A Robotic Soil Excavator for Truck Loading

By

Ehab M. Iqnaibi

Abdullah Abu Shokor

Laith Alsayed Ahmad

Supervisor

Prof. Dr. Karim Tahboub

Submitted to the College of Engineering

In partial fulfillment of the requirements for the

Bachelor degree in Mechatronics Engineering

Palestine Polytechnic University

August 2, 2021

`Palestine Polytechnic University

Collage of Engineering

Mechanical Engineering Department

Hebron – Palestine

A Robotic Soil Excavator for Truck Loading

Project Team:

Ehab M. Iqnaibi

Abdullah Abu Shokor

Laith Alsayed Ahmad

Submitted to the Collage of Engineering

In partial fulfillment of the requirements for the

Bachelor degree in Mechatronics Engineering.

Supervisor Signature

..........................[illegible]..........

Testing Committee Signature

....

Chair of the Department Signature

.......[illegible]......[illegible]..................

Abstract

Hydraulic soil excavators are commonly used in construction sectors and soil removal which are distinguished by high power capabilities and good performance. In this graduation project and for the purpose of automating soil excavation, a robotic soil excavator prototype is developed. It consists of four degree of freedom (4 DoF) with four rigid links connected by four revolute joints. The soil removal requires an expert operator to perform such tasks and consumes time and human power. Consequently, the prototype is conceptualized to achieve a semi-autonomous motion control to dig soil from a given excavation point, carry soil, and finally throw it to a truck loading within a given time duration. Furthermore, it can be operated by an operator remotely.

Developing this robotic excavator system involves the machining of mechanical frames, selection of proper electromechanical components, integration of all the parts together. Kinematics and dynamics models are derived for the obtained design to analyze and plan motions, find necessary driving torques and accompanying reaction forces, and to serve as the core of several model-based motion control algorithms. These algorithms are developed and tested using MATLAB and Simulink software packages.

The robotic excavator prototype is built following procedures of material selection, design of mechanical structure, and analysis using SOLIDWORKS program. Following that, mechanical components are assembled and tested to satisfy desired functions and given specifications. Each robot joint is equipped with a position sensor and a DC-motor that is controlled via a driver operated in torque mode.

The developed robotic soil excavator is considered as an embedded system as it has its own information processing and control unit on board. For this, Raspberry Pi microcomputer is employed to implement a centralized control algorithm that is developed and coded in Python. Experiments show that a trajectory can be generated either through direct or inverse kinematics and can be tracked within acceptable accuracy. Specifically, PD control with gravity compensation is tested in depth.

ROS ideas and concepts are examined and implemented. It turned out as discussed in this report that, given the microcontroller chosen with its imposed limitations, relying fully on ROS is not straightforward and requires developing efforts and time beyond the scope of this project.

Dedication

We dedicate this project to Allah Almighty. It is with genuine gratitude and warm regard that we dedicate this research project to our family and friends. A special feeling of gratitude to our loving parents, whose words of encouragement and push for tenacity ring in my ears.

Acknowledgment

We wish to express our sincere gratitude to our supervisor Prof. Dr. Karim Tahboub for providing us all support and guidance, whose insightful leadership and knowledge benefited us to complete this project successfully. We are respectful of your continuous support and presence whenever needed.

We would like to express our deep appreciation and indebtedness to our teachers and supervisors for their endless support, and kindness.

Last but not the least, we would like to thank everyone who is involved in the project and helped us with their suggestions to make the project better.

Contents

Abstract i
Dedication ii
Acknowledgment iii
Chapter One: Introduction 1
1.1 Introduction 1
1.2 Recognition of the need 1
1.3 Literature review and existing solutions 2
1.3.1 A Robotic Excavator for Autonomous Truck Loading 2
1.3.2 Learning Task-Based Instructional Policy for Excavator-Like Robots 3
1.3.3 Dynamic modeling of the front structure of an excavator 3
1.3.4 Design, Implementation and Digital Control of a Robotic Arm 4
Chapter Two: Conceptual Design and Functional Specifications 5
2.1 Introduction 5
2.2 Conceptual design schematic 6
2.3 Components and functional specifications 7
Chapter Three: Kinematics and Dynamics Analysis of The Robotic Manipulator 9
3.1 Introduction 9
3.2 Kinematics model 9
3.2.1 Robot forward kinematics 9
3.2.2 Robot inverse kinematics 11
3.2.3 Workspace 14
3.2.4 Robot differential kinematics 14
3.3 Dynamic model 17
Chapter Four: Design of Structure and Mechanical Components 26
4.1 Introduction 26
4.2 Material selection and analysis 26
4.2.1 Introduction 26
4.2.2 Material selection and analysis 27
4.3 Mechanical components of structure 32
4.4 Structure and load analysis through SOLIDWORKS 36

Chapter Five: Selection of Electrical Parts and Components ... 39
5.1 Introduction ... 39
5.2 Comparison and selection of system actuators ... 39
5.2.1 Comparison between actuators ... 39
5.2.2 Selection of actuators ... 42
5.3 Electrical Parts ... 44
5.3.1 Robot motors ... 44
5.3.2 Motor drivers ... 45
5.3.3 Sensors ... 45
5.3.4 Human machine interface ... 46
5.3.5 Power source ... 47
5.4 Components selection ... 47
5.4.1 Motors selection ... 47
5.4.2 Driver selection ... 50
5.4.3 Sensor selection ... 52
5.4.4 Human machine interface ... 55
5.4.5 Power source ... 56
Chapter Six: Information Processing ... 58
6.1 Introduction ... 58
6.2 Software and information processing ... 58
6.3 Processing unit ... 60
6.4 Hardware connections and interfacing between components ... 61
6.4.1 Interfacing protocols ... 61
6.4.2 Hardware connection (GPIO pins connection) ... 64
Chapter Seven: Robot Motion Control ... 66
7.1 Introduction ... 66
7.2 Motion planning ... 66
7.2.1 Introduction ... 66
7.2.2 Operational space trajectory ... 67
7.2.3 Trajectory translating to joint space ... 69
7.3 Motion control in joint space ... 72
7.3.1 Introduction ... 72

7.3.2 Decentralized control ... 73
7.3.3 Position and velocity feedback control (PID controller) ... 74
7.3.4 I-PD control (State-feedback with integral control) ... 79
7.3.5 Centralized control ... 84
7.3.6 Inverse dynamics ... 84
7.3.7 PD with Gravity Compensation ... 89
7.4 Motion control in operational space ... 92
7.4.1 Introduction ... 92
7.4.2 PD control with gravity compensation ... 92
7.5 Conclusion ... 96
Chapter Eight: Prototype Building, Experiments and Results ... 97
8.1 Introduction ... 97
8.2 Prototype structure and system components ... 97
8.2.1 Build robotic soil excavator prototype ... 97
8.2.2 Electrical components ... 101
8.3 Information processing techniques and software capabilities ... 104
8.3.1 MATLAB and Simulink software ... 104
8.3.2 Robot Operating System - ROS ... 105
8.3.3 Programming control algorithms using stand-alone code ... 113
8.4 Motion control experiments and results ... 113
8.4.1 Introduction ... 113
8.4.2 PD with gravity compensator ... 114
8.5 Conclusion ... 119
Chapter nine: Conclusion and Future Work ... 120
9.1 Conclusion ... 120
9.2 Future work ... 121
Appendices ... 122
A. MATLAB and SIMULINK ... 122
A.1 Forward kinematics derivation - verification using robotics toolbox: ... 122
A.2 Example: verification for forward kinematics with inverse kinematics: ... 123
A.3 Using MATLAB with per-hand derived code for verification of inverse kinematics:. 124
A.4 Verification of the jacobian analytically: ... 125

A.5 Using MATLAB robotics toolbox (verification numerically): 127
A.6 Jacobian's determinate and singularities analysis:.. 129
References 131

List of figures

Figure 1. 1: Typical dig for truck loading.. 2
Figure 1. 2: Truck is loaded after six passes.. 2
Figure 1. 3: The robot's description.. 3
Figure 1. 4: The generalized forces of the front structure... 4
Figure 1. 5: The Robotic arm... 4

Figure 2. 1: An intelligent Excavator working in real workspace. 5
Figure2. 2: Conceptual Design Schematic.. 6

Figure 3. 1: Coordinate diagram of the manipulator in side view.. 9
Figure 3. 2: The reachable end-effector poses 14
Figure 3. 3: Coordinates diagram of the manipulator in side view....................................... 18

Figure 4. 1: Schematic diagram of an excavator.. 27
Figure 4. 2: Equivalent robot beam.. 28
Figure 4. 3: Stresses on the desired section are superposition of shear stress due to V and normal stress due to M 29
Figure 4. 4: Hollow square cross-section... 31
Figure 4. 5: HITACHI model EX21. 32
Figure 4. 6: Fixed Base 33
Figure 4. 7: base... 33
Figure 4. 8: Boom part 34
Figure 4. 9: Arm part 34
Figure 4. 10: Bucket part 35
Figure 4. 11: Shafts of motors: (a) for motor-1, (b) for motor- 2, (c) for motor-3, and (d) for motor-4 35
Figure 4. 12: Excavator prototype (Part assembly) 36
Figure 4. 13: Result using SOLIDWORKS for 1060 Alloy (a) deflection, (b) Factor of Safety (FOS) and (c) Von-mess stress.. 38

Figure 5. 1: Hydraulic linear actuators. 40
Figure 5. 2: Many types of electrical actuators... 41
Figure 5. 3: Forces acting on the manipulator 44
Figure 5. 4: The selected DC brush motor for joints 1 and 2... 47
Figure 5. 5: The selected DC brush motor for joint 3 48

Figure 5. 6: The selected DC servo-motor for joint 4 49
Figure 5. 7: SOLO UNO Driver with described ports 50
Figure 5. 8: Open-loop control through PWM signal to S/T terminal 51
Figure 5. 9: l293D H-bridge with terminal names. 52
Figure 5. 10: The used rotary encoders: 400-pulse resolution (left), 2048-pulse resolution (right) 53
Figure 5. 11: 32-bit encoder reader 54
Figure 5. 12: limit Switch. 54
Figure 5. 13: Raspberry Pi screen. 55
Figure 5. 14: Power supply 56

Figure 6. 1: Raspberry Pi 4 microcomputer 59
Figure 6. 2: Mechatronics system integration [12] 60
Figure 6. 3: GPIO pins for Raspberry Pi 4 – model B 61
Figure 6. 4: RPi-SPI port connected to many devices with multiple select lines (CS) [13] 62
Figure 6. 5 : RPi-UART port connected to a device [13] 63
Figure 6. 6: Hardware connectio 65

Figure 7. 1: The desired end-effector trajectory; Px(t), Py(t), Pz(t). 69
Figure 7. 2: corresponding Joint timing laws; q1(t), q2(t), q3(t). 70
Figure 7. 3: Velocity and acceleration curves for the joint motions. 71
Figure 7. 4: straight line motion in workspace graphically. 71
Figure 7. 5: General scheme of joint space control. 72
Figure 7. 6: Block scheme of the system of manipulator with the drive 73
Figure 7. 7: Block scheme of general independent joint control. 75
Figure 7. 8: equivalent control scheme of PID type. 75
Figure 7. 9: SIMULINK block diagram for PID control system with feedforward compensation. 76
Figure 7. 10:(a), (b) and (c): q(t) vs.qd(t)- PID 77
Figure 7. 11: The error between q(t) and qd(t) - PID. 78
Figure 7. 12: Torque required at each joint- PID 78
Figure 7. 13: state-feedback control with integral control (I-PD). 79
Figure 7. 14: Block scheme of I-PD controller 81
Figure 7. 15:(a), (b) and(c) q(t) vs. qd(t), I-PD. 82
Figure 7. 16: The error between q(t) and qd(t), I-PD. 83
Figure 7. 17: Torque at each joint, I-PD. 84
Figure 7. 18: Block scheme of joint space inverse dynamics control. 85
Figure 7. 19: Block scheme of inverse dynamics control. 86
Figure 7. 20: q(t) vs. qd(t), Inverse dynamics. 87
Figure 7. 21: The error between q(t) and qd(t), inverse dynamics. 88
Figure 7. 22: Torque required at each joint- Inverse Dynamics. 88
Figure 7. 23: scheme of joint space PD control with gravity compensation. 89
Figure 7. 24: PD with gravity compensation using MATLAB SIMULINK 89

Figure 7. 25: q(t) vs. qd(t), PD with Gravity 90
Figure 7. 26: Resulting error for motion tracking – PD with Gravity 91
Figure 7. 27: Joint efforts – PD with Gravity 91
Figure 7. 28: Block scheme of Jacobian transpose control. 92
Figure 7. 29: Block scheme of operational space PD control with gravity compensation. 93
Figure 7. 30: Block diagram of PD with Gravity in operational space 93
Figure 7. 31:(a),(b) and (c) xd(t) vs. xe(t), PDGC. 95
Figure 7. 32: Error between xd and xe. 95
Figure 7. 33: Torques of joints. 95

Figure 8. 1: The base of Excavator manipulator. 98
Figure 8. 2: Boom Link-2 of manipulator. 98
Figure 8. 3: Arm Link-3 of manipulator. 98
Figure 8. 4: Bucket end-effector and shaft printed of the manipulator. 99
Figure 8. 5: The shafts of motors. 99
Figure 8. 6 Prototype of Excavator manipulator. 100
Figure 8. 7: An arm with a suspended mass 101
Figure 8. 8: Backlash of motor gearbox 104
Figure 8. 9: Topic Message Communication. [16] 105
Figure 8. 10: The developed ROS package using Topics 106
Figure 8. 11: Service Message Communication. [15] 107
Figure 8. 12: The developed ROS package using Services 107
Figure 8. 13: URDF graph 110
Figure 8. 14: Communication scheme with move_group node. [15] 110
Figure 8. 15: Planning a specific motion in MoveIt 111
Figure 8. 16: Nodes and topics of Movelt 111
Figure 8. 17: Planning the manipulator by moveIt with RViz 112
Figure 8. 18: Flow chart of control algorithm 114
Figure 8. 19: generated joint trajectories via inverse kinematics 115
Figure 8. 20: Joint states of PD with Gravity motion control 115
Figure 8. 21: Error evolution for the end-effector position 116
Figure 8. 22: Resulted control efforts of PD with Gravity control 116
Figure 8. 23: Generated joint trajectories in joint space for multi-tasks 117
Figure 8. 24: Responses for executing many tasks in joint space, PD with Gravity control 117
Figure 8. 25: Error evolution for each joint position 118
Figure 8. 26: Required torques for executing many motions 118

Figure A. 1: MATLAB live script of the homogenous transformation matrix for each joint. ... 122
Figure A. 2: MATLAB live script of the forward kinematics. 123
Figure A. 3: Forward kinematics for the given joint variables. 123
Figure A. 4: Forward kinematics verification using MATLAB toolbox 124
Figure A. 5: Re-fined the forward kinematics for Frame (3) wrt Frame (1) 124
Figure A. 6: The verified inverse kinematics 125

Figure A. 7: Forward kinematics using MATLAB 125
Figure A. 8: Jacobian using MATLAB 126
Figure A. 9: Substituting numbers instead of variables 127
Figure A. 10: Finding Jacobian numerically using Robotics toolbox for the specified variables 128
Figure A. 11: Finding Jacobian 1 using MATLAB 129
Figure A. 12: Finding Jacobian 2 using MATLAB 130

List of Table

Table 3. 1: The D-H Coordinates Parametric 10

Table 4. 1: mechanical properties for steel and aluminum alloy. 28
Table 4. 2: Find maximum deflection between two materials. 31

Table 5. 1: comparison between some types of DC motors. 43
Table 5. 2: It contains the parameters of the chosen DC motor for joints 1 and 2. 48
Table 5. 3: It contains the parameters of the chosen DC motor for joint 3. 48
Table 5. 4: It contains the parameters of the chosen DC servo-motor for joint 4. 49
Table 5. 5: It contains the specifications of the selected H-bridge. 52
Table 5. 6: It shows the specifications of the 400-pulse encoders. 53
Table 5. 7: It shows the specifications of the 2048-pulse encoder. 53
Table 5. 8: Shows the specifications of the 32-bit encoder counter. 54
Table 5. 9: limit switch sensor specifications. 54
Table 5. 10: RPI screen specifications. 55
Table 5. 11: System electrical components with current consumption 56
Table 5. 12: Shows the selected power supply specifications. 57

Table 8. 1: Specifications of the manipulator mechanical parts. 100

Chapter One: Introduction

1.1 Introduction

The robotic soil excavator is a manipulator that consists of a series of four links (base, boom, arm, and bucket) connected by means of revolute joints which are actuated by electrical motors. The kinematic chain starts with a stationary base while its end consists of the bucket. The robotic excavator should excavate soil from a given point and transmit it to a truckload destination, and it can be controlled by manual operation or semi-autonomously.

In this project, we propose to develop an excavator prototype. Field of study involves the mechatronics design, includes; structure design, selection of electrical components, the appropriate microprocessor and software, robotic modeling and control dealing with robot's kinematics and dynamic model. After that a teleoperation technique will be utilized as a remotely control strategy.

The MATLAB and SIMULINK software packages are used for development, analysis and simulation of the robotic excavator dynamic model and the implementation of nonlinear control methods. Raspberry Pi microcomputer is the main system controller that forms an embedded system with purpose of developing, implementing the control algorithms and interfacing with hardware components.

1.2 Recognition of the need

In this section, we define the need of the robotic soil excavator design and the requirements to be met. The following categories encompass the types of user requirements for excavator design enable to load the soil into truck for haulage storage areas.

1. Able to be loaded with $0.5Kg$ and capacity of the bucket is 0.35L
2. Structure that achieves the light weight with high strength (payload ratio 4:1).
3. Average speed of 20s to complete one task.
4. The robot software includes fault diagnosis and monitoring to be displayed on HMI.
5. Avoid the risk of robot self-collisions or obstacles within the environment.
6. Resolution of ± 3mm, and powerful tracking performance for trajectory motion.
7. Upgradability in design and software (addition of vision and mobility).
8. The robot is friendly to use for the operator, with existence of (HMI, Remote control, etc.).
9. The total cost must be acceptable (<2000$).

1.3 Literature review and existing solutions

This section is a compilation of literature review performed for this project. Literature review includes the previous studies about the topic with alternative solutions. Here some literatures that taken into consideration.

1.3.1 A Robotic Excavator for Autonomous Truck Loading

ANTHONY STENTZ in [1], present a system that completely automates the truck-loading task the excavator's software decides where to dig in the soil, where to dump in the truck, and how to quickly move between these points while detecting and stopping for obstacles. And the results of this paper that were able to the typical loading times are 15 to 20 seconds per pass, with six passes needed to load the truck as shown Figure 1. 1, Figure 1. 2. The loading times are close to our requirement of 20s time for one task.

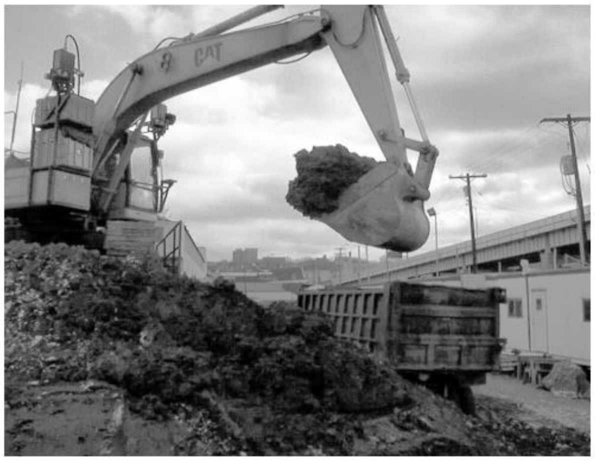


Figure 1. 1: Typical dig for truck loading.

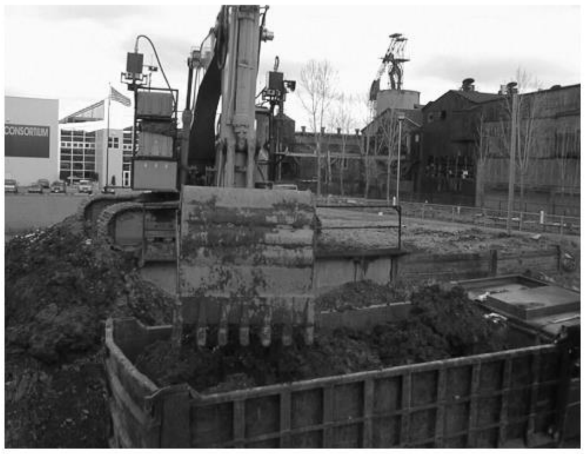


Figure 1. 2: Truck is loaded after six passes.

1.3.2 Learning Task-Based Instructional Policy for Excavator-Like Robots

Harshal Maske et.al. in [2] present a learning instructional policy model to develop robots that learn from expert operators, and generate instructions to assist and guide novice operators, to perform complex construction task. were performed on a 1/14th scaled 345D Wedico excavator model, a 4 DOF hydraulic robotic arm manipulator, controlled by a radio transmitter, see Figure 1. 3. Exhaustive experiments on using instructional policy to guide novice operators demonstrate that the operators guided by the robot were able to perform a complex task more efficiently in comparison to those learning by observing the experts.

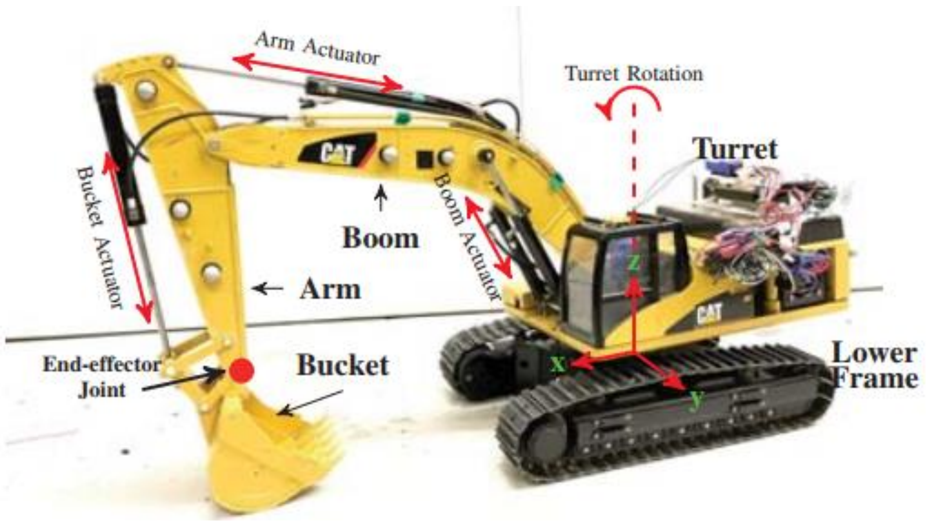


Figure 1. 3: The robot's description

1.3.3 Dynamic modeling of the front structure of an excavator

In [3], the authors present a new mathematical model of 4 DOF describing the dynamic behavior of the front structure of the excavator. The forces of coupling effect are involved. They followed the kinematics analysis of the manipulator. Control simulation is adopted that gives results verify the effects of the addition of the coupling forces that should be taken into consideration for accurate dynamics analysis. The coupling effects shows varying in power consumption that when considering it in control systems, the system becomes more efficient and less power is consumed. This work is useful to verify our dynamic model derivation and motion analysis, see Figure 1. 4.

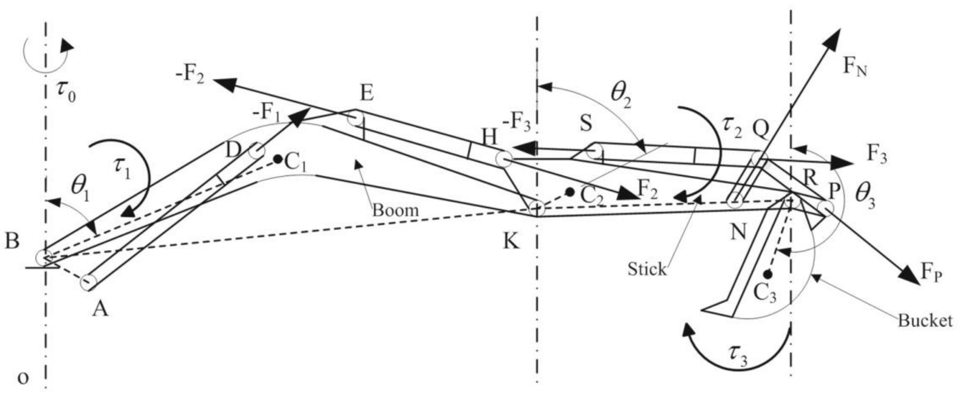


Figure 1. 4: The generalized forces of the front structure.

1.3.4 Design, Implementation and Digital Control of a Robotic Arm

In [4], they build 4DOF robot arm see Figure 1. 5 with low cost, a MATLAB with DAQ card is used to design and test the PID controller of the joint positions. Also, they used a digital camera and MATLAB image processing toolbox. Based on the visual feedback information, control signals are calculated and sent to the arm. The robotic Arm can identify different objects located in its workspace according to their shape and color and places the objects in a certain location. This thesis is very useful for our work that it introduces the most of the topics that we cover.

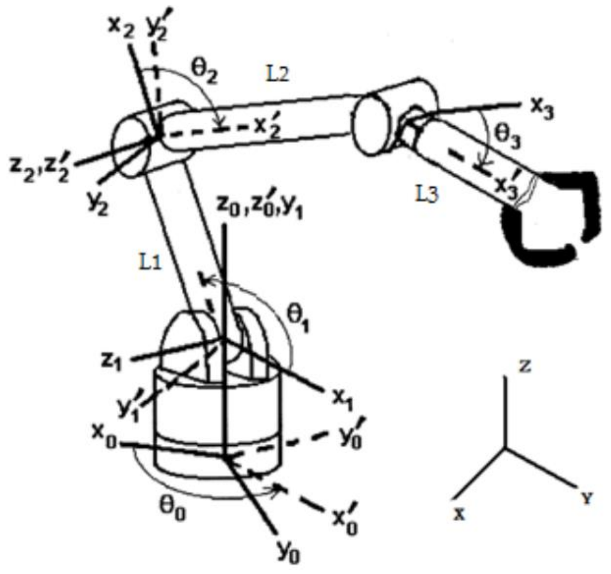


Figure 1. 5: The Robotic arm.

Chapter Two: Conceptual Design and Functional Specifications

2.1 Introduction

This section shows the conceptual design of the robotic soil excavator, consisting of the subsystems and components, functions, relation between elements and functional specifications. A robotic excavator is an Anthropomorphic Manipulator Arm that have four joint Actuators of synchronized motion. The manipulator design resembling the excavator's arm. Its structure consists of structurally rigid links coupled by revolute joints.

The manipulator cooperation with other electromechanical parts for automated or semi-automated equipment to achieve tasks, the robot is designed, built, and controlled via the Raspberry Pi that deployed with a program or algorithm. The robot consists of feedback sensing elements, such as encoders and limit switches. A teleoperation technique will be implemented with Remote control technique for manual control, and the excavator controller should perform many predefined motion tasks. Figure 2. 1 shows an Industrial Soil-Hydraulic Excavator that excavates a soil pile and moving it to a truck.

Figure 2. 1: An intelligent Excavator working in real workspace.

2.2 Conceptual design schematic

Figure2. 2 describes the system conceptual design.

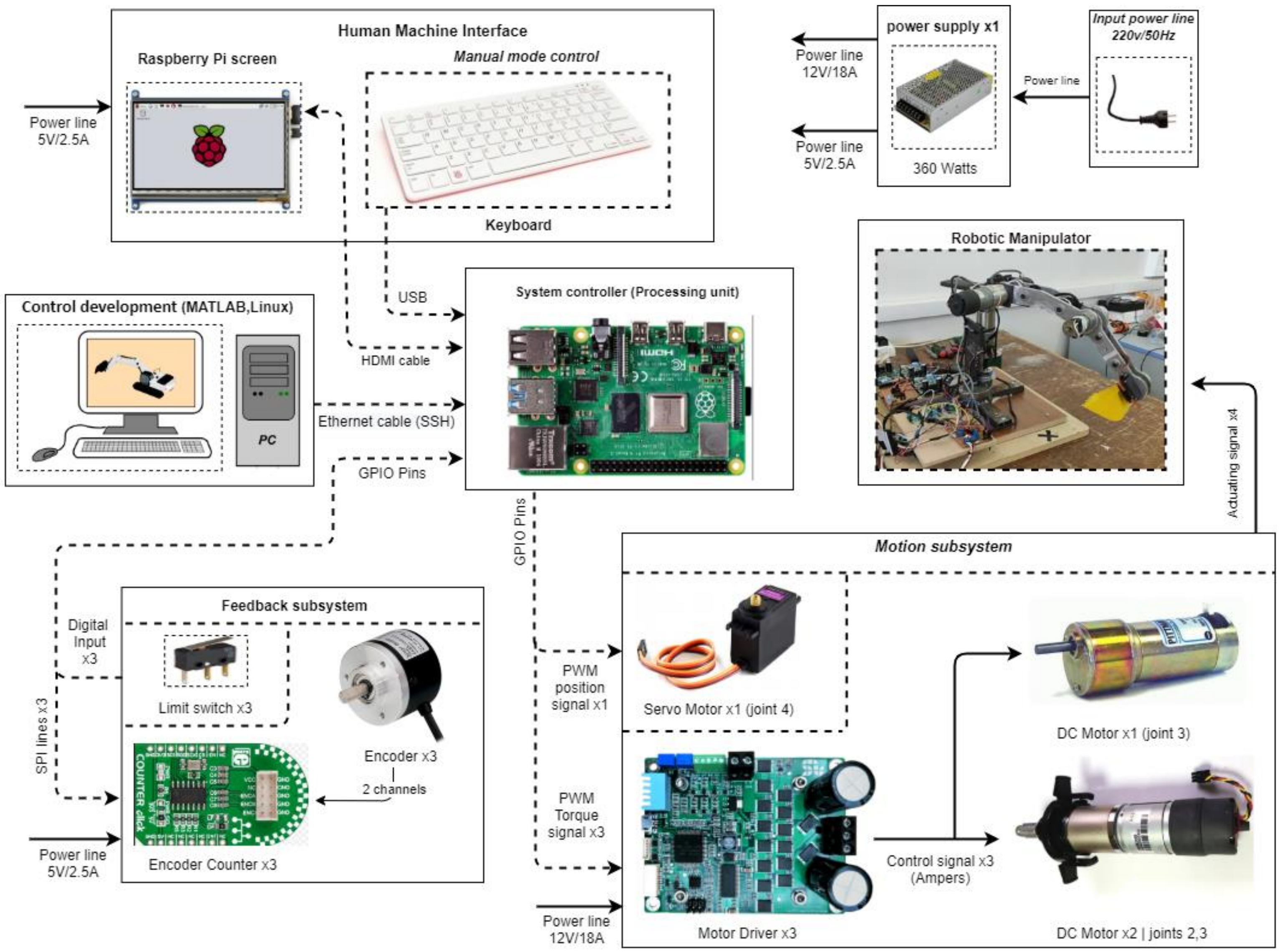


Figure2. 2: Conceptual Design Schematic

2.3 Components and functional specifications

Robotic manipulator:

The manipulator excavates, carries and throws the soil from an excavation point to a truck loading. The mechanical structure consists of the following components:

1. Four links: base, boom, arm and bucket. The base lifts the manipulator and rotates the other three links about the vertical axis. Boom connects the base to the arm. The arm lifts the bucket. The bucket is the end-effector that digs and carries the soil. The motion of the last three links is in a vertical plane. The four links form an open chain manipulator.
2. Four joints: the joints connect the links with each other and they are all revolute.
3. Coupling parts: shaft for each joint.

System controller:

The system controller will be used mainly in controlling and interfacing the other subsystems. Raspberry Pi (RPi) microcomputer is chosen which is powerful in processing and data analysis, and suitable for intelligent systems with huge computational capacity. Most of RPi GPIO pins are connected to the other subsystems as feedback, motion, manual-mode and HMI subsystems. The RPi requires power of 5.1V and 3A that is supplied by a special power converter.

RPi has all the required functional specifications such as, 4 PWM ports for driving motors, 3 SPI ports for encoder counters, powerful performance, 4 USB and 2 HDMI ports.

Control development:

MATLAB software with PC is used as control development interface at the first levels. For further work, the RPi is considered as the stand-alone system controller. This controller is connected with PC via Ethernet that allows data transmission through SSH protocol, it provides access to the Linux Terminal in RPi for programming and development.

Feedback subsystem:

The excavator robot has three encoders to read the position and direction of each joint, each encoder has an independent counter to count its pulses, and the reading will be transferred to RPi through SPI protocol. The first three joints have one limit switch for each to indicate the lower limit of joint movement (Protection and homing). The system components as the following:

1. Three Encoders: each one has 4 terminals, are: channel A, B, 5Vcc and GND.
2. Three limit switches: each one has two terminals, are 5Vcc and OUTPUT (signal to RPi).

Encoders should satisfy the functional specification of $3mm$ resolution of the end-effector.

Motion Subsystem:

The motion subsystem contains the following components:

1. Three DC-motors: for the first three joints, each motor receives control signal (current) from its driver to actuate its joint with the required torque.
2. One Servo-motor: for the fourth joint, it receives PWM signal directly from RPi for position control.
3. Three motor drivers: each driver receives PWM (torque) signal from RPi output pins.

Motor selection should provide the maximum needed torque on each joint, and gearbox has to satisfy the $3mm$ resolution with a minimum backlash angle.

Human Machine Interface:

This subsystem allows the interaction between the operator and the system controller. It consists of RPi touch screen and keyboard. These components have the following functions:

1. RPi touch screen: has a main role of control development and resources management. It is used to show responses, such as joint positions, torques, errors and allows user to command and monitor. It is connected by HDMI cable to the RPi.
2. Bluetooth keyboard: it is used for teleoperation mode to command and rotate each joint independently to achieve a specific motion or any command. It is connected to RPi via Bluetooth connectivity with USB adapter.

Power supply

A suitable PC power source is selected to satisfy the required power consumption. It transforms input of 220VAC/50HZ to 12VDC/18A and 5VDC/2.5A. The power supply can feed all the subsystems with the required power.

All specifications and information processing of these components are listed in chapter 5 and 6 respectively.

Chapter Three: Kinematics and Dynamics Analysis of The Robotic Manipulator

3.1 Introduction

In this chapter, kinematics and dynamics of the excavator robot is derived and described. Kinematics analysis studies the motion in order to manipulate an object in space without implementing the acting forces, and analysis of manipulator's structure, so here the goal is to define the end-effector pose and its velocity as functions of the joint-variables and the angular velocities of the manipulator's revolute joints respectively. Finally, the chapter ends with description of the relationship between the joint actuator torques and the motion of the structure with derivation of the dynamic model.

Book [5] has been followed for most derivations in this chapter.

3.2 Kinematics model

3.2.1 Robot forward kinematics

For the 4-DOF excavator manipulator, the forward kinematics is derived. It describes the end-effector pose of frame O_4 in terms of chosen joint angles with respect to the base frame O_0, as shown in Figure 3. 1.

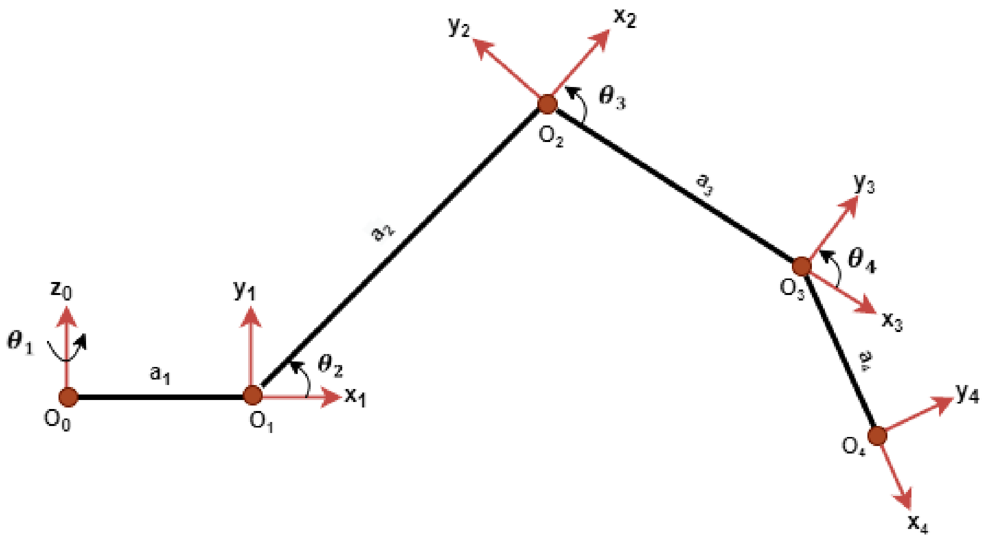


Figure 3. 1: Coordinate diagram of the manipulator in side view.

The D-H Coordinates Parametric table can be defined as shown:

Table 3. 1: The D-H Coordinates Parametric

Link No.	θ_i	d_i	α_i	a_i
1	θ_1	0	$\pi/2$	a_1
2	θ_2	0	0	a_2
3	θ_3	0	0	a_3
4	θ_4	0	0	a_4

Note:

s_i: refers to sin (θ_i), c_i: refers to cos (θ_i)

$s_{ijk} = \sin(\theta_i + \theta_j + \theta_k)$, $c_{ijk} = \cos(\theta_i + \theta_j + \theta_k)$

Finding the Homogenous Transformation matrix for each joint, using equation (3.1) [5],

$$A_i^{i-1}(q_i) = A_{i'}^{i-1} A_i^{i'} = \begin{bmatrix} c_{\vartheta_i} & -s_{\vartheta_i} c_{\alpha_i} & s_{\vartheta_i} s_{\alpha_i} & a_i c_{\vartheta_i} \\ s_{\vartheta_i} & c_{\vartheta_i} c_{\alpha_i} & -c_{\vartheta_i} s_{\alpha_i} & a_i s_{\vartheta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

The first joint is revolute in z_0-axis, the homogenous matrix (1) becomes:

$$A_1^0 = \begin{bmatrix} c1 & 0 & s1 & c1a_1 \\ s1 & 0 & -c1 & s1a_1 \\ 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

Equation (3.2) shows that the projection of $\boldsymbol{y1}$-axis in frame O_0 as follows:

Projection in x_0, y_0 is 0 and in z_0 is **1**.

Since the other three joints (O_1, O_2 and O_3) are revolute in the same plane in z_{i+1} axis, the homogeneous transformation matrix defined in (3.1) has the same structure for each joint, yields

$$A_i^{i-1} = \begin{bmatrix} c_i & -s_i & 0 & a_i c_i \\ s_i & c_i & 0 & a_i s_i \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, i = 2, 3, 4. \tag{3.3}$$

By multiplying the matrix (3.2) with the other three matrices in (3.3), resulting the homogenous transformation matrix (3.4) that describes the end-effector position and orientation with respect to the reference frame,

$$T_4^0 = A_1^0 * A_2^1 * A_3^2 * A_4^3$$

$$= \begin{bmatrix} c1 & 0 & s1 & c1a_1 \\ s1 & 0 & -c1 & s1a_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c2 & -s2 & 0 & c2a_2 \\ s2 & c2 & 0 & s2a_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c3 & -s3 & 0 & c3a_3 \\ s3 & c3 & 0 & s3a_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c4 & -s & 0 & c4a_4 \\ s4 & c4 & 0 & s4a_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^0 = \begin{bmatrix} c_1c_{234} & -c_1s_{234} & s_1 & c_1(a_1 + a_2c_2 + a_3c_{23} + a_4c_{234}) \\ s_1c_{234} & -s_1s_{234} & -c_1 & s_1(a_1 + a_2c_2 + a_3c_{23} + a_4c_{234}) \\ s_{234} & c_{234} & 0 & a_2s_2 + a_3s_{23} + a_4s_{234} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.4)$$

The results have been verified using MATLAB (Robotics Toolbox), see Appendix (A.1).

3.2.2 Robot inverse kinematics

For inverse kinematics, end-effector pose is given $(x, y, z, \emptyset)$, and joint-variables have to be determined. Algebraic solution technique is used for this problem:

Algebraic method:

The end-effector (desired pose) is required to be at the end of the bucket for the excavation purpose, so when the driver of the excavator commands joint 4 to obtain the desired pose, his goal to put the end of the bucket at the soil surface with specified angle $\emptyset$ for the end-effector orientation, so only we have to find $\theta_1, \theta_2, \theta_3$ and θ_4. Now, the driver can excavate the soil by changing θ_4 (rotating the bucket down) then $\emptyset$ will be constant for the coming tasks. For inverse kinematic analysis, bucket will not be included for this algebraic solution.

Consider the **Error! Reference source not found.**:

Let: P_{4x} is the position vector from O1 to O4 refers to x_1-axis.

P_{4y} is the position vector from O1 to O4 refers to y_1-axis.

P_{3x} is the position vector from O1 to O3 refers to x_1-axis.

P_{3y} is the position vector from O1 to O3 refers to y_1-axis.

We assume that P_{4x} and P_{4y} are given as inputs.

$\emptyset = \theta_2 + \theta_3 + \theta_4$ is the end-effector orientation angle.

Using the equations (3.3), finding P_4 by T_4^1:

$$P_4 = \begin{bmatrix} a_2c_2 + a_3c_{23} + a_4c_{234} \\ a_2s_2 + a_3s_{23} + a_4s_{234} \\ 0 \end{bmatrix}$$

then P_3 becomes,

$$P_{3x} = P_{4x} - a_4c_{234} = a_2c_2 + a_3c_{23} \tag{3.5}$$

$$P_{3y} = P_{4y} - a_4s_{234} = a_2s_2 + a_3s_{23} \tag{3.6}$$

Which describe position of Frame (3) that depends on θ_2 and θ_3. Squaring and summing equations (3.5) and (3.6) yields:

$$P_{3x}^2 + P_{3y}^2 = a_2^2 + a_3^2 + 2a_2a_3c_3$$

$$c_3 = (P_{3x}^2 + P_{3y}^2 - a_2^2 - a_3^2)/2a_2a_3 \tag{3.7}$$

$-1 \leq c_3 \leq 1$, for θ_3 to be inside the reachable space.

$$s_3 = \pm\sqrt{1 - c_3^2} \tag{3.8}$$

angle θ_3 can be computed as

$$\theta_3 = \tan^{-1}\frac{s_3}{c_3} \tag{3.9}$$

Find θ_2 by solving equations (3.5) and (3.6):

$$s_2 = \frac{(a_2+a_3c_3)P_{3y} - a_3s_3P_{3x}}{P_{3x}^2 + P_{3y}^2}\ ,\ c_2 = \frac{(a_2+a_3c_3)P_{3x} - a_3s_3P_{3y}}{P_{3x}^2 + P_{3y}^2} \tag{3.10}$$

$$\theta_2 = \tan^{-1}\frac{s_2}{c_2} \tag{3.11}$$

If $\theta_3, \theta_4 = 0\ or\ \pi$ there will be a Kinematic singularity.

$$\theta_4 = \emptyset - \theta_2 - \theta_3 \tag{3.12}$$

$$\text{Finally, } \theta_1 = \tan^{-1}\frac{P^0_{4y}}{P^0_{4x}} = \tan^{-1}\frac{y}{x}. \qquad (3.13)$$

Refer to Appendix (A.2 and A.3) describing the MATLAB verification of both forward and inverse kinematics with numerical example.

In the same way, the same approach of analytical inverse kinematics solution can be developed for the complete homogenous transformation matrix (Frame 4 or 3 with respect to Frame 0).

This solution will take time analytically (algebraic), but as shown in MATLAB code it is easier and shorter (only solve the position equations to obtain the joints angles or use the closed-form solution above with some additions).

Probability of multiple solutions

For equations (3.8, 3.9), angle θ_3 can do multiple solution if s_3 has two solutions; the positive sign is relative to the elbow-down posture (solution 1) and the negative sign to the elbow-up posture (solution 2), but for the chosen range of angles below, θ_3 only has one solution which is elbow-up posture.

For equation (3.12), if $\emptyset$ is not specified, then the arm is redundant and there exist infinite solutions to the inverse kinematics problem, but here $\emptyset$ is known. In terms of the assigned joint limits, the multiple solutions are avoided, and always we have one solution for the inverse kinematics problem.

The desired joint limits considering avoiding multiple solutions:

$\theta_1 = \quad 0^o \ to \ \ 180^o$

$\theta_2 = \ -60^o \ to \ \ 60^o$

$\theta_3 = -100^o \ to \ -30^o$

$\theta_4 = -150^o \ to \ \ 30^o$

Note: For the given angle range of θ_3 , it will not do kinematic singularity because it will never reach $0 \ or \ \pi$, but if $\theta_4 = 0$, it will do a kinematic singularity, then 1DOF will be lost (i.e., $\emptyset$).

3.2.3 Workspace

The workspace (Operational space) is shown in Figure 3. 2, It is drawn with respect to limits of joint variables:

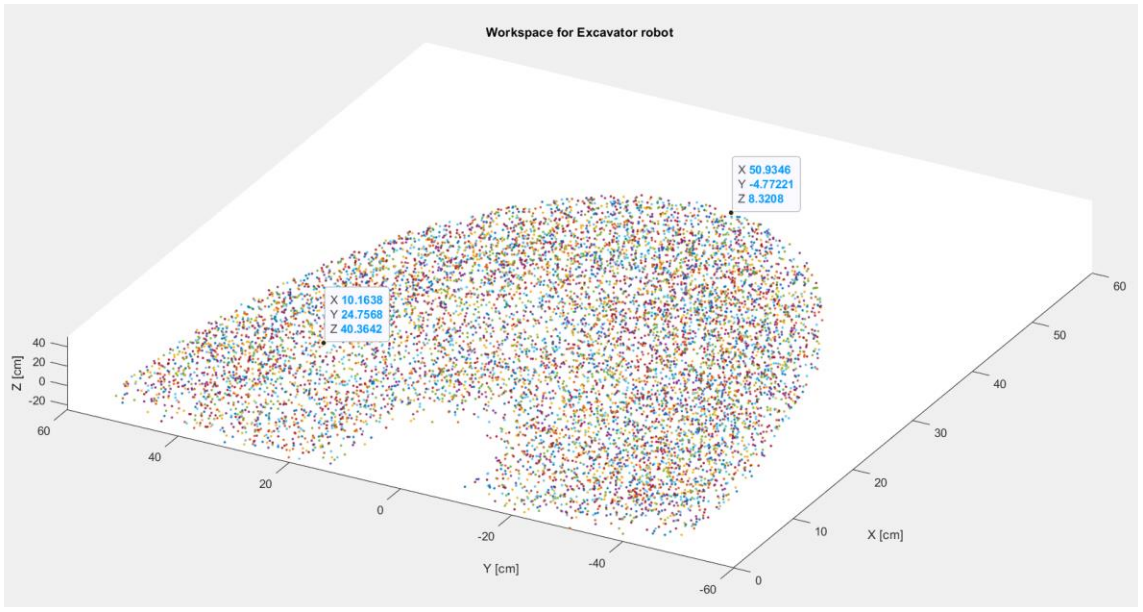


***Figure 3. 2**: The reachable end-effector poses*

Consider the length of each link in table 3.1 as follows:

$$a_1 = 8\ cm, \quad a_2 = 20cm, \quad a_3 = 15cm, \quad a_4 = 10\ cm$$

The end-effector can reach the following maximum positions:

- $50\ cm$ in $+x$
- $50\ cm$ in $\pm y$
- $15\ cm$ in $+z$, $-25\ cm$ in $-z$

3.2.4 Robot differential kinematics

Derivation of Jacobean for the manipulator:

Differential kinematics gives the relationship between the joint velocities and the corresponding end-effector linear and angular velocity. This mapping is described by a matrix, termed geometric Jacobian. Fig 3.1 shows the manipulator configuration of the excavator arm.

Consider equation (3.4),

$$T_4^0 = \begin{bmatrix} c_1c_{234} & -c_1s_{234} & s_1 & c_1(a_1 + a_2c_2 + a_3c_{23} + a_4c_{234}) \\ s_1c_{234} & -s_1s_{234} & -c_1 & s_1(a_1 + a_2c_2 + a_3c_{23} + a_4c_{234}) \\ s_{234} & c_{234} & 0 & a_2s_2 + a_3s_{23} + a_4s_{234} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

Using equation (3.30) in [5]:

$$\begin{bmatrix} J_{P_i} \\ J_{O_i} \end{bmatrix} = \begin{cases} \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix} & for\ a\ prismatic\ joint \\ \begin{bmatrix} z_{i-1} \times (P_e - P_{i-1}) \\ z_{i-1} \end{bmatrix} & for\ a\ revolute\ joint \end{cases} \tag{3.15}$$

For this manipulator we have four revolute joints, using equation (3.15), the Jacobian is

$$J(\theta) = \begin{bmatrix} z_0 \times (P_4 - P_0) & z_1 \times (P_4 - P_1) & z_2 \times (P_4 - P_2) & z_3 \times (P_4 - P_3) \\ z_0 & z_1 & z_2 & z_3 \end{bmatrix} \tag{3.16}$$

Computation of the position vectors of the various links gives

$$P_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad P_1 = \begin{bmatrix} a_1c_1 \\ a_1s_1 \\ 0 \end{bmatrix}, \quad P_2 = \begin{bmatrix} c_1(a_1 + a_2c_2) \\ s_1(a_1 + a_2c_2) \\ a_2s_2 \end{bmatrix}, \quad P_3 = \begin{bmatrix} c_1(a_1 + a_2c_2 + a_3c_{23}) \\ s_1(a_1 + a_2c_2 + a_3c_{23}) \\ a_2s_2 + a_3s_{23} \end{bmatrix}$$

$$P_4 = \begin{bmatrix} c_1(a_1 + a_2c_2 + a_3c_{23} + a_4c_{234}) \\ s_1(a_1 + a_2c_2 + a_3c_{23} + a_4c_{234}) \\ a_2s_2 + a_3s_{23} + a_4s_{234} \end{bmatrix}$$

Now, compute the unit vectors of revolute joint axes

$$z_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad z_1 = z_2 = z_3 = \begin{bmatrix} s_1 \\ -c_1 \\ 0 \end{bmatrix}$$

Substitute these in equation (3.16), we obtain the following geometric Jacobian

$$J(\theta) = \begin{bmatrix} -s_1(a_1 + a_2c_2 + a_3c_{23} + a_4c_{234}) & -c_1(a_2\,s_2 + a_3s_{23} + a_4s_{234}) & -c_1(a_3s_{23} + a_4s_{234}) & -c_1(a_4s_{234}) \\ c_1(a_1 + a_2c_2 + a_3c_{23} + a_4c_{234}) & -s_1(a_2\,s_2 + a_3s_{23} + a_4s_{234}) & -s_1(a_3s_{23} + a_4s_{234}) & -s_1(a_4s_{234}) \\ 0 & a_2c_2 + a_3c_{23} + a_4c_{234} & a_3c_{23} + a_4c_{234} & a_4c_{234} \\ 0 & s_1 & s_1 & s_1 \\ 0 & -c_1 & -c_1 & -c_1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad (3.17)$$

Only three of the six rows of the Jacobian in equation (3.17) are linearly independent. Having 3 DOFs of independent rows, so only consider the upper (3 × 3) block of the Jacobian. Since, $\omega_x = s_1(\dot{\theta}_2 + \dot{\theta}_3 + \dot{\theta}_4)$, $\omega_y = -c_1(\dot{\theta}_2 + \dot{\theta}_3 + \dot{\theta}_4)$, $\omega_z = \dot{\theta}_1$, means that ω_x and ω_y are not linearly independent; $\frac{\omega_x}{\omega_y} = -\tan(\theta_1)$.

For the end-effector velocity, we are interested in its linear velocity $\dot{x}, \dot{y}\ and\ \dot{z}$, and the angular velocity $\dot{\phi}$, so it is worth to take it into consideration for the Jacobian as the fourth DOF. The four DOFs allow specification of at most four end-effector variables (υ_x, υ_y, υ_z, $\dot{\phi}$) that forming the analytical Jacobian.

There are three scenarios for the desired tasks:

- 1st scenario:

This Jacobian relates $\dot{x}, \dot{y}\ and\ \dot{z}$ of the end-effector motion in terms of the joint velocities $\dot{\theta}_1, \dot{\theta}_2\ and\ \dot{\theta}_3$.

$$J_1(q) = \begin{bmatrix} -s_1(a_1 + a_2c_2 + a_3c_{23} + a_4c_{234}) & -c_1(a_2\,s_2 + a_3s_{23} + a_4s_{234}) & -c_1(a_3s_{23} + a_4s_{234}) \\ c_1(a_1 + a_2c_2 + a_3c_{23} + a_4c_{234}) & -s_1(a_2\,s_2 + a_3s_{23} + a_4s_{234}) & -s_1(a_3s_{23} + a_4s_{234}) \\ 0 & a_2c_2 + a_3c_{23} + a_4c_{234} & a_3c_{23} + a_4c_{234} \end{bmatrix} \quad (3.18)$$

- 2nd scenario:

For the end-effector motion, we are interested in its linear velocity $\dot{x}, \dot{y}, \dot{z}$, and the angular velocity $\dot{\phi}$ in terms of the joint velocities $\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3\ and\ \dot{\theta}_4$, so it is worth to take into consideration $\dot{\phi}$ = $(\dot{\theta}_2 + \dot{\theta}_3 + \dot{\theta}_4)$ as the 4th row, Jacobian becomes:

$J_2(q) =$

$$\begin{bmatrix} -s_1(a_1 + a_2c_2 + a_3c_{23} + a_4c_{234}) & -c_1(a_2\,s_2 + a_3s_{23} + a_4s_{234}) & -c_1(a_3s_{23} + a_4s_{234}) & -c_1(a_4s_{234}) \\ c_1(a_1 + a_2c_2 + a_3c_{23} + a_4c_{234}) & -s_1(a_2\,s_2 + a_3s_{23} + a_4s_{234}) & -s_1(a_3s_{23} + a_4s_{234}) & -s_1(a_4s_{234}) \\ 0 & a_2c_2 + a_3c_{23} + a_4c_{234} & a_3c_{23} + a_4c_{234} & a_4c_{234} \\ 0 & 1 & 1 & 1 \end{bmatrix} \quad (4.19)$$

- 3rd scenario, rotational motion of the whole manipulator (transmission of the soil):

  Extracting the last row of equation (3.17),

  $$\omega_z = \dot{\theta}_1 \quad (3.20)$$

  this relation will describe the angular velocity (ω_z) in terms of the joint velocity $\dot{\theta}_1$, where ω_z is the angular velocity of the end-effector about z_0.

See Jacobian verification using MATLAB Appendices (A.4 and A.5). The Jacobian singularities are described for each scenario in Appendix (A.6)

1st scenario: $\det(J_1) = 0$ when $\theta_3 = \theta_4 = 0$
2nd scenario: $\det(J_2) = 0$ when $\theta_3 = 0$.

3.3 Dynamic model

Derivation of the dynamic model by Lagrange formulation:

Derivation of the dynamic model of a manipulator arm is essential for simulation of motion and control design which provides a description of the relationship between the joint actuator torques and the motion of the structure. The analysis of the dynamic model can be useful for mechanical design, prototyping, control and simulation of a robotic arms. There are two main methods for derivation of motion equations. The first one is based on Lagrange formulation which is simple and systematic, and the second method is based on Newton-Euler formulation which derives the system in recursive form.

Now, we are concerned with the Lagrange formulation. The derivation of the equations of motion can be achieved independently of the base frame. Initially we assume that link 4 of the excavator manipulator is not included, see **Error! Reference source not found.**. We h ave a set of chosen generalized coordinates of $q_i = [\,\theta_1\ \theta_2\ \theta_3]^T$ which describe the link positions of the 3-DOF excavator manipulator. The sequence of derivation will be described shortly.

The book [5] will be followed in solving this problem. The equation of motion is

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F_v\dot{q} + g(q) = \tau \tag{3.21}$$

Where,

$B(q)$: the inertia matrix.

$C(q,\dot{q})$: the Coriolis and centrifugal forces matrix.

F_v: the matrix of viscous friction coefficients.

$g(q)$: the gravitational force vector.

Consider the 3-DOF manipulator in Figure 3. 3, shows the manipulator coordinates diagram. Let the distances of the centers of mass of the three links from the respective joint axes are $l1,\ l2\ and\ l3$. Also, let m_1, m_2, m_3 be the masses of the three links, and $I_1, I_2\ and\ I_3$ the moments of inertia relative to the centers of mass of each link, respectively.

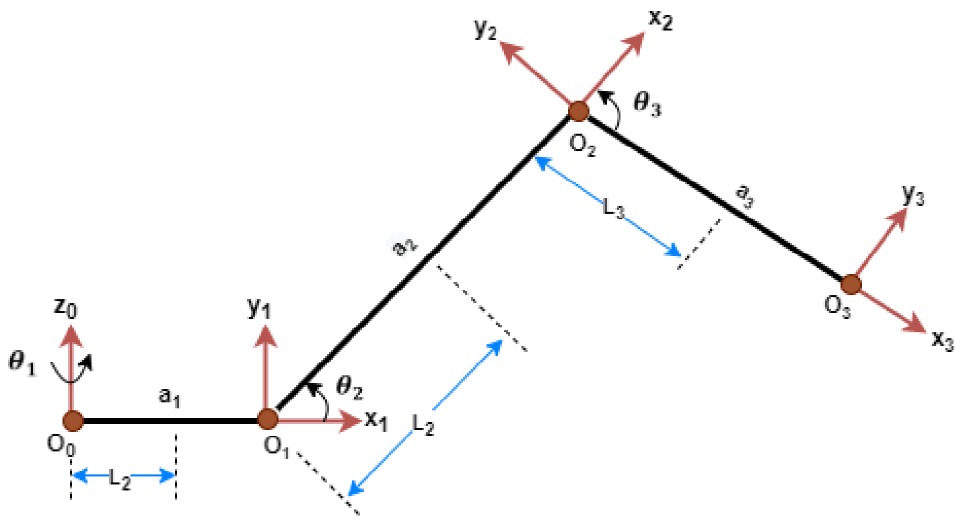


Figure 3. 3: Coordinates diagram of the manipulator in side view.

Finding the geometric Jacobian of each link to be applied to the center of mass of the link instead of the end-effector. Firstly, find the required forward kinematics of each link from the current frame to its the center of mass, i.e., substitute l_i instead of a_i in equations (3.2, 3.3) for $i = 1, 2$ *and* 3, then find the homogeneous transformation matrices A_1^0, A_2^0 *and* A_3^0. The homogenous transformation matrices become,

$$A_1^0 = \begin{bmatrix} c1 & 0 & s1 & c1l_1 \\ s1 & 0 & -c1 & s1l_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.22}$$

$$A_2^0 = \begin{bmatrix} c1 & 0 & s1 & c1a_1 \\ s1 & 0 & -c1 & s1a_1 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c2 & 0 & s2 & c2l_2 \\ s2 & 0 & -c2 & s2l_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c_2c_1 & -s_2c_1 & s_1 & c_1c_2l_2 + c_1a_1 \\ c_2s_1 & -s_2s_1 & -c_1 & c_2s_1l_2 + s_1a_1 \\ 0 & c_2 & 0 & s_2l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.23}$$

$$A_3^0 = \begin{bmatrix} c_2c_1 & -s_2c_1 & s_1 & c_1c_2l_2 + c_1a_1 \\ c_2s_1 & -s_2s_1 & -c_1 & c_2s_1l_2 + s_1a_1 \\ 0 & c_2 & 0 & s_2l_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c3 & -s3 & 0 & c3l_3 \\ s3 & c3 & 0 & s3l_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 & c_1(a_1 + a_2c_2 + l_3c_{23}) \\ s_1c_{23} & -s_1s_{23} & -c_1 & s_1(a_1 + a_2c_2 + l_3c_{23}) \\ s_{23} & c_{23} & 0 & a_2s_2 + l_3s_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.24}$$

Consider the following set of equations [5] where p_{j-1} is the position vector of the origin of Frame$j-1$, and z_{j-1} is the unit vector of axis z of Frame$j-1$:

$$\boldsymbol{J}_P^{(\ell_i)} = \begin{bmatrix} \boldsymbol{j}_{P1}^{(\ell_i)} & \dots & \boldsymbol{j}_{Pi}^{(\ell_i)} & \boldsymbol{0} & \dots & \boldsymbol{0} \end{bmatrix}$$
$$\boldsymbol{J}_O^{(\ell_i)} = \begin{bmatrix} \boldsymbol{j}_{O1}^{(\ell_i)} & \dots & \boldsymbol{j}_{Oi}^{(\ell_i)} & \boldsymbol{0} & \dots & \boldsymbol{0} \end{bmatrix};$$

$$\boldsymbol{j}_{Pj}^{(\ell_i)} = \begin{cases} \boldsymbol{z}_{j-1} & \text{for a } \textit{prismatic} \text{ joint} \\ \boldsymbol{z}_{j-1} \times (\boldsymbol{p}_{\ell_i} - \boldsymbol{p}_{j-1}) & \text{for a } \textit{revolute} \text{ joint} \end{cases}$$

$$\boldsymbol{j}_{Oj}^{(\ell_i)} = \begin{cases} \boldsymbol{0} & \text{for a } \textit{prismatic} \text{ joint} \\ \boldsymbol{z}_{j-1} & \text{for a } \textit{revolute} \text{ joint.} \end{cases}$$

With the chosen coordinate frames in Fig. 3.3, computation of the Jacobians using the above equations with (3.22, 3.23 and 3.24) and all joint are revolute, yields,

$$J_p^{(l1)} = \begin{bmatrix} -l_1 s_1 & 0 & 0 \\ l_1 c_1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.25}$$

$$J_p^{(l2)} = \begin{bmatrix} -(a_1 s_1 + l_2 s_1 c_2) & -l_2 c_1 s_2 & 0 \\ a_1 c_1 + l_2 c_1 c_2 & -l_2 s_1 s_2 & 0 \\ 0 & l_2 c_2 & 0 \end{bmatrix} \tag{3.26}$$

$$J_p^{(l3)} = \begin{bmatrix} -(a_1 s_1 + a_2 s_1 c_2 + l_3 s_1 c_{23}) & -a_2 c_1 s_2 - l_3 c_1 s_{23} & -l_3 c_1 s_{23} \\ a_1 c_1 + a_2 c_1 c_2 + l_3 c_1 c_{23} & -a_2 s_1 s_2 - l_3 s_1 s_{23} & -l_3 c_1 s_{23} \\ 0 & a_2 c_2 + l_3 c_{23} & l_3 c_{23} \end{bmatrix} \tag{3.27}$$

$$J_o^{(l1)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{3.28}$$

$$J_o^{(l2)} = \begin{bmatrix} 0 & s_1 & 0 \\ 0 & -c_1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{3.29}$$

$$J_o^{(l3)} = \begin{bmatrix} 0 & s_1 & s_1 \\ 0 & -c_1 & -c_1 \\ 1 & 0 & 0 \end{bmatrix} \tag{3.30}$$

The inertia matrix $B(q)$ for the links and motors [5], select the first term considering only the inertia for the links, $i = 1,2,3$

$$\boldsymbol{B}(\boldsymbol{q}) = \sum_{i=1}^{n} \Big(m_{\ell_i} \boldsymbol{J}_P^{(\ell_i)T} \boldsymbol{J}_P^{(\ell_i)} + \boldsymbol{J}_O^{(\ell_i)T} \boldsymbol{R}_i \boldsymbol{I}_{\ell_i}^{i} \boldsymbol{R}_i^T \boldsymbol{J}_O^{(\ell_i)} \\ + m_{m_i} \boldsymbol{J}_P^{(m_i)T} \boldsymbol{J}_P^{(m_i)} + \boldsymbol{J}_O^{(m_i)T} \boldsymbol{R}_{m_i} \boldsymbol{I}_{m_i}^{m_i} \boldsymbol{R}_{m_i}^T \boldsymbol{J}_O^{(m_i)} \Big)$$

For the three links, the inertia matrix size is 3x3 with the following elements:

$$b_{11} = m_1 l_1^2 + I_{1zz} + I_{2zz} + m_3(l_2 c_2 + a_2)^2 \\ + m_3(a_1^2 + 2a_1 a_2 c_2 + a_2^2 c_2^2 + 2a_1 l_3 c_{23} + 2a_2 l_3 c_2 c_{23} + l_3^2 c_{23}^2) + I_{3zz}$$

$$b_{12} = -I_{2zx} s_1 + I_{2zy} c_1 - I_{3zx} s_1 + I_{3zy} c_1$$

$$b_{13} = I_{3zy} c_1 - I_{3zx} s_1$$

$$b_{21} = I_{2yz} c_1 - I_{2xz} s_1 + I_{3yz} c_1 - I_{3xz} s_1$$

$$b_{22} = I_{2xx} s_1^2 + I_{2yy} c_1^2 + 2I_{2yx} s_1 c_1 + m_2 l_2^2 s_2^2 + m_3(a_2^2 + l_3^2 + 2a_2 l_3 c_3) + I_{3xx} s_1^2 + \\ I_{3yy} c_1^2$$

$b_{23} = m_3 * (l_3^2 + l_3 * a_2 * c_3) + I_{3xx} * s_1^2 + I_{3yy} * c_1^2$

$b_{31} = -b_{13}$

$b_{32} = b_{23}$

$b_{33} = m_3 * l_3^2 + I_{3xx} * s_1^2 + I_{3yy} * c_1^2$

Forming the inertia matrix, yields

$$B(q) = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \tag{3.31}$$

Notice that the inertia matrix is configuration-dependent.

The matrix $C(q, \dot{q})$, with elements C_{ij}, where $i = j = 1,2,3$. Using the Christoffel symbols C_{ijk} [5] to reduce the errors in derivation, using the following equation:

$$C_{ijk} = \frac{1}{2}\left(\frac{\partial b_{ij}}{\partial q_k} + \frac{\partial b_{ik}}{\partial q_j} - \frac{\partial b_{jk}}{\partial q_i}\right)$$

with $i = j = k = 1,2,3$, yields

$C_{112} = C_{121} = l_2 m_2 s_2 (l_2 c_2 + a_2) + m_3 (a_1 a_2 s_2 + a_2^2 c_2 s_2 + a_1 l_3 s_{23} + l_3^2 c_{23} s_{23} +$
$l_3 a_2 * s(2\theta_2 + \theta_3))$

$C_{113} = C_{131} = -m_3 (a_1 l_3 s_{23} + a_2 l_3 c_2 s_{23} + l_3^2 c_{23} s_{23})$

$C_{122} = c_1 s_1 (I_{2yy} - I_{2xx} + I_{3yy}) + I_{2yx} (s_1^2 - c_1^2)$

$C_{123} = C_{132} = \frac{1}{2} m_3 l_3 a_2 s_3$

$C_{133} = -s_1 c_1 (I_{3xx} - I_{3yy})$

$C_{211} = -\left(c_1 * (I_{2xz} + I_{3xz}) + s_1 * (I_{2zy} + I_{3zy})\right) - C_{112}$

$C_{212} = C_{221} = -c_{122}$

$C_{213} = C_{231} = s_1 c_1 * (I_{3xx} - I_{3yy})$

$$C_{223} = C_{232} = -m_3 l_3 a_2 s_3$$

$$C_{233} = -m_3 l_3 a_2 s_3$$

$$C_{311} = -\left(I_{3yz} * s_1 + I_{3xz} * c_1\right) - C_{113}$$

$$C_{312} = C_{321} = \left(I_{3xx} - I_{3yy}\right) * s_1 c_1$$

$$C_{313} = C_{331} = \left(I_{3xx} - I_{3yy}\right) * s_1 c_1$$

$$C_{322} = -C_{223}$$

$$C_{323} = C_{332} = -\frac{1}{2} * m_3 l_3 a_2 s_3 + C_{123}$$

For simplification, let

$$h_1 = C_{112} = C_{121} = \frac{1}{2}\frac{\partial b_{11}}{\partial \theta_2}$$

$$h_2 = C_{113} = C_{131} = \frac{1}{2}\frac{\partial b_{11}}{\partial \theta_3}$$

$$h_3 = C_{122} = -\frac{1}{2}\frac{\partial b_{22}}{\partial \theta_1}$$

$$h_4 = C_{123} = C_{132} = -\frac{1}{2}\frac{\partial b_{23}}{\partial \theta_3}$$

$$h_5 = C_{133} = -\frac{1}{2}\frac{\partial b_{33}}{\partial \theta_1}$$

$$h_6 = C_{211} = \frac{\partial b_{21}}{\partial \theta_1} - h_1$$

$$h_7 = C_{213} = C_{131} = \frac{1}{2}\frac{\partial b_{23}}{\partial \theta_1}$$

$$h_8 = C_{223} = C_{232} = \frac{1}{2}\frac{\partial b_{22}}{\partial \theta_3}$$

$$h_9 = C_{233} = \frac{1}{2}\frac{\partial b_{23}}{\partial \theta_3}$$

$$h_{10} = C_{311} = \frac{\partial b_{31}}{\partial \theta_1} - h_2$$

$$h_{11} = C_{312} = C_{321} = \frac{1}{2}\frac{\partial b_{32}}{\partial \theta_1}$$

$$h_{12} = C_{313} = C_{331} = \frac{1}{2}\frac{\partial b_{33}}{\partial \theta_1}$$

$$h_{13} = C_{323} = C_{332} = \frac{1}{2}\frac{\partial b_{32}}{\partial \theta_3} + h_4$$

As a consequence, the generic elements of $\boldsymbol{C}$ [5] are

$$C_{ij} = \sum_{k=1}^{n} C_{ijk}\dot{q}_k$$

For $k = 1,2,3$, the C matrix elements are:

$$C_{11} = h_1\dot{\theta}_2 + h_2\dot{\theta}_3$$

$$C_{12} = h_1\dot{\theta}_1 + h_3\dot{\theta}_2 + h_4\dot{\theta}_3$$

$$C_{13} = h_2\dot{\theta}_1 + h_4\dot{\theta}_2 + h_5\dot{\theta}_3$$

$$C_{21} = h_6\dot{\theta}_1 - h_3\dot{\theta}_2 + h_7\dot{\theta}_3$$

$$C_{22} = -h_3\dot{\theta}_1 + h_8\dot{\theta}_3$$

$$C_{23} = h_7\dot{\theta}_1 + h_8\dot{\theta}_2 + h_9\dot{\theta}_3$$

$$C_{31} = h_{10}\dot{\theta}_1 + h_{11}\dot{\theta}_2 + h_{12}\dot{\theta}_3$$

$$C_{32} = h_{11}\dot{\theta}_1 - h_8\dot{\theta}_2 + h_{13}\dot{\theta}_3$$

$$C_{33} = h_{12}\dot{\theta}_1 + h_{13}\dot{\theta}_2$$

Leading to the matrix,

$$C(q,\dot{q}) = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \tag{3.32}$$

As for gravitational force terms, with $g_0 = [0 \quad 0 \quad -g]^T$ that gravitational acceleration in direction of the negative z_0 axis. Using the potential energy approach with the equation [5],

$$\frac{\partial \mathcal{U}}{\partial q_i} = -\sum_{j=1}^{n}\left(m_{\ell_j}\boldsymbol{g}_0^T\frac{\partial \boldsymbol{p}_{\ell_j}}{\partial q_i} + m_{m_j}\boldsymbol{g}_0^T\frac{\partial \boldsymbol{p}_{m_j}}{\partial q_i}\right)$$
$$= -\sum_{j=1}^{n}\left(m_{\ell_j}\boldsymbol{g}_0^T\boldsymbol{j}_{Pi}^{(\ell_j)}(\boldsymbol{q}) + m_{m_j}\boldsymbol{g}_0^T\boldsymbol{j}_{Pi}^{(m_j)}(\boldsymbol{q})\right) = g_i(\boldsymbol{q})$$

For each link i=1,2,3 substituting $j = 1,2,3$, yields

$$g_1 = 0$$

$$g_2 = g(m_2 l_2 c_2 + m_3(a_2 c_2 + l_3 c_{23}))$$

$$g_3 = g(m_3 l_3 c_{23})$$

The gravity vector becomes,

$$g = [g_1 \quad g_2 \quad g_3]^T \tag{3.33}$$

Using equation (3.21) with the absence of the friction and tip contact forces ($F_v = 0$), the resulting equations of motion are

$$[b_{11} \quad b_{12} \quad b_{13}]\begin{bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\\ \ddot{\theta}_3\end{bmatrix} + [C_{11} \quad C_{12} \quad C_{13}]\begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \dot{\theta}_3\end{bmatrix} + g_1 = \tau_1 \tag{3.34}$$

$$\begin{aligned}
&(m_1 l_1^2 + I_{1zz} + I_{2zz} + m_2(l_2 c_2 + a_2)^2 \\
&+ m_3(a_1^2 + 2a_1 a_2 c_2 + a_2^2 c_2^2 + 2a_1 l_3 c_{23} + 2a_2 l_3 c_2 c_{23} + l_3^2 c_{23}^2) + I_{3zz})\ddot{\theta}_1 \\
&+ (-I_{2zx}s_1 + I_{2zy}c_1 - I_{3zx}s_1 + I_{3zy}c_1)\ddot{\theta}_2 + \left(I_{3zy}c_1 - I_{3zx}s_1\right)\ddot{\theta}_3 \\
&+ 2\Big(-l_2 m_2 s_2(l_2 c_2 + a_2) \\
&- m_3\big(a_1 a_2 s_2 + a_2^2 c_2 s_2 + a_1 l_3 s_{23} l_3 c_{23} s_{23} + l_3 a_2 s(2\theta_2 + \theta_3)\big)\Big)\dot{\theta}_1\dot{\theta}_2 \\
&+ 2\big(-m_3(a_1 l_3 s_{23} + a_2 l_3 c_2 s_{23} + l_3^2 c_{23} s_{23})\big)\dot{\theta}_1\dot{\theta}_3 \\
&+ \Big(c_1 s_1\big(I_{2yy} - I_{2xx} - I_{3yx} + I_{3yy}\big) + I_{2yx}(s_1^2 - c_1^2)\Big)\dot{\theta}_2^2 \\
&+ (m_3 l_3 a_2 s_3)\dot{\theta}_2\dot{\theta}_3 + \Big(-s_1 c_1\big(I_{3xx} - I_{3yy}\big)\Big)\dot{\theta}_3^2 = \tau_1
\end{aligned}$$

$$[b_{21} \quad b_{22} \quad b_{23}]\begin{bmatrix}\ddot{\theta}_1\\ \ddot{\theta}_2\\ \ddot{\theta}_3\end{bmatrix} + [C_{21} \quad C_{22} \quad C_{23}]\begin{bmatrix}\dot{\theta}_1\\ \dot{\theta}_2\\ \dot{\theta}_3\end{bmatrix} + g_2 = \tau_2 \tag{3.35}$$

$$
\begin{aligned}
&(I_{2xx}s_1^2 + I_{2yy}c_1^2 + 2I_{2yx}s_1c_1 + m_2l_2^2s_2^2 + m_3(a_2^2 + l_3^2 + sa_2l_3c_3) \\
&+ I_{3xx}s_1^2I_{3yy}c_1^2)\ddot{\theta}_1 + (I_{2xx}s_1^2 + I_{2yy}c_1^2 + 2I_{2yx}s_1c_1 + m_2l_2^2s_2^2 \\
&+ m_3(a_2^2 + l_3^2 + sa_2l_3c_3) + I_{3xx}s_1^2 + I_{3yy}c_1^2)\ddot{\theta}_2 + (m_3 \\
&* (l_3^2 + l_3 * a_2 * c_3) + I_{3xx} * s_1^2 + I_{3yy} * c_1^2)\ddot{\theta}_3 \\
&+ (\left(-c_1(I_{2xz} + I_{3xz}) + s_1\left(I_{2zy} + I_{3zy}\right)\right) + (l_2m_2s_2(l_2c_2 + a_2) \\
&+ m_3(a_1a_2s_2 + a_2^2c_2s_2 + a_1l_3s_{23}l_3^2c_{23}s_{23} + l_3a_2s(2\theta_2 + \theta\_3)))\dot{\theta}_1^2 \\
&- 2(c_1s_1\left(I_{2yy} - I_{2xx} - I_{3yx} + I_{3yy}\right) + I_{2yx}(s_1^2 - c_1^2))\dot{\theta}_1\dot{\theta}_2 \\
&+ 2(-m_3l_3a_2s_3)\dot{\theta}_2\dot{\theta}_3 + 2(s_1c_1(I_{3xx} - I_{3yy}))\dot{\theta}_1\dot{\theta}_3 + (-m_3l_3a_2s_3)\dot{\theta}_3^2 \\
&+ g(m_2l_2c_2 + m_3(a_2c_2 + l_3c_{23})) = \tau_2
\end{aligned}
$$

$$
\begin{bmatrix} b_{31} & b_{32} & b_{33} \end{bmatrix}\begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \\ \ddot{\theta}_3 \end{bmatrix} + \begin{bmatrix} C_{31} & C_{23} & C_{33} \end{bmatrix}\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} + g_3 = \tau_3 \qquad (3.36)
$$

$$
\begin{aligned}
&(I_{3zy}c_1 - I_{3zx}s_1)\ddot{\theta}_1 + (m_3 * (l_3^2 + l_3 * a_2 * c_3) + I_{3xx} * s_1^2 + I_{3yy} * c_1^2)\ddot{\theta}_2 \\
&+(m_3 * l_3^2 + I_{3xx} * s_1^2 + I_{3yy} * c_1^2)\ddot{\theta}_3 + (-\left(I_{3yz}s_1 + I_{3xz}c_1\right) \\
&+m_3(a_1l_3s_{23} + a_2l_3c_2s_{23} + l_3^2c_{23}s_{23}))\dot{\theta}_1^2 + (m_3l_3a_2s_3)\dot{\theta}_2^2 \\
&+2\left(\left(I_{3xx} - I_{3yy}\right)s_1c_1\right)\dot{\theta}_2\dot{\theta}_1 + 2\left(\left(I_{3xx} - I_{3yy}\right)s_1c_1\right)\dot{\theta}_3\dot{\theta}_1 \\
&+m_3l_3gc_{23} = \tau_3
\end{aligned}
$$

Where τ_1, τ_2 and τ_3 denotes the torques applied to the joints

Chapter Four: Design of Structure and Mechanical Components

4.1 Introduction

This chapter discuss the mechanical design and material selection of the manipulator prototype that will be scaled from a real excavator. The design requirements are set, such as the maximum deflection, maximum stress, lightest weight and minimum cost. We challenged to convert and realize the derived kinematic model to a physical structure by means of material selection, mechanical parts (e.g., rigid links, shafts), selecting suitable actuators in terms to dimensioning, supporting and fastening of the various components.

4.2 Material selection and analysis

4.2.1 Introduction

The mechanical structure of an excavator robot provides the means needed to dig and load the soil into trucks for haulage storage areas for a given job. The choice of materials has a direct impact on performance, precision, repeatability, and mechanical noise transfer into the parts.

The mechanical subsystem is comprised of the drive system, and the frame structure, assembly of rigid bodies linked by four joints, three planar implements connected through revolute joints known as the boom, arm, and bucket, and one vertical revolute joint on base as shown in Figure 4. 1 [6], so the link can rotate with one degree of freedom around a designated axis of rotation. The motion of each part is achieved using a servo drive with its own control system, according to a prescribed motion.

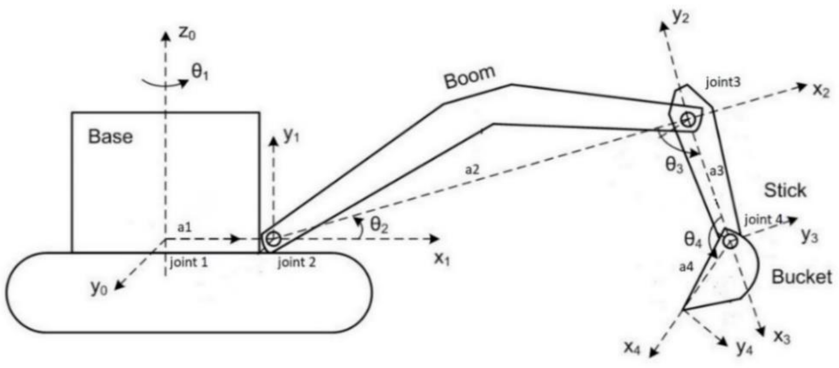


Figure 4. 1: Schematic diagram of an excavator.

The design process is one of the most extensive and essential processes that need to be carried out in order to meet the needs, and the first step in the design is to define the design requirements.

- The maximum deflection for beam less than 3mm.
- The minimum Factor of Safety (FOS) greater than three.
- The mechanical properties (in particular the yield stress).
- The costs.

4.2.2 Material selection and analysis

Selection of material is very important procedure before detail design of product, a suitable material is selected for excavator in terms of the desired mechanical properties. The material considered for manufacturing of structure is Aluminum or Steel, among these chosen materials, one should be selected to increase performance, the selected material for excavator structure must give high strength, low density, economic, good machinability and mechanical properties.

The aim of this study is to choose between steel alloy and aluminum alloy for a lighter weight and order to maintain the value of the safety factor and deformability.

A lighter weight material will ultimately translate to greater payload capacity, productivity and decreased energy consumption. Luigi Solazzi [7] proposed a substitute the steel alloy for an aluminum alloy for classical excavator machine, (Replacing steel alloys with aluminum alloys for lighter weight and increase the load capacity of the bucket).

As the results, the final geometry of the component presents a reduction in weight of about 50%, and increase the capacity of the bucket, while the increased in the cost.

Table 4.1 below shows the mechanical properties for steel and aluminum alloy.

Table 4. 1: mechanical properties for steel and aluminum alloy.

Materials	σ_{yield} (MPs)	E (GPs)	ρ (Kg/m^3)
Aluminum	195	67	2.7
Steel	355	210	7.8

As shown in the Table 4. 1, the ratio between Aluminum and Steel in terms of density is (2.89: 1 respectively), and with respect to the modulus of elasticity (3.13: 1 respectively).

School of Engineering Grand Valley State University, Brad Peirson [8] presented report Comparing of Specific Properties of Engineering Materials, by calculation its specific strength (the strength-to-weight ratio of the material), this means that the material has a lightweight, but it also has a high strength. Where the density of a material can be a crucial factor in determining the material that is best suited for an application.

Where the specific modulus between steel alloy and aluminum alloy (C1018 and AA6061) similar, the specific yield strength for the steel alloy is twice the value of the aluminum alloy, in addition, the cost per unit yield strength of the steel alloy is half the cost of the aluminum.

Consideration of the various requirements and factors affecting the design and dimensions of the structure and results in the choice of one or perhaps several alternative types of structure and materials, which offer the best general solution. The primary consideration is the function of the structure. Secondary considerations such as aesthetics, economics, and the environment may also be taken into account.

Now, we want to find the dimensions of the cross section of the beams shown in the Figure 4. ***2***[9], when selection materials the equivalent of the excavator robot is one beam and equivalent force applied (static force) at the end of beam.

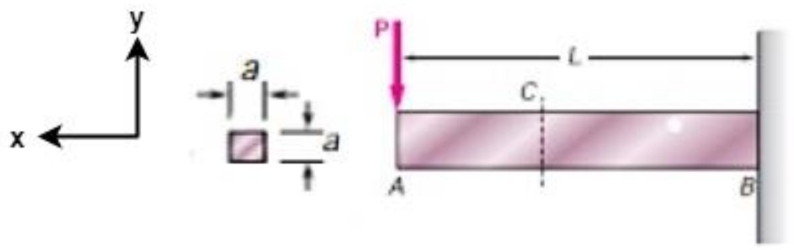


Figure 4. 2: Equivalent robot beam.

Determine the minimum cross-section for the beam to provide a minimum factor of safety of 3 based on the Distortion energy (DE) theory, while beam AB has a length L=1m and square cross-sectional area, the bar is subjected as shown to a load P=50 N,

The beam AB experiences maximum bending moment (M) at the wall on which distance is larger. Bending stress will be a maximum on the outer surface. The critical stress element will be at the wall, at either the top (tension) or the bottom (compression) on the y-axis see Figure 4. 3[9].

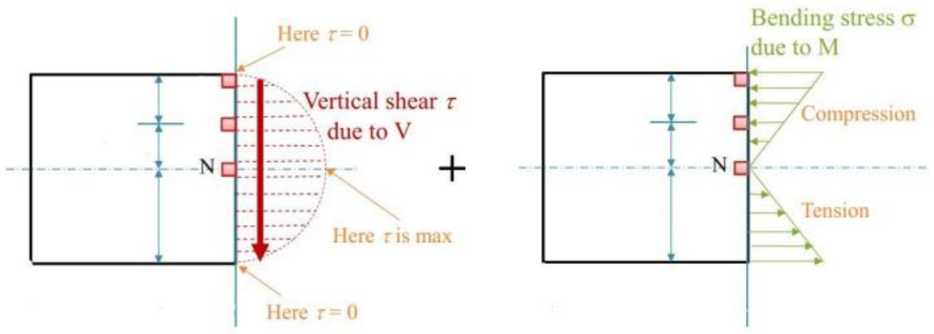


Figure 4. 3: Stresses on the desired section are superposition of shear stress due to V and normal stress due to M

The moment M produced tensile stress

$$M = r \times P$$

Substitute $r = L = \ 1\hat{\imath}$,

$$M = -P\ \hat{k}\ \ [N.m]$$

The general formula for bending or normal stress on the section is given by:

$$\sigma = \frac{M * C}{I_x}$$

Found the moment of inertia about the neutral axis to be

$$I_x = \frac{a^4}{12}\ m^4$$

Clearly the top of the section is further with a distance c = a/2 from the neutral axis

$$\sigma_x = \frac{1 * P * 12}{2 * a^3}$$

$$\sigma_x = \frac{-6 * P}{a^3}$$

$$a = \sqrt[3]{\frac{6 * P}{\sigma_x}}$$

The three principal stresses at B are

$$\sigma_{A,B} = \frac{\sigma_x + \sigma_y}{2} \pm \sqrt{\left(\frac{\sigma_x - \sigma_y}{2}\right)^2 + \tau_{xy}^2}$$

$$\sigma_{A,B} = \frac{-6}{a^3} \mp \frac{-6}{2 * a^3}$$

$$\sigma_A = \frac{-6}{a^3}$$

$$\sigma_B = 0 \,, \sigma_C = 0$$

Therefore, $\sigma_1 = 0.0 \,, \sigma_3 = \frac{-6}{a^3} \ \,, \sigma_2 = 0.0$

The maximum normal stress at point B is

$$\sigma_{max} = \frac{-6}{a^3}$$

The maximum shear stress at point B is

$$\tau_{max} = \frac{\sigma_1 - \sigma_3}{2} = \frac{3}{a^3}$$

For plane stress, the von Mises stress can be represented by the principal stresses

$$\sigma' = \left[\frac{(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2}{2}\right]^{1/2}$$

$$\sigma' = \sigma_3$$

Distortion Energy (DE) Failure theory:

$$n = \frac{S_y}{\sigma\prime}, \quad \text{at} \ \ n = 3$$

$$\sigma' = \frac{S_y}{3}$$

The beam has reached a maximum deflection value δ_m at point A. In Table 4. 2 below, show the mechanical properties and computed value for maximum deflection and von-mises stress of square cross-section.

Table 4. 2: Find maximum deflection between two materials.

Materials	E (MPa)	S_y (MPs)	σ' (MPa)	a (mm)	δ_{max} (mm)
Aluminum	75	105	35	20.11	-13.9
Steel	200	220	73	15.7	-15.6

The results obtained showed that deflection is unacceptable, so we will choose another cross-section for the beam, so we need to increase the value by using the two plates. which reduces the deflection, increases the coefficient of safety, reduces stress, and weight of the structure.

In Figure 4. 4, see hollow square cross-section parameters.

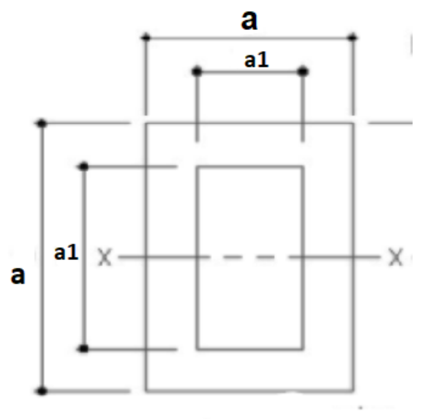


Figure 4. 4: Hollow square cross-section.

From what we found out, the use of steel is convenient in designing excavator frame, there are structural and constructional requirements and limitations, which may affect the type of structure to be designed. In selecting materials for a given project one should consider all requirements simultaneously and taking into account the available components such as motors that provide the torque required for the design. For the convenience of the procedure, it is beneficial to combine some of them.

With the materials and the related properties, one can go through the procedure as mentioned above to reach the final selection of material. Aluminum is selected to be the metal for the plates for the construction of manipulator links because it has a lightweight and high stiffness, and the dimensions of the manipulator and the load required on the end-effector were re-selected, so that the total length of the manipulator became 60cm while reducing the capacity of the bucket with a maximum load of 500g, and to fit the design, the materials were combined in the design of the manipulator.

A steel material was chosen to form the base of the manipulator and shafts of the motors, and Aluminum was used to form the link plates for each arm, boom and, bucket. Delrin[1] was used to make columns to connect the link plates.

4.3 Mechanical components of structure

The mechanical design will be illustrated in terms of the parts, coupling among them and building the manipulator structure. HITACHI model EX21 of the excavator is selected and scaled to build the initial design, see Figure 4. 5. The dimensions are 1/10th scaled to obtain suitable prototype with same structure.

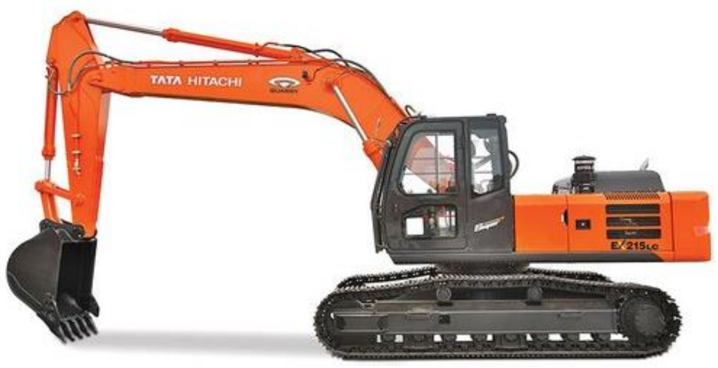


Figure 4. 5: HITACHI model EX21.

The excavator is a robot manipulator, the links of a manipulator are connected by a revolute joint. The design shows the links, joints and dimensions.

For the selected materials, the structure is designed in two steps. Firstly, a rough estimation of weights, motors, and gears is taken to allow the selection of motors (see section 5.4.1). Finally, a full design will be introduced for the whole structure to satisfy the design requirements.

The structure consists of four serial links base, Boom, Arm, Bucket, and motor shafts, all are designed using SOLIDWORKS software as following,

[1] Delrin is a crystalline plastic that offers an excellent balance of properties that bridge the gap between metals and plastics. Delrin possesses high tensile strength, creep resistance, and toughness. It also exhibits low moisture absorption. [10]

a. Base: The first link consists of two parts: the fixed base is fixed with the ground see **Error! Reference source not found.**, and the moving part with the shaft of t he motor and rotates about the vertical axis see Figure 4. 7, and it lifts the structure. The offset between its joint and the second link joint is $a_1 = 8\ cm$.

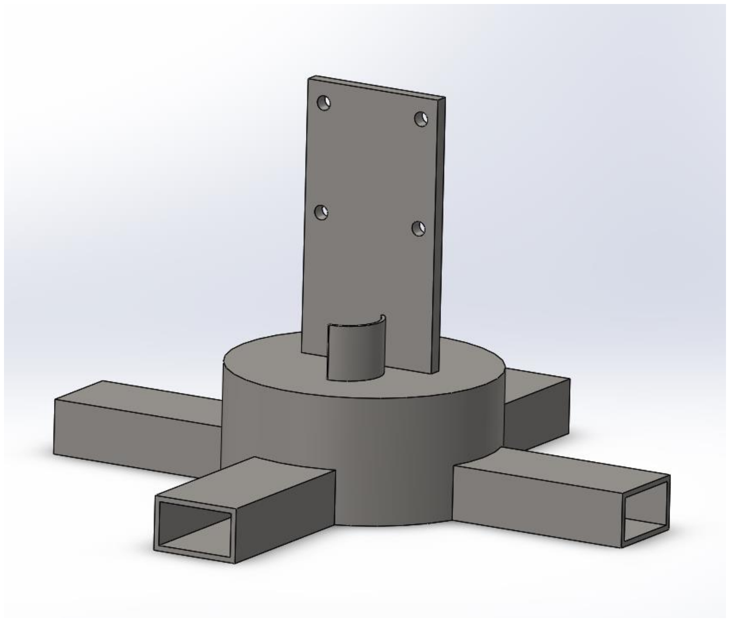


Figure 4. 6: Fixed Base

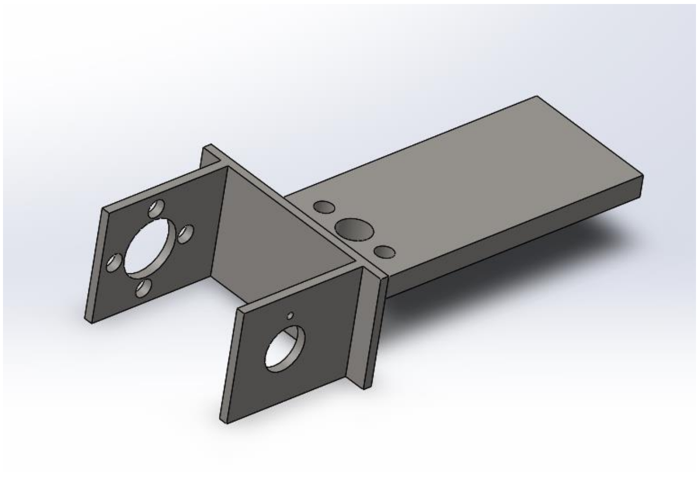


Figure 4. 7: base

b. Boom: the second link, its length is a_2 = 20cm, illustrate in Figure 4. 8

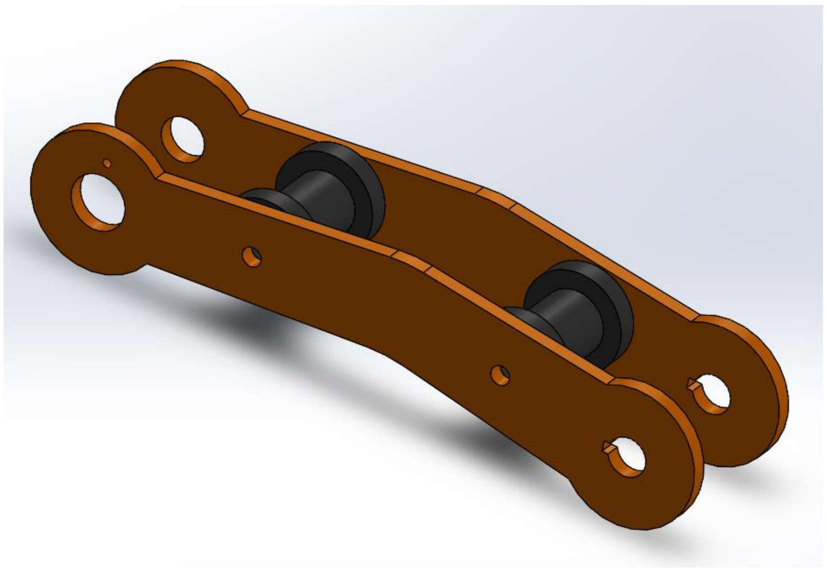


Figure 4. 8: Boom part

c. Arm: the third link associated with boom and bucket, has length of $a_3 = 15$ cm, illustrate in Figure 4. 9

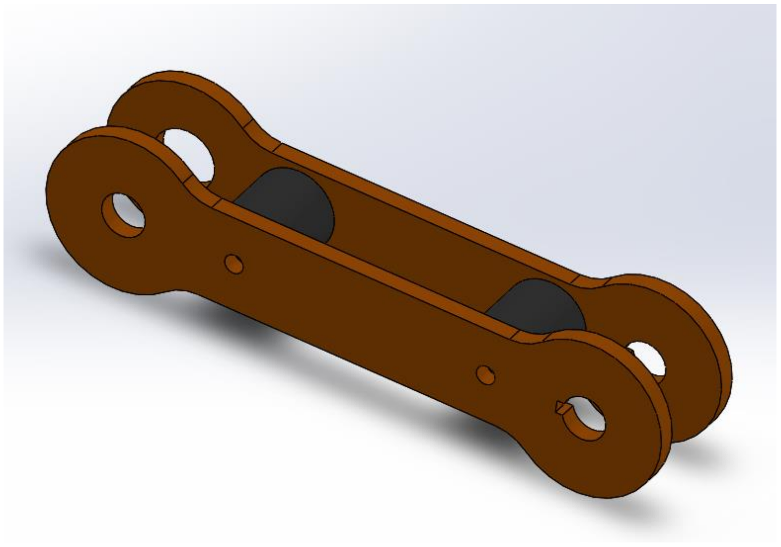


Figure 4. 9: Arm part

d. Bucket: the last link of the manipulator that is the end-effector, capacity of the bucket with a maximum load of 500g, and length of $a_4 = 10$cm. illustrates in Figure 4. 10

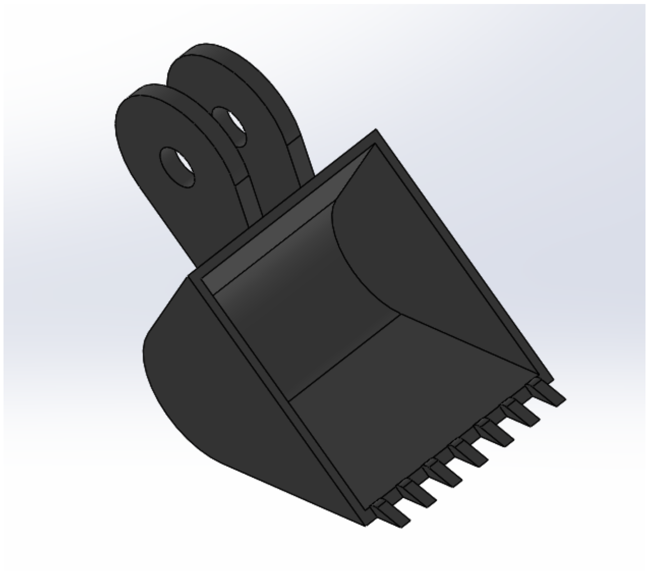


Figure 4. 10: **Bucket part**

e. Shaft of motors: A shaft is a rotating machine element, circular cross-section, which is used to transmit power from motors to links, see Figure 4. 11. used key to connect a rotating machine element to the shaft. A key prevents a relative rotation between the two parts and may enable torque transmission to occur.

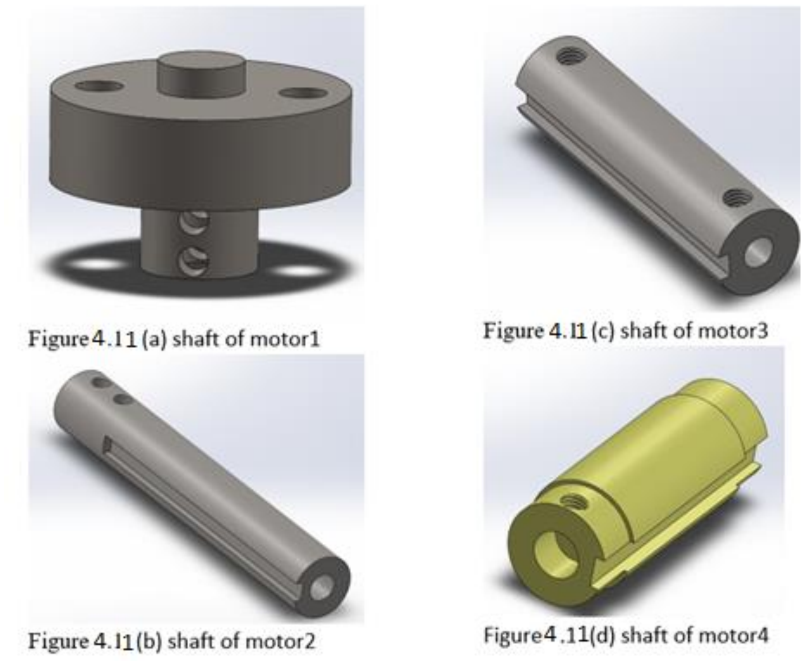


Figure4.11 (a) shaft of motor1
Figure 4.11 (c) shaft of motor3
Figure 4.11(b) shaft of motor2
Figure4 .11(d) shaft of motor4

Figure 4. 11: **Shafts of motors: (a) for motor-1, (b) for motor- 2, (c) for motor-3, and (d) for motor-4**

Figure 4. 12 shows the manipulator graphically with all its assembled parts as a simple design. The dimensions and joints are illustrated.

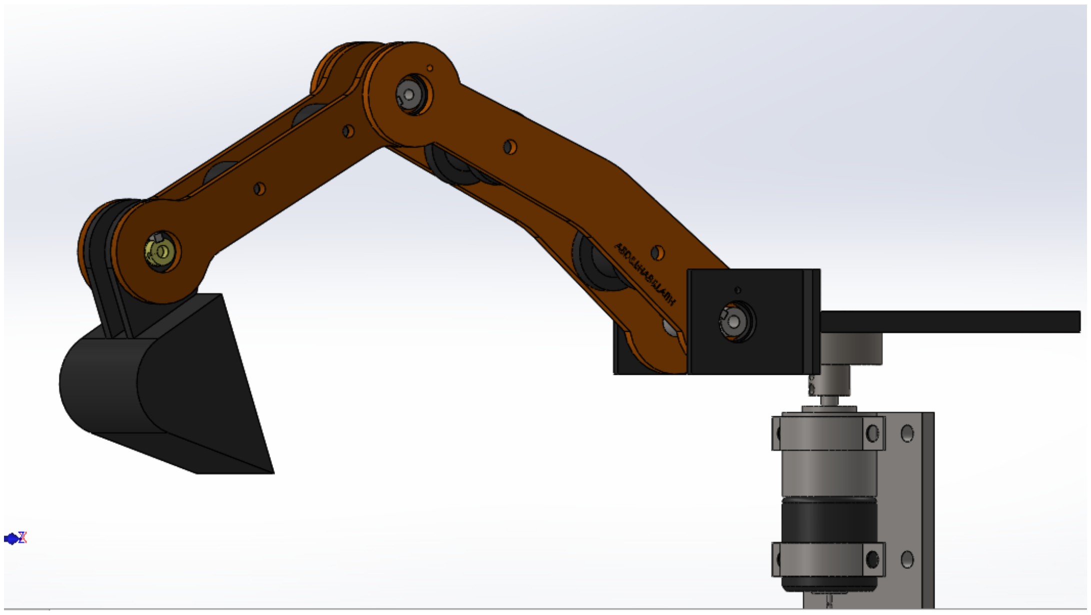


Figure 4. 12: Excavator prototype (Part assembly)

4.4 Structure and load analysis through SOLIDWORKS

In this section, the structure involves establishing the loading, which must be supported by the structure and therefore must be considered in its design. Structural analysis is the determination of the effects of loads on physical structures and finds the maximum stress and maximum deflection, which act when the maximum possible load placed on the end-effector of the manipulator, the maximum load on the end-effector is $0.5\ Kg$ which equals approximately $5\ N.m$.

Undertaking virtual prototyping instead of testing a physical model provides huge time and cost savings, the stress analysis will be performed using SOLIDWORKS by applying $5\ N.m$ vertically downwards at the end of the 2-plates beam.

SOLIDWORKS® Simulation of structural analysis tools that use Finite Element Analysis (FEA) to predict a product’s real-world physical behavior by virtually testing CAD models. The portfolio provides linear, non-linear static and dynamic analysis capabilities. [11]

Simulation is integrated directly within SOLIDWORKS which makes going from design to Simulation. Load condition has been implemented in SOLIDWORKS in order to obtain the force (inclusive of inertia effect). The stress and deflection analysis are as follow,

Figure 4. 13, shows the result obtained using SOLIDWORKS for Aluminum (1060 Alloy) deflection, Factor of Safety (FOS) and Von-mess stress. The maximum value of deflection is less than 3 mm and the minimum factor of safety 10.74.

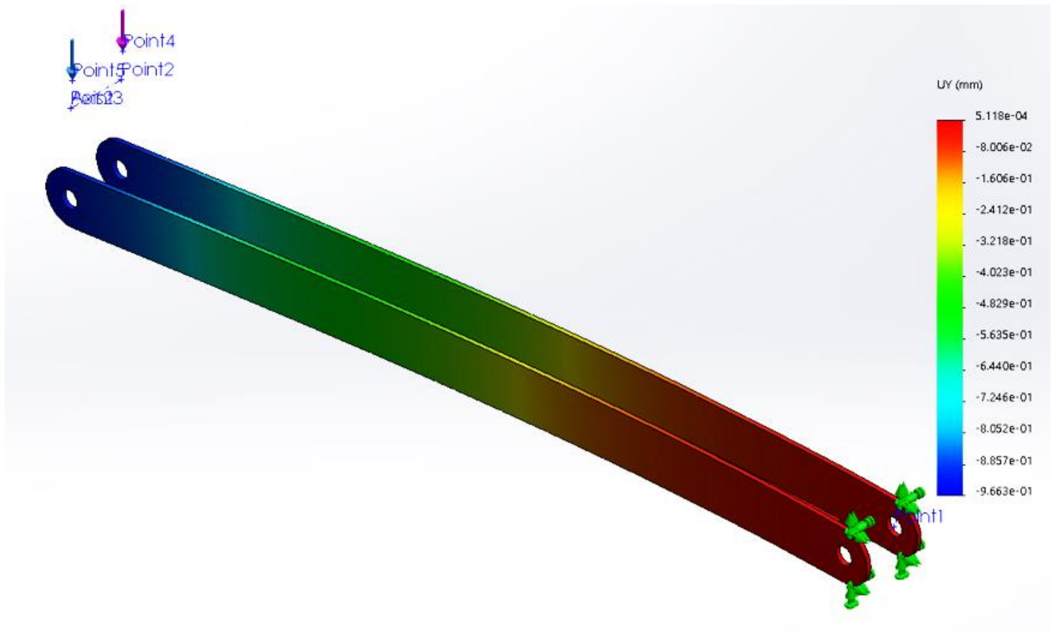


Figure 4.13 (a): Deflection

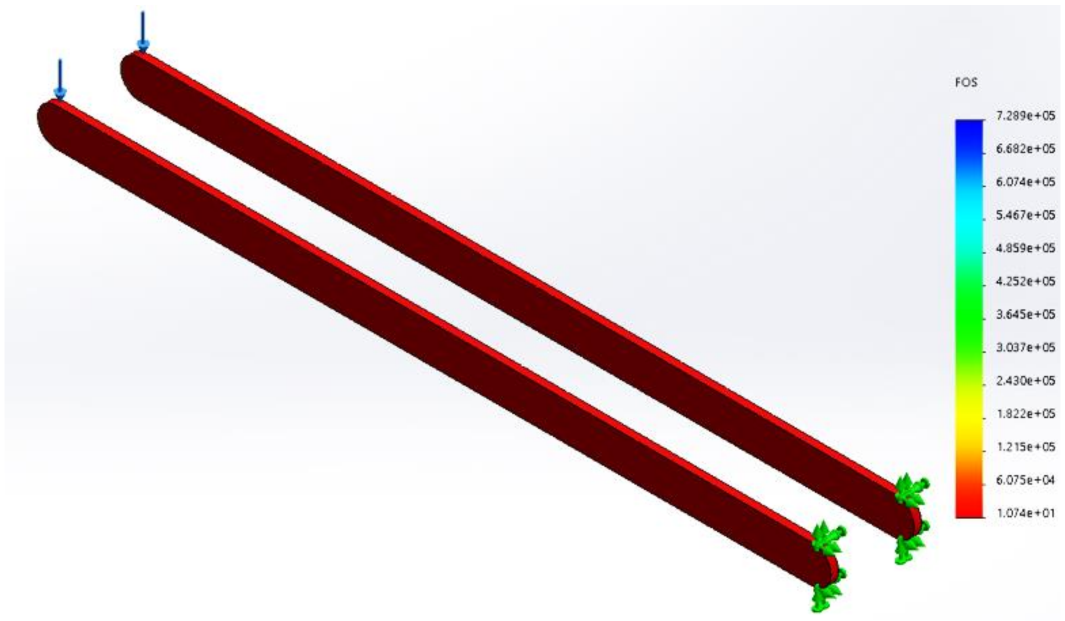


Figure 4.13(b): FOS

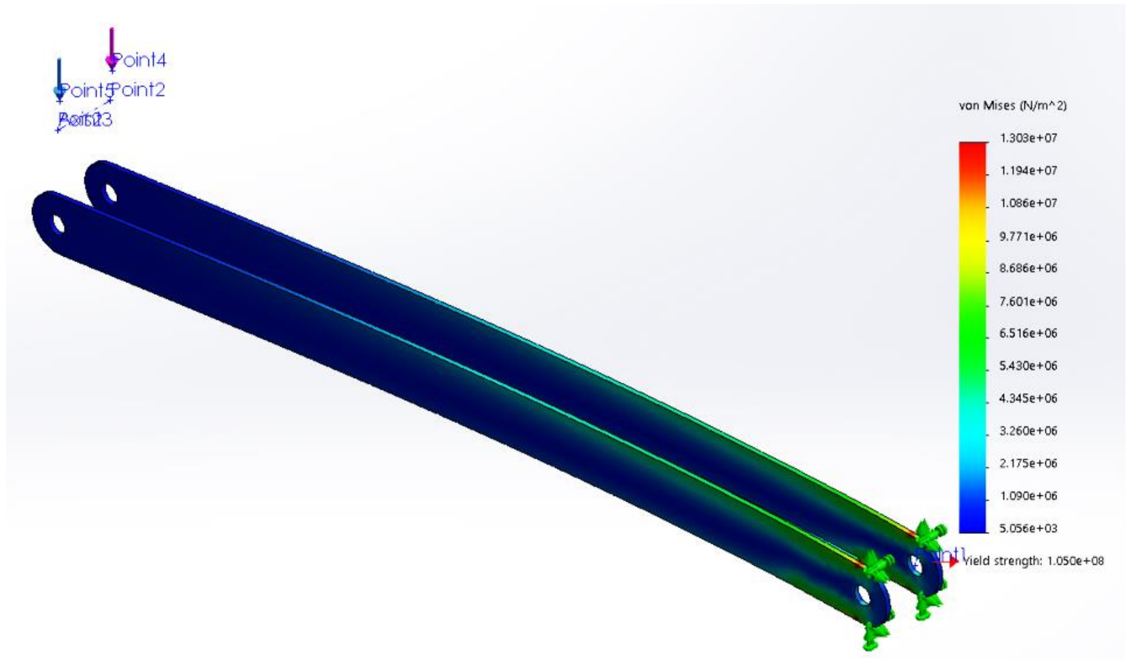


Figure 4.13(c): Von-Mises stress

Figure 4. 13: Result using SOLIDWORKS for 1060 Alloy (a) deflection, (b) Factor of Safety (FOS) and (c) Von-mess stress.

The basic objective in structural analysis and design is to produce a structure capable of resisting all applied loads without failure during its intended life. The primary purpose of a structure is to transmit or support loads. If the structure is improperly designed or fabricated, or if the actual applied loads exceed the design specifications, the device will fail to perform its intended function. The criteria used to judge whether particular proportions will result in the desired behavior.

Through the calculation of torque, motor power, transmission ratio, strength (bending, yield, tensile), and safety factor, the design of the robot arm such as what material and machine elements should be used to build the robot arm is achieved.

Chapter Five: Selection of Electrical Parts and Components

5.1 Introduction

This chapter introduces the comparison, design and selection of the system actuators, drivers and sensors, and explaining the motors design, electric drives, encoders, sensors and controller. Different challenges must be considered when selecting the electrical parts, such as delivery time by online markets, experiments of used parts from local market to identify their capabilities.

5.2 Comparison and selection of system actuators

In order to actuate each joint, many actuators can be used, such as electrical motors, hydraulic actuators and even pneumatics ones. The pneumatic actuators will be skipped because pressure losses and air's compressibility make pneumatics less efficient than other actuators, also precise control of speed isn't possible.

5.2.1 Comparison between actuators

This section introduces shortly the specifications of each the electrical and hydraulic actuators, then it compares between them in order to meet motion and control requirements, as speed, force, accuracy, repeatability and efficiency.

Hydraulic actuators:

Hydaulic linear actuators consist of a piston inside a hollow cylinder. Pressure from an external pump through an incompressible liquid (as oil) moves the piston inside the cylinder. As pressure increases, the cylinder moves along the axis of the piston, creating a linear force. The piston returns to its original position by either a spring-back force or fluid being supplied to the other side of the piston, see Figure 5. 1,

Figure 5. 1: Hydraulic linear actuators.

Hydraluics actuators has many advantages and disadvantegs, they will be discussed shortly. Advantages:

1. Hydraulic actuators are rugged and suited for high-force applications. They can produce forces 25 times greater than pneumatic cylinders of equal size. They also operate in pressures of up to 4,000 psi.
2. Hydraulic motors have high horsepower-to-weight ratio by 1 to 2 hp/lb. greater than a pneumatic motor.
3. A hydraulic actuator can hold force and torque constant without the pump supplying more fluid or pressure due to the incompressibility of fluids.
4. Hydraulic actuators can have their pumps and motors located a considerable distance away with minimal loss of power.

Disadvantages:

1. Hydraulics will leak fluid, and loss of fluid leads to less efficiency. However, leakage leads to change in control performance that designed before, cleanliness problems and potential damage to surrounding components and parts.
2. Hydraulic actuators require many companion parts, including a fluid reservoir, motors, pumps, release valves, and heat exchangers, along with noise-reduction equipment. This makes for linear motions systems that are large and difficult to accommodate.

Electrical Actuators:

An electric actuator is a electromechanical device used to convert electricity into kinetic energy. The input electrical power of voltage multiplied by current is converted to output mehanical power of speed multiplied by torque on the motor’s shaft. The two possiple types of motions are rotary and linear, where linear actuators convert the power into straight line motion, and rotary actuators alter energy to provide rotary motion. We are considered with the rotary electrical motors, see Figure 5. 2.

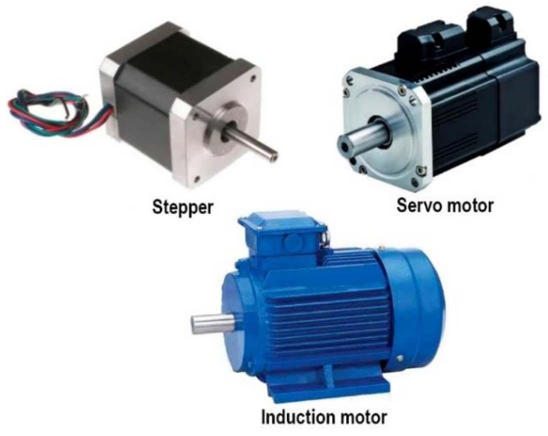


Figure 5. 2: Many types of electrical actuators.

Advantages

1. Electrical actuators offer the highest precision-control positioning. An example of the range of accuracy is ± 0.000315 in. and a repeatability of less than 0.0000394 in. Their setups are scalable for any purpose or force requirement, and are quiet, smooth, and repeatable performance.
2. Electric actuators can be networked and reprogrammed quickly. They offer immediate feedback for diagnostics and maintenance.
3. They provide complete control of motion profiles and can include encoders to control velocity, position, torque, and applied force.
4. The output torque can be increased easily with high gear reduction ratios that give high torques with less speeds, and these gears are available with many sizes and ratios.
5. In terms of noise, they are quieter than hydraulic actuators
6. Because there are no fluids leaks, environmental hazards are eliminated.

Disadvantages:

1. The initial unit cost of an electrical actuator is higher than that of pneumatic and hydraulic actuators. According to the example from Bimba Manufacturing, an electrical actuator can range from $150 to greater than $2,000 depending on its design and the required electronics.
2. A continuously running motor will overheat, increasing wear and tear on the reduction gear. The motor can also be large and create installation problems.
3. The motor chosen locks in the actuator’s force, thrust, and speed limits to a fixed setting. If a different set of values for force, thrust, and speed are desired, the motor must be changed.

5.2.2 Selection of actuators

The hydraulic actuators are not available in the appropriate size and weight in the local market for our Intelligent Excavator Prototype, and the hydraulic actuators require many companion parts. Hydraulics present certain advantages over electrical, but in our prototype, we don’t need high-force applications as the real soil excavators.

Electric actuators can provide superior performance over hydraulic actuators, in our application; for the excavator prototype, we look at the actuator of the higher accuracy and repeatability, suitable size with upgradability (utilizing of gear ratios), programmable control of motion parameters and better data collection with reporting capabilities and availability in market. All these interests are existed for many types of electrical actuators. As a consequence, the electrical actuator is selected because it achieves our requirements. It is worth to compare between the various types of the electrical actuators in terms to many aspects, this allows to choose the appropriate motor/s for our application. Table 5.1 compares between some of DC motors, such as: Servo, Stepper and Brushless motors:

Characteristics	Servo Motors	Stepper Motors	Brush Motors
Positioning	Precise motion control	Very Precise positioning	With special driver, we can maintain a precise position
Speed & Torque	High Torque at high Speeds (>2000rpm)	- Precise speed control. - Excellent torque characteristics at low speeds: achieve maximum torque at low speeds (< 2000 rpm)	High speeds. Same speed-torque characteristics as the typical DC Motors.
Controlling	Easy to control, provided with feedback position Control (PI), Smart (Contain Rotary encoder)	Easy to control, through a driver by sending pulses doing fixed angle increments/small steps	Using encoder sensors to directly measure the position of the motor shaft.
Range of Motion	Two types: - Positional rotation servos; 180 deg. (Position Control). - Continuous rotation servos; CW/CCW (Position & Speed Control)	Continuous Rotation.	Continuous Rotation.
Braking	It is not needed; braking is done by the Control itself	It is needed, some drivers give this option.	It is not needed; braking is done by the driver.
Limitations	Limited range of motion for Positional rotation servos (180 deg.)	- have less torque at high speeds than at low speeds. - they might skip steps at high loads.	Controller: requires a specialized regulator/driver for position, speed or torque modes.
Efficiency	High	Low to Moderate, due to high current consumption, so it tends to become hot.	High, as they are able to continuously achieve maximum rotational force/torque and acceptable speeds.
Power Consuming	Low to Moderate	power-hungry device, draws maximum current	Low to Moderate, according to the required speed and torque.
Temperature in Operation	low	High, especially for long time operation	Moderate.
Cost	Cheap at all, especially for small sized (>20$)	Moderate to High, according to size and load characteristics (>35$)	Moderate to High, according to size and load characteristics (>30$)
Pros	Precise & Smooth movement	Excellent torque to maintain position; Suitable for applications with high holding torque.	- Quiet less electrical noise. - less electrical connections.
Cons	- Large scall and Continuous rotation servos increase cost.	- Generate some noise during operation.	- more expensive and complex - less durability as there are brushes to be replaced.

Table 5. 1: comparison between some types of DC motors.

5.3 Electrical Parts

The excavator electrical parts consisting of sensors, motors, interfacing and driving circuits that allow connection between these parts and the system control unit. They are described as follows:

5.3.1 Robot motors

Selecting a suitable motor to actuate a joint has many considerations. when selecting the motor, we have to take into our consideration the three specifications: power, torque and speed. For the mechanical and electrical specifications as the power, current and voltage, the electrical power should be greater than the mechanical (i.e., the torque for the selected motor should be larger than the calculated required torque at the joint). Figure 5. 3, shows the forces acting on the manipulator.

For motor selection, the second motor is the critical one because it required to move the whole manipulator (i.e., The largest torque is applied on it). The others motors will have less torque. Torque at each joint will be calculated as below,

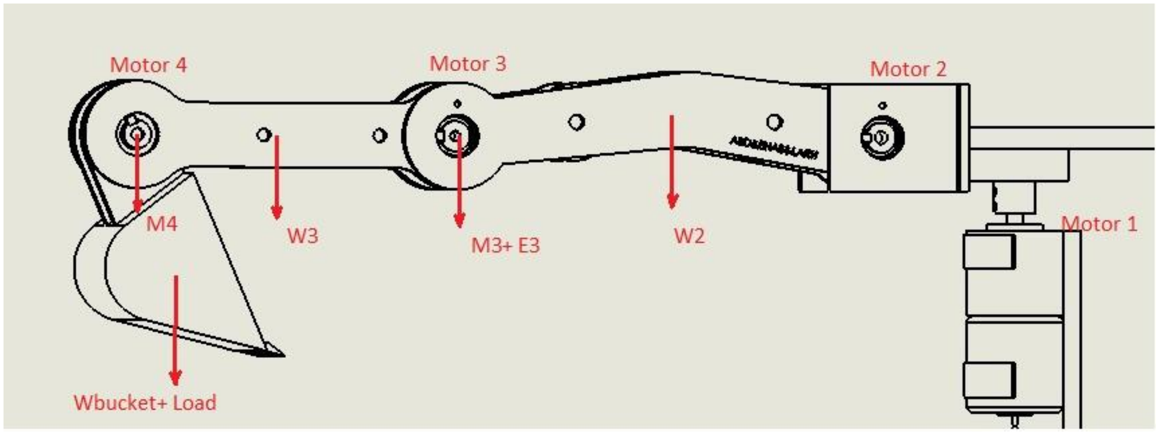


Figure 5. 3: Forces acting on the manipulator

Finally, after reselecting the links material to be the Aluminum, the soil weight is changed instead of the previous 5Kg to be 0.5Kg which is suitable for the material strength with reasonable deflections (see Chapter 4). Equation (5.1-5.3) represent the maximum torques for each joint,

$$\tau_4 = (B + L) * \frac{L_4}{2} = (0.5 + 0.215) * \frac{0.1}{2} * 9.81 = 0.35\ N.m \qquad (5.1)$$

$$\tau_3 = w_3 * \frac{L_3}{2} + (J_4 + L + B) * L_3$$

$$= 0.253 * \tfrac{0.15}{2} * 9.81 + (0.072 + 0.5 * 0.215) * 0.15 * 9.81 = 1.344\ N.m \quad (5.2)$$

$$\tau_2 = w_2 * \frac{L_2}{2} + (J_3 + e_3) * L_2 + W_3 * \left(L_2 + \frac{L_3}{2}\right) + (M_4 + L_4 * B) * (L_2 + L_3) + M_2 * \frac{L_2}{2}$$

$$= 0.358 * \frac{0.2}{2} * 9.81 + (0.31 + 0.118) * 0.2 * 9.81 + 0.253 * \left(0.2 + \frac{0.15}{2}\right) * 9.81 +$$

$$0.787 * (0.2 + 0.15) * 9.81 = 4.3\ N.m \quad (5.3)$$

Where,

W_2 and W_3 are the weights of each link, the effecting point of those weights are at the center of each link.

J_2 and J_3 are the weights of each joint (motor with shaft).

M_3 and M_4 are the weights of motors.

E_3 is the weight of encoder 3.

τ_1, τ_2 and τ_3 are the maximum torques on each joint.

B is the weight of the End-Effector (Bucket).

L is the weight of the Load that the arm will lift

5.3.2 Motor drivers

The motor driver is a device allows interfacing between the motors, power and system microcontroller. The control signal from the microcontroller has low power while the motor needs high power to be driven properly, so the driver amplifies the input control signal with new characteristics to the motor to provide the required power of actuating. The driver has many other functions, as position, speed and torque control modes. Also, driver protects the microcontroller from reverse current that comes from the motor.

5.3.3 Sensors

This application requires many types of sensors to satisfy the required tasks, such as:

1. **Position sensor**

Define position through the generation of a signal that provides positional feedback to achieve the required trajectory planning. It is used to sense the links orientation and direction of rotation, such as encoder sensor that can provide precise information for the position, direction and speed of a piece of mechanical equipment which is used to know a system state at that moment.

2. **Limit switch sensor (mechanical sensor)**

Defines the limit or endpoint over which an object could travel before being stopped. At a desired point the switch will be engaged to give a signal that used to control the stroke of the movement. Two limit switches are needed for each joint to indicate the start or end of the joint angular movement (see section 3.2.2 about joint limits), so it is needed at least 8 limit switches for that purpose. Another function of these sensors is the homing process to initial positions of the robot links at each operation.

5.3.4 Human machine interface

Touch screen

It is an input and output interaction device that used for an information processing system, allows to interact with and command other functions. It can be useful to show diagnostics and system status with monitoring. The screen can be used as output to display joint positions and limits, torques, consumed power, faults and errors, and used as input to command the manipulator to perform a predetermined (recorded) motions or trajectories to the joint actuators.

Joystick

It is an input device consisting of a stick that pivots on a base and reports its angle or direction than can be sent to the microcontroller, it has at least 4 motions that can be reported with many buttons. The Joystick is essential in controlling the manipulator motion, two Joysticks are needed for the whole manipulator. The joysticks are used in manual-mode operation when the user wants to control the excavator manually, each joint position can be controlled in CW/CCW direction independently, the bucket excavation and throwing motions and other features.

5.3.5 Power source

Choosing the appropriate power supply demands on how much rated power is needed for the electrical system components to work well. The overall required power should be calculated carefully to allow the power source selection in terms to satisfy and perform the manipulator tasks correctly, such as motion trajectories that performed by actuators, feedback data from sensors and the required microcontroller power, etc.

5.4 Components selection

5.4.1 Motors selection

First and second joints:

Excavator robot requires four motors with gearboxes, Figure 5. 4, shows the selected motors for the first and second joints according to the maximum torque at each joint. The DC brushed motor is selected because it achieves maximum torques with acceptable and moderate speeds, a gear reduction ratio is used for torque amplification.

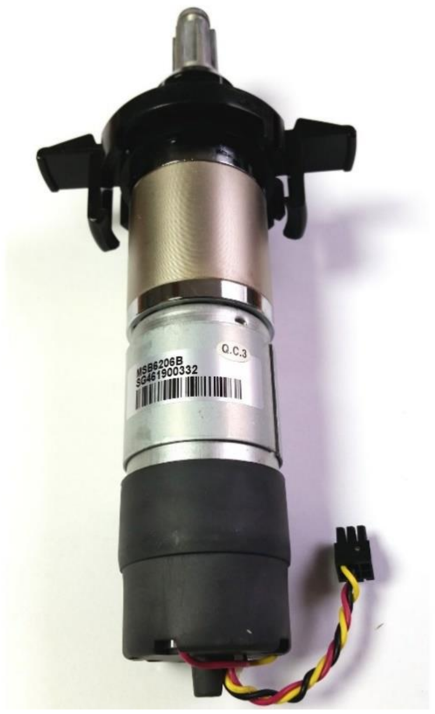


Figure 5. 4: The selected DC brush motor for joints 1 and 2

Table 5. 2: It contains the parameters of the chosen DC motor for joints 1 and 2.

Model name	SMSB6206B
Input Voltage	12 – 24 VDC
No load speed	40 RPM
No load current	1.0 A
Maximum load current	about 5A
Gear ratio	70:1
Rated torque	7 N.m
Mass	500 g
specifications	Dual shaft (for encoder/ gearbox), motor shaft: 8mm diameter

Third joint:

This joint requires less torque, and a less motor weight because it is hanged and its weight efforts a torque on the previous joint. See Figure 5.6, and table 5.3 lists its specifications.

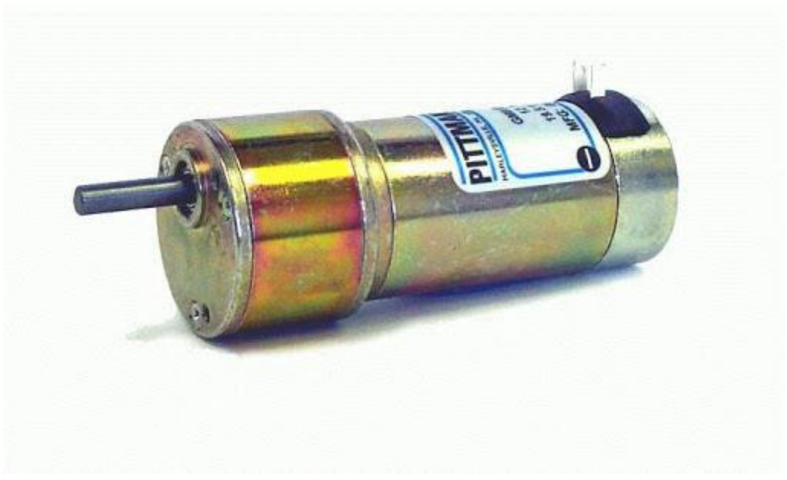


Figure 5. 5: The selected DC brush motor for joint 3

Table 5. 3: It contains the parameters of the chosen DC motor for joint 3.

Model name	GM8224S023
Input Voltage	12 – 24 VDC
No load speed	75 RPM
No load current	0.5 A
Maximum load current	about 4A
Gear ratio	60.5:1
Rated torque	3 N.m
Mass	250 g
specifications	Dual shaft (for encoder/ gearbox), motor shaft: 4mm diameter

Gearbox needs

The available gear ratios are 70:1 for joints 1 and 2, and 60.5:1 for joint 3, the gear reduction ratio allows to increase torque and reduce speed. When higher gear reduction ratios are used, it reduces the nonlinearities and the coupling effects of the system dynamics.

These motors are selected to provide the maximum required torques, in local market these motors are the best and most suitable for our application. The design, dimensions and materials are changed according to these motors. It is motivating that these motors provides more than the required torques for the new design which give robust and strong torque control to pay a load and transfer it.

Fourth joint:

The last joint has a small torque values and a limited task of excavation to open/close the bucket, so a Servo-motor is chosen to perform that task. The servo gives a precise position with a suitable handle torque to resist the load on its shaft, a gear transmission is added to this motor model which gives higher torques. The Figure 5.7 shows the selected servo-motor, with a listed features in table 5.4.

Figure 5. 6: The selected DC servo-motor for joint 4

Table 5. 4: It contains the parameters of the chosen DC servo-motor for joint 4.

Model name	MG996R
Input Voltage	4.8 - 7.2 VDC
Stall torque	9.4 kgf·cm (4.8 V), 11 kgf·cm (6 V)
Operating speed	0.17 s/60º (4.8 V), 0.14 s/60º (6 V)
Running Current	0.5A – 0.9A, stall: 2.5A (6V)
Mass	55 g

5.4.2 Driver selection

Each motor requires a special driver for the desired purpose, the first 3 motors will be controlled in close-loop fashion, while the last motor has its own controller. Three motor drivers called SOLO UNO are selected to control each of the first three motors, and an H-bridge driver will be used for the servo-motor for amplifying the control signal.

First three joints:

SOLO UNO is able to drive a brush and brushless DC motor, and it provides position, velocity and torque control modes in closed/open loop fashions, see Figure 5. 7, and it is compatible with Raspberry Pi microcomputer with especial library and numerous functions. Each board supports control of 1-motor with reading 1-encoder.

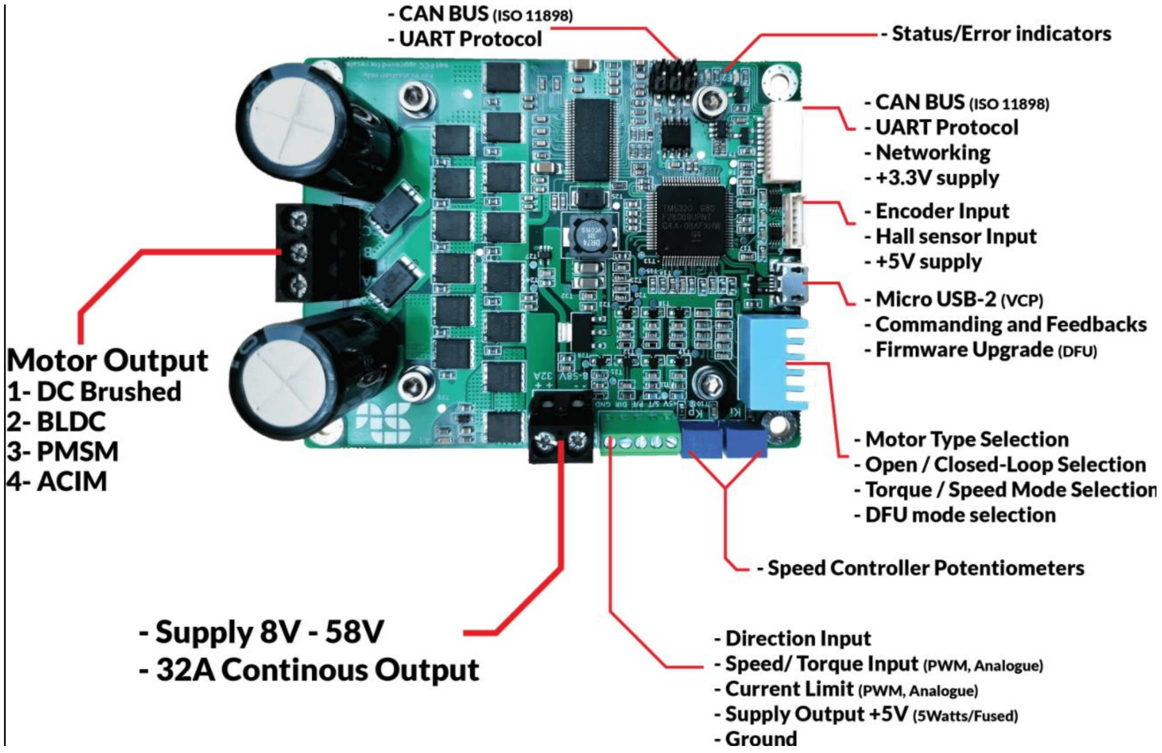


Figure 5. 7: SOLO UNO Driver with described ports

The driver has many features, the most usable are the following:

- Wide input voltage supply range (from 8V to 58V)
- The continuous output current of 32A, Max Current of 100A
- Consumes 0.5A at no load on motor, and same power for motor.
- Capable of controlling DC and BLDC motors.
- Dual Core with parallel processing architecture
- Selectable output PWM switching frequency (from 8kHz to 80kHz)
- Open-loop or Closed-loop Control modes

- Speed, Torque or Position control
- PWM and Analogue voltage input for Controlling Speed and Torque
- Many methods to design and execute the controllers in Hardware or Software fields
- USB, UART, and CAN Bus (ISO 11898) Communication enabled
- Encoder and Hall Sensor Input with +5V supply
- Weight of only 99 g.
- Can be accessed via Python or C++ libraires.

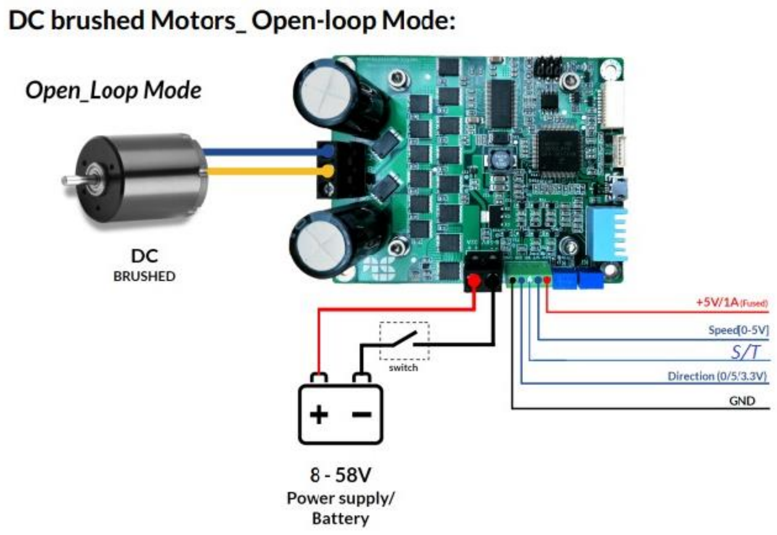


Figure 5. 8: Open-loop control through PWM signal to S/T terminal

In practice, driver has open/closed loop torque control modes, see Figure 5. 8 for open-loop control by sending PWM control signal to S/T terminal. Initially, each motor is tested with the open-loop control, through commanding the stall torque to each driver to actuate its coupled motor with the maximum load, and observing the current consumption via UART connection, the results were verified that drivers are capable of providing the desired power to each motor.

Fourth joint:

The servo-motor is controlled through the PWM signal to obtain the desired position. It is convenient to use an amplifier or H-bridge to provide the desired power, the l293D H-bridge is suitable for that action, see Figure 5.9 shows the H-bridge with its terminals, and Table 5.5 lists its specifications.

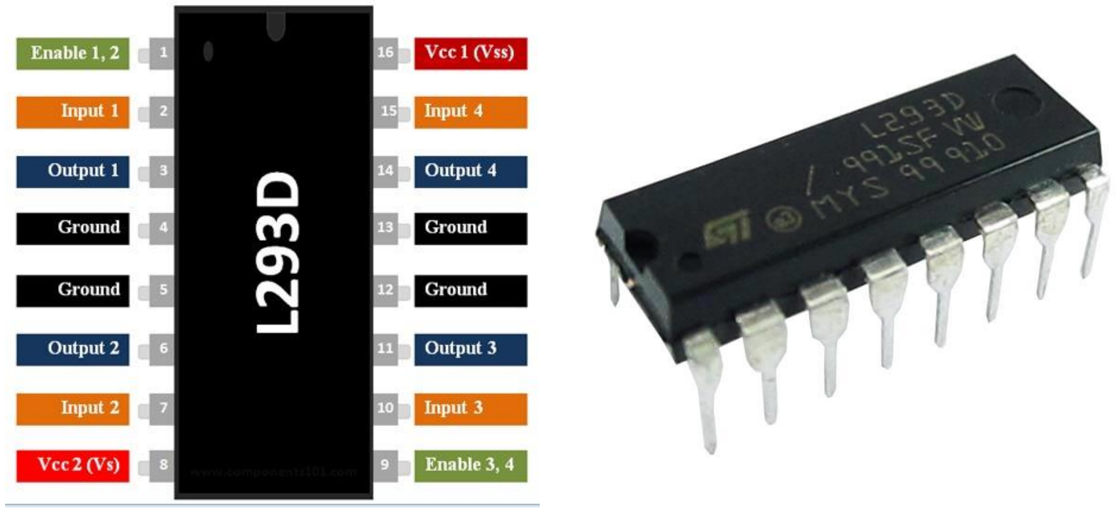


Figure 5. 9: l293D H-bridge with terminal names.

Table 5. 5: It contains the specifications of the selected H-bridge.

Model name	L293D
Supply and output Voltage	Max of 36 VDC
Maximum continuous current	1.0 A
Controlled channels	2 isolated channels with two directions for each

5.4.3 Sensor selection

1. Position sensor

Encoder sensors are a type of mechanical motion sensors that create a digital signal by detecting a motion. There are different categories of encoders, one of them is the optical encoder which is a type of rotary encoder, it is an electro-mechanical device that provides users (commonly those in a motion control capacity) with information on position, velocity and direction. Three encoders are selected for the first three joints.

For the first and third joints, two incremental encoders of 400-pulse resolution are used to measure the angular motion of these two joints (coupled with the motor gear shaft. Another encoder of 2048-pulse is adopted for the second joint, the reason of choosing this encoder with more pulses is to give a better resolution for joint 2 measurement; because the second motor gear has about 3^o of backlash and the joint dynamics are fast and of high interest, for other details see Figures 5.10 with table 5.6, and 5.7,

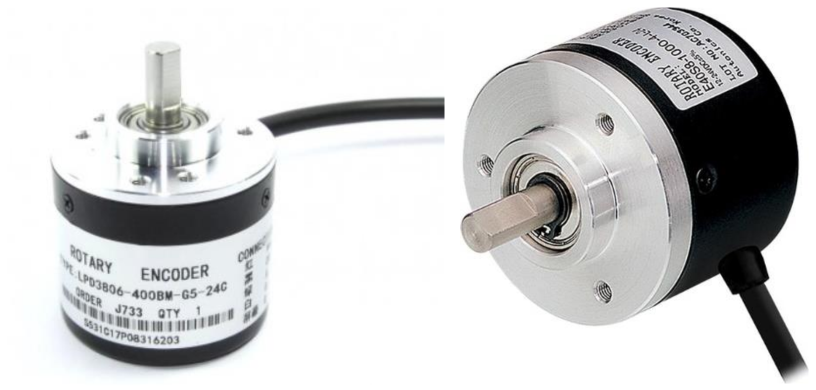


Figure 5. 10: The used rotary encoders: 400-pulse resolution (left), 2048-pulse resolution (right)

Table 5. 6: It shows the specifications of the 400-pulse encoders.

Model	LPD3806-400BM-G5-24C
Supply voltage	DC5-24V
Draw Current	30mA
channels	A, B, and Z index
Performance	400 pulses / rev
Accuracy	0.9 degree/pulse
Maximum mechanical speed	5000 rev / min

Table 5. 7: It shows the specifications of the 2048-pulse encoder.

Model	E40S6-2048-6-L-5
Supply voltage	DC5-24V ±5%
Draw Current	80mA
channels	A, B, and Z index
Performance	2048 pulses / rev
Accuracy	0.18 degree/pulse
Maximum mechanical speed	6000 rev / min

Encoder readers

Each encoder has an independent hardware counter, reading will be counted using the chip LS7366R 32bit counter which counts the pulse of channel A and B by its own hardware, the encoder reading (pulses count) is transferred through the SPI protocol to the Raspberry Pi, this encoder reader with the interfacing protocol provides a high reading and transferring rates of above 1MHz. see Figure 5.11, and table 5.8 for details.

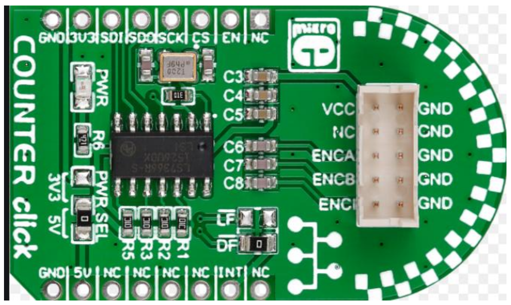


Figure 5. 11: 32-bit encoder reader

Table 5. 8: Shows the specifications of the 32-bit encoder counter.

Model	MIKROE-1917
Supply voltage	3.3V or 5V DC
Draw Current	15mA
Ports	SPI Port, A, B, and Z encoder ports
Hardware chip	LS7366R counter
Resolution	32-bit
Supported software	Python libraries

2. Limit switch

Each joint of the first 3 joints will have at least 1 limit switch to detect the lower limit or homing position. The selected limit switch is shown in Figure 5. 12

Figure 5. 12: limit Switch.

Table 5. 9: limit switch sensor specifications.

Brand	Twidec
Current rating	0.3A
Weight	45g
Switch Type	Limit Switch

5.4.4 Human machine interface

Raspberry Pi screen

The touch RPI screen is a display device that allows the operator to interact and command. It registers the event and sends it to the system controller for processing, it is used for control development and testing and connected through HDMI cable. The advantage is to have one touch screen instead of several buttons, and many outputs can be seen, one can input the references to the control algorithm and see the response data as the error, current position, effort and time, see Figure 5. 13.

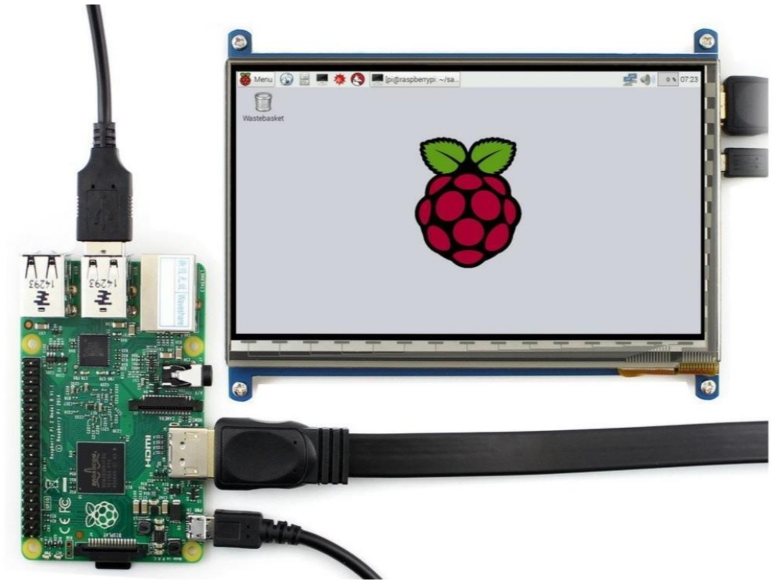


Figure 5. 13: Raspberry Pi screen.

Table 5. 10: RPI screen specifications.

Model name	7-inch HDMI display
compatible with	Raspberry Pi 4 3B+ 3B 2B+, DIRECTLY plugged.
Power	5V/ 2A
Resolution	1024×600
specifications	I/Os for touch function and power supply (USB), HDMI interface for displaying.

Joystick/ Bluetooth keyboard

The joystick is an input device that can be used to control the excavator movements in manual mode operation. One of the noticeable advantages of the joystick is its ability to provide fast interactions which is better in quality compared to that provided by other input devices. It has a simple design and is easy to learn and use, and often inexpensive.

In practice, the Bluetooth keyboard is used instead of the Joysticks. Keyboard provides direct and easiest interfacing with Raspberry Pi through USB-Bluetooth adapter, but the available Joysticks needs many analog ports to interact with, where RPi has no analog inputs, since there is no need to make this topic more complex in wiring and coding, keyboard is a suitable and flexible choice for many purposes.

5.4.5 Power source

Supplies electric power to an electrical device (load) by deliver output current and voltage requirement. The current consumption of the whole system electrical components is listed in Table 5.8:

Table 5. 11: System electrical components with current consumption

Current for the 4 motors (A)	$2 * 5A + 4A + 2.5A = 16.5A$
3 Encoders with their readers	2*0.03A + 0.08A + 3*0.015A = 1.005A
3 Drivers (at no load on motors)	3 * 0.5A = 1.5A
3 limit switches	$3 * 0.3A = 0.9A$
H-bridge	1.2A
Display	2A
Summation	23A

A PC power source is selected that provide too many channels of different Voltage and Current combinations. The power source, see Figure 5. 14 , gives max current of 30A, the mainly used terminals are 12V/17A, 5V/25A, 5V/2.5A each one is used according to the previous power requirements.

Figure 5. 14: Power supply

Table 5. 12: Shows the selected power supply specifications.

Brand	Xunba
Output voltage	12V
Output current	30 A
Input voltage	AC 85-220 V
power	360 Watts
Frequency	50/60 Hz

Chapter Six: Information Processing

6.1 Introduction

This chapter introduces information processing requirements, such as software, interfacing, microcomputer, performance, processing speed and selection of suitable components and hardware connections.

6.2 Software and information processing

After discussing many hardware components, it is essential to consider the software in terms to many aspects that related to the needed processes and tasks to performed efficiently. This robot is vision based, and robot is supposed to have the ability to learn how to perform many tasks and interact with the surrounding environment by machine learning algorithms, however this will be performed in future work. A programmable manipulator has to be designed to move the structure, and to have specialized devices to execute various of programmed motions. A software may be adopted to coordinate, manage and execute such processes and control algorithms which is the Robot Operating System (ROS), this software needs a Linux operating system, such as Raspbian or Ubuntu.

As a consequence, basically MATLAB software with PC will be used for the first levels in prototyping, and in control system development, since PC has powerful computational performance. We aspire to develop an embedded system which contains its independent controller. The embedded system controller must specify the processing capability in presence of information processing, control architecture and supports operating systems as Raspbian, therefore the suitable choice is Raspberry Pi (RPi) microcomputer which is widely used in such application due to its powerful processing performance and specifications, see Figure 6. 1.

Human-excavator interaction that satisfied by human machine interface which shows system status and allows the operator to command through the screen or Joysticks. the commands are translated to desired tasks that are analyzed and broken down into a sequence of actions by the system processing unit (CPU) and stored. The choice of actions is performed on the basis of knowledge models that stored in the system memory.

All these processes form many specifications of the information processing components (PC, RPi) that must be taken into consideration. The following features are suitable for our embedded system being capable of performing variety of tasks,

1. A fast CPU, at least 4 cores with frequency of 1500Hz.
2. Large amounts of RAM. For best processing capacity, RPi supports versions of 4 or 8GB RAM.
3. Can support programming languages and software, especially MATLAB and SIMULINK for simulations (PC), and Python or C++ for RPi with practical work.
4. Human machine interface as RPi screen, see section 5.3.4.
5. Capable of interfacing with many hardware components, such as motor drivers, feedback devices and sensors for the RPi.
6. Many input/output peripherals, such as digital and PWM channels, and other data exchanging protocols as SPI and UART.

Other specifications that may be necessary for future work:

- Graphics card for image processing and vision.
- Storage needed for ROS.

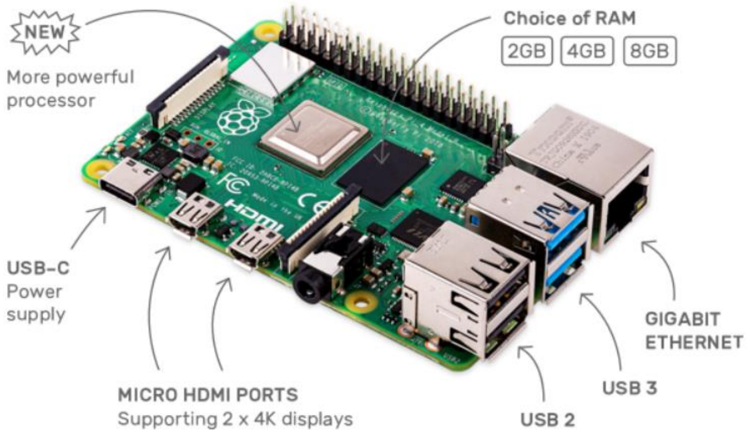


Figure 6. 1: Raspberry Pi 4 microcomputer

Figure 6. 2 shows the mechatronics design levels of Hardware and software integration. The hardware level (integration of components) consists of RPi microcomputer, four electrical actuators, the manipulator (process) and sensors (3 encoders, 3 limit switches). The RPi transmits control signals to motor drivers and feedback data from sensors. Drivers outputs the actuating signals to the motors which actuate the joints (process).

Software level (PC) contains the development, design and simulation of control algorithms with the derived mathematical model as a knowledge base. These functions will be tested and redesigned to be performed in real applications on the RPi as a stand-alone code. It contains control process includes the calculation of the control efforts (joint torques) that are required to actuate the joints to track desired trajectories.

The supervision diagnosis improves the system performance through observing the system states, error detection capability and gathering information about system faults, such as computational errors of inverse kinematics, wrong reference inputs of joint trajectories out the joint limits, etc. Such diagnosis shows error on HMI, recovers the correct operation functions and determines the root(s) of the output of control status. Process fault diagnosis involves interpreting the current status of the plant given sensor readings and process knowledge.

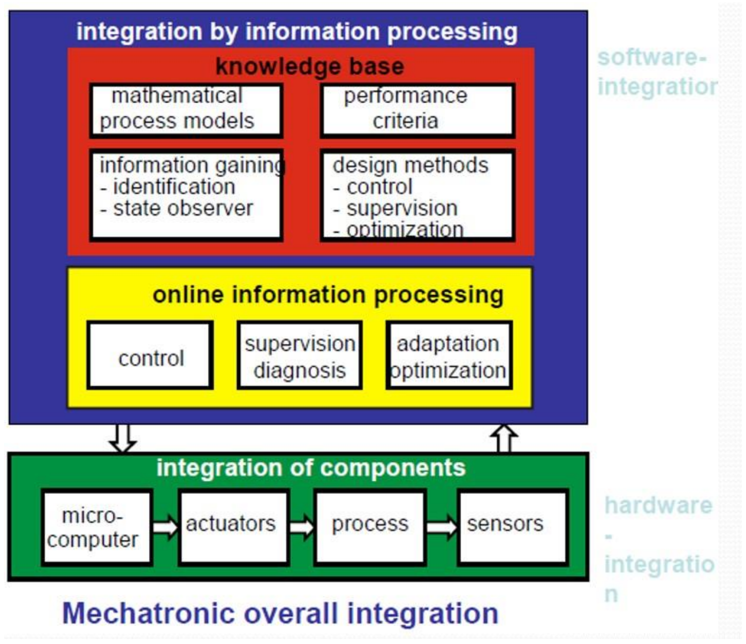

integration by information processing
knowledge base
mathematical process models
performance criteria
information gaining - identification - state observer
design methods - control - supervision - optimization
online information processing
control
supervision diagnosis
adaptation optimization
integration of components
micro-computer
actuators
process
sensors
software-integration
hardware-integration
Mechatronic overall integration

Figure 6. 2: Mechatronics system integration [12]

6.3 Processing unit

Processing unit must be selected to satisfy many considerations, as processing power (CPU, RAM, Storage, etc.), general purpose I/O ports, compatibility with other devices (PC, motor drivers, HMI) and a convenient software environment. In this work, the prototyping is based on the Raspberry Pi platform, where RPi 4 model B microcomputer with 4GB RAM version (see Figure 6. 1) is selected for further work with embedded system developments, the analysis and computation resources, since it covers all the above-mentioned requirements.

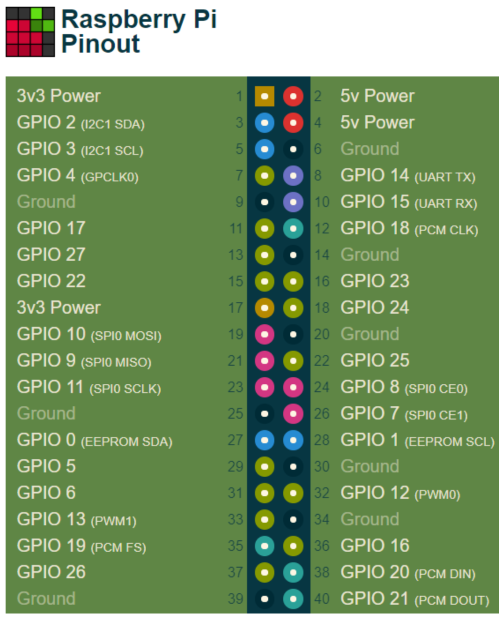
Levels of information processing

There are two levels for information processing using RPi. Firstly, the high level consists mainly of RPi computer that will be utilized as the information processing unit, it will be operated and accessed through an Ethernet protocol interfaces it with PC, or even connecting it with HDMI touch screen for further developments and interaction. RPi treats with the low-level components through its GPIO peripherals, the low level contains the electrical components as sensors, motors and drivers and mechanical parts as mechanical structure and process. Finally, RPi forms a complete stand-alone system that controls, process, sends and receives data from/to the electrical components with a Human-machine interface.

6.4 Hardware connections and interfacing between components

An interface is a shared boundary across which separate components of a system exchange information. The exchange can be between software, computer hardware, peripheral devices, humans, and combinations of these. RPi has various and useful interfacing protocols, like SPI and UART, and many I/O ports of digital and PWM functionalities. It is worth to mention that the selected motor drivers, encoder readers and the display (HMI) are all compatible with RPI in terms to hardware and software. This option allows more easier programming, control and interfacing.

Raspberry Pi 4 has 40 GPIO pins, see Figure 6.3



Raspberry Pi Pinout

3v3 Power	1	2	5v Power
GPIO 2 (I2C1 SDA)	3	4	5v Power
GPIO 3 (I2C1 SCL)	5	6	Ground
GPIO 4 (GPCLK0)	7	8	GPIO 14 (UART TX)
Ground	9	10	GPIO 15 (UART RX)
GPIO 17	11	12	GPIO 18 (PCM CLK)
GPIO 27	13	14	Ground
GPIO 22	15	16	GPIO 23
3v3 Power	17	18	GPIO 24
GPIO 10 (SPI0 MOSI)	19	20	Ground
GPIO 9 (SPI0 MISO)	21	22	GPIO 25
GPIO 11 (SPI0 SCLK)	23	24	GPIO 8 (SPI0 CE0)
Ground	25	26	GPIO 7 (SPI0 CE1)
GPIO 0 (EEPROM SDA)	27	28	GPIO 1 (EEPROM SCL)
GPIO 5	29	30	Ground
GPIO 6	31	32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33	34	Ground
GPIO 19 (PCM FS)	35	36	GPIO 16
GPIO 26	37	38	GPIO 20 (PCM DIN)
Ground	39	40	GPIO 21 (PCM DOUT)

Figure 6. 3: GPIO pins for Raspberry Pi 4 – model B

6.4.1 Interfacing protocols

SPI - Serial Peripheral Interface

The Serial Peripheral Interface or SPI bus is a synchronous serial data link that operates in full duplex mode. In other words, data can be sent and received at the same time. Devices communicate in master/slave mode, where the master device (RPi) initiates the data exchange with one or more slaves. Multiple slave devices are allowed with individual slave select lines.

The encoders reading (pulses count) are transferred through the SPI buses to the Raspberry Pi. The main advantage in SPI that it exchanges the data with rates above 1MHz. Using this method will reduce the computations on the Raspberry Pi; because the reading of pulses is done by the board itself, and the counter library only receives the number of pulses and the rotation direction.

The SPI bus specifies four logic signals:

- SCLK: Serial Clock (a clock signal that is sent from the master).
- MOSI: Master Output, Slave Input (data sent from the master to the slave).
- MISO: Master Input, Slave Output (data sent from the slave to the master).
- CS: Chip select (sent from the master, active on low signal). Often paired with the slave Select (SS) line on an integrated circuit that supports SPI.

Figure 6.4 shows the conceptual connection for one SPI with many devices,

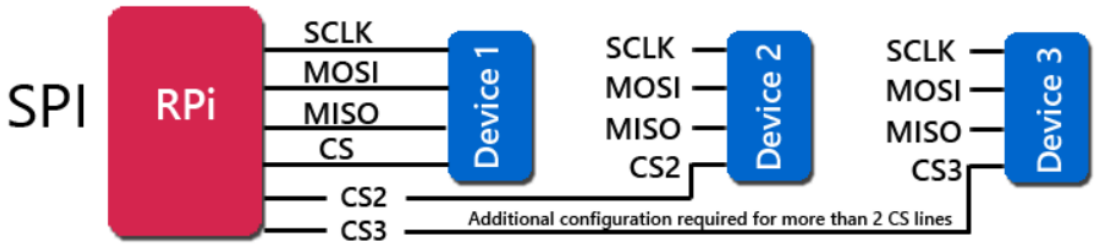


Figure 6. 4: RPi-SPI port connected to many devices with multiple select lines (CS) [13]

Three SPI ports have been used with the three encoder readers, all SPI connections are described in table 6.1,

Table 6.1: Encoder counters (1,2,3) pins to RPi SPI (0,1,3) connections:

Encoder counters pins	Raspberry Pi (Port No., GPIO pin)
SDI: (1/2/3)	MOSI: (0, 10)/ (1, 20)/ (3, 2)
SDO: (1/2/3)	MISO (0, 9)/ (1, 19)/ (3, 28)
CLK: (1/2/3)	SCLK (0, 23)/ (1, 21)/ (3, 3)
SS: (1/2/3)	CS (0, 8)/ (1, 12)/ (3, 0)

CS0 is used for each SPI port (one device at each SPI), it means that multiple SPI ports can be operated independently, this has an advantage for the control system that reads many feedback signals at the same time which provides fast knowledge with the system states.

Note: SPI2 is not exist in RPi 4.

UART - Universal Asynchronous Receiver/Transmitter

UART, or universal asynchronous receiver-transmitter, is one of the most used device-to-device communication protocols. In UART communication, two UARTs communicate directly with each other. The transmitting UART converts parallel data from a controlling device, transmits it in serial to the receiving UART, which then converts the serial data back into parallel data for the receiving device.

Only two wires are needed to transmit data between two UARTs, see Figure 6.5. Data flows from the Tx pin of the transmitting UART to the Rx pin of the receiving UART.

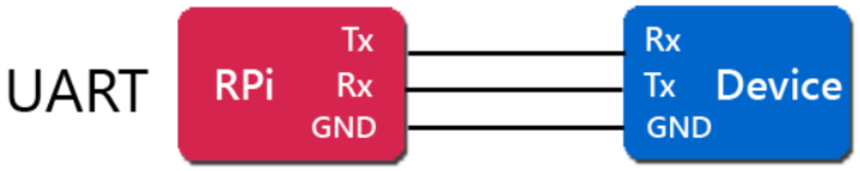


Figure 6. 5 : RPi-UART port connected to a device [13]

UART in RPi has many ports, UART0 is located on the following GPIO pins:

- Transmit (TXD): GPIO 14.
- Receive (RXD): GPIO 15.

These two pins can be connected to any driver with its UART port on pins 1, 3 sequentially.

Feedback communication protocols – capabilities and selection

To read the encoders is exist which is the SOLO driver itself. SOLO UNO has the ability to read one encoder for each. However, the reading must be transferred via UART protocol which has a speed of maximum 150KHz that is slower than the speed of SPI protocol (>1MHz), this may affect negatively on the control performance. This assumption is verified practically since the position reading through SPI gives smooth tracking and more stable performance. In contrast, using UART for controlling one joint makes the system more oscillatory.

Another crucial consideration, it was intended to use 3 UART ports with the Raspberry Pi to read the encoders with the SOLO drivers, it needs multiplexers with digital coding, many digital I/O pins and code complexity, this choice has been avoided for feedback devices after many tests on its usability and the control performance.

UARTs transmit data asynchronously, which means there is no clock signal to synchronize the output of bits from the transmitting UART to the sampling of bits by the receiving UART. In Raspberry Pi's, UART has limited data sizes and baud rates (max. speed up to 150KHz) compared to the SPI and I^2C capabilities. As a consequence, when fast transferring rates are needed, the UART may suffer a traffic in data exchanging which leads to many drawbacks in control systems. Finally, 3 SPI's have been used for feedback communication instead of UART's.

PWM – Pulse Width Modulation

The PWM is a technique widely used to regulate the speed of rotation of a DC motor, in this case, the duty cycle of the PWM is used to provide the torque necessary to the joints of the robot in order to look for a linear relationship but using the right frequency of the characterized DC motor.

The RPi 4 has two Hardware PWM pins located on GPIO 12 and 13, and many Software PWM pins. RPi pins give a voltage level of 3.3V while the SOLO driver treats with a logic level of 5V, therefore the amplifier (L293D) is used to amplify the RPi signal to 5V. The connection of PWM pins to SOLO drivers and servo motor is listed in table 6.2,

Table 6.2: RPi-PWM pins connected to drivers and servo motor: [14]

RPi PWM GPIO pins	Driver
Software PWM: 17	(Speed/torque reference): S/T1
Hardware PWM: 12	(Speed/torque reference): S/T2
Hardware PWM: 13	(Speed/torque reference): S/T3
Software PWM: 27	Servo motor PWM pin

Hardware PWM gives continuous and stable signal compared to the software; because Software PWM can be interrupted by any other process while Hardware PWM operates behind the CPU with its own module. The joints 2 and 3 have fast dynamics compared to other joints, it is favorable to use Hardware PWMs for these joints giving smooth and robust tracking.

6.4.2 Hardware connection (GPIO pins connection)

The Figure 6.6 shows the hardware connection between RPi and the actuation, feedback and other devices.

See next page for hardware connection.

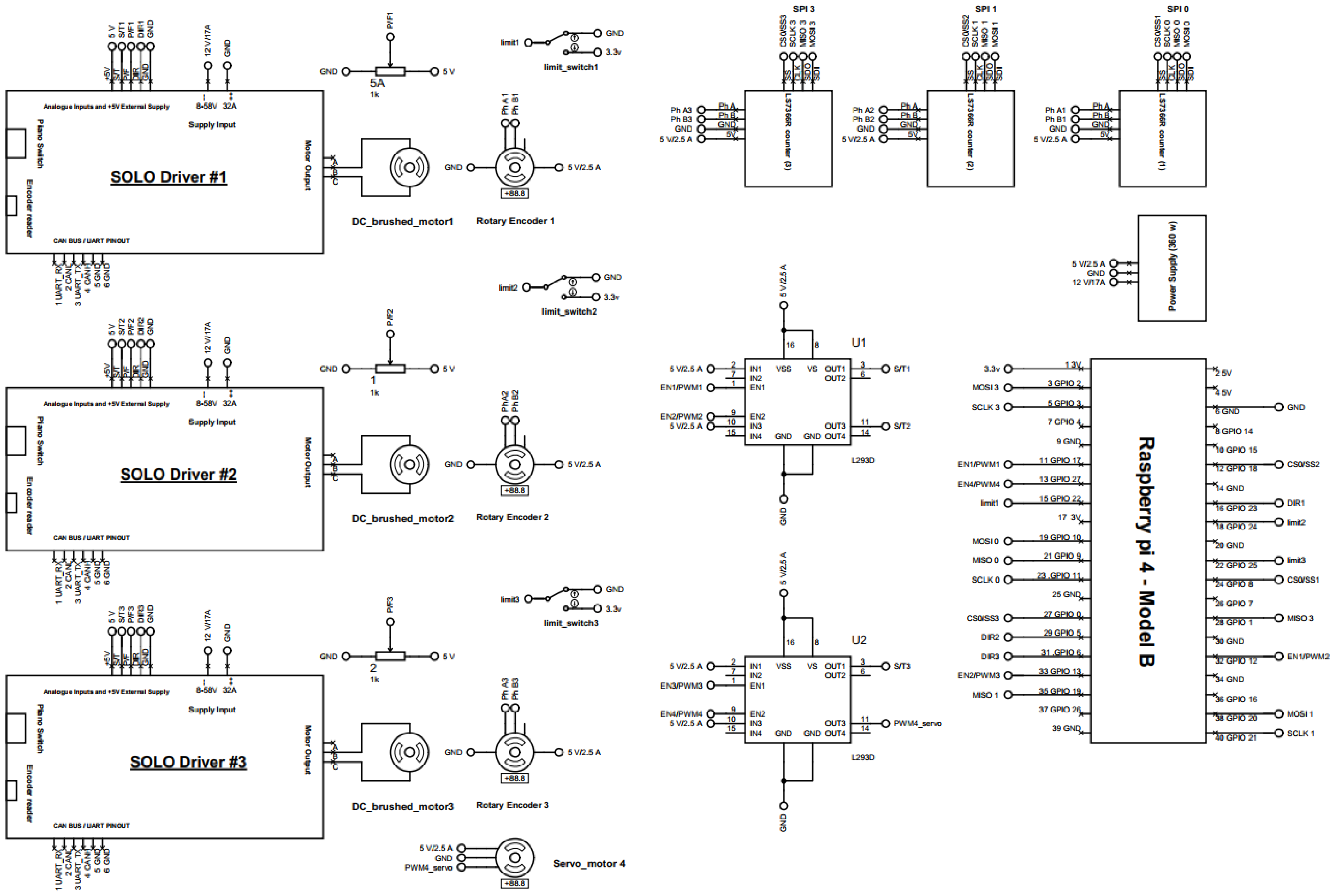


Figure 6. 6: Hardware connectio

Chapter Seven: Robot Motion Control

7.1 Introduction

To perform a specific task of the excavator manipulator, motion planning algorithms are used. The trajectory planning technique which allows the generation of the reference inputs to the motion control system.

Derivation of the manipulator dynamic model provides a description of the relationship between the joint actuators torques, analysis of manipulator structure and design of control algorithms. Lagrange formulation was followed in driving the equations of motion for the manipulator in a systematic way, with chosen coordinate frames according to Denavit-Hartenberg convention.

We applied several techniques for controlling excavator manipulator, these techniques determine the torques to be developed by the joint actuators for the manipulator to achieve the desired trajectory planning.

7.2 Motion planning

7.2.1 Introduction

For a desired end-effector motion with a specified initial and final pose, velocity and acceleration conditions in operational space, the trajectory polynomial of such motion must be determined. The goal of motion planning is to generate the reference inputs to motion control system which ensure the manipulator executes the planned trajectories. Planning is to generate time sequence of values attained by interpolating polynomial or any other function suites the desired trajectory; this trajectory polynomial should be smooth, continuous and allows the actuators to exert generalized forces on joints within the saturation limits with suitable frequencies far away from the resonant ones of the structure.

As pointed before, a trajectory is a path with a specified timing law of variables that describe the end-effector position and orientation over time with respect to the given initial and final conditions. It is known that control actions of the manipulator are performed in joint space, so it is worth to translate the desired end-effector motion through the inverse kinematics that reconstruct the corresponding time sequence of the joint variables in terms to that in task space.

In the following sections, a desired path of the end-effector motion is given in terms of many constraints over time. The trajectory polynomial will be derived in operational space, then it will be transformed to the joint space using inverse kinematics, the obtained timing laws will be used later as reference inputs to many control schemes.

7.2.2 Operational space trajectory

The desired end-effector motion is 0.5m straight line in 2s that combining vertical and horizontal displacements, with the constraints; zero initial and final velocities and accelerations, then the initial and final position is to be selected.

These constraints require fifth polynomial, since there are 6 constraints have to be satisfied, so the motion timing law for the end-effector motion is given by,

$$p(t) = a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0 \qquad (7.1)$$

The coefficients can be determined by solving the following system of linear equations:

$$p_i = a_0$$

$$\dot{p}_i = a_1$$

$$\ddot{p}_i = 2a_2$$

$$p_f = a_5 t_f^5 + a_4 t_f^4 + a_3 t_f^3 + a_2 t_f^2 + a_1 t_f + a_0$$

$$\dot{p}_f = 5a_5 t_f^4 + 4a_4 t_f^3 + 3a_3 t_f^2 + 2a_2 t_f + a_1$$

$$\ddot{p}_f = 20a_5 t_f^3 + 12a_4 t_f^2 + 6a_3 t_f + 2a_2$$

p_i, p_f are selected to satisfy the straight motion with distance between them = 0.5m as required, and the most important thing is to ensure that this path will be in the limits of the working space and taking into consideration the joint limits in the joint space and avoiding the singularities, joints limits are:

$$\begin{aligned} \theta_1 &= \quad 0^o \quad to \ 180^o \\ \theta_2 &= -60 \ \ to \ \ 60^o \\ \theta_3 &= -100^o \ to \ -30^o \end{aligned}$$

The selected initial and final positions are:

$$p_i = [\,0.2511 \quad -0.2037 \quad 0.1661\,]^T \text{m}$$

$$p_f = [0.1 \quad 0.2704 \quad 0.2696]^T \text{m}$$

Where, $p = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T$.

With $\dot{p}_i = \dot{p}_f = \ddot{p}_i = \ddot{p}_f = 0$

Solving for the polynomial's coefficients, we obtain

$$a_0 = [0.2511 \quad -0.2037 \quad 0.1661]^T$$

$$a_3 = [-0.1888 \quad 0.63428 \quad 0.5446]^T$$

$$a_4 = [0.14165 \quad -0.4756 \quad -0.40846]^T$$

$$a_5 = [-0.02833 \quad 0.095137 \quad 0.08169]^T$$

Substitute these coefficients in equation (7.1), yields

$$P_x(t) = -0.02833t^5 + 0.14165t^4 - 0.1888t^3 + 0.2511$$

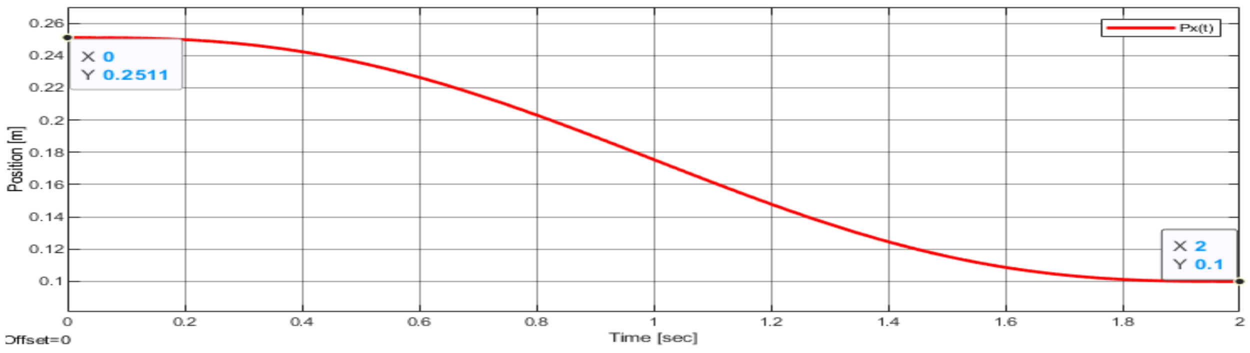
$$P_y(t) = 0.095137t^5 - 0.4756t^4 + 0.63428t^3 - 0.2037$$

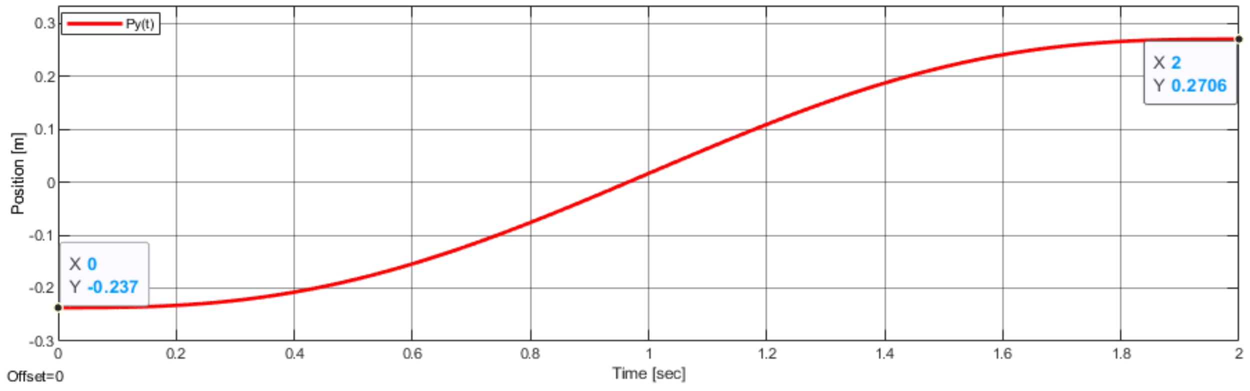
$$P_z(t) = 0.08169t^5 - 0.40846t^4 + 0.5446t^3 + 0.1661$$

reconstructing them in compact form, obtaining:

$$P(t) = \left[P_x(t) \quad P_y(t) \quad P_z(t)\right]^T \tag{7.2}$$

Which is the motion trajectory for end-effector in the operational space. Figure 7. 1, shows the operational space trajectory,





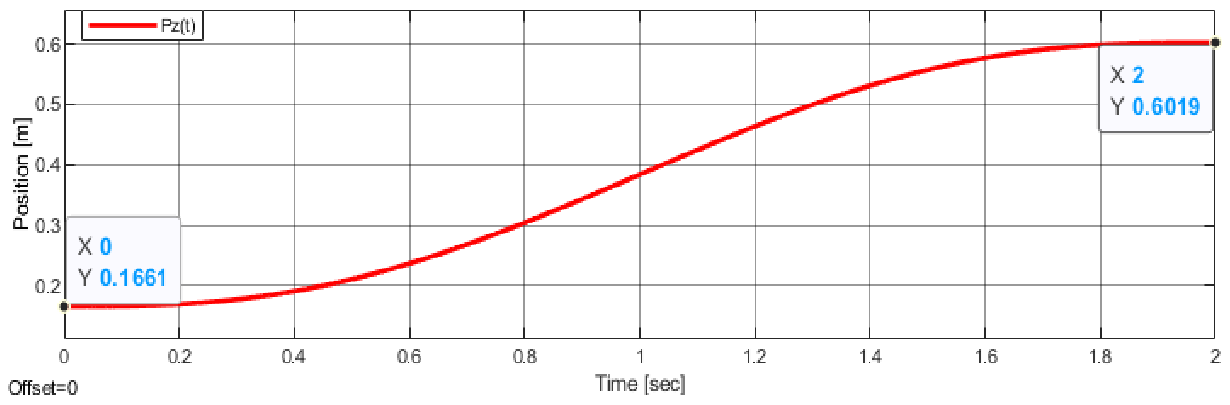


Figure 7. 1: The desired end-effector trajectory; Px(t), Py(t), Pz(t).

7.2.3 Trajectory translating to joint space

As described before, the control actions are carried out in joint space, once the values of the operational space variables are determined, it will be utilized in real time to obtain the corresponding time sequence of values for joint variables $q(t)$. This can be obtained through computing the inverse kinematics for the end-effector trajectory.

Translating methodology

Through the previously derived inverse kinematics, and computing $q(t)$ in real time, taking the admissible solutions with no jumping cases, and pointing here that is no solution will occur outside the joint limits. Because the initial and final positions of end-effector have been selected carefully to avoid singularity and to generate the joint motions within their limits.

A method is followed for selecting p_i and p_f. Firstly, the initial position p_i is chosen in the workspace and transferred via inverse kinematics to the joint space ensuring that the configuration of the initial joint positions q_i are within the set limits, and inspecting that there is a convenient range of motion for the end-effector to move more than 0.5m in its workspace, and verifying that this range has no singularities. Finally, p_f is selected with same procedure and converted to its q_f and tested as previous. A set of intermediate points is tested in the two spaces to validate this solution.

The test initial and final joint positions are:

$$q_i = [-43.330^o \quad -42.794^o \quad -19.033^o]' = [-0.7560 \quad -0.7469 \quad -0.3322]'\text{rad}$$

$$q_f = [\,69.720^o \quad 52.391^o \quad -69.900^o]' = [1.2170 \quad 0.9144 \quad -1.2200]'rad$$

Joint trajectories generation

The operational space trajectory is translated through the inverse kinematics with on-line computation. The angles generation is processed with sampling time of 0.01s in duration of 2s, so 200 points (angles) are created which are forming the joint trajectories as reference inputs.

Passing the end-effector position in equation (7.2) to the inverse kinematics algorithm in off-line computation. The joint timing laws are obtained as followed,

$$q_1(t) = 0.1145t^5 - 0.5725t^4 + 0.7634t^3$$
$$q_2(t) = 0.144t^5 - 0.7198t^4 + 0.9597t^3 - 0.13169$$
$$q_3(t) = -0.255t^5 + 1.2752t^4 - 1.7002t^3 - 0.627$$

Figure 7. 2, the translated trajectory in joint space,

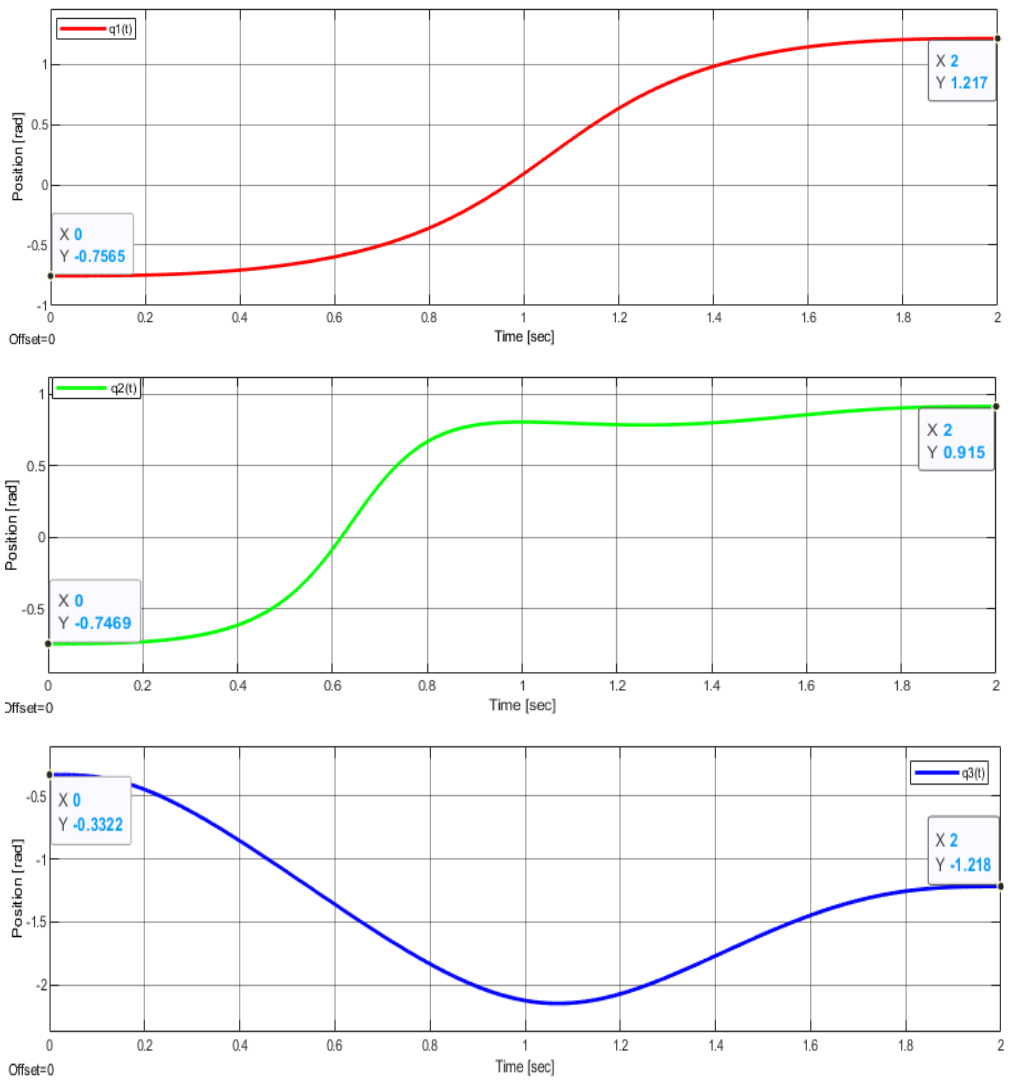


Figure 7. 2: corresponding Joint timing laws; q1(t), q2(t), q3(t).

Figure 7. 3, the velocities and accelerations for all the joint motions that they are determined computationally in SIMULINK as follows,

$$\dot{q}(t) = J^{-1}(q)\,\dot{P}(t)\,,\quad \ddot{q}(t) = J^{-1}(q)\ddot{P}(t) - J^{-1}(q)\,\dot{J}(q,\dot{q})\,\dot{q}(t),\ [5]$$

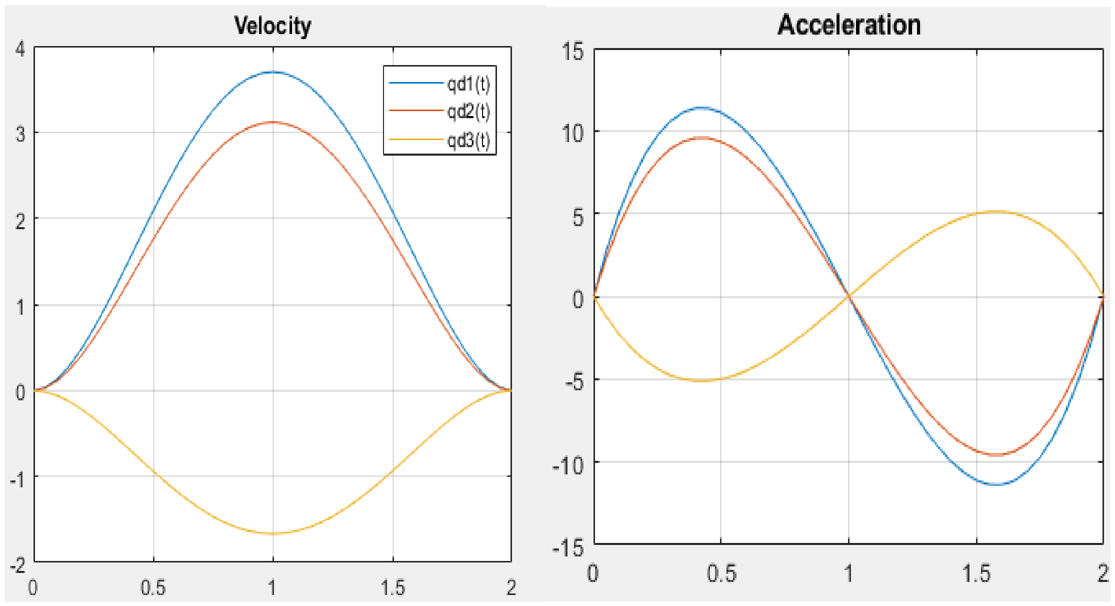


Figure 7. 3: Velocity and acceleration curves for the joint motions.

Figure 7. 4, the end-effector straight motion graphically,

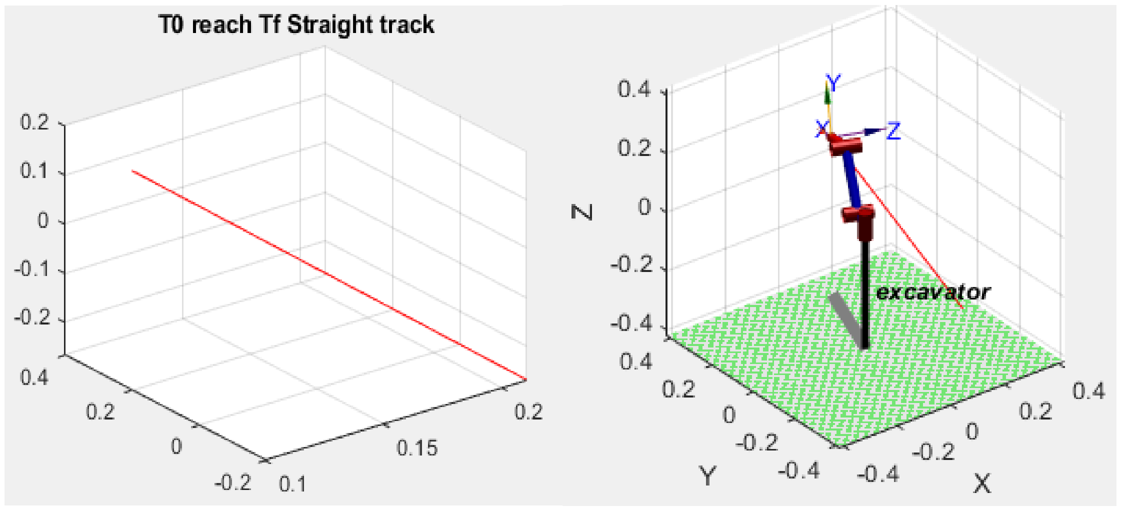


Figure 7. 4: straight line motion in workspace graphically.

Figures. 7.1, 7.2, 7.3 and 7.4 verify that the trajectory is smooth, continuous and achieves the imposed constraints.

7.3 Motion control in joint space

7.3.1 Introduction

The joints control problem requires the manipulator inverse kinematics to be solved for the desired end-effector trajectory x_d from the operational space into the corresponding joint space motion q_d, see Figure 7. 5 [5]. The control action should provide the three components of generalized joint torques to allow the execution of the desired joint motions $q_d(t)$, the transient response and the steady-state error. The generalized toques must be as less as possible, so a designer should compromise between the control performance and the power consumed for the desired motion.

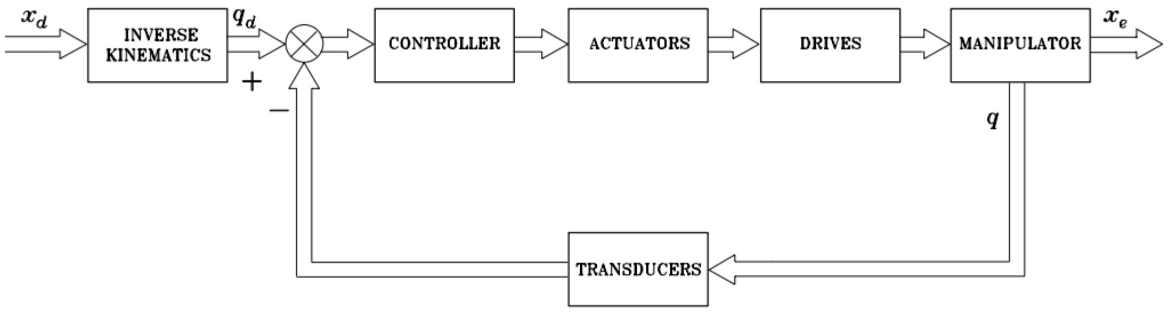


Figure 7. 5: General scheme of joint space control.

A joint space control scheme is designed that allows the actual joint motion q to track a reference input q_d, and pointing here that control in joint space is considered as an open loop fashion scheme in terms of x_e. i.e., joint space control scheme does not influence the operational space variables x_e directly that is no feedback for the actual end-effector trajectory x_e, and the whole motion is executed through the manipulator structure.

Two type of control schemes can be adopted, one is the decentralized control, this control treats the drive as velocity-controlled generator, and when a single joint is controlled independently of the others. The other type is the centralized control, so the drive is treated as torque-controlled generator that the driving torque is computed for the reduced or complete manipulator dynamics, i.e., the dynamic interaction effects and coupling are taken in consideration.

It is worth to mention that for all the coming control schemes, all motor actuators are assumed as direct-drive actuators for each joint, this means the transmission gear ratio $K_r = I_3$, in this case, using the relation ($\tau_m = K_r^{-1}\tau$); shows that the motor generated torque τ_m equals the generalized torques τ at each joint. If K_r is considered much greater than unity, this makes the tracking accuracy of the joint variables with respect to the desired trajectory is improved, the system becomes more robust to parameter changes and tends to linearize system dynamics.

7.3.2 Decentralized control

This control strategy regards this excavator manipulator as formed by 3-independent subsystems which is the simplest control strategy that can be deigned. It controls each joint as a single input/single output system (SISO), as a consequence the coupling effects between joints are treated as disturbance inputs that result due to varying configurations during motion.

It is known that task specification (end-effector motion and forces) is carried out in the operational space, but control actions are performed in joint space for the desired end-effector trajectory using the inverse kinematics to obtain the corresponding joint motions.

The disturbance of dynamic coupling effects is canceled by the integral control that allows disturbance rejection due to increasing the type number of the closed-loop system. As required, each joint controller must guarantee good performance in terms of high disturbance rejection and enhanced trajectory tracking capabilities.

Therefore, the control system actuates each joint corresponding to SISO subsystem of the decoupled and linear part of the following scheme in Figure 7. 6[5],

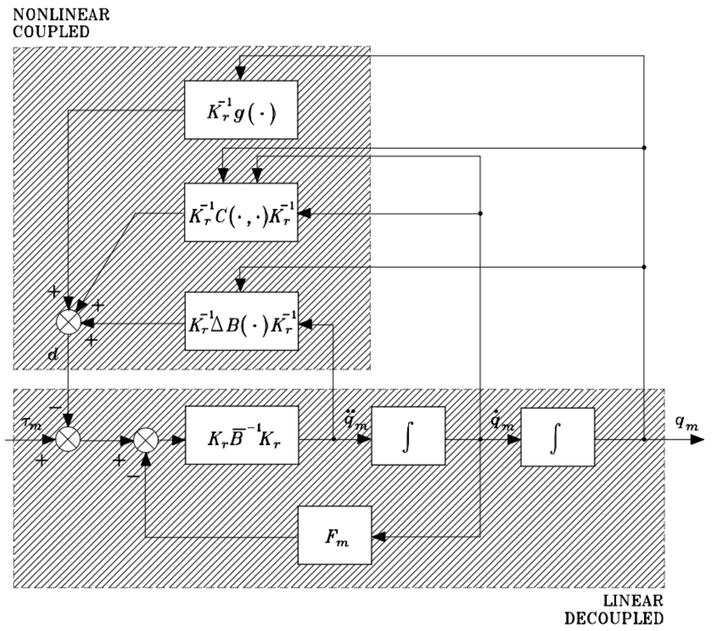


Figure 7. 6: Block scheme of the system of manipulator with the drive

Taking the diagonal constant elements of $B(q)$ that form the decoupled link inertias $\bar{B}$ by subtracting the nonlinear terms $\Delta B(q)$ from the inertia matrix $B(q)$ as follows,

$$\bar{B} = B(q) - \Delta B(q) \tag{7.3}$$

$\bar{B}$ is a diagonal matrix with constant elements of the resulting average inertia at each joint, and observing the equations 7.4, 7.5 and 7.6 [5] which describe the disturbance d of the interactions between the joints that includes the coupling effects due to configuration dependency, τ_m the motor torque. In Figure 7. 6, the subsystem at the bottom is linear and decoupled, and the upper one is nonlinear and coupled which contains all nonlinear and coupling terms of the manipulator joint dynamics, so each joint drive component i is affected by component i of d, comparing this to the following equations, [5]:

$$K_r^{-1}\bar{B}K_r^{-1}\ddot{q}_m + F_m\dot{q}_m + d = \tau_m \tag{7.4}$$

$$F_m = K_r^{-1}F_v K_r^{-1} \tag{7.5}$$

$$d = K_r^{-1}\Delta B(q)K_r^{-1}\ddot{q}_m + K_r^{-1}C(q,\dot{q})K_r^{-1}\dot{q}_m + K_r^{-1}g(q) \tag{7.6}$$

So, here we have three independent control systems for the three joints, each one of SISO system of the linear-decoupled part, and the interactions with the other joints is added as a disturbance of component i at each joint. Two control schemes will be introduced, PID and I-PD controllers.

7.3.3 Position and velocity feedback control (PID controller)

Now we will design position and velocity feedback control scheme of PID with decentralized feedforward compensation. PID controller consists of three terms; P (Proportional), I (Integral) and D (Derivative), it feeds the error (proportional) plus the integration and the differentiation of the error forward to the plant.

The P-element allows the stabilization of the system, and I-element reduces the steady-state error (adds pole at origin, then increase the closed-loop system type number) while the D-element improves the transient response (adds zero to the closed-loop system).

The design requirements have to be determined, let the desired $\%OS = 0 \rightarrow \zeta = 1$ (critically damped) and $T_s = 0.1s \rightarrow \omega_n = 40rad/s$ for each joint control system, and following the Root-locus technique to find the controller gains: K_p, K_i and K_d for the 3-SISO independent system controllers, yields

$$K_p = [453.2 \quad 279.6 \quad 36.8]^T$$
$$K_i = [34.34 \quad 26.76 \quad 3.543]^T$$
$$K_d = [22.66 \quad 13.98 \quad 1.84]^T$$

Figure 7. 7 [9], shows the general independent joint control scheme, where $C_P(s)$, $C_V(s)$, $C_A(s)$ respectively represent position, velocity, acceleration controllers. k_{TP}, k_{TV}, k_{TA} are the respective transducer constants. ϑ_r is the reference input, which is related to the desired output ϑ_{md}, and the disturbance torque D has been suitably transformed into a voltage by the factor $R_a/\, k_t$, (k_t: torque constant, R_a: the armature resistance).

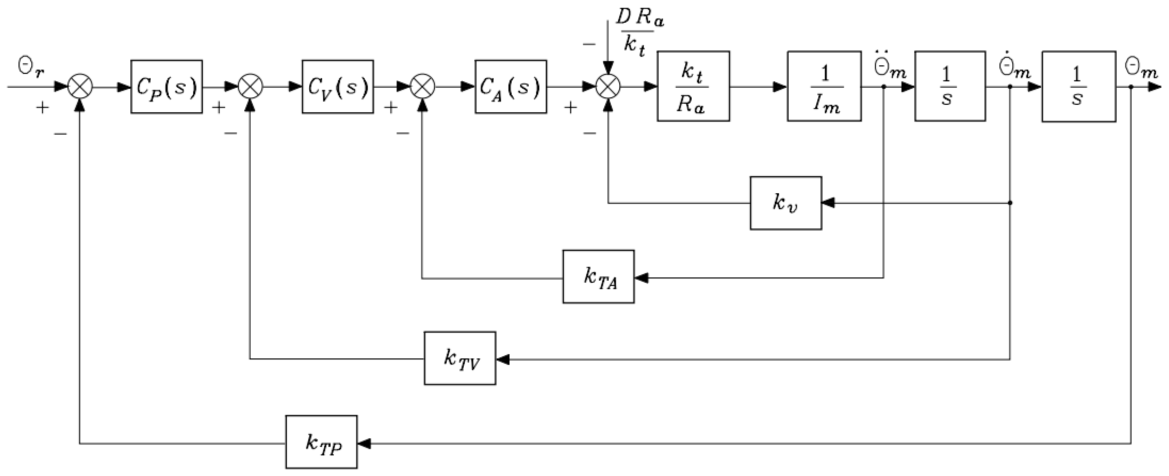


Figure 7. 7: Block scheme of general independent joint control.

Goal is to design Position, velocity feedback with decentralized feedforward compensation, by Figure 7. 7; $C_P(s)$ and $C_V(s)$ are the designed position and velocity controllers, where $C_A(s) = 1$, $K_{TA} = 0$, and the feedback position and velocity gains (k_{TP}, k_{TV}) are assumed unity.

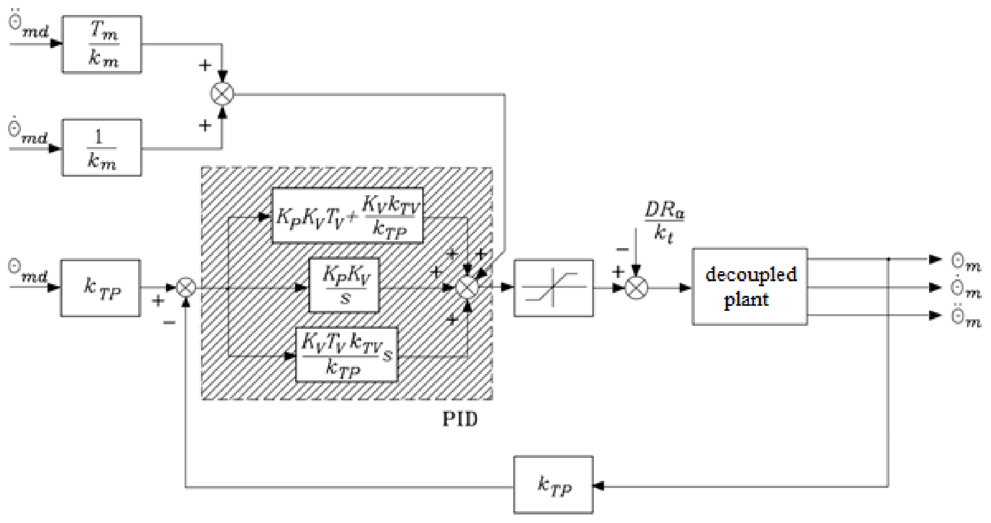


Figure 7. 8: equivalent control scheme of PID type.

Figure 7. 8 is the equivalent control scheme for PID with decentralized feedforward compensation [5], where T_m, k_m are motor parameters.

Many design iterations have been performed to improve the transient response and reducing the steady-state error. The new design characteristics for all joints are selected as, $\zeta = 1$ and $\omega_n = 90 rad/s$, and all the following simulation results for these specifications, with control gains,

$$K_p = [2294 \quad 1415 \quad 186]^T$$
$$K_i = [61.5 \quad 33 \quad 5.34]^T$$
$$K_d = [60 \quad 31.5 \quad 4.2]^T$$

Using MATLAB and SIMULINK for design and simulation of the control system. Figure 7. 9 shows the SIMULINK block diagram for this control system.

The following symbols are used in the control schemes:

qd: $q_d(t)$ $[rad]$, qd_dot:$\dot{q_d}(t)$ $[rad/s]$, qd_Ddot: $\ddot{q}_d(t)[rad/s^2]$,
q: $q(t)$ $[rad]$, q_dot:$\dot{q}(t)$ $[rad/s]$

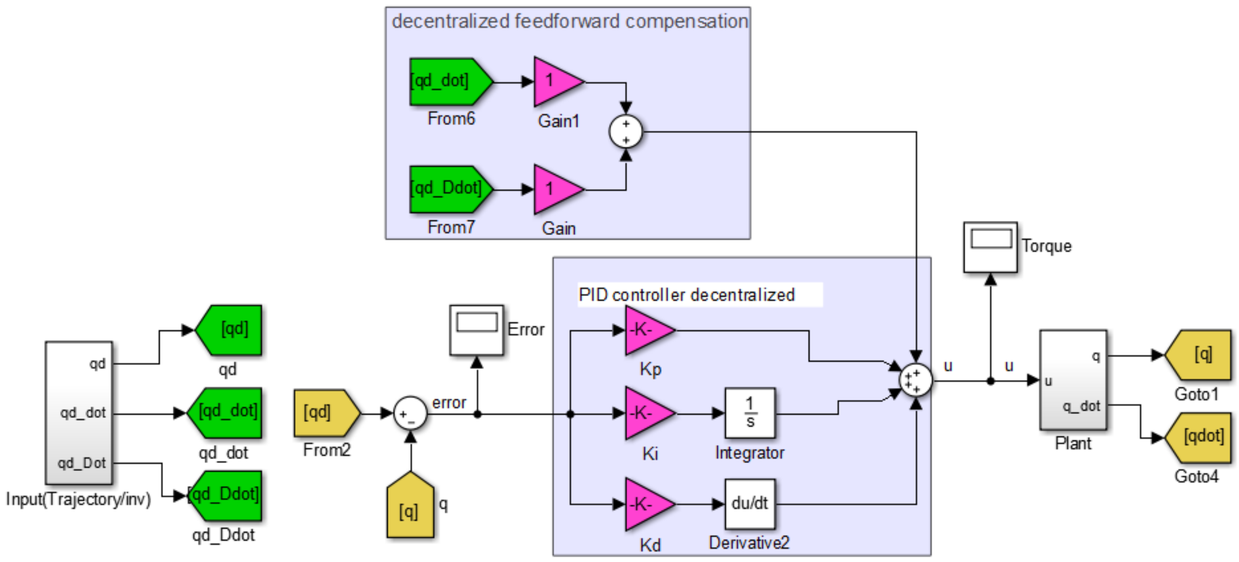


Figure 7. 9: SIMULINK block diagram for PID control system with feedforward compensation.

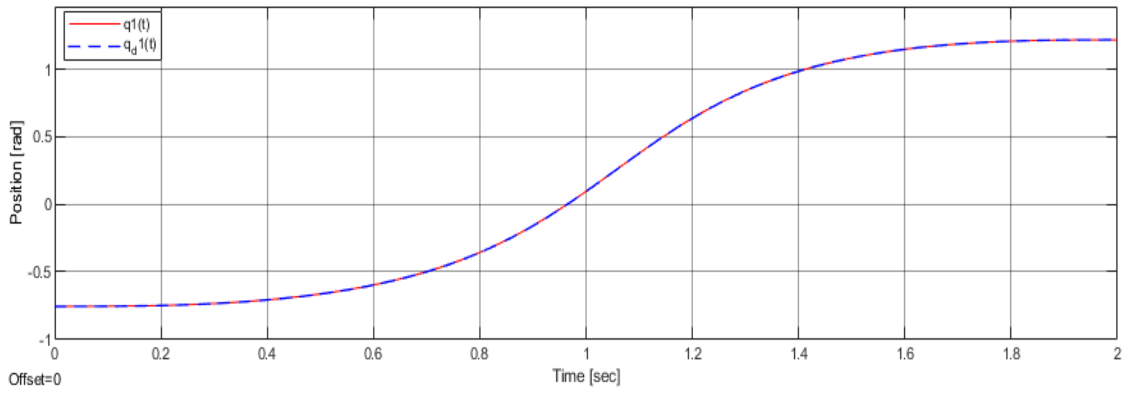


Figure 7.10(a): q1(t) vs. qd1(t) - PID

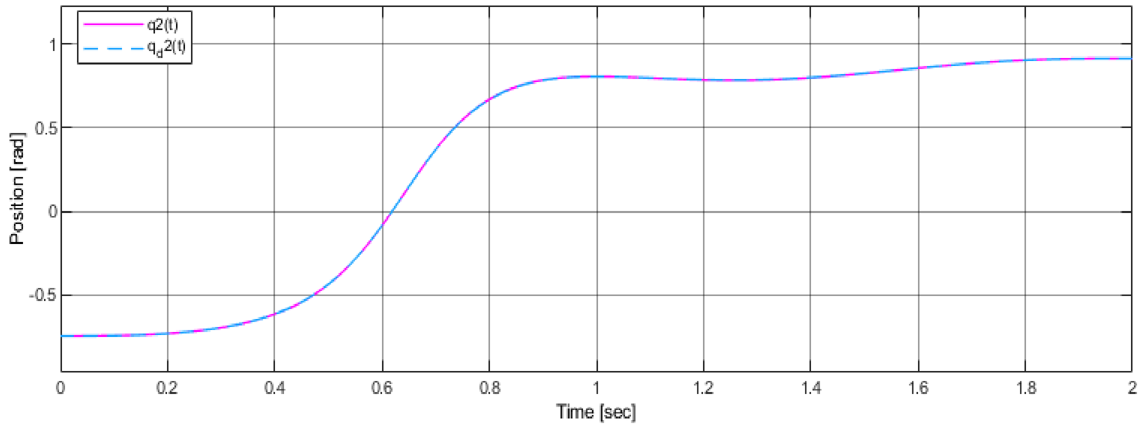


Figure 7.10(b): q2(t) vs. qd2(t) - PID

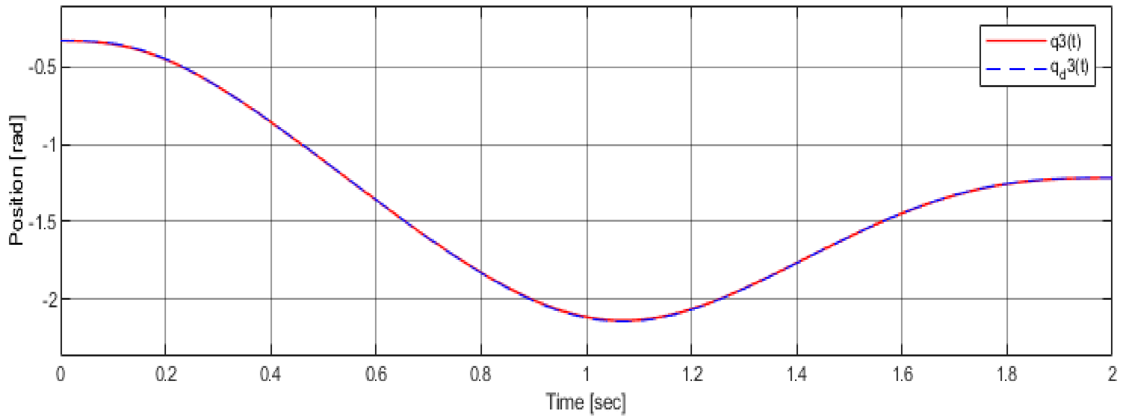


Figure 7.10(c): q3(t) vs. qd3(t) – PID

Figure 7. 10:(a), (b) and (c): q(t) vs.qd(t)- PID

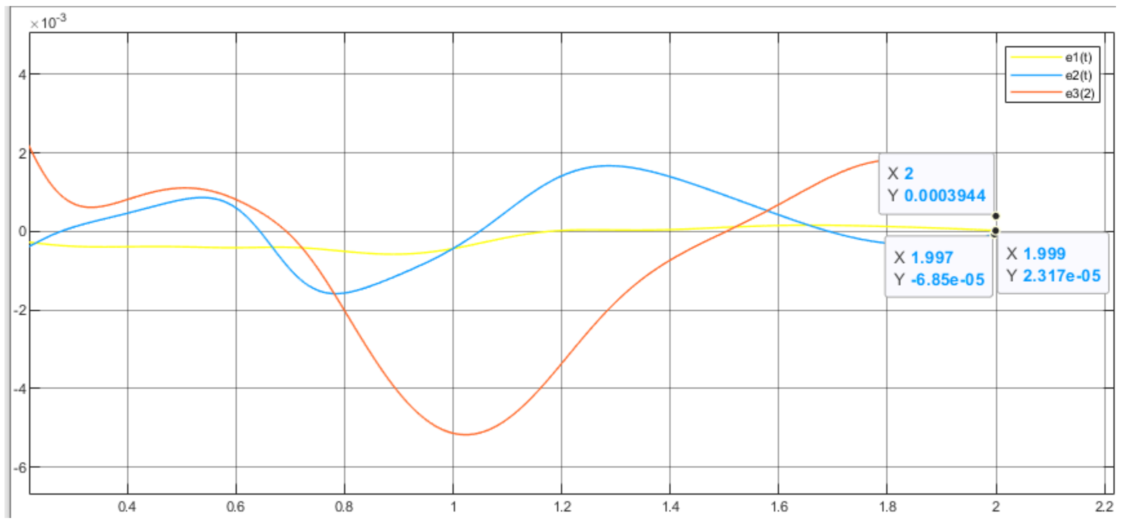


Figure 7. 11: The error between q(t) and qd(t) - PID

Figure 7. 10, shows the simulation results for each joint, it is obvious that PID control method provides acceptable tracking performance of the desired reference inputs $q_d(t)$, but with slow dynamics of error reduction. In Figure 7. 11, $e_{ss1} = 0\%$, $e_{ss2} = 0\%$, $e_{ss3} = 0\%$.

Figure 7. 12; the joint (1) needs at least torque of 0.512 Nm, and joint (2) requires torque of 3.52 Nm while joint (3) torque is 1.298 N.m. It is concluded that joint (2) has the highest torque needed because it is affected by the other two joints to cancel their disturbances. In other word, the performance is accepted in terms to the consumed torque, and the controller can be redesigned to give better performance with some increase in joint torques.

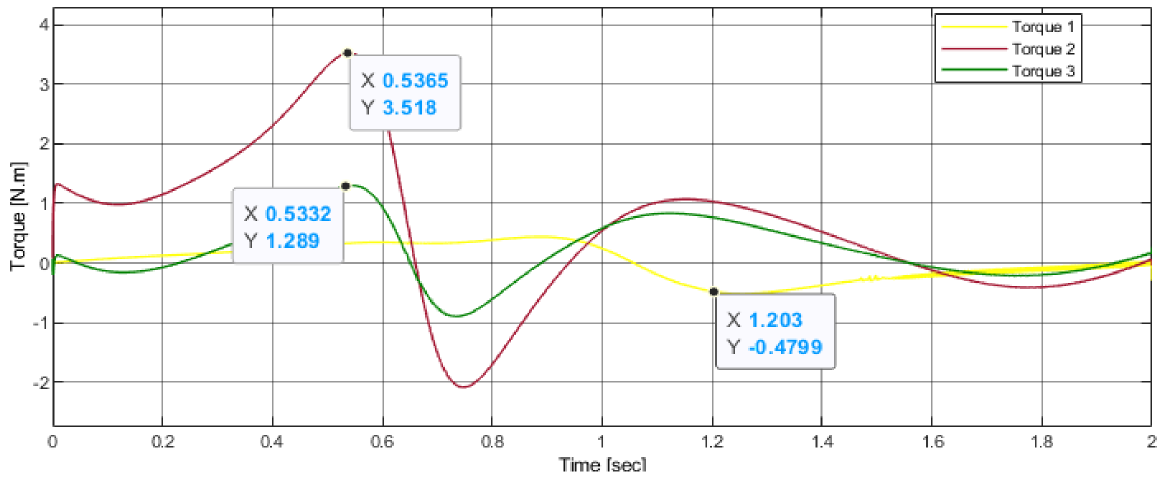


Figure 7. 12: Torque required at each joint- PID

7.3.4 I-PD control (State-feedback with integral control)

Now for the same design requirements, I-PD controller is to be designed following the state-feedback control technique. This control strategy has the same response improvements of PID controller, but it is more efficient that it makes the system more stable than PID because it does not add zeros to the closed-loop system.

State-feedback control allows to replace the closed-loop poles to the desired locations in S-plane through feedback control gains K_{3x1} and another gain $K_{e_{1x1}}$ for the error reduction dynamics. This technique is applicable for linear control systems, so with the decentralized control method, the gains K and K_e will be determined for the decoupled and linear part of the system dynamics.

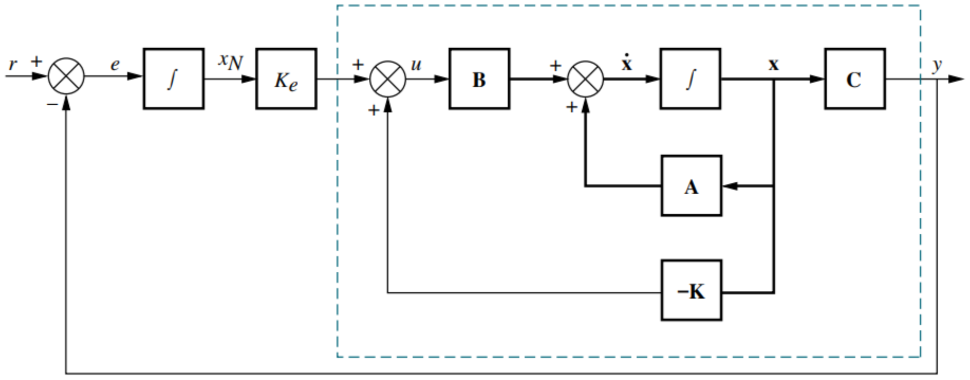


Figure 7. 13: state-feedback control with integral control (I-PD).

Figure 7. 13 [15], shows the general block diagram for the I-PD controller. The integral action for the error feedback is to achieve zero steady-state error, and x are system state variables, A is the system matrix, B is the input matrix, C is the output matrix. The input references of the joints r, and joint motions output is y.

The decoupled and linear part (7.3) of the manipulator dynamics is the diagonal $\bar{B}_{3x3}$ matrix, for each element b_{ii} $(i = 1,2,3)$, a state-space system (A, B, C) with phase variable form can be formed for each joint as SISO system, so we have 3 SISO controllers.

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}u \\ \dot{x}_N &= -\mathbf{C}\mathbf{x} + r \\ y &= \mathbf{C}\mathbf{x} \end{aligned} \tag{7.7}$$

$$\begin{bmatrix}\dot{\mathbf{x}}\\ \dot{x}_N\end{bmatrix} = \begin{bmatrix}\mathbf{A} & 0\\ -\mathbf{C} & 0\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ x_N\end{bmatrix} + \begin{bmatrix}\mathbf{B}\\ 0\end{bmatrix}u + \begin{bmatrix}\mathbf{0}\\ 1\end{bmatrix}r$$
$$y = [\mathbf{C} \quad 0]\begin{bmatrix}\mathbf{x}\\ x_N\end{bmatrix} \qquad (7.8)$$

$$u = -K\,x \; + K_i\,x_n, \qquad (7.9)$$

Where x are the states of the system, and x_n is the additional state for the integral control, The system states and outputs (7.7), and the extended system in state space representation (7.8), with the control law (7.9). Each joint will be represented as constant mass needed to be actuated to execute the desired motion. Matrix $\bar{B}$ is found as follows,

$$\bar{B} = \begin{bmatrix}\bar{b}_{11} & 0 & 0\\ 0 & \bar{b}_{22} & 0\\ 0 & 0 & \bar{b}_{33}\end{bmatrix} = \begin{bmatrix}1.004 & 0 & 0\\ 0 & 0.522 & 0\\ 0 & 0 & 0.31612\end{bmatrix} Kg.m^2, \quad (7.10)$$

The new state-space representation for each mass (i) is,

$$\dot{x} = A_i x + B_i u_i$$
$$y_i = C_i x_i, \qquad (7.11)$$

Where,

$$x_{1i} = q_i, \qquad x_{2i} = \dot{q}_i$$
$$\dot{x}_{1i} = x_{2i}, \;\; \dot{x}_{2i} = \frac{1}{\bar{b}_{ii}} u_i, \qquad (7.12)$$

$$A_i = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}, B_i = \begin{bmatrix}0\\ \frac{1}{\bar{b}_{ii}}\end{bmatrix} \text{ and } C_i = [1 \quad 0], \qquad (7.13)$$

While the third state is $x_{N_i} = -C_i x_i + r_i$, and applying state-derivatives (7.12) and matrices (7.13) in (7.8, 7.11) we obtain the extended system:

$$\begin{bmatrix}\dot{x}_i\\ \dot{x}_{N_i}\end{bmatrix} = \begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 0\\ -1 & 0 & 0\end{bmatrix} x + \begin{bmatrix}0\\ \frac{1}{\bar{b}_{ii}}\\ 0\end{bmatrix} u + \begin{bmatrix}0\\ 0\\ 1\end{bmatrix} r_i$$
$$y_i = [1 \;\; 0 \;\; 0\,]\begin{bmatrix}x_i\\ x_{N_i}\end{bmatrix}.$$

Using MATLAB (place function) to find the gains K and K_i for all masses,

$$K_{b_{11}} = [K_1 \quad K_2] = [9.8486e4 \quad 0.0683e4].$$
$$K_{i_{b_{11}}} = 4.0657e6$$

$$K_{b_{22}} = [K_1 \quad K_2] = [5.1205e4 \quad 0.0355e4].$$
$$K_{i_{b_{22}}} = 2.1139e6$$

$$K_{b_{33}} = [K_1 \quad K_2] = [3.1010e4 \quad 0.0215e4].$$
$$K_{i_{b_{33}}} = 1.28e6$$

These three sets of gains ($K_{b_{ii}}, K_{i_{bii}}$, $i = 1,2,3$) for the 3-SISO independent controllers. This controller has many outcomes, as disturbance rejection and makes the system robust against changes in its parameters, and this will be tested in the following simulation.

Figure 7. 14, block scheme for I-PD Controller,

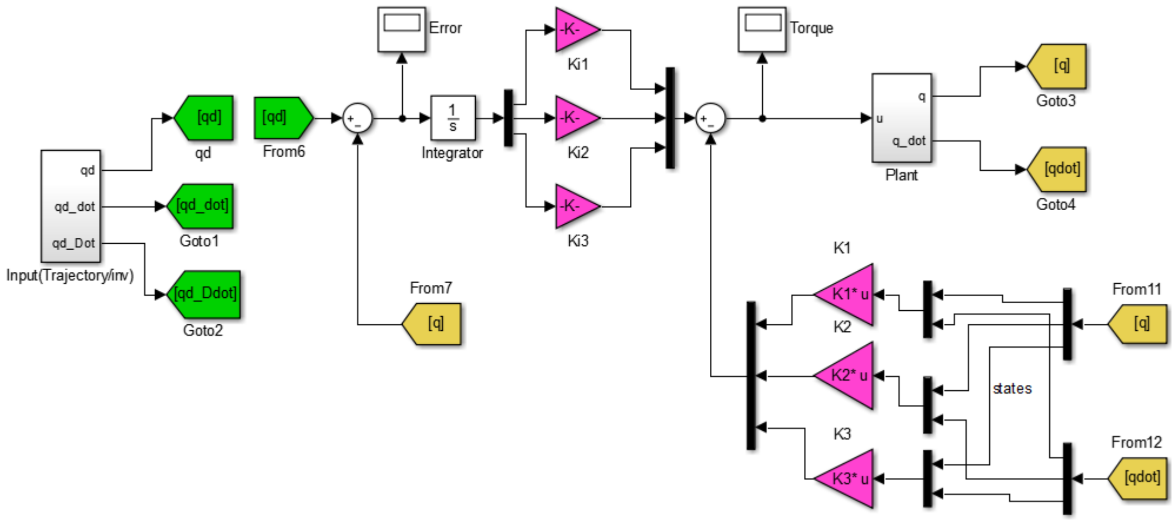


Figure 7. 14: Block scheme of I-PD controller

Simulation results in Figure 7. 15, show that I-PD controller provides accurate tracking with fast transient response and error reduction dynamics. Figure 7. 16, verifies that this controller has powerful effect on error reduction, as seen it gives approximately zero steady-state error for all joints.

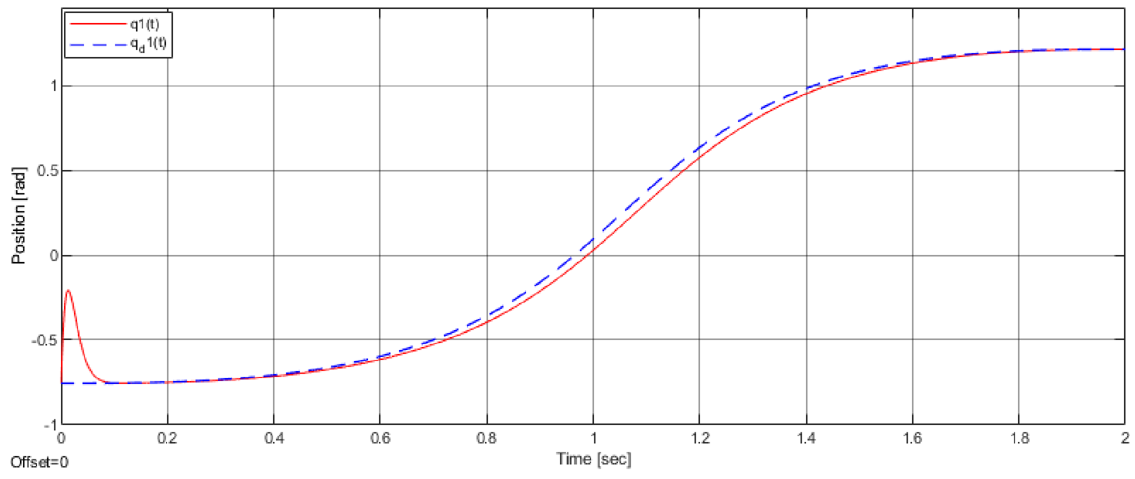


Figure 7.15(a): q1(t) vs. qd1(t), I-PD

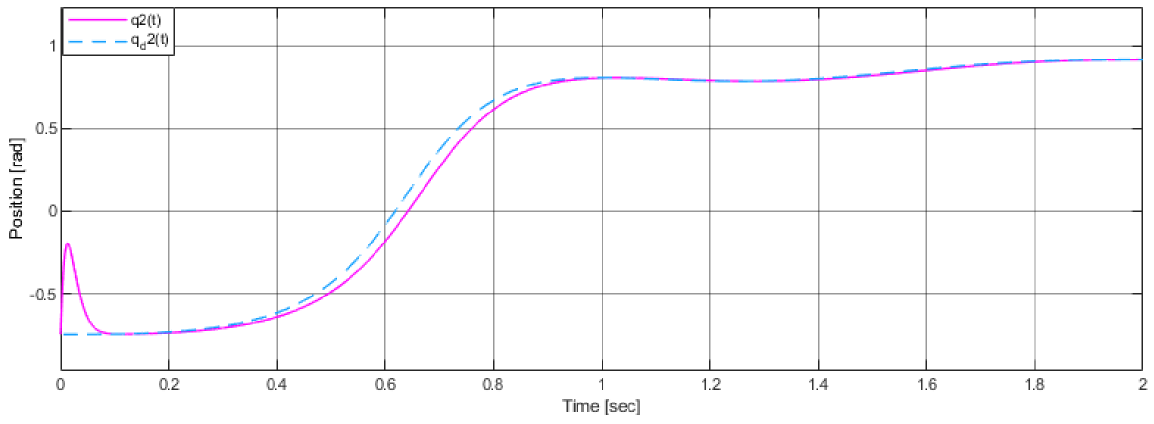


Figure 7.15(b): q2(t) vs. qd2(t), I-PD

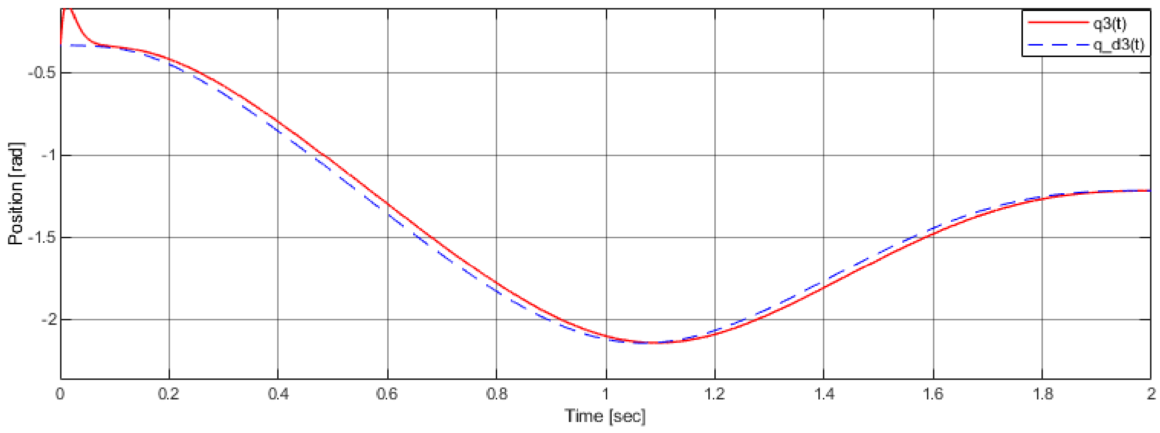


Figure 7.15(c): q3(t) vs. qd3(t), I-PD

Figure 7. 15:(a), (b) and(c) q(t) vs. qd(t), I-PD.

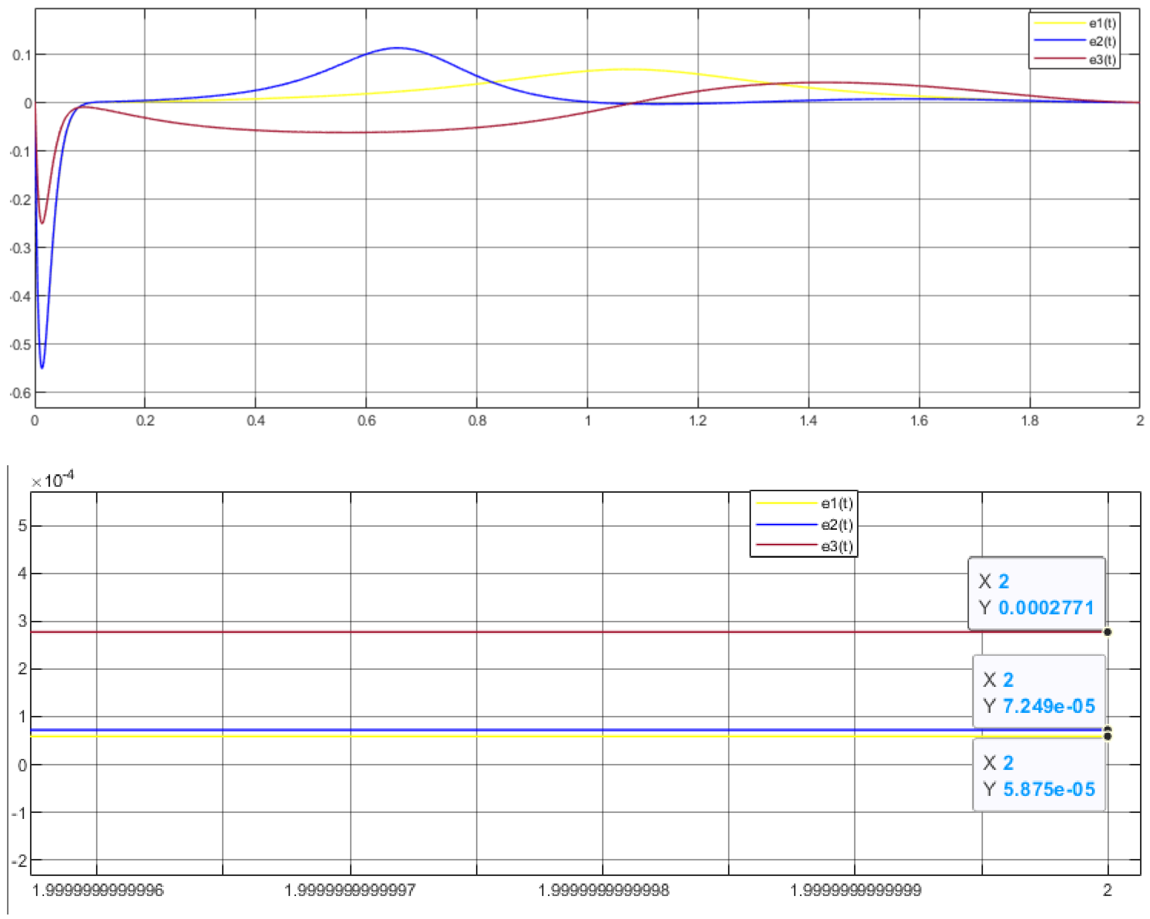


Figure 7. 16: The error between q(t) and qd(t), I-PD

The joint torques are shown in Figure 7. 17, at starting, the torques (2,3) jump highly to time of 0.05s, then decreased to lower values about (5N.m, 2N.m), these high torques appear because this controller gives fast dynamics of error reduction, and MATLAB solver has no constraints about maximum torque, so for the given gains and performance, the solver gives high torques. The absence of gear reduction ratios makes the torques at motor side more highly. Torque (2) always larger than other torques, since this joint has to reject two disturbances (coupling effects) from the other joint motions. Torque (1) is the less torque because that gravitational force does not affect on the first joint.

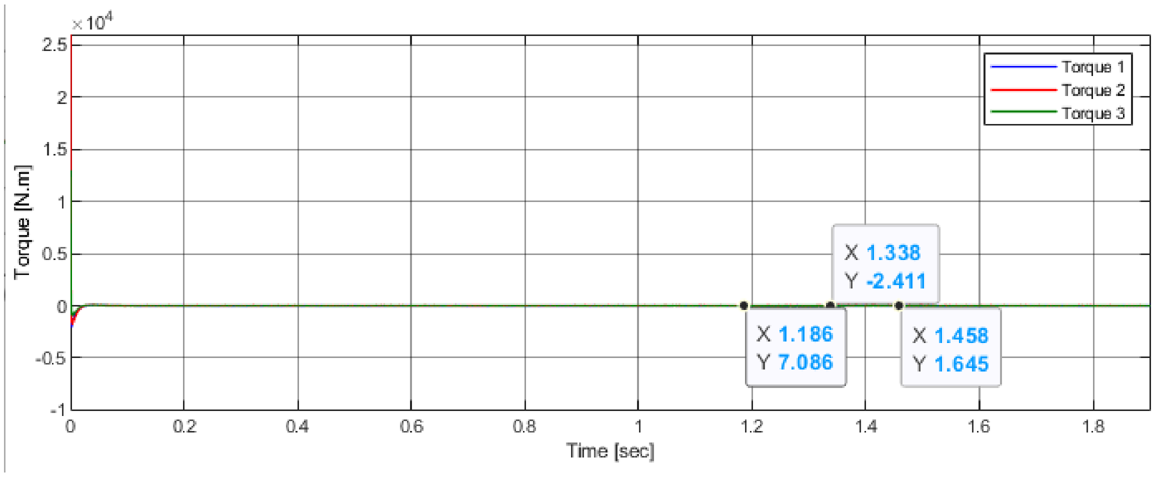


Figure 7. 17: Torque at each joint, I-PD.

7.3.5 Centralized control

This methodology is adopted with control design that takes into account the manipulator dynamic model, finding through it the nonlinear centralized control law. In this case, the design of the control takes advantage of detailed knowledge of excavator manipulator dynamic model, and to compensate the nonlinear coupling terms of the model. These control methods are considered as Multi-Inputs/Multi-Outputs (MIMO) system.

Centralized control algorithms are needed for high manipulator dynamic performance, since the driving joint torque with compensating actions are computed which provide a model-based nonlinear term to enhance trajectory tracking performance.

The main two advantages of the centralized (nonlinear) control techniques. First is the cancelation of the nonlinearities by generating compensating torques, i.e., such techniques allow elimination of the causes, not reducing the effects induced by them. The other positivity when the desired manipulator motion requires large joint speeds and/or acceleration, or direct drive actuation is employed (gear ratio $Kr = I$), the nonlinear coupling terms will strongly influence system performance, so compensating nonlinear terms are essential. The inverse dynamics control method will be presented.

7.3.6 Inverse dynamics

This approach depends on the idea of finding a compensating control low u, as a function of the system dynamics and states. the dynamic model of the 3-joints excavator manipulator is expressed by

$$B(q)\ddot{q} + n(q,\dot{q}) = \tau \tag{7.14}$$

Where,

$$n(q,\dot{q}) = C(q,\dot{q})\dot{q} + F\dot{q} + g(q)$$

This control method is essential to generate a linear input/output relationship through an exact linearization of system dynamics with no approximations, this is performed by the nonlinear state feedback of the dynamics system.

To generate a suitable compensating torque, this method suggests to set u (7.14) as the generalized joint torques τ, so the control law becomes,

$$u = B(q)\ddot{q} + n(q,\dot{q}) \tag{7.15}$$

The mass matrix is full rank and its invertible, then set $\ddot{q} = y$ which is the inverse dynamics control action and a stabilizing control law whose magnitude has to be found through the control design. By observing the Figure 7. 18[5], it describes that function of the inner loop is to obtain a linear and decoupled input/output relationship, and the outer loop is to stabilize the overall system. As a consequence, y is considered as a new input of linear and decoupled system after compensating the nonlinearities, and u is the input to the nonlinear-coupled dynamic system as compensating torque.

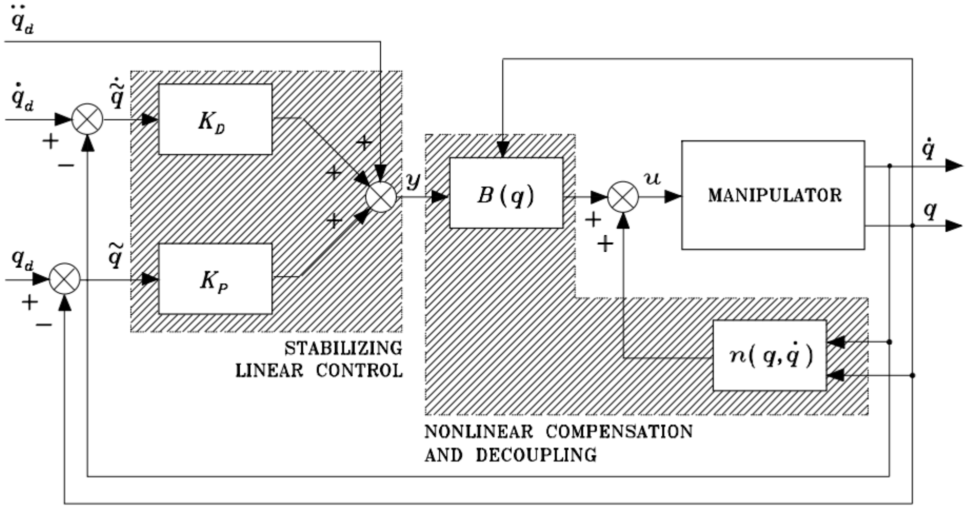


Figure 7. 18: Block scheme of joint space inverse dynamics control.

Setting the new control law,

$$y = -K_p q - K_D \dot{q} + r \tag{7.16}$$

Where r_{3x1} torques actuate the joints q_i $(i = 1,2,3)$ independently as decoupled subsystems to insure tracking the reference inputs $q_d(t)$ by selecting,

$$r = \ddot{q}_d + K_D\dot{q}_d + K_p q_d \tag{7.17}$$

This equation (7.17) leads the control law y_{3x1} with respect to the control gains $K_{P_{3x3}}$ and $K_{D_{3x3}}$ which are positive definitive matrices, with diagonal elements that are designed by the desired transient response characteristics of vectors ζ_{3x1} and ω_{3x1},

$$K_P = \begin{bmatrix} \omega_{n1}^2 * b_{11} & 0 & 0 \\ 0 & \omega_{n2}^2 * b_{22} & 0 \\ 0 & 0 & \omega_{n3}^2 * b_{33} \end{bmatrix}$$

$$K_D = \begin{bmatrix} 2\zeta_1\omega_{n1} * b_{11} & 0 & 0 \\ 0 & 2\zeta_2\omega_{n2} * b_{22} & 0 \\ 0 & 0 & 2\zeta_3\omega_{n3} * b_{33} \end{bmatrix}, \qquad (7.18)$$

for the three decoupled systems with dynamics of error reduction (transient response) that depend on these gains (7.18) to track a given trajectory.

After these advantages, it is worth to remark that inverse dynamics control needs accurate knowledge of system dynamics and its parameters, to allow accurate compensating of the nonlinear effects. This operation made in online calculations which requires high speed and power in computation with less sampling rates.

Finding the gains K_P and K_D with the same design requirements $\zeta = 1$ and $\omega_n = 90\ rad/s$ for all joints by (7.18),

$$K_P = 1e3 * diag[\,8.13424 \quad 4.2282 \quad 2.5606\,]$$
$$K_D = diag[\,180 \quad 93.96 \quad 56.9]$$

Design and simulate the control scheme in SIMULINK, see Figure 7. 19 showing the block diagram, and Figure 7. 20 for the results.

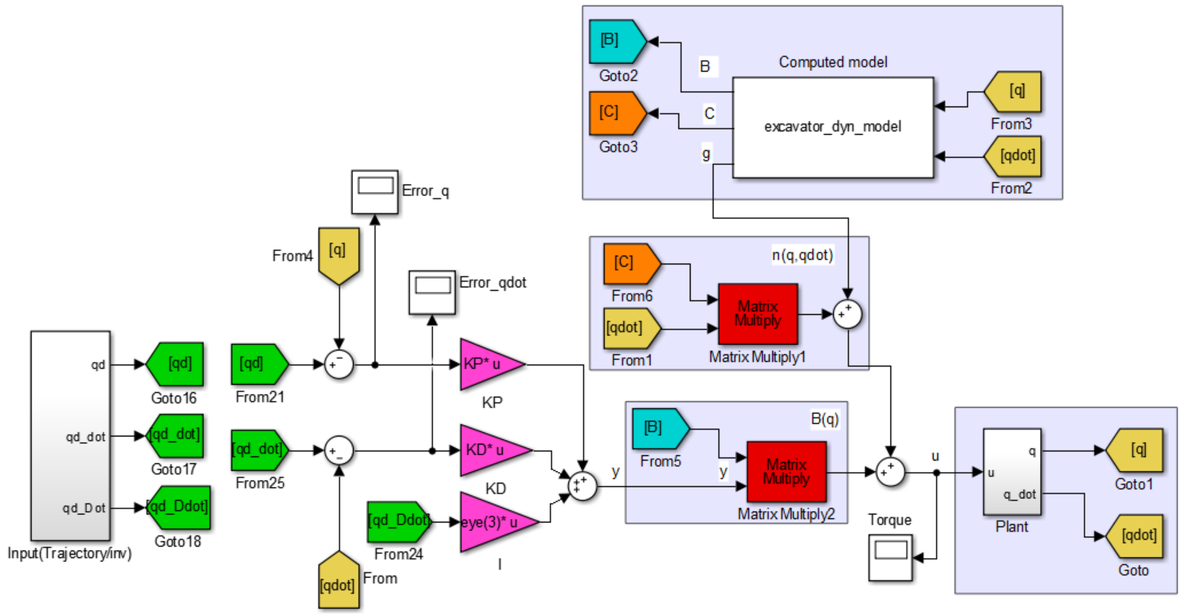


Figure 7. 19: Block scheme of inverse dynamics control.

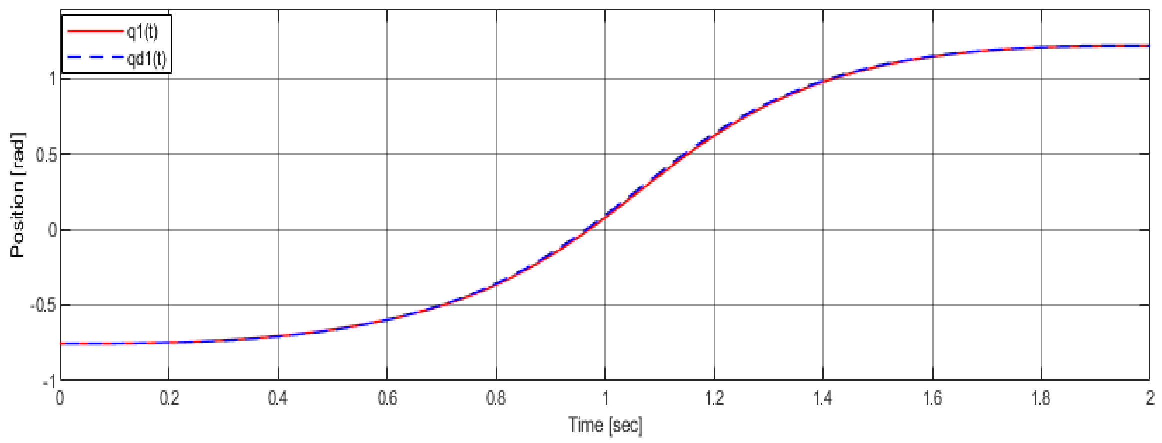


Figure 7.20(a): q1(t) vs. qd1(t), Inverse dynamics

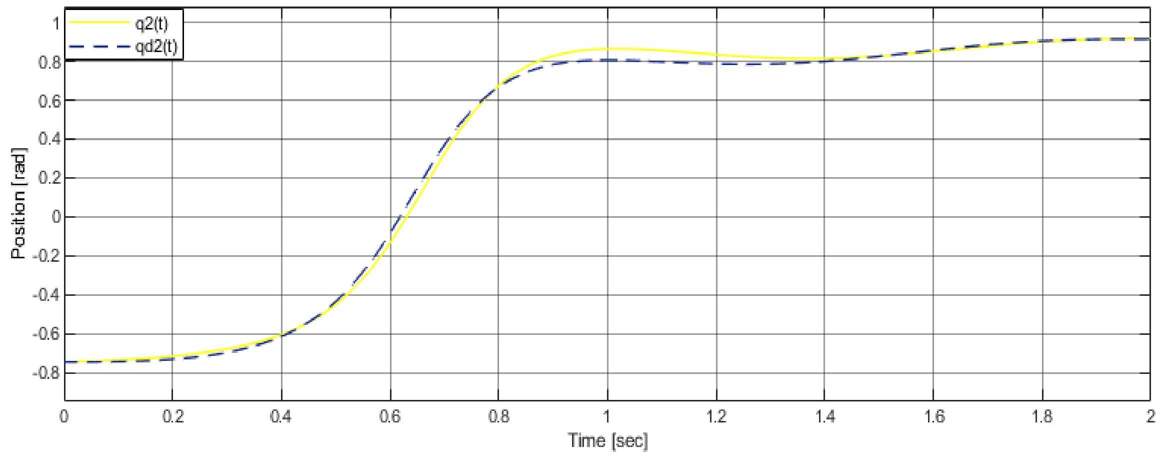


Figure 7.20(b): q2(t) vs. qd2(t), Inverse dynamics.

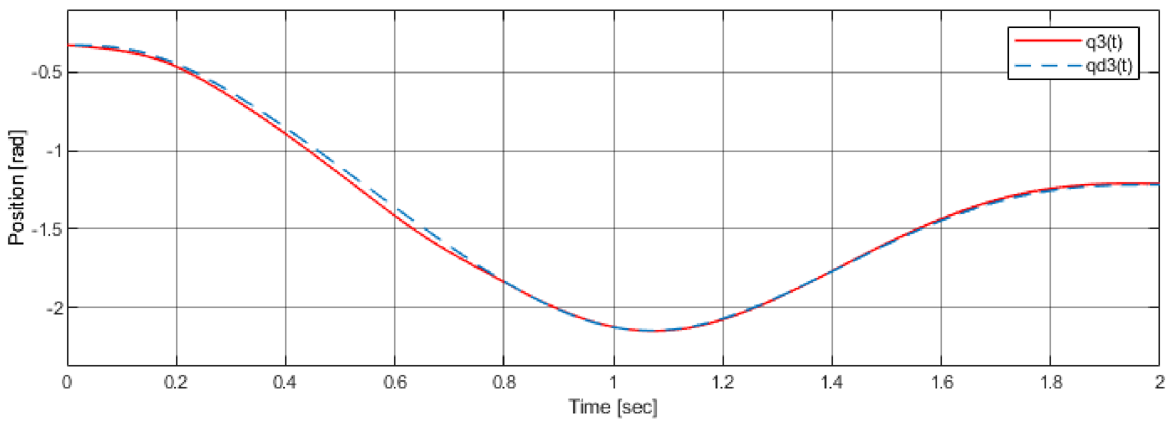


Figure 7.20(c): q3(t) vs. qd3(t), Inverse dynamics.

Figure 7. 20: q(t) vs. qd(t), Inverse dynamics.

Tracking performance is more accurate because this control method compensates all the nonlinearities of the dynamics model, and Figure 7. 21 reveals that error is about zero.

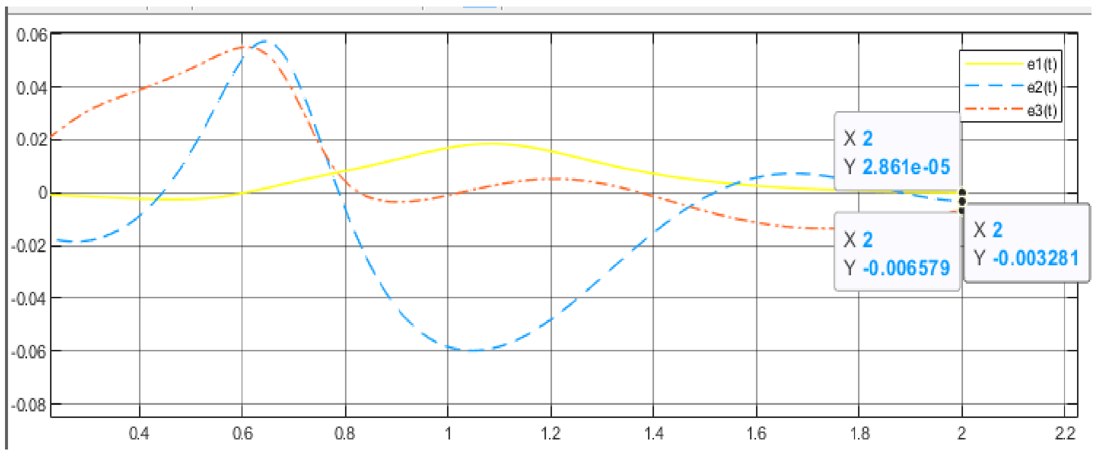


Figure 7. 21: The error between q(t) and qd(t), inverse dynamics.

The greatest advantage that this control method requires less torque, see Figure 7. 22, because it has previous knowledge of the varying configuration effects and system dynamics, this allows to cancel these nonlinear effects not to only compensate them. We notice that this compensator continues to cancel the nonlinearities especially the gravity component on each joint, hence the torques will be developed during the whole motion.

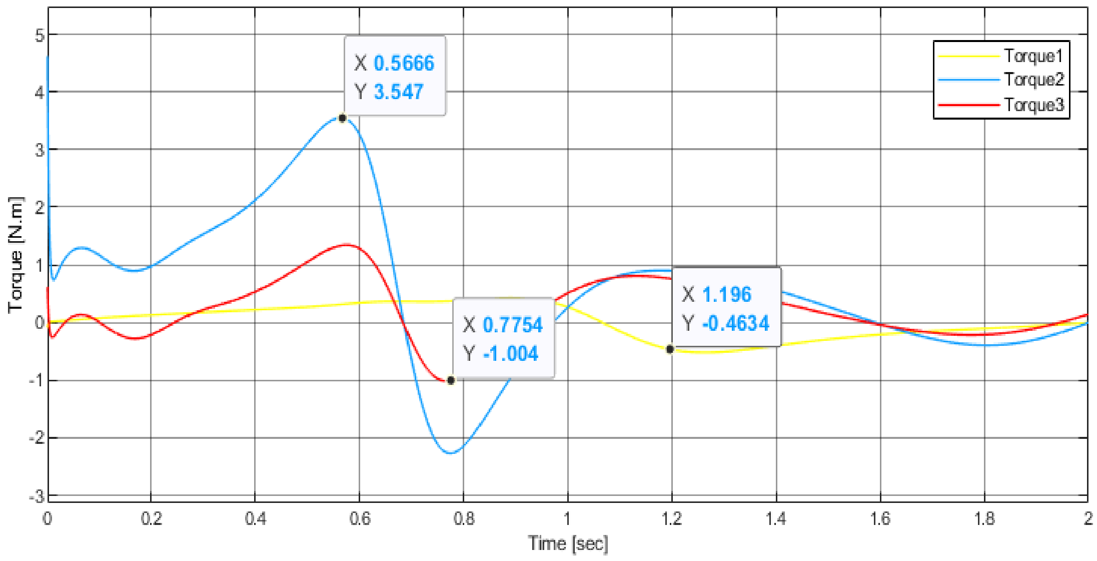


Figure 7. 22: Torque required at each joint- Inverse Dynamics.

7.3.7 PD with Gravity Compensation

A proportional-derivative (PD) control with online gravity compensation is proposed for regulation tasks of robot manipulators with rigid links. The control law is an extension of a previous PD control with constant gravity compensation at the desired configuration. It is based on a gravity-biased modification of the desired link deflection and requires measuring only position and velocity on the joint side. Global asymptotic stability of the control law at the desired robot configuration is proven via Lyapunov argument.

Figure 7. 23 [5] shows the block scheme of joint space PD control with gravity compensation,

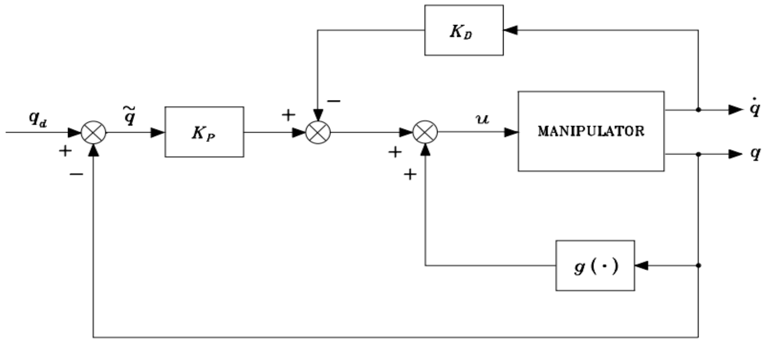


Figure 7. 23: scheme of joint space PD control with gravity compensation.

The law of control (7.19) depended in gravity matrix, error, and speed of all joints.

$$u = g(q) + K_p q^{\sim} - K_D \dot{q} \qquad 7.19$$

The gain K_p and K_D we should diagonal matrix

$$K_P = diag[\omega_n^2 , \omega_n^2, \omega_n^2] = diag(150, 25, 25)$$

$$K_D = diag[2 * \zeta * \omega_n ,2 * \zeta * \omega_n, 2 * \zeta * \omega_n] = diag(0.1, 0.1, 0.1)$$

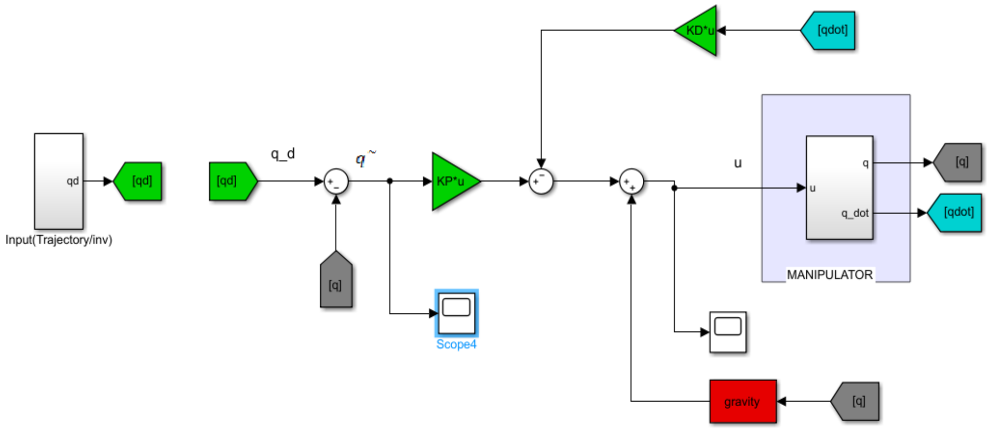


Figure 7. 24: PD with gravity compensation using MATLAB SIMULINK

Design and simulate the control scheme in SIMULINK, see Figure 7. 24 showing the block diagram.

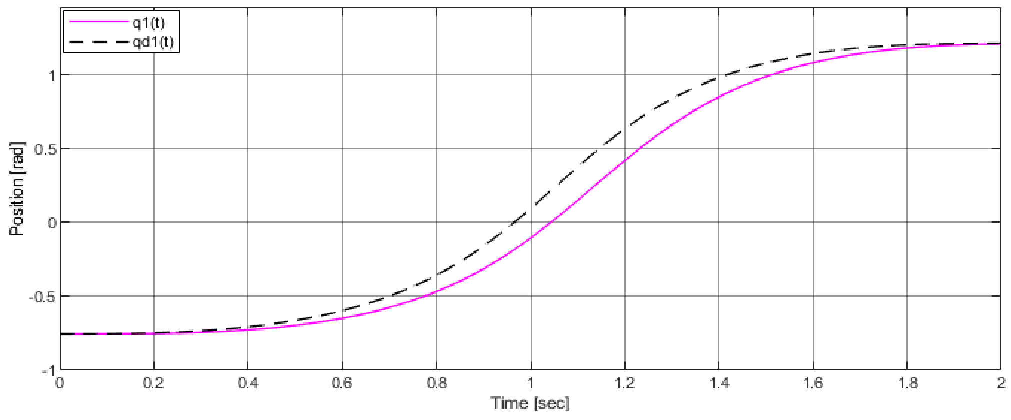


Figure 7.25(a): q1(t) vs. qd1(t), PD with Gravity

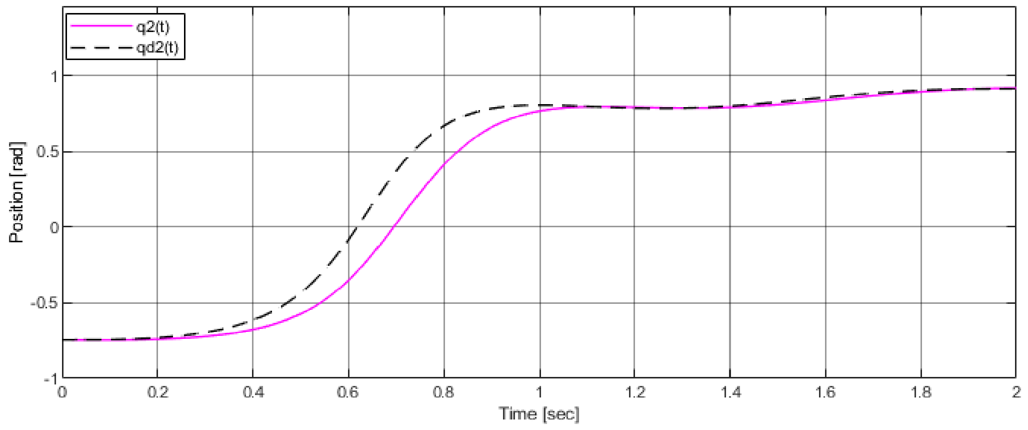


Figure 7.25(b): q2(t) vs. qd2(t), PD with Gravity

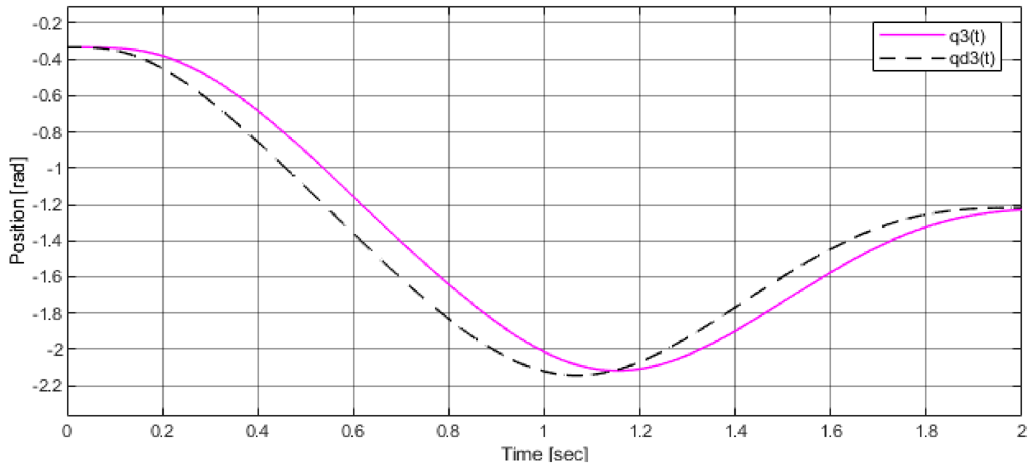


Figure 7.25(c): q3(t) vs. qd3(t), PD with Gravity

Figure 7. 25: q(t) vs. qd(t), PD with Gravity

The simulation results in Figure 7. 25 reveal that PDG compensator has an acceptable tracking performance with a notable tracking error during the whole motion. This is expected because that the nonlinear coupling terms affect on the tracking of the desired configurations and there is no integral action for error reduction during the process. The tracking error at the final instant is approximately zero, see Figure 7. 26.

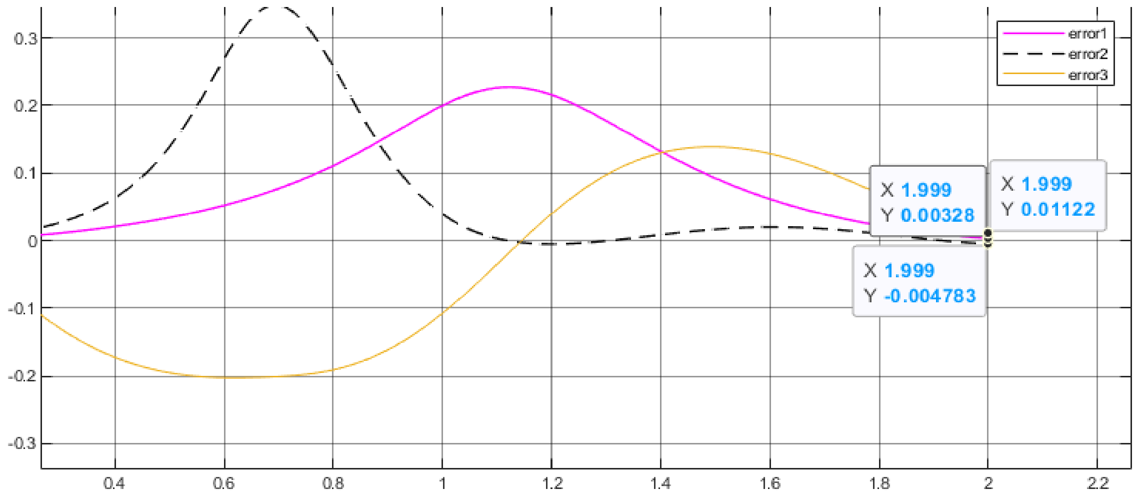


Figure 7. 26: Resulting error for motion tracking – PD with Gravity

The torque is reasonable and it seems less than it in the inverse dynamics because there is no compensating for nonlinear effects, see Figure 7. 27,

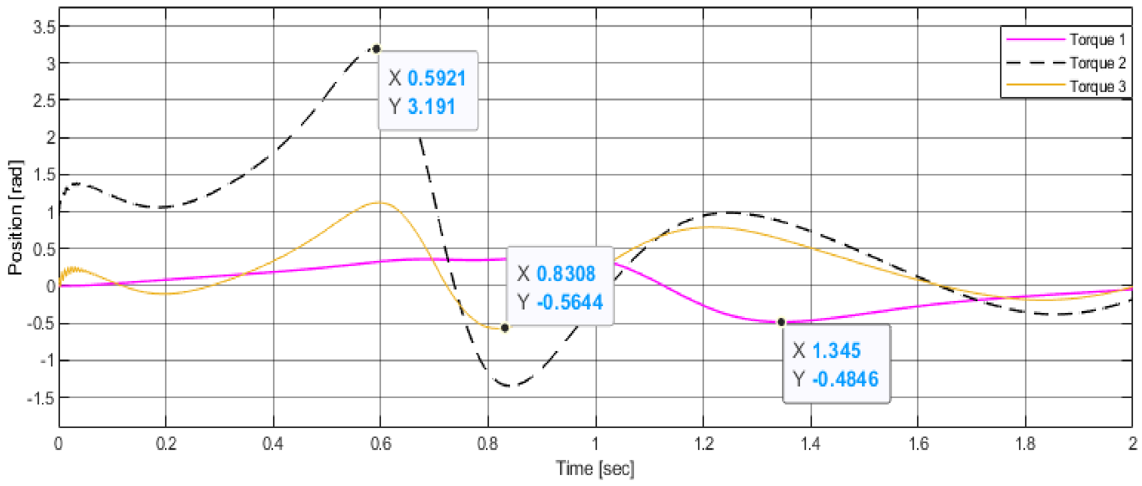


Figure 7. 27: Joint efforts – PD with Gravity

7.4 Motion control in operational space

7.4.1 Introduction

This approach considers that control is developed directly in the operational space when a trajectory is assigned for the end-effector motion. If the motion is specified in the joint space, then it can be translated to the task space through the direct kinematics. It is known that control actions are performed in the joint space, but the error Δx is specified in the operational space.

To reduce the error Δx, it is treated first through a matrix gain which generates forces for the end-effector that drive the joints to move along a direction reducing the position deviation Δx, these forces have to be transformed to generated toques into the joint space.

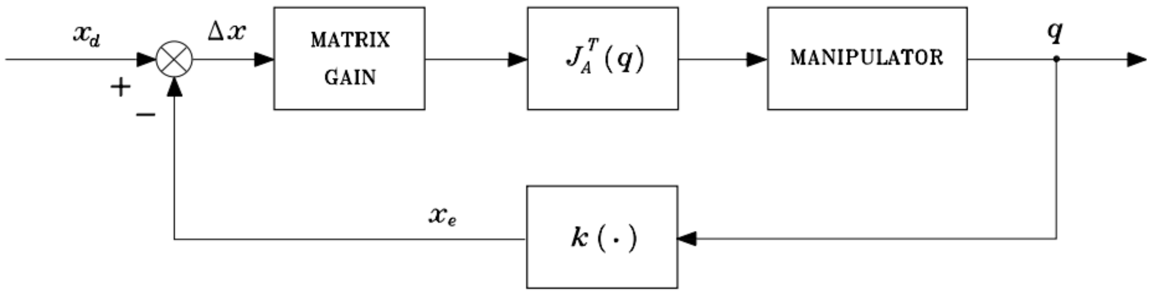


Figure 7. 28: Block scheme of Jacobian transpose control.

Figure 7. 28[5] shows the general control scheme of the Jacobian transpose control, the Jacobian transpose transforms the computed end-effector forces to joint torques. This scheme is not guaranteed to be effective in terms of stability and trajectory tracking accuracy. The PD control with gravity compensation is the equivalent mathematical solution for the above scheme that will be introduced below.

7.4.2 PD control with gravity compensation

The control law is an extension of a PD control with constant gravity compensation (PDGC) at the desired configuration, Figure 7. 29 [5]. Operational space trajectory specified motion assigned in the operational space, if measurements of operational space quantities are indirect, the controller has to compute the direct kinematics function. Whenever it is desired that the end-effector motion follows a geometrically specified path in operational space, with joint space stability analysis, given a constant end-effector x_d, it is desired to find the operational space error,

$$\tilde{x} = x_d - x_e$$

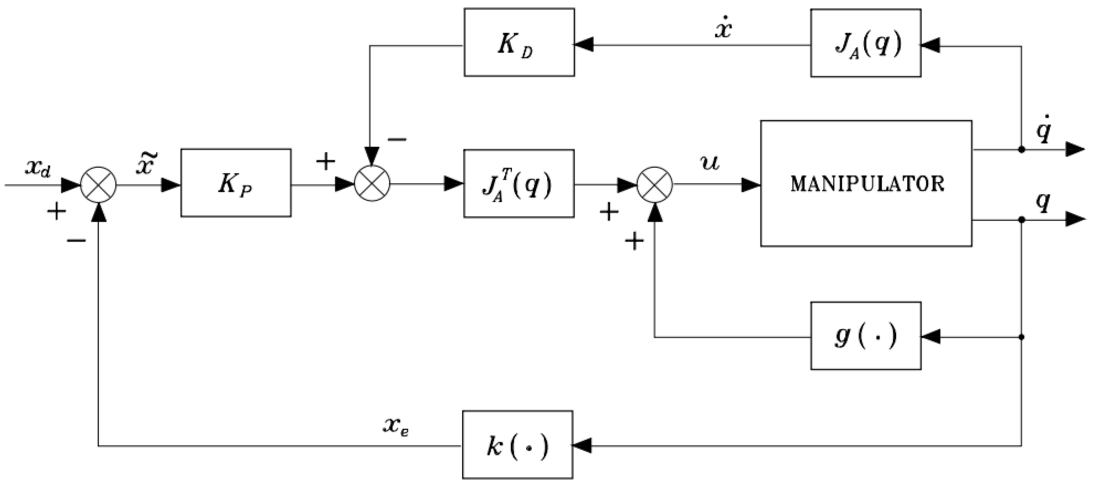


Figure 7. 29: Block scheme of operational space PD control with gravity compensation.

Global asymptotic stability of the control law at the desired robot configuration is proven via Lyapunov argument Theorem.

Control law suggests,

$$u = g(q) + J_A^T K_P \tilde{x} - J_A^T K_D J_A \dot{q}$$

PD controller necessary to measure q to update both $J_A^T(q)$ and nonlinear gravity compensation term $g(q)$ are evaluated (on-line).

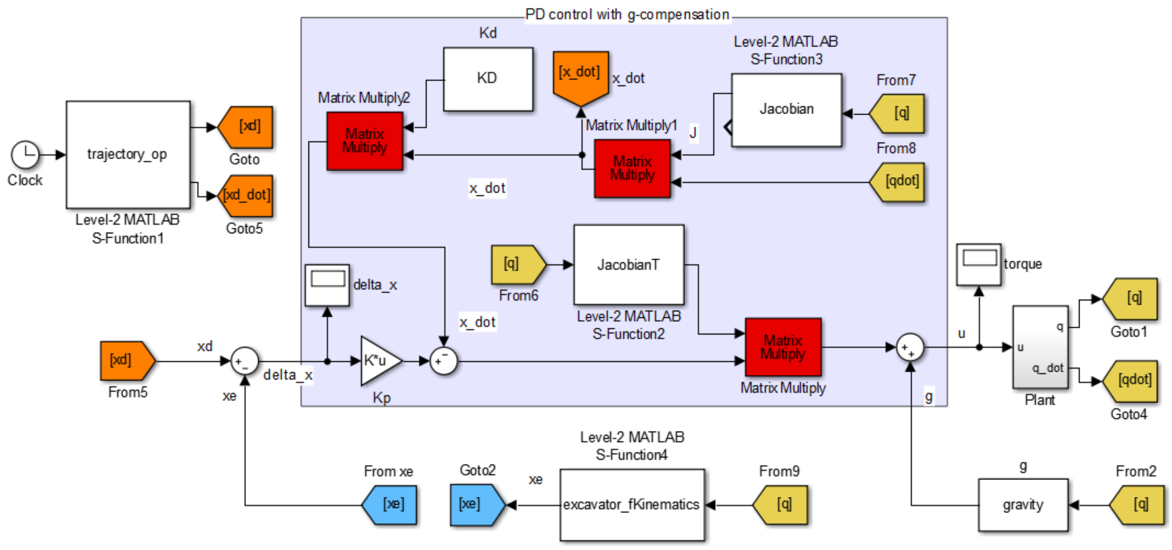


Figure 7. 30: Block diagram of PD with Gravity in operational space

Finding the gains K_P and K_D with the same design requirements $\zeta = 1$ and $\omega_n = 90\ rad/s$ for all joints by (7.18),

$$K_P = 1e3 * diag[\, 2.2946 \quad 1.4159 \quad 0.1863\,]$$
$$K_D = diag[\, 50.9904 \quad 31.4640 \quad 4.14]$$

Design and simulate the control scheme. Figure 7. 30 shows the block diagram.

Used symbols in block diagram:

xd: $P_d(t)$ [m], xd_dot: $\dot{P}_d(t)$ [m/s], xe: $P(t)$ [m], delta_x: $\tilde{x}$

By Figure 7. 31, the transient response and steady state error are acceptable in x and y motions, but the motion in z has problems in performance, since it is affected by the gravity force in $(-z)$ and the gravity compensation action, and does not reach the steady-state. Figure 7. 32, errors in x and y are approximately zero while error in z is $e_{ss3} = 34\%$.

state. Figure 7. 32, errors in x and y are approximately zero while error in z is $e_{ss3} = 34\%$.

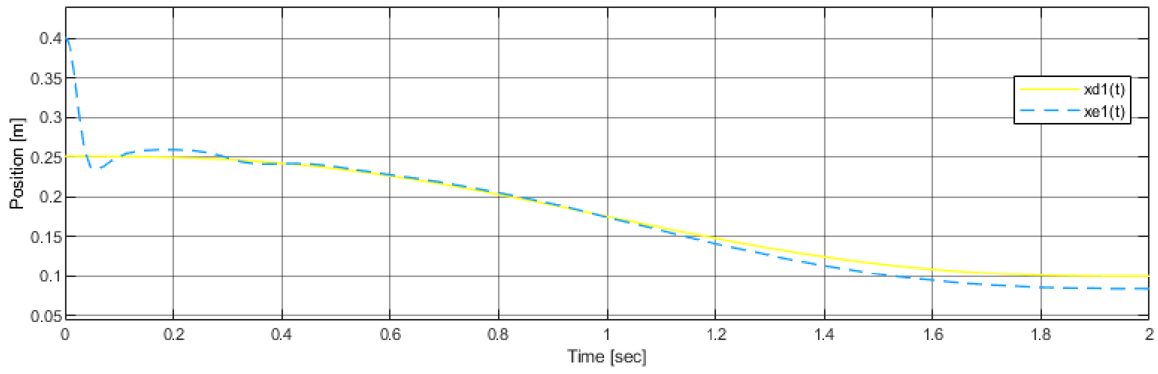


Figure 7.31(a): xd1(t) vs. xe1(t), PDGC

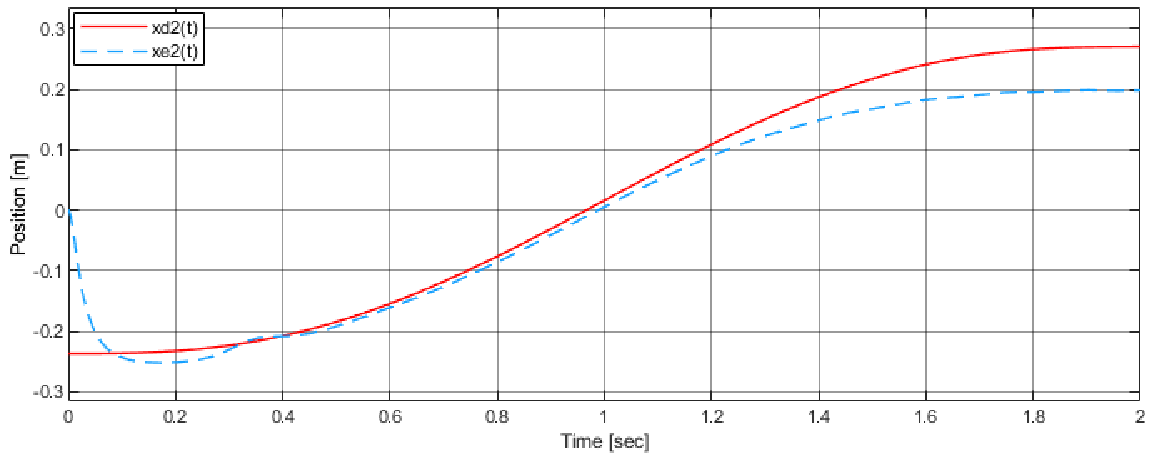


Figure 7.31(b): xd2(t) vs. xe2(t), PDGC

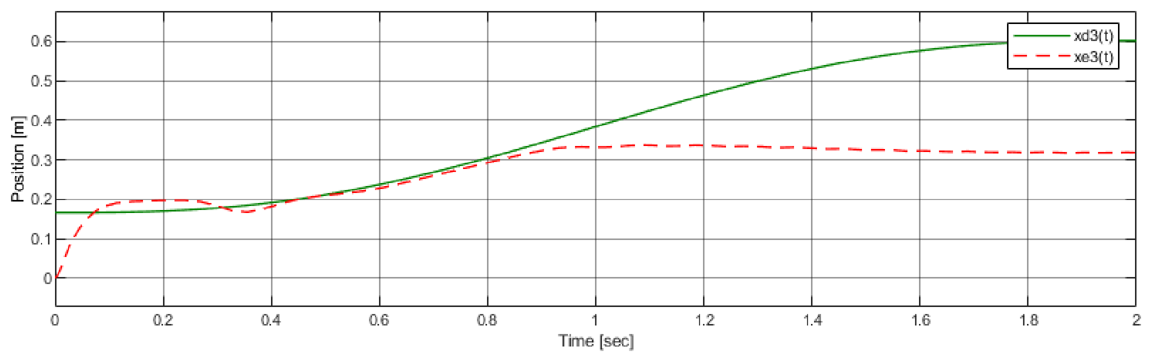


Figure 7.31(c): xd3(t) vs. xe3(t), PDGC

Figure 7. 31:(a),(b) and (c) xd(t) vs. xe(t), PDGC.

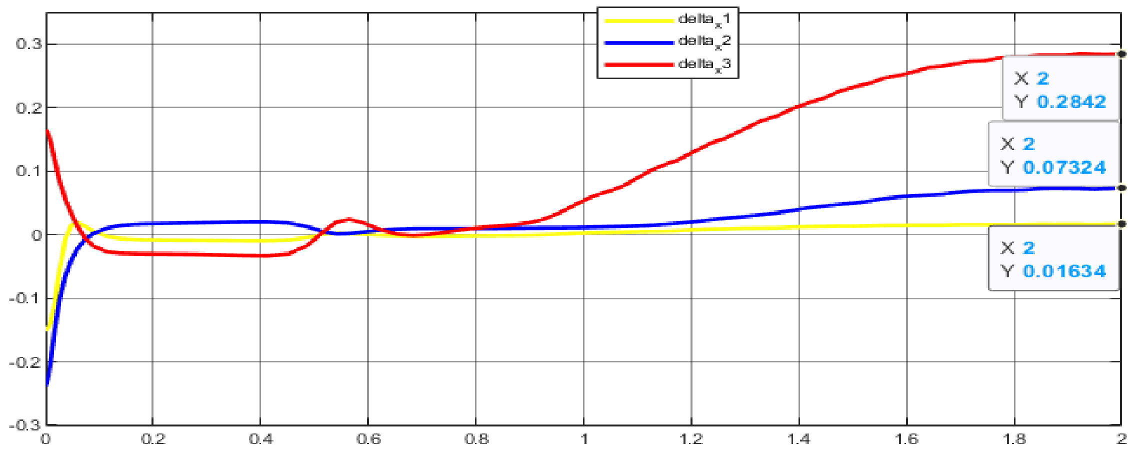


Figure 7. 32: Error between xd and xe.

Torques at joints seems a little bit larger than the others, Figure 7. 33. As seen, torque curves are not quite smooth and did not give accepted performance as the previous controllers.

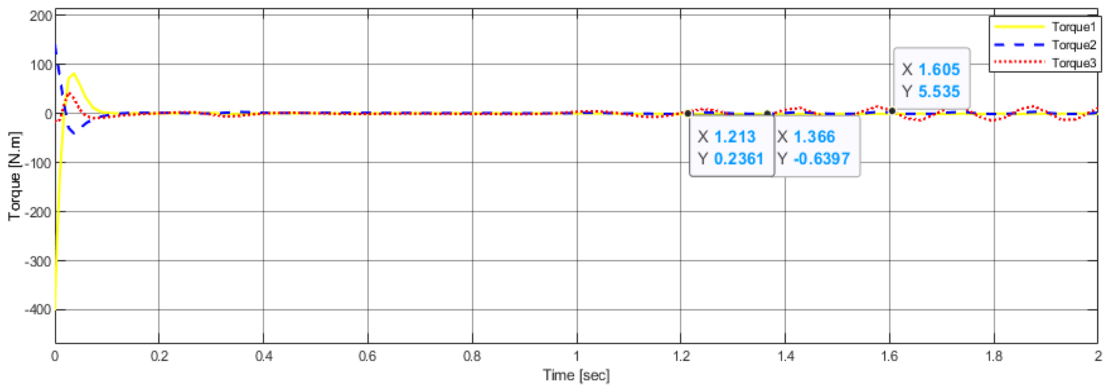


Figure 7. 33: Torques of joints.

7.5 Conclusion

In terms of tracking of trajectory, required torques, disturbance rejection and robustness to parameter changes. A comparison between the control schemes in terms of these aspects according to our design and dynamic system will be introduced.

High tracking performance in I-PD and inverse dynamics control schemes while in PID and PDG in joint space is accepted, with a little steady-state error, but PDG in operational space has the worst performance especially in z direction.

The lower consumed torques for the desired specifications occurs in inverse dynamics and PDG control in joint space. In contrast, in general PDG in operational space has the highest joint torques, and I-PD consumes torques at the starting with short interval due to integral action.

Disturbance rejection and Robustness to parameter changes are the highest for the inverse dynamics method, and moderate in the others, except the PDG in operational space which consumes more power to reject a disturbance, and has less robustness.

We conclude that centralized control in joint space (inverse dynamics and PDG schemes) is the best in terms of all these aspects, especially for the direct drive actuators, the inverse dynamics method gives the choice to select actuators without high gear ratios (less cost). It is intended to implement centralized control schemes in real control applications.

Another problem occurs about the sampling time. If the sampling time is set to be larger than $\frac{\pi}{\omega_B} = \frac{\pi}{90} = 0.038s$ of the closed loop system, then the response becomes oscillatory and tends to change its desired shape. The higher sampling time, the less input samples taken for the input trajectories and the worst performance occurs, so the system loses information about the desired input references (signal distortion), this phenomenon called "Aliasing". For more enhanced performance, the sampling time $T_{sampling}$ must be less or equal to $0.1 * \pi/\omega_n$. When $T_{sampling} > 0.038s$,

$\omega_s > 2\omega_B$

Where,

ω_B: is the maximum bandwidth frequency of the closed loop system.

ω_s: is the sampling frequency.

$\frac{2\pi}{T_s} > 2 * 2 * \pi * f_B \rightarrow \frac{1}{2*f_B} > T_s$, for real time control systems, a reduction factor of 0.1 is multiplied by T_s:

$\frac{\pi*0.1}{\omega_B} > T_s$, is the favorable sampling time value for real-time discrete control systems.

Chapter Eight: Prototype Building, Experiments and Results

8.1 Introduction

The purpose of this section is to evaluate the overall practical work on the project. This work addresses the design and construction issues of the excavator's arm robot. First of all, the robotic arm performance analysis has been accomplished using MATLAB software. The obtained knowledge has been utilized to develop suitable algorithms for analyzing robotic arm kinematics and control that perform tasks that programmable stand-alone code by a Raspberry Pi. On the other hand, the manipulator has four degrees of freedom. Three degrees of freedom correspond to the robotic arm, and one of the end-effector. Moreover, the necessary electronic modules in order to allow a successful communication with Raspberry Pi microcomputer.

8.2 Prototype structure and system components

This section introduces the building of the prototype structure and interrelated system components. Various challenges were revealed during the development, interfacing and operating of the robot electro-mechanical components.

8.2.1 Build robotic soil excavator prototype

Analyzes a full range of structural design and a comprehensive selection of components. It illustrates the details and key points of producing a type of robot from an idealization to actual production, as well as whether the actual design parameters will meet the actual needs (through SOLIDWORKS simulation), which will build-up to the excavator robot arm with a completely new design.

Having the actuators, it is only required that the structure as the robot arm. For this, it is essential that the pieces which form the union and support for all the excavator arm robot system must be strong enough to support the weight and force of the actuators, but also should be light and not waste power trying to move its own weight. So based on the available motors, and calculating the maximum torque, the materials and final dimensions of the prototype were selected.

Mechanical parts

- Links of the excavator:

The base area we designed consists of two parts. The first one is the fixed, which provides fixed support to the remaining moving arms, The second parts are the rotating ones, where the material chosen for the base is steel. They are together assembled as shown in the Figure 8. 1,

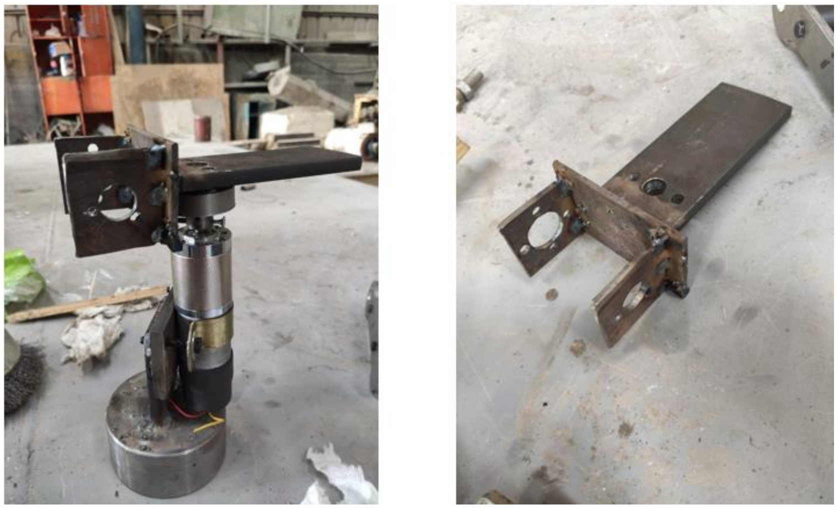


Figure 8. 1: The base of Excavator manipulator.

The boom is the second link the boom associated to the base with a joint 2, this link is shown in the Figure 8. 2, The arm is a link connecting of the boom with bucket shown below, we chose to design the boom and arm parts with aluminum. and the bucket is the end-effector of excavator is shown in the Figure 8. 3,

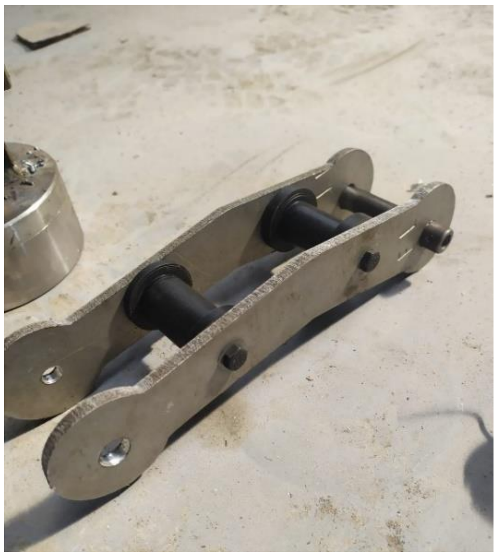


Figure 8. 2: Boom Link-2 of manipulator.

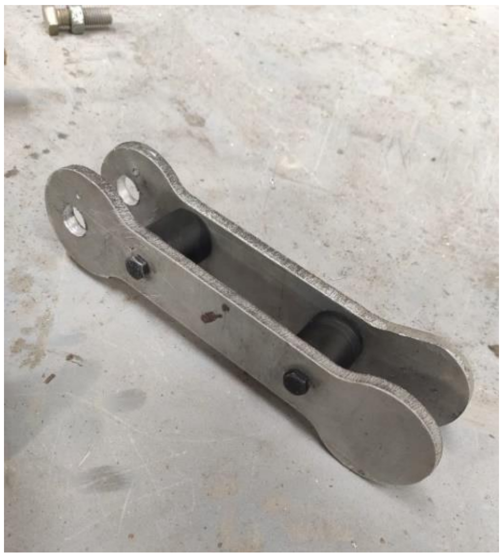


Figure 8. 3: Arm Link-3 of manipulator.

To design a bucket part affordable 3D printing technology that’s favored for rapid and low-cost prototyping, Figure 8. 4 shown the printed bucket with shaft,

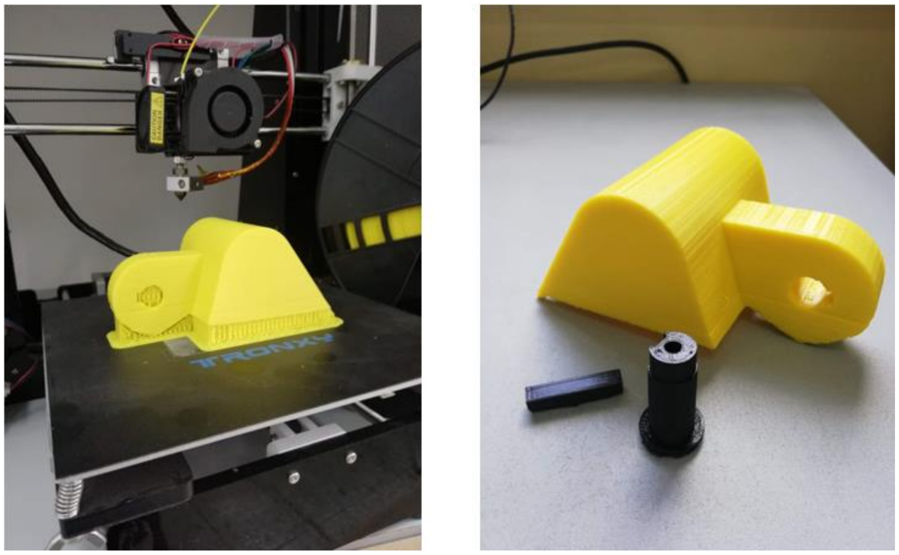


Figure 8. 4: Bucket end-effector and shaft printed of the manipulator.

- Shafts of motors:

In the structure of the robot, the design of the drive shaft in the transmission process is extremely important shown in the Figure 8. 5, and the shafts of manipulator torque moment are under load. As a result, the chosen material for the shaft is steel.

The key is used to connect the links to the shafts for the manipulator. The key prevents relative rotation between the two parts and enables torque transmission.

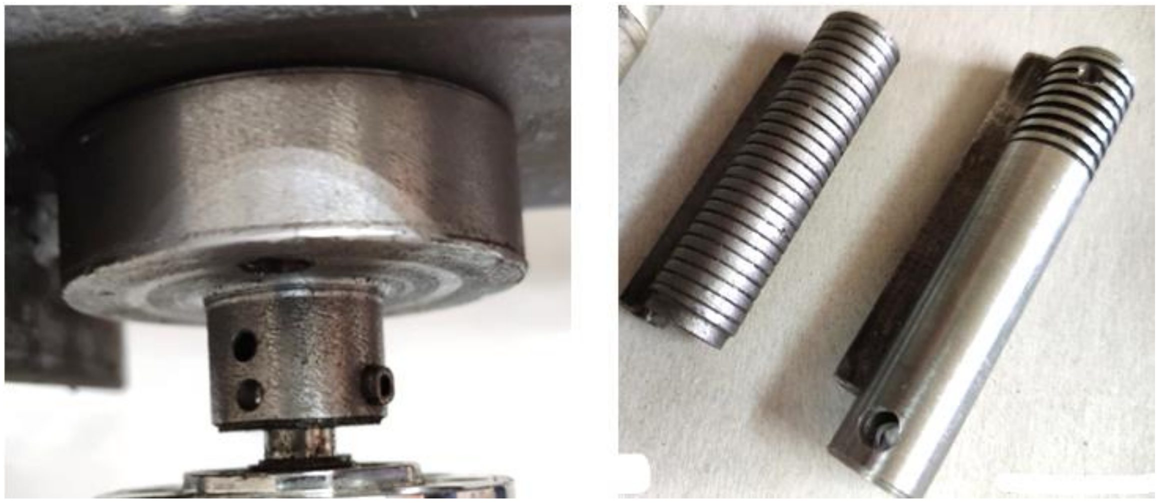


Figure 8. 5: The shafts of motors.

- Excavator manipulator prototype:

After the design of each individual part, the next step is that we need to assemble the parts to form a complete excavator arm. The assembly has the base, boom, arm and end-effector the bucket shown in the Figure 8. 6.

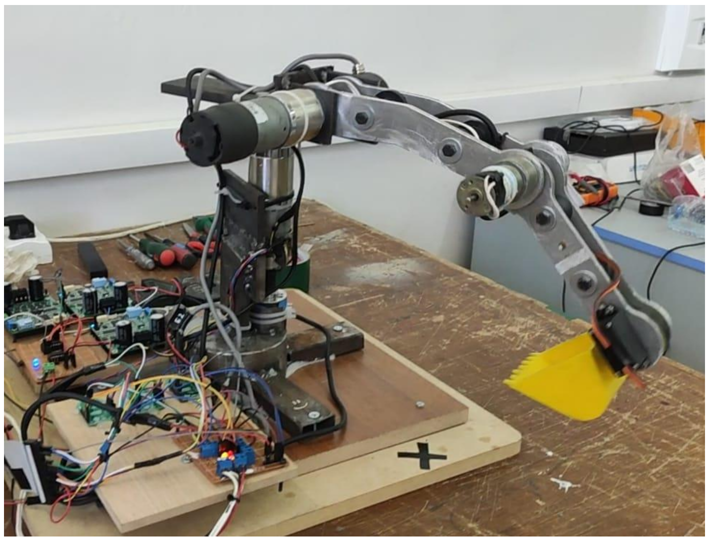


Figure 8. 6 Prototype of Excavator manipulator.

- Specifications of parts:

The critical specifications and parts weight sotted out in Table 8. 1, manipulator is mostly built of common materials, Steel and aluminum are the most often used materials for the arms and bases of robots. Aluminum is a softer material and therefore easy to work with while steel is stronger.

Table 8. 1: Specifications of the manipulator mechanical parts.

Parts/ Specifications	Material	Weight [grams]	Length [cm]	Plate Thickness [cm]
Link 1	steel	1010	20	0.5
Link 2	aluminum	360	20	0.6
Link 3	aluminum	255	15	0.6
Link 4	PLA carbon fiber	110	10	0.3
Parts/ Specifications	Material	Weight [grams]	Length [cm]	Diameter [cm]
Shaft 1	steel	256	3.5	20
Shaft 2	steel	72	7.2	1.4
Shaft 3	steel	50	4.8	1.4
Shaft 4	PLA carbon fiber	10	3.5	1.4

The total weight of the manipulator structurer on the first motor is 2.18 [kg].

The total time and number of iterations required for designing functional, assembling multiple workpieces is a complicated task in design robots. Thus, verification of the design in assembly operations helps to measure the performance of the robotic excavator.

8.2.2 Electrical components

Identification of motor parameters

As mentioned previously in Chapter 5, the motors are selected to provide a suitable torque for our application for each robot joint. The selected motors were tested in terms to the stall torques and currents, speeds and consumed power, since the tested motors are used before and do not have datasheets. Each motor is tested by hanging a known and changeable mass m_{test} at one end of an arm (of negligible mass) with known length l_{test}, see Figure 8. 7. The motor was operated with its maximum voltage V_{test} to give the stall torque τ_{stall} with measuring the stall current I_{stall}. Motor speed were measured at no load $\omega_{no\ load}$ using a coupled encoder with gear shaft, and these data were concluded with the existence of the gearbox (Torques and speeds are measured at gear side not motor side).

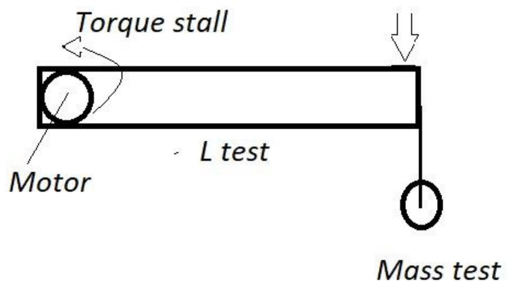


Figure 8. 7: An arm with a suspended mass

Another essential parameter is the gear reduction ratio, it was identified by rotating the gear shaft one complete revolution, and observing the number of rotations n_{motor} at the side of motor shaft with encoder sensor, giving a gear ratio of $r = 1{:}\,n_{motor}$.

- Motors at first and second joints:

The first two motors are same, their parameters were identified with an arm of length $l_{test} = 0.36\ m,$ with a suspended mass of $m_{test} = 2Kg$ at the arm end that makes the motor to be stalled with torque $\tau_{stall12} = 7\ N.m,$ and current $I_{stall12} = 5A$. The motor is operated with voltage input of $V_{test} = 24V$. The resulting torque-current constant is $T_{c12} = \frac{\tau_{stall12}}{I_{stall}12} = \frac{7}{5} = 1.4\ N.m/A$. The resulted ω_{load12} is about $20\ RPM$.

One revolution at gear side gives 70 revolutions at motor side with a gear ratio of $r = 1{:}70$.

- Motor at third joint:

The third motor were coupled with an arm of length $l_{test} = 0.15\ m$, with a suspended mass of $m_{test} = 2Kg$ at the arm end that makes the motor to be stalled with torque $\tau_{stall3} = 3\ N.m$, and current $I_{stall3} = 4A$. The motor is operated with voltage input of $V_{test} = 24V$. The resulting torque-current constant is $T_{c3} = \frac{\tau_{stall3}}{I_{stall3}} = \frac{3}{4} = 0.75\ N.m/A$. The resulted ω_{load3} is about 35 RPM.

One revolution at gear side gives 60.5 revolutions at motor side with a gear ratio of $r = 1{:}60.5$.

- Motor at fourth joint:

The last joint has a limited task of motion, the servo is selected to perform such motion. The required torque is tested by rotating the bucket with a payload of $m_{test4} = 0.5\ Kg$ and arm length of $l_{test4} = 0.05m$, giving $\tau_{required4} = 0.25\ N.m$ which is smaller than the rated servo torque of $\tau_{servo} = 0.92\ N.m$, and the consumed maximum current for this task equals $I_{consumed} = 0.6\ A$. Maximum speed at $\tau_{required4}$ is about $\omega_4 = 60^o/s$ that is fast enough for opening or closing the bucket with range of $0^o - 180^o$. As a result, the servo selection is verified in terms to the robot payload, power and speed.

- Specifications of SOLO drivers:

Each SOLO driver is supposed to provide the required power to actuate the motors with the desired torque and speed. For the first three motors, the maximum current needed at the rated torque is $5A$ with speed of $35RPM$, while driver gives maximum continuous output current up to 32A with maximum voltage of 58V. The driver's power specifications are theoretically suitable for our application. By the way, for future work if the structure is modified with steel material to handle more payloads or interact with the environment, new motor selection will be introduced to provide higher torques that require more power allows system upgradability.

Accuracy of robot end-effector position

One of the main barriers in the excavator arm robot has been the reduction of error between the tool frame at each joint and the goal frame of the end-effector. The sources of this error are readily identified. Modeling differences between the controller and the robot account for most of the errors between the base frame and the tool frame. Inaccurate fixturing and manufacturing processes can account for the differences between the station and goal frames.

One requirement is the end-effector position accuracy, it was set to be $3\ mm$. There are many factors that influence the robot accuracy, such as encoders resolution, backlash of gearboxes and the control quality.

- Encoder resolution:

Encoder resolution affects the goal frame accuracy. Each joint has a specific encoder resolution, the first and third encoders have a resolution of 400 pulse/rev while the second one has 2048 pulse/rev resolution. According to the base frame of the manipulator, maximum distances between each joint frame and the goal frame along $x - axis$ are $0.43m$ for joint 1, $0.35m$ for joint 2 and third one is $0.15m$, see Figure 4.1.

Calculating the resultant end-effector position accuracy due to the measurement resolution. Joint 1 gives $0.43 \sin 0.9^o/pulse = 0.00675m$, joint 2 gives $0.35 \sin 0.1758^o/pulses = 0.00107m$ and finally joint 3 gives $0.15\ sin\ 0.9^o/pulse = 0.0023m$ are the accuracies on the goal position. The total and maximum accuracy on the end-effector is equals $10\ mm$.

If the desire is to improve upon the resolution or accuracy of the robot, a precise metrology system is needed to perform these measurements. However, for our application that does not require high repeatability or positional accuracy in excavation and throwing tasks, a $10\ mm$ accuracy is initially accepted and we note that control algorithm could improve the final accuracy.

- Backlash of motor gearbox:

Backlash refers to the angle that the output shaft of a gearhead can rotate without the input shaft moving. It can be a serious issue in controlling endpoint motions, due to the limited resolution of sensing the gearhead output shaft angle using an encoder attached to the motor shaft, see Figure 8. 8.

By observation, each selected motor has a significant gearbox backlash angle that cannot be ignored in control problems. The accurate method the backlash could be measured for each gearbox, is to rotate the gearbox shaft a known angle that is measured by encoder, then observing the input shaft to stay constant during this rotation, accordingly at the first motion of the input shaft, the backlash angle is recoded. Another method is to measure the angle directly by effecting the gear shaft with the limited motion of backlash itself.

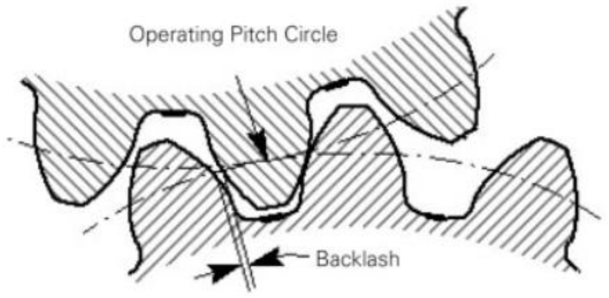


Figure 8. 8: Backlash of motor gearbox

The measured angles of backlash are 5^o for first two motor gearboxes and 1^o for the third one. These angles are large enough to affect negatively on the tracking performance and the final end-effector position. In control problem, we observe that joint 2 has the best tracking performance due to its high encoder resolution, and joint 3 has an acceptable tracking error, but the first joint is affected negatively by the encoder resolutions and backlash with about $\pm\ 2.5^o$ tracking error.

The backlash problem cannot be avoided directly, however the gearbox drive has a notable advantage that it reduces the nonlinear coupling effects on the motor side, means that to rotate the input shaft it is needed a large and fast rotational motion on joint side, and for intermediate speeds, the coupling effects and disturbances will be less on joint side, accordingly they are approximately goes to zero on motor side.

Finally, all these factors propagate the positional errors for the robot end-effector while a robust tracking control algorithm can reduce such effects. They will be discussed with the tracking responses in motion control results.

8.3 Information processing techniques and software capabilities

8.3.1 MATLAB and Simulink software

MATLAB and Simulink are widely used in control systems at all stages of development from plant modeling to designing and tuning control algorithms. Using these tools for Model-Based Design, it allows to not only simulate our control algorithms but also the physical hardware. By automatically generating code for the control software and the simulated physical model, and reducing development time and implementing changes quickly. We visualized simulation and test results for many control methods (Chapter 7), which gave us confidence in the design that will finally implemented in real controller such as Raspberry Pi 4.

Once we have designed our control system algorithms, we can refine them for implementation in real control processors. It was intended to use MATLAB and Simulink for hardware-in-the-loop (HIL) testing by executing the control algorithm on the RPi embedded controller, and running the plant model in real time on a target computer connected to this controller. In practice, we have tested MATLAB and SIMULINK toolbox for interfacing them with RPi hardware, and it reveals that these tools are not fully matching or accessing the RPi 4 hardware in terms to usability of transferring protocols as SPI and UART or even controlling hardware PWM pins. Another consideration, the electrical components as feedback and actuating devices are accessed by Python libraries.

8.3.2 Robot Operating System - ROS

ROS is referring to the Robot operating system, it is a software platform that provides libraries and tools to help software developers create robotics applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. ROS is similar in some respects to 'robot frameworks'. We redirect the compass to another alternative of software which is the Robot Operating System (ROS) for its common usability for robot systems.

- Why ROS has been used in this project?

The main purpose of implementing our control algorithms in ROS is the suitable code organization. The advantage in subdividing the normal code to many software pieces makes the programming more convenient and easier to modify for future goals. ROS provides integration between the written software and hardware modules, many tasks can be distributed in many ROS nodes, each node can communicate and exchange information to others through registering it to the Master node. One way of nodes communication in ROS is using a Publisher/Subscriber architecture like the Figure 8. 9.

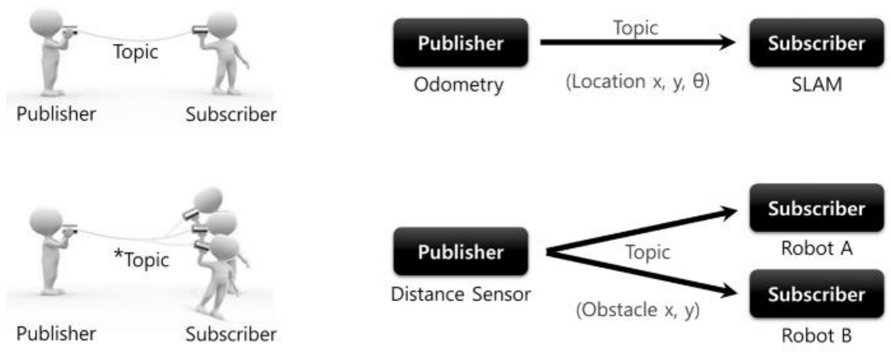


Figure 8. 9: Topic Message Communication. [16]

A ROS node is the equivalent of a function or a mini-program. Its role is to publish information (feedback, position, efforts …) that we call messages, each message is published on a specific topic. A Node that publishes information is called a Publisher. Those who listen to and use this information are called Subscribers. Generally, a node is both Subscriber and Publisher. Many nodes form a ROS package that can be run through that package name.

Initially, we have developed an embedded control system on RPi using the ROS topics. It implements many nodes, such as control process, joint trajectories, joint states and actuating joints, see Figure 8. 10 describes the package that contains nodes, topics and messages.

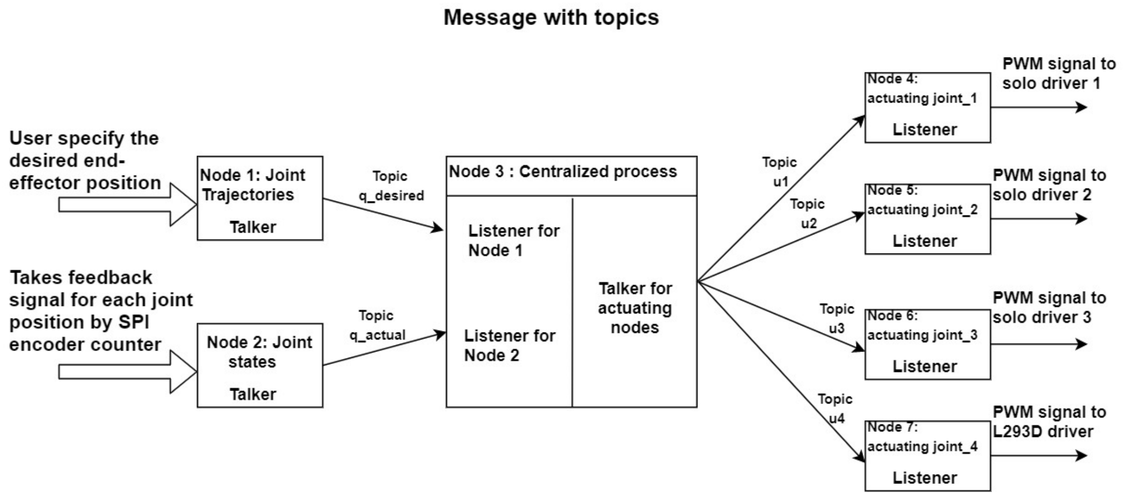

Message with topics
User specify the desired end-effector position
Node 1: Joint Trajectories
Talker
Takes feedback signal for each joint position by SPI encoder counter
Node 2: Joint states
Talker
Topic q_desired
Topic q_actual
Node 3 : Centralized process
Listener for Node 1
Listener for Node 2
Talker for actuating nodes
Topic u1
Topic u2
Topic u3
Topic u4
Node 4: actuating joint_1
Listener
Node 5: actuating joint_2
Listener
Node 6: actuating joint_3
Listener
Node 7: actuating joint_4
Listener
PWM signal to solo driver 1
PWM signal to solo driver 2
PWM signal to solo driver 3
PWM signal to L293D driver

Figure 8. 10: The developed ROS package using Topics

Describing each node function as follows,

Node 1: It receives the desired end-effector initial and final poses, then computes the trajectories in operational space, translates them to joint space through inverse kinematics, and finally publishes the joint trajectories to the node (3) of the main control process though the topic $q_{desired}$.

Node 2: It contains the hardware interface with the Encoder readers through SPI protocol, receives the joint states, then sends them to the control process through topic q_{actual}.

Node 3: It subscribes to the two topics, computes the joints error and publishes the joint torques to the actuating nodes (4, 5, 6 and 7) though topics (u_1, u_2, u_3 and u_4).

Nodes 4-7: each one receives the required joint effort, translates the torque to a suitable PWM signal and outputs it to the specific motor's driver.

The same concept can be applied by using ROS Services instead of Topics, see Figure 8. 11 and Figure 8. 12

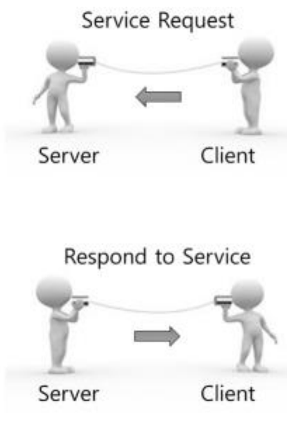


Figure 8. 11: Service Message Communication. [15]

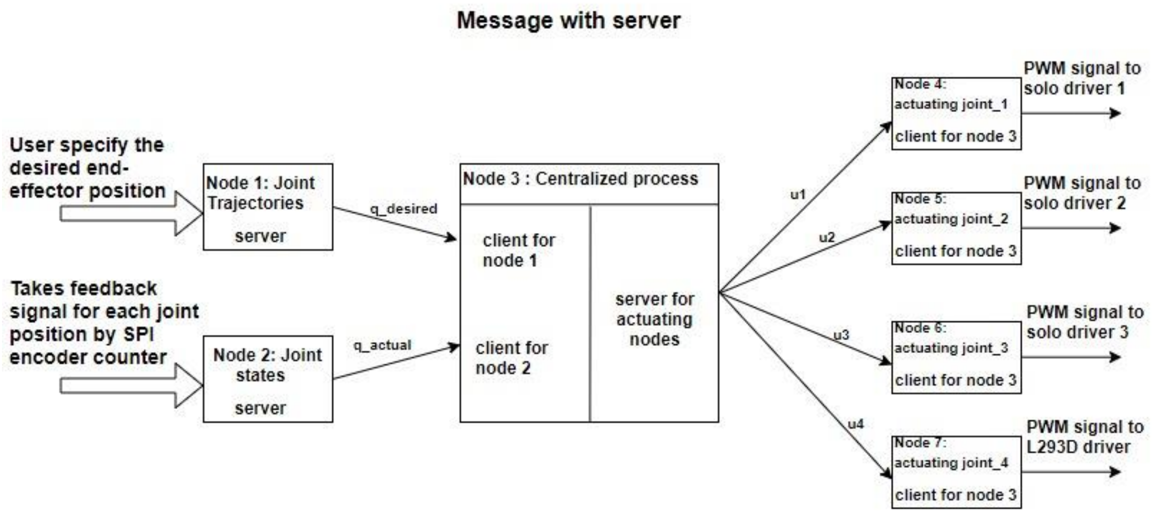

Message with server
User specify the desired end-effector position
Node 1: Joint Trajectories
server
q_desired
Takes feedback signal for each joint position by SPI encoder counter
Node 2: Joint states
server
q_actual
Node 3 : Centralized process
client for node 1
client for node 2
server for actuating nodes
u1
u2
u3
u4
Node 4: actuating joint_1
client for node 3
PWM signal to solo driver 1
Node 5: actuating joint_2
client for node 3
PWM signal to solo driver 2
Node 6: actuating joint_3
client for node 3
PWM signal to solo driver 3
Node 7: actuating joint_4
client for node 3
PWM signal to L293D driver

Figure 8. 12: The developed ROS package using Services

Each method has an advantage and disadvantage over the other. Using ROS Topics is faster in terms to publish/subscribe of messages, but there is no guarantee that all messages are received. While the using of ROS Services guarantees that the message will arrive when it requested, but this method is slower in exchanging messages (the client sends a request, then server responds and sending the message).

The above two methods were tested for controlling one joint. Initially, the response was accepted, however when the package was developed to control the whole robot, the responses were oscillatory and the system tends to be unstable. There are many factors affect negatively on the control process, after looking for many related researches, we conclude the following challenges:

1. In general, ROS is not a hard real time architecture:

If the correct computations cannot be provided at the correct time, this called soft or not real-time execution, so it’s about organizing processes not performance at all. Because that the most of Linux and many operating systems that ROS support do not guarantee a real time computing. It is known that Linux OS has no real-time capabilities in general, ROS gives the ability to run nodes always while other nodes being running for a specific time, this can be exploited carefully, we clam that if a node (or any program) is always executed without any interrupting from the other processes with acceptable process speeds, then the process is about to be a hard real-time with minimum latency. This clam has been verified with this research [16] that provides the proof of how the programs/nodes are organized and executed in real-time, the idea is to separate the execution of the wanted real-time process (as closed-loop control process) with an independent CPU-core of multi-core processor device (Micro-controller/ PC/ …), so the process will not be interrupted by any other processes.

It is worth to mention that there are another OS gives the hard real time capabilities as Xenomai OS, see [17].

A proposed solution is the using a Real-time OS such as Xenomai or Preempt patches with RPi Raspbian OS. ROS consumes too much of system resources, we need a solution to organize the system processes. This solution has been taken into account for a while, these patches make the OS acting like a real-time OS. Our purpose is not to convert the system to be a hard real-time, but is to exploit these solutions to make the system interrupts more organized. RPi has many software and hardware interrupts that affect negatively on the control process, such as peripherals interrupts (checking USB, Bluetooth, HDMI screen, etc.,), CPU interrupts and many more. Using those patches make your selected process with high executing priority between many other tasks, and this is urgent for the sampling times and real-time control requirements.

The negative side of these patching on our running Raspbian OS, is that it disables many essential hardware interrupts, such as SPI, UART and PWM's. This violates our needs for these peripherals. Since these patches are specially designed for Ubuntu OS, but this OS does not provide the full ability to reach the hardware interfacing techniques with the needed peripherals in RPi.

2. ROS versions and compatibility with Linux systems:

Most of ROS packages are designed and provided especially for Ubuntu Linux OS. While Raspbian OS that we use with RPi is not tested with all ROS packages yet. Using ROS on Raspberry Pi, it must to be connected to some hardware components to actuate the robot and get feedback data. A not-so-good practice here would be to import ROS in all classes and programs that directly deal with hardware. It means that we cannot really export any hardware library for other non-ROS projects, and would have a hard time integrating already existing libraries/plugins to your ROS application.

We success to develop the two previous packages for controlling robot using ROS Messages, but in ROS there are more developed packages for ROS control and Real-time applications. ROS_Control is a set of packages that includes controller interface, controller manager, transmissions, hardware interfaces and control toolbox. All these packages together allow to interact and control the joint actuators of the robot. The essential role for ROS control packages is the controller manager that updates the control process and organize the control actions in terms to read joint states, trajectories and producing joint efforts at a fixed sampling time.

These packages are not tested to work with Raspbian OS. It requires more time to learn work with ROS_control packages on RPi where it is not guaranteed to work efficiently, and other consideration about hardware integration between ROS_control and RPi is that Control packages use only C++ language, but our electrical components are developed to communicate with by Python language. As a result, choice of ROS control packages has been postponed to future work to allow testing the Robot motion control with a stand-alone code.

Excavator robot simulation using ROS tools:

Visualizing the Excavator arm using the ROS and the ability to generate 3D visualization of the arm, and you can see how a plan is provided by MoveIt and provide animation as a way of showing you how the planned trajectory will look like when executed by the robot.

Mechanical Model of the Excavator Arm robot, this part of the application does the following:

- Robot description - URDF

The URDF (Universal Robot Description Format) model created by SOLIDWORKS describes the excavator arm robot's physical description to ROS. These files used by the ROS to tell the computer what the robot actually looks like in real life. URDF files are needed in order for ROS to understand and be able to simulate situations with the robot. Figure 8. 13 Shows the URDF diagram for the excavator robot,

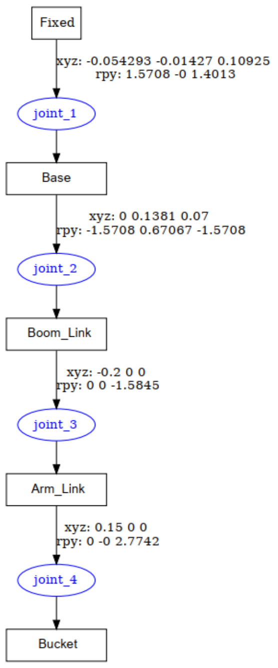


Figure 8. 13: URDF graph

The original file for performing simulation is the Universal Robot Description Format, or URDF. This xml-like file type is used heavily in ROS for simulation and testing; it is a supported file type for rviz and other ROS tools.

- Motion planning using MoveIt:

The basic task of the MoveIt! the system is to provide the necessary trajectories for the excavator arm of a robot to put the end effector in a target place. The Figure 8. 14 shows the communication with move_group node exchange commands with the user using ROS actions and services.

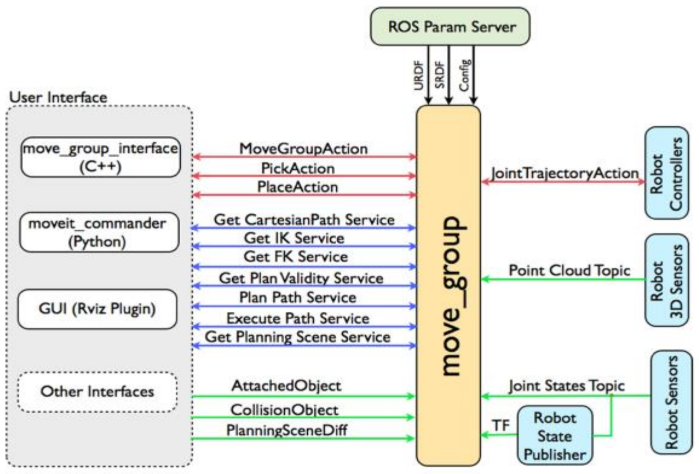
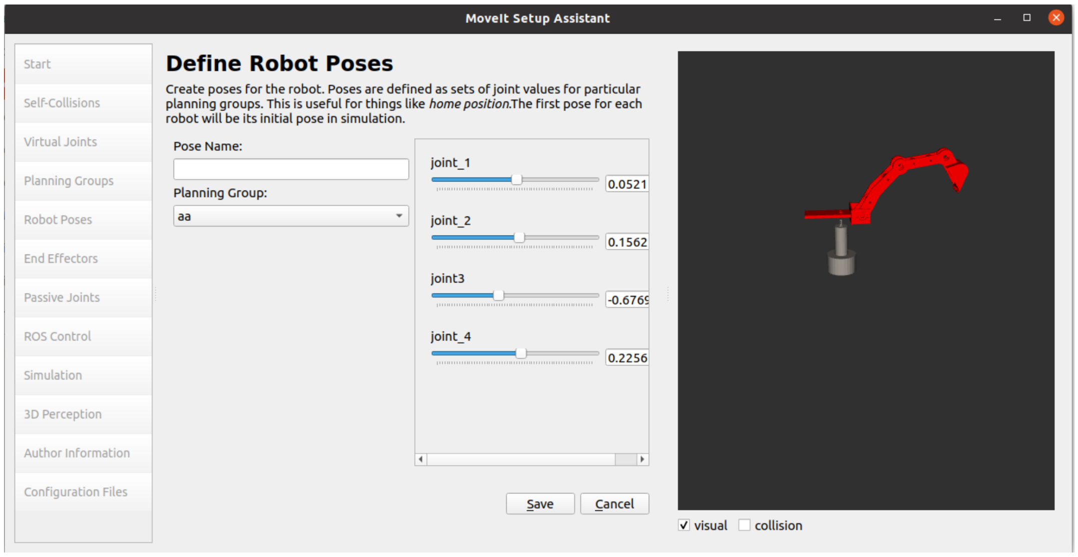


Figure 8. 14: Communication scheme with move_group node. [15]

When we want to dig and load the soil into trucks with the excavator robot arm, we need to move all the joints of the excavator's arm so that the final part of the arm, the one that has the bucket (the end-effector), can be at the proper location.

Produce the sequence of values that every joint of the manipulator must follow (in coordination with the other joints), so the end-effector moves from its current place to the desired place. This task is called motion planning. The result of motion planning is the sequence of movements that all the joints of the arm have to perform in order to move from the current location to the desired one, see Figure 8. 15.


MoveIt Setup Assistant

Define Robot Poses

Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like *home position.*The first pose for each robot will be its initial pose in simulation.

Pose Name:

Planning Group: aa

joint_1 0.0521

joint_2 0.1562

joint3 -0.676

joint_4 0.2256

☑ visual ☐ collision

Figure 8. 15: Planning a specific motion in MoveIt

Figure 8. 16 describes how the MoveIt nodes and topics communicates to each other’s,

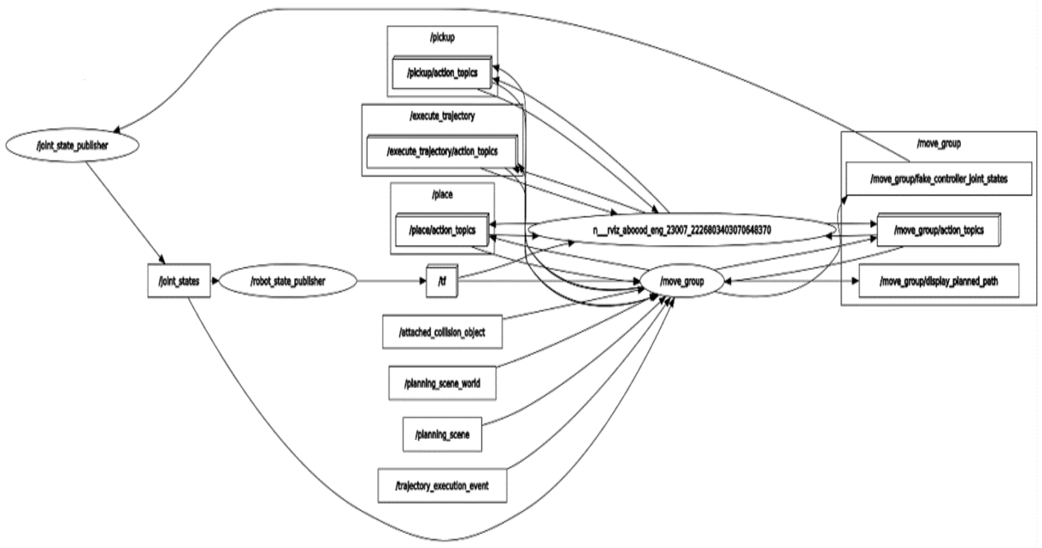

/pickup
/pickup/action_topics
/execute_trajectory
/execute_trajectory/action_topics
/place
/place/action_topics
/joint_state_publisher
/joint_states
/robot_state_publisher
/tf
n__rviz_aboood_eng_23007_2226803403070648370
/move_group
/attached_collision_object
/planning_scene_world
/planning_scene
/trajectory_execution_event
/move_group
/move_group/fake_controller_joint_states
/move_group/action_topics
/move_group/display_planned_path

Figure 8. 16: Nodes and topics of MoveIt

- RViz tool:

To show ROS messages in 3D RViz the 3D visualization tool of ROS provides a view of your excavator arm robot model, and replay captured data. allowing us to visually verify data. The following Figure 8. 17 describes how planning the manipulator for first three joints,

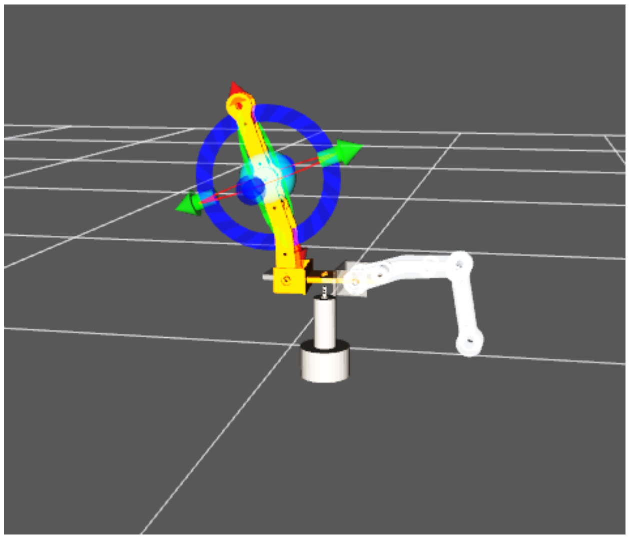


Figure 8. 17: Planning the manipulator by moveIt with RViz

A manipulator simulation can be developed consists of setting up a virtual environment and adding to this environment a virtual representation of the robot to be trained or tested (control page, avoid obstacles, and collision avoidance, etc ...).

8.3.3 Programming control algorithms using stand-alone code

Stand-alone microprocessors can provide a high level of control over simple integrated circuits, motors, and actuators. Once programmed, they can repeatedly perform the same task with precision and accuracy, making them an integral part of mechatronic engineering design. It is an alternative to the previous software platforms that introduce many advantages, such as direct hardware accessibility via a set of Python or C++ codes, management of RPi resources and running the desired task independently on a single or multi-CPU cores provides more stable performance with approximately no system interrupts occur.

It was a great experiment to learn more about running a Raspberry Pi "headless" (i.e., connecting just power and a LAN cable, then you can use SSH into your RPi without needing a monitor), auto-launching python modules on bootup, file management, file parsing, general GPIO control (as PWM), using the built-in SPI hardware on the Raspberry Pi, and even some logic level conversion.

The main goal of implementing our control algorithm with a stand-alone code is to ensure executing control and tracking tasks in real application. Python language is used to program our controller, we get benefited of using Object-Oriented-Programming (OOP) as functions, constructors and classes to make the code more usable, readable and well organized. Despite the code complexity of organizing things, the process has been succussed and control performance is accepted.

8.4 Motion control experiments and results

8.4.1 Introduction

In this section, we will discuss the actual results of PD with Gravity compensator applied to the robot. The centralized controllers provide a detailed knowledge of manipulator dynamic model, and cancellate the nonlinearities by generating compensating torques which provide a model-based nonlinear term to enhance trajectory tracking performance. Initially, PD with Gravity control is implemented through a stand-alone code using Python language on RPi. A set of functions were programmed, such as inverse kinematics, trajectory generation in both joint and operational spaces, hardware integration and recording the joints response, computing torques and tracking errors. Figure 8. 18 shows a flow chart implements the control algorithm in general.

Practically, the use of PD with Gravity compensator is suitable for our application. The gravity compensation provides a previous knowledge of the gravity terms, while the nonlinear coupling effects are reduced by default because of the existence of high motor's gear ratio at each joint.

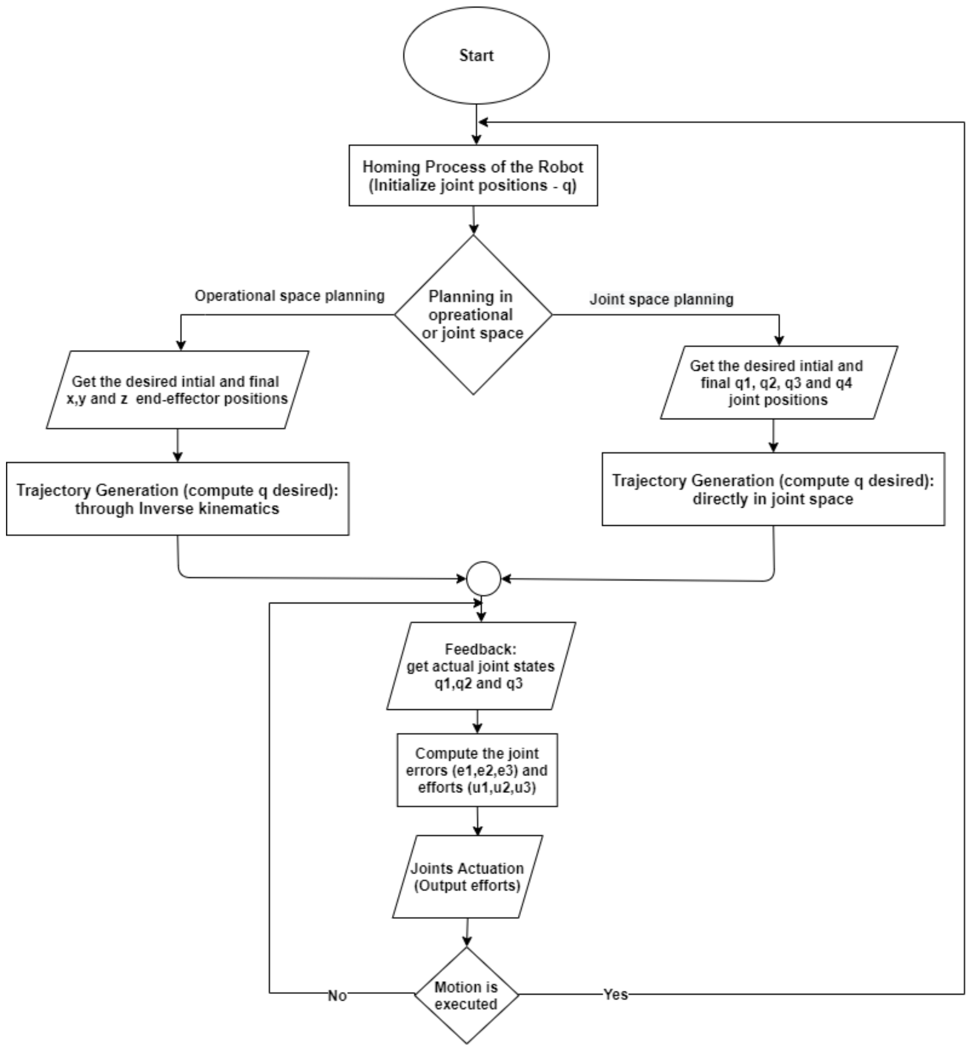

Start
Homing Process of the Robot (Initialize joint positions - q)
Planning in opreational or joint space
Operational space planning
Joint space planning
Get the desired intial and final x,y and z end-effector positions
Get the desired intial and final q1, q2, q3 and q4 joint positions
Trajectory Generation (compute q desired): through Inverse kinematics
Trajectory Generation (compute q desired): directly in joint space
Feedback: get actual joint states q1,q2 and q3
Compute the joint errors (e1,e2,e3) and efforts (u1,u2,u3)
Joints Actuation (Output efforts)
Motion is executed
No
Yes

Figure 8. 18: Flow chart of control algorithm

8.4.2 PD with gravity compensator

Design of the controller

The control process was run with the previous designed proportional and derivative gains:

$$k_p = diag[150\ 25\ 25], \quad k_d = diag[\ 0.1\ 0.1\ 0.1]$$

Motion in operational space

User inputs two desired positions for the end-effector, then the motion trajectories will be generated in operational space. Using Inverse kinematics function to translate the desired motion to the joint space, see Figure 8. 19. Joint trajectories are supplied to the control process as references, errors and efforts are calculated. The initial and final point are similar to the used in Chapter 7,

$$p_i = [\,0.2511 \quad -0.237 \quad 0.1661\,]^T \mathrm{m}$$

$$p_f = [0.1 \quad 0.2704 \quad 0.2696]^T \mathrm{m}$$

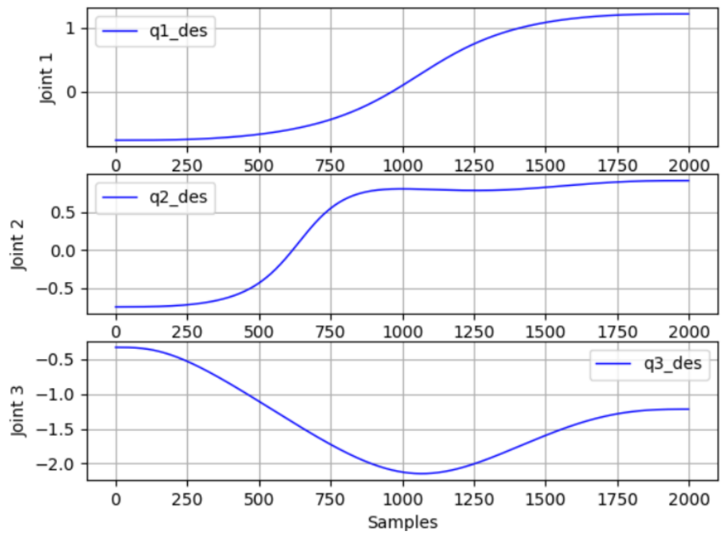


Figure 8. 19: generated joint trajectories via inverse kinematics

The motion is executed with 2000 sample of a 1 ms sample time, resulting motion time of 2s.

- Real results:

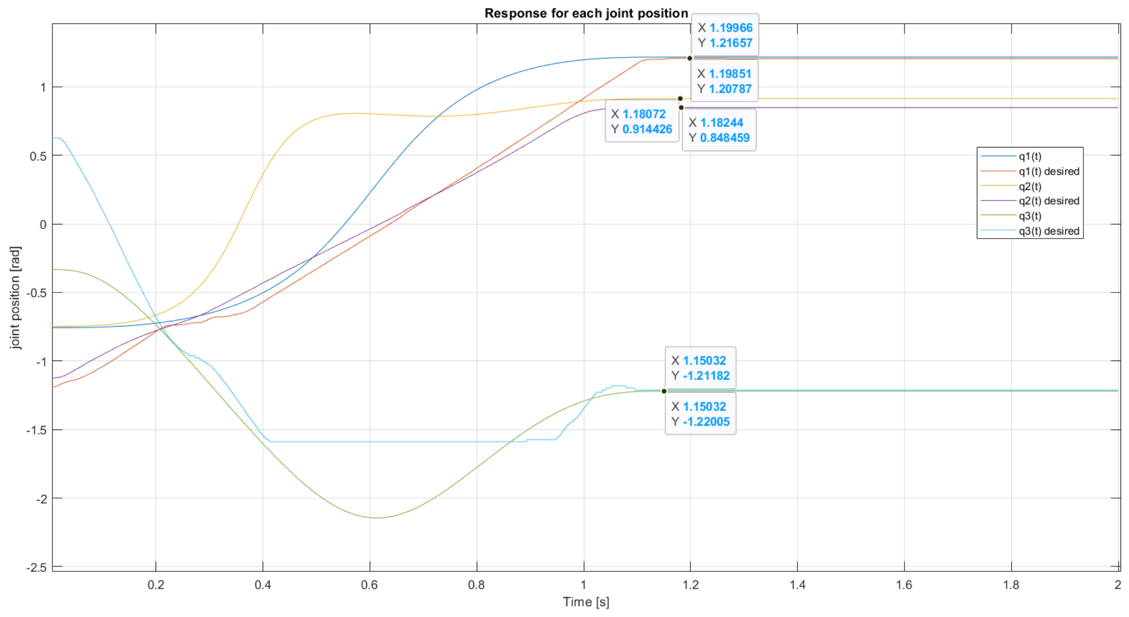


Figure 8. 20: Joint states of PD with Gravity motion control

Figure 8. 20 reveal that each joint angle approximately reaches its final position with a steady-state error of -2.57^o for the first joint, -0.6^o for the second one and -0.8^o for the third joint. These errors are expected due to each encoder resolution and the gearbox backlash that has been discussed before (section 8.2.2). Figure 8. 21 shows end-effector position error, it is about $6mm$ in x and y, $1.66cm$ in z axes which has the largest error due to many mechanical effects such as, link deflection, gearbox backlash and encoder resolutions for the second and third links.

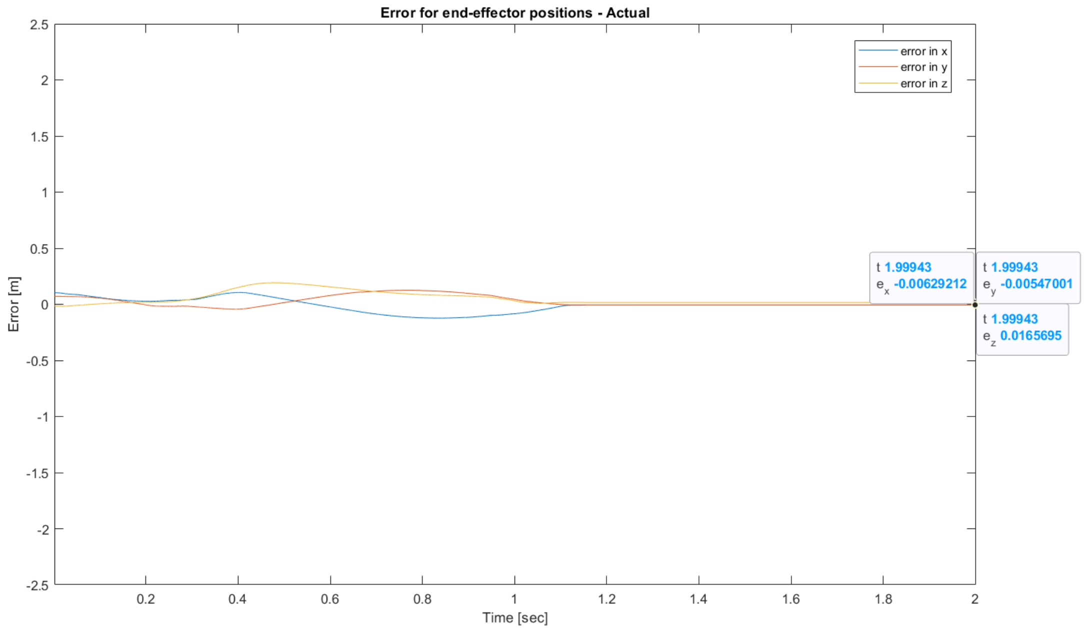


Figure 8. 21: Error evolution for the end-effector position

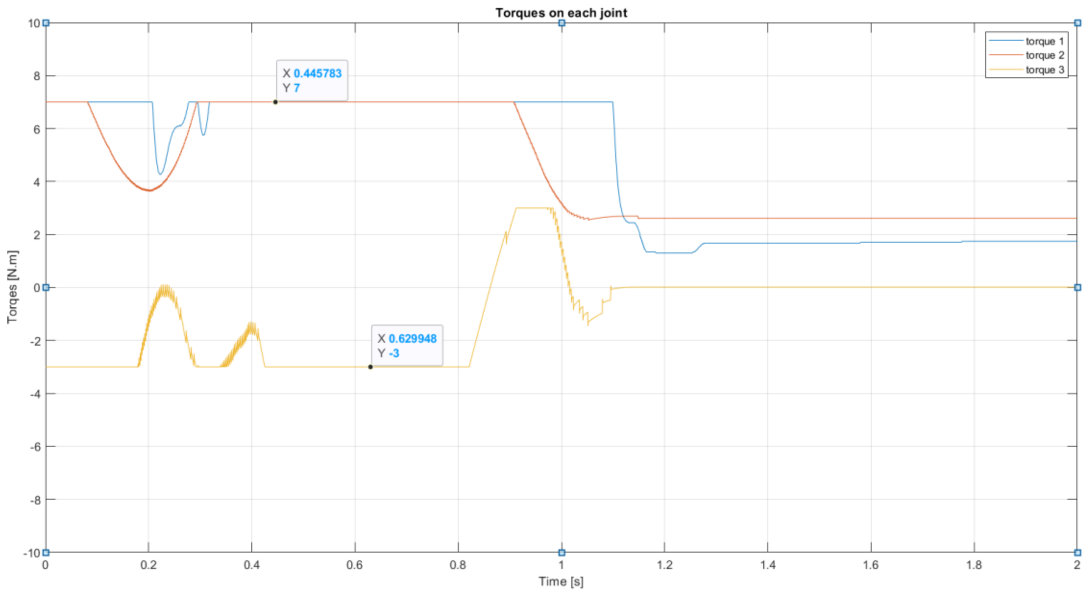


Figure 8. 22: Resulted control efforts of PD with Gravity control

Figure 8. 22 shows that joint torques are smooth, and the maximum required torque is $7N.m$ for joint 1 and 2, and $3N.m$ for joint 3.

Executing multi-motions directly in joint space

User can assign many tasks in joint space, entering the desired initial and final joint positions for each task. Trajectories in joint space are directly calculated and motion executed. Same gains are used for the control process.

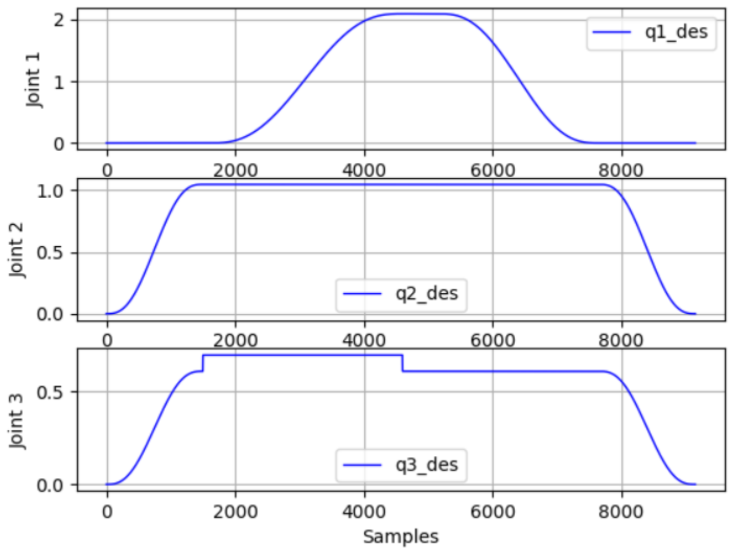


Figure 8. 23: Generated joint trajectories in joint space for multi-tasks

The motion is executed with 9000 sample of a 2 ms sample time, resulting motion time of $18s$.

- Real results:

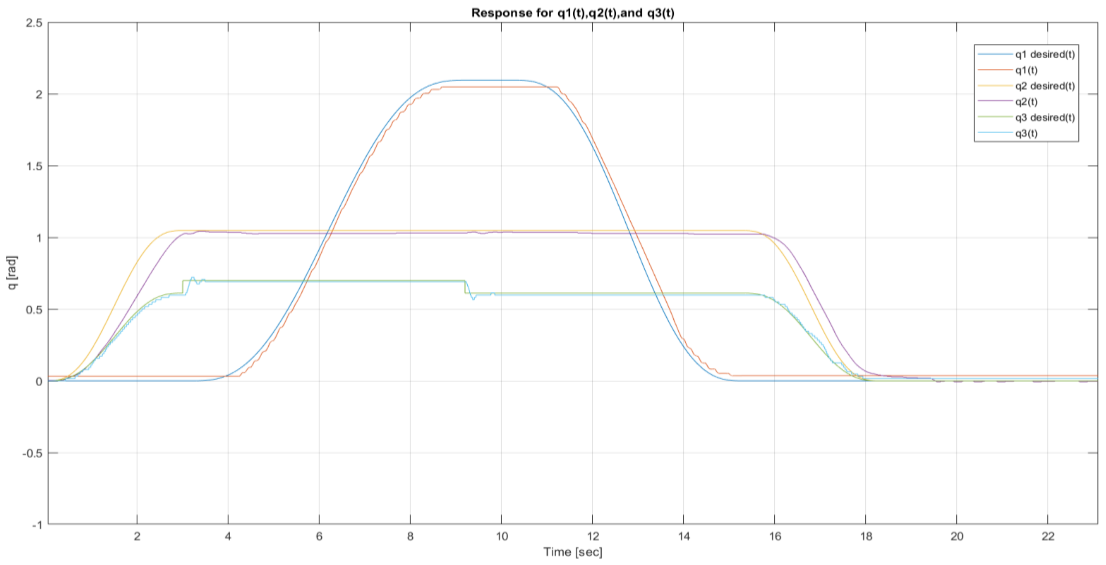


Figure 8. 24: Responses for executing many tasks in joint space, PD with Gravity control

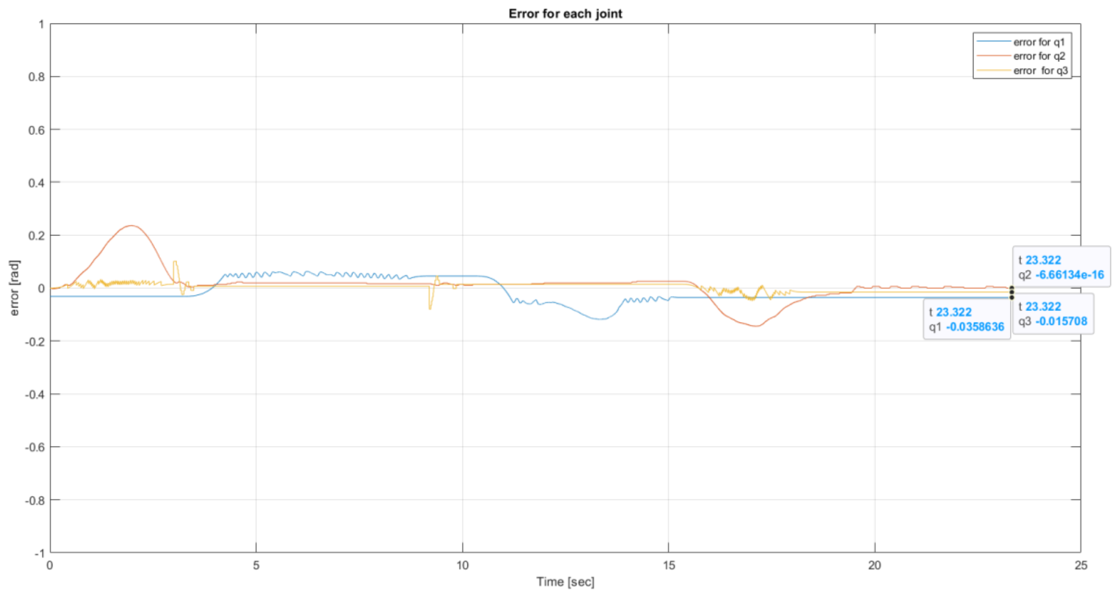


Figure 8. 25: Error evolution for each joint position

The response shows a powerful tracking performance for each joint with zero steady-state error for joint 2, 0.9^o for joint 3, and 2.055^o for joint 1, see **Error! Reference source not found.** and Figure 8. 25 which show the resulted joint positions error.

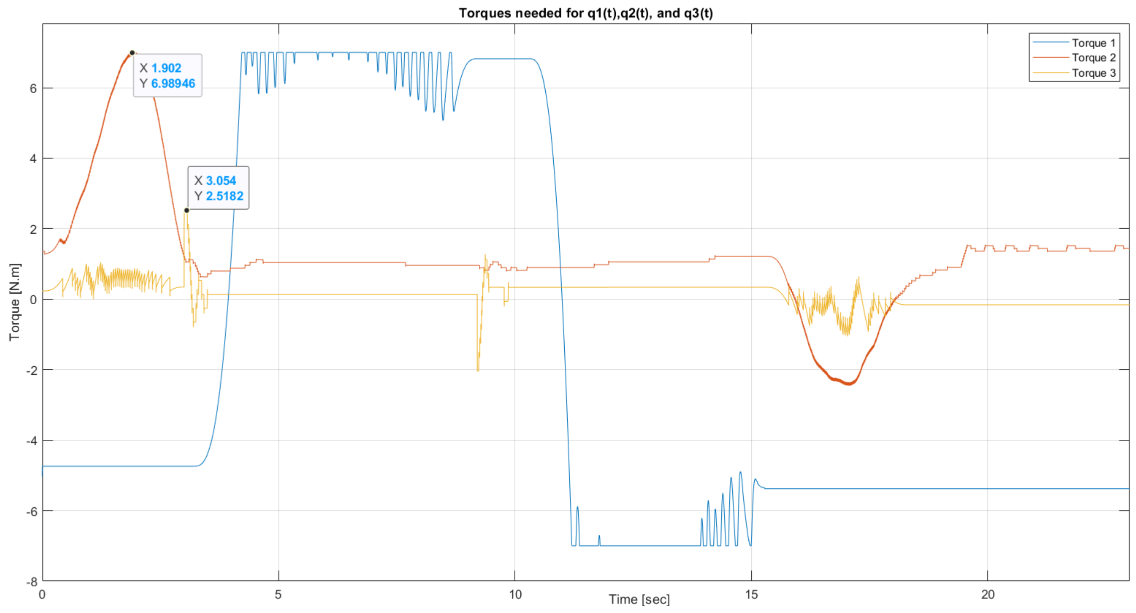


Figure 8. 26: Required torques for executing many motions

Joint torques are continuous and have maximum values of $7N.m$ for joints 1 and 2, $2.52N.m$ for the third one, see Figure 8. 26.

8.5 Conclusion

We conclude that method of PD with Gravity control has improved the tracking performance and reduces the effects of gearbox backlash and inaccuracies of encoder measurements. In contrast, the requirement of $\pm 3mm$ end-effector resolution was not achieved perfectly. The consumed torques are saturated with $7N.m$ for the first and second motors, and $3N.m$ for the third one; provides acceptable response for the desired specifications. The controller provides good disturbance rejection (e.g., effects of external forces with hand) and robustness to parameter changes (e.g., excavated soil).

The PD Control with gravity compensation has been implemented to show the robustness of this control scheme in our robot application. The control algorithm is developed using the low-cost Raspberry Pi board where the stand-alone code is deployed. The novelty of this approach is the development of new functions with the stand-alone code to make the PD control with gravity compensation in low-cost systems.

As a consequence, with the discussed results we conclude that PD with Gravity compensator covers our application interests. It was firstly programmed in purpose of testing robot motion and the system integration because of its easy programming and turning, however this control method satisfies the basic robot motion of excavation, travelling and dumping with smooth responses and reasonable time durations. Inverse dynamics control laws can be implemented with the pre-programmed stand-alone code for further development.

Chapter nine: Conclusion and Future Work

9.1 Conclusion

The Excavator robotic system has been prototyped. The mechanical structure (frames, and connectors) is set up first and the actuation system (DC motors, servomotor) is then attached to the mechanical structure, after that, other electromechanical components are attached. With all the components integrated, the size of the excavator is 20cm (width) × 65cm (length) × 10cm (height), and the total weight is 2.18 kg. The excavator has been tested to work properly, and it achieves a complete excavation process with a traveling speed of about $40\ RPM$.

The prototyping of the excavator robotic system involves the machining of mechanical frames, selection of proper electromechanical components, integration of all the parts together, as well as testing and debugging of the implemented robotic system. To save time and reduce the cost, most of the materials and components of the excavator are directly bought from local or online stores.

During the implementation, some problems in the design phase have exposed. Therefore, the design is dynamically adjusted based on the problems found in the prototyping process. Some critical thinking and problem-solving skills are necessary for the prototyping of the excavator.

The mechanical frames of the excavator offer the platform for all the other electromechanical components, the frames should be stable and maintain proper balance during the excavation, they also should be lightweight considering the strict weight limit. As a student project, the cost is also an important consideration in prototyping. The design and implementation of the excavator with all the functional modules are discussed.

The driving system activates the actuation electromechanical components of the excavator to perform the desired tasks. The success of the excavator robot not only depends on the design but also the proper control of the excavator. In the prototyping of the excavator robotic system, remote control can be used by the operator as a teleoperation technique. Alternatively, a predefined motion can be assigned to the system controller to be executed directly. Raspberry Pi 4 OS collects the information about the excavator states and the surrounding environment and sends it back to the operator on the RPi screen, this is urgent for the operator to make correct decisions and control the excavator to perform the required motion.

Once the excavator travels to the proper spot in the excavation area, it switches between traveling to excavating tasks, once the excavator collects soil in its dumpster in the excavating task, it stops excavation and switches to the traveling task moving towards the soil collector, the excavator then switches to the dumping mode to dump the excavated soil into the collector. Finally, our semi-autonomous excavator can perform the tasks of excavating soil, collecting it in the excavator's dumpster, and depositing it into the assigned collector box.

9.2 Future work

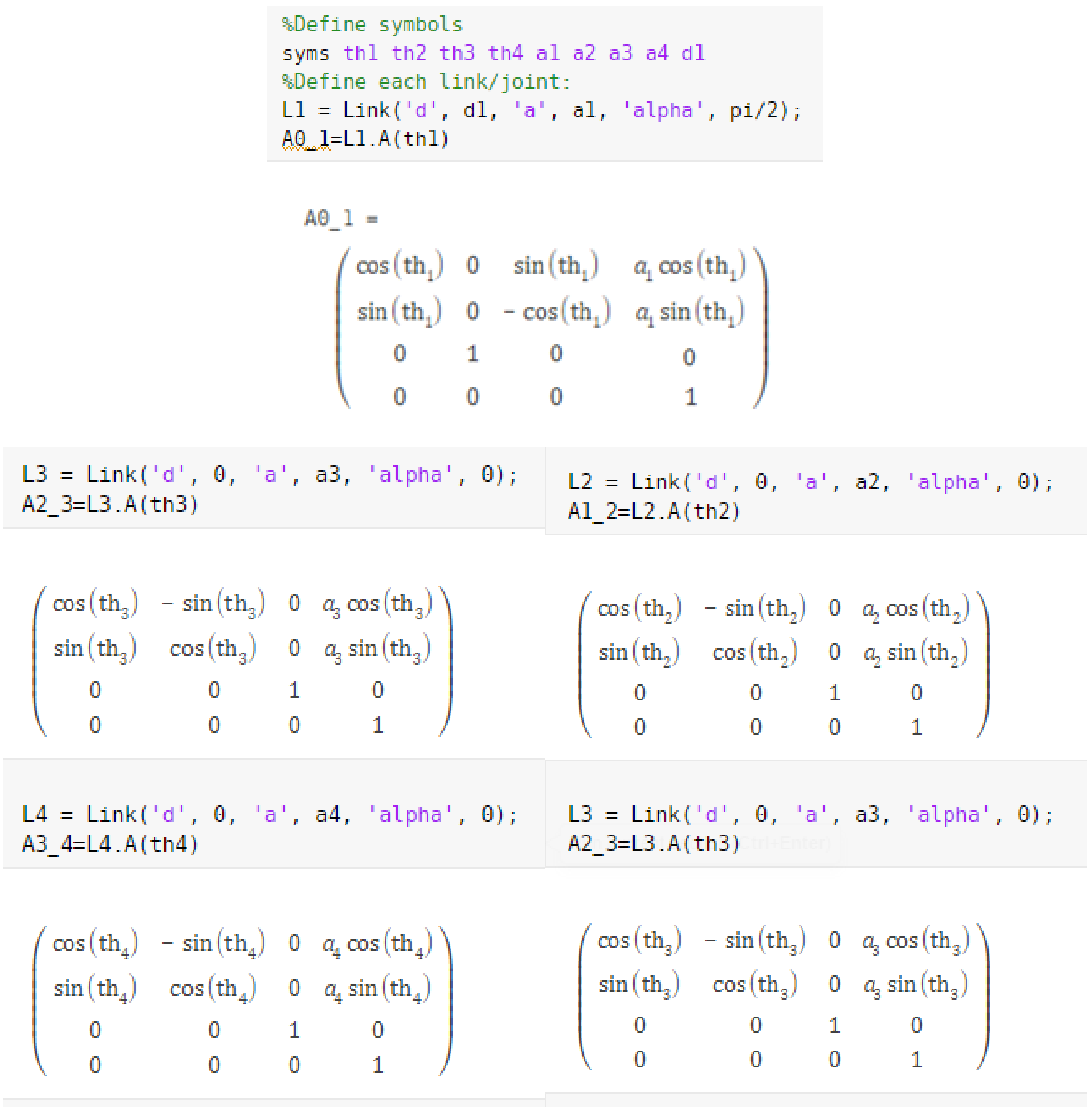
One important area of research in robotics is to enable the robot to autonomously its working environment. The robot requires several capabilities to be autonomous and be able to operate in an intelligent behavior. For a robot to be fully autonomous, it should have the capability of gaining information through vision and to imitate tasks without human assistance. These capabilities can be subdivided to many categories: Robot sensing, thinking and imitating, motion planning and control.

The manipulator autonomously learns an instructional a mapping function that enables a robot to select an action given its current state. Learning tasks by imitation means to learn the robot doing a specific dynamic movement for a desired trajectory that will be generated by itself through vision or given information about the surrounding environment, this satisfies by imitate an expert operator for a set of previous tasks, then the robot will gain experience to instruct novice operators or execute tasks autonomously.

Appendices

A. MATLAB and SIMULINK

A.1 Forward kinematics derivation - verification using robotics toolbox:



```
%Define symbols
syms th1 th2 th3 th4 a1 a2 a3 a4 d1
%Define each link/joint:
L1 = Link('d', d1, 'a', a1, 'alpha', pi/2);
A0_1=L1.A(th1)
```

```
A0_1 =
```

$$\begin{pmatrix} \cos(\mathrm{th}_1) & 0 & \sin(\mathrm{th}_1) & a_1\cos(\mathrm{th}_1) \\ \sin(\mathrm{th}_1) & 0 & -\cos(\mathrm{th}_1) & a_1\sin(\mathrm{th}_1) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
L3 = Link('d', 0, 'a', a3, 'alpha', 0);
A2_3=L3.A(th3)
```

$$\begin{pmatrix} \cos(\mathrm{th}_3) & -\sin(\mathrm{th}_3) & 0 & a_3\cos(\mathrm{th}_3) \\ \sin(\mathrm{th}_3) & \cos(\mathrm{th}_3) & 0 & a_3\sin(\mathrm{th}_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
L2 = Link('d', 0, 'a', a2, 'alpha', 0);
A1_2=L2.A(th2)
```

$$\begin{pmatrix} \cos(\mathrm{th}_2) & -\sin(\mathrm{th}_2) & 0 & a_2\cos(\mathrm{th}_2) \\ \sin(\mathrm{th}_2) & \cos(\mathrm{th}_2) & 0 & a_2\sin(\mathrm{th}_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

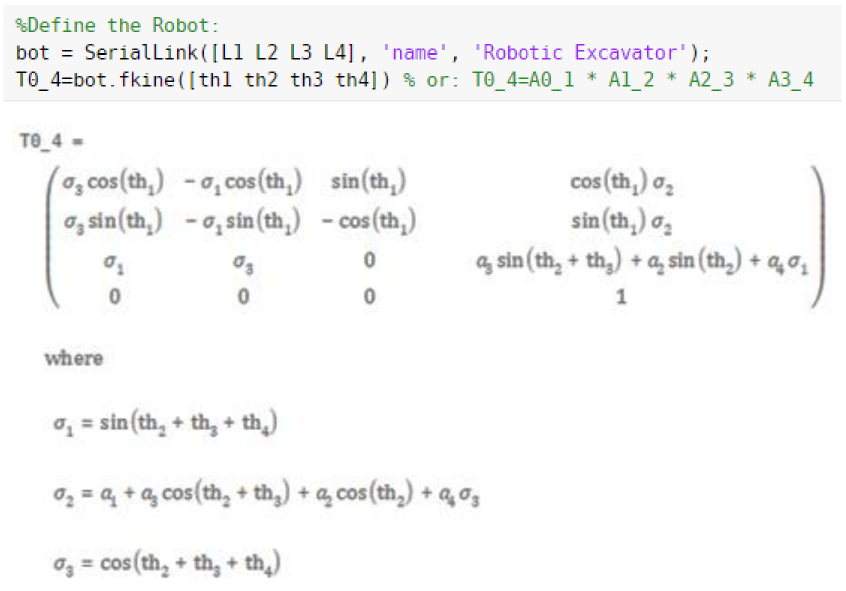
```
L4 = Link('d', 0, 'a', a4, 'alpha', 0);
A3_4=L4.A(th4)
```

$$\begin{pmatrix} \cos(\mathrm{th}_4) & -\sin(\mathrm{th}_4) & 0 & a_4\cos(\mathrm{th}_4) \\ \sin(\mathrm{th}_4) & \cos(\mathrm{th}_4) & 0 & a_4\sin(\mathrm{th}_4) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
L3 = Link('d', 0, 'a', a3, 'alpha', 0);
A2_3=L3.A(th3)
```

$$\begin{pmatrix} \cos(\mathrm{th}_3) & -\sin(\mathrm{th}_3) & 0 & a_3\cos(\mathrm{th}_3) \\ \sin(\mathrm{th}_3) & \cos(\mathrm{th}_3) & 0 & a_3\sin(\mathrm{th}_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure A. 1: MATLAB live script of the homogenous transformation matrix for each joint.



```
%Define the Robot:
bot = SerialLink([L1 L2 L3 L4], 'name', 'Robotic Excavator');
T0_4=bot.fkine([th1 th2 th3 th4]) % or: T0_4=A0_1 * A1_2 * A2_3 * A3_4
```

T0_4 =

$$\begin{pmatrix} \sigma_3\cos(\mathrm{th}_1) & -\sigma_1\cos(\mathrm{th}_1) & \sin(\mathrm{th}_1) & \cos(\mathrm{th}_1)\,\sigma_2 \\ \sigma_3\sin(\mathrm{th}_1) & -\sigma_1\sin(\mathrm{th}_1) & -\cos(\mathrm{th}_1) & \sin(\mathrm{th}_1)\,\sigma_2 \\ \sigma_1 & \sigma_3 & 0 & a_3\sin(\mathrm{th}_2+\mathrm{th}_3)+a_2\sin(\mathrm{th}_2)+a_4\sigma_1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

where

$$\sigma_1 = \sin(\mathrm{th}_2+\mathrm{th}_3+\mathrm{th}_4)$$

$$\sigma_2 = a_1 + a_3\cos(\mathrm{th}_2+\mathrm{th}_3) + a_2\cos(\mathrm{th}_2) + a_4\sigma_3$$

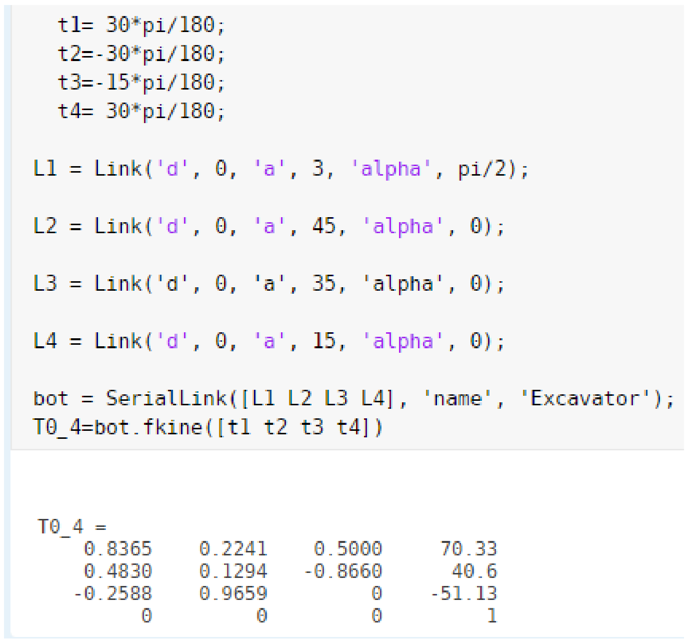
$$\sigma_3 = \cos(\mathrm{th}_2+\mathrm{th}_3+\mathrm{th}_4)$$

Figure A. 2: MATLAB live script of the forward kinematics.

A.2 Example: verification for forward kinematics with inverse kinematics:

First, compute the Forward kinematics for the desired joint-variables using (Robotics Toolbox), as Fig. A.3:

Let $\theta_1 = 30^o, \theta_2 = -30^o, \theta_3 = -15^o \; and \; \theta_4 = 30^o$

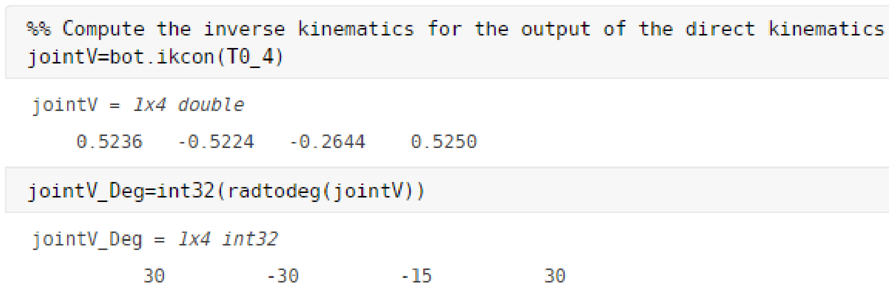


```
  t1= 30*pi/180;
  t2=-30*pi/180;
  t3=-15*pi/180;
  t4= 30*pi/180;

L1 = Link('d', 0, 'a', 3, 'alpha', pi/2);

L2 = Link('d', 0, 'a', 45, 'alpha', 0);

L3 = Link('d', 0, 'a', 35, 'alpha', 0);

L4 = Link('d', 0, 'a', 15, 'alpha', 0);

bot = SerialLink([L1 L2 L3 L4], 'name', 'Excavator');
T0_4=bot.fkine([t1 t2 t3 t4])
```

```
T0_4 =
    0.8365    0.2241    0.5000     70.33
    0.4830    0.1294   -0.8660      40.6
   -0.2588    0.9659         0    -51.13
         0         0         0         1
```

Figure A. 3: Forward kinematics for the given joint variables.

Then, verify the angles by substitute T_4^0 in the inverse kinematic function, Fig. A.4:



```
%% Compute the inverse kinematics for the output of the direct kinematics
jointV=bot.ikcon(T0_4)
```

```
jointV = 1x4 double

    0.5236   -0.5224   -0.2644    0.5250
```

```
jointV_Deg=int32(radtodeg(jointV))
```

```
jointV_Deg = 1x4 int32

          30         -30         -15          30
```

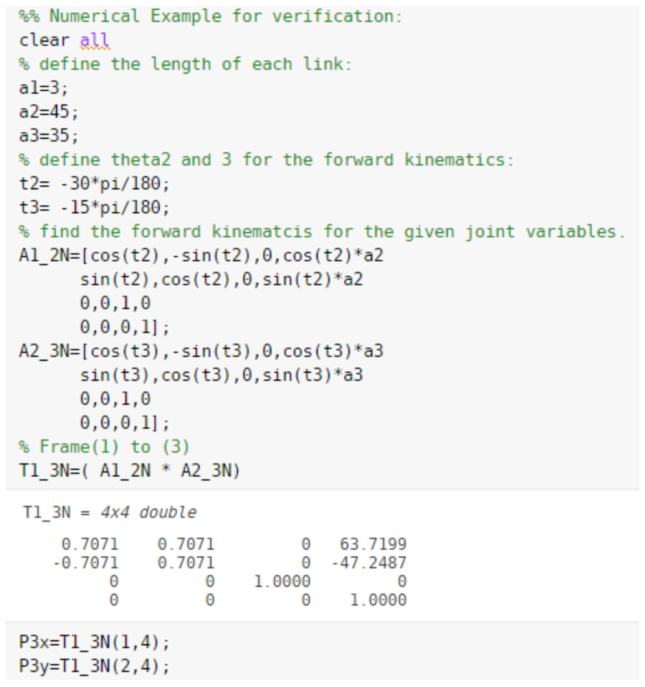
Figure A. 4: Forward kinematics verification using MATLAB toolbox

Figure (A.4) shows that forward kinematics is verified that we have obtained the same joint variables

$$\theta_1 = 30, \theta_2 = -30, \theta_3 = -15 \; and \; \theta_4 = 30$$

A.3 Using MATLAB with per-hand derived code for verification of inverse kinematics:

This code is used to verify the Algebraic solution for θ_2 *and* θ_3.



```
%% Numerical Example for verification:
clear all
% define the length of each link:
a1=3;
a2=45;
a3=35;
% define theta2 and 3 for the forward kinematics:
t2= -30*pi/180;
t3= -15*pi/180;
% find the forward kinematcis for the given joint variables.
A1_2N=[cos(t2),-sin(t2),0,cos(t2)*a2
      sin(t2),cos(t2),0,sin(t2)*a2
      0,0,1,0
      0,0,0,1];
A2_3N=[cos(t3),-sin(t3),0,cos(t3)*a3
      sin(t3),cos(t3),0,sin(t3)*a3
      0,0,1,0
      0,0,0,1];
% Frame(1) to (3)
T1_3N=( A1_2N * A2_3N)
```

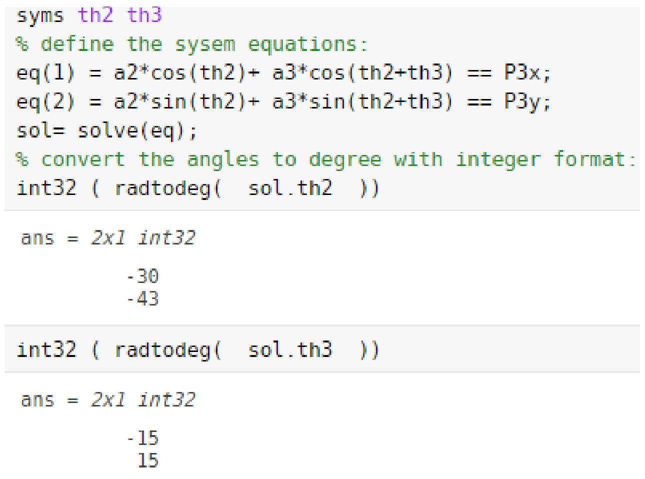
```
T1_3N = 4x4 double

    0.7071    0.7071         0   63.7199
   -0.7071    0.7071         0  -47.2487
         0         0    1.0000         0
         0         0         0    1.0000
```

```
P3x=T1_3N(1,4);
P3y=T1_3N(2,4);
```

Figure A. 5: Re-fined the forward kinematics for Frame (3) wrt Frame (1)

Write equations (4.5) and (4.6) in MATLAB, solving for θ_2 *and* θ_3 and taking P_{3x} = 63.7199 and P_{3y} = -47.2487 by the forward kinematics shown in Fig. (A.5).

As shown in Fig. (A.6); $\boldsymbol{\theta_2 = -30^o}$, $\boldsymbol{and\ \theta_3 = -15^o}$ for the specified range of joint variables in section (4.2.3), and they are same as the previous Toolbox solution.



```
syms th2 th3
% define the sysem equations:
eq(1) = a2*cos(th2)+ a3*cos(th2+th3) == P3x;
eq(2) = a2*sin(th2)+ a3*sin(th2+th3) == P3y;
sol= solve(eq);
% convert the angles to degree with integer format:
int32 ( radtodeg(  sol.th2  ))
```

```
ans = 2x1 int32

        -30
        -43
```

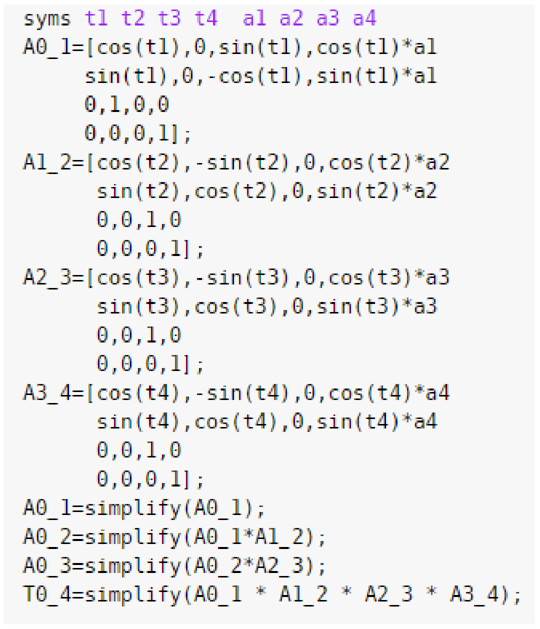
```
int32 ( radtodeg(  sol.th3  ))
```

```
ans = 2x1 int32

        -15
         15
```

Figure A. 6: The verified inverse kinematics

A.4 Verification of the jacobian analytically:

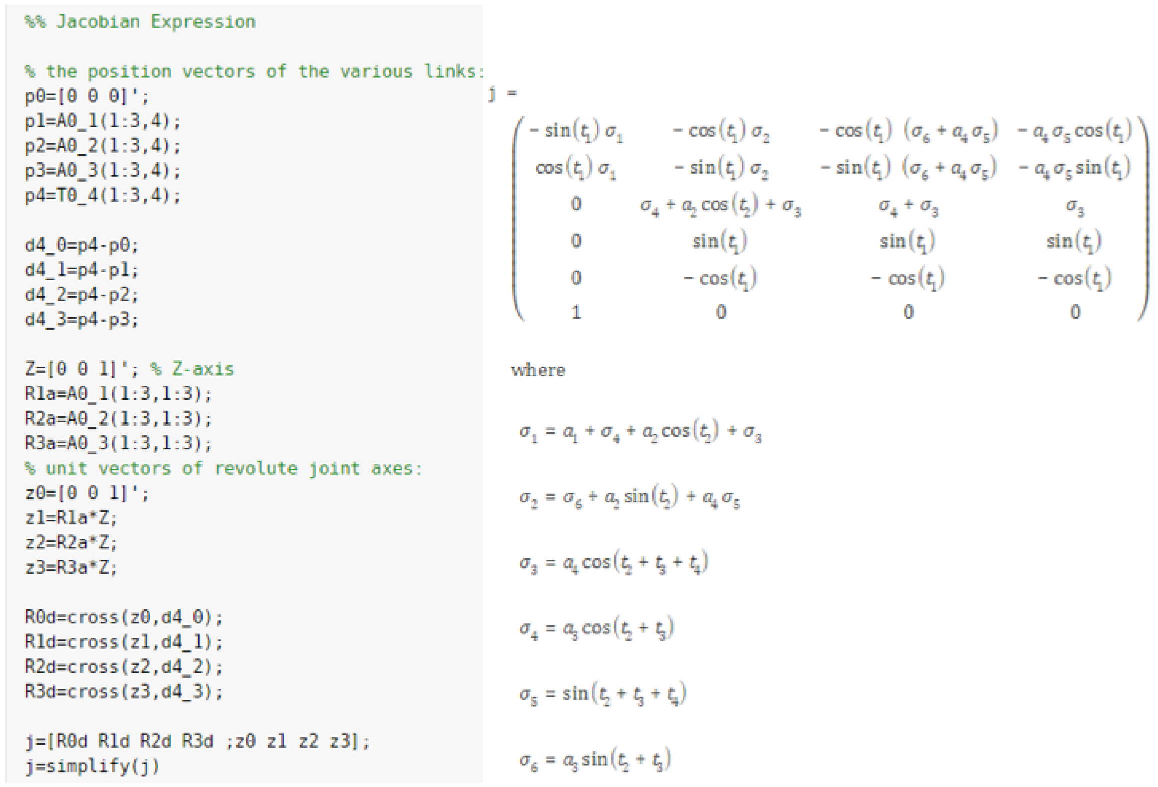
Using forward kinematics, as Fig. A.7



```
syms t1 t2 t3 t4  a1 a2 a3 a4
A0_1=[cos(t1),0,sin(t1),cos(t1)*a1
     sin(t1),0,-cos(t1),sin(t1)*a1
     0,1,0,0
     0,0,0,1];
A1_2=[cos(t2),-sin(t2),0,cos(t2)*a2
      sin(t2),cos(t2),0,sin(t2)*a2
      0,0,1,0
      0,0,0,1];
A2_3=[cos(t3),-sin(t3),0,cos(t3)*a3
      sin(t3),cos(t3),0,sin(t3)*a3
      0,0,1,0
      0,0,0,1];
A3_4=[cos(t4),-sin(t4),0,cos(t4)*a4
      sin(t4),cos(t4),0,sin(t4)*a4
      0,0,1,0
      0,0,0,1];
A0_1=simplify(A0_1);
A0_2=simplify(A0_1*A1_2);
A0_3=simplify(A0_2*A2_3);
T0_4=simplify(A0_1 * A1_2 * A2_3 * A3_4);
```

Figure A. 7: Forward kinematics using MATLAB

Finding the Jacobian same as the previous procedure in (4.16 and 4.17) using MATLAB, as described in Fig. A.8;



```
%% Jacobian Expression

% the position vectors of the various links:
p0=[0 0 0]';
p1=A0_1(1:3,4);
p2=A0_2(1:3,4);
p3=A0_3(1:3,4);
p4=T0_4(1:3,4);

d4_0=p4-p0;
d4_1=p4-p1;
d4_2=p4-p2;
d4_3=p4-p3;

Z=[0 0 1]'; % Z-axis
R1a=A0_1(1:3,1:3);
R2a=A0_2(1:3,1:3);
R3a=A0_3(1:3,1:3);
% unit vectors of revolute joint axes:
z0=[0 0 1]';
z1=R1a*Z;
z2=R2a*Z;
z3=R3a*Z;

R0d=cross(z0,d4_0);
R1d=cross(z1,d4_1);
R2d=cross(z2,d4_2);
R3d=cross(z3,d4_3);

j=[R0d R1d R2d R3d ;z0 z1 z2 z3];
j=simplify(j)
```

j =

$$\begin{pmatrix} -\sin(t_1)\,\sigma_1 & -\cos(t_1)\,\sigma_2 & -\cos(t_1)\,(\sigma_6 + a_4\sigma_5) & -a_4\sigma_5\cos(t_1) \\ \cos(t_1)\,\sigma_1 & -\sin(t_1)\,\sigma_2 & -\sin(t_1)\,(\sigma_6 + a_4\sigma_5) & -a_4\sigma_5\sin(t_1) \\ 0 & \sigma_4 + a_2\cos(t_2) + \sigma_3 & \sigma_4 + \sigma_3 & \sigma_3 \\ 0 & \sin(t_1) & \sin(t_1) & \sin(t_1) \\ 0 & -\cos(t_1) & -\cos(t_1) & -\cos(t_1) \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

where

$$\sigma_1 = a_1 + \sigma_4 + a_2\cos(t_2) + \sigma_3$$

$$\sigma_2 = \sigma_6 + a_2\sin(t_2) + a_4\sigma_5$$

$$\sigma_3 = a_4\cos(t_2 + t_3 + t_4)$$

$$\sigma_4 = a_3\cos(t_2 + t_3)$$

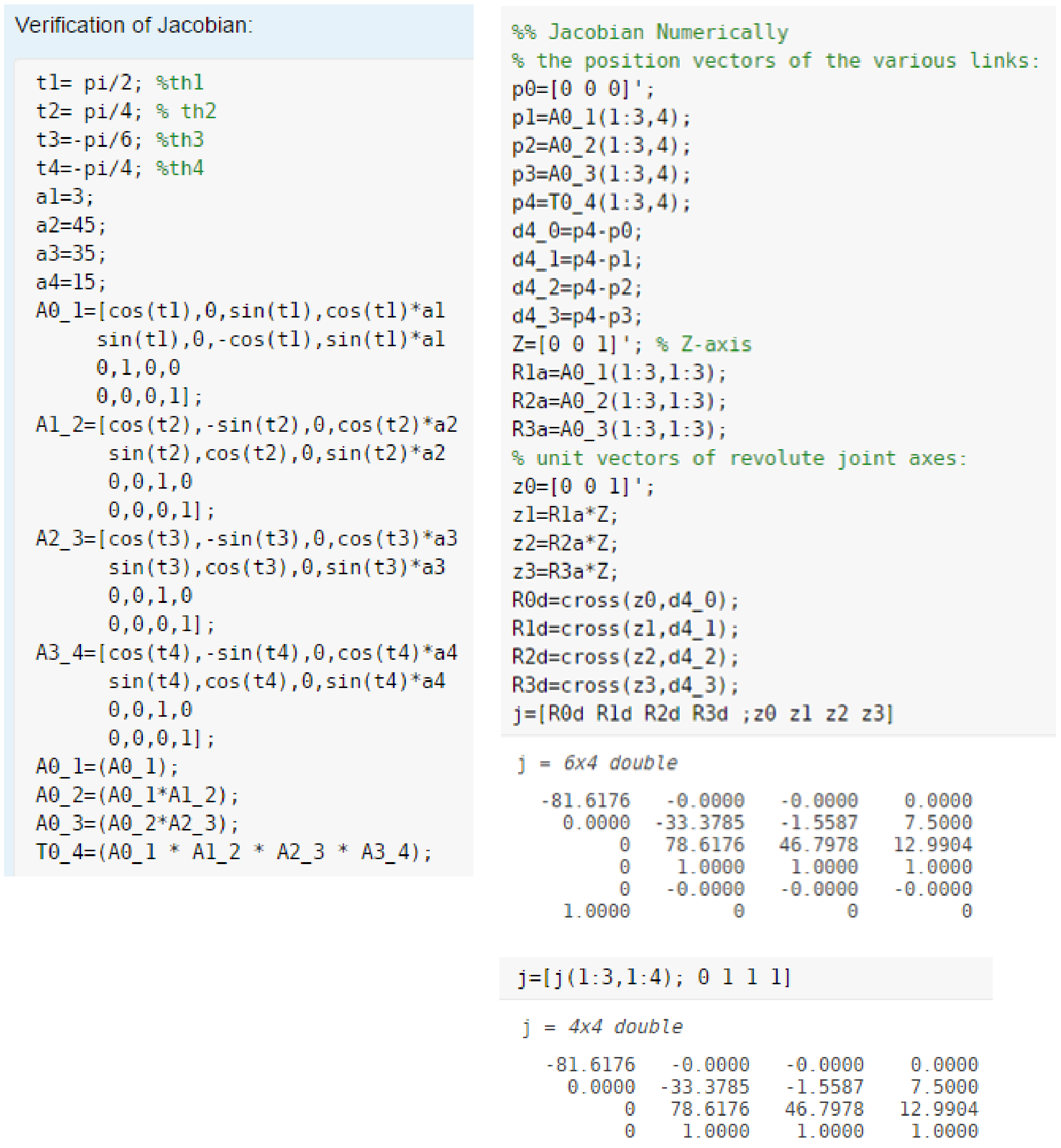
$$\sigma_5 = \sin(t_2 + t_3 + t_4)$$

$$\sigma_6 = a_3\sin(t_2 + t_3)$$

Figure A. 8*:* Jacobian using MATLAB

A.5 Using MATLAB robotics toolbox (verification numerically):

1st solution, for the previous code, substitute the joint variables and lengths of the links numerically, with $\theta_1 = 90^o, \theta_2 = 45^o, \theta_3 = -30^o, \theta_4 = -45^o, a_1 = 3,\ a_2 = 45, a_3 = 35\ and\ a_4 = 15cm$, see Fig. A.9;


Verification of Jacobian:

```
t1= pi/2; %th1
t2= pi/4; % th2
t3=-pi/6; %th3
t4=-pi/4; %th4
a1=3;
a2=45;
a3=35;
a4=15;
A0_1=[cos(t1),0,sin(t1),cos(t1)*a1
     sin(t1),0,-cos(t1),sin(t1)*a1
     0,1,0,0
     0,0,0,1];
A1_2=[cos(t2),-sin(t2),0,cos(t2)*a2
      sin(t2),cos(t2),0,sin(t2)*a2
      0,0,1,0
      0,0,0,1];
A2_3=[cos(t3),-sin(t3),0,cos(t3)*a3
      sin(t3),cos(t3),0,sin(t3)*a3
      0,0,1,0
      0,0,0,1];
A3_4=[cos(t4),-sin(t4),0,cos(t4)*a4
      sin(t4),cos(t4),0,sin(t4)*a4
      0,0,1,0
      0,0,0,1];
A0_1=(A0_1);
A0_2=(A0_1*A1_2);
A0_3=(A0_2*A2_3);
T0_4=(A0_1 * A1_2 * A2_3 * A3_4);
```

```
%% Jacobian Numerically
% the position vectors of the various links:
p0=[0 0 0]';
p1=A0_1(1:3,4);
p2=A0_2(1:3,4);
p3=A0_3(1:3,4);
p4=T0_4(1:3,4);
d4_0=p4-p0;
d4_1=p4-p1;
d4_2=p4-p2;
d4_3=p4-p3;
Z=[0 0 1]'; % Z-axis
R1a=A0_1(1:3,1:3);
R2a=A0_2(1:3,1:3);
R3a=A0_3(1:3,1:3);
% unit vectors of revolute joint axes:
z0=[0 0 1]';
z1=R1a*Z;
z2=R2a*Z;
z3=R3a*Z;
R0d=cross(z0,d4_0);
R1d=cross(z1,d4_1);
R2d=cross(z2,d4_2);
R3d=cross(z3,d4_3);
j=[R0d R1d R2d R3d ;z0 z1 z2 z3]
```

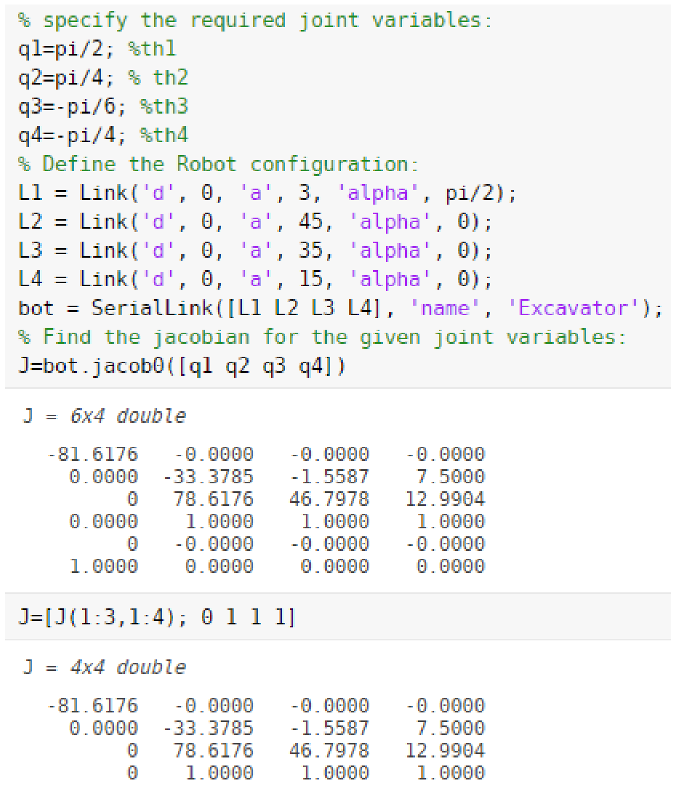
```
j = 6x4 double

  -81.6176   -0.0000   -0.0000    0.0000
    0.0000  -33.3785   -1.5587    7.5000
         0   78.6176   46.7978   12.9904
         0    1.0000    1.0000    1.0000
         0   -0.0000   -0.0000   -0.0000
    1.0000         0         0         0
```

```
j=[j(1:3,1:4); 0 1 1 1]
```

```
j = 4x4 double

  -81.6176   -0.0000   -0.0000    0.0000
    0.0000  -33.3785   -1.5587    7.5000
         0   78.6176   46.7978   12.9904
         0    1.0000    1.0000    1.0000
```

Figure A. 9: Substituting numbers instead of variables

2nd solution, verification of the calculated Jacobian matrix (j) in Figure 1.4, using Robotics Toolbox, as shown in Fig. A.10:



```
% specify the required joint variables:
q1=pi/2; %th1
q2=pi/4; % th2
q3=-pi/6; %th3
q4=-pi/4; %th4
% Define the Robot configuration:
L1 = Link('d', 0, 'a', 3, 'alpha', pi/2);
L2 = Link('d', 0, 'a', 45, 'alpha', 0);
L3 = Link('d', 0, 'a', 35, 'alpha', 0);
L4 = Link('d', 0, 'a', 15, 'alpha', 0);
bot = SerialLink([L1 L2 L3 L4], 'name', 'Excavator');
% Find the jacobian for the given joint variables:
J=bot.jacob0([q1 q2 q3 q4])
```

```
J = 6x4 double

  -81.6176   -0.0000   -0.0000   -0.0000
    0.0000  -33.3785   -1.5587    7.5000
         0   78.6176   46.7978   12.9904
    0.0000    1.0000    1.0000    1.0000
         0   -0.0000   -0.0000   -0.0000
    1.0000    0.0000    0.0000    0.0000
```

```
J=[J(1:3,1:4); 0 1 1 1]
```

```
J = 4x4 double

  -81.6176   -0.0000   -0.0000   -0.0000
    0.0000  -33.3785   -1.5587    7.5000
         0   78.6176   46.7978   12.9904
         0    1.0000    1.0000    1.0000
```

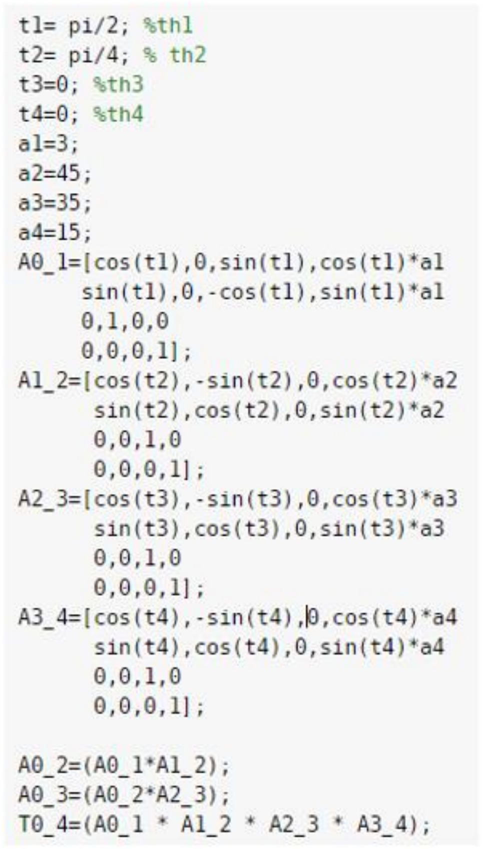
Figure A. 10: Finding Jacobian numerically using Robotics toolbox for the specified variables

For the same joint variables and lengths, the Jacobian is same for the two solutions.

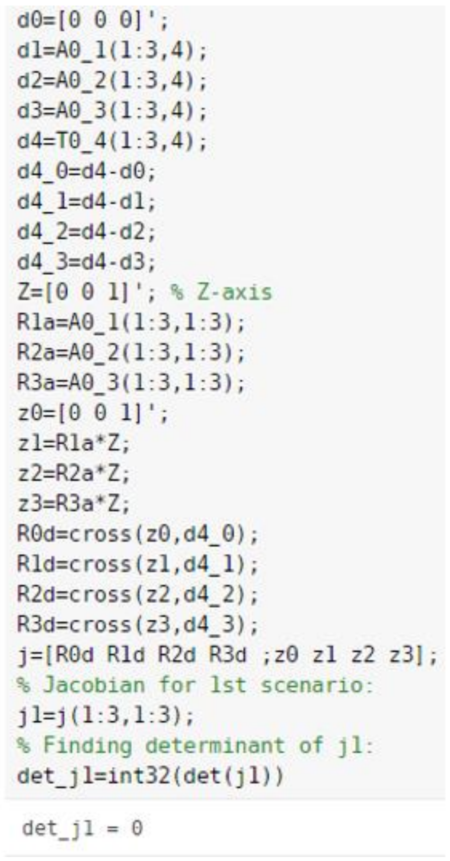
A.6 Jacobian's determinate and singularities analysis:

By Finding determinant for each of the Jacobian matrices for equations (5) and (6), for each scenario:

1st scenario, the determinant of $J_1 = 0$ at $\theta_3 = \theta_4 = 0$, this shown by Fig. A.11,



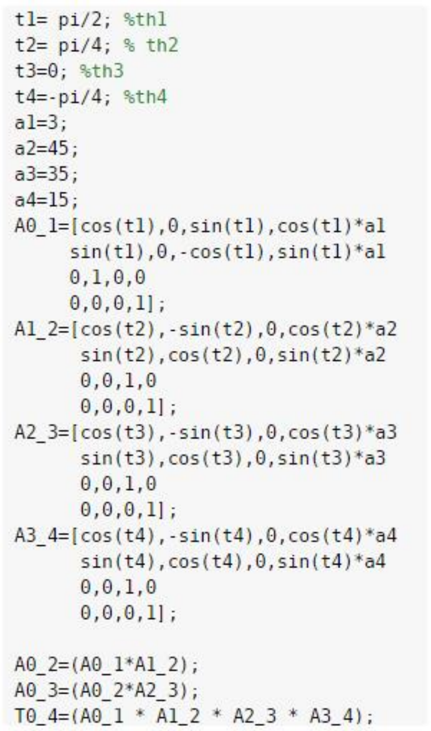
```
t1= pi/2; %th1
t2= pi/4; % th2
t3=0; %th3
t4=0; %th4
a1=3;
a2=45;
a3=35;
a4=15;
A0_1=[cos(t1),0,sin(t1),cos(t1)*a1
     sin(t1),0,-cos(t1),sin(t1)*a1
     0,1,0,0
     0,0,0,1];
A1_2=[cos(t2),-sin(t2),0,cos(t2)*a2
      sin(t2),cos(t2),0,sin(t2)*a2
      0,0,1,0
      0,0,0,1];
A2_3=[cos(t3),-sin(t3),0,cos(t3)*a3
      sin(t3),cos(t3),0,sin(t3)*a3
      0,0,1,0
      0,0,0,1];
A3_4=[cos(t4),-sin(t4),0,cos(t4)*a4
      sin(t4),cos(t4),0,sin(t4)*a4
      0,0,1,0
      0,0,0,1];

A0_2=(A0_1*A1_2);
A0_3=(A0_2*A2_3);
T0_4=(A0_1 * A1_2 * A2_3 * A3_4);
```



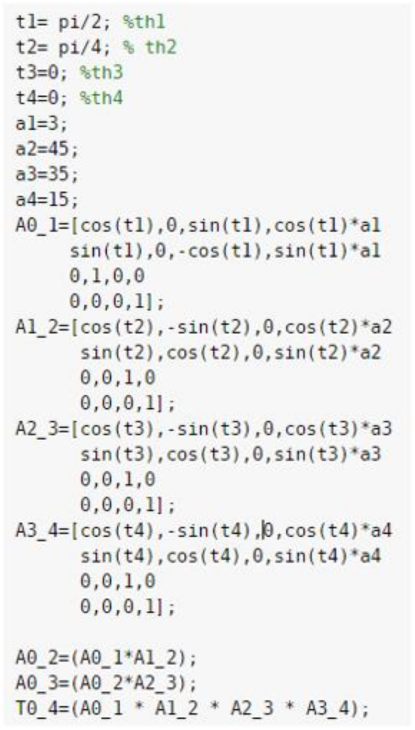
```
d0=[0 0 0]';
d1=A0_1(1:3,4);
d2=A0_2(1:3,4);
d3=A0_3(1:3,4);
d4=T0_4(1:3,4);
d4_0=d4-d0;
d4_1=d4-d1;
d4_2=d4-d2;
d4_3=d4-d3;
Z=[0 0 1]'; % Z-axis
R1a=A0_1(1:3,1:3);
R2a=A0_2(1:3,1:3);
R3a=A0_3(1:3,1:3);
z0=[0 0 1]';
z1=R1a*Z;
z2=R2a*Z;
z3=R3a*Z;
R0d=cross(z0,d4_0);
R1d=cross(z1,d4_1);
R2d=cross(z2,d4_2);
R3d=cross(z3,d4_3);
j=[R0d R1d R2d R3d ;z0 z1 z2 z3];
% Jacobian for 1st scenario:
j1=j(1:3,1:3);
% Finding determinant of j1:
det_j1=int32(det(j1))

det_j1 = 0
```

Figure A. 11: Finding Jacobian 1 using MATLAB

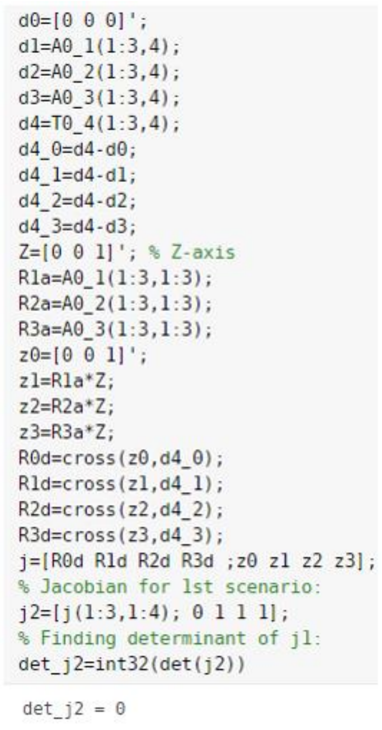
2nd scenario, the determinant of $J_2 = 0$ at $\theta_3 = 0 \; or \; \theta_3 = \theta_4 = 0$, this shown by Fig. A.12,



```
t1= pi/2; %th1
t2= pi/4; % th2
t3=0; %th3
t4=-pi/4; %th4
a1=3;
a2=45;
a3=35;
a4=15;
A0_1=[cos(t1),0,sin(t1),cos(t1)*a1
     sin(t1),0,-cos(t1),sin(t1)*a1
     0,1,0,0
     0,0,0,1];
A1_2=[cos(t2),-sin(t2),0,cos(t2)*a2
      sin(t2),cos(t2),0,sin(t2)*a2
      0,0,1,0
      0,0,0,1];
A2_3=[cos(t3),-sin(t3),0,cos(t3)*a3
      sin(t3),cos(t3),0,sin(t3)*a3
      0,0,1,0
      0,0,0,1];
A3_4=[cos(t4),-sin(t4),0,cos(t4)*a4
      sin(t4),cos(t4),0,sin(t4)*a4
      0,0,1,0
      0,0,0,1];

A0_2=(A0_1*A1_2);
A0_3=(A0_2*A2_3);
T0_4=(A0_1 * A1_2 * A2_3 * A3_4);
```



```
t1= pi/2; %th1
t2= pi/4; % th2
t3=0; %th3
t4=0; %th4
a1=3;
a2=45;
a3=35;
a4=15;
A0_1=[cos(t1),0,sin(t1),cos(t1)*a1
     sin(t1),0,-cos(t1),sin(t1)*a1
     0,1,0,0
     0,0,0,1];
A1_2=[cos(t2),-sin(t2),0,cos(t2)*a2
      sin(t2),cos(t2),0,sin(t2)*a2
      0,0,1,0
      0,0,0,1];
A2_3=[cos(t3),-sin(t3),0,cos(t3)*a3
      sin(t3),cos(t3),0,sin(t3)*a3
      0,0,1,0
      0,0,0,1];
A3_4=[cos(t4),-sin(t4),0,cos(t4)*a4
      sin(t4),cos(t4),0,sin(t4)*a4
      0,0,1,0
      0,0,0,1];

A0_2=(A0_1*A1_2);
A0_3=(A0_2*A2_3);
T0_4=(A0_1 * A1_2 * A2_3 * A3_4);
```



```
d0=[0 0 0]';
d1=A0_1(1:3,4);
d2=A0_2(1:3,4);
d3=A0_3(1:3,4);
d4=T0_4(1:3,4);
d4_0=d4-d0;
d4_1=d4-d1;
d4_2=d4-d2;
d4_3=d4-d3;
Z=[0 0 1]'; % Z-axis
R1a=A0_1(1:3,1:3);
R2a=A0_2(1:3,1:3);
R3a=A0_3(1:3,1:3);
z0=[0 0 1]';
z1=R1a*Z;
z2=R2a*Z;
z3=R3a*Z;
R0d=cross(z0,d4_0);
R1d=cross(z1,d4_1);
R2d=cross(z2,d4_2);
R3d=cross(z3,d4_3);
j=[R0d R1d R2d R3d ;z0 z1 z2 z3];
% Jacobian for 1st scenario:
j2=[j(1:3,1:4); 0 1 1 1];
% Finding determinant of j1:
det_j2=int32(det(j2))

det_j2 = 0
```

Figure A. 12: Finding Jacobian 2 using MATLAB

References

[1] A. Stentz, J. Bares, S. Singh and P. Rowe, "A robotic excavator for autonomous truck loading," Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190), Victoria, BC, Canada, 1998, pp. 1885-1893 vol.3, doi: 10.1109/IROS.1998.724871.

[2] Maske, Harshal, et al. "Learning task-based instructional policy for excavator-like robots." 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018.

[3] Cao, Yuanguo, and Youbai Xie. "Dynamic modeling of the front structure of an excavator." Nonlinear Dynamics 91.1 (2018): 233-247.

[4] Mahmoud Khaled, Design Implementation And Digital Control of a Robotics Arm. MINIA UNIVERSITY, 2009.

[5] Siciliano, Bruno, et al. Robotics: modelling, planning and control. Springer Science & Business Media, 2010.

[6] A. J. Koivo, M. Thoma, E. Kocaoglan, and J. AndradeCetto, "Modelling and control of excavator dynamics during digging operation," J. Aerosp. Eng., vol. 9, no. 1, pp. 10-18, 1996.

[7] Solazzi, Luigi. "Design of aluminium boom and arm for an excavator." Journal of Terramechanics 47.4 (2010): 201-207.

[8] Peirson, Brad. "Comparison of specific properties of engineering materials." Laboratory Module 5 (2005).

[9] Budynas, Richard Gordon, and J. Keith Nisbett. Shigley's mechanical engineering design. Vol. 8. New York: McGraw-Hill, 2008.

[10] Modern plastics, Delrin, https://modernplastics.com/products/delrin

[11] SOLIDWORKS, SOLIDWORKS Simulation, https://www.solidworks.com/product/solidworks-simulation

[12] Shetty, D. and Kolk, R.A., 2010. Mechatronics system design. Cengage Learning.

[13] MBTechworks, Raspberry Pi I2C /SPI /UART Communications, https://www.mbtechworks.com/hardware/raspberry-pi-UART-SPI-I2C.html

[14] Solomotorcontrollers, DATASHEET – SOLO UNO, https://www.solomotorcontrollers.com/datasheet-solo-uno/

[15] NORMAN S. NISE, Control System Engineering, sixth edition, Wiley, 2010.

[16] Pyo, Y., Cho, H., Jung, R. and Lim, T., 2017. ROS. Robot Programming, pp.1-308.

[17] Raimarius Delgado, Bum-Jae You, Byoung Wook Choi, Real-time control architecture based on Xenomai using ROS packages for a service robot, Journal of Systems and Software, Volume 151, 2019, Pages 8-19, ISSN 0164-1212