $A$, $B$ and $C$ are the real-valued state, input and output matrices respectively having appropriate dimensions. $u(i,1)$ is the control input vector for first batch which may be externally
specified or left to be zero \([38, 71]\). The input-output relation for a LTI system is described in Eq. (2.3) below.

\[
y(k) = H u(k)
\]  

(2.3)

where the vectors \(y(k)\), \(y^*\), \(u(k)\) and \(e(k)\) are the actual output, desired output, input and errors respectively, given as follows.

\[
y(k) = \left[ y(1,k), y(3,k), \ldots, y(M,k) \right]^T
\]  

(2.4)

\[
y^* = \left[ y^*(1), y^*(2), \ldots, y^*(M) \right]^T
\]  

(2.5)

\[
u(k) = \left[ u(1,k), u(2,k), \ldots, u(M,k) \right]^T
\]  

(2.6)

\[
e(k) = y^* - y(k) = \left[ e(1,k), e(2,k), \ldots, e(M,k) \right]^T
\]  

(2.7)

Matrix \(H\) in lifted form has elements which are impulse response coefficients or the Markov parameters of the plant \(G(z)\) in Eq. (2.8).

\[
H = \begin{bmatrix}
  h_r & 0 & 0 & \ldots & 0 \\
  h_{r+1} & h_r & 0 & \ldots & 0 \\
  h_{r+2} & h_{r+1} & h_r & \ldots & 0 \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  h_{r+M-1} & h_{r+M-2} & h_{r+M-3} & \ldots & h_r
\end{bmatrix}
\]  

(2.8)

The Markov parameters are generally given by \(h_i = CA^{i+1}B\).

### 2.9 ILC Update Rules

The researchers have introduced many variants of ILC. The more common have been briefly discussed below.
2.9.1 D-ILC

The generic form of ILC is D-ILC. It updates the control input for next iteration using the derivative of the output error of the current iteration [8]. For a deterministic system, which is initialised after each iteration, the D-ILC update mechanism is given in Eq. (2.9) below.

\[ u(i, k + 1) = u(i, k) + K_d \dot{e}(i, k) \tag{2.9} \]

where \( \dot{e}(i, k) = \dot{y}^*(i) - \dot{y}(i, k) \) is the derivative of the output or state error and gain matrix \( K_d \) is the learning gain parameter. For identical initial conditions, the initial error is zero at the start of each iteration, i.e., \( e(1, k) = 0 \) or \( y(1, k) = y^*(1) \). The asymptotic convergence of output \( \lim_{k \to \infty} y(i, k) = y^*(i) \) is achieved for all time indices \( i \in [0, T] \) under the following condition in Eq. (2.10).

\[ \max |\text{eig} (I - K_d CB)| < 1 \tag{2.10} \]

For monotonic convergence, the norm condition is specified as under [67].

\[ \|I - K_d CB\|_p < 1 \tag{2.11} \]

where Eq. (2.11) is the \( p^{th} \) operator norm, such that \( p \in \{1, 2, ..., \infty\} \). If \( p \) is not mentioned, it is assume that it is 2-norm. The product \( CB \neq 0 \) is \( h_1 \), the 1st Markov parameter in the matrix of Markov parameters as given in Eq. (2.8). Since, the above condition is independent of the system matrix \( A \), it points towards the ability of ILC to generate
desired control input even when the plant parameters are not fully known [72-75].

D-ILC has the capability to capture the trend similar to derivative part in conventional PD or PID controller design. The negative factor about D-ILC is that it requires derivative of the output error which may not be measurable or obtained through numerical differentiation. It may contain noise which can degrade effectiveness and accuracy. This is usually avoided by using forward difference error as an alternative to the derivative of the output error.

### 2.9.2 P-ILC

In P-ILC, the input is updated using the error itself which is multiplied with learning gain $K_p$. Hence, its simpler as compared to D-ILC as there are no derivative or integral components [76].

\[
u(i,k+1) = u(i,k) + K_p e(i,k)
\] (2.12)

However, the use of P-ILC algorithm ensures convergence only if the uncertainties or disturbances are absent. Robustness against uncertainties is achieved when scalar forgetting factor $\gamma$ is used. Then P-ILC is modified as given in Eq. (2.13) [72].

\[
u(i,k+1) = (1 - \gamma) u(i,k) + \gamma u(i,0) + K_p e(i,k)
\] (2.13)

The forgetting factor $\gamma$ may be either fixed or variable from iteration to iteration. The bias term $\gamma u(i,0)$ constrains so that input generated for next iteration does not vary extra-ordinarily. Thus, use of forgetting factor
reduces variance of the output in the early iterations [67]. Another option has been to use both positive and negative learning gains. It has achieved learning of more frequency components as compared to conventional P-ILC [8, 38]. However, still some gaps in frequency could not be learned by this scheme. It has been overcome by using anticipatory ILC discussed later in Section 2.9.6.

2.9.3 PD-ILC

The PD-ILC update has both the proportional and derivative of the error term as given in Eq. (2.14) below.

\[ u(i,k + 1) = u(i,k) + K_d \dot{e}(i,k) + K_p e(i,k) \]  

(2.14)

Use of proportional (P) and derivative (D) components together in ILC helps to achieve better convergence of tracking errors [77]. In this work it has been observed to acquire faster rate of convergence as compared to D-ILC and given in detail in chapter 4. The convergence achieved for the Cyclic PD-ILC case has been given in chapter 5.

2.9.4 PI-ILC

PI-ILC has input update based on proportional and integral components of the error as given below.

\[ u(i,k + 1) = u(i,k) + K_p e(i,k) + K_i \int e(i,k) dt \]  

(2.15)

The added advantage is that PI-ILC achieves monotonic convergence for discrete-time LTI systems in the sense of any norm besides the
exponentially weighted sup-norm. As compared to P-ILC, better performance of the PI-ILC has been shown in terms of convergence rate. However, if the time instants are large for each batch, then the integral component of PI-ILC does not give any significant advantage [72].

2.9.5 PID-ILC

In line with PID algorithm for conventional feedback, the PID-ILC has been formulated as shown below [78].

\[
u(i, k + 1) = u(i, k) + K_p e(i, k) + K_d \dot{e}(i, k) + K_i \int e(i, k) di
\] (2.16)

PID-ILC update algorithm uses the error information \(e(i, k)\) from the previous iteration and possesses linear effect of the past input \(u(i, k)\) as well [79].

2.9.6 Anticipatory ILC

The Anticipatory ILC avoids the use of derivative \(\dot{y}_k(t)\) to capture the trend. Usually, the time delayed version of the derivative, i.e. \(y(i + \Delta, k)\) is used as given below [80].

\[
u(i, k + 1) = u(i, k) + L\{y^*(i + \Delta) - y(i + \Delta, k)\}
\] (2.17)

Use of time-delayed version of derivative ensures that causal pair of input and output \([u(i, k), y(i + \Delta, k)]\) from \(k^{th}\) iteration are utilised for the next iteration.
2.9.7 Optimal ILC

For using ILC in an optimized manner, control input is updated so as to minimize a specific cost function. Different cost functions have been formulated, such as the controller which minimizes a weighted cost function as given in Eq. (2.18) [44, 81].

\[ J_{k+1,M}(u(i, k + 1)) = \sum_{i=1}^{M} \lambda^{i-1} \left\{ \| e(i, k + 1) \|^2 + \| u(i + 1, k) - u(i, k + 1) \|^2 \right\} \]  \hspace{1cm} (2.18)

where \( \lambda \) is the weight selected such as to minimize the cost function \( J \) for each successive iteration.

2.9.8 Cyclic ILC and Use for Robotics

In ILC literature, the initial states which are the final states of the last repetition have been termed as “Cyclic learning control” [79], “No-reset ILC (NRILC)” [82], ‘continuous’ ILC [81] or “alignment condition” [83, 84]. In case of both the reference and the output trajectory following the “alignment condition”, robust ILC algorithm based on the inverse dynamics of robotic manipulator was employed to achieve convergence [83]. Conditions for monotonic convergence were further formulated using PD-ILC with selective learning under restrictive assumptions for non-linear systems [85] and systems with non-parametric uncertainties [86]. The bipedal walking has been posed as impacting Cyclic ILC scenario where each step constitutes an iteration whose
state at the last moment does effect the initial state of the next consecutive step/iteration.

2.9.8.1 Cyclic D-ILC for CG walking robot

It has been argued that motions such as walking, running etc are perfected by humans using simple repetitive learning rules and do not require deep conscious involvement. Keeping in view the biological plausibility, the cognitive robotics need to be developed around simple learning rules [36, 87-90]. The Cyclic D-ILC algorithm has been used for car-like mobile robots by transforming the systems in chained form via feedback. However, the techniques were not further implemented to solve the walking problem [91]. For non-zero initial errors, the initial state learning has been an active research area [84, 92-96]. Using only the final state to generate initial state to be used as necessary control action has been proposed [97]. In this research we have employed initial state learning in Cyclic ILC scenario for a CG (compass gait) robot for which the ankle and hip actuation has been generated using Cyclic D-ILC. During the swing phase the robot is un-actuated and moves under the action of gravity only. With carefully selected initial states for the 1st step, stable walking gait could be realized as illustrated by phase portraits in chapter 6.

2.9.8.2 Modified Cyclic PD-ILC for CG walking

The length of iteration or time period of a step cannot be fixed due to variation in step size and/or speed. For this the non-standard approach
of non-uniform repetition length as proposed in conjunction with Cyclic PD-ILC to track periodic signals in non-linear systems has been utilized in which shorter of two consecutive step lengths is considered and if the ground impacts occurs earlier so that last time index is missed, then it is calculated from the dynamics so that error signal could be generated from it [98]. The standard Cyclic PD-ILC could not give successful results for the CG walking. But using a modified Cyclic PD-ILC which allowed using the acceleration term has been shown to achieve the stable gait for CG walking after few hundred steps as demonstrated in chapter 6.

### 2.10 Summary

In this chapter, a review of the bipedal locomotion has been presented with emphasis on the learning techniques with a discussion of the research using ILC schemes. The common ILC schemes have been presented. The conventional zero-initial error ILC techniques and the related research have been briefly summarized. Further, the efforts to relax the zero-initial error condition have been discussed. These were later extended to the Cyclic ILC scheme in which the state errors from the end of iteration are transferred to the beginning of the next iteration as required in the legged locomotion of the bipedal robot.
In this chapter, the compass gait robot dynamics for (un-actuated) passive and actuated modes have been modelled using Lagrangian dynamics. In particular, a linearised dynamic model is formulated which will be central to the Cyclic ILC convergence presented in chapter 6. While walking over a flat surface, at every heel strike, the angular momentum of the robot is conserved but the energy is lost. Hence, some actuation is required to keep it in walking mode. Ted McGeer had proposed un-actuated bipedal robot walking downhill which works under the action of gravity only [16]. These passive dynamic walkers have been further augmented with joint actuation to achieve stable walking on flat surface [17, 59]. Humans learn to walk with a stable gait from an early age of 12 to 18 months. The simple and repetitive nature of walking makes it a suitable candidate for realization with ILC schemes [6, 7, 35].

3.1 Two-Legged Human Locomotion

Human walking is a complex bipedal locomotion since it lacks static equilibrium of the moving body. It’s an extensively studied research
area. There are various phases in walking such as the nonlinear swing phase during single support, the impact with ground which resets the state, the double support phase in which the two feet are switched and the transition to single support phase where an “impulsive toe-off” occurs. Therefore, walking is modelled as a hybrid system for which the control system has to be designed so that desired dynamic performance i.e., stable walking is achieved along with safety and reliability. A passive compass gait robot provides the simplest model to study the biped walking dynamics as discussed in the next section. The periodic motion of the robot link mechanism and its interactions with the ground produces a displacement on the ground which is called biped walking gait. This is also known as “limit cycle walking” [31, 54].

3.1.1 Compass Gait Model for Biped Robot

A compass gait robot is a simple model in sagittal plane which can be used to study the nonlinear dynamics of the biped locomotion. The two legs of the compass gait have no knees or feet and are connected with each other by a friction-less hip joint. It has been named due to its similarity to a pair of compasses. It is a rigid body system having two links only, as shown in Figure 3.1. The whole robot rotates around the fixed reference point \( p_0 \) which acts as pointed “stance or support foot”. The “swing leg” is the free leg which moves forward (left to right) and is swapped with the stance leg on impact with the ground [16, 21, 58].
The joint angles $q_1$ and $q_2$ are measured anti-clockwise. $q_1$ is the angle between vertical axis and the stance leg. The inter-leg angle $q_2$ for swing leg is measured with reference to the stance leg. Legs are assumed symmetric and mass-less except that each leg has equal point mass $m$. Each leg has equal length $l = a + b$. The hip mass $m_h$ summarizes the upper body mass.

### 3.1.2 Assumptions for Compass Gait Model

Following assumptions have been used in modelling the compass gait robot [19, 58, 99-101]

- No knees so the legs are straight rigid links.
- Retractable lower legs to prevent foot scuffing.
- Hip mass represents the upper body.
• Feet are pointed having no mass.
• Pointed feet do not slip.
• Friction-less joints.
• Walking is only in two-dimensional plane, i.e. no swaying.
• Both feet touch the ground instantaneously during double support phase.
• Motor dynamics are ignored for actuated walking.

The joint angles and respective velocities constitute the state of the compass gait robot. Hence, the state vector $x$ is defined as follows.

$$x = \begin{bmatrix} \theta, \dot{\theta} \end{bmatrix}^T$$  \hspace{1cm} (3.1)

where, vectors $\theta = [\theta_1, \theta_2]^T$ and $\dot{\theta} = [\dot{\theta}_1, \dot{\theta}_2]^T$ represent the joint angles and the angular velocities respectively.

### 3.2 Dynamics of Compass Gait Robot

When the robot is in contact with the ground with only one foot, it is said to have single support and the other leg swings freely hence called the swing phase. During the swing phase, the dynamics are similar to the 2nd order non-linear differential of an inverted pendulum coupled with a normal pendulum. The equations of motion for the swing phase can be obtained using Euler-Lagrange approach. The Lagrangian ($L$) is defined as the difference between the Kinetic ($KE$) and Potential ($PE$) energies of the robot.

$$L(q, \dot{q}) = KE(q, \dot{q}) - PE(q)$$  \hspace{1cm} (3.2)
The equations of motion for the joint angles are obtained from the Lagrangian given in Eq.(3.2) as follows.

$$\frac{d}{dt}\left( \frac{\partial L(q, \dot{q})}{\partial \dot{q}} \right) - \frac{\partial L(q, \dot{q})}{\partial q} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \tau$$  \hspace{1cm} (3.3)

where $\tau$ represents the vector of applied torques. The ankle joint torque $u_1$ causes the stance leg to rotate around it so that robot moves forward. The hip torque $u_2$ rotates the swing leg anti-clockwise to move it forward. Using Eq.(3.2), the Lagrangian ($L$) is substituted with $KE$ and $PE$ in Eq.(3.3), so that

$$\frac{d}{dt}\left( \frac{\partial KE(q, \dot{q})}{\partial \dot{q}} \right) - \frac{\partial KE(q, \dot{q})}{\partial q} + \frac{dPE(q)}{dq} = \tau$$  \hspace{1cm} (3.4)

The $PE$ for the compass gait can be easily calculated using trigonometry to give

$$PE(q) = \left( m(a + l) + m_h l \right) g \cos(q_1) - mg b \cos(q_1 + q_2)$$  \hspace{1cm} (3.5)

The instantaneous $KE$s of the individual point masses are added to give the total $KE$ as follows.

$$KE = \frac{1}{2} m \| \dot{\mathbf{p}}_1 \|^2 + \frac{1}{2} m_h \| \dot{\mathbf{p}}_2 \|^2 + \frac{1}{2} m \| \dot{\mathbf{p}}_3 \|^2$$  \hspace{1cm} (3.6)

The velocities for each point mass are given by

$$\dot{\mathbf{p}}_1 = -a \dot{q}_1 \begin{bmatrix} \cos(q_1) \\ \sin(q_1) \end{bmatrix}$$ \hspace{1cm} (3.7)

$$\dot{\mathbf{p}}_2 = -l \dot{q}_1 \begin{bmatrix} \cos(q_1) \\ \sin(q_1) \end{bmatrix}$$ \hspace{1cm} (3.8)
\[ \dot{q}_3 = -l \begin{bmatrix} \cos(q_1) \\ \sin(q_1) \end{bmatrix} + \frac{1}{2} b (\dot{q}_1 + \dot{q}_2) \begin{bmatrix} \cos(q_1 + q_2) \\ \sin(q_1 + q_2) \end{bmatrix} \] (3.9)

Substituting the point mass velocities in Eq. (3.6) and after simplification using trigonometric identities gives the total KE for the compass gait robot.

\[
KE = \left[ \frac{1}{2} m_h l^2 + \frac{1}{2} m \left( l^2 + a^2 + b^2 - 2bl \cos(q_2) \right) \right] \dot{q}_i^2 \\
+ \frac{1}{2} m b^2 \dot{q}_2^2 + \frac{1}{2} m \left( b^2 - bl \cos(q_2) \right) 2\dot{q}_i \dot{q}_2
\] (3.10)

The relation in Eq.(3.10) can be converted to the familiar form of KE using the velocity vector \( \dot{q} \) as

\[
KE(q, \dot{q}) = \frac{1}{2} \dot{q}^T M(q) \dot{q}
\] (3.11)

where

\[
M(q) = \begin{bmatrix}
{m_h l^2 + m \left( l^2 + a^2 + b^2 - 2bl \cos(q_2) \right) \quad m \left( b^2 - bl \cos(q_2) \right)}
\end{bmatrix}
\] (3.12)

\( M(q) \) is known as the inertia matrix which is independent of stance leg angle \( q_1 \) and only a function of the masses, lengths and inter-leg angle \( q_2 \).

Substituting the derivatives from Appendix A in Euler-Lagrange Eq.(3.4) and further simplification, the following 2nd order, nonlinear equation describes the dynamics of the compass gait robot.
\( M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau \)  

(3.13)

where, \( C(q, \dot{q}) \) is the centripetal/coriolis matrix

\[
C(q, \dot{q}) = \frac{dM(q)}{dt} \dot{q} - \frac{\partial KE(q, \dot{q})}{\partial q} \\
= \begin{bmatrix}
2mbl\dot{q}_2 \sin(q_2) & mbl\dot{q}_2 \sin(q_2) \\
-mbl\dot{q}_1 \sin(q_2) & 0
\end{bmatrix}
\]  

(3.14)

and \( G(q) \) represents the gravity vector as follows.

\[
G(q) = \frac{dPE(q)}{dq} \\
= \begin{bmatrix}
-(m(a + l) + m_h l) g \sin(q_1) + mgb \sin(q_1 + q_2) \\
-mgb \sin(q_1 + q_2)
\end{bmatrix}
\]  

(3.15)

The swing phase continues until the swing leg strikes the ground and robot enters the double support phase followed by another swing action for the previously stance leg [54].

### 3.2.1 Swing Phase Dynamics in State Space Form

To simulate the 2\(^{nd}\) order nonlinear equation of motion for the swing phase in Eq.(3.13), it has to be represented in general nonlinear form as

\[
\dot{x} = f(x, u)
\]  

(3.16)

The state vector \( x \) is 4-dimensional and given as

\[
x = \begin{bmatrix}
q \\
\dot{q}
\end{bmatrix} = [q_1, q_2, \dot{q}_1, \dot{q}_2] \quad \vec{r}
\]  

(3.17)

Hence, the nonlinear dynamics of the actuated compass gait robot during swing phase in state-space form is given by
\[ \dot{x} = \begin{bmatrix} \dot{q} \\ (M(q))^{-1} \begin{bmatrix} \tau - C(q, \dot{q}) \dot{q} - G(q) \end{bmatrix} \end{bmatrix} \]  
\hspace{1cm} (3.18)

The output is given by output equation as follows.

\[ y = Cx = I \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \]  
\hspace{1cm} (3.19)

In this research, the Eq.(3.18) has been implemented using ODE solvers from MATLAB®.

### 3.2.2 Impact Reset During Stance Phase

For an appropriate initial condition, the compass gait robot moves forward such that the stance leg rotates around the stance foot and the other free leg swings left to right until it touches the ground surface. In the double support phase both feet are on the ground. This is modelled as an instantaneous event and swapping of feet occurs, i.e., the previously swinging leg becomes stance leg as new reference point and vice versa as shown in Figure 3.2. The coordinates are re-labelled at impact using + and − superscripts to represent pre- and post- impact conditions, respectively as

\[ q_1^+ = -q_1^- \text{ and } q_2^+ = -q_2^- \]  
\hspace{1cm} (3.20)

Eq.(3.20) gives the relationship for both the transformed pre- and post-impact joint.
The transformed angles in vector-matrix notation are as follows

$$ q^+ = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} q^- $$  \hspace{1cm} (3.21)

The geometry at the impact has determined that the inter-leg angle is twice the stance leg angle as below

$$ q_2 = -2q_1 $$  \hspace{1cm} (3.22)

However, the joint velocities have changed abruptly at impact. The KE is decreased due to energy loss at impact. Assuming a perfect impact, the angular momentum remains constant as under

$$ M^-(q) \dot{q}^- = M^+(q) \dot{q}^+ $$  \hspace{1cm} (3.23)

The pre- and post-impact inertia matrices are calculated in Appendix A to find the transformed joint velocities at impact as follows

$$ \dot{q}^+ = R \dot{q}^- $$  \hspace{1cm} (3.24)
where reset matrix $R$ describes the switch over at impact for the angular velocities. Hence, the state vector update at impact after combining Eq.(3.21) and Eq.(3.24) is given below

$$
\mathbf{x}^+ = \begin{bmatrix}
-I & 0 \\
0 & R
\end{bmatrix} \mathbf{x}^-
$$

(3.25)

Since the derivation of angular momentum for even a simple compass gait model is quite complex, the models with higher DOFs require symbolic tools to simplify the process.

### 3.3 Simulations of Passive Compass Gait Robot

A passive compass gait robot has no input torques applied upon it. Hence, it represents the unforced homogeneous model of the robot which can be studied to analyse the natural response as under

$$
M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = 0
$$

(3.26)

The natural unforced dynamics of compass gait have been analysed with suitable initial states walking over a flat surface.

#### 3.3.1 Simulations for Passive CG Walking Over Flat Surface

The passive compass gait robot is simulated with parameters given in Table 3-1 from the initial state vector $\mathbf{x}_0 = [0.2, -0.4, -0.8, 2.1]^T$. 
Table 3-1 Compass gait robot parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leg mass ((m))</td>
<td>5</td>
<td>kg</td>
</tr>
<tr>
<td>Hip mass ((m_h))</td>
<td>10</td>
<td>kg</td>
</tr>
<tr>
<td>Half leg lengths ((a = b))</td>
<td>0.5</td>
<td>m</td>
</tr>
</tbody>
</table>

A stick diagram shows multiple snapshots for ten steps of the passive compass gait robot walking on a flat surface in Figure 3.3. Stance and swing legs are represented by solid and the dotted lines, respectively.

![Figure 3.3 Stick diagram of CG robot walking on flat surface](image)

The corresponding phase portrait in Figure 3.4 is used to show the relationship between joint angles and respective angular velocities of a leg during all phases of walking, such as swing phase (I-II) to impact...
reset (II-III), then stance phase (III-IV) followed by impact for the other swing leg (IV-I) leading to swing phase to start the next step.

The contraction in limit cycles shows that the un-actuated passive compass gait robot is losing energy [31, 57]. The steps of compass gait robot are getting smaller and smaller along with decreasing angular velocities until it comes to a stand still or either falls down. If torques are applied to the joints of the robot, it can continue to move forward and may achieve stable walking gait as discussed in next section.

**3.4 Compass Gait Robot on Slope**

When a passive compass gait walks down a slope having angle $\gamma$ with respect to the horizontal axis, as shown in Figure 3.5, gravity acts upon
it and can provide the necessary actuation to keep it walking in a stable fashion.

![Diagram of compass gait robot walking down a slope](image)

Figure 3.5 Compass gait robot walking down a slope

Using suitable initial values of the joint angles and angular velocities, sustained periodic walking patterns/gaits can be obtained for slopes having suitable shallow gradients, for example, up to 5 degrees [57, 102].

### 3.4.1 Equations of Motion for Down-hill Walking CG

The $KE$ in Eq.(3.11) of the compass gait robot is not affected by downhill walking. Only the $PE$ in Eq.(3.5) changes depending on the slope $\gamma$ as shown below

$$PE = \left( m(a + l) + m_bl \right) g \cos(q_1 + \gamma) - mgb \cos(q_1 + q_2 + \gamma)$$  \hspace{0.5cm} (3.27)

This change in $PE$ only changes the gravity vector $G(q)$ in Eq.(3.27) as
\[
G(q) = \begin{bmatrix}
-m(a + l) g \sin(q_1 + \gamma) + mgb \sin(q_1 + q_2 + \gamma) \\
mg \sin(q_1 + q_2 + \gamma)
\end{bmatrix}
\] (3.28)

The other two matrices \(M(q)\) and \(C(q,\dot{q})\) remain same as for the compass gait on flat surface in Eq.(3.12) and Eq.(3.14), respectively. The gravity acts as external torque acting on the joints to keep the compass gait robot to continue walking downhill.

### 3.4.2 Simulations of Passive CG Robot Walking on Slope

A passive compass gait robot is allowed to walk down a slope having gradient of -3 degrees under the force of gravity. The stick diagram in Figure 3.6 shows the symmetric walking pattern for compass gait robot walking down hill where the initial values for the state vector have been chosen as \(x_0 = [0.2, -0.4, -0.8, 2.1]^T\). The corresponding phase portrait of a selected leg in Figure 3.7 shows that down hill walker has achieved symmetric gait since it has converged to a stable limit cycle. Thus, the gravity has compensated for the loss of energy due to impacts. The compass gait under the action of gravity is sensitive to initial conditions as well as to external disturbances.
Figure 3.6 Stick diagram of CG robot walking down hill

Figure 3.7 Stable limit cycle for CG walking on slope
3.5 The Actuated Compass Gait Robot

To realize flexible walking on a flat surface, the compass gait robot needs to be actuated instead of just relying on gravity as discussed in the previous section. After adding the actuating torques at stance foot (ankle) and the hip joint, the two torque inputs are collectively called “hip and ankle joint actuation”. The ankle joint actuator provides rotational motion for the stance leg and hip joint actuator causes the free leg to swing forward [17, 20, 59].

In this research, the actuated compass gait robot have been implemented using D-ILC and modified PD-ILC algorithms to generate the input torques \( \tau \) for stable walking gait as given in chapter 6 [12].

3.6 Linearised Model of the Compass Gait Robot

The nonlinear equation for the compass gait robot can be linearised using a Taylors Series to obtain the linear equation of the unit leg length robot during swing phase [31, 57, 59]. The Eq. (3.13) is re-written as two separate unforced differential equations as under

\[
m_h \ddot{q}_1 + m \left( \frac{3}{2} \cos(q_2) \dot{q}_1 + \frac{m}{4} \left( 1 - 2 \cos(q_2) \right) \right) \ddot{q}_2 + m \sin(q_2) \dot{q}_1 \dot{q}_2 + \frac{m}{2} \sin(q_2) (\dot{q}_2)^2 - \frac{3}{2} m g \sin(q_1) + \frac{1}{2} m g \sin(q_1 + q_2) - m_h \sin(q_1) = 0
\]

\[
\frac{1}{4} m \left( 1 - 2 \cos(q_2) \right) \ddot{q}_1 + \frac{1}{4} m \ddot{q}_2 - \frac{1}{2} m \sin(q_2) (\dot{q}_1)^2 + \frac{1}{2} m g \sin(q_1 + q_2) = 0
\]
Re-arrange Eq. (3.29) as under

\[
\begin{bmatrix}
m_h + \frac{3}{2} m
\end{bmatrix} \ddot{q}_1 - m \cos(q_2) \ddot{q}_1 + \frac{1}{4} m \ddot{q}_2 - \frac{1}{2} m \cos(q_2) \ddot{q}_2 + m \sin(q_2) \ddot{q}_2 \ddot{q}_2
\]

\[
+ \frac{1}{2} m \sin(q_2) (\ddot{q}_2)^2 - \left( m_h + \frac{3}{2} m \right) g \sin(q_1) + \frac{1}{2} m_h g \sin(q_1 + q_2) = 0
\]

(3.31)

To find equilibrium points, set all derivatives to zero

\[
\left( m_h + \frac{3}{2} m \right) g \sin(q_1) + \frac{1}{2} m_h g \sin(q_1 + q_2) = 0
\]

(3.32)

For the above Eq. (3.32), to be valid, it is required that

\[
\sin(q_1) = 0 \quad \text{and} \quad \sin(q_1 + q_2) = 0
\]

(3.33)

The solution of Eq.(3.33) \( q_1 = q_2 = 0 \), provides suitable equilibrium points as \( q_1 = 0 \) and \( q_2 = 0 \).

Similarly, re-arranging Eq.(3.30), gives

\[
\frac{1}{4} m \ddot{q}_1 - \frac{1}{4} m \cos(q_2) \ddot{q}_1 + \frac{1}{4} m \ddot{q}_2 - \frac{1}{2} m \sin(q_2) (\ddot{q}_2)^2 + \frac{1}{2} m g \sin(q_1 + q_2) = 0
\]

(3.34)

To find the equilibrium points, put all derivatives to zero in Eq.(3.34)

\[
\frac{1}{2} m g \sin(q_1 + q_2) = 0
\]

(3.35)

For Eq.(3.35), to be valid \( q_1 + q_2 = 0 \), and again provides suitable equilibrium points as \( q_1 = 0 \) and \( q_2 = 0 \).
Hence, the suitable equilibrium condition occurs when the compass gait is in straight standing position with no movement. Hence, equilibrium occurs when joint angles are zero as under

\[ q_1 = q_2 = 0 \]  \hspace{1cm} (3.36)

and the respective angular velocities are zero as

\[ \dot{q}_1 = \dot{q}_2 = 0 \]  \hspace{1cm} (3.37)

Hence, the equilibrium state \( x_{eq} \) is given by

\[ x_{eq} = [q_{1eq}, q_{2eq}, \dot{q}_{1eq}, \dot{q}_{2eq}]^T = [0, 0, 0, 0]^T \]  \hspace{1cm} (3.38)

The first-order Taylor's series expansion for a function of several variables \( f(x_1, \ldots, x_n) \) around equilibrium points \( x_{1eq}, \ldots, x_{neq} \) is given as follows

\[ T(x_1, \ldots, x_n) = f(x_{1eq}, \ldots, x_{neq}) + \sum_{j=1}^{n} \frac{\partial f(x_{1eq}, \ldots, x_{neq})}{\partial x_j} (x_j - x_{jeq}) \]  \hspace{1cm} (3.39)

Applying the expansion in Eq. (3.39) to the Eq. (3.31) for biped robot to linearise around the chosen equilibrium point \( x_{eq} \) while using

\[ q_1 - q_{1eq} = \Delta q_1, \; q_2 - q_{2eq} = \Delta q_2, \; \dot{q}_1 - \dot{q}_{1eq} = \Delta \dot{q}_1, \; \dot{q}_2 - \dot{q}_{2eq} = \Delta \dot{q}_2, \]
\begin{align*}
\begin{pmatrix} m_h + \frac{3}{2} m \\ 0 \\
\end{pmatrix} \ddot{q}_1 + \frac{1}{4} m \ddot{q}_2 - \left( m_h + \frac{3}{2} m \right) g \left[ \sin(q_i) \right]_{k=0}^{h=0} + \frac{d}{dq_i} \left[ \sin(q_i) \right]_{k=0}^{h=0} \Delta q_i \\
- m \left[ \cos(q_2) \dot{q}_{k=0}^{h=0} + \frac{\partial}{\partial q_2} \left( \cos(q_2) \dot{q}_{k=0}^{h=0} \right) \right] \Delta q_2 + \frac{\partial}{\partial \dot{q}_2} \left( \cos(q_2) \dot{q}_{k=0}^{h=0} \right) \Delta \dot{q}_2 \\
- \frac{1}{2} m \left[ \cos(q_2) \dot{q}_{k=0}^{h=0} + \frac{\partial}{\partial q_2} \left( \cos(q_2) \dot{q}_{k=0}^{h=0} \right) \right] \Delta q_2 + \frac{\partial}{\partial \dot{q}_2} \left( \cos(q_2) \dot{q}_{k=0}^{h=0} \right) \Delta \dot{q}_2 \\
+ m \left[ \sin(q_2) \dot{q}_{k=0}^{h=0} + \frac{\partial}{\partial q_2} \left( \sin(q_2) \dot{q}_{k=0}^{h=0} \right) \right] \Delta \dot{q}_2 \\
+ \frac{1}{2} m \left[ \sin(q_2) (\dot{q}_2)^2 \right]_{k=0}^{h=0} + \frac{\partial}{\partial \dot{q}_2} \left( \sin(q_2) (\dot{q}_2)^2 \right) \Delta q_2 + \frac{\partial}{\partial \dot{q}_2} \left( \sin(q_2) (\dot{q}_2)^2 \right) \Delta \dot{q}_2 \\
+ \frac{1}{2} mg \left[ \sin(q_i) + q_i \right]_{k=0}^{h=0} + \frac{\partial}{\partial q_i} \left[ \sin(q_i) + q_i \right]_{k=0}^{h=0} \Delta q_i + \frac{\partial}{\partial q_i} \left[ \sin(q_i) + q_i \right]_{k=0}^{h=0} \Delta \dot{q}_i \\
= 0 \\
\end{align*}

(3.40)

On evaluating the differentials and putting the values

\begin{align*}
\begin{pmatrix} m_h + \frac{3}{2} m \\ 0 \\
\end{pmatrix} \ddot{q}_1 + \frac{1}{4} m \ddot{q}_2 - \left( m_h + \frac{3}{2} m \right) g \Delta q_i - m \left[ 0 - 0 + \Delta \dot{q}_1 \right] \\
- \frac{1}{2} m \left[ 0 + 0 + \Delta \ddot{q}_2 \right] + m \left[ 0 + 0 + 0 + 0 \right] + \frac{1}{2} m \left[ 0 + 0 + 0 \right] \\
+ \frac{1}{2} mg \left[ 0 + \Delta q_i + \Delta q_2 \right] = 0 \\
\end{align*}

(3.41)

Further simplification of Eq. (3.41) gives

\begin{align*}
\begin{pmatrix} m_h + \frac{1}{2} m \\ 0 \\
\end{pmatrix} \ddot{q}_1 - \frac{1}{4} m \Delta \ddot{q}_2 - \left( m_h + m \right) g \Delta q_i + \frac{1}{2} mg \Delta q_2 = 0 \\
\end{align*}

(3.42)
Similarly, Eq. (3.34) is linearised around equilibrium point as under

\[
\frac{1}{4} m \ddot{q}_i - \frac{1}{2} m \left[ \cos(q_z)(\ddot{q}_i) \right]_{q_i=0} + \frac{\partial}{\partial q_z} \left[ \cos(q_z)(\ddot{q}_i) \right]_{q_i=0} \Delta \ddot{q}_i
\]

\[
\frac{1}{4} m \ddot{q}_2 - \frac{1}{2} m \left[ \cos(q_z)(\ddot{q}_2) \right]_{q_i=0} + \frac{\partial}{\partial q_z} \left[ \cos(q_z)(\ddot{q}_2) \right]_{q_i=0} \Delta \ddot{q}_2
\]

\[
\frac{1}{2} \left[ \sin(q_1 + q_2) \right]_{q_i=0} + \frac{\partial}{\partial q_1} \left[ \sin(q_1 + q_2) \right]_{q_i=0} \Delta \ddot{q}_1
\]

\[
\frac{1}{2} \left[ \sin(q_1 + q_2) \right]_{q_i=0} + \frac{\partial}{\partial q_2} \left[ \sin(q_1 + q_2) \right]_{q_i=0} \Delta \ddot{q}_2
\]

\[
\frac{1}{4} m \Delta \ddot{q}_i - \frac{1}{2} m \left[ 0 + 0 + \Delta \ddot{q}_i \right] + \frac{1}{4} m \Delta \ddot{q}_2 - \frac{1}{2} m \left[ 0 + 0 + 0 \right]
\]

On evaluating the differentials and putting values

\[
\frac{1}{4} m \Delta \ddot{q}_i - \frac{1}{2} m \left[ 0 + 0 + \Delta \ddot{q}_i \right] + \frac{1}{4} m \Delta \ddot{q}_2 - \frac{1}{2} m \left[ 0 + 0 + 0 \right] + \frac{1}{2} mg \left[ 0 + \Delta q_1 + \Delta q_2 \right] = 0
\]

Further simplification of Eq. (3.44) gives

\[
-\frac{1}{4} m \Delta \ddot{q}_i + \frac{1}{4} m \Delta \ddot{q}_2 + \frac{1}{2} mg \Delta q_1 + \frac{1}{2} mg \Delta q_2 = 0
\]

The equations (3.42) and (3.45) can be written in matrix form as below

\[
\begin{bmatrix}
  m_h + m & -\frac{1}{4} m \\
  -\frac{1}{4} m & m_h + m
\end{bmatrix}
\begin{bmatrix}
  \Delta \ddot{q}_i \\
  \Delta \ddot{q}_2
\end{bmatrix}
+ \begin{bmatrix}
  -(m_h + m) & \frac{1}{2} mg \\
  \frac{1}{2} mg & \frac{1}{2} mg
\end{bmatrix}
\begin{bmatrix}
  \Delta q_1 \\
  \Delta q_2
\end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\]

The \Delta can be removed for simplicity, so Eq.(3.46) can be compactly written as
\[ M_0 \ddot{q} + G_0 \dot{q} = 0 \] (3.47)

where, the \( M_0 \) is the inertia matrix for the linearised model as

\[
M_0 = \begin{bmatrix}
m_b + \frac{m}{2} & -\frac{m}{4} \\
-\frac{m}{4} & \frac{m}{4}
\end{bmatrix}
\] (3.48)

and \( G_0 \) is the gravity matrix below

\[
G_0 = \begin{bmatrix}
-(m_b + m)g & \frac{mg}{2} \\
\frac{mg}{2} & \frac{mg}{2}
\end{bmatrix}
\] (3.49)

Similarly, the linearised equation of motion for actuated compass gait robot is given by

\[ M_0 \ddot{q} + G_0 \dot{q} = \tau \] (3.50)

The linearised state space description for the actuated compass gait robot from Eq. (3.50) can be formulated further as

\[
\begin{bmatrix}
\dot{q} \\
\dot{\dot{q}}
\end{bmatrix} =
\begin{bmatrix}
0_{2 \times 2} & I_{2 \times 2} \\
-M_0^{-1}G_0 & 0_{2 \times 2}
\end{bmatrix}
\begin{bmatrix}
q \\
\dot{q}
\end{bmatrix} +
\begin{bmatrix}
0_{2 \times 2} \\
M_0^{-1}
\end{bmatrix} \tau
\] (3.51)

The Eq. (3.51) can be written in standard form as under

\[ \dot{x} = Ax + Bu \] (3.52)

for the state vector \( x \) chosen as

\[
x = 
\begin{bmatrix}
q \\
\dot{q}
\end{bmatrix} = 
\begin{bmatrix}
q_1, q_2, \dot{q}_1, \dot{q}_2
\end{bmatrix}^T
\] (3.53)
and, the matrices \( A \) and \( B \) are given by

\[
A = \begin{bmatrix} 0_{2\times 2} & I_{2\times 2} \\ -M_0^{-1}G_0 & 0_{2\times 2} \end{bmatrix}
\]

\[ (3.54) \]

\[
B = \begin{bmatrix} 0_{2\times 2} \\ M_0^{-1} \end{bmatrix}
\]

\[ (3.55) \]

The linearised model needs to be a reasonable approximation of the nonlinear model. This can be achieved since the average values of joint angles are within \( \pm 0.3 \) radians which can be safely approximated as zero for linear model. The average values for the joint angular velocities lie within \( \pm 1 \) rad/s. The effect of the velocity needs further analysis for which the linear model is also simulated with same parameters as used for nonlinear compass gait robot.

![Figure 3.8 Comparison between nonlinear and linear models of CG](image)

The comparison given in Figure 3.8 shows that the performance of the linearised model is close to the nonlinear model. Hence it can be safely
assumed that for normal walking of compass gait the chosen linear model is a reasonable approximation of the nonlinear model and hence can be used for the ILC analysis.

3.7 Model Uncertainties in Compass Gait Robot

Following reasons cause the uncertainty in the mathematical model of the robot:
Linearization, point mass assumption, neglecting the hardware such as electrical wiring cables, the centres of gravity may also change from original ones and other simplifying assumptions.

3.7.1 Effects of Model Uncertainties

The output may not match with the real robot, phase portraits can also differ and stable walking gait may not be achieved over a real robot.

3.7.2 How to Analyse the Model Uncertainties

Variations in the model parameters Inertia matrix, Coriolis matrix and the gravity vector may be introduced by perturbations in masses, lengths and etc. The next subsection gives the uncertainty analysis for D-ILC which is an extended mathematical formulation of the earlier work from Danwei Wang [103].
3.7.3 Formulation to Analyse Effects of Uncertainties

Consider following discrete-time nonlinear dynamic system with the uncertainties and other disturbances lumped and denoted by \( w(i, k) \)

\[
x(i + 1, k) = f \left( x(i, k), i \right) + B \left( x(i, k), i \right) u(i, k) + w(i, k)
\]

(3.56)

and the output equation as

\[
y(i, k) = C(i)x(i, k) + v(i, k)
\]

(3.57)

where, \( k \) denotes the batch number having \( M \) number of samples in each trial, \( i \in [1, M] \) is the time index or sample number during each batch, state vector \( x \in \mathbb{R}^n \), input \( u(i, k) \in \mathbb{R}^r \) and output of the system is \( y(i, k) \in \mathbb{R}^p \).

The uncertainty \( w(i, k) \) is assumed bounded by \( b_w \) so that \( \| w(i, k) \| \leq b_w \) on the interval \([1, M]\). Similarly \( v(i, k) \) is bounded by \( b_v \) such that \( \| v(i, k) \| \leq b_v \).

The desired output \( y^*(k) \) with initial state \( x^*(1) \), is considered achievable, so there exists unique input \( u^*(k) \) and state \( x^*(k) \) which correspond to the desired output.

The functions \( f(x, i) \) and \( B(x, i) \) are globally uniformly Lipschitz in \( x \) on the interval \([1, M]\) such that \( \| f \left( x_1, i \right) - f \left( x_2, i \right) \| \leq c_f \| x_1 - x_2 \| \) and \( \| B \left( x_1, i \right) - B \left( x_2, i \right) \| \leq c_B \| x_1 - x_2 \| \) are valid for positive constants \( c_f \) and \( c_B \). The matrices \( B(x, i) \) and \( C(x, i) \) are bounded as \( \| B(x, i) \| \leq b_B \) and \( \| C(x, i) \| \leq b_C \). The product \( CB \) is full rank.
Every batch starts within the vicinity of desired initial state $x^*(1)$ such that for a positive constant $b_{x_0}$, it satisfies $\|x^*(1) - x(1,k)\| \leq b_{x_0}$.

The input is updated using following ILC algorithm

$$u(i,k+1) = u(i,k) + K_d \{e(i+1,k) - e(i,k)\} \tag{3.58}$$

where $K_d$ is the learning gain matrix with bound $b_{Kd}$, such that $\|K_d\| \leq b_{Kd}$ for all $i \in M$. $e(i,k)$ is the error between desired and actual output.

Assume that for the time-varying system in Eq. (3.56) and (3.57) with ILC update algorithm in Eq. (3.58) under the assumptions presented, the following inequality holds for all $(x,i)$

$$\|I - K_dC(i+1)B(x,i)\| \leq \rho < 1 \tag{3.59}$$

The $\alpha$-norm with $\alpha \geq 1$, has been defined as under for a positive real function $q : N \rightarrow R$,

$$\|q(.)\|_\alpha = \sup_{i \in N} q(i) \left(\frac{1}{\alpha}\right)^\alpha$$

When there are no state or modelling errors, i.e., $b_w = 0$ and $b_{x_0} = 0$, the control input $u(i,k)$ converges to the desired $u^*(i)$, the state $x(i,k)$ to $x^*(i)$ such that output $y(i,k)$ follows the desired output $y^*(i)$ as batch number $k \rightarrow \infty$. However, when there is uncertainty as state or modelling errors, the convergence is achieved with error bounds as given below:

The error in ILC update algorithm in Eq. (3.58) by denoting $u^*(i) - u(i,k) = \Delta u(i,k)$ can be written as
\[ \Delta u(i, k + 1) = \Delta u(i, k) - K_d C(i + 1) \Delta x(i + 1, k) \]
\[ + K_d C(i) \Delta x(i, k) + K_d \left\{ v(i + 1, k) - v(i, k) \right\} \]  
\hspace{1cm} (3.60)

and using Eq. (3.56) for desired dynamics with out disturbance as
\[ x^*(i + 1) = f \left( x^*(i), i \right) + B \left( x^*(i), i \right) u^*(i) \]  
\hspace{1cm} (3.61)

then the Eq. (3.60) becomes
\[ \Delta u(i, k + 1) = \left[ I - K_d C(i + 1) B \left( x(i, k), i \right) \right] \Delta u(i, k) \]
\[ + K_d C(i + 1) w(i, k) + v(i + 1, k) - v(i, k) \]
\[ - K_d \left\{ f \left( x^*(i), i \right) - f \left( x(i, k), i \right) \right\} + K_d C(i) \left[ x^*(i) - x(i, k) \right] \]  
\hspace{1cm} (3.62)

Taking norms on both sides of Eq. (3.62),
\[ \| \Delta u(i, k + 1) \| \leq \rho \| \Delta u(i, k) \| + b_1 + h_2 \| \Delta x(i, k) \| \]  
\hspace{1cm} (3.63)

where used inequality (3.59) and \( \| u^* \| \leq b_w \) and
\[ b_1 = b_{kd}(b_c b_w + 2b_h), \ h_1 = c_f + 1 \ \text{and} \ \ h_2 = b_{kd} b_c h_1 \]

Similarly, subtracting both sides of Eq. (3.56) from \( x^*(i + 1) \) and denoting \( x^*(i + 1) - x(i + 1, k) = \Delta x(i + 1, k) \), it becomes
\[ \Delta x(i + 1, k) = f \left( x^*(i), i \right) - f \left( x(i, k), i \right) \]
\[ + \left\{ B \left( x^*(i), i \right) - B \left( x(i, k), i \right) \right\} u^*(i) + B \left( x(i, k), i \right) \Delta u(i, k) - w(i, k) \]  
\hspace{1cm} (3.64)

Taking norms on both sides of Eq. (3.64), to get as follows
\[ \| \Delta x(i + 1, k) \| \leq c_f \| \Delta x(i, k) \| + b_u c_B + b_w \| \Delta u(i, k) \| + b_w \]  
\hspace{1cm} (3.65)

Using lemma in Appendix-B, following inequality holds for Eq. (3.65)
\[ \| \Delta x(i, k) \| \leq \sum_{j=0}^{i-1} c_f^{i-1-j} \left[ b_B \| \Delta u(j, k) \| + b_u c_B + b_w \right] + c_f b_x \]  
\hspace{1cm} (3.66)
Substitute, Eq. (3.66) into Eq. (3.63), to get
\[
\|\Delta u(i, k + 1)\| \leq \rho \|\Delta u(i, k)\| + b_1 + h_2 c_j^i b_{z_0} \\
+ h_2 \sum_{j=0}^{j-1} c_j^{i-j} [b_B \|\Delta u(j, k)\| + b_u c_B + b_w ]
\] 

(3.67)

Multiply both sides of Eq. (3.67) by \( \left(\frac{1}{\alpha}\right)^i \) such that \( \alpha > \max[1, h_i] \)
\[
\|\Delta u(i, k + 1)\| \left(\frac{1}{\alpha}\right)^i \leq \rho \|\Delta u(i, k)\| \left(\frac{1}{\alpha}\right)^i + b_1 \left(\frac{1}{\alpha}\right)^i + h_2 c_j^i \left(\frac{1}{\alpha}\right)^i \\
+ \frac{h_2}{\alpha} \sum_{j=0}^{j-1} \left(\frac{c_j}{\alpha}\right)^{i-j} \left[b_B \|\Delta u(j, k)\| \left(\frac{1}{\alpha}\right)^j + \{b_u c_B + b_w\} \left(\frac{1}{\alpha}\right)^j \right]
\]

(3.68)

Since, the norm of a constant is also a constant, Eq. (3.68) becomes
\[
\|\Delta u(i, k + 1)\| \leq \rho \|\Delta u(i, k)\| + b_1 + h_2 c_j^i b_{z_0} \\
+ \frac{h_2}{\alpha} \sum_{j=0}^{j-1} \left(\frac{c_j}{\alpha}\right)^{i-j} \left[b_B \|\Delta u(j, k)\| + \{b_u c_B + b_w\} \right]
\]

(3.69)

Evaluate the sum and simplify Eq. (3.69),
\[
\|\Delta u(i, k + 1)\| \leq \rho + \frac{h_2 b_B}{\alpha - c_f} \left[1 - \left(\frac{c_f}{\alpha}\right)^n\right] \|\Delta u(i, k)\| + \varepsilon
\]

(3.70)

where
\[
\varepsilon = b_1 + h_2 c_j^i b_{z_0} + \frac{h_2}{\alpha - c_f} \left[b_u c_B + b_w\right] \left[1 - \left(\frac{c_f}{\alpha}\right)^n\right]
\]

(3.71)

For suitable values of \( \alpha \), so that
\[
\hat{\rho} = \rho + \frac{h_x b_B \left( 1 - \left( \frac{c_f}{\alpha} \right)^n \right)}{\alpha - c_f} < 1
\] (3.72)

This ensures that error norm \( \| \Delta u(i,k) \| \) decreases as batch no. \( k \to \infty \), so the relation becomes

\[
\limsup_{k \to \infty} \| \Delta u(i,k) \| \leq \frac{\varepsilon}{1 - \hat{\rho}}
\] (3.73)

Similarly for the states, multiply both sides of Eq. (3.66) by \( \left( \frac{1}{\alpha} \right)^i \),

\[
\| \Delta x(i,k) \| \left( \frac{1}{\alpha} \right)^i \\
\leq \frac{1}{\alpha} \sum_{j=0}^{i-1} \left( \frac{c_f}{\alpha} \right)^{i-j} \left[ b_B \| \Delta u(j,k) \| \left( \frac{1}{\alpha} \right)^j + \left\{ b_x c_B + b_w \right\} \left( \frac{1}{\alpha} \right)^j \right] + \left( \frac{c_f}{\alpha} \right)^i b_{x0}
\] (3.74)

Simplifying Eq. (3.74)

\[
\| \Delta x(i,k) \| \leq b_B \left[ 1 - \left( \frac{c_f}{\alpha} \right)^n \right] \| \Delta u(j,k) \| + \left\{ b_x c_B + b_w \right\} \left[ 1 - \left( \frac{c_f}{\alpha} \right)^n \right] + b_{x0}
\] (3.75)

which implies

\[
\limsup_{k \to \infty} \| \Delta x(i,k) \| \leq \frac{b_B \left[ 1 - \left( \frac{c_f}{\alpha} \right)^n \right] \varepsilon}{(\alpha - c_f)(1 - \hat{\rho})} + \left\{ b_x c_B + b_w \right\} \frac{1 - \left( \frac{c_f}{\alpha} \right)^n}{\alpha - c_f} + b_{x0}
\] (3.76)

Subtracting \( y^*(i,k) \) from both sides of output Eq. (3.57)
\[ \Delta y(i,k) = C \Delta x(i,k) - v(i,k) \]  
\hfill (3.77)

Taking norm on both sides of Eq. (3.77) and multiplying both sides with \( \left( \frac{1}{\alpha} \right)^i \), to get

\[ \| \Delta y(i,k) \| \left( \frac{1}{\alpha} \right)^i \leq c_y \| \Delta x(i,k) \| \left( \frac{1}{\alpha} \right)^i + b_v \left( \frac{1}{\alpha} \right)^i \]  
\hfill (3.78)

where used \( \| C(i)(x_1,i) - C(i)(x_2,i) \| \leq c_y \| x_1 - x_2 \| \). Further simplifying Eq. (3.78), gives

\[ \| \Delta y(i,k) \| \leq c_y \| \Delta x(i,k) \| + b_v \]  
\hfill (3.79)

Using Eq. (3.75) in Eq. (3.79), the limiting relation for the output error becomes

\[ \lim_{k \to \infty} \| \Delta y(i,k) \| \leq \frac{c_y b_B \left[ 1 - \left( \frac{c_f}{\alpha} \right)^n \right] \varepsilon}{(\alpha - c_f) (1 - \hat{\rho})} + \frac{c_y \left\{ b_v c_B + b_w \right\} \left[ 1 - \left( \frac{c_f}{\alpha} \right)^n \right]}{\alpha - c_f} \]
\hfill (3.80)

\[ + c_y b_{x0} + b_v \]

**Remarks:**

For the ideal case when there are no disturbances or uncertainty along with identical initial conditions, the bounds are \( b_w = 0, b_v = 0, b_{x0} = 0 \) and \( b_{x0} = 0 \), so that error bound \( \varepsilon = 0 \), as well and hence the input, state and output errors tend to zero as per the inequalities in equations (3.73), (3.76) and (3.80) respectively as:

\[ \| \Delta u(i,k) \|_p \to 0, \| \Delta x(i,k) \|_p \to 0 \text{ and } \| \Delta y(i,k) \|_p \to 0 \]
3.7.4 Simulations with variable parameters

Here, we have analyzed the effects of a range of parameter variations using phase portraits. The hip and leg masses have been changed from +/- 20% of that used in the nominal model of the compass gait robot. The phase portraits for un-actuated compass gait robot on flat surface are shown in Figure 3.9 and Figure 3.10 below.

![Phase Portraits](image)

Figure 3.9 Effects of +/- 20% change in hip mass of CG robot

The changes in phase portraits point towards the effect of the parameter variations on the walking gait of the robot.
For stable walking gait the phase portraits have to stay inside a permissible band.

3.8 Summary

The mathematical model for the compass gait bipedal walking robot has been formulated. The dynamic relationship is represented by a second order nonlinear equation which is further expressed as a state space system. The linearised relationship has also been derived. Effects of parameter uncertainty have been discussed.
Chapter 4

Convergence Analysis - ILC with Zero Initial Errors

This chapter deals with the rate of convergence of conventional ILC algorithms with respect to control input. Discrete-time linear state space representation of a linear time-invariant system has been considered along with usual assumptions which ensure D-ILC algorithm convergence in terms of output error as well. The relationship for the rate of convergence of control input up to component level has been formulated from the matrix controlling the evolution of batch to batch input errors. Using this relationship gives the rate for any component during earlier iterations as well rather than the usual asymptotic analysis developed for output error case earlier [104, 105].

4.1 Discrete-time LTI System

A discrete-time LTI system is considered in Eq. (4.1) below.

\[
x(i + 1, k) = A x(i, k) + B u(i, k) \\
y(i, k) = C x(i, k)
\]  

(4.1)

where \( k \) denotes the batch number having \( M \) number of samples in each trial, \( i \in [1, M] \) is the time index or sample number during each batch, state vector \( x \in \mathbb{R}^n \), input \( u(i, k) \in \mathbb{R}^r \) and output of the system
is $y(i,k) \in \mathbb{R}^p$. $A$, $B$ and $C$ are the real-valued state, input, and output matrices respectively having appropriate dimensions. The initial condition $x(1,k) = x_0$ is same at the start of each batch. $u(i,1) = u_0(i)$ is the control input vector for first batch which may be externally specified or left to be zero [38, 71].

The additional assumptions made are as follows.

- Desired output $y^*(i)$ for the complete batch is known.
- The plant parameters $A$, $B$, $C$ remain unchanged.
- Noise free cases are considered.
- Solution of the DT system is given in Eq.(4.1) [106, 107].

$$y(i,k) = CA^i x_0 + \sum_{j=0}^{i-1} CA^{i-j-1} Bu(j,k) \tag{4.2}$$

With the knowledge of the desired output $y^*(k)$, the ILC searches the desired input $u^*(k)$. To find the vector of desired inputs is the objective of any control algorithm. So, it is an inverse problem.

The difference between the desired and the actual outputs is the error as given in Eq. (4.3).

$$e(k) = y^*(k) - y(k) = [e(1,k), e(2,k), \ldots, e(M,k)]^T \tag{4.3}$$

This error is the basis for formulation of performance index to update the control input generated by the ILC.
4.2 **D-ILC Control Input Update**

Using D-ILC the control input is updated on the basis of the derivative of error in the previous iteration at the given time index as given in Eq. (4.4) below.

\[
    u(i,k + 1) = u(i,k) + K_d \{e(i + 1,k) - e(i,k)\}
\]  

(4.4)

where \( K_d \) is the real-valued learning gain matrix and the derivative of error \( \dot{e}(i,k) \) has been approximated as forward difference given in Eq. (4.5) below

\[
    \dot{e}(i,k) = e(i + 1,k) - e(i,k)
\]

(4.5)

with the identical initial conditions or same output \( y^*(1) = y(1,k) \) at the 1st time index for each batch, the initial error, \( e(1,k) = 0 \). The objective of the D-ILC algorithm is to find the sequence \( u(i,k) \) which matches the desired input sequence as shown in Eq. (4.6). Owing to linear relationship between input \( u \) and output \( y \) of the LTI system, the convergence of control input is equivalent to convergence of output.

Hence, \( u(i,k) \) converges under the same conditions as \( y(i,k) \) converges.

\[
    \lim_{k \to \infty} u(i,k) = u^*(i) \quad \text{for } \forall i = 1,2,\ldots,M
\]

(4.6)

It also ensures convergence of the output in Eq. (4.7) due to LTI system as under.

\[
    \lim_{k \to \infty} y(i,k) = y^*(i) \quad \text{for } \forall i = 1,2,\ldots,M
\]

(4.7)
In ILC domain, usually the convergence of algorithm has been investigated in terms of output error and conditions for convergence and monotonicity have been established as mentioned in next section.

4.3 Convergence of ILC in Terms of Output Error

Convergence of an ILC algorithm is generally stated as the minimisation of the tracking error $e(i,k)$, so that the perfect tracking of ideal/desired output $y^*(i)$ is achieved as $k$, the number of batches increases [108].

$$\lim_{k \to \infty} y(i,k) \to y^*(i) \quad \text{for } \forall i = 1,2,\ldots,M$$

(4.8)

Or, equivalently

$$\lim_{k \to \infty} e(i,k) \to 0 \quad \text{for } \forall i = 1,2,\ldots,M$$

(4.9)

The convergence of error has been analysed using various measures or norms [32]. In the following sub-section, the relationship for the output error evolution upto component level (i.e., for each time index) has been formulated so that convergence of output error can be analysed.

4.3.1 Batch to Batch Output Error Evolution

The output at first sample time $i = 1$ is fixed for each iteration due to the reset to the same initial condition $x(1,k) = x_0$ for each batch as follows:

$$y(1,k) = Cx(1,k) = Cx_0 = y^*(1) \quad \forall k$$

(4.10)

Obviously, the error for first time index is always zero.
\[ e(1,k) = 0 \quad \forall k \] (4.11)

But, the errors at other time-indices need more investigation as follows.

As for the time index \( i = 2 \), the output at \((k + 1)^{th}\) iteration is

\[ y(2,k + 1) = Cx(2,k + 1) = CAx(1,k) + CBu(1,k + 1) \] (4.12)

Subtracting both sides of Eq.(4.12) from the desired output \( y^*(2) \) and substituting \( u(1,k + 1) \) from Eq.(4.4) and using the fact that error for the 1st time index \( e(1,k) = 0 \) for all batches

\[ \begin{align*}
  y^*(2) - y(2,k + 1) &= y^*(2) - CAx(1,k) - CBu(1,k) - CBK_d\{e(2,k) - e(1,k)\} \\
  \Rightarrow e(2,k + 1) &= y^*(2) - Cx(2,k) - CBK_d\{y^*(2) - y(2,k)\} \\
  \Rightarrow e(2,k + 1) &= y^*(2) - y(2,k) - CBK_dy^*(2) + CBK_dy(2,k)
\end{align*} \] (4.13)

On further simplification, the output error evolution at the time index \( i = 2 \), is

\[ e(2,k + 1) = [I - CBK_d]\{y^*(2) - y(2,k)\} \]
\[ \therefore e(2,k + 1) = [I - CBK_d]e(2,k) \] (4.14)

For the \((k + 1)^{th}\) iteration, the output at \( i = 3 \) is

\[ y(3,k + 1) = CAx(2,k + 1) + CBu(2,k + 1) \] (4.15)

Subtracting both sides of Eq.(4.15) from the desired output \( y^*(3) \), substituting \( u(2,k + 1) \) from Eq.(4.4) and using same initial condition \( x(1,k + 1) = x(1,k) \), the relationship becomes
\begin{align*}
y^*(3) - y(3,k + 1) &= y^*(3) - CAx(2,k + 1) - CBu(2,k) - CBK_d\{e(3,k) - e(2,k)\} \\
&= y^*(3) - CA\{Ax(1,k + 1) + Bu(k + 1)\} - CBu(2,k) - CBK_d\{e(3,k) - e(2,k)\} \\
&= y^*(3) - CA\{Ax(1,k) + Bu(k + 1)\} - CBu(2,k) - CBK_d\{e(3,k) - e(2,k)\}
\end{align*}

(4.16)

Re-arranging and further simplifying using the fact that error at 1st time index \(e(1,k) = 0 \quad \forall k\)

\begin{align*}
e(3,k + 1) &= y^*(3) - CA^2x(1,k) - CAB\left\{u(1,k) + K_d\left(e(2,k) - e(1,k)\right)\right\} \\
&\quad - CBu(2,k) + CBK_d e(2,k) - CBK_d e(3,k) \\
&= y^*(3) - CA\left\{Ax(1,k) + Bu(1,k)\right\} - CABK_d e(2,k) \\
&\quad - CBu(2,k) + CBK_d e(2,k) - CBK_d e(3,k) \\
&= y^*(3) - CAx(2,k) - CBu(2,k) - CBK_d e(3,k) \\
&\quad + CBK_d e(2,k) - CABK_d e(2,k)
\end{align*}

(4.17)

Finally, the relationship for the output error evolution at time index \(i = 3\) is given by

\begin{align*}
e(3,k + 1) &= e(3,k) - CBK_d e(3,k) + C(I - A)BK_d e(2,k) \\
\therefore e(3,k + 1) &= [I - CBK_d] e(3,k) + C(I - A)BK_d e(2,k)
\end{align*}

(4.18)

The general relationship for output error evolution for time index \(i = 2,\ldots, M\), is

\begin{align*}
e(i,k + 1) &= [I - CBK_d] e(i,k) + \sum_{j=2}^{i-1} CA^{i-(j+1)}(I - A)BK_d e(j,k)
\end{align*}

(4.19)

Eq.(4.19) can be re-written in matrix form as follows
\[
\begin{bmatrix}
    e(2, k + 1) \\
e(3, k + 1) \\
\vdots \\
e(M, k + 1)
\end{bmatrix} =
\begin{bmatrix}
    I - CBK^e_d & 0 & \cdots & 0 \\
    C(I - A)BK^e_d & I - CBK^e_d & \ddots & \vdots \\
    \vdots & \ddots & \ddots & 0 \\
    CA^{M-3}(I - A)BK^e_d & CA^{M-2}(I - A)BK^e_d & \cdots & I - CBK^e_d
\end{bmatrix}
\begin{bmatrix}
e(2,k) \\
e(3,k) \\
\vdots \\
e(M,k)
\end{bmatrix}
\]
(4.20)

Compactly, Eq.(4.20) is given as
\[
e(k + 1) = L_e e(k)
\]
(4.21)

where \(e(k)\) is the output error of the batch at the \(k^{th}\) iteration with corresponding output error \(e(k+1)\) at the \((k+1)^{th}\) iteration. The matrix \(L_e\) controls the output error evolution. Its spectral radius should be less than one for output error convergence. Since \(L_e\) is a Toeplitz matrix whose diagonal entries are the eigenvalues, hence the output error in Eq.(4.20) and Eq.(4.21) obtained using D-ILC algorithm in Eq. (4.4) is convergent under following condition [8, 48, 109].

\[
\max|\text{eig}(I - CBK^e_d)| < 1
\]
(4.22)

The product \((CBK^e_d)\) has to be non-singular for Eq. (4.22) to be valid.

### 4.4 Convergence of Control Input at Component Level Using D-ILC

In this section, the component-level relationship for \(u(i,k+1)\) at the \((k+1)^{th}\) batch is presented in terms of static and dynamic components as well as the control input \(u(i,k)\) from previous batch \(k\). Convergence
of control input \( u(i,k) \) approaching the desired input \( u^*(i) \) has been investigated as well.

\[
\lim_{k \to \infty} u(i,k) = u^*(i) \quad (4.23)
\]

So that the desired output sequence \( y^*(i) \) is generated when sequence \( u^*(i) \) is applied to the system having transfer function \( G(z) \) as shown below in Eq. (4.24)

\[
y^*(i) = G(z)u^*(i) \quad (4.24)
\]

In following sections, convergence condition for control input and rate of convergence for individual components of the control input have been derived. Bounds of convergence rates have also been formulated.

### 4.4.1 Batch to Batch Control Input Sequence

The D-ILC algorithm generates the batch to batch control input sequence, i.e., from 1,2,...,\( k,k+1,... \) for each time index \( i \). In this section a recurrence relationship is derived to perform a convergence analysis on \( u(i,k) \). The control input sequences for individual time indices are derived using Eq. (4.4).

For \( i = 1 \), we get

\[
u(1,k + 1) = u(1,k) + K_d \{ e(2,k) - e(1,k) \}
\]

\[
u(1,k + 1) = u(1,k) + K_d \{ e(2,k) \} \quad \text{since} \quad e(1,k) = 0 \quad (4.25)
\]

On further simplification, the Eq.(4.26) is obtained.
\[ u(1, k + 1) = u(1, k) + K_d \{ y(2) \} - K_d C \{ Ax(1, k) + Bu(1, k) \} \]  
\[ u(1, k + 1) = (I - K_d CB) u(1, k) + K_d \{ y(2) \} - K_d CA x(1, k) \]  

Similarly, for \( i = 2 \), there is
\[ u(2, k + 1) = (I - K_d CB) u(2, k) + K_d \{ y(3) - y(2) \} + K_d C (I - A) x(2, k) \]  

The sequences in Eq. (4.25) can be generalised for \( i^{th} \) component as.
\[ u(i, k + 1) = (I - K_d CB) u(i, k) + K_d \{ y(i + 1) - y(i) \} + K_d C (I - A) x(i, k) \]  

Solution for Eq. (4.28) is a recurrence relation in Eq.(4.29).
\[ u(i, k) = (I - K_d CB)^{k-1} u(i, 1) + \sum_{j=1}^{k-1} (I - K_d CB)^{k-j-1} \left[ K_d \{ y(i + 1) - y(i) \} + K_d C (I - A) x(i, k) \right] \]  

For zero control input at first batch, i.e., \( u(i, 1) = 0 \), the Eq. (4.29) reduces to Eq. (4.30) below.
\[ u(i, k) \]
\[ = \sum_{j=1}^{k-1} (I - K_d CB)^{k-j-1} \left[ K_d \{ y(i + 1) - y(i) \} + K_d C (I - A) x(i, k) \right] \]  

As batch number \( k \) increases, the control input components achieve convergence one after the other with increasing time index \( i \) inside a
batch. This can be observed from re-writing the sequences in Eq. (4.28) to show the dependency of \( u(i,k+1) \) on \( u(i,k) \) from the previous batch including all the other previous control input components \( u(i-1,k), u(i-2,k), \ldots, u(1,k) \) occurring in the same batch as follows

\[
\begin{align*}
  u(i,k+1) &= (I - K_d CB) u(i,k) + K_d C (I - A) Bu(i-1,k) \\
  &+ K_d C (I - A) A x(i-1,k) + K_d \{ y^*(i+1) - y^*(i) \} 
\end{align*}
\]  

(4.31)

From the expansion of \( x(i-1,k) \) down to \( x(1,k) \) using the dynamic relationship in Eq.(4.1), the Eq. (4.31) can be expressed in terms of the initial state and previous input terms from the same iteration \( k \) as given below

\[
\begin{align*}
  u(i,k+1) &= (I - K_d CB) u(i,k) + K_d C (I - A) Bu(i-1,k) \\
  &+ \cdots + K_d C (I - A) A^{i-2} Bu(1,k) + K_d C (I - A) A^{i-1} x(1,k) \\
  &+ K_d \{ y^*(i+1) - y^*(i) \} 
\end{align*}
\]  

(4.32)

The Eq. (4.32), has four groups of terms. 1st term shows the effect of the input \( u(i,k) \) at same time index during previous iteration. To obtain a stable and bounded sequence of control inputs, \( u(i,k+1) \), the poles of the scaling factor \( (I - K_d CB) \) must lie inside unit circle, i.e., maximum absolute value of eigenvalue of \( (I - K_d CB) \) also termed as spectral radius, should be less than 1 as given in Eq. (4.33)below.

\[
\max |\text{eig}(I - K_d CB)| < 1
\]  

(4.33)

2nd group of components consist of scaled input values which occurred during earlier in the last iteration. All of these have a common term
\((I - A)\) which has magnitude less than one due to stable system with eigenvalues of \(A\) less than one. These components shall always possess finite values. The 3\textsuperscript{rd} group is the scaled initial condition \(x(1,k)\) and 4\textsuperscript{th} is the scaled error \(\{y^* (i + 1) - y^*(i)\}\) between desired output values at consecutive time indices. Both of these have finite values. Hence, if the Eq. (4.33) is satisfied the sequence shall be bounded over the iterations. So Eq. (4.33) gives the necessary condition for convergence of D-ILC algorithm. It is clear that this condition does not depend on the system matrix \(A\), which implicitly points to the ability of ILC algorithm to achieve convergence even when the model parameters are unknown.

Owing to the presence of previous input components at all the earlier time indices of the previous iteration in Eq. (4.32), it can be inferred that first the convergence of the 1\textsuperscript{st} input component \(u(1,k) \rightarrow u^*(1)\) occurs, which causes the convergence for the 2\textsuperscript{nd} output component \(y(2,k) \rightarrow y^*(2)\). Then after one or few batches, 2\textsuperscript{nd} input component \(u(2,k) \rightarrow u^*(2)\) convergence is achieved followed by 3\textsuperscript{rd} output component \(y(3,k) \rightarrow y^*(3)\). Similarly, the convergence of other inputs \(u(i,k)\)'s and corresponding outputs \(y(i + 1,k)\)'s occur in a sequential manner in further batches. The process of batch to batch sequential convergence of samples is shown in the Figure 4.1.
4.5 Rate of Convergence of Control Input Error

In this section, rate of convergence of input sequence for D-ILC has been formulated at component level. Smallest singular value has also been utilised by some researchers to formulate the upper bound for convergence rate of the input sequence in case of norm optimal P-ILC [104]. For noise-free systems, exponential rate of convergence for control input sequence has been established in terms of infinity-norm [110]. The rate of convergence up to component level has not been observed in earlier works. The formulation of the evolution and rate of convergence of D-ILC algorithm in terms of control errors at component level allows the rate analysis for real applications having smaller batch numbers.

4.5.1 Input Error Evolution at Component Level - Formulation

Using D-ILC algorithm in Eq. (4.4), the component-wise control input error between desired control input \( u^*(i) \) and \( u(i,k+1) \) is calculated in Eq. (4.34) below.
\[ u^* (i) - u(i, k + 1) = u^* (i) - [u(i, k) + K_d \{ e(i + 1, k) - e(i, k) \}] \]
\[ = u^* (i) - u(i, k) - K_d \left[ \{ Cx^* (i + 1) - Cx(i + 1, k) \} - \{ y^* (i) - y(i, k) \} \right] \]

(4.34)

It can be re-arranged and further expressed as follows.

\[ u^* (i) - u(i, k + 1) = \]
\[ (I - K_dCB) \{ u^* (i) - u(i, k) \} + K_dC(I - A) \{ x^* (i) - x(i, k) \} \]

(4.35)

Let’s denote.

\[ u^* (i) - u(i, k) = \Delta u(i, k) \]

(4.36)

\[ u^* (i) - u(i, k + 1) = \Delta u(i, k + 1) \]

The Eq. (4.35) can now be re-written as.

\[ \Delta u(i, k + 1) = (I - K_dCB) \{ \Delta u(i, k) \} \]
\[ + K_dC(I - A) \{ x^* (i) - x(i, k) \} \]

(4.37)

Starting from time index \( i = 1 \), the relation for \( \Delta u(1, k + 1) \) is given as Eq. (4.38) below.

\[ \Delta u(1, k + 1) = (I - K_dCB) \{ \Delta u(1, k) \} \]
\[ + K_dC(I - A) \{ x^* (1) - x(1, k) \} \]

(4.38)

Since the initial conditions are preserved, hence, the initial error is zero, i.e., \( x^* (1) = x(1, k) \). At first time index the input error is given by.

\[ \Delta u(1, k + 1) = (I - K_dCB) \{ \Delta u(1, k) \} \]

(4.39)
Continuing for further time indices, the general relation for \( i^{th} \) time index can be written as under.

\[
\Delta u(i, k + 1) = (I - K_dCB)\Delta u(i, k) + K_dC(I - A)B\Delta u(i - 1, k) + \ldots + K_dCA^{i-1}(I - A)B\Delta u(2, k) + K_dCA^{i-2}(I - A)B\Delta u(1, k)
\]

(4.40)

From the equations (4.39), and (4.40) assuming similar relations for other time indices, these can be put in matrix form in Eq. (4.41) below

\[
\begin{bmatrix}
\Delta u(1, k + 1) \\
\Delta u(2, k + 1) \\
\Delta u(3, k + 1) \\
\vdots \\
\Delta u(M, k + 1)
\end{bmatrix}
= 
\begin{bmatrix}
(I - K_dCB) & 0 & 0 & \ldots & 0 \\
K_dC(I - A)B & (I - K_dCB) & 0 & \ldots & 0 \\
K_dCA(I - A)B & K_dC(I - A)B & (I - K_dCB) & \ddots & \vdots \\
\vdots & \vdots & \ddots & \ddots & 0 \\
K_dCA^{M-2}(I - A)B & \ldots & \ldots & \ldots & (I - K_dCB)
\end{bmatrix}
\begin{bmatrix}
\Delta u(1, k) \\
\Delta u(2, k) \\
\Delta u(3, k) \\
\vdots \\
\Delta u(M, k)
\end{bmatrix}
\]

(4.41)

OR more compactly in matrix-vector notation as in Eq. (4.42) below.

\[
\Delta u(k + 1) = T \Delta u(k)
\]

(4.42)

Here \( T \) is the operator matrix which controls the evolution of control input errors from batch \( k \) to \( k + 1 \).
The solution of Eq. (4.42) is given by Eq. (4.44) below.

\[
\Delta u(k) = T^{k-1} \Delta u(1)
\]  

(4.44)

4.5.2 Properties of Relation for Control Input Error Evolution

The properties of control input error evolution from Eq. (4.41) and (4.42) are as follows:

- It is a 1st order homogeneous relationship along \(k\).
- The matrix \(T\) has a lower Toeplitz structure, provided the parameters \(A, B, C\) and \(K_d\) are time-invariant.
- The matrix \(T\) has \(M\) repeated eigenvalues, i.e., \(\lambda_i = (I - K_dCB), \ \forall i = 1, \ldots, M\)
- Constant diagonal elements show that each \(\Delta u(i, k + 1)\) is effected by the corresponding \(\Delta u(i, k)\) from the previous batch in the same manner, i.e. same scaling factor \((I - K_dCB)\) which is independent of system matrix \(A\).
- Each \(\Delta u(i, k + 1)\) has additional coupling from other control input errors which occurred in the previous batch \(k\) before the
time index \( i \), such as \( \Delta u(1,k), \Delta u(2,k), \ldots, \Delta u(M-1,k) \). Hence \( \Delta u(M,k) \) is the slowest component.

- The scaling factors \( K_d CA^{i-2} (I - A) B \) are not independent of system matrix \( A \). Hence, although the ILC convergence condition is independent of system matrix \( A \), but the convergence rate for the components \( \Delta u(2,k), \Delta u(3,k), \ldots, \Delta u(M-1,k) \) except the 1st component \( \Delta u(1,k) \) does depend on matrix \( A \).

- The asymptotic convergence condition is given as follows.

\[
\max |\text{eig}(I - K_dCB)| < 1
\]  

(4.45)

which is similar to that for output error convergence.

But it has not been explored as to how the errors evolve in the initial part of the learning curve which corresponds to initial iterations? What is the relative rate of convergence? What factors affect? What is the rate of the learning process here? Does it diverge? How much divergence is acceptable?

4.5.3 Rate of Convergence for the Input Error at Component Level

It was motivated by need to analyse the effects of errors at the end of iteration upon initial state for next iteration. Additionally, the effect of batch size needs to be investigated so as to ascertain how much slower
is the learning at later time indices. The Theorem 4.1 below gives the relation for the rate of convergence for the input error at component level.

**Theorem 4.1**

The D-ILC algorithm produces a sequence of bounded control inputs which sequentially converge component-wise asymptotically to the desired control input sequence at the rate equal to the magnitude of the eigenvalue of the matrix relating the evolution of control input error provided initial conditions are same and desired output matches with the measured output at the beginning of each batch.

**Proof**

The evolution of control input errors follows relation.

\[
\Delta u(k + 1) = T \Delta u(k) \tag{4.46}
\]

And its solution in terms of initial value \( \Delta u(1) \) in Eq. (4.44).

\[
\Delta u(k) = T^{k-1} \Delta u(1) \tag{4.47}
\]

Due to repeated eigenvalues, matrix \( T \) cannot be diagonalised using eigen-decomposition or SVD because there are repeated eigenvectors which make the matrix of eigenvectors singular. However, the rate of convergence of each component \( \Delta u(i,k) \) can be found if matrix \( T \) with dimension \((M \times M)\) is decomposed into Jordan Normal form. For a
non-singular matrix $Q$ i.e. $Q^{-1} \times Q = I$, Jordan Decomposition is given by Eq. (4.48).

$$T = Q^{-1} D Q$$  \hspace{1cm} (4.48)$$

where the matrix $D$ has the eigenvalues of matrix $T$ as its diagonal elements along with a sub-diagonal containing all ones as in Eq. (4.49) below.

$$D = \begin{bmatrix}
\lambda & 0 & 0 & \cdots & 0 \\
1 & \lambda & 0 & \cdots & \vdots \\
0 & 1 & \lambda & \ddots & \vdots \\
\vdots & \ddots & \ddots & \ddots & 0 \\
0 & 0 & \cdots & 1 & \lambda
\end{bmatrix}$$  \hspace{1cm} (4.49)$$

To calculate the convergence rates of individual components of control input error vector $\Delta u(k)$ at $k^{th}$ batch in Eq. (4.47), decompose as $T^{k-1} = Q^{-1} D^{k-1} Q$. Now, the matrix $D^{k-1}$ is given below in Eq. (4.50).

$$D^{k-1} = \begin{bmatrix}
\lambda^{k-1} & 0 & 0 & \cdots & 0 \\
(k - 1) \lambda^{k-2} & \lambda^{k-1} & 0 & \cdots & \vdots \\
\frac{(k - 1)(k - 2)}{2!} \lambda^{k-3} & (k - 1) \lambda^{k-2} & \lambda^{k-1} & \ddots & \vdots \\
\vdots & \ddots & \ddots & \ddots & 0 \\
\frac{(k - 1)(k - 2) \cdots (k - M + 1)}{(M - 1)!} \lambda^{k-M} & \cdots & \cdots & \cdots & \lambda^{k-1}
\end{bmatrix}$$  \hspace{1cm} (4.50)$$

The control input error in Eq. (4.47) can now be written as Eq. (4.51) below
\[ \Delta u(k) = T^{k-1}Du(1) = Q^{-1}D^{k-1}Q\Delta u(1) \]  \hspace{1cm} (4.51)

The \( i^{th} \) term \( \Delta u(i,k) \) from the corresponding row in Eq. (4.51) is given in Eq. (4.52) below.

\[ \Delta u(i,k) = Q^{-1} \times D^{k-1}(i,:) \times Q \times [\Delta u(1,1), \Delta u(2,1), \ldots, \Delta u(M,1)]^T \]  \hspace{1cm} (4.52)

Where \( D^{k-1}(i,:) \) represents the \( i^{th} \) row of the matrix \( D^{k-1} \). Similarly, the \( i^{th} \) term \( \Delta u(i,k-1) \) from the previous \( (k-1)^{th} \) batch is given below.

\[ \Delta u(i,k-1) = Q^{-1} \times D^{k-2}(i,:) \times Q \times [\Delta u(1,1), \Delta u(2,1), \ldots, \Delta u(M,1)]^T \]  \hspace{1cm} (4.53)

The rate of convergence \( \Gamma_{\Delta u(i,k)} \) is the ratio of \( \infty \)-norm of the control input errors vector at batch \( k \) with respect to its \( \infty \)-norm at batch \( (k-1) \) as given in Eq. (4.54) below.

\[
\Gamma_{\Delta u(i,k)} = \frac{\lVert \Delta u(k) \rVert_\infty}{\lVert \Delta u(k-1) \rVert_\infty}
= \begin{vmatrix}
\lambda^k - 1 & \frac{(k-1)!}{(i-1)!} & \frac{(k-1)(k-2)!}{2!} \lambda^{k-3} \\
& \ldots & \\
& \frac{(k-2)!}{(i-1)!} & \frac{(k-2)(k-3)!}{2!} \lambda^{k-4} \\
& \ldots & \\
& \frac{(k-1-i+1)!}{(i-1)!} & \lambda^{k-1-i} \\
\end{vmatrix}
\]  \hspace{1cm} (4.54)
After simplification of Eq. (4.54), the Eq. (4.55) below is obtained

\[
\Gamma_{\Delta u(i,k)} = |\lambda| \left[ 1 + \left| \frac{(k-1)}{\lambda} \right| + \left| \frac{(k-1)(k-2)}{2!} \left| \frac{1}{\lambda} \right|^2 \right| \\
+ \ldots + \left| \frac{(k-1)\ldots(k-i+1)}{(i-1)!} \left| \frac{1}{\lambda} \right| \right| \\
+ \ldots + \left| \frac{(k-2)\ldots(k-1-i+1)}{(i-1)!} \left| \frac{1}{\lambda} \right| \right| \\
\right]^{(4.55)}
\]

More compact form for the rate of convergence is given by Eq. (4.56)

\[
\Gamma_{\Delta u(i,k)} = |\lambda| \left| \sum_{j=0}^{i-1} \left| \binom{k-1}{j} \left| \frac{1}{\lambda} \right|^j \right| \right| \\
= |\lambda| \left| \sum_{j=0}^{i-1} \frac{k-1}{k-1-j} \right|^{(4.56)}
\]

where

\[
\binom{k-1}{j} = \frac{(k-1)!}{(k-1-j)!j!}
\]

(4.57)

In the Eq. (4.56) there are as many terms inside summation as the index number. For components higher in index number the rate of convergence has more terms added thus they are slower as compared to the components with smaller index number occurring earlier. For a selected component, although the terms are same in number but at smaller number of iterations, the ratio terms inside summation are larger as compared to the corresponding terms at greater iteration.
number. Thus, the rate is slower at smaller iteration number for a
given component. The component at 1st time index has same rate
throughout which equals the absolute eigenvalue of matrix $T$.

From Eq. (4.56) it is observed that in the limit $k \to \infty$, the ratio to
the right of summation is unity, so that every component $\Delta u(i, k)$ shall
have long term convergence rate equal to the magnitude of the
eigenvalue $|\lambda| = |\text{eig}(I - K_0CB)|$. □

**Corollary 4.2**

The D-ILC algorithm produces a sequence of bounded control inputs
which converge to the desired control input sequence at the rate less
than or equal to the magnitude of the norm of matrix relating the
evolution of control input error under identical initial conditions such
that the desired output matches with the measured output at the
beginning of each batch.

**Proof**

Rate of convergence $r_u$ of the control input error vector is defined as
ratio between the norms of control input errors at $(k + 1)^{th}$ and $k^{th}$
batches in Eq. (4.58) below.

$$r_u \triangleq \frac{\|\Delta u(k + 1)\|}{\|\Delta u(k)\|}$$

(4.58)

Using Eq. (4.42) and the norm properties, the following inequality
results
Hence, the upper bound for the convergence rate is the norm of matrix $T$ which relates the control input error vector $\Delta u(k)$ to the vector $\Delta u(k + 1)$ in next batch.

4.6 Case Study: 1st Order SISO System

A first order discrete-time plant is investigated for convergence of control input errors using D-ILC algorithm. The state space description of the plant is given in Eq. (4.60) below.

$$
x(i + 1, k) = Ax(i, k) + Bu(i, k)$$

$$
y(i, k) = Cx(i, k)$$

The 1st order SISO plant parameters have been chosen as $A = 0.50$, $B = 0.50$, $C = 1$. These values are selected to achieve a normalized step response. For this system, the time constant is 2 samples. Hence the output within a batch shall converge in 6 to 8 samples. The batch size $M$ of 20 samples have selected so that the transient period is covered along with sufficient number of samples for steady state period. While satisfying the assumptions and preserving the initial conditions for each batch, the D-ILC algorithm in Eq. (4.4) has been run for number of batches with the learning rate $K_d = 0.5$. The initial values of control input $u(i,1)$ and output $y(i,1)$ for the 1st batch are all zeros. The matrix $T$ relates the control input error vector $\Delta u(k)$ to the vector $\Delta u(k + 1)$ as given below.
\[ \Delta u(k + 1) = T \Delta u(k) \]  

(4.61)

The singular values and the eigenvalues of matrix \( T \) are less than 1 as seen in Figure 4.2. Batch to batch evolution of the control input error in Figure 4.3 shows that different components \( \Delta u(i, k) \) start off at different rates. The error for the 1\textsuperscript{st} sample time, \( \Delta u(1, k) \) has the fastest rate.

![Figure 4.2: Singular values and eigenvalues of matrix \( T \)](image)
Figure 4.3: Evolution of components of control input errors

Figure 4.4: Short range evolution of control input errors
Figure 4.4 shows semi-log plot for short range, $k \leq 2M$, i.e., batch number up to twice the batch size. All the control input error components converge at different rates. $\Delta u(1,k)$ has rate 0.75 which equals the eigenvalue of matrix $T$. The slope of the norm of error vector $\Delta u(k)$ indicates that it is slower than all the individual components. For medium range, i.e., $2M < k < k_{LARGE}$ an earlier portion is shown in Figure 4.5. The components $\Delta u(2,k)$ to $\Delta u(20,k)$ achieve constant rates in succession. The convergence rate of vector $\Delta u(k)$ matches closely with last component $\Delta u(20,k)$.

![Control input residuals: Log Scale](image)

**Figure 4.5: Medium range evolution of control input errors**
Figure 4.6: Long range evolution of control input errors

Figure 4.7: Convergence rates of control input components
The D-ILC algorithm achieves its limit at large batch numbers, i.e., at $k \geq k_{\text{LARGE}} \approx 2600$ as shown in Figure 4.6. At this stage, just before the control input errors are below the threshold for the minimum number which can be represented numerically, all the individual components $\Delta u(i,k)$ achieve the same rate 0.75. Overall batch to batch convergence rates in Figure 4.7 has also confirmed that control input error components $\Delta u(i,k)$ approach 0.75, the eigenvalue of $T$. This proves the claim in Theorem 4.1 that long-term convergence rates of all the individual components equals the eigenvalue of $T$.

![Figure 4.8: Bounds for rate of convergence](image-url)

The bounds of convergence rate demonstrated in Figure 4.8 show that overall convergence rate of the control input error vector $\Delta u(k)$ lies between the two extremes. These extremes are the maximum singular
value of matrix $T$ as the upper limit and eigenvalue of matrix $T$ as the lower limit. This demonstrates the observation in Corollary 3.2 that rate is bounded by the maximum singular value and the eigenvalue of matrix $T$.

### 4.7 Case Study: ILC Algorithm for a Damped Pendulum

A damped pendulum in Figure 4.9 is an interesting control problem which has been widely studied. The D-ILC algorithm is applied for learning to track the desired angle and corresponding angular velocity by generating the desired control signal at each sample time. The set of desired control signal for a complete swing has been obtained from a fine-tuned PD controller. The angle $\theta$ is measured anti-clockwise. The control input torque $u$ is considered positive in anti-clockwise direction. The pendulum is at rest with initial position $\theta_0 = \pi / 4$ radians at the left side. The initial angular velocity is $\omega_0 = 0$ rad/s. The state space representation of the simple pendulum using state vector $x = [\theta, \omega]^T$ is given by Eq. (4.62) below

$$
\dot{x} = \begin{bmatrix}
0 & 1 \\
-g/L & -b/mL^2 \\
\end{bmatrix} x + \begin{bmatrix}
0 \\
1/mL^2 \\
\end{bmatrix} u \\
\end{bmatrix}
\begin{bmatrix}
1 & 0 \\
0 & 1 \\
\end{bmatrix} x \\
$$

(4.62)
The pendulum has been simulated with the parameters in Table 4-1.

![Diagram of a damped pendulum]

**Figure 4.9: A damped pendulum**

The pendulum has been simulated with the parameters in Table 4-1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length ($L$)</td>
<td>1 m</td>
</tr>
<tr>
<td>Mass ($m$)</td>
<td>0.5 Kg</td>
</tr>
<tr>
<td>Acceleration due to gravity ($g$)</td>
<td>9.81 m/s²</td>
</tr>
<tr>
<td>Damping co-efficient ($b$)</td>
<td>0.25 N-s/m</td>
</tr>
</tbody>
</table>

The linearised discrete space state matrices for the pendulum sampled at 0.05 seconds are as follows
The time period for the pendulum is 2 seconds, so there are 40 samples for each swing from right to left and back. Each swing is equivalent to a batch or iteration as used in standard ILC literature.

Here three forms of gain matrices have been considered to analyse the rate of convergence of the control inputs over a number of swings. The identical initial conditions, i.e. $\theta_0 = \pi/4$ radians and $\omega_0 = 0$ rad/s are maintained for each repetition/swing. The control input during 1st swing is assumed zero at all sampling instances.

**4.7.1.1 Case I: Gain Matrix $K_d = 0.5 \times [0,1]$**

For ILC update, the error considered is the state error. The selected gain matrix allows the forward difference error of the angular velocity only to be used for updating the control inputs. The tracking of the desired control input $u$, angle $\theta$ and angular velocity $\omega$ has been monitored for 2500 swings. See Figure 4.10 and Figure 4.11. For the first swing, there is zero control input and the pendulum exhibited as a damped, un-forced pendulum as shown by dotted lines. In the later swings the ILC learns to generate control input signals as shown by dashed lines for 100th swing. Although the output is closer to the required, but perfect tracking has not yet occurred. The $\times$'s represent the perfect tracking achieved after 2500 swings. The RMS error for the
control input and the corresponding angle and velocity exhibit convergence as shown in Figure 4.12. The root mean squared values have reached minima after 2000 iterations as shown in Figure 4.12. The final RMS values have been given in Table 4-2 below. These values show that limit of precision have been reached.

Table 4-2: Final RMS values

<table>
<thead>
<tr>
<th>Errors in variable</th>
<th>RMS values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle</td>
<td>1.2015e-015</td>
</tr>
<tr>
<td>Angular velocity</td>
<td>2.2210e-015</td>
</tr>
<tr>
<td>Control input</td>
<td>1.6876e-014</td>
</tr>
</tbody>
</table>

Figure 4.10: Tracking control input with ILC algorithm
Figure 4.11: Tracking desired angle and velocity using ILC

Figure 4.12: RMS control input, angle and velocity errors
The evolution of control input errors at selected time indices in Figure 4.13 shows how the D-ILC algorithm learns to minimise the error. For the 1st and 2nd time indices, the control input matches the desired, since there is no significant error in the angles and angular velocities until 3rd time index due to identical initial conditions at each iteration. When the error becomes significant for the 4th time index, the control input for the 3rd time index starts learning due to forward difference in the later swings/iterations. But due to minimal errors, it is quite close to the desired input. The errors for other time indices go on increasing because the damping effect increases with the angle and slows down the pendulum. Hence the control input lags behind the desired and ILC learns to catch up the difference in consecutive swings/iterations. The convergence occurs sequentially as the control input for earlier time
indices converge before the later ones. The rate of convergence varies in
the earlier swings. The semi-log plot in Figure 4.14 shows that near the
final convergence, all the slopes are almost parallel to each other
indicating same convergence rates. The absolute values of the control
input errors have been used to plot the convergence rates, because the
negative errors cannot be plotted on logarithmic scale along with the
positive errors simultaneously. The kinks in convergence rates for initial
swings in Figure 4.15 occur due to oscillations or zero-crossings of the
control input errors. Hence this portion is ignored for convergence rate
analysis. Only the medium and long-term convergence rates are valid
for the rate analysis as shown in Figure 4.16 and Figure 4.17.

Figure 4.14: Convergence rates for control input errors
Figure 4.15: Short-term rates for control input errors

Figure 4.16: Medium-term rates for control input errors
It can been observed that the control input errors for all the time indices finally converge at the rate equal to the eigenvalue (0.9754) of matrix $T$. For the eigenvalue of 0.9754, single time constant corresponds to 40 iterations. For 5 time constants equal to 200 iterations, the decay of 0.01 has been marked in Figure 4.17.

4.7.1.2 Case 2: Gain Matrix $K_d = 0.5 \times [1,1]$

For Case 2, the gain matrix is selected so that derivative of the full state error vector is employed. In this case, the performance is better because the convergence has been achieved 200 swings earlier than Case 1. The root mean squared errors in angle, angular velocity and control input errors reach minima around 1800th swing as shown in Figure 4.18.
Figure 4.18: RMS errors in angle, velocity and control input

Figure 4.19: Evolution of input error at selected time indices
Evolution of input errors is illustrated in Figure 4.19. The parallel slopes in Figure 4.20 indicate that the long-term convergence rates for control input errors at all the time indices have achieved same rate which equals the eigenvalue (0.9748) of matrix $T$. One time constant corresponds to 40 iterations. The decay of 0.01 for 5 time constants, i.e. 200 iterations has been marked in the Figure 4.20.

### 4.7.1.3 Case 3: Gain Matrix $K_d = 0.5 \times [1,0]$

The gain matrix for Case 3 allows the use of forward difference error in angle only. Here, the convergence occurs much slower as compared to the earlier two cases. RMS of the errors reaches minima after 170000 iterations in Figure 4.21.
Figure 4.21: RMS values for D-ILC with zero initial conditions

Figure 4.22: Convergence rate for input error
The slopes in Figure 4.22 indicate that the long-term convergence rates for control inputs achieve the rate which equals the eigenvalue (0.9994) of matrix $T$. On comparing the eigenvalues for the above three cases, it is observed that convergence is fastest for case 2 (using full state error) for which the matrix $T$ has smallest eigenvalue. Rate is slowest for the case 3 for which matrix $T$ has the greatest eigenvalue.

### 4.7.2 Controllability via D-ILC

The rank of product $CB$ is 1, which points that the algorithm can only control one output variable and achieve tracking for it. Since, the desired values of inputs as well as outputs were obtained from a fine tuned PD controller; the controller successfully tracked both the angle and velocity. It has been a very specific mapping from input to the output space. The algorithm in general can only track one variable, namely the velocity.

### 4.7.3 Effect of Offset State Errors

With identical initial conditions the states have been perturbed with an offset error. The D-ILC algorithm with gain matrix $K = 0.5 \times [0,1]$ has converged for velocity only with same rate as noted for case 1. It has been shown in Figure 4.23 under the limitation that only the velocity error has been used to generate the control input update.
However, convergence in velocity has not been observed for the cases 2 and 3 with offset states. The input and angle errors have remained bounded but have not converged over the iterations for any case.

### 4.8 Summary

In this chapter, mathematical formulation for conventional identical initial condition ILC has been presented. The convergence of D-ILC algorithm in terms of control input sequence has been formulated and investigated with simulations. The convergence conditions similar to the output sequence have been achieved. Moreover, rate of convergence of control input up to component level has been formulated using the eigenvalues of the operator matrix controlling the evolution of batch to batch input errors. It has been demonstrated that components of the
input sequence converge sequentially. The component at 1\textsuperscript{st} sample has fastest convergence rate. The remaining components initially converge at lower rates. However, all the components later on acquire the same rate equal to the eigenvalue of the operator matrix. The bounds of the convergence rate have been established in terms of maximum singular value and eigenvalue of the operator matrix. The theorems have been stated and proved with mathematical reasoning about the convergence rates and factors affecting the rate at the component level. The results from simulations have supported these concepts presented in the chapter. The rate of convergence analysis in this chapter shall be expanded to non-zero initial condition Cyclic ILC in next chapter for repetitive impacting systems such as walking robots where the gait is sensitive to the errors at last time index of the previous step.
Chapter 5

Convergence Analysis for Cyclic ILC

For impacting systems such as pendulums, walking robots, etc., the identical initial conditions cannot be maintained. So, there are two problems. Firstly, there is non-zero initial error at the start of the batch. Secondly, the batch length or number of samples in an iteration may not be same during each iteration. In this chapter, convergence of the ILC algorithms has been analysed after removal of the identical initial conditions constraint. The performance of Cyclic D-ILC and Cyclic PD-ILC algorithms have been compared and analysed. Specifically focus has been upon the Cyclic ILC for which the errors in each batch are transferred to the beginning of next batch.

5.1 Control Input & Initial State Errors for Cyclic D-ILC

The evolution of errors in Cyclic ILC is dependent on the state error which is transferred from the end of the previous batch. The evolution in terms of control input errors has been presented in following sections.
5.1.1 Problem Formulation

A discrete-time MIMO LTI system is considered in Eq.(5.1) below.

\[
\begin{align*}
    x(i + 1, k) &= A x(i, k) + B u(i, k) \\
    y(i, k) &= C x(i, k)
\end{align*}
\]

(5.1)

where \(k\) denotes the batch number having \(M\) number of samples in each iteration, \(i \in [1, M]\) is the time index or sample number during each iteration. \(x \in \mathbb{R}^n\) denotes the state vector. \(u(i, k) \in \mathbb{R}^p\) and \(y(i, k) \in \mathbb{R}^m\) represent the inputs and outputs of the system, respectively. \(A \in \mathbb{R}^{n \times n}\) is real-valued state matrix. The input matrix is \(B \in \mathbb{R}^{n \times p}\) and \(C \in \mathbb{R}^{m \times n}\) is the output matrix such that all states are fully observable. \(u(1, k)\) is the control input for first batch which may be externally specified or left to be zero. The system in Eq.(5.1) is considered stable.

Control inputs for consecutive iterations are updated using D–ILC algorithm as follows.

\[
\begin{align*}
    u(i, k + 1) &= u(i, k) + K_d \left\{ e(i + 1, k) - e(i, k) \right\}
\end{align*}
\]

(5.2)

where \(u(i, k)\) is the vector of \(p\) inputs at \(i^{th}\) time index in batch number \(k\), \(u(i, k + 1)\) is the vector of \(p\) inputs at the same \(i^{th}\) time index in next batch no. \(k + 1\). \(K_d \in \mathbb{R}^{m \times p}\) is the learning gain matrix, \(\left\{ e(i + 1, k) - e(i, k) \right\}\) is the forward difference error vector used to approximate the derivative of the output error vector of \(m\) outputs at \(i^{th}\) time index in \(k^{th}\) batch.
5.1.2 Modelling Cyclic ILC

At the start of each batch $k$, the initial state vector $x(1,k)$ without resetting has been assumed to consider Cyclic ILC. The state error at the last time index of each batch is retained and used as the state error for the 1st time index of the next consecutive batch such that $\Delta x(1,k + 1) = \Delta x(M,k)$. Since, the initial state errors also contribute in the control input update, hence, these have to be accounted for in the evolution of control input errors over the iterations. Hence, for the analysis, the control input error $\Delta u(k) \in \mathbb{R}^{pM}$ which constitutes the input errors for the time indices $1 \rightarrow M$ during $k^{th}$ batch is augmented with initial state error vector $\Delta x(1,k) \in \mathbb{R}^n$. Thus a new augmented error vector $\Delta x_{aug}(k) \in \mathbb{R}^{n+pM}$ has been formulated as follows.

$$
\Delta x_{aug}(k) = \begin{bmatrix}
\Delta x(1,k) \\
\Delta u(1,k) \\
\Delta u(2,k) \\
\vdots \\
\Delta u(M,k)
\end{bmatrix}
$$  (5.3)

where $\Delta u(1,k)$ is the vector containing control input errors of all the $p$ inputs at first time index of $k^{th}$ batch. Likewise, $\Delta u(2,k)$ represents input errors at 2nd time index and so on until $\Delta u(M,k)$ for $M$ the last time index.
5.1.3 Evolution of Control Input and Initial State Errors

The evolution of the augmented error vector $\Delta x_{aug}(k)$ over batches has been expressed in terms of the homogeneous relationship in Eq. (5.4) below

$$\Delta x_{aug}(k+1) = T_D \Delta x_{aug}(k)$$

where the augmented operator matrix $T_D$ which relates the errors in the initial state and control input sequence at batch $k$ with the next consecutive batch $k+1$ is given below.

$$T_D = 
\begin{bmatrix}
A^M & A^{M-1}B & A^{M-2}B & \cdots & B \\
K_d C(I - A) & I - K_d CB & 0 & \cdots & 0 \\
K_d CA(I - A) & K_d C(I - A)B & I - K_d CB & \ddots & \vdots \\
\vdots & \vdots & \ddots & \ddots & 0 \\
K_d CA^{M-1}(I - A) & K_d CA^{M-2}(I - A)B & \cdots & \cdots & I - K_d CB \\
\end{bmatrix}
$$

(5.5)

5.1.4 Comments on the Structure of Error Evolution Matrix

The matrix $T_D$ has four distinct blocks. The first block $A^M$ gives the dynamic relationship of the initial state error vector $\Delta x(1,k)$ over the consecutive batches, i.e., as to how the initial state error is related to itself depends on the system matrix $A$ and specifically upon the size of the batch $M$. As $M$ gets larger for a given matrix $A$, the contribution becomes lesser and vice versa. But since the term $A^M$ is independent of
index number $i$ and batch number $k$, hence initial state error for every iteration is connected to the consecutive in the same manner. Second block consists of second element $A^{M-1}B$ to the last element $B$ from the first block row. This block gives the scaling factors for the input errors which affect the initial state error vector $\Delta x(1,k+1)$ in the next batch. The input errors $\Delta u(1,k)$ to $\Delta u(M,k)$ are multiplied by decreasing powers of $A$ from $A^{M-1}B$ to $B$, respectively which means that input error $\Delta u(M,k)$ at end of the batch has greatest contribution to $\Delta x(1,k+1)$ and lessens for each input error at previous time index until the input error $\Delta u(1,k)$ at the 1st time index which has the least contribution.

Third block consists of second element $K_dC(I-A)$ to the last element $K_dCA^{M-1}(I-A)$ from the first block column. This block consists of the scaling factors for the initial state error vector which determines the contribution to the input error at each time index during next batch. Fourth block constitutes the remaining $M \times M$ elements in the lower right similar to the Toeplitz matrix $T$ which describes the evolution of the input errors obtained via D-ILC algorithm under the zero initial state error condition as described in chapter 4.

5.1.5 Convergence of Error Evolution Matrix

For stable learning the error evolution matrix $T_D$ should have eigenvalues having magnitude less than or equal to one. This differs from the usual condition for zero-initial error case where only the
diagonal elements \( I - K_dCB \) have eigenvalues less than unit magnitude for stable learning. In case of Cyclic ILC case here, the eigenvalues cannot be just obtained from the diagonals since \( T_D \) is not a triangular matrix. Eigen analysis has been utilized to verify the stability of matrix \( T_D \). If there exists an eigenvector associated with unity eigenvalue then it points towards the case when augmented error vector \( \Delta x_{aug}(k) \) does not decrease to zero over batches rather it aligns with the eigenvector associated with unity eigenvalue being the dominant one. This fact has been stated and proved as Theorem 5.1 below.

**Theorem 5.1**

There exists an eigenvector of matrix \( T_D \) in Eq.(5.4) such that corresponding absolute eigenvalue equals one.

For proving the theorem, proceed by assuming that such an eigenvector exists and shall construct it from the matrix \( T_D \).

**Proof**

Let us denote the augmented eigenvector of matrix \( T_D \) associated with absolute unit eigenvalue as \( v^* = [v^*_{x0}, v^*_u]^T \), where \( v^*_{x0} \) is the initial state error vector and \( v^*_u = [v^*_{u1}, v^*_{u2}, \ldots, v^*_{uM}]^T \) is the vector of control input error vectors \( v^*_u \) to \( v^*_u \) for each time index 1 to \( M \). Each of the \( v^*_{u1} \) to \( v^*_{uM} \) has \( p \) elements for every input.
For unit eigenvalue, the eigenvalue equation of the matrix $T_D$ reduces to Eq.(5.6) below.

$$ (I - T_D) v^* = 0. \quad \text{(5.6)} $$

The Eq.(5.6) in expanded form as follows.

$$
\begin{bmatrix}
I - A^M & -A^{M-1}B & -A^{M-2}B & \cdots & -B \\
-K_dC(I - A) & K_dCB & 0 & \cdots & 0 \\
-K_dCA(I - A) & -K_dC(I - A)B & K_dCB & \ddots & \vdots \\
\vdots & \vdots & \ddots & \ddots & 0 \\
-K_dCA^{M-1}(I - A) & -K_dCA^{M-2}(I - A)B & \cdots & \cdots & K_dCB \\
\end{bmatrix}
\begin{bmatrix}
v_{x0}^*
\vdots 
v_{u1}^*
\vdots 
v_{u2}^*
\vdots 
v_{uM}^*
\end{bmatrix}
= \begin{bmatrix}
0 \\
0 \\
0 \\
\vdots \\
0
\end{bmatrix}
$$

(5.7)

where each element of 1st block column in Eq. (5.7) is associated with $v_{x0}^*$ which corresponds to the state errors at the start of each iteration. Rest of the block columns are associated with each of the single elements $v_{u1}^*$ to $v_{uM}^*$ which correspond to the errors in control input sequence at each time index $i \in 1, \ldots, M$. The elements of the augmented eigenvector $v^*$ can be obtained from Eq.(5.7) by simplifying the individual block rows.

Thus, from 2nd block row Eq. (5.8) is obtained as below.

$$-K_dC(I - A)v_{x0}^* + K_dC Bv_{u1}^* = 0 \quad \text{(5.8)}$$

Under the assumption of full observability $K_dC \neq 0$, the relationship in Eq.(5.8) holds if Eq.(5.9) holds.

$$Bv_{u1}^* - (I - A)v_{x0}^* = 0 \quad \text{(5.9)}$$
This gives the relationship between initial state errors $v^*_{x0}$ and the control input error at first time index $v^*_{u1}$ as in Eq.(5.10) below.

$$v^*_{x0} = (I - A)^{-1}Bv^*_{u1}$$  \hspace{1cm} (5.10)

For non-zero initial state error vector $v^*_{x0}$, a non-zero control input error $v^*_{u1}$ is obtained from Eq.(5.10) if the term $(I - A)$ is invertible. It is valid due to $\max|\text{eig}(A)|<1$.

Further proceed to construct the full eigenvector $v^*$ as follows

From 3rd block row in Eq.(5.7), following relation is obtained.

$$-K_dCA(I - A)v^*_{x0} - K_dC(I - A)Bv^*_{u1} + K_dCBv^*_{u2} = 0$$

$$\Rightarrow A(I - A)v^*_{x0} + (I - A)Bv^*_{u1} = Bv^*_{u2}$$  \hspace{1cm} (5.11)

Putting $v^*_{x0} = (I - A)^{-1}Bv^*_{u1}$ from Eq.(5.10) in to Eq. (5.11) and simplifying under the fact that $B \neq 0$, the Eq.(5.11) becomes.

$$ABv^*_{u1} + (I - A)Bv^*_{u1} = Bv^*_{u2}$$

$$\Rightarrow B\left(v^*_{u1} - v^*_{u2}\right) = 0$$

$$\Rightarrow v^*_{u2} = v^*_{u1}$$  \hspace{1cm} (5.12)

Similarly, from 4th block row in Eq.(5.7), the relation is.

$$v^*_{u3} = v^*_{u1}$$  \hspace{1cm} (5.13)

Simplifying in the same way for other block rows until the last block row of Eq.(5.7), relationship for the last element $v^*_{uM}$ is obtained as under.

$$v^*_{uM} = v^*_{u1}$$  \hspace{1cm} (5.14)
From Eqs. (5.10), (5.12), (5.13) and (5.14), and assuming under symmetry for the elements which occur in between, finally the eigenvector $v^*$ containing $n + pM$ elements associated with unity absolute eigenvalue in terms of only one element $v^*_{u1}$ is given as follows.

$$v^* = [(I - A)^{-1}B, 1, 1, ..., 1]^T \times v^*_{u1}$$ (5.15)

The Eq.(5.15) gives $v^*$, the eigenvector for the matrix $T_D$ associated with unity eigenvalue.

To prove the validity of Eq.(5.15), there is need to check the 1st block row also.

From 1st block row the relation is

$$\left(I - A^M\right) v^*_{x0} - A^{M-1}B v^*_{u1} - A^{M-2}B v^*_{u2} - ... - A B v^*_{u(M-1)} - B v^*_{uM} = 0$$ (5.16)

Using $v^*_{x0} = (I - A)^{-1}B v^*_{u1}$ from Eq.(5.10) and $v^*_{u1} = v^*_{u2} = v^*_{u3} = ... = v^*_{uM}$ from Equations (5.12), (5.13) and (5.14), write Eq.(5.16) in terms of $v^*_{u1}$ as follows.

$$\left(I - A^M\right) \left(I - A\right)^{-1}B v^*_{u1} - A^{M-1}B v^*_{u1} - A^{M-2}B v^*_{u1} - ...$$

$$- A B v^*_{u1} - B v^*_{u1} = 0$$ (5.17)

Re-arranging and simplifying Eq.(5.17) as follows

$$\left(I - A^M\right) \left(I - A\right)^{-1} = \left(A^{M-1} + A^{M-2} + ... + A + I\right)$$ (5.18)
and employing the infinite geometric series $\frac{1}{1-x} = \sum_{j=0}^{\infty} x^j$ for $|x|<1$ to matrix $A$ such that $(I-A)^{-1} = \sum_{j=0}^{\infty} A^j$, and simplify L.H.S. of Eq.(5.18) to find that it is equal to R.H.S. Hence, the Eq.(5.16) for 1st block row has been validated.

Thus, it is proved that eigenvector $v^*$ corresponding to unit magnitude eigenvalue exists for the matrix $T_D$ as given by Eq.(5.15).

$\square$

**Corollary 5.2**

The dominant eigenvector $v^*$ of matrix $T_D$ has the last $M$ elements $v_{u1}^*$ to $v_{uM}^*$ equal to each other which correspond to input errors. So, in the long term, the corresponding output error terms from $e(1,k)$ to $e(M,k)$ also converge such that they are equal to each other due to linear relationship between input and output. A constant offset error in the output shall be maintained over the long term.

**Corollary 5.3**

Further to corollary 4.2, as the consecutive output error terms from $e(1,k)$ to $e(M,k)$ which are used to update the control input for Cyclic D-ILC become equal to each other, the values of forward difference $e(i+1,k) - e(i,k)$ become zero for consecutive time indices of the $k^{th}$ iteration. Hence, no further update for control input can occur with Cyclic D-ILC algorithm. Thus the learning process has ceased.
Remarks 5.4

It has been observed that matrix $T_D$ obtained from Cyclic D-ILC has maximum absolute eigenvalue as unity, which means that learning process is non-expanding. However, it cannot asymptotically reduce the input errors to zero. Use of proportional error term in the input update such as Cyclic PD-ILC can minimize the input errors asymptotically as stated in Theorem 5.2 in section 5.3.

5.2 State Errors for Cyclic D-ILC

The state errors also evolve similar to the input error case. The contribution of state error at last time index $\Delta x(M,k)$ to each time sample in the next batch/swing has been incorporated through the entries in the last column of the evolution matrix $L_D$ in Eq. (5.19) below.

\[
L_D = \begin{bmatrix}
0 & 0 & 0 & \cdots & I \\
BK_d - A & I - BK_d & 0 & \cdots & A \\
A(BK_d - A) & (I - A)BK_d & \ddots & \ddots & A^2 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
A^{M-2}(BK_d - A) & A^{M-3}(I - A)BK_d & \cdots & \cdots & A^{M-1} + I - BK_d
\end{bmatrix}
\]

(5.19)

Use following matrix-vector notation for the evolution of state error.

\[
\Delta x(k + 1) = L_D \Delta x(k)
\]

(5.20)
The matrix $L_D$ has dimension $2M \times 2M$. But it is important to note that it is not a triangular matrix. Two eigenvalues of matrix $L_D$ are at unity due to the identity matrix block as the last element of 1st block row. Additionally, there are $M - 2$ matrix blocks $I - BK_d$ at the diagonal which also contribute $M - 2$ eigenvalues at the stability boundary which are of interest. Total number of eigenvalues at unity equals the number of samples in each swing. So there are as many dominant eigenvectors to which the state errors converge over the batches. Since, the eigenvectors are non-zero, the resulting state errors are non-zero as well. This also explains as to why the state errors do not converge using the Cyclic D-ILC algorithm for different initial conditions such as Cyclic D-ILC.

5.3 Convergence of Cyclic PD–ILC

To achieve convergence for the input and angle along with the velocity, Cyclic PD–ILC is employed for which the control input update utilizes both the state error and its derivative in Eq. (5.21).

$$u(i, k + 1) = u(i, k) + K_p e(i, k) + K_d \{e(i + 1, k) - e(i, k)\} \quad (5.21)$$

Using PD–ILC the error evolution matrix $T_p$ for Cyclic D-ILC is amended as $T_{PD}$ below.
Cyclic PD-ILC does not have an eigenvector associated with unit magnitude eigenvalue as stated in the Theorem 5.2 below.

**Theorem 5.2**

There does not exist an eigenvector of matrix $T_{PD}$ such that corresponding absolute eigenvalue equals one.

For proving the theorem, proceed by assuming that such an eigenvector exists and shall try construct it from the matrix $T_{PD}$.

**Proof**

Let us denote the augmented eigenvector of matrix $T_{PD}$ associated with absolute unit eigenvalue as $\mathbf{v}^* = [v_{x0}^T, v_u^T]^T$. So there is

$$(I - T_{PD})\mathbf{v}^* = 0 \quad (5.23)$$

Write Eq.(5.23) in expanded form as follows.
\[
\begin{bmatrix}
I - A^M & -A^{M-1}B & -A^{M-2}B & \ldots & -B
\end{bmatrix}
\begin{bmatrix}
v^*_0
v^*_1
\vdots
v^*_{u1}
\end{bmatrix} =
\begin{bmatrix}
0
0
\vdots
0
\end{bmatrix}
\]

(5.24)

The elements of the augmented eigenvector \( v^* \) can be obtained from Eq. (5.24) by simplifying the individual block rows. Thus, from 2\(^{nd}\) block row the relation is obtained as

\[
\begin{align*}
\{ -K_d C (I - A) + K_p C \} v^*_x + K_d C B v^*_{u1} &= 0 \\
K_d C \{ B v^*_{u1} - (I - A) v^*_x \} + K_p C v^*_{x0} &= 0
\end{align*}
\]

(5.25)

As \( K_d C \neq 0 \) and \( K_p C \neq 0 \), the relationship in Eq. (5.25) holds if \( v^*_x = 0 \) along with the condition that following Eq. (5.26) holds.

\[
B v^*_{u1} - (I - A) v^*_x = 0 \\
\Rightarrow v^*_x = B(I - A)^{-1} v^*_{u1}
\]

(5.26)

Eq. (5.26) gives the relationship between initial state errors and the control input error at first time index.

The additional requirement for validity of Eq. (5.25) that initial state error \( v^*_x = 0 \), is contrary to the Cyclic ILC assumption except that when the initial state error has converged to zero.

From 3\(^{rd}\) block row of Eq. (5.24),
\begin{align}
\left\{-K_dCA(I - A) + K_pCA\right\}v^*_x - \left\{K_dC(I - A)B + K_pCB\right\}v^*_u + K_dCb^*_u &= 0 \\
\Rightarrow K_dC \left\{Bv^*_{u2} - (I - A)Bv^*_{u1} - A(I - A)v^*_x\right\} + K_pC \left\{Av^*_x - Bv^*_u\right\} &= 0 \\
\Rightarrow K_dC \left\{Bv^*_{u2} - Bv^*_{u1} + A\left\{Bv^*_{u1} - (I - A)\right\}v^*_x\right\} + K_pC \left\{Av^*_x - Bv^*_u\right\} &= 0
\end{align}

(5.27)

Since $K_dC \neq 0$ and $K_pC \neq 0$, the Eq.(5.27) is valid if the following are valid.

\begin{align}
Bv^*_{u2} - Bv^*_{u1} + A\left\{Bv^*_{u1} - (I - A)\right\}v^*_x &= 0 \quad \text{(5.28)} \\
Av^*_x - Bv^*_u &= 0 \quad \text{(5.29)}
\end{align}

If Eq.(5.26) holds and using the fact that $B \neq 0$ then Eq.(5.28) reduces to Eq.(5.30) below.

\begin{align}
v^*_{u2} = v^*_{u1}
\end{align}

(5.30)

Eq.(5.29) is simplified using $Bv^*_{u1} = (I - A)v^*_x$ from Eq.(5.26) as under.

\begin{align}
Av^*_x - (I - A)v^*_x &= 0 \\
2Av^*_x &= v^*_x
\end{align}

(5.31)

Eq.(5.31) requires that $A = \frac{1}{2}$, a fixed scalar quantity which is a conservative requirement against the assumption that $A$ is a matrix of dimension $n \times n$.

Likewise, checking for the remaining block rows, it is found that a valid eigenvector associated with unity eigenvalue cannot be constructed from matrix $T_{PD}$. Thus, the Theorem 5.2 has been proved. \qed
Corollary 5.5

Since, there is no eigenvector associated with unity eigenvalue, hence the dominant eigenvector as well as all the other eigenvectors are associated with eigenvalues less than unity. As the batches proceed, the contribution of all the eigenvectors to the overall augmented error vector $\Delta x_{aug}(k)$ diminish to zero, asymptotically. Thus, the augmented error vector $\Delta x_{aug}(k)$ converges to zero. In other words, with Cyclic PD-ILC the errors in the control input converge to zero even under non-zero initial state error assumption.

5.4 Evolution of State Errors for Cyclic PD-ILC

The state errors evolve for Cyclic PD-ILC under following relationship.

$$\Delta x(k+1) = L_{PD}\Delta x(k)$$ \hspace{2cm} (5.32)

where the state error evolution matrix $L_{PD}$ is given below.

$$\begin{bmatrix}
0 & 0 & 0 & \cdots & 0 & I \\
BK_d - A & I - B(K_p + K_d) & 0 & \cdots & A \\
A(BK_d - A) & BK_d - AB(K_p + K_d) & \ddots & \ddots & A^2 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
A^{M-2}(BK_d - A) & A^{M-3}\{BK_d - AB(K_p + K_d)\} & \cdots & \cdots & A^{M-1} + I - B(K_p + K_d) \\
\end{bmatrix}$$ \hspace{2cm} (5.33)
The matrix $L_{PD}$ has dimension $2M \times 2M$, and it has elements different from $L_D$ matrix in Eq. (5.19) for the Cyclic D-ILC case. The proportional gain $K_p$ has not only changed the state error evolution matrix but the corresponding eigenvalues and eigenvectors have changed as well. Now there are $M - 2$ number $I - B(K_p + K_d)$ block matrices at the diagonal. Each of these block matrices contributes one eigenvalue at unity.

5.5 Damped Pendulum with Cyclic ILC- Simulations

In this section, Cyclic D–ILC and Cyclic PD-ILC algorithms have been simulated to analyse the rate of convergence of the initial state and control input errors over a number of swings/batches for different initial conditions of the damped pendulum discussed in Section 3.7. Each swing consists of 20 samples. The initial conditions for the 1st swing only are similar to the desired, i.e., $\theta_0 = \pi/4$ radians and $\omega_0 = 0$ rad/s and control input during 1st swing is zero for all time indices. The formulation is based on the damped pendulum discussed in the previous chapter. The angle and the angular velocity at the last sample time of each swing are retained and used as the initial errors for the successive swing. It can be mathematically described by the following Eq. (5.34)
\[
\begin{bmatrix}
\Delta \theta(1, k + 1) \\
\Delta \omega(1, k + 1)
\end{bmatrix}
= 
\begin{bmatrix}
\Delta \theta(M, k) \\
\Delta \omega(M, k)
\end{bmatrix}
\]
\hspace{1cm} (5.34)

The initial conditions correspond to the errors in initial angle $\Delta \theta(1, k + 1)$ and the angular velocity $\Delta \omega(1, k + 1)$ of the pendulum at the start of a new swing/batch number $k + 1$.

5.5.1 CASE 1: Cyclic D-ILC

For gain matrix $K_d = 0.5 \times [1,1]$, both the root mean squared errors in the initial state and input are minimised for the earlier 200 iterations, but then stay constant at 0.0648 and 0.8984 respectively (Figure 5.1). The velocity has converged (Figure 5.3). However, a constant off-set is observed in the tracking for both the input and angle in Figure 5.2 and Figure 5.3. This off-set occurs due to an eigenvalue at the stability boundary, i.e., $\lambda = 1$. The response has not blown up which occurs as none of the eigenvalues is outside the unit circle. The eigenvalue 1 is thus the dominant one. The corresponding eigenvector is given in Eq. (5.35). The 2\textsuperscript{nd} element of the dominant eigenvector is 0.000000000000000 which corresponds to the angular velocity having value equal to zero. Thus, elements of the dominant eigenvector satisfy the Theorem 5.1.
\[ v^* = \begin{pmatrix}
0.02278784223275 \\
-0.000000000000000 \\
0.223548732298625 \\
0.223548732298620 \\
0.223548732298620 \\
0.223548732298620 \\
0.223548732298623 \\
0.223548732298618 \\
0.223548732298620 \\
0.223548732298619 \\
0.223548732298621 \\
0.223548732298620 \\
0.223548732298618 \\
0.223548732298619 \\
0.223548732298618 \\
0.223548732298618 \\
0.223548732298618 \\
0.223548732298618 \\
0.223548732298617 \\
0.223548732298617
\end{pmatrix} \]

For given \( A \) and \( B \) matrices of the pendulum and using \( v_3^* = 0.223548732298625 \), the values of components of the dominant eigenvector \( v^* \) are same as those given above. Specifically the values for \( v_1^*, v_2^* \) are found as follows.
\[
\begin{bmatrix}
v_1^*, v_2^*
\end{bmatrix}^T = (I - A)^{-1} B v_3^* = \begin{bmatrix}
0.022787842232275 \\
-0.000000000000000
\end{bmatrix}
\]

Eq. (5.36) is a mapping of \( v_3^* \) to \( v_1^*, v_2^* \). It is validated since it corresponds to input to output mapping under the condition that \( C \) is an identity matrix and matrix \( D = 0 \). As the batches proceed, the augmented error vector \( \Delta x_{\text{aug}}(k) \) aligns with the dominant eigenvector and the contribution to velocity goes to zero when the other eigenvectors have diminished. The rate at which this convergence occurs is \( \left| \frac{\lambda_2}{\lambda_1} \right| \) where \( \lambda_1 \) is the dominant eigenvalue and \( \lambda_2 \) is the 2\textsuperscript{nd} dominant eigenvalue. This corresponds to about 200 batches. The RMS values of the initial state and input errors (Figure 5.1) as well as input error (Figure 5.4), angle error (Figure 5.6) and velocity error (Figure 5.7) converge to their approximate final values in 200 batches but do not minimize to zero. Also the evolution of input and angle errors indicates that these have not converged to zero using D-ILC algorithm (See Figure 5.4 and Figure 5.6). Rather a certain amount of offset is maintained. Only the velocity has long-term convergence at a rate which approximately equals the 2\textsuperscript{nd} dominant eigenvalue 0.9749 (Figure 5.7 and Figure 5.8). This corresponds to 180 batches for the decay of 0.01 as marked in Figure 5.8.
Figure 5.8: Convergence rate for velocity error

Figure 5.1: Root mean squared initial state and input errors
Figure 5.2: Tracking of desired control input

Figure 5.3: Tracking of desired angle and velocity
Figure 5.4: Evolution of input error at selected time indices

Figure 5.5: Convergence rate of input error at selected indices
Figure 5.6: Evolution of angle error at selected time indices

Figure 5.7: Evolution of velocity error at selected time indices
The control input has not achieved convergence because Cyclic D-ILC algorithm depends on the difference of error between two consecutive time indices. The errors in angle become same for consecutive time indices in Figure 5.6 instead of becoming zero while the error in velocity goes to zero in Figure 5.7. The overall error vector becomes constant after 180 batches/swings, hence further learning or control input update has ceased. Thus, the learning algorithm requires update which not only depends on derivative of error but also penalises the error itself to handle different initial conditions. The constant off-set error can be minimised by introducing the proportional part in the
Cyclic ILC algorithm. It is Cyclic PD–ILC which has been analysed in next section 5.5.2.

5.5.2 CASE 2: Cyclic PD-ILC

Cyclic PD–ILC has successfully minimised constant off-set errors in the input and angle as well as guaranteed convergence of angular velocity. Using $K_p = 0.5 \times [1,1]$ and $K_d = 0.5 \times [1,1]$, the initial state errors and the input errors have been minimised to $1.6184e-016$ and $3.8579e-015$ respectively (Figure 5.9).

![RMS initial state and input errors using Cyclic PD-ILC](image)

Figure 5.9: RMS initial state and input errors using Cyclic PD-ILC

Tracking performance using Cyclic PD-ILC has shown that desired trajectories for all the three variables, i.e., control input, angle and velocity have been successfully tracked (Figure 5.10 and Figure 5.11).
Figure 5.10: Tracking of control input using Cyclic PD-ILC

Figure 5.11: Tracking of angle and velocity using Cyclic PD-ILC
Figure 5.12: Evolution of input error using Cyclic PD-ILC

Figure 5.13: Convergence rate of control input errors using Cyclic PD-ILC
The control input errors have evolved as shown in Figure 5.12. The rates for these time indices in Figure 5.13 have reached the numerical zero after 6000 batches. The maximum absolute eigenvalue of the matrix has been 0.9944 which has indicated the maximum rate of convergence. In Figure 5.13, it has been observed that slope consists of innumerable kinks which occur due to oscillations of the control input errors (Figure 5.12). To minimise these oscillations and hence the kinks in slope, $K_p < K_d$ has been employed in next section.

### 5.5.3 CASE 3: Cyclic PD-ILC Using P Gain Less Than D Gain

In this case, the gain matrices have been selected as $K_p = 0.05 \times [1,1]$, $K_d = 0.5 \times [1,1]$ such that $K_p << K_d$.

![Graph of RMS initial state and input errors](image)

Figure 5.14: RMS initial state and input errors
Figure 5.15: Tracking for different initial states with Cyclic PD-ILC

Figure 5.16: Tracking input for different initial states using Cyclic PD-ILC
Final values of RMS for initial state and input errors are $1.6184 \times 10^{-16}$ and $4.4479 \times 10^{-15}$ respectively (Figure 5.14). The desired angle, velocity and input have been tracked perfectly (Figure 5.15 and Figure 5.16). Since, the input errors do not have oscillations/zero-crossings (Figure 5.17), the rate in Figure 5.18 is almost a straight line and input errors for all the time indices converge at the same rate, i.e., 0.9949, equal to magnitude of the maximum eigenvalue for matrix $L_{PD}$. For the eigenvalue 0.9949, one time constant corresponds to 195 iterations. Decay of 0.01 corresponding to 4 to 5 time constants, i.e. 975 iterations has been marked in Figure 5.18. The angle and velocity errors have evolved with oscillating behaviour in the initial iterations (Figure 5.19 and Figure 5.20). This phenomenon has occurred due to complex
eigenvalues and corresponding complex eigenvectors. However, the 1st and 2nd dominant eigenvalues have real values 0.9949 and 0.9762 respectively. The ratio $\left| \lambda_2/\lambda_1 \right|$ gives the rate of convergence to the dominant eigenvector which corresponds to about 260 iterations. The dominant eigenvector has been given in Eq. (5.37).

$$v_{PD}^* = \begin{bmatrix}
0.22851482876001 \\
-0.00115687661548 \\
0.224087250705004 \\
0.224030418650714 \\
0.223973601009918 \\
0.22391679778979 \\
0.223860008954231 \\
0.223803234532024 \\
0.223746474508705 \\
0.223689728880617 \\
0.223632997644116 \\
0.223576280795552 \\
0.223519578331271 \\
0.223462890247629 \\
0.223406216540975 \\
0.223349557207666 \\
0.22329291224056 \\
0.223236281646501 \\
0.223179665411357 \\
0.223123063534982 \\
0.223066476013736 \\
0.223009902843975 \\
\end{bmatrix} \tag{5.37}$$
Figure 5.18: Long-term rate of input error with Cyclic PD-ILC

Figure 5.19: Evolution of angle error using Cyclic PD-ILC
The dominant eigenvector has all real entries hence no oscillations have been observed after 260 iterations for any of the input, angle and velocity errors (See Figure 5.17, Figure 5.19 and Figure 5.20).

**Remarks:**

The forward difference employed in Cyclic D-ILC algorithm only penalises the difference in the output/state error at consecutive sample times. Thus, whenever the output has stabilised using an offset type error in which consecutive errors have same magnitude, the Cyclic D-ILC algorithm ceases updating the control input during further repetitions/swings. The learning is therefore interrupted. This phenomenon has been observed when the identical initial conditions are not maintained for each repetition/swing. Additionally, the input at
last time index $M$ does not contribute to generate the output, rather it goes waste. Hence only the first $M - 1$ eigenvalues are of interest. The rank of the matrix product $CB$ determines the number of output variables/states which are controllable. For all outputs to be controllable through Cyclic D-ILC the product $CB$ should be full rank.

5.6 **Parameters for Stable Cyclic PD-ILC Learning**

The stability analysis has been carried out to identify the values of $K_p$ and $K_d$, for which the eigenvalues of the matrix $L_{PD}$ lie inside the unit circle. The region inside Figure 5.21 shows the region inside which the pair of values of $K_p$ and $K_d$ gives stable learning performance.

![Figure 5.21: Stability boundary for Cyclic PD-ILC](image_url)
Any other pair of gains outside this sector shape would cause divergence of the control input from desired. As a consequence output shall diverge as well.

5.7 Summary

In this chapter, the convergence analysis techniques developed in previous chapter have been extended to handle the non-zero initial error case for Cyclic D-ILC algorithm. The use of augmented operator matrix facilitates the analysis. The eigen-analysis has been presented to explain in detail the factors affecting the evolution, non-convergence to zero and convergence rate of the state variables. Further, the use of Cyclic PD-ILC algorithm has been used to solve the problem of offset error faced with Cyclic D-ILC. But the selection of learning gain matrices has to done very carefully so as to avoid lying outside the stability region which causes divergence.
Chapter 6

ILC for Robotics

In this chapter, the Cyclic D-ILC algorithm has been used to generate the input torques for both the hip and ankle joints of compass gait robot. The simulations indicate stable walking patterns for the compass gait robot walking on flat surface as well as slopes. The Cyclic D-ILC algorithm converges to desired values for the input torques resulting in stable walking patterns.

6.1 Cyclic D-ILC for Actuated Compass Gait Robot

In this section, Cyclic D-ILC has been implemented to generate input torques for the compass gait robot walking. Each step is considered as a batch. Every step starts after the impact reset at which the Cyclic D-ILC algorithm is used to calculate the control input update for the hip and ankle torques as follows.

\[
\tau(k+1) = \tau(k) + K_d \{e(k+1) - e(k)\}
\]

(6.1)

where \( \tau \) is the vector of ankle and hip joint torques \( u_1 \) and \( u_2 \), \( k \) is the step number, \( K_d \) is the gain matrix having appropriate dimensions. The error between the desired and new state vectors has been defined as under.

\[
e(k) = x^* - x_0(k)
\]

(6.2)
where \( x_0(k) \) is the new state vector after impact and marks the beginning of the \( k^{th} \) new step.

The torque vector \( \tau \) has been calculated using Cyclic D-ILC algorithm and applied at the beginning of each step like an impulsive input rather than during the whole step. Thus, Cyclic D-ILC provides an inverse-kinematics solution of computing the joint torques from the knowledge of joint motions and past inputs. However, explicit knowledge of the robot dynamics is not required as Cyclic D-ILC algorithm learns to generate the desired input torques. These torques further actuate the joints to the desired joint angles.

### 6.1.1 Simulations of CG Robot Walking on Flat Surface Actuated with Cyclic D-ILC Generated Torques

A compass gait robot having legs of unit length and point mass 5 kg each at the centre of the legs is considered. The hip mass is 10 kg. Starting from the selected initial conditions, i.e. \( x_0 = [0.2, -0.4, -0.8, 2.1] \), the Cyclic D-ILC as per Eq. (6.1) is employed to generate torques at ankle and hip joint for 500 steps in Figure 6.1. The gain matrix \( K_d \) in Eq. (6.3) is selected so that torque update depends on the respective joint angles and angular velocities only. It can be observed that Cyclic D-ILC algorithm has achieved convergence after 20 steps as the torques have reached steady state values.

\[
K_d = \begin{bmatrix}
0.25 & 0 & -0.25 & 0 \\
0 & 0.25 & 0 & 0.25 \\
\end{bmatrix}
\]  \quad (6.3)
Figure 6.1: Torques generated using Cyclic D-ILC on flat surface

Figure 6.2: Phase portrait on flat surface using Cyclic D-ILC
The phase portrait of compass gait robot in Figure 6.2 shows that CG robot has achieved a stable walking pattern or gait.

### 6.1.2 Simulations of CG Walking Downhill with Cyclic D-ILC

The downhill walking of a compass gait robot actuated by Cyclic D-ILC is represented by the phase portrait in Figure 6.4. It has been observed that the torques generated by Cyclic D-ILC have achieved convergence in about 40 steps in Figure 6.3. The positive values of ankle torque indicate that it acts against the downward motion of the robot due to gravity. This way it prevents the robot from falling over.

![Figure 6.3: Torques using Cyclic D-ILC for CG robot walking downhill](image)

Figure 6.3: Torques using Cyclic D-ILC for CG robot walking downhill
Figure 6.4: Limit cycle - CG robot walking on slope

Figure 6.5: Cyclic D-ILC torques for CG using reduced $K_d$
The oscillatory behaviour of Cyclic D-ILC generated ankle torque during the few initial steps can be avoided by scaling down the gain matrix $K_d$. By lowering $K_d$ ten times, the torques have achieved steady state values smoothly as shown in Figure 6.5.

### 6.2 Modified Cyclic PD-ILC for Bipedal Walk

Following update algorithm based on Cyclic PD-ILC has been employed.

$$u(i, j + 1) = u(i, j) + K_p e(i + 1, j) + K_d \{ e(i + 1, j) - e(i, j) \}$$

(6.4)

### 6.3 Simulations for CG Walk Using Modified Cyclic PD-ILC

The compass gait robot model has been discretized at sample time of 0.1 second. Reference trajectory for robot walking on flat surface has been obtained via a carefully tuned PD controller. The compass gait robot has been simulated for 4000 steps using gain matrices as follows.

$$K_p = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \end{bmatrix} \text{ and } K_d = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$
The gain matrix $K_d$ has structure such that it utilizes forward difference errors of both the angle joints and their respective velocities. Effectively, the acceleration term is also incorporated. The Cyclic ILC learns to generate the desired values of the ankle and hip joint torques as shown in Figure 6.6 and Figure 6.7 respectively. As the joint input torques are tracked, the desired joint angles of the compass gait robot are achieved which result in a steady gait as observed from the tracking of angles of stance foot and hip joint in Figure 6.8 and Figure 6.9 respectively.

\[ \text{Ankle torques} \]

\[ \text{time index} \]

\[ \text{last step} \]

Figure 6.6: Tracking ankle joint torque
Figure 6.7: Tracking hip joint torque

Figure 6.8: Tracking stance foot angle
Maximum eigenvalue for operator matrix for ankle input torque error is 0.9951 for which the time constant is about 204. So for 5 time constants there are 1016 iterations or steps. The evolution of ankle input error at component level is shown in Figure 6.10. Convergence rate, although not well-defined as for the pendulum, has been illustrated in Figure 6.11. Maximum eigenvalue of the operator matrix for hip joint input errors is 0.9952. Hence, 5 time constants equal about 1040 iterations or steps. Log-log RMS values of the state, ankle and hip joint input errors in Figure 6.14 show that these errors have minimized after 1000 steps. The stable gait is validated from the phase portraits in Figure 6.15 and Figure 6.16, respectively.
Figure 6.10: Evolution of ankle input torque error

Figure 6.11: Slopes showing rate of ankle input torque error
Figure 6.12: Evolution of Hip Joint torque

Figure 6.13: Slopes showing rate of hip joint error
Figure 6.14: Log-log RMS values of state and input errors

Figure 6.15: Phase portrait of ankle joint at last step
Figure 6.16: Phase portrait of hip joint at last step

Figure 6.17: RMS values with modified Cyclic PD-ILC
6.4 Summary

In this chapter, introduced the model of CG robot and its motion under the action of gravity only. Then the Cyclic D-ILC algorithm under has been applied to the compass gait robot walking on a flat surface. Stable gaits or symmetric walking patterns have been achieved for specific initial conditions. Secondly, the modified Cyclic PD-ILC algorithm has been applied to solve the biped walking problem where each step has been treated as a new batch for the ILC operation. The encouraging results can be extended to robots having higher number of joints and links as well.
Chapter 7

Conclusions

In the end, the thesis is concluded with some general overall comments and remarks about the work presented in earlier Chapters. Some suggestions for future work have been referred as well.

In this thesis, Iterative Learning algorithms have been studied with focus on D-ILC, PD-ILC and Cyclic ILC. Rate of Convergence in terms of input errors have been investigated with simulations showing the long term relation with eigenvalues of the matrix which relates the input error vectors which occur consecutively in time. The mathematical frame-work to analyse the convergence rate has been developed on the basis of conventional zero initial error for D-ILC algorithm. It has been further developed to analyse the convergence analysis of Cyclic PD-ILC algorithm and proved its superiority over Cyclic D-ILC for higher order systems by eigen-analysis in detail. Further, the framework has been extended to analyse the Cyclic ILC scenario which has more complexity as compared to the conventional zero-initial error constraint. Since, errors from the end of iteration are inherited as initial errors for the next consecutive iteration in Cyclic ILC. Zero-initial error constrain has to be relaxed to incorporate the initial state errors. Although the convergence rate analysis has been developed for linear time-invariant systems such as the linearised
damped pendulum, but it can be extended to nonlinear systems as well. These algorithms have been employed to solve compass gait bipedal robot walking problem modelled as a nonlinear system. The simulations have indicated that stability in walking pattern or gait has been achieved.

**Recommendations for Future Work**

- For future work, more elaborate higher order models of bipedal robots can be investigated for walking using Cyclic ILC algorithms.
- Implementation of Iterative Learning algorithms to kneed robot model with feet and toe-off impulse.
- Optimality and energy analysis of the robot while using iterative learning algorithms for bipedal walking.
- Development of optimal and robust learning algorithms for walking as well as running over difficult terrains.
APPENDIX A

Swing Phase Dynamics of Compass gait robot

The equations of motion for the passive compass gait robot are derived using Euler-Lagrange approach. Under the effect of gravity, the difference between Kinetic Energy ($KE$) and Potential Energy ($PE$) gives the relation for the Lagrangian below

$$L(q, \dot{q}) = KE(q, \dot{q}) - PE(q) \quad (A.1)$$

For no external torques acting on the joints, the equations of motion for the joint angles are obtained from the Lagrangian given in Eq. (A.1) as follows

$$\frac{d}{dt} \left( \frac{\partial L(q, \dot{q})}{\partial \dot{q}} \right) - \frac{\partial L(q, \dot{q})}{\partial q} = 0 \quad (A.2)$$

Using Eq. (A.1), the Lagrangian is substituted with $KE$ and $PE$ in Eq. (A.2).

$$\frac{d}{dt} \left( \frac{\partial KE(q, \dot{q})}{\partial \dot{q}} \right) - \frac{\partial KE(q, \dot{q})}{\partial q} + \frac{dPE(q)}{dq} = 0 \quad (A.3)$$

The $PE$ and $KE$ for compass gait robot are found as

$$PE(q) = \left( m(a + l) + m_hl \right) g \cos(q_1) - mgb \cos(q_1 + q_2) \quad (A.4)$$

$$KE(q) = \frac{1}{2} \dot{q}^T M(q) \dot{q} \quad (A.5)$$

where the inertia matrix $M(q)$ is given below
\[ M(q) = \begin{bmatrix} m_l^2 + m\left(l^2 + a^2 + b^2 - 2bl\cos(q_2)\right) & m\left(b^2 - bl\cos(q_2)\right) \\ m\left(b^2 - bl\cos(q_2)\right) & mb^2 \end{bmatrix} \tag{A.6} \]

Here, the inertia matrix \( M(q) \) independent of \( q_1 \) is a function of interleg angle \( q_2 \) and masses only. The derivatives of \( PE \) and \( KE \) for Euler-Lagrange expression are as follows

\[
\frac{dPE(q)}{dq} = \begin{bmatrix} -\left(m(a + l) + m_h l\right)g\sin(q_1) + mb\sin(q_1 + q_2) \\ mb\sin(q_1 + q_2) \end{bmatrix} \tag{A.7} \]

\[
\frac{d}{dt}\left(\frac{\partial KE(q, \dot{q})}{\partial \dot{q}}\right) - \frac{\partial KE(q, \dot{q})}{\partial q} = M(q)\ddot{q} + \frac{dM(q)}{dt}\dot{q} - \frac{1}{2}\frac{\partial (\dot{q}^TM(q)\dot{q})}{\partial q} \tag{A.8} \]

The derivatives in Eq. (A.8) are shown expanded below

\[
\frac{dM(q)}{dt} = \begin{bmatrix} m\{\sin(q_2)\}\dot{q}_2 & \frac{1}{2}m\{\sin(q_2)\}\dot{q}_2 \\ \frac{1}{2}m\{\sin(q_2)\}\dot{q}_2 & 0 \end{bmatrix} \tag{A.9} \]

\[
\frac{\partial (\dot{q}^TM(q)\dot{q})}{\partial q} = \begin{bmatrix} 0 \\ m\{\sin(q_2)\} \times \left((\dot{q}_1)^2 + \dot{q}_1\dot{q}_2\right) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ m\{\sin(q_2)\}\dot{q}_1 + m\{\sin(q_2)\}\dot{q}_2 & 0 \end{bmatrix} \tag{A.10} \]

Substituting the derivatives from Eq. (A.7) and Eq. (A.8) in Euler-Lagrange expression Eq. (A.3), the following 2nd order, nonlinear equation describes the dynamics of the compass gait robot

178
\[ M(q) \ddot{q} + C(q, \dot{q}) \dot{q} + G(q) = 0 \]  
(A.11)

where

\[
C(q, \dot{q}) = \frac{dM(q)}{dt} \dot{q} - \frac{\partial KE(q, \dot{q})}{\partial q} = 
\begin{bmatrix}
2mb\dot{L}_2 \sin(q_2) & mb\dot{L}_2 \sin(q_2) \\
-mb\dot{L}_1 \sin(q_2) & 0
\end{bmatrix}
\]  
(A.12)

and

\[
G(q) = 
\begin{bmatrix}
-(m(a + l) + m_h l)g \sin(q_1) + mb \sin(q_1 + q_2) \\
b \sin(q_1 + q_2)
\end{bmatrix}
\]  
(A.13)

**Impact Reset during Stance Phase**

At impact, the reference point for the robot changes to the new stance foot. Use + and – superscripts to represent pre- and post- impact conditions respectively. The relationships for both the pre- and post-impacts are as follows

\[
q_2 = -2q_1 \\
q_1^+ = -q_1^- 
\]  
(A.14)

The KE is decreased due to impact. However, assuming a perfect impact, the angular momentum remains constant. The abrupt change in joint velocities is calculated from the conservation of pre- and post-impact momentums as follows

\[
M^-(q) \dot{q}^- = M^+(q) \dot{q}^+ 
\]  
(A.15)
Using the relationships between joint angles given in Eq. (A.14) at impact, Eq. (A.15) has angular momentums as functions of inter-leg angle $q_2$ only as given below

$$M^-(q_2) \dot{q}^- = M^+(q_2) \dot{q}^+ \quad (A.16)$$

The pre- and post-impact momentums are described as follows

$$M^-(q_2) = \begin{bmatrix} -\frac{m}{4} & 0 \\ m_h \cos(q_2) - \frac{m}{2} \{1 - 2 \cos(q_2)\} & -\frac{m}{4} \end{bmatrix}$$

$$M^+(q_2) = \begin{bmatrix} \frac{m}{4} \{1 - 2 \cos(q_2)\} & \frac{m}{4} \\ m_h + m \left\{\frac{3}{2} - \cos(q_2)\right\} & \frac{m}{4} \{1 - 2 \cos(q_2)\} \end{bmatrix} \quad (A.17)$$

The post-impact velocities are found from Eq. (A.16) as under

$$\dot{q}^+ = (M^+(q_2))^{-1} M^-(q_2) \dot{q}^- = R \dot{q}^- \quad (A.18)$$

where $R$, the reset matrix for joint angular velocities is given by

$$R = \frac{1}{4m_h + 3m - 2m \cos(2q_2)}$$

$$\times \begin{bmatrix} -m + 2(m + 2m_h) \cos(q_2) & -m \\ 4(m + m_h) \{\cos(2q_2) - \cos(q_2)\} & m \{1 - 2 \cos(q_2)\} \end{bmatrix} \quad (A.19)$$
APPENDIX B

For a difference inequality such as

\[ z(i + 1) \leq \beta(i) + h z(i) \]  

(B.1)

where \( z(.) \) and \( \beta(.) \) are scalar functions of \( i \geq 0 \) and \( h \) is a positive constant. So, for \( i \geq 1 \), the following relation holds:

\[ z(i) \leq \sum_{j=0}^{i-1} h^{i-1-j} \beta(j) + h^i z(0) \]  

(B.2)

The Eq. (B.2) gives the upper bound.
REFERENCES


47. Owens, D.H. Iterative learning control - Convergence using high gain feedback. in Proc. 31st IEEE Conf. on Decision and Control. 1992. Tucson, AZ, USA.


