ROBOTICS CONTROLS FOR EXCAVATOR

Lappeenranta–Lahti University of Technology LUT

Master’s Programme in Mechanical Engineering, Master's thesis

2021

Ruslan Malygin

Examiners: Professor Aki Mikkola

Ph. D. candidate Ilya Kurinov

ABSTRACT

Lappeenranta–Lahti University of Technology LUT

LUT School of Energy Systems

Mechanical Engineering

Ruslan Malygin

ROBOTICS CONTROLS FOR EXCAVATOR

Master’s thesis

2021

83 pages, 33 figures, 13 tables and 3 appendices

Examiners: Professor Aki Mikkola

Ph. D. candidate Ilya Kurinov

Keywords: Inverse Kinematic, excavator, path-planning algorithms, Cyclic coordinate descent, Triangulation Method, Method FABRIK, PID controller.

The main direction of development of such earthmoving machines as excavators is aimed at increasing its efficiency. Efficiency is increased by improving automatic control systems, accuracy, and speed. Modern research is aimed at studying robotic excavators. In this master's thesis, the main objective is to choose the most appropriate method of inverse kinematics applied on the excavator model. Also, the implementation and tuning of PID controllers to increase the efficiency of the application of IK.

In the theoretical part of the master's thesis a literature review was carried out, which allows to immerse and deepen in the studied topic. The fundamentals of multibody systems in space, three methods of inverse kinematics (CCD, Triangulation, FABRIK), and implementation of PID controller were considered.

In the practical part of the master's thesis all three considered methods of Inverse Kinematics were implemented on the excavator model. Such software as Mevea and Simulink (MATLAB) were used for the research. The PID controllers were implemented in the actuator controls. Experimentally, the most suitable IK method was identified according to such parameters as: bucket-to-target path, accuracy, and time to reach the target. The most suitable method makes sense to apply in the research and design of robotic excavators as well as digital twins.

ACKNOWLEDGMENTS

This Master's thesis research was conducted at Energy School Systems, Mechanical Engineering department at the Lappeenranta-Lahti University of Technology (LUT). I wish to express my gratitude to Professor Aki Mikkola whose outstanding expertise and knowledge in machine design assisted my research endeavour. It was an amazing experience learning from my colleagues along with acquiring the necessary knowledge and expertise within this field. Furthermore, this thesis would not be possible without the advice and supervision of the Ph.D. candidate Ilya Kurinov. His persistence and willingness to help me during my research journey allowed me to overcome new challenges and directed my efforts towards novel methods. I am thankful that I had the chance to work closely with such an expert in the automation of machinery.

I would also like to thank my family and friends for encouraging me to look for the bright side. At most I want to express my sincere gratitude to my close friends Maria and Jad who literally acted as my thesis motivation coaches. People can do anything when being supported.

In conclusion, I would like to express my gratitude to the Lappeenranta-Lahti University of Technology (LUT) for the opportunity to acquire scientific knowledge and professional competencies. I am honoured to go through this stage of my life and would like to use that knowledge gained for my future career.

Ruslan Malygin

Lappeenranta, 04.10.2021

SYMBOLS AND ABBREVIATIONS

Roman characters:

$\mathbf{A}$	the rotation matrix
a	the first side of the triangle of the Triangulation method
b	the second side of the triangle of the Triangulation method
$\mathbf{C}$	the constraint equation
$\mathbf{C_q}$	the Jacobian Matrix
c	the third side of the triangle of the Triangulation method
d_i	the distance between the previous position and the considered one
$\mathbf{F}$	the applied forces on the body
$\mathbf{I}$	the identity matrix
K_p	the proportional coefficient of PID controller
K_i	the integral coefficient of PID controller
K_d	the derivative coefficient of PID controller
l_i	the body length
$\mathbf{M}$	the mass matrix of the body
n	the number of generalized coordinates
n_c	the number of independent constraint equations
$\mathbf{P}_c$	the current position of the end effector
$\mathbf{P}_i$	the position of the joint i
$\mathbf{P}_t$	the target position of the end effector
$\mathbf{Q}_e$	the vector containing the externally applied forces
$\mathbf{Q}_{iner}$	the vector of generalized inertial forces
$\mathbf{Q}_V$	the vector of quadratic velocity

$\mathbf{q}$ the vector of Generalized Coordinates

$\dot{\mathbf{q}}$ the vector of the velocity (the time derivative of the vector of Generalized Coordinates)

$\ddot{\mathbf{q}}$ the vector of the acceleration (the second-time derivative of the vector of Generalized Coordinates)

$\mathbf{R}$ the position of the body coordinate system relative to the global coordinates

$\dot{\mathbf{R}}$ the time derivative of the position of the body coordinate system

$\ddot{\mathbf{R}}$ the second time derivative of the position of the body coordinate system

$\mathbf{r}_{A_P}$ the vector of the position of point P on the body A in global coordinates

$\dot{\mathbf{r}}_{A_P}$ the time derivative of the vector of the position of point P

$\ddot{\mathbf{r}}_{A_P}$ the second-time derivative of the vector of the position of point P

$\bar{\mathbf{r}}_i^x$ the rotation vector X of the joint i

$\bar{\mathbf{r}}_i^y$ the rotation vector Y of the joint i

$\bar{\mathbf{r}}_i^z$ the rotation vector Z of the joint i

$\bar{\mathbf{u}}_{A_P}$ the vector of the position of particle P relative to the coordinate system of the body A

V the volume of the body

$\bar{\mathbf{v}}_i$ the vector from rotation joint i to the current position

$\bar{\mathbf{v}}_i'$ the projection of the vector from rotation joint i to the current position

W_{ext} the virtual work of the body

$\bar{\mathbf{w}}_i$ the vector from rotation joint i to the target position

$\bar{\mathbf{w}}_i{}'$ the projection of the vector from rotation joint i to the target position

Greek characters:

θ	the angle of rotation in space around the X-axis
$\dot{\theta}$	the time derivative of the angles of rotation in space
$\ddot{\theta}$	the second-time derivative of the angles of rotation in space
λ_i	the ratio of the body length and distance between the joints
ρ	the density of the body
ϕ	the angle of rotation in space around the Z-axis
ψ	the angle of rotation in space around the Z-axis

Abbreviations:

IK	Inverse Kinematics
PID	Proportional–Integral–Derivative
MSD	Multibody System Dynamic
CCD	Cyclic Coordinate Descent
FABRIK	Forward and Backward Reaching Inverse Kinematics
CPU	Central Processing Unit

Table of contents

1. Introduction

Earth moving is an essential part of the construction of structures such as civil and industrial buildings, communications and networks, main roads. This work involves the use of special earthmoving equipment excavators, bulldozers, graders, rollers, loaders, etc. (Figure 1). Engineers designed each machine for specific types of earthworks and each machine has distinctive technical characteristics: excavators, rippers for digging and bulldozers, loaders, for transporting.

Figure 1. Types of earthmoving machines (Aamirberg 2020).

However, the most well-known and practical earthmoving equipment is the single-bucket excavator. Not every earthmoving job is possible without them. Single-bucket excavators are an indispensable part of any earthmoving operation.

1.1. Background of the topic

The main operating device of the single-bucket excavator is the mobile bucket attached to the boom, arm, or ropes. The hydraulic system ensures both the movement of the bucket and the digging process itself.

Nowadays, the most important challenge in earthmoving machinery is to improve its efficiency. Efficient use includes aspects such as improved control systems, energy efficiency, shorter cycle times and reduced operator fatigue. (A. Gurko & I. Kolobova 2013, p. 59.)

Significant improvements in excavator efficiency are the focus of today's developments. This can be achieved by improving the automatic control systems which include its subsystems and the digging process. In the digging process, the system must ensure that the crane and bucket follow a defined path with a defined accuracy, energy efficiency and speed. (Zhang et al. 2017, Pp. 1-2.) These days the digging process is trending towards the use of robotic excavators. Therefore, it is advisable to use appropriate robotization methods in this system. It is an interesting idea to research and design robotic excavators in specialized applications that focus on the simulation of robotic systems. In this case, Mevea software was used.

Figure 2. Single-bucket excavator model in Mevea software.

1.2. Objectives of research

The main objectives of this master thesis are to select the most appropriate Inverse Kinematics (IK) method for an excavator crane. The Inverse Kinematics method should be fast, has the least number of calculations and have the shortest path. The chosen method must be implemented on an excavator model in Mevea software.

Moreover, the objective of this thesis is to set up a Proportional–Integral–Derivative (PID) controller in the control scheme of excavator crane. The PID controller can reduce energy consumption and increase the productivity of the excavator. The controller should be installed in a proper manner for which it reduces the rising and setting time of the excavator. In other words, PID controller improves the time needed to operate the excavator.

1.3. Questions of research

This research adopts an explorative and descriptive study for which the main objective of this master's thesis is to discover the most effective inverse kinematics method to improve the excavator crane functions. Such analysis is supported by a quantitative study for which the researcher uses the Mevea software to build the required model and Simulink for creating the schemes. In addition, the equations formulated are described and examined in detail. Thus, to achieve the above objectives of this master's thesis, it is necessary to respond to the main research question:

· Which inverse kinematics method is the most appropriate for an excavator crane?

In order to answer the main research question, this research will firstly assess the implementation process. This process will help the researcher to understand the diverse effects, benefits, or implications of each inverse kinematics method. Hence, the first sub-research question is presented:

· How to implement inverse kinematics on an excavator model?

Secondly, an interpretation and explicit description of the PID controller is considered. As previously mentioned, the PID controller plays an imperative role at increasing the productivity of an excavator and achieving valuable results. Thus, it is necessary to comprehend how to use such PID controller in an excavator crane control. This leads the researcher to the second sub-research question:

· How to realize a PID controller in an excavator crane control?

2. Methods

In this chapter of the master's thesis a theoretical description of processes and equations is performed, which are the basic knowledge in the research of the excavator model operation.

The methods chapter includes such sections as:

- Multibody systems. This subchapter provides a complete description of the basic mechanics used for the excavator model. The fundamental knowledge of mechanics includes the description of rigid bodies in space, their coordinates and constraint equations, kinematic and dynamic analysis with the equations of motion.
- Inverse Kinematics (IK). This subchapter contains a description of the three most popular algorithms used in inverse kinematics. Each inverse kinematics algorithm is analysed in detail and compared with the others to select the most appropriate method in the practical part that can be applied to an excavator crane model. The strengths and weaknesses of each method are identified.
- PID controller. In this subchapter, a detailed analysis of the operation of the PID controller according to its components, as well as its adjustment is performed.
- Necessary software for conducting is researched. This section contains a description of the software necessary to carry out the research of this master’s thesis.

Computer analysis allows the research of an object by modelling it in a virtual environment. Due to modern computer technology and enormous computing power, humans have the ability to model and analyse an object of study in a short time and without damaging the experimental prototype or the environment.

2.1. Multibody System

Multibody System Dynamic (MSD) is a tool that allows to perform an analysis by describing its mechanical system. A mechanical system is considered Multibody if the connection of bodies is made by joints that limit and control the movements of bodies.

In this master's thesis, the main object of this analysis is an excavator boom. To realize such analysis, of the MSD is applied. The excavator boom can be represented as a manipulator. In the following, this term will be used more often than the term excavator boom itself.

A manipulator is a kinematic chain formed by connecting rigid bodies, which are called kinematic links, to move the links to the desired position. The connection of the links is most often serial, but also can be parallel and series parallel. The places where kinematic chains connect are called joints. A kinematic chain that forms a manipulator has two ends that are called a base joint and an end effector. If the number of links and joints is large, they are denoted by the index i $(i = 1, \dots, n)$. The base joint is usually denoted by the index 1 and the end effector by the index n. Kinematic links are also denoted by indexes from 1 to $(n - 1)$.

2.1.1. Rigid body

As described above, a manipulator can be called a kinematic chain, the first joint of which is fixed stationary (most often with the ground) and the remaining links can move. A manipulator is spatial if the movement occurs in three-dimensional space. The term is also applicable when moving in one or more parallel planes. A kinematic link consists of a rigid body.

A set of particles for which the distance between them remains constant can be defined as a rigid body. A rigid body does not undergo deformation, that is, its shape remains constant. However, in real life, bodies change shape, but the change in the shape of the body is minimal relative to the motion of the whole body. For this reason, the change in shape of the body is not considered.

2.1.2. Description of a particle that is in space

The description of a rigid body in a space is reduced to the description of one of the particles of this body in space. Figures (3) and (4) show a schematic representation of the body *A* and a particle of the body *P* in two- and three-dimensional spaces. (Flores 2015, p. 13.)

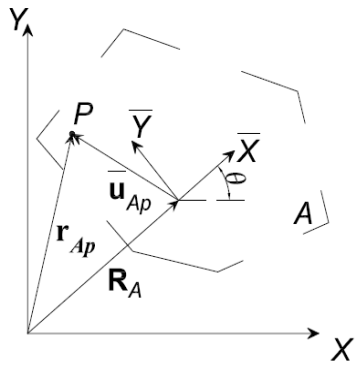


Figure 3. A particle is in two-dimensional space.

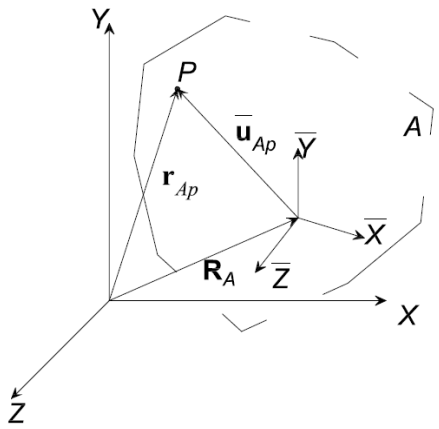


Figure 4. A particle is in three-dimensional space.

The particle *P* of the body *A* in Figure 3 is described with respect to the global coordinate system. To describe the position of the particle *P* in space in matrix form, formula below is used:

$$\mathbf{r}_{A_P} = \mathbf{R}_A + \mathbf{A}_A \bar{\mathbf{u}}_{A_P} \tag{1}$$

where $\mathbf{R}_A$ - position of the body coordinate system relative to the global coordinates (in two-dimensional space it consists of two components ($R_{X,A}, R_{Y,A}$), in three-dimensional space it consists of three components ($R_{X,A}, R_{Y,A}, R_{Z,A}$)). $\bar{\mathbf{u}}_{A_P}$ - vector which describe a position of a body particle relative to the body frame reference (in two-dimensional space consists of two components ($\bar{u}_{X,A_P}, \bar{u}_{Y,A_P}$), in three-dimensional space consists of three components ($\bar{u}_{X,A_P}, \bar{u}_{Y,A_P}, \bar{u}_{Z,A_P}$)). $\mathbf{A}_A$ - rotation matrix describing the rotation of the coordinate system of the body relative to the global coordinate system. The rotation matrix is defined by the expression below:

$$\mathbf{A}_A = \begin{bmatrix} \cos(\theta_A) & -\sin(\theta_A) \\ \sin(\theta_A) & \cos(\theta_A) \end{bmatrix} \tag{2}$$

where θ_A – the angle of the rotation of the body's coordinate system relative to the global coordinate system. For two-dimensional space, formula (1) in vector form is converted into expression below. (Jaiswal et al. 2019, p. 172696.)

$$\begin{bmatrix} r_{X,A_P} \\ r_{Y,A_P} \end{bmatrix} = \begin{bmatrix} R_{X,A} \\ R_{Y,A} \end{bmatrix} + \begin{bmatrix} \cos(\theta_A) & -\sin(\theta_A) \\ \sin(\theta_A) & \cos(\theta_A) \end{bmatrix} \begin{bmatrix} \bar{u}_{\bar{X},A_P} \\ \bar{u}_{\bar{Y},A_P} \end{bmatrix} \tag{3}$$

To describe the position and rotation of a body in three-dimensional space, such methods as Euler angles and quaternions are used. During the Mechanical Engineering program, the Euler angles method was studied. (Flores 2015, p. 17.) Therefore, this method will be

described later in the paper. For three-dimensional space, the rotation matrix according to the Euler angles consists of three components:

$$\mathbf{A} = \mathbf{A}_1\mathbf{A}_2\mathbf{A}_3 \tag{4}$$

where the $\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3$ matrices in turn are expressed in full form as:

$$\mathbf{A}_1 = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{A}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{5}$$

$$\mathbf{A}_3 = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

as a result of solving the expression (4) obtains the rotation matrix for three-dimensional space:

$$\mathbf{A} = \begin{bmatrix} \cos(\phi)\cos(\psi) - \sin(\phi)\cos(\theta)\sin(\psi) & -\cos(\phi)\sin(\psi) - \sin(\phi)\cos(\theta)\cos(\psi) & \sin(\phi)\sin(\theta) \\ \sin(\phi)\cos(\psi) + \cos(\phi)\cos(\theta)\sin(\psi) & -\sin(\phi)\sin(\psi) + \cos(\phi)\cos(\theta)\cos(\psi) & -\cos(\phi)\sin(\theta) \\ \sin(\theta)\sin(\psi) & \sin(\theta)\cos(\psi) & \cos(\theta) \end{bmatrix} \tag{6}$$

The time derivative of the particle position P of the body A with respect to the global frame system gives the velocity $\dot{\mathbf{r}}_{A_P}$ of the particle P with respect to the global frame system. An expression describing the velocity is shown below:

$$\dot{\mathbf{r}}_{A_P} = \dot{\mathbf{R}}_A + \dot{\theta}_A\mathbf{A}_{\theta,A}\bar{\mathbf{u}}_{A_P} \tag{7}$$

where $\dot{\boldsymbol{R}}_A$ is the derivative of the position of the particle with respect to the global reference frame; $\overline{\mathbf{u}}_{A_P}$ - this component remains unchanged and corresponds to the tangential velocity; $\dot{\theta}_A \mathbf{A}_{\theta,A}$ - the derivative of the rotation matrix, which corresponds to the angular velocity.

The time derivative of the velocity gives the acceleration of a point of a solid body. The formula for point acceleration is shown below:

$$\ddot{\mathbf{r}}_A = \ddot{\mathbf{R}}_A + \dot{\theta}_A^2 \mathbf{A}_A \overline{\mathbf{u}}_A + \ddot{\theta}_A \mathbf{A}_{\theta,A} \overline{\mathbf{u}}_A \tag{8}$$

The above expression contains three components of acceleration: translational acceleration ($\ddot{\mathbf{R}}_A$), normal accelerations ($\dot{\theta}_A^2 \mathbf{A}_A \overline{\mathbf{u}}_A$), and tangential acceleration ($\ddot{\theta}_A \mathbf{A}_{\theta,A} \overline{\mathbf{u}}_A$).

2.1.3. The coordinate vectors

A set of coordinates is the coordinates of all bodies that define the system of the bodies and their mechanisms within a certain space. This set of coordinates changes over time if the bodies are in motion. Expression below denotes the set of coordinates of the body system through the vector columns:

$$\mathbf{q} = [q_1, q_2, \dots, q_n]^T \tag{9}$$

where n is the total number of coordinates used in describing the system.

Cartesian and Lagrange coordinate systems are the most used ones. The general difference between Lagrange and Cartesian coordinate systems is that the Lagrange

coordinate system allows us to determine the position of a body relative to the moving coordinate system, while the Cartesian coordinate system is that the position of each body in space was determined relative to the stationary global coordinate system. (Syzrantsev 2016, p. 125.) Thus, if a system of bodies is considered in the Cartesian coordinate system, more coordinates are needed to determine the position of each body in the system (displacement and rotation).

To determine the position of the mechanism in space, it is also necessary to use the coordinate system attached to each body. Suppose a body i (i is the identification number given to each body) is defined by using the global translational coordinates $\mathbf{R}_i = [R_X, R_Y]_i^T$ describing the positions of the body coordinate system and the rotation angle of the body coordinate system relative to the global coordinate system θ. The column vector for two-dimensional space will be $\mathbf{q}_i = [R_X, R_Y, \theta]_i^T$ is the vector of the coordinates of body i in the plane. For three-dimensional space, Euler angles are often used. (Nikravesh, 1988, p. 39.)

2.1.4. Generalized Coordinates

Generalized coordinates are the basis of multibody system dynamics. This topic is the foundation in kinematic analysis, defining the position of a point of a body in space based on a body reference coordinate system. Generalized coordinates are denoted by $\mathbf{q}$ and consist of a description of two components: translation and rotation. The principle of describing a particle of a body by generalized coordinates in a plane coordinate system is as follows:

$$\mathbf{q} = [\mathbf{R}^T \quad \theta^T] = [R_X \quad R_Y \quad \theta]^T \tag{10}$$

where R_X, R_Y is displacement in the plane system, θ is rotation in the plane system.

The principle of describing a body particle by generalized coordinates in a three-dimensional system is similar and presented below:

$$\mathbf{q} = [R_X \quad R_Y \quad R_Z \quad \phi \quad \theta \quad \psi]^T \tag{11}$$

where R_X, R_Y, R_Z are displacement in three-dimensional space, ϕ, θ, ψ are angles of rotation in three-dimensional space.

2.1.5. Constraint Equations

As clarified above, the multibody system dynamics is described by the generalized coordinates shown below.

$$\mathbf{q} = [\, q_1 \quad q_2 \quad q_3 \quad \dots q_n] \tag{12}$$

where n is the number of generalized coordinates.

When two bodies are connected, they are limited in their movements in space. Since the motion of one body determines the motion of the second body, such a two-body construction has fewer degrees of freedom than two bodies taken separately. This feature is called the Constraint Equation. Constraint equations describe joints of a multibody system that relate generalized coordinates. (Flores 2015, Pp. 31–34.) The constraint equation is a function of the generalized coordinates, time, and is expressed in general form as presented below.

$$\mathbf{C} = (\, q_1 \quad q_2 \quad \dots q_n, t) \tag{13}$$

The combination of the constraint equations of the entire multibody system can be written as:

$$\mathbf{C} = [\, C_1(\mathbf{q}, t) \;\; C_2(\mathbf{q}, t) \;\; \dots C_{n_c}(\mathbf{q}, t)] \tag{14}$$

where $n_c$is the number of the independent constraint equations of the system ($n_c \leq n$, where n – the number of generalized coordinates).

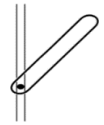
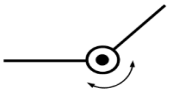
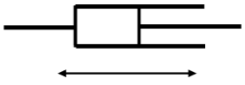
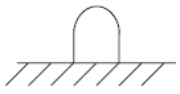
The constraint equations can be:

- Holonomic (Constraint equations are expressed in terms of generalized coordinates and time)
- Nonholonomic (Constraint equations contain inequalities and velocities)
- Scleronomic (Constraint equations are not time-dependent)
- Rheonomic (Constraint equations are time-dependent)

2.1.6. Joints

The places where two bodies connect are called joints. Mathematically, joints are described by constraint equations. Joints can be of different kinds, which are shown in the tables below. Types of joints in two-dimensional space are shown below in Table 1.

Table 1. Joint types in two-dimensional space.

Planar case		
Type of joint	Number of constrain equations, n_c	Example
Primitive Joint	1	
Revolute Joint	2	
Translation Joint	2	
Fixed Joint	3	

Joint types in three-dimensional space are shown in Table 2 below.

Table 2. Joint types in three-dimensional space.

Three-dimensional space case		
Type of joint	Number of constrain equations, n_c (Translation + Rotation = Total)	Example
Planar Joint	1 + 2 = 3	
Spherical Joint	3 + 0 = 3	
Cylindrical Joint	2 + 2 = 4	
Revolute Joint	3 + 2 = 5	
Translation Joint	2 + 3 = 5	
Fixed Joint	3 + 3 = 6	

The number of degrees of freedom of a multibody dynamic system depends on the types of joints. (Flores 2015, Pp. 44–48.)

2.1.7. Degrees of Freedom

The set of the independent coordinates of translation and rotation that completely determine the position of a body or system is called the degrees of freedom. A body in three-dimensional space has six degrees of freedom (3 translation, 3 rotations). The expression below is used to determine the degrees of freedom of a multidimensional system in the MSD:

$$number\ the\ degrees\ of\ freedom = n - n_c \tag{15}$$

where n is the number of generalized coordinates, n_c is the number of constraints, which depends on the joints. Each constraint equation reduces the movement or rotation of the system in a particular direction.

There are three possible versions of the number of degrees of freedom:

- $n - n_c > 0$. (The system is dynamic)
- $n - n_c = 0$. (The system is kinematic)
- $n - n_c < 0$. (System with redundant constraints)

2.1.8. Kinematic Analysis

Kinematic analysis is used in multibody dynamics to determine the position, velocity, and acceleration of the mechanical system. To perform kinematic analysis, the constraint equations of the mechanical system are required.

Kinematic analysis can be performed on systems with the zero degrees of freedom since forces are not considered in the analysis. This means that the number of unknowns expressed in generalized coordinates is equal to the number of constraint equations of the mechanical system.

The set of nonlinear constraint equations of the system can be solved using the Newton-Rapson method, which is shown below in expression:

$$\mathbf{C}(\mathbf{q}, t) = 0 \quad \rightarrow \quad \frac{\partial \mathbf{C}(\mathbf{q},t)}{\partial \mathbf{q}} \Delta \mathbf{q} = -\mathbf{C}(\mathbf{q}, t) \tag{16}$$

where expression $\frac{\partial \mathbf{C}(\mathbf{q},t)}{\partial \mathbf{q}}$ is the Jacobian Matrix, $\mathbf{C}_q$.

The Jacobian matrix is calculated by differentiating by the components. In the Jacobian matrix, the number of columns corresponds to the number of generalized coordinates of the system and the number of matrix rows corresponds to the number of constraint equations.

$$\mathbf{C_q} = \begin{bmatrix} \frac{\partial C_1}{\partial q_1} & \frac{\partial C_1}{\partial q_2} & \cdots & \frac{\partial C_1}{\partial q_n} \\ \frac{\partial C_1}{\partial q_1} & \frac{\partial C_1}{\partial q_2} & \cdots & \frac{\partial C_1}{\partial q_n} \\ \vdots & \vdots & \ddots & \\ \frac{\partial C_{n_c}}{\partial q_1} & \frac{\partial C_{n_c}}{\partial q_2} & & \frac{\partial C_{n_c}}{\partial q_n} \end{bmatrix} \tag{17}$$

The value of $\mathbf{\Delta q}$ is necessary to update the generalized coordinates at each iteration and is closely related to the calculation of the Jacobian matrix. The value of $\mathbf{\Delta q}$ can be calculated by the expression:

$$\mathbf{\Delta q} = -\mathbf{C_q^{-1}C} \tag{18}$$

where $\mathbf{C}$ is the set of constraint equations, $\mathbf{C_q}$ is the set of Jacobian matrices.

After performing the kinematic analysis of the position of the system, the analysis of velocity and acceleration is then performed. To find the velocity, the constraint equations (13) are differentiated with respect to time and the derivative of the generalized coordinates is extracted, as shown in the expression below:

$$\mathbf{\dot{q}} = \mathbf{C_q^{-1}}[-\mathbf{C}_t] \tag{19}$$

where $\mathbf{C}_t$ is the vector formed by the time derivative.

The analysis of the acceleration of the system is performed by differentiating in time of the velocity. The result is the expression shown below:

$$\ddot{\mathbf{q}} = \mathbf{C}_{\mathbf{q}}^{-1}(-\mathbf{C}_{tt} - \left(\mathbf{C}_{\mathbf{q}}\dot{\mathbf{q}}\right)_{\mathbf{q}}\dot{\mathbf{q}} - 2\mathbf{C}_{\mathbf{q}t}\dot{\mathbf{q}}) \tag{20}$$

Kinematic analysis examines a mechanical system of bodies without considering the forces that cause motion, to determine the position, velocity, and acceleration of the system.

2.1.9. Dynamic Analysis

The dynamic analysis of a multibody system can be performed as provided that the multibody system has one or more degrees of freedom. In order to solve a dynamic system, the dynamic equilibrium of the system is necessary, since the set of constraint equations is not sufficient for this. The dynamic equilibrium is expressed by a set of equations of the motion, that in mathematical terms are second-order differential equations. Dynamic equilibrium occurs when the sum of applied forces equals the sum of resultant inertial forces.

Virtual displacement is an endlessly small displacement of a body in a coordinate system at a certain point in time. Virtual work is the work performed at a virtual displacement. Virtual displacement applies to both static equilibrium in between the system and dynamic equilibrium. To express the virtual work of the body, the expression was used below:

$$\delta W_{ext} = \delta \mathbf{r}_o^T \mathbf{F} = \begin{bmatrix} \delta \mathbf{R}^T \\ \delta \theta \end{bmatrix}^T [\mathbf{I} \quad \mathbf{A}_\theta \bar{\mathbf{u}}_o]\mathbf{F} = \begin{bmatrix} \delta \mathbf{R}^T \\ \delta \theta \end{bmatrix}^T \begin{bmatrix} \mathbf{F} \\ \mathbf{A}_\theta \bar{\mathbf{u}}_o \mathbf{F} \end{bmatrix} = \begin{bmatrix} \delta \mathbf{R}^T \\ \delta \theta \end{bmatrix}^T \mathbf{Q}_e \tag{21}$$

where $\mathbf{Q_e}$ is the vector containing the externally applied forces; **F** is the applied forces.

Generalized inertial forces can be expressed according to the Dalembert principle:

$$\mathbf{Q}_{\text{iner}} = \underbrace{\int_{\text{V}} \rho \begin{bmatrix} \mathbf{I} \\ \mathbf{A}^{T}{}_{\theta}\overline{\mathbf{u}}^{T} \end{bmatrix} [\mathbf{I} \quad \mathbf{A}_{\theta}\overline{\mathbf{u}}] dV_A}_{\mathbf{M}} \begin{bmatrix} \ddot{\mathbf{R}}_A \\ \ddot{\theta}_A \end{bmatrix} - \underbrace{\dot{\theta}^2 \int_{\text{V}} \rho \begin{bmatrix} \mathbf{I} \\ \mathbf{A}^{T}{}_{\theta}\overline{\mathbf{u}}^{T} \end{bmatrix} \mathbf{A}_{\theta}\overline{\mathbf{u}}\, d\text{V}}_{\mathbf{Q}_V} \tag{22}$$

where some expressions can be chosen in vector form. $\mathbf{M}$ is the mass matrix of the body, $\mathbf{Q}_V$ is the vector of quadratic velocity; where ρ is the density of the body; V is the volume of the body, $\ddot{\mathbf{r}}$ is the vector of acceleration of the body particles. Formula (22) is an expression, that transforming inertial forces from global coordinates to generalized coordinates. (Khadim et al. 2021, Pp. 4-5.)

The equations of the system below is used to express the virtual displacement which is subject to constraints:

$$\begin{cases} \mathbf{M}\ddot{\mathbf{q}} - \mathbf{Q}_v - \mathbf{Q}_e + \mathbf{C}_{\mathbf{q}}^{T}\boldsymbol{\lambda} = 0 \\ \mathbf{C} = 0 \end{cases} \tag{23}$$

The expression (23) was twice differentiated with respect to time and combine it in matrix form, to obtain Ordinary differential equations used both to find accelerations and to find Lagrange multipliers.

$$\begin{bmatrix} \mathbf{M} & \mathbf{C}_{\mathbf{q}}^{T} \\ \mathbf{C}_{\mathbf{q}} & 0 \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{Q}_e + \mathbf{Q}_v \\ \mathbf{Q}_c \end{bmatrix} \tag{24}$$

The dynamic analysis of the system is necessary to determine the relationship between the forces applied to the mechanical system and its kinematic parameters. The masses of the bodies of the system are also considered. (Flores 2015, Pp. 62–67.)

2.2. Inverse kinematic

Inverse kinematics is the process of determining the parameters of connected bodies (such as a kinematic chain) to achieve the desired position, orientation, and arrangement of these bodies. Inverse kinematics is a method of motion planning and is actively used in robotics, 3D computer animation and computer game development. The solution derived from inverse kinematics problem is considered as the basis for obtaining control over the actions of robot actuators. However, in several cases the solution is complicated by the kinematic redundancy of the mechanism, as well as by the limitations observed within the changes in the generalized coordinates (in turn conditioned by the geometry of the robot's workspace and/or the design of its actuating mechanism).

Kinematic redundancy is an ambiguity an obstacle within the solution of the inverse kinematics problem that occurs when the number of degrees of freedom of the executive mechanism exceeds the dimensionality of the working space. However, even when the mechanisms present with two degrees of freedom, it is possible to achieve the required results from the inverse kinematics method especially when it admits several solutions within certain situations (Figure 5).

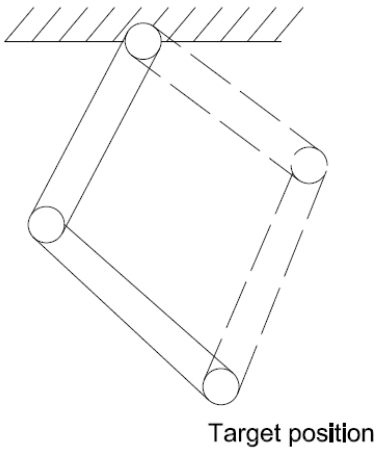


Figure 5. Two possible solutions for a two-link arm.

There are many methods of inverse kinematics, but they all have both advantages and disadvantages. This raises the question of implementing an algorithm that is guaranteed

to find a solution in several iterations, easy to implement, and considers possible constraints on the kinematic chain nodes. (Karginov 2016, p. 38.)

Also, it is necessary to verify whether the inverse kinematics algorithm can easily cope with imposing restrictions on the kinematic chain that simulates the movements of real joints with limited physical loads and velocities at which they are able to operate.

2.2.1. The Cyclic Coordinate Descent Method

One of the most popular methods for solving Inverse Kinematics is the Cyclic Coordinate Descent method (CCD). This method is the simplest and fastest in computations and its application in practice is not problematic. In Inverse Kinematics, the CCD method is a numerical method with repeated iterative calculations, which is easy enough in computations and does not require complex mathematical calculations and decompositions of matrices. However, knowledge of vector algebra is necessary to understand the method in this case. To implement the method of Cyclic Coordinate Descent in reality, and to implement its solution in practice, it is sometimes necessary to correct the existing solution on the part of the engineer. These corrections should not affect the solution of the algorithm in any way and the system to be solved should remain simple, fast, and stable.

Also, the CCD algorithm does not require large mathematical operations and manipulations with matrices solving Inverse Kinematics problems. However, its implementation in complex problems requires technical upgrades and adjustments in the algorithm. Despite the fact, that the matrix of the CCD algorithm assumes the solution of circuits consistently, modifications and amendments in it can cause a difficult solution when calculating complex circuits affecting each other. (Aristidou et al. 2018, Pp. 8–9.)

The Cyclic Coordinate Descent method also has such a disadvantage as the non-convergence of the solution. This disadvantage of the algorithm is formed due to the problems of oscillation conditions and unsteady discontinuities. To avoid these

challenging issues in solving Inverse Kinematics problems, it is necessary to adhere to several determining factors. Some of the main controlling factors are shown below:

- The solution should be quick, and the execution should be done in real time.
- Solutions should be as simple as possible.
- The application and adaptation of constraints if required.
- Balance maintenance (e.g., the centre of mass control).
- Consistency, if changes need to be made, a similar solution is required
- Morphological adaptations

To implement the CCD method, the first task that needs to be applied is defined and the explanation of the entire system is also be introduced. The organs are positioned at the desired position using a set of required articulation angles. The set of articulation angles is used to calculate the distance between the position of the working elements and the target position. This is done by changing, most often increasing, the angle and length of the body of the system forward from the base of the body system to the end body, as in forward kinematics. This action allows us to determine the distance, as an error, between the target position and the position of the final body. Figure 6 shows the implementation of this step of the algorithm.

The next step of the cyclic coordinate descent method is to gradually calculate the following joints closer to the final body and rotate the bodies toward the integer position. This is performed, as stated above, by increasing the angle and length of the following body. Each time the distance to the target position is calculated. (Kenwright 2012, Pp. 177–180.)

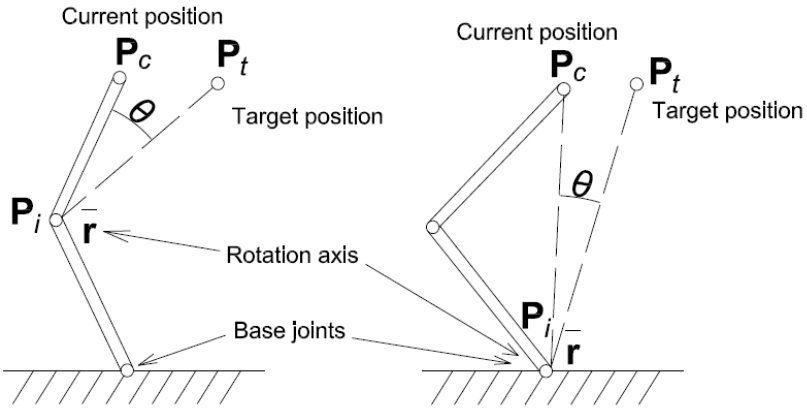


Figure 6. Angle calculation according to the CCD method.

The CCD method algorithm performs the calculation jointly and rotates the body system as close as possible to the desired target. The distance calculation in the CCD method is performed after each iteration of the system. The distance between the final body and the target is calculated to determine the residual distance. Also, to avoid infinite calculations for unreachable and unsolvable problems, it is necessary to set a finite number of iterations.

In most cases, the main joint (the first joint) in inverse kinematics is stationary, and the subsequent joints of the system perform a rotation around a given axis and angle, to reach the target finite body. This step, calculating the axis and angle, is done using the expressions, shown below:

$$\cos(\theta) = \frac{\mathbf{P}_c - \mathbf{P}_i}{\|\mathbf{P}_c - \mathbf{P}_i\|} \cdot \frac{\mathbf{P}_t - \mathbf{P}_i}{\|\mathbf{P}_t - \mathbf{P}_i\|} \tag{25}$$

$$\bar{\mathbf{r}} = \frac{\mathbf{P}_c - \mathbf{P}_i}{\|\mathbf{P}_c - \mathbf{P}_i\|} \times \frac{\mathbf{P}_t - \mathbf{P}_i}{\|\mathbf{P}_t - \mathbf{P}_i\|} \tag{26}$$

where the values $\mathbf{P}_c$, $\mathbf{P}_i$ and $\mathbf{P}_t$ refer to the positions in Figure 6.

One of the most popular mistakes when performing the CCD method is to completely calculate the system from the end joint to the start joint without updating previous

iterations. It should be understood that the iteration of each joint affects the position of the other joints. That is, it also affects the previous and subsequent joints. However, in this algorithm, the calculation can be performed in any order, only the results of the solution will not coincide with each other. As stated above, the iteration of each joint affect's subsequent joints and their iterations, hence the calculation in different orders leads to different solutions. The choice of calculation order can be used if it is required to establish some joints of the system in a stationary, stationary state. Most often such a condition is applied to the initial (base) joint. (Kenwright 2012, Pp. 183–184.)

In the CCD method, it is necessary to pay attention to some points that have a big influence on the system solution:

- The choice of joints to calculate and the order in which they are updated (bottom-up or top-down calculation, or calculation from joints with a large error)
- Increasing or decreasing the value of the correction angle
- Permissible positions (areas in space in which the calculated position of the system satisfies the required solution)

Below consider the application of the CCD method on the example of the robot arm. In this case there is a kinematic chain having n bodies connected to each other by axial joints. At the end of robot arm there is a working body represented by a point $\mathbf{P}_c$. Below is a schematic representation of the considered kinematic chain.

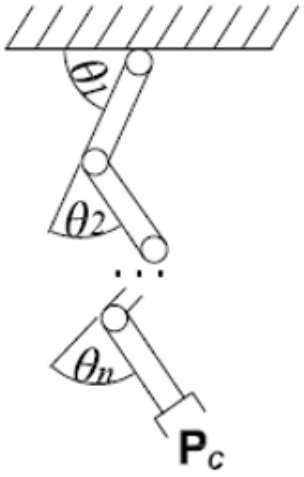


Figure 7. Rotating the bodies of the robot arm at θ_i angles.

Suppose at the initial moment of time the kinematic chain, which is in three-dimensional space and in an admissible position, is defined by a set of angles of rotation θ in the joints of the robot arm. The end working body at the initial moment is at a point $\mathbf{P}_c$. There is also some point $\mathbf{P}_t$ in this space. In this case, the problem of inverse kinematics of controlling a manipulator arm with a working body at the end is to calculate the rotation angles $\Delta\theta_i$ that will move the working body of the robot arm from point $\mathbf{P}_c$ to point $\mathbf{P}_t$. It is also possible to move to the point closest to the target point $\mathbf{P}_t$ if it is not attainable. In addition, constraints $\theta_{i,min} \leq \Delta\theta_i \leq \theta_{i,max}$ can be imposed on the rotation angles. This problem can have either one solution, or several solutions, or no solutions at all. This problem does not consider the orientation of the working body, but only its position in space. (Strashnov & Mikhailyuk 2017, Pp. 189–191.)

Consider rotation at joint i. Let denote the position of joint i as $\mathbf{P}_i$, and denote the vector of the rotation vector of joint i on axis as $\bar{\mathbf{r}}_i^z$. Two vectors $\bar{\mathbf{v}}_i$ and $\bar{\mathbf{w}}_i$ must be determined, which are calculated using the expressions below:

$$\bar{\mathbf{v}}_i = \mathbf{P}_c - \mathbf{P}_i \tag{27}$$

$$\bar{\mathbf{w}}_i = \mathbf{P}_t - \mathbf{P}_i \tag{28}$$

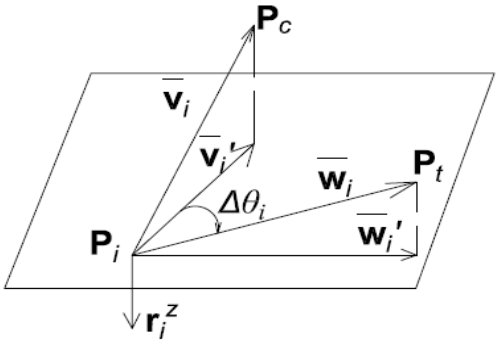


Figure 8. Rotation of the i joint.

If to draw a plane perpendicular to the vector $\bar{\boldsymbol{r}}_i^z$ and project vectors $\bar{\boldsymbol{v}}_{\boldsymbol{i}}$ and $\bar{\mathbf{w}}_{\boldsymbol{i}}$ onto it, as in Figure 8, these projections can be written as expressions below:

$$\bar{\mathbf{v}}_i{}' = (\bar{\mathbf{v}}_i{}' \cdot \bar{\mathbf{r}}_i^x)\bar{\mathbf{r}}_i^x + (\bar{\mathbf{v}}_i{}' \cdot \bar{\mathbf{r}}_i^y)\bar{\mathbf{r}}_i^y \tag{29}$$

$$\bar{\mathbf{w}}_i{}' = (\bar{\mathbf{w}}_i \cdot \bar{\mathbf{r}}_i^x)\bar{\mathbf{r}}_i^x + (\bar{\mathbf{w}}_i \cdot \bar{\mathbf{r}}_i^y)\bar{\mathbf{r}}_i^y \tag{30}$$

where $\bar{\mathbf{r}}_i^x$ and $\bar{\mathbf{r}}_i^y$ are mutually perpendicular vectors lying in the plane.

In order to superpose the working body of kinematic chain at the point $\mathbf{P}_c$ with point $\mathbf{P}_t$, vector $\bar{\mathbf{v}}_i{}'$ and vector $\bar{\mathbf{w}}_i{}'$ should be superposed. This is achieved by rotating vector $\bar{\mathbf{v}}_i{}'$ around vector $\bar{\mathbf{r}}_i^z$. The angle of this rotation is determined by the expression below:

$$\Delta\theta_i = \arccos(\bar{\mathbf{v}}_i{}' \cdot \bar{\boldsymbol{w}}_i{}') \cdot sign(\bar{\mathbf{r}}_i^z \cdot (\bar{\mathbf{v}}_i{}' \times \bar{\mathbf{w}}_i{}')) \tag{31}$$

The calculated angle $\Delta\theta_i$ needs to be corrected if there are constraints on θ_i. This correction procedure is presented below:

$$\begin{aligned}
&if\ \theta_i + \Delta\theta_i < \theta_{i,min},\ then\\
&\Delta\theta_i = \theta_{i,min} - \theta_i;\\
&if\ \theta_i + \Delta\theta_i > \theta_{i,max},\ then\\
&\Delta\theta_i = \theta_{i,max} - \theta_i;
\end{aligned}$$

Then the constrained new angle and the total change of angle in joint i after the calculations are determined as:

$$\theta_i' = \theta_i + \Delta\theta_{i,new}, \theta_i = \theta_i' \tag{32}$$

$$\Delta\theta_i' = \Delta\theta_i + \Delta\theta_{i,new}, \Delta\theta_i = \Delta\theta_i{}' \tag{33}$$

The Cyclic Coordinate Descent method is iterative. For this method, the most preferred approach is the backward descent, that is, the calculation of joints starts from the closest joint to the working body at the end of the kinematic chain, and then a gradual descent to the initial (base) joint. (Strashnov & Mikhailyuk 2017, p. 191.) The iteration in this method is a rotation in each joint. The calculation process begins with the n joint. Then for each joint i a rotation in $n, n-1, \dots, i$ joints are performed. There are cases when for some joint j the rotation angle is nonzero, then in this case the joint j is rotated by the calculated angle and the same process is performed for $i = n$. If all calculated angles of joints $n, n-1, \dots, i$ turned out to be zero, then it is necessary to switch one joint higher and execute the process with $i = i - 1$. One of the two conditions must be met to complete the iterations:

- The number of iterations performed is greater than the given maximum number of iterations.
- The working body (the end of the kinematic chain) has reached the necessary position with the required accuracy. (Strashnov & Mikhailyuk 2017, p. 192.)

2.2.2. The Target Triangle Method (The Triangulation Method)

Muller-Cajar and Mukundan introduced the Target Triangle Method in 2007. This method is to some extent an improvement and modernization of the CCD method. The triangulation method takes into account the distance from the target to the base and rotates the entire chain, including the base, if the target is unreachable. The Triangulation IK is calculated the joint angle using the law of the cosine. Thanks to the law of the cosine, this method makes it possible to find the desired position in a few iterations. Another big advantage of the triangulation method is that it is not computationally intensive. However, using triangulation in inverse kinematics to solve problems with multiple end effectors is challenging. This method is suitable for circuits with a few links.

The law of cosines states that the square of any side of a triangle is equal to the sum of the squares of the other two sides of the triangle minus twice the product of those sides by the cosine of the angle between them.

$$c^2 = a^2 + b^2 - 2ab \cdot cos\varphi \tag{34}$$

where a, b, and c are the three sides of the triangle; φ is the angle opposite of c.

Using the scalar product of two normalized vectors to find the angle between them, this formula allows to find the angles needed to complete the triangle in three-dimensional space. (Muller-Cajar & Mukundan 2007, p. 2.)

The law of the cosine allows the triangulation method in order to use the properties of a triangle to compute the angles required when moving a chain to a target. An adapted cosine law for this method is shown below:

$$\varphi = cos^{-1}(-\frac{c^2-a^2-b^2}{2ab}) \tag{35}$$

where a is the length of the body (chain) moving in space, b is the remaining chain length, and c is the distance from the target to the base joint.

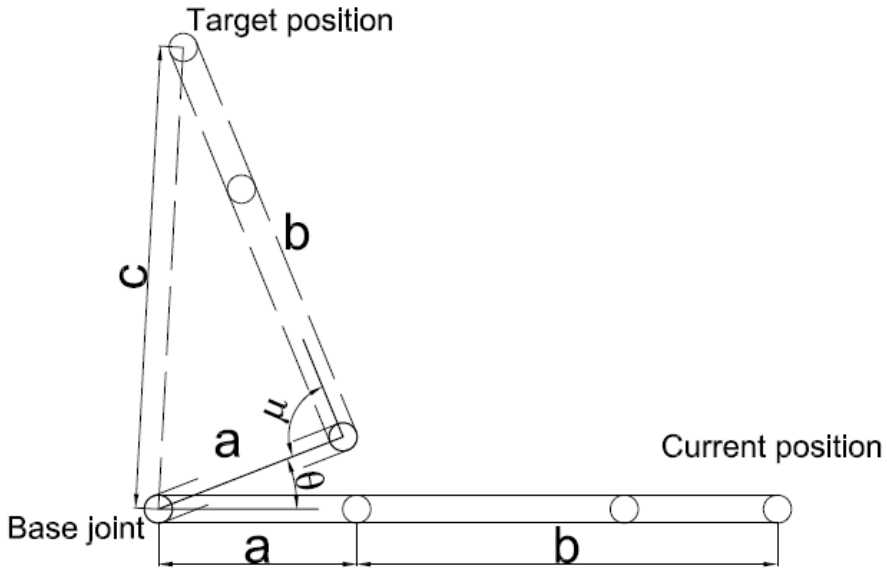


Figure 9. The application of the Triangulation method on a kinematic chain.

The figure schematically shows the circuit and its potential displacement according to the Triangular method. To realize this displacement, it is necessary to find the angle between vector $\bar{\mathbf{a}}$ with respect to the potential position of vector $\bar{\mathbf{a}}$. In order, to find the angle θ the expression below is used:

$$\theta = cos^{-1}(\bar{\mathbf{a}} \times \bar{\mathbf{c}}) - \varphi \tag{36}$$

The angle θ is the angle by which the joint is rotated in the direction of the axis of rotation $\bar{\mathbf{r}}$. The axis of rotation $\bar{\mathbf{r}}$ can be found from the expression below:

$$\bar{\mathbf{r}} = (\bar{\mathbf{a}} \times \bar{\mathbf{c}}) \tag{37}$$

With proper calculation and the most appropriate system, the Triangulation method rotates performs a rotation only in the final two joints of the chain, when the rest of the system bodies are stationary. This action is an attempt to keep calculations to a minimum, but this does not often happen. There are cases of applying the Triangulation Method when the above equations cannot be calculated in some system bodies. Such cases should be considered separately. There is a possible case when the rotation axis $\bar{\mathbf{r}}$ that cannot be defined. This occurs if vector $\bar{\mathbf{c}}$ is parallel to the direction of vector $\bar{\mathbf{a}}$. In this case, the rotation vector $\bar{\mathbf{r}}$ is chosen randomly, or the rotation axis that was used when solving the last joint is used. Constant axes, such as the vertical y-axis, can also be used.

The successful calculations of equations (36) and (37) produce a triangle with sides a, b, c. Next, after successfully finding the sides, according to the algorithm, it is necessary to rotate vectors $\bar{\mathbf{a}}$ and $\bar{\mathbf{b}}$ relative to the rotation axis $\bar{\mathbf{r}}$ toward the target vector $\bar{\mathbf{c}}$.

The Triangulation method of inverse kinematics cannot construct the required triangle, and hence cannot be applied in two cases:

1. If $c > a + b$. This means that the link cannot reach the target by considering the current joint. Since the target is too far away to be reached, it is only possible to move the final working body at the end of the chain as close as possible to the target in space. To do this, the chain is rotated so that:

$$\bar{\mathbf{a}} = \bar{\mathbf{c}} \tag{38}$$

2. If $c < |a + b|$; this means that the target is very close to the current joint. Suppose that the side $b > a$. Consequently, to reach the target, vector $\bar{\mathbf{a}}$ must be rotated to the opposite side of vector $\bar{\mathbf{c}}$, and then the end of vector $\bar{\mathbf{b}}$ will be as close to the target as possible. This case can be expressed as shown below:

$$\bar{\mathbf{a}} = -\bar{\mathbf{c}} \tag{39}$$

$$\lim_{a \to b \to c} \delta_b = 180^o \tag{40}$$

If side $a > b$; then there is no solution in this situation.

Unfortunately, the above solutions to the resulting problems do not work on joints that have constraints. In this case, if the chain consists of many bodies, it is possible to shorten one of the sides as shown in Figure 10. From the figure, it is seen that the side b in this case is shortened, by folding the bodies.

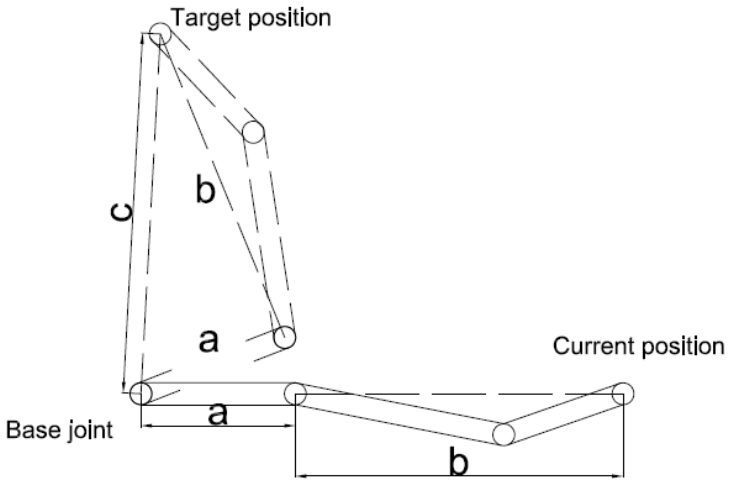


Figure 10. The shortening of the b side of the kinematic chain.

If the target cannot be reached because of limitations, or because it is too close, or because it is too far away, in such cases it is simply necessary to move the body at the end of the chain as close as possible to the target. (Song & Hu 2011, p. 352.)

The Triangulation method algorithm performs calculations on each joint, starting at the beginning (base) joint and finishing at the end joint. The calculations for each joint are similar. If the solution is found before the last joint or at the last joint, the Triangulation method is applicable. In general, one equation system is used for each joint, but the calculation and rotation start from the base joint that is the farthest to the end of the chain. An example would be the shoulder joint of the human arm. Below is an algorithm for implementing the Triangulation method:

For each Joint i do
    Calculate $\bar{c}_i$, c;
    if $c \geq a + b$ then
        $\bar{a}_i = \bar{c}_i$;
    end
    else if $c < |a + b|$ then
        $\bar{a}_i = -\bar{c}_i$;
    end
    else
    $\theta = cos^{-1}(\bar{a} \times \bar{c}) - cos^{-1}(-\frac{c^2-a^2-b^2}{2ab})$;
    if $\bar{a}_i = -\bar{c}_i$ or $\bar{a}_i = \bar{c}_i$ then
        $\bar{r} = (0, 1, 0)$;
    end
    else
        $\bar{r} = (\bar{a} \times \bar{c})$;
    end
    $rotate\ (\bar{a}_i\ by\ \theta\ about\ \bar{r})$;
    end
end

Figure 11. The algorithm of the Triangulation Method (Muller-Cajar & Mukundan 2007, p. 4).

When joint i is calculated, the Triangulation method algorithm keeps the orientation of this joint unchanged relative to the last joint and calculates its position in global space. If this action cannot be performed according to the algorithm, then after calculating the

position of joint i in global space, it is necessary to recalculate the previous joints and perform a rotation by the θ angle. Unfortunately, the Triangulation method of the IK becomes less effective in the case of frequent recalculations of the entire chain. Also in this algorithm, needs to calculate the maximum length of the chain and save it to determine the target reach.

If constraints are imposed on some joints of the chain, the calculation of the solution to the inverse kinematics problem gets larger, and the application of the Triangulation method becomes less effective. To solve an inverse kinematics problem with a constraint on a joint relative to the previous joint, it is necessary to perform a constraint check before turning, and if possible, reduce the turning angle. Unfortunately, in solving a problem with constraints, the Triangulation method requires frequent recalculations of the entire chain of bodies. (Muller-Cajar & Mukundan 2007, p. 4.)

Comparing the CCD method and the Triangulation method:

The comparison of the Cyclic Coordinate Descent method and the Triangulation method was made on the basis of article by (Muller-Cajar & Mukundan 2007) and article by (Song & Hu 2011). Referring to the experiments carried out in these articles, it is possible to identify certain differences in these two methods of inverse kinematics.

The experiments consisted in the achievement of a certain goal of space available by the system of bodies, using various methods of solving the problems of inverse kinematics. The results of the experiments are shown below in Figure 12.

Figure 12. Comparison of the Triangulation method and the CCD method (Song & Hu 2011, p. 354).

The authors of the article note that, because of the experiment, the use of the Triangulation method is preferable for several reasons:

1) Speed of calculations. The Triangulation method in this aspect is better than the CCD method. This is explained by the fact that Triangulation does not calculate joints every time, only once, while CCD frequently iterates joints. Even despite the constraints of the joints used on the body chain, the Triangulation method is preferable.
2) Body Position. When using the Triangulation method algorithm, the position of bodies in space is observed to be more natural and straightforward. This difference in the Triangulation method is justified by starting calculations from the initial (base) the joint of the chain, while the CCD method calculates from the end of the chain, i.e., from the closest joint to the working body. It should also be noted that the position of bodies in the Triangulation method is observed to be straighter, with a minimum amount of twisting, which affects the computational capacity, the speed of calculation. (Mukundan 2009, p. 306.)

It should be noted that these distinctions refer to theoretical data. In the system considered in this master's thesis, the results of the study may be different. Comparison of the CCD method and the Triangulation method for the excavator model under study will be carried out in the practical part of the work.

1.2.3. The FABRIK Method

Moving towards a given goal within a certain position is done by moving a system of rigid bodies connected by joints, and by rotating these joints. This is one of the tasks considered for the inverse kinematics. The solution of this problem requires the most accurate and rapid reach towards the goal, and it is desirable that the movement should be soft and natural. Existing inverse kinematics methods do not always satisfy the required parameters. The popular disadvantages of the old inverse kinematics methods: the high computational power is the requirement, combined with the unnatural final position. One of the most modern, convenient, and fastest methods is the Forward and Backward Reaching Inverse Kinematics (FABRIK) heuristic method, which calculates forward and the Inverse Kinematics of the system. (Tao et al. 2021, Pp. 1–2.)

The FABRIK method is useful because it does not require high computational power. This method can calculate the position of the body system in several iterations, i.e., the calculation of matrices and the using of rotations is not required, since the direct calculation of positions occurs. Also, when solving the inverse kinematics problem using the FABRIK method, the body system takes a natural position as possible, even with the calculation of constraints and several final effectors. (Danilov et al. 2018, p. 3.)

As an example of using this method, consider a manipulator consisting of a chain of rigid bodies connected to each other through the joints. This manipulator can perform both the function of a robot arm and the function of an animated character. Using the FABRIK method of the Inverse Kinematics, the rotation and movement of each body in the chain of rigid bodies affects all other bodies in the chain. The base joint of the chain of rigid bodies, or it is called the root joint, is usually considered stationary in Inverse Kinematics problems. To calculate the IK problem by the FABRIK method, each joint of the chain of rigid bodies is a final effector, from which, going down the chain of rigid bodies to the base joint, a new calculated chain is formed.

The FABRIK method stands for Forward and Backward Reaching Inverse Kinematics. In this method, the positions of the joints are calculated by forward and reverse calculation. The rotation of the rigid body of each joint is performed according to the algorithm of the method, hence, the error in rotation is minimal. Calculation of the positions of the chain of solids starts from the end effector and goes down to the base joint, bearing in mind to adjust each joint angle. Then the same calculation is performed in the reverse direction: from the initial joint to the final effector. The FABRIK method of inverse kinematics does not solve the rotation transformation problem. This method solves the problem of positioning joints and bodies according to the problem of finding a point on a straight line. With this method, the time spent on solving the inverse kinematics problem is reduced and high computational power is not required for its implementation. (Aristidou & Lasenby 2011, Pp. 243–244.)

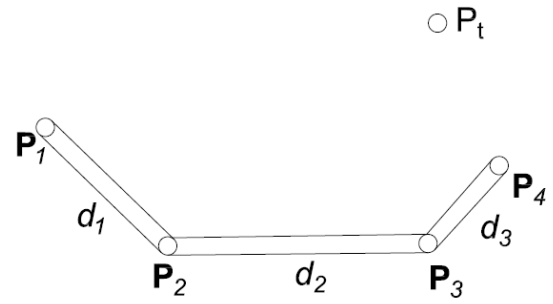


Figure 13. An example of a kinematic chain.

Consider the example shown in Figure 13. In this example, there is a manipulator consisting of three bodies and has joint positions $\mathbf{P}_i$ ($i = 1 \ldots 4$), where $\mathbf{P}_1$ is the base joint of the chain of solids, $\mathbf{P}_4$ is the end effector of the chain (the end node). $\mathbf{P}_t$ is the designation of the target which should be achieved by the chain of rigid bodies in this example. Figure 14 shows the implementation of the FABRIK method of the Inverse Kinematics schematically which consider the example of the chain of rigid bodies.

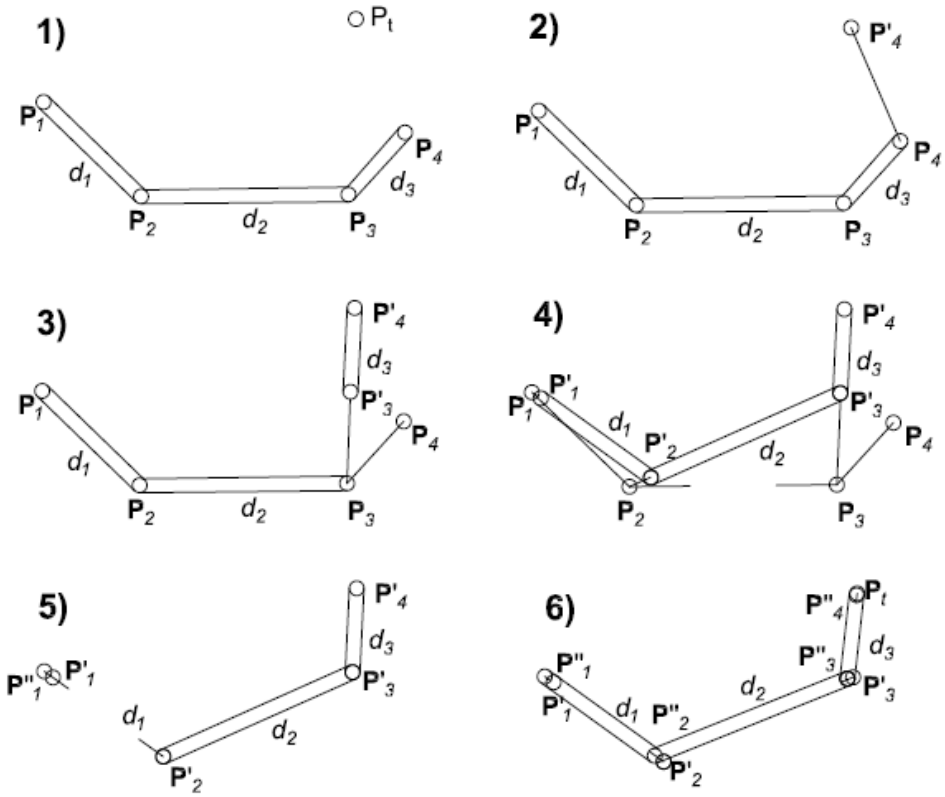


Figure 14. The graphic interpretation of the cycle of the FABRIK method.

Consider the algorithm of the FABRIK inverse kinematics method according to the example shown in Figure 14.

1. The manipulator in the initial position with positions **P** and the goal $\mathbf{P}_t$ to be reached by the chain of rigid bodies is depicted.
2. The end effector $\mathbf{P}_4$ is moved to the position of the target according to its coordinates $\mathbf{P'}_4$.
3. Then the calculation of the joint position $\mathbf{P'}_3$, which is at the length of the third body, and in the direction from the joint position $\mathbf{P'}_4$ to the joint position $\mathbf{P}_3$ is performed.
4. The previous step (calculation of the new joint position) is performed for the following joints to the base point.
5. The next step of the algorithm begins with the basic joint position.
6. The Forward Calculation step is performed from the basic joint and goes up through the joints to the end effector.

This FABRIK method algorithm calculates the positions of the joints until the end effector approaches a distance that is sufficient according to the assignment. Also, before implementing the FABRIK method algorithm, it is necessary to check the reachability of the target position. To do this, compare the distance from the base joint to the target and the length of sum of each body of the chain of rigid bodies. If the sum of the body lengths is less than the distance from the base joint to the target, then the target is unattainable. If the sum of lengths is greater than the distance to the target, then the target is attainable and then the algorithm is executed. (Santos et al. 2021, p. 53425.)

Next, consider the algorithm of the FABRIK method from a computational point of view. First, before the computation, it is necessary to determine all the positions of the joints of the body chain from the base joint $\mathbf{P}_1$ to the end effector $\mathbf{P}_4$. The next step is to set the end effector $\mathbf{P}_4$ according to the target position $\mathbf{P}_t$, denote the new position as $\mathbf{P'}_4$. Next, calculate the new position of the third joint $\mathbf{P'}_3$ at a distance from the previous joint $\mathbf{P'}_4$ and toward the current position of the third joint $\mathbf{P}_3$. To calculate this position, the

mathematical term "division of a segment in a given ratio" is used. That is, to calculate a new position at a given distance and, in each direction, it is necessary to use formulas below:

$$d_i = |\mathbf{P}_{i+1} - \mathbf{P}_i| \tag{41}$$

$$\lambda_i = l_i / d_i \tag{42}$$

$$\mathbf{P}_i = (1 - \lambda_i)\mathbf{P}_{i+1} + \lambda_i \mathbf{P}_i \tag{43}$$

where d_i is the distance between the previous position and the considered one; λ_i is the ratio of the body length and distance between the joints; $\mathbf{P}_i$ is a position.

For the following joints, the chain of rigid bodies is calculated like formulas described above. The new position relative to the previous joint in the direction of the joint in consideration is also calculated. This procedure calculates all positions of the chain of solids, including the new position of the base joint. If a body system is considered where the root element also moves to a certain position, then the FABRIK inverse kinematics method performs the calculation in the same way as described above. Only this way, the new position of the base joint will not be the initial position of the joint, but the target position. (Aristidou & Lasenby 2011, Pp. 245–246.)

The end effector of the chain of solids approaches closer and closer to the target position after performing each iteration. This algorithm of the FABRIK method is executed until the final effector is set in the target position or is set in a tolerance area around the target position. The FABRIK method is easily implemented if the joints and bodies are unconstrained. If the target position is reachable for the end effector and there are no constraints on the joints, then the inverse kinematics problem is easily solved by the FABRIK method. If the target position is unattainable, it requires a comparison operator before the algorithm is executed, or an interrupt to stop the algorithm. It is also necessary to set a certain number of iterations, in order to avoid the huge number of calculations that can happen at some hard-to-reach positions.

Constraints are used to make the chain of rigid bodies take the most natural position similar to organisms (arms, legs) after the implementation of the algorithm. Each joint can be described by three degrees of freedom, of which two degrees of freedom describe just rotation (final position), and the third degree of freedom describes rotation around its own axis. Consequently, to control a joint, it is possible to divide the motion of the joint into two phases with the addition of constraints.

Since the FABRIK method algorithm is iterative, the joint constraints are calculated at each iteration of the algorithm. The constraints imposed on the joints do not affect the execution of the algorithm. In this case, the joints of the chain of rigid bodies will be reoriented and rearranged according to the constraints imposed on them. (Aristidou et al. 2016, p. 37.) Next, analyse an example of the implementation of the FABRIK method algorithm with joint constraints to the example shown in Figure 15.

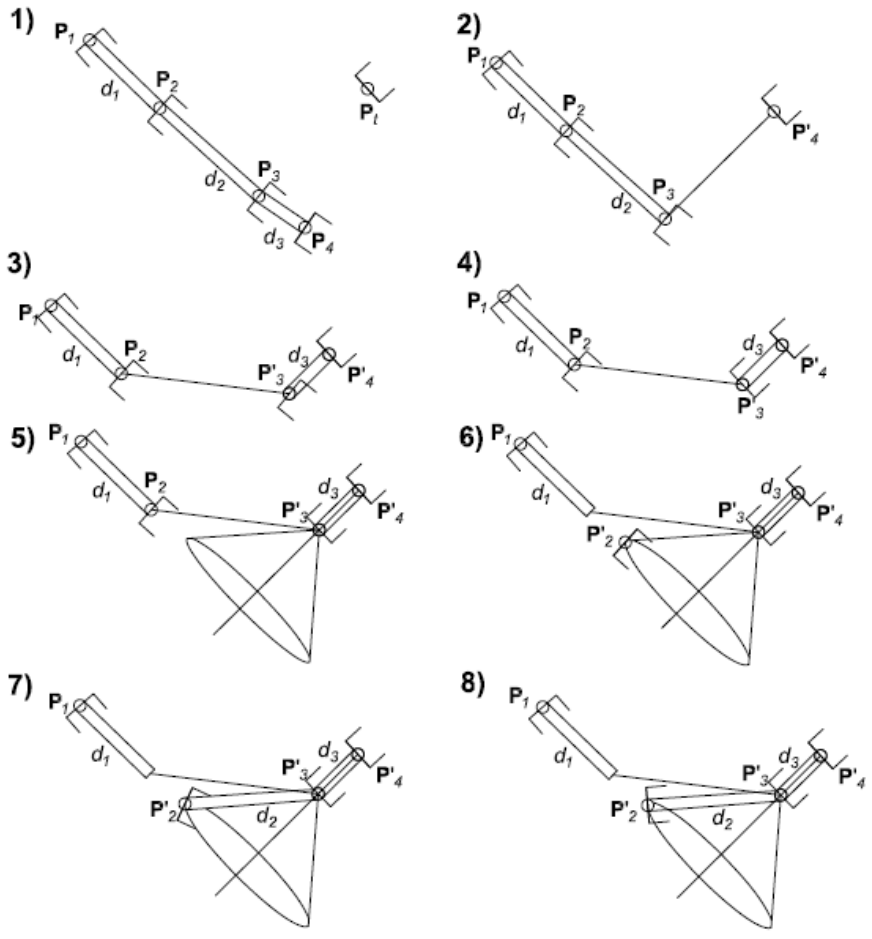


Figure 15. The graphic interpretation of the cycle of the FABRIK method with constraints.

1. The initial position of the chain of solids with joint constraints and target position of the end effector $\mathbf{P}_t$.
2. The end effector from position P_4 is moved and oriented to the target position pt, and denoted by $\mathbf{P'}_4$
3. Calculation of the new position of the third joint $\mathbf{P'}_3$ at a distance d_3 from $\mathbf{P'}_4$ and in the direction from $\mathbf{P'}_4$ toward $\mathbf{P}_3$
4. Rotating the $\mathbf{P'}_3$ joint so that its direction according to the constraints is toward $\mathbf{P'}_4$
5. This step calculates the ellipse within which the new position of the second joint can be located according to the constraints of $\mathbf{P'}_3$.
6. Moving the second joint from position $\mathbf{P}_2$ to the new position $\mathbf{P'}_2$, so that new position is within the constraints of the third joint.
7. Next, make the displacement of the second joint $\mathbf{P'}_2$ so that the distance to the third joint corresponds to the length of the body d_2.
8. Rotate the second joint according to the constraints of the joint.

This algorithm is performed in the forward and reverse direction for each result, as well as the algorithm without constraints. (Aristidou & Lasenby 2011, Pp. 247–249.)

The FABRIK inverse kinematics method is quite easily applicable to different body systems. This method is often used in animating the motion of the bodies of virtual characters (the movement of arms and legs) and this method is also used to drive robotic manipulators (arms) in production facilities. This method provides the most natural positioning of body systems and does not require high computing power. (Aristidou et al. 2018, p. 9.)

2.3. PID controller

PID controllers are an integral part of modern industry and production. PID controllers are used to controlling manufacturing processes. More than 95 percent of closed-loop operations on manufacturing are controlled by a PID controller. A PID controller includes

3 components: Proportional, Integral, and Differential. This controller allows to accept the control signal at the desired level due to feedback.

Previously, PID controllers were analogy controllers. It was controlled by analogy electronic components. However, nowadays, thanks to the invention of microprocessor technology, PID controllers are based on microprocessors. Programmable logic controllers (PLC) involve using a PID controller and setting it up inside the system. Modern PID controllers are reliable and flexible, and its use in production or industry is still relevant today. (Munshi 2012, Pp. 46–47.)

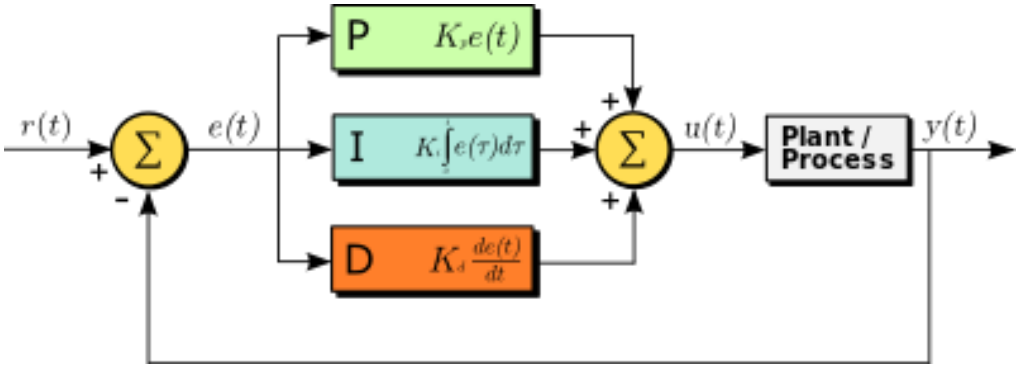


Figure 16. PID controller schematic (Mehta et al. 2017, p. 118).

The most primitive and simple controllers are two-position controllers. They have two positions: ON or OFF. For some processes, this two-state controller is sufficient to control the process. Most of the time these processes have limited control. However, in most cases the two-state controller is not suitable for process control. The signal output must be set so that the error between the current value and the set point is as low as possible. For this purpose, a PID controller with closed-loop operations is used.

2.3.1. PID controller components

This section analyses the components of the PID controller.

P-controller:

The principle of the proportional controller or P-controller is based on a signal change proportional to the current error $e(t)$. The controller multiplies the proportional constant by the error obtained by comparing the current value and the set point value to obtain the correct output signal with the minimum error. The output of the controller is zero at zero error.

This controller is unstable, so an offset or manual reset must be always entered. This is necessary when used separately. The operation of the P controller is stable, but there is always a constant error. The response speed depends on the proportional coefficient K_p. If the K_p increases, the response speed increases.

I-controller:

The I-regulator is needed to eliminate the steady-state error that the P-regulator, which always present with an error, has. The I controller eliminates the regulation error by integrating the error that appeared from the P controller. The error is zero after some time. If there is a negative error, the I controller reduces the output. It is also a disadvantage that the system is unstable, and the response speed of the system is reduced when using the I controller. If the integral gain K_i is reduced, the response speed rises.

With a PI controller, the system is provided with a high response speed. And the controller is limited when P and I controllers are used together. This is due to the limitations of the integral coefficient and the output of the integral coefficient can be increased with non-linearity.

D-controller:

The future behaviour of the system and the system error depends on the D-coefficient. The D-regulator solves this future error problem by expecting future system behaviour.

The rate of change of the system error affects the D-coefficient. The response of the system becomes faster with the D-coefficient.

2.3.2. Setting methods for the PID controller

Adjustment of the PID controller is necessary to obtain the process output data of the required quality. The controller is tuned separately for each P, I and D coefficient. Most often these are set to one at the beginning and then the desired setting is made based on the results. There are many adjustment methods available at the moment. The designers themselves choose the most suitable method for their requirements. The result of the tuning must be an output that meets the requirements of the application.

Process curve tuning: This method is one of the most popular. To implement this method, a process curve must be obtained with the coefficients set by the engineer. Once the curve is obtained, the rise time, decay time, and slope of the curve are calculated. Thanks to the values obtained the P, I and D coefficients for the PID controller are calculated.

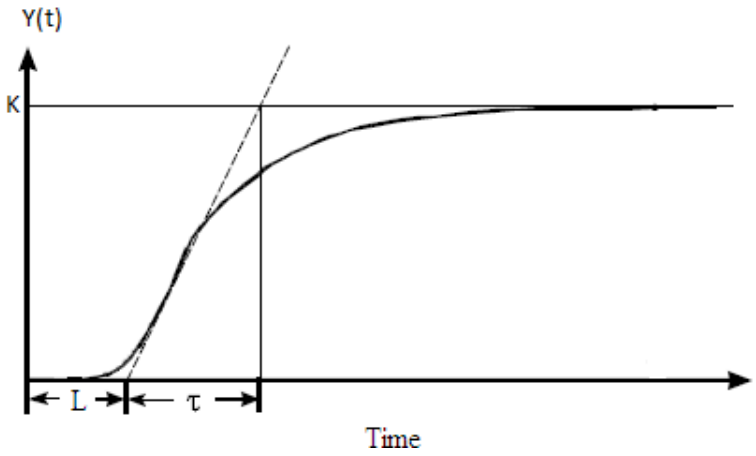


Figure 17. Process curve (Daraz et al. 2017, p. 517).

Trial and error tuning: This is the most primitive method of tuning regulators. The controller tuning starts with finding the proportional coefficient. After adjusting the proportional coefficient and setting the process curve to oscillating mode, the integral coefficient must be adjusted. The I coefficient affects and eliminates oscillations. Then, the D coefficient is adjusted to increase the response speed.

Autotuning: This type of tuning is usually used in engineering software where the regulators are tuned automatically. The system itself selects the most appropriate coefficients according to the implemented algorithm. It is also possible for an engineer to correct the coefficients afterwards. (Çinar et al. 2019, Pp. 4–5.)

Ziegler-Nichols method: Tuning with this method is the most accurate in the field of automatic control. This method is used in a closed loop. The essence of the method is oscillation reduction and cycling. In this method, as in the trial-and-error method, the proportional coefficient is adjusted before any oscillations occur. In this system two values exist: period of oscillation (P_l), final amplification (K_u). Using these two values P, I, and D coefficients are calculated according to Table 3. Different tables exist for different types of regulators.

Table 3. Calculation of controller coefficients according to the Ziegler-Nichols Method. (Mustafa & Nsaif 2019, p. 10.)

Ziegler-Nichol's method giving K values (loop tomes considered constant and equal to dT)			
Control Type	K_p	K_i	K_d
P	$0.50\ K_u$	0	0
PI	$0.45\ K_u$	$1.20\ K_p dT/P_l$	0
PID	$0.60\ K_u$	$2\ K_p dT/P_l$	$K_p P_l/(8dT)$

The Method Ziegler-Nichols is the easiest to tune PID controllers. Using Table 3, it is easy to calculate the controller coefficients for the regulated system.

2.4. Necessary software for conducting research

This section of the master's thesis describes the software used to write this work. The main programs were: MEVEA Software and Simulink Software by MATLAB. Also, this section describes the capabilities of the software and the requirements for working in these programs.

2.4.1. MEVEA Software

Mevea software is a special software for operating in the complex field of mechanical 3 modelling in real time. This software was created by Mevea ltd. Mevea ltd. was established by researchers from the Lappeenranta University of Technology in 2005. Since then, Mevea has made significant progress in the development and simulation of complex mechatronic systems. Today, Mevea ltd. helps customers around the world to innovate by enabling Multiphysics Digital Twins throughout the product lifecycle in a wide variety of industries.

Mevea software includes operations such as creation and simulation of complex mechanical systems, the visualization of the virtual environment, the adjustment of forces according to physics laws (friction, collisions), development and adjustment of hydraulic and drive systems. In Mevea software there is no possibility to create 3D models, so in order to perform simulation it is necessary first of all to import 3D models from specific CAD software into Mevea software. The Mevea software includes a few separate programs for convenient and comfortable work in the mechatronic system simulation framework:

- Mevea Modeller. The program serves as a tool for creating, building, and configuring the simulated system. In this program the basic operations (mechanics, hydraulics, and powertrain) are performed for creating a complex mechanical system.

- Mevea Solver. The program that serves as the user interface. This program run built models in real time.
- Mevea Visual Channel. The program for visualizing a simulation on one or more computers.
- Mevea Remote Access. This program allows to perform a simulation investigation from a remote computer.

The programs used in this research are Mevea Modeller and Mevea Solver. The tools of these programs are enough to complete the research of the master's thesis.

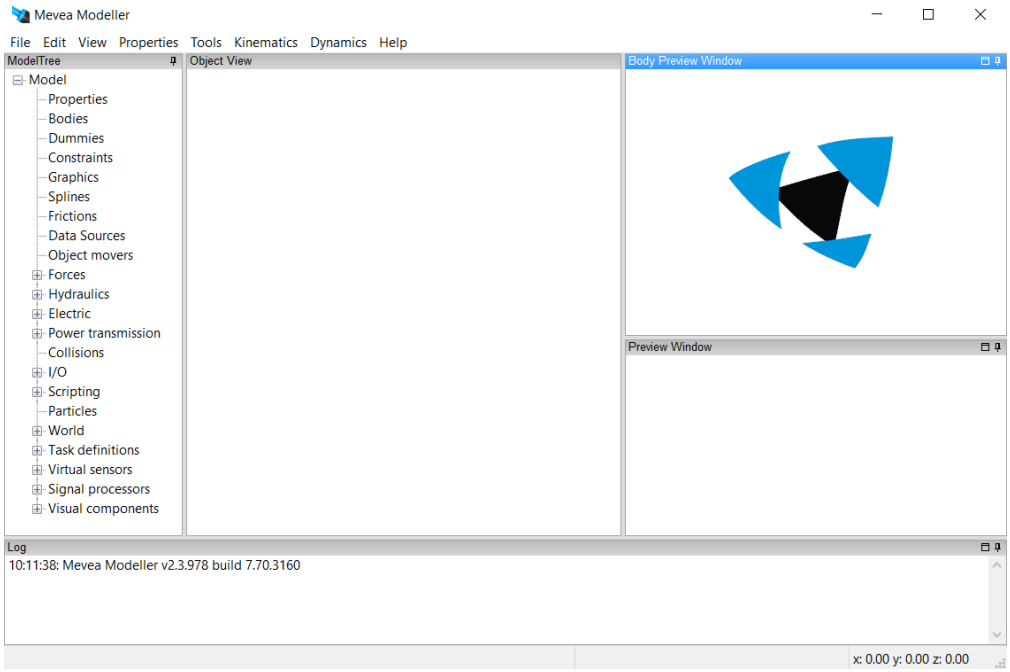


Figure 18. The user interface of Mevea Modeler.

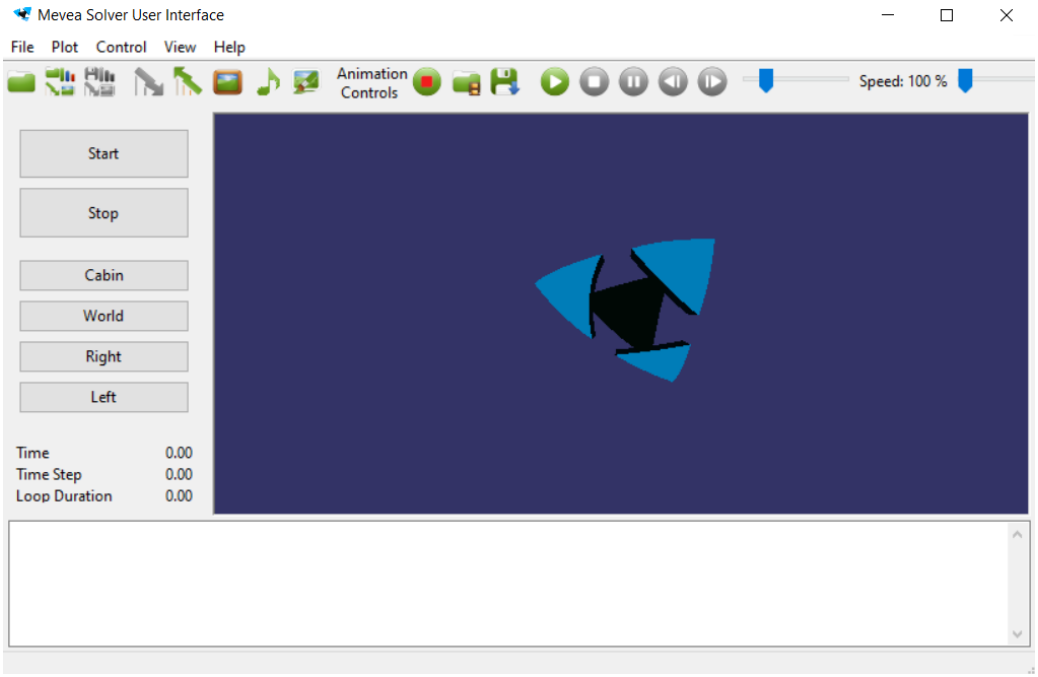


Figure 19. The GUI interface of Mevea Solver.

It is also necessary to make sure that the computer used meets the requirements of the Mevea software. The minimum and recommended requirements for the Mevea software are provided in Table 4.

Table 4. Requirements for the Mevea Software.

	Minimum	Recommended
OS	Windows 7	Windows 7
Processor	Intel Core2 Duo 2.66GHz	Intel Core i7 3.5GHz
Memory	512 MB	4 GB
Display	NVidia GeForce 6600GT	NVidia GeForce GTX1070

The Central Processing Unit (CPU), memory, and display adapter requirements depend on the complexity of the simulation. For complex tasks, a faster processor and more cores are required.

2.4.2. Simulink Software

Simulink is an engineering tool for modelling various types of systems (mechanical, hydraulic, electrical, etc.). This program is one of the basic programs among engineers for system design. Simulink has a huge library of tools that help to simulate the system. Simulation is performed with the help of graphical diagrams. Simulink and MATLAB are deeply integrated tools. It is possible to use them together. Simulink is a very useful tool for creating automated systems.

Simulink is designed for modeling and simulation at the system level, which allows a comprehensive study of the developed system in a single design environment. Simulink software is the most convenient tool for designing and implementing complex control systems. The Simulink interface is simple and intuitive for the user doing research in this program. The requirements of the Simulink Software are shown in Table 5.

Table 5. Requirements for the Simulink Software.

	Minimum	Recommended
OS	Windows 10	Windows 10
Processor	Intel or AMD x86-64	Intel or AMD x86-64
RAM	4 GB	8 GB
Memory	3.4 GB of HDD	HDD, SSD
Display	No requirements	OpenGL 3.3 with 1GB GPU memory

In this master's thesis, the Simulink software is necessary to implement the inverse kinematics of the model under investigation. For its implementation it is initially necessary to import the simulation data of the excavator model into the Simulink software from the Mevea software. Data importing should be performed in real time. Next, the output data from the excavator model must be calculated according to the selected inverse kinematics algorithm. The values calculated in the Simulink software must be imported

back into the Mevea software for further crane control of the excavator model. This procedure must be performed in real time.

The Simulink software is also needed to implement a PID controller. The PID controller can be implemented using the existing PID controller block in the Simulink library.

3. Results and analysis

This chapter of the master's thesis describes the application of three Inverse Kinematics methods on an excavator model in MAVEA Software. The algorithms of Inverse Kinematics methods are made with the help of Simulink MATLAB software. The data necessary to build the Inverse Kinematics algorithm is sent to Simulink software, and then the commands necessary to set the excavator arm in the calculated position are sent back to MAVEA software. The image of the excavator model, on which the Inverse Kinematics algorithms described in the Methods chapter are applied, is shown in Figure 20 below.

Figure 20. The Excavator model in MEVEA Software.

The model in Figure 20 describes as realistically as possible the work of the excavator in accordance with the real excavator. The model in consideration respects the laws of physics: gravity, collisions, hydraulics, etc. To denote the final target of the inverse kinematics, a grey ball is used. In this case such objects of the excavator model are considered as:

- Undercarriage
- Uppercarriage
- Main Boom
- Dipper Arm
- Bucket Tilt Frame
- Bucket
- Ball

3.1. Description of the model and experiments

In this master's thesis for the excavator model used in the Mevea software, four joints are considered. These joints are responsible for connecting the chain of bodies with each other and their rotation.

- Joint 1. This joint is located between Uppercarriage and Undercarriage. Joint 1 rotates Uppercarriage in the XZ plane.
- Joint 2. This joint is located between Uppercarriage and Main Boom. Joint 2 is the initial (base) joint for the excavator arm.
- Joint 3. This joint is located between Main Boom and Dipper Arm bodies and their projections rotate in the XY plane.
- Joint 4. This joint connects Dipper Arm and Bucket. It is the final moving joint of the excavator arm. The Bucket is the final effector of the excavator arm.

Figure 21 below shows the excavator model schematically. As can be noticed, all the bodies and joints are shown. The illustrated joints are used below to solve the inverse kinematics problem.

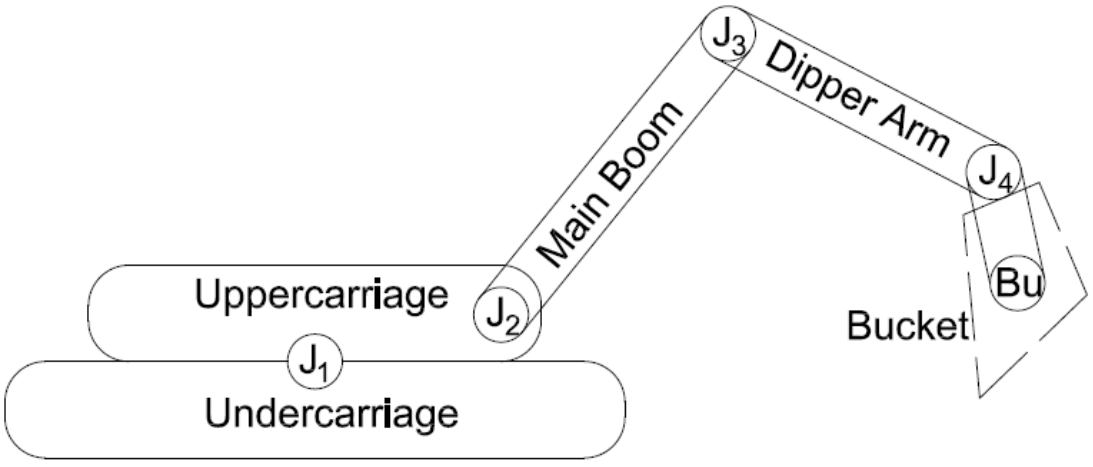


Figure 21. The schematic illustration of the excavator model.

To implement the Inverse Kinematics algorithms, it is necessary to upload some data from the excavator model to Simulink MATLAB in real-time. In this case, it is necessary the connection points of two bodies, i.e., the joints. It is by focusing on their positions that Inverse Kinematics calculations are performed. Table 6 shows the data that are sent to MATLAB to calculate the IK algorithms.

Table 6. Output signals transferred from Mevea to Simulink.

Name of Output	Channel Block, Index	Type of data	Location
Joint_1_X	10,1	Coordinate System	Between Undercarriage and Uppercarriage
Joint_1_Y	10,2	Coordinate System	Between Undercarriage and Uppercarriage
Joint_1_Z	10,3	Coordinate System	Between Undercarriage and Uppercarriage
Joint_2_X	10,4	Coordinate System	Between Uppercarriage and Main Boom
Joint_2_Y	10,5	Coordinate System	Between Uppercarriage and Main Boom
Joint_2_Z	10,6	Coordinate System	Between Uppercarriage and Main Boom
Joint_3_X	10,7	Coordinate System	Between Main Boom and Dipper Arm
Joint_3_Y	10,8	Coordinate System	Between Main Boom and Dipper Arm
Joint_3_Z	10,9	Coordinate System	Between Main Boom and Dipper Arm
Joint_4_X	10,10	Coordinate System	Between Dipper Arm and Bucket Tilt Frame
Joint_4_Y	10,11	Coordinate System	Between Dipper Arm and Bucket Tilt Frame
Joint_4_Z	10,12	Coordinate System	Between Dipper Arm and Bucket Tilt Frame
Bucket_X	10,13	Body	Centre of body
Bucket_Y	10,14	Body	Centre of body
Bucket_Z	10,15	Body	Centre of body
Ball_X	10,16	Dummy	Centre of dummy
Ball_Y	10,17	Dummy	Centre of dummy
Ball_Z	10,18	Dummy	Centre of dummy

Also, to achieve the position of the excavator arm according to the calculated Inverse Kinematics values, it is necessary to send commands to move the components of the excavator arm. The input commands sent to Mevea Software are shown in Table 7.

Table 7. Input signals transferred from Simulink to Mevea.

Name of Output	Channel Block, Index	Type of signal	Type of control Input
Input_Slew (J_1)	0,0	Analog	DV63 (Directional Valve)
Input_BoomLift (J_2)	0,1	Analog	DV63 (Directional Valve)
Input_DipperArm (J_3)	0,2	Analog	DV63 (Directional Valve)
Input_Bucket (J_4)	0,3	Analog	DV63 (Directional Valve)

This master's thesis raises the question of the most suitable Inverse Kinematics method for the Mevea excavator model. To solve the problem of choosing the method, it is necessary to compare the IK methods in as many areas and directions as possible: left, right, front, back, top, and bottom of the excavator. Only by comparing IK methods in all the different directions can the most reliable result be obtained. In this case, a measurement plan for the experiment was made. This plan takes into account all sides relative to the excavator. The positions of the target relative to the excavator in height are taken into account, and the measurements of the targets in the areas within reach and out of reach of the excavator are also considered. Figure 22 below shows the measurement plan for each IK method. Based on the measurements from the plan below, the most appropriate IR method is selected.

To perform the measurements of the experiment, it is also necessary to set the position of the targets according to the above-described measurement plan. Table 8 shows the coordinates of the target positions (ball) of the experiments. These coordinates take into account the maximum possible number of target positions in space relative to the excavator model.

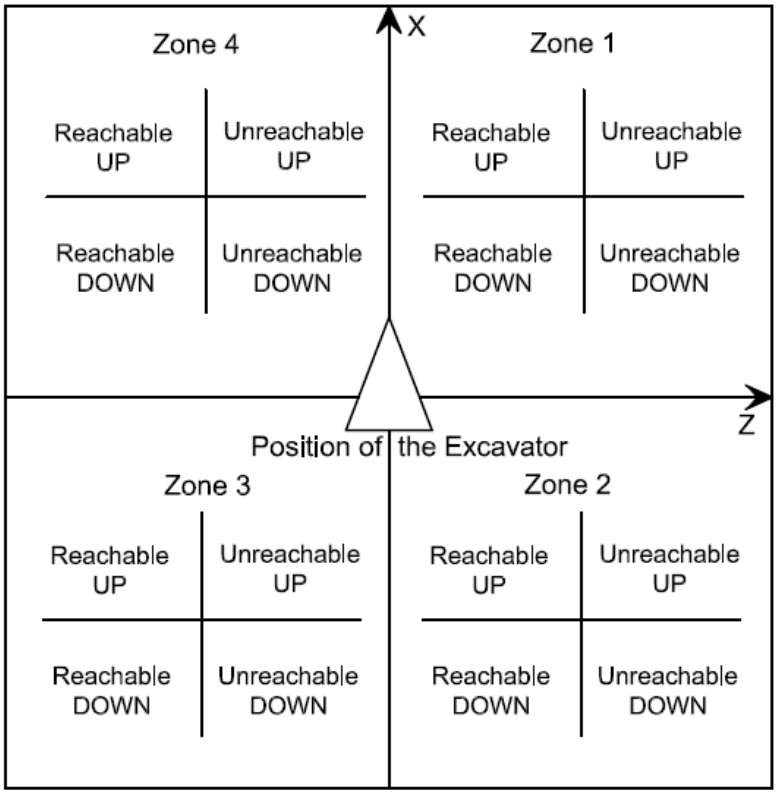


Figure 22. Research measurement plan.

Table 8. Coordinates of the target positions of the experiment.

№ Zone	Position in a space		X	Y	Z
1 ZONE	UP	Reachable	-52	4,5	-119
		Unreachable	-48	4,5	-119
	DOWN	Reachable	-52	2	-119
		Unreachable	-48	2	-119
2 ZONE	UP	Reachable	-61	4,5	-119
		Unreachable	-65	4,5	-119
	DOWN	Reachable	-61	2	-119
		Unreachable	-65	2	-119
3 ZONE	UP	Reachable	-61	4,5	-125
		Unreachable	-65	4,5	-125
	DOWN	Reachable	-61	2	-125
		Unreachable	-65	2	-125
4 ZONE	UP	Reachable	-52	4,5	-125
		Unreachable	-48	4,5	-125
	DOWN	Reachable	-52	2	-125
		Unreachable	-48	2	-125

It is also necessary to set the conditions for ending the measurement of the experiment. After a thorough study of the model and the connection of Mevea and Simulink software, it was decided that the measurement process of the experiment should stop under two conditions:

1. The data measurement process of the experiment stops when the target is reached. The bucket of the excavator gets into the position of a grey ball. For this condition, the position accuracy (tolerance) is 0.1 meter.
2. The measurement process of the experiment is stopped when the position of the excavator bucket does not change for 5 seconds. That is, if the bucket does not move for more than five seconds, it is considered that it has taken the best position relative to the target, but it has not reached the target.

Figure 23 shows the Simulink subsystem, which is responsible for fulfilling the two measurement stopping conditions described above. Simulink blocks such as: if condition block, MATLAB function, step and delay were used to fulfil the conditions.

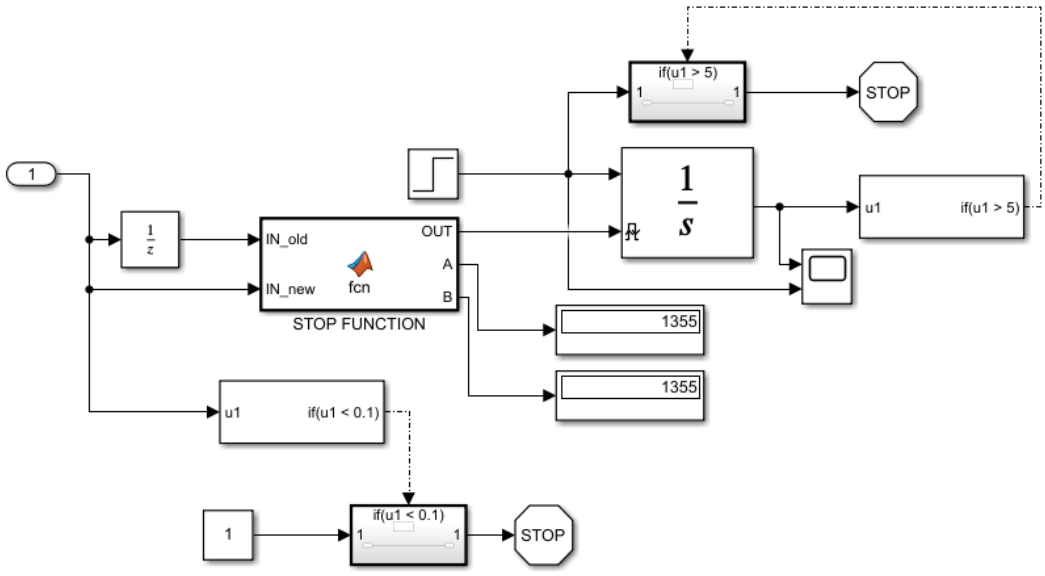


Figure 23. Schematic for stopping the measurement process.

Figure 24 shows the MATLAB "Stop Function" block code, which values are compared to fulfil the second stopping condition.

```
function [OUT,A,B]= fcn(IN_old,IN_new)
A = round (IN_new*10);
B = round (IN_old*10);
if A == B
    OUT = 1;
else
    OUT = 0;
end
end
```

Figure 24. Value comparison function for stopping measurements.

The above description of the model, its joints, the measurement plan of the experiment, the position of the targets and stops of the measurements of the experiment contribute to the practical part of the master's thesis.

3.2. Implementation of PID controllers

The controller is used to this study when the signal is applied to the actuator (Table 7). In this study, the controller's built-in Autotune tuning function was used to tune the controller. In the course of this work, 4 regulators were added, per actuator.

When implementing PID controllers on the Simulink circuit, feedback was used to form the control signal with the required accuracy and transient quality. Also, the ready block Simulink PID Controller was used. This block is convenient to set up and implement PID controllers. PID controllers are installed before the signal blocks in Mevea to the actuators.

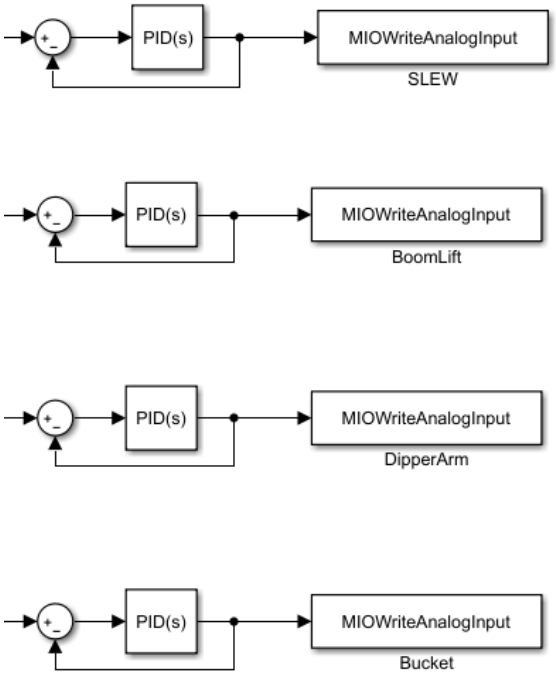


Figure 25. Implementation of PID controllers in the Simulink scheme.

To determine the best Inverse Kinematics method, the controller settings were used similarly for all three methods. In the process of tuning Autotune the I controller was chosen because the transient curves changed only from changes in the I coefficient.

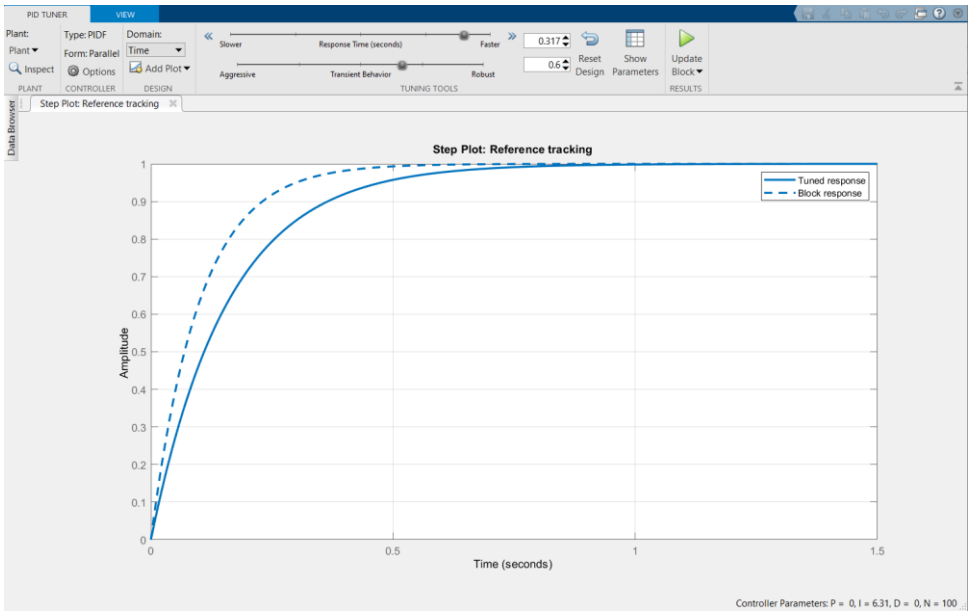


Figure 26. The controller configuration in Simulink.

Table below shows the adjustment coefficients of four regulators for four actuators.

Table 9. The controllers coefficients.

Actuator	P	I	D
Slew (J_1). Power 1	0	10	0
BoomLift (J_2). Power 2	0	2	0
DipperArm (J_3). Power 3	0	2	0
Bucket (J_4). Power 4	0	2	0

During the adjustment of the regulators, it was decided that the coefficient I of the Slew actuator (J1) should be greater than that of the other actuators. This is required for faster rotation of the excavator. The Upper carriage should start moving before the excavator arm moves for a more natural movement. For the other actuators, it was decided to use a coefficient I of equal to 2. This value ensures a smoother, more uniform movement of the excavator arm.

Using PID controllers before signalling the actuators ensure the smooth motion of the excavator arm chain of bodies. Correct tuning made it possible to reduce the time of movement, and to ensure smooth motion. In this case, the controller was adjusted only by the coefficient I.

3.3. Applying the Cyclic Coordinate Descent Method

This method is quite simple and the most popular Inverse Kinematics method. CCD method is considered as one of the very first Inverse Kinematics methods.

The schematic implementation of the CCD Inverse Kinematics method is shown in the figure below. Figure 27 shows a schematic implementation of this method in the Simulink MATLAB software.

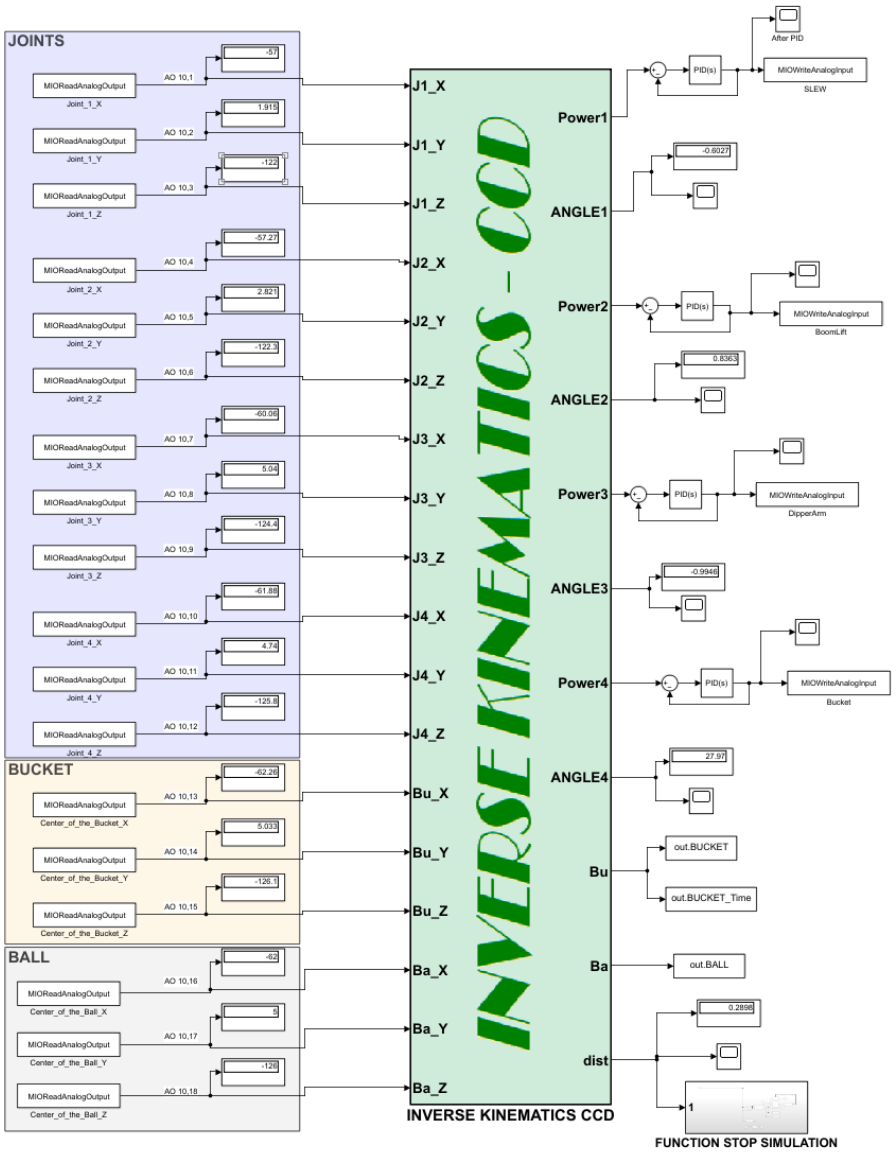


Figure 27. The schematic implementation of the CCD method.

The essence of the CCD method of the IK is to rotate each joint of the kinematic chain of bodies one by one. The first rotation from the current position to the target position starts from the end joint (J4). Then the calculation and rotation of the lower joint, up to the base joint, are performed. The base joint of the excavator arm is joint J2. Also, in parallel at the same time, the calculation and rotation of joint J1 is performed.

To implement the CCD method on the excavator, model the formulas (25 – 33) described above in theoretical part of master's thesis were used. Based on these formulas, an algorithm for writing code for the CCD method was created.

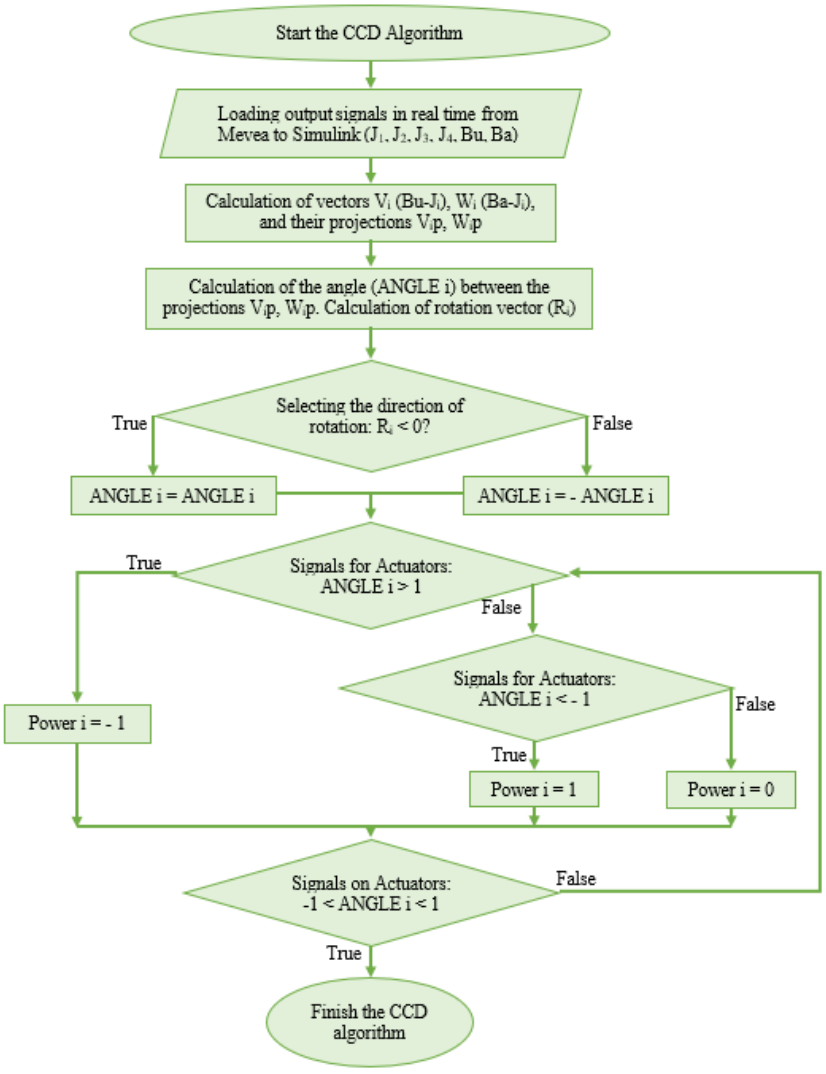


Figure 28. The algorithm of CCD method implementation on the excavator model.

Following the algorithm shown in Figure 28, the CCD method code was written for Mevea excavator model. The algorithm is not an exact description of the code, but a guiding algorithm, the plan of which was followed when writing the CCD method code.

The code was written in a MATLAB function block called "INVERSE KINEMATIKS CCD". The source code itself can be seen in Appendix 1 of master's thesis under the name Listing 1. After writing the code-algorithm of Inverse Kinematics CCD method the experiment with measurements according to Figure 22 was carried out. The target positions were used according to Table 8. The tuning of the regulators was done in the above-described way.

As a result of applying the Cyclic Coordinate Descent method of Inverse Kinematics, the following values are obtained, presented in Table 10.

Table 10. Results of applying the CCD Method.

1 ZONE							
UP				DOWN			
Reachable		Unreachable		Reachable		Unreachable	
PATH	4,811	PATH	4,662	PATH	5,297	PATH	4,684
DIST	0,100	DIST	2,524	DIST	0,100	DIST	2,405
TIME	4,650	TIME	3,550	TIME	9,850	TIME	3,650
2 ZONE							
UP				DOWN			
Reachable		Unreachable		Reachable		Unreachable	
PATH	22,486	PATH	21,001	PATH	21,170	PATH	20,696
DIST	0,100	DIST	1,604	DIST	0,100	DIST	1,478
TIME	8,950	TIME	6,700	TIME	6,200	TIME	6,650
3 ZONE							
UP				DOWN			
Reachable		Unreachable		Reachable		Unreachable	
PATH	21,218	PATH	19,690	PATH	19,769	PATH	19,411
DIST	0,100	DIST	1,607	DIST	0,100	DIST	1,479
TIME	8,650	TIME	6,400	TIME	11,450	TIME	6,350
4 ZONE							
UP				DOWN			
Reachable		Unreachable		Reachable		Unreachable	
PATH	3,702	PATH	3,890	PATH	4,686	PATH	3,899
DIST	0,100	DIST	2,524	DIST	0,100	DIST	2,407
TIME	3,050	TIME	3,550	TIME	4,300	TIME	3,600

This table shows the results in three criteria: the excavator bucket path length, the distance from the bucket of the excavator end position to the target position (accuracy), and the time to stop the experiment (stopping performs under the two conditions described above). The excavator bucket has reached all reached positions, which proves that the CCD method Inverse Kinematics code was written correctly.

Also, during the investigation of the CCD method, the graphs of the excavator bucket path were taken before the experiment was stopped. These graphs can be observed in Appendix 2 of the master thesis (Figures 34-41). The final position of the excavator arm according to the CCD Method is provided in Appendix 3 in Figures 58-65.

3.4. Applying the Target Triangle Method

This method is famous for its fast calculations and the most natural finite positions of the body system. In terms of computational speed, this method is considered faster than the CCD method. However, in this case time, accuracy and path are considered the most important. Figure 29 shows a schematic implementation of this method in the Simulink MATLAB software package.

The essence of this method is to represent the kinematic chain of bodies in the form of a triangle and with the subsequent calculation of the angles of the conditional triangle in accordance with the law of cosines. Calculation of the target triangle method begins with the base joint. After the angles are calculated, signals are sent to the actuator to set the chain of bodies in accordance with the calculated angles of the triangle. If the existing triangle is not able to reach the target, a triangle not including the base joint is calculated. The triangle is calculated from the joint above the base joint in the body chain.

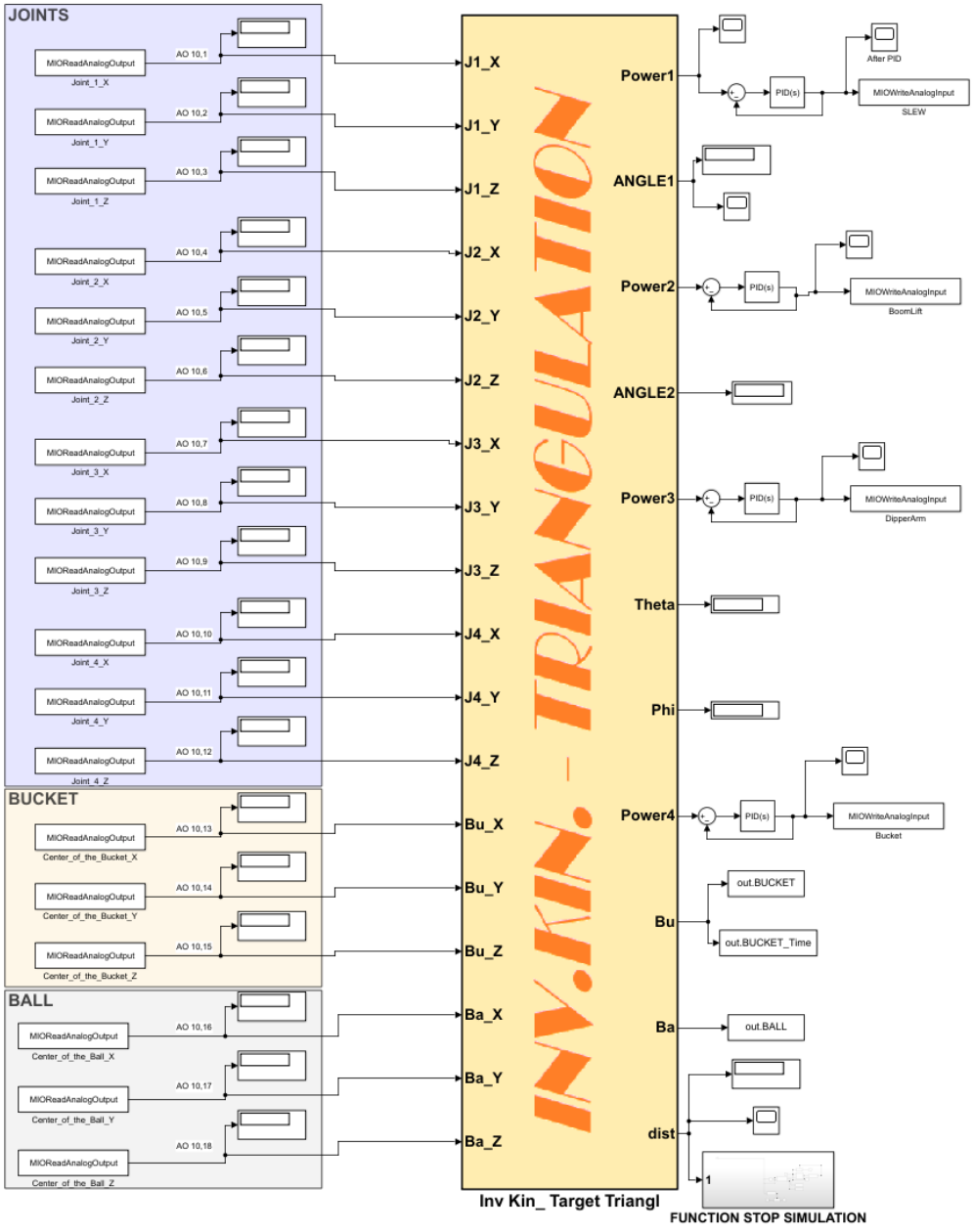


Figure 29. The schematic implementation of Triangulation method.

Figure 30 shows the algorithm of the Inverse Kinematics Target Triangle method for the excavator model. The Triangulation algorithm is based on formulas (34-37) described above in the theoretical part of the master's thesis. Based on the depicted algorithm, the source code of the triangulation method was written in the MATLAB function "InvKin_Target Triangle" block. The algorithm in Figure 31 is not an exact description

of the code. Triangulation method code was written with reference to the algorithm composed.

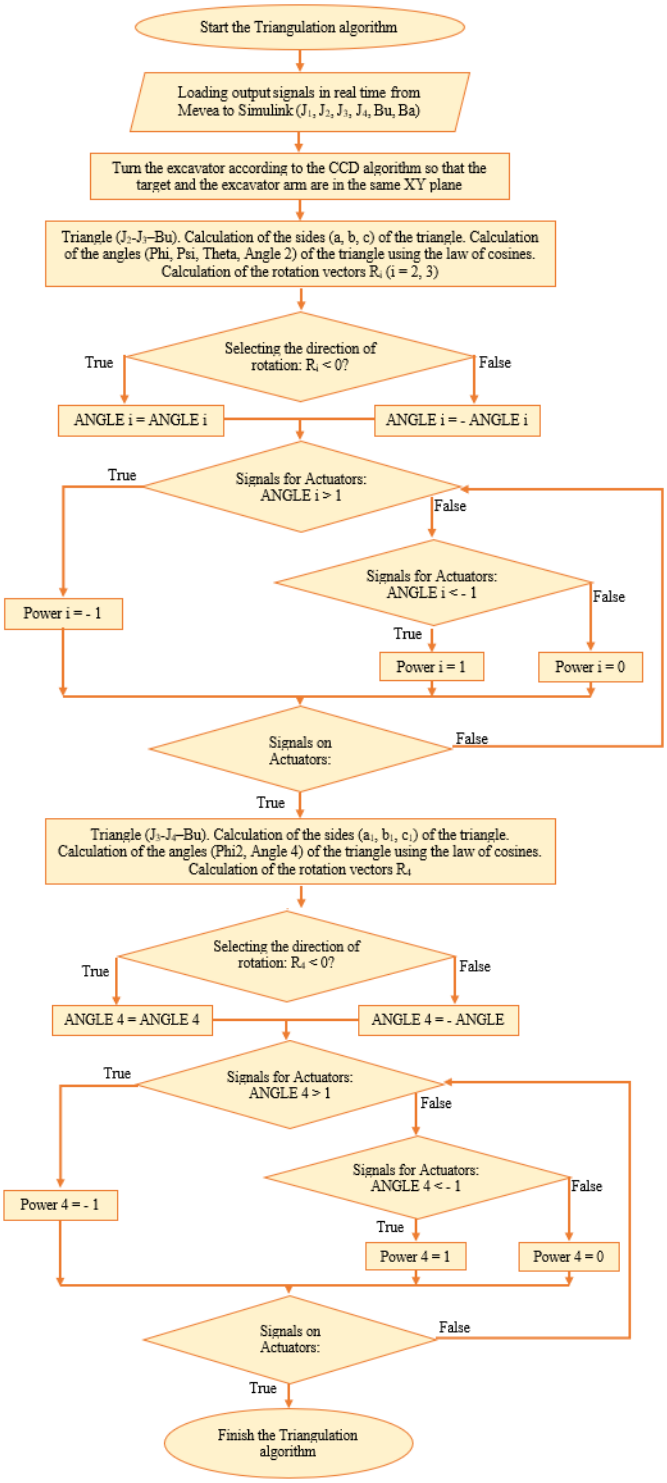


Figure 30. The algorithm of the Triangulation method implementation on the excavator model.

The source code of the Triangulation method is presented in Listing 2 of Appendix 1 of the master's thesis. Further, after creating a scheme in Simulink MATLAB (Figure 31), writing the code of the Triangulation method (Listing 2) and setting PID regulators (described above), an experiment was conducted according to Figure 22 and Table 8. This experiment allows taking it easy to measure the values of the three evaluation criteria of the method under study (path, accuracy, time) in all possible positions. As a result of applying the Target Triangle Method, or as it is called the Triangulation Method, the following values are obtained, presented in Table 11.

Table 11. Results of applying Triangulation Method.

1 ZONE							
UP				DOWN			
Reachable		Unreachable		Reachable		Unreachable	
PATH	4,946	PATH	4,976	PATH	6,298	PATH	5,605
DIST	0,100	DIST	2,540	DIST	0,100	DIST	2,422
TIME	5,200	TIME	6,850	TIME	6,600	TIME	6,800
2 ZONE							
UP				DOWN			
Reachable		Unreachable		Reachable		Unreachable	
PATH	15,158	PATH	18,385	PATH	15,465	PATH	18,643
DIST	0,100	DIST	1,621	DIST	0,100	DIST	1,491
TIME	9,400	TIME	13,850	TIME	11,650	TIME	13,200
3 ZONE							
UP				DOWN			
Reachable		Unreachable		Reachable		Unreachable	
PATH	14,286	PATH	17,568	PATH	15,573	PATH	17,831
DIST	0,100	DIST	1,622	DIST	0,100	DIST	1,491
TIME	9,150	TIME	13,050	TIME	12,000	TIME	12,850
4 ZONE							
UP				DOWN			
Reachable		Unreachable		Reachable		Unreachable	
PATH	3,962	PATH	4,023	PATH	5,407	PATH	4,430
DIST	0,100	DIST	2,541	DIST	0,100	DIST	2,420
TIME	4,800	TIME	6,200	TIME	6,500	TIME	5,850

As a result of applying the Triangulation method of the IK, the bucket of the excavator model has set in position of all the achievable target positions of the experiment. This indicates the correct application of the Triangulation method. The excavator path graphs of this method are shown in Appendix 2 in Figures 42-49. The final positions of the excavator after stopping the experiments are presented in Appendix 3 in Figures 66-73.

3.5. Applying the FABRIK Method

The FABRIK method of Inverse Kinematics is considered in theory to be the most accurate and simplest in terms of calculations. The essence of the FABRIK method is to calculate the joint positions of the kinematic chain of bodies in space and their angles of rotation of the kinematic chain links. This calculation is carried out by forward and reverse calculation of the positions of the chain of bodies, up to the moment when the final effector reaches the target position (enters the admissible target zone). This method is simple in that it does not require the calculation of rotation matrices. The calculation of the target positions of the kinematic chain of solids usually does not exceed more than ten iterations, which is an advantage of the FABRIK method. Figure 31 shows a schematic application of the FABRIK method in the Simulink MATLAB software. The schematic depicted is used to implement Inverse Kinematics for the excavator model.

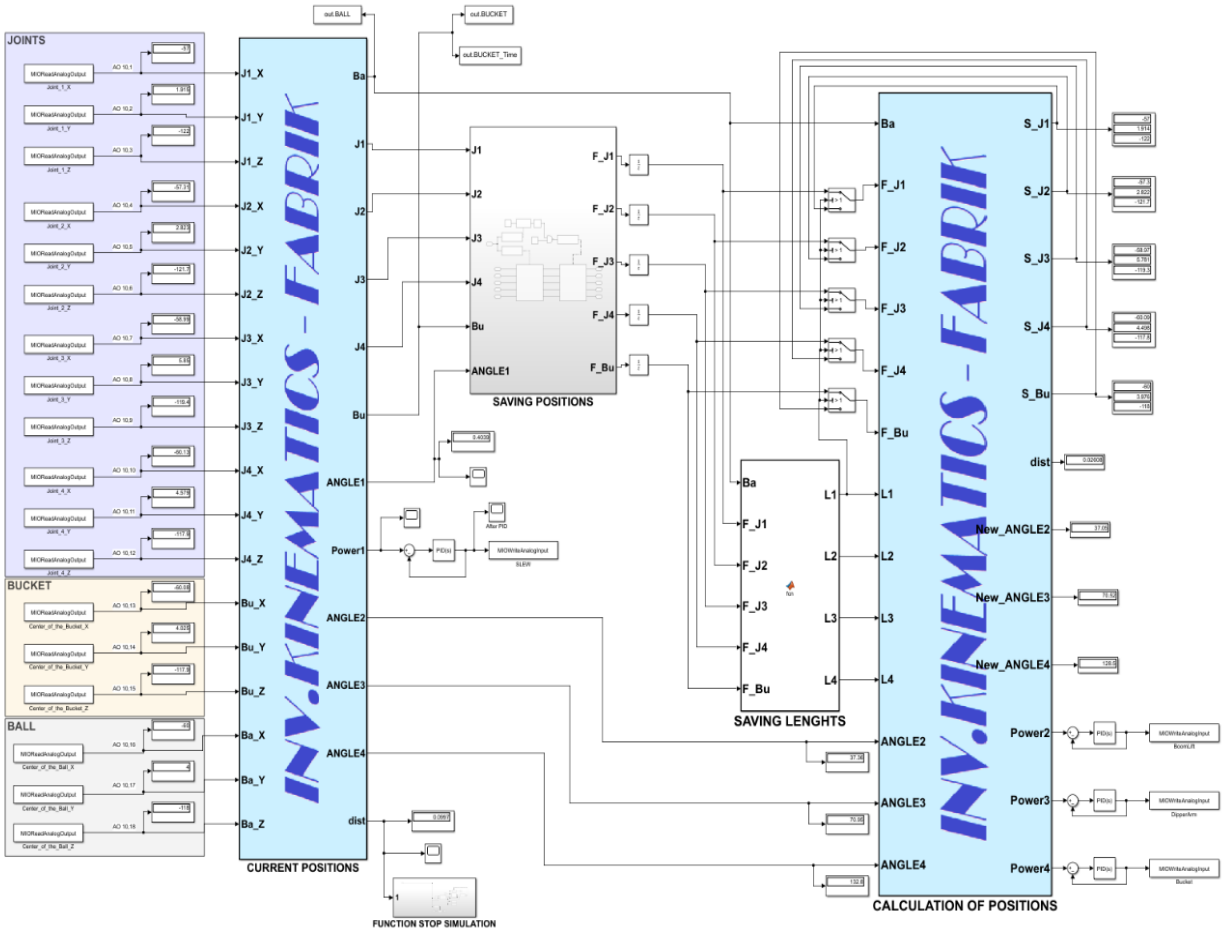


Figure 31. The schematic implementation of the FABRIK method.

Implementation of the FABRIK method in Simulink MATLAB software turned out to be the most complicated in this master's thesis. The application of the first two Inverse Kinematics methods to the excavator model was much easier. This is since the algorithm of the FABRIK method calculates the new positions of the kinematic chain (excavator arms) relative to the original positions of the joints. However, in this case the simulation and calculation processes are performed in real time. That is, new values (the current values of the excavator arm joints at the moment) are continuously coming to the calculation unit. Consequently, the calculation units are constantly updated, which leads to calculation failure.

Figure 31 shows that the FABRIK method scheme is constructed in a different way from the CCD and Triangulation methods. This method uses additional Simulink "Switch" blocks to update the results during computation. Figure 32 shows the "SAVING POSITIONS" subsystem, which saves the positions of the excavator arm after it is rotated in the vertical plane so that the excavator arm and the target (ball) are in the same plane. The saved positions from the subsystem are used to calculate new positions by which the excavator bucket reaches the target. The "SAVING LENGHTS" block is also used to calculate the lengths between the joints, which are then used to calculate new positions in the "CALCULATION POSITIONS" block.

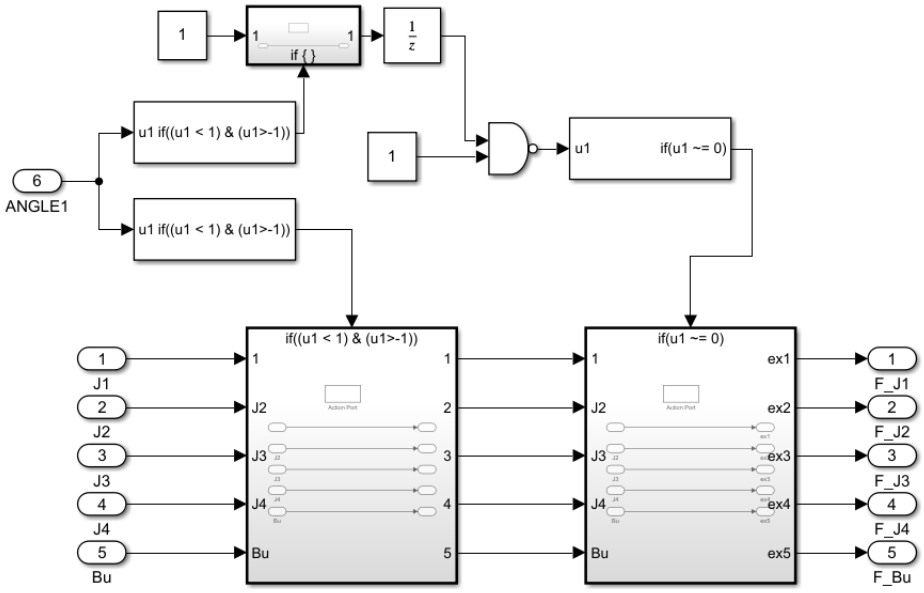


Figure 32. The Subsystem "SAVING POSITION".

To implement the Inverse kinematics of the FABRIK method, an algorithm was compiled based on the information the theoretical part. The formulas (41-43) of the investigated FABRIK method were used to compose the algorithm. Figure 33 shows the algorithm of the FABRIK method. Following this algorithm, codes were written for the blocks "CURRENT POSITIONS", "SAVING LENGHTS", "CALCULATION POSITION". The source codes of the FABRIK method are shown in Appendix 1, Listings 3, 4, and 5.

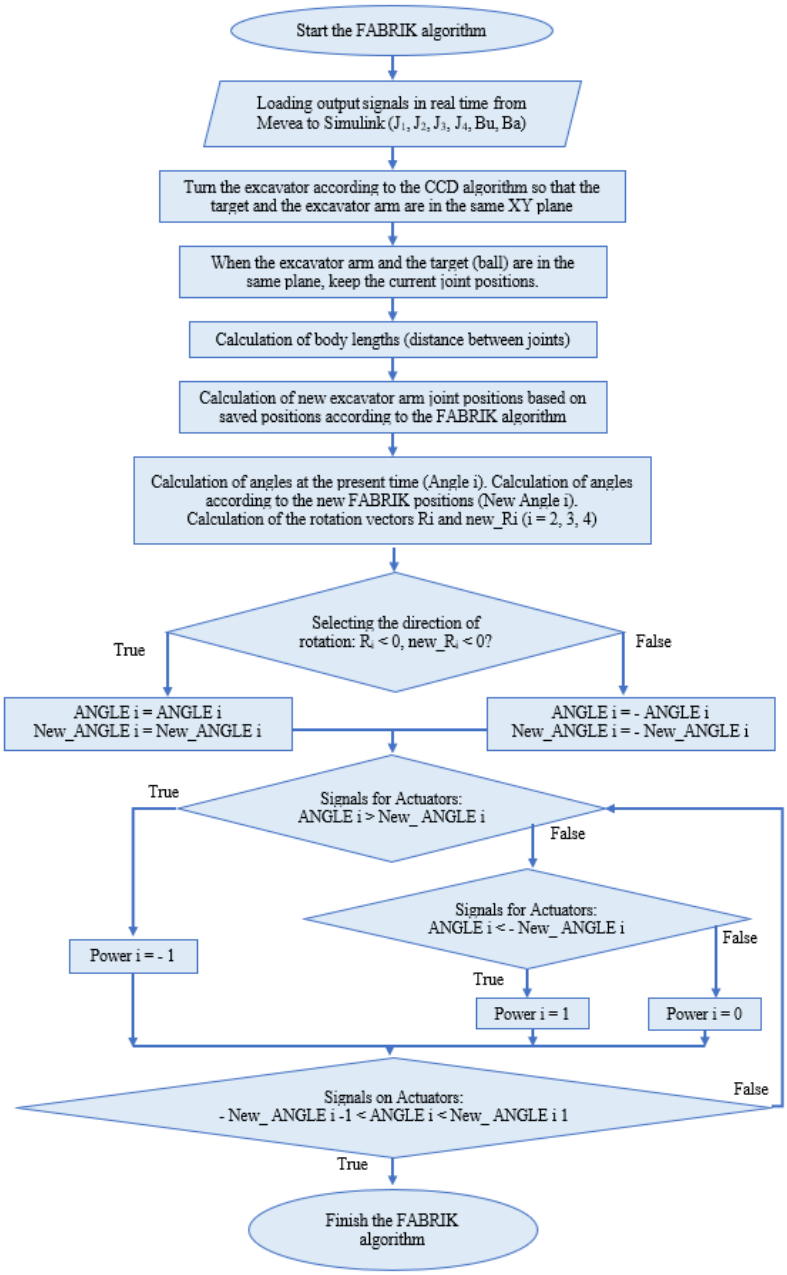


Figure 33. The algorithm of the FABRIK method implementation on the excavator model.

After compiling the Simulink scheme block codes, an experiment was conducted to evaluate the FABRIK method on the excavator model. The experiment was conducted according to Figure 22 and Table 9. The controller settings were similar to the first two IK methods. The measurement results of the FABRIK method experiment are presented in Table 12.

Table 12. Results of applying the FABRIK.

1 ZONE							
UP				DOWN			
Reachable		Unreachable		Reachable		Unreachable	
PATH	5,284	PATH	5,192	PATH	6,573	PATH	5,683
DIST	0,100	DIST	2,523	DIST	0,100	DIST	2,403
TIME	4,400	TIME	3,650	TIME	4,750	TIME	4,150
2 ZONE							
UP				DOWN			
Reachable		Unreachable		Reachable		Unreachable	
PATH	15,234	PATH	18,914	PATH	16,595	PATH	18,515
DIST	0,100	DIST	1,604	DIST	0,100	DIST	1,475
TIME	6,300	TIME	8,400	TIME	7,650	TIME	6,800
3 ZONE							
UP				DOWN			
Reachable		Unreachable		Reachable		Unreachable	
PATH	15,067	PATH	17,824	PATH	15,640	PATH	17,948
DIST	0,100	DIST	1,606	DIST	0,100	DIST	1,478
TIME	7,200	TIME	8,300	TIME	8,700	TIME	7,300
4 ZONE							
UP				DOWN			
Reachable		Unreachable		Reachable		Unreachable	
PATH	4,463	PATH	3,946	PATH	5,436	PATH	4,671
DIST	0,100	DIST	2,523	DIST	0,100	DIST	2,405
TIME	4,100	TIME	3,400	TIME	5,500	TIME	3,250

This table evaluates the FABRIK method according to three criteria: the path length of the excavator bucket, the distance from the end position of the bucket to the target position, and the time to stop the experiment The table shows that the bucket has reached all the positions in the area to be reached. This indicates that the FABRIK method code compiled works correctly. Also, during the experiment, the graphs of the excavator path of the Bucket in space were obtained. These graphs are presented in Appendix 2 in Figures 50-57. The final position of the excavator when reaching the target is presented in Appendix 3 in Figures 74-81.

4. Discussion

This chapter of the master's thesis summarizes and analyses the results of the work performed in the practical part. In this chapter, there is an analysis of the results and conclusions are derived.

4.1. About the Excavator model

The research used an existing model of a single-bucket excavator in Mevea software. One of the objectives of the master's thesis was to apply Inverse Kinematics method on this excavator model. However, in order to apply the IK, it is necessary to be familiar with and understand the basics of multibody systems with their dynamics. Based on the basic knowledge described in Chapter 2.1. "Multibody System", the excavator model was investigated and analysed for further implementation of IK. Due to the acquired knowledge of multibody systems, it took a short period of time to analyse the excavator model.

Excavator model data, specifically joint positions, were transferred from Mevea software to Simulink MATLAB. In the process, there were difficulties with the transferring target (ball) positions into Simulink. Many methods were used to create a ball as a body with fixation, graphics, but there were software failures. However, the results were inconvenient and led to failure as shown by the software. Thus, it was decided to use the ball as "Dummies" and set the target coordinates through Simulink. Simulink software was used to implement algorithms for Inverse Kinematics methods. Additionally, an algorithm was created to stop the experiment and load the experimental data, which simplified the research.

4.2. About Controllers settings

The use of PID controllers is an essential part of automatic control systems. Regulators are used to improving accuracy, reduce time, improve the quality of the transient process. In this master's thesis PID regulator blocks were used when sending the control signal to the actuator. Tuning was done using the built-in tuning function in Simulink. Only the integral (I) coefficients influenced the change in the transient curve. This can be explained by the fact that during automatic tuning, the Simulink software selects by the algorithm the optimal values of the coefficients so that the regulator works most effectively. Consequently, the regulation of the integral link is the most advantageous under the given conditions.

The obtained coefficients of the regulators were used in all three methods the same. This was in order to compare the methods of Inverse Kinematics in as much as possible same experimental conditions. It would be more correct to adjust the regulators for each method separately, but in this case, it needs the same conditions for all of them.

4.3. Comparison of three methods of Inverse Kinematics

The objective of the master's thesis is to determine the most appropriate Inverse Kinematics method for the existing excavator model. During the research all three Inverse Kinematics methods (CCD, Triangulation, FABRIK) were implemented on the excavator model using Simulink MATLAB software. Only one difficulty arose during the implementation of CCD and Triangulation methods. It consisted in choosing the rotation direction of joints when changing the target position relative to X and Z axes. However, this difficulty was solved. The most difficult in the realization of the IK for the excavator the model was the FABRIK method. The difficulty was saving the joint positions when the excavator was set in the vertical plane to calculate new joint positions. The solution is to use the "SAVING POSITION" subsystem and switch blocks to further calculate the FABRIK algorithm. This solution is shown in Figure 32.

As a result of the implementations of three IK methods, Tables 10-12 with the results of experiments were created. These tables have three criteria for evaluating the methods: the path, distance (accuracy), and time. Based on these tables with three criteria, a table was created which compares all three methods according to the criteria. The results of the comparison of the CCD, Triangulation and FABRIK methods are presented below in Table 13. This table illustrates the best methods for each experimental position and for each criterion. The name of the best method is recorded in a cell. A value of 0.1 in the cells of the table means that the goal was achieved by the bucket of the excavator model with each method, hence all methods are useful in this case. This table is one of the most significant in the entire master's thesis.

Table 13. Comparison of three Inverse Kinematics methods.

Position			Path	Distance	Time
1 Zone	UP	Reachable	CCD	0,1	CCD
		Unreachable	CCD	CCD	CCD
	DOWN	Reachable	CCD	0,1	FABRIK
		Unreachable	CCD	CCD	CCD
2 Zone	UP	Reachable	TRIANGLE	0,1	CCD
		Unreachable	TRIANGLE	CCD	CCD
	DOWN	Reachable	TRIANGLE	0,1	CCD
		Unreachable	FABRIK	CCD	CCD
3 Zone	UP	Reachable	TRIANGLE	0,1	CCD
		Unreachable	TRIANGLE	CCD	CCD
	DOWN	Reachable	TRIANGLE	0,1	CCD
		Unreachable	TRIANGLE	CCD	CCD
4 Zone	UP	Reachable	CCD	0,1	CCD
		Unreachable	CCD	CCD	CCD
	DOWN	Reachable	CCD	0,1	CCD
		Unreachable	CCD	CCD	CCD

According to the results of the table, the most preferred method of Inverse Kinematics is the CCD method. This method has the best results according to accuracy and time criteria. According to the excavator bucket path criterion, the CCD method shares first place with the Triangulation method. The excavator bucket path is much shorter with the Triangulation method when changing target positions relative to the X and Z axes. The

reason for this is that the CCD method in this case is a bit confusing to calculate until the bucket is on the same side as the target (ball) relative to the excavator itself on the X and Z axes. This feature of the CCD method is demonstrated in the path graphs in Appendix 2 in Figures 36-39.

However, there is another variant of the most preferred Inverse Kinematics method for the available excavator model, which does not correspond to Table 13. It can be assumed that the best inverse kinematics method for the available excavator model is the Triangulation method. The following arguments are given as proof:

- The Triangulation method path looks more adequate relative to the CCD method. This feature can be seen in the graphs in Appendix 2, Figures 42-49.
- The movement of the excavator arm is smoother and more natural than in the CCD and FABRIK methods. Appendix 3, Figures 66-73.
- Experimental results (path, accuracy, time) are not very different from the CCD method.

Based on the above facts, it can be concluded that the Triangulation method is not inferior to the CCD method, and in some parameters, it is even superior. Consequently, the Triangulation method is the most preferable for the available excavator model.

4.4. Limitations

There are some limitations in this study that may influence the choice of the best IK method. Firstly, only one excavator model was used, though it would have been better to conduct the study on several earthmoving machine models. Ideally, it would have been a good idea to apply IK to a real machine. This deepening is a significant basis for further scientific research. Secondly, the implementation of IK algorithms on other software, in other programming languages, which are more flexible in contrast to Simulink. Despite the described limitations, the research of the master's thesis is completed and contains the answer to the goals and objectives.

5. Conclusions

This chapter describes the conclusions of the master's thesis.

5.1. Summary of the research

Earthmoving machines are an integral part of our lives. Without earthmoving machines, life today is unimaginable. This construction equipment performs a huge range of tasks that are usually beyond the reach of man. The potential of this technology is enormous. At present, the development of earthmoving machinery is aimed at increasing the efficiency of work, through robotization and automated control.

During the performance of the master's thesis, the work of Inverse Kinematics on the example of the excavator model in MEVEA Software was investigated. On the example of the available model were carried out all the studies that were interested in this topic of work.

The Methods chapter focused on the basics needed to work with the excavator model, implement Inverse Kinematics, and adjust PID controllers. In this chapter, multibody systems are analysed to understand the basics of mechanical systems, kinematic and dynamic analysis, which are applied to the excavator model. Also in the methods chapter, three popular Methods of Inverse Kinematics - the Cyclic Coordinate Descent Method, the Triangulation Method, the FABRIK Method were analysed. The basic principles of these methods and algorithms for their implementation were studied. Also, the work of the PID regulator, and ways of adjustment, for controlling actuators on the available model of the excavator were studied.

In the practical part, the three methods of Inverse Kinematics were implemented on the MEVEA excavator model using the Simulink MATLAB software. The implementation was performed in accordance with the experiment plan and the results were measured in accordance with the table of target positions. The experiment included all possible positions of the excavator in space. The results of three IK methods were presented in the tables. The evaluation of the applied methods was based on such parameters as: the path of the excavator bucket, the distance from the end position to the target position, and the time to stop the experiment. Based on the results of the experiment and the graphs of the excavator bucket path, it was found that the best Inverse Kinematics method is the Triangulation Method.

5.2. Future work

This work has high potential in the field of digital twins. Also, this research is a huge contribution in automation area of construction machinery and minimization of human involvement in the digging process. Future research will focus on improving existing Inverse Kinematics methods and algorithms, as well as on problems such as automatic digging (extraction) of a certain amount of material. The research is performed in a narrowed application for a specific object. Future research should be directed to a broader application of Inverse Kinematics in earthmoving machinery. Future research should also be directed towards calculating the effectiveness of Inverse Kinematics on the process and business. The potential for research is very high.

References

A. Gurko, & I. Kolobova. 2013. Simulation of excavator kinematics in MATLAB robotics toolbox. Vestnik KNADU, Pp. 59–65.

Aamirberg. 2020. Construction Equipment Rental Market to Witness Massive Growth. Boels Rentals, H&E Equipment Services, Herc Rentals Inc. 2020, October 13. [Referred 20.5.2021]. Available: https://ipsnews.net/business/2020/10/13/construction-equipment-rental-market-to-witness-massive-growth-boels-rentals-he-equipment-services-herc-rentals-inc/

Aristidou, A., Chrysanthou, Y., & Lasenby, J. 2016. Extending FABRIK with model constraints. Computer Animation and Virtual Worlds, 27, Pp. 35–57.

Aristidou, A., & Lasenby, J. 2011. FABRIK: A fast, iterative solver for the Inverse Kinematics problem. Graphical Models, 73(5), Pp. 243–260.

Aristidou, A., Lasenby, J., Chrysanthou, Y., & Shamir, A. 2018. Inverse Kinematics Techniques in Computer Graphics: A Survey: Inverse Kinematics Techniques in Computer Graphics. Computer Graphics Forum, 37(6), Pp. 35–58.

Çinar, S. M., Balci, Z., & Yabanova, İ. 2019. DC Motorun Otomatik Ayarlamalı PID ile Hız Kontrolünün Gerçekleştirilmesi. Afyon Kocatepe University Journal of Sciences and Engineering, 19(3), Pp. 690–696.

Danilov, A. V., Kropotov, A. N., & Trifonov, O. V. 2018. General approach to solving the inverse problem of kinematics for a manipulator of a sequential structure by means of a finite rotation and displacement. Keldysh Institute Preprints. Pp. 1–15.

Daraz, A., Malik, S. A., Saleem, T., & Bhati, S. A. 2017. Ziegler Nichols Based Integral Proportional Controller for Superheated Steam Temperature Control System. 11(5), Pp. 516-520.

Flores, P. 2015a. Concepts and Formulations for Spatial Multibody Dynamics. Springer International Publishing. Pp. 10-70.

Jaiswal, S., Korkealaakso, P., Aman, R., Sopanen, J., & Mikkola, A. 2019. Deformable Terrain Model for the Real-Time Multibody Simulation of a Tractor with a Hydraulically Driven Front-Loader. Pp. 172694-172697.

Karginov, L. 2016. Hierarchical Approach to Solving an Inverse Problem of Kinematics. Science and Education of the Bauman MSTU, 16(03). Pp. 37-40.

Kenwright, B. 2012. Inverse Kinematics – Cyclic Coordinate Descent (CCD). Journal of Graphics Tools, 16(4), Pp. 177–217.

Khadim, Q., Kiani, M., Jaiswal, S., Matikainen, M., & Mikkola, A. 2021. Estimating the Characteristic Curve of a Directional Control Valve in a Combined Multibody and Hydraulic System Using an Augmented Discrete Extended Kalman Filter. Sensors, 21, 5029. Pp. 2-9.

Mehta, N., Chauhan, D., Patel, S., & Mistry, S. 2017. Design of HMI Based on PID Control of Temperature. International Journal of Engineering Research And, V6. Pp. 117-120.

Mukundan, R. 2009. A robust inverse kinematics algorithm for animating a joint chain. International Journal of Computer Applications in Technology, 34(4), Pp. 300-308.

Muller-Cajar, R., & Mukundan, R. 2007. Triangulation: A new algorithm for Inverse Kinematics. Pp. 1-6.

Munshi, S. M. 2012. Analysis, Investigation and Design of Flexible Universal Pneumatic Industrial Manipulators Involving Cartesian and Joint Control in the Basis of Economic Feasibility and Appropriate Technology. Pp 46-47.

Mustafa, M. N., & Nsaif, M. 2019. DC Servo Motor Speed Control based on Lab-View. Pp. 9-11.

Nikravesh, P. E. 1988. Vector of Coordinates. In: Basic Concepts and Numerical Methods in Kinematics. Computer-Aided Analysis of Mechanical Systems. Englewood Cliffs, NJ. Prentice Hall. January 1988. Pp. 35-40.

Santos, M. C., Molina, L., Carvalho, E. A. N., Freire, E. O., Carvalho, J. G. N., & Santos, P. C. 2021. FABRIK-R: An Extension Developed Based on FABRIK for Robotics Manipulators. Pp. 53423–53435.

Song, W., & Hu, G. 2011. A Fast Inverse Kinematics Algorithm for Joint Animation. Procedia Engineering. 24, Pp. 350–354.

Strashnov, E. V., & Mikhailyuk, M. V. 2017. Simulation of semi-automatic control of manipulator robots in virtual environment systems. Proceedings in Cybernetics, 0(4), Pp. 189–196.

Syzrantsev, V. N. 2016. Determination of the instantaneous gear transmission ratio in spatial gears. Oil and Gas Studies, 6, Pp. 122–129.

Tao, S., Tao, H., & Yang, Y. 2021. Extending FABRIK with Obstacle Avoidance for Solving the Inverse Kinematics Problem. Journal of Robotics 2021, Pp. 1–10.

Zhang, B., Wang, S., Liu, Y., & Yang, H. 2017. Research on Trajectory Planning and Autodig of Hydraulic Excavator. Mathematical Problems in Engineering. 2017, Pp. 1–10.

Appendix 1: Codes of Inverse Kinematics algorithms

This appendix contains the source codes of the Inverse Kinematics algorithms. The codes were written in the MATLAB Function blocks in the Simulink software.

Listing 1: CCD algorithm code in the "Inverse Kinematics CCD" block

```
function [Power1,ANGLE1,Power2,ANGLE2,Power3,ANGLE3,...
    Power4,ANGLE4,Bu,Ba,dist] = fcn(J1_X,J1_Y,J1_Z,...
    J2_X,J2_Y,J2_Z,J3_X,J3_Y,J3_Z,J4_X,J4_Y,J4_Z,...
    Bu_X,Bu_Y,Bu_Z,Ba_X,Ba_Y,Ba_Z)
J1 = [J1_X; J1_Y; J1_Z]; J2 = [J2_X; J2_Y; J2_Z]; J3 = [J3_X; J3_Y; J3_Z];
J4 = [J4_X; J4_Y; J4_Z]; Bu = [Bu_X; Bu_Y; Bu_Z]; Ba = [Ba_X; Ba_Y; Ba_Z];
Jx = [1; 0; 0]; Jy = [0; 1; 0]; Jz = [0; 0; 1];
%% 1 joint
V1 = Bu - J1; % Vector from Joint1 to Bucket
W1 = Ba - J1; % Vector from Joint1 to Ball
V1p = V1.*Jx + V1.*Jz; % The projection of V1 on the XZ plane
W1p = W1.*Jx + W1.*Jz; % The projection of W1 on the XZ plane
% The angle of rotation between vectors V1 and W1
ANGLE1 = acosd(dot((V1p/norm(V1p)),(W1p/norm(W1p))));
% Direction of rotation vector R1
R1 = cross([V1p(1); 0; V1p(3)],[W1p(1); 0; W1p(3)]);
% Angle change when rotating backwards
 if R1(2) < 0
   ANGLE1 = -ANGLE1; %
 end
% Signal to the actuator 1 (SLEW)
if   ANGLE1 > 1
    Power1 = -1;
elseif  ANGLE1 < -1
    Power1 = 1;
else
    Power1 = 0;
end
%% 2 joint
V2 = Bu - J2; % Vector from Joint2 to Bucket
W2 = Ba - J2; % Vector from Joint2 to Ball
V2p = V2.*Jx + V2.*Jy; % The projection of V2 on the XY plane
W2p = W2.*Jx + W2.*Jy; % The projection of W2 on the XY plane
% The angle of rotation between vectors V2 and W2
ANGLE2 = acosd(dot((V2p/norm(V2p)),(W2p/norm(W2p))));
% Direction of rotation vector R2
R2 = cross(([V2p(1); V2p(2); 0]),([W2p(1); W2p(2); 0]));
% Angle change when rotating backwards
if (R2(3) < 0) && (Ba_X > J2_X)
    ANGLE2 = -ANGLE2;
end
if (R2(3) > 0) && (Ba_X < J2_X)
    ANGLE2 = -ANGLE2;
end
% Signal to the actuator 2 (BoomLift)
if  ANGLE2 > 1
    Power2 = -0.8;
elseif  ANGLE2 <-1
    Power2 = 0.8;
else
    Power2 = 0.1;
end
%%  3 joint
V3 = Bu - J3; % Vector from Joint3 to Bucket
W3 = Ba - J3; % Vector from Joint3 to Ball
V3p = V3.*Jx + V3.*Jy; % The projection of V3 on the XY plane
W3p = W3.*Jx + W3.*Jy; % The projection of W3 on the XY plane
% The angle of rotation between vectors V3 and W3
ANGLE3 = acosd(dot((V3p/norm(V3p)),(W3p/(norm(W3p)))));
% Direction of rotation vector R3
R3 = cross([V3p(1); V3p(2); 0],[W3p(1); W3p(2); 0]);
% Angle change when rotating backwards
```

```
if (R3(3) < 0) && (Ba_X > J3_X)
    ANGLE3 = -ANGLE3;
end
if (R3(3) > 0) && (Ba_X < J3_X)
    ANGLE3 = -ANGLE3;
end
% Signal to the actuator 3 (DipperArm)
if  (ANGLE3 >1)
   Power3 = 0.8;
elseif  (ANGLE3 < -1)
    Power3 = -0.8;
else
    Power3 = 0;
end
%% 4 joint
V4 = Bu - J4; % Vector from Joint4 to Bucket
W4 = Ba - J4; % Vector from Joint4 to Ball
V4p = V4.*Jx + V4.*Jy; % The projection of V4 on the XY plane
W4p = W4.*Jx + W4.*Jy; % The projection of W4 on the XY plane
% The angle of rotation between vectors V4 and W4
ANGLE4 = acosd(dot((V4p/norm(V4p)),(W4p/(norm(W4p)))));
% Direction of rotation vector R4
R4 = cross([V4p(1); V4p(2); 0],[W4p(1); W4p(2); 0]);
% Angle change when rotating backwards
if (R4(3) < 0) && (Ba_X > J4_X)
    ANGLE4 = -ANGLE4;
end
if (R4(3) > 0) && (Ba_X < J4_X)
    ANGLE4 = -ANGLE4;
end
% Signal to the actuator 4 (Bucket)
if  ANGLE4 > 1
   Power4 = 0.8;
elseif  ANGLE4 < 1
    Power4 = -0.8;
else
    Power4 = 0;
end
%% Distance between Bucket and Ball
dist = sqrt((Ba(1)-Bu(1))^2+(Ba(2)-Bu(2))^2+(Ba(3)-Bu(3))^2);

end
```

Listing 2: Triangulation algorithm code in the "Inv Kin_Target Triangle" block

```
function [Power1,ANGLE1,Power2,ANGLE2,...
    Power3,Theta,Phi,Power4,Bu,Ba,dist]= ...
    fcn(J1_X,J1_Y,J1_Z,J2_X,J2_Y,J2_Z,J3_X,J3_Y,J3_Z,J4_X,J4_Y,J4_Z,...
    Bu_X,Bu_Y,Bu_Z,Ba_X,Ba_Y,Ba_Z)
J1 = [J1_X; J1_Y; J1_Z]; J2 = [J2_X; J2_Y; J2_Z]; J3 = [J3_X; J3_Y; J3_Z];
J4 = [J4_X; J4_Y; J4_Z]; Bu = [Bu_X; Bu_Y; Bu_Z]; Ba = [Ba_X; Ba_Y; Ba_Z];
Jx = [1; 0; 0]; Jy = [0; 1; 0]; Jz = [0; 0; 1];
%% 1 joint
V1 = Bu - J1; % Vector from Joint1 to Bucket
W1 = Ba - J1; % Vector from Joint1 to Ball
V1p = V1.*Jx + V1.*Jz; % The projection of V1 on the XZ plane
W1p = W1.*Jx + W1.*Jz; % The projection of W1 on the XZ plane
% The angle of rotation between vectors V1 and W1
ANGLE1 = acosd(dot((V1p/norm(V1p)),(W1p/norm(W1p))));
% Direction of rotation vector R1
R1 = cross([V1p(1); 0; V1p(3)],[W1p(1); 0; W1p(3)]);
% Angle change when rotating backwards
 if R1(2) < 0
   ANGLE1 = -ANGLE1;
 end
% Signal to the actuator 1 (SLEW)
if   ANGLE1 > 1
   Power1 = -0.8;
elseif  ANGLE1 < -1
    Power1 = 0.8;
else
    Power1 = 0;
```

```
end
%% TRIANGLE J2 - J3 - Bu
% The sides of the triangle a=J2-J3, b=J3-Bu, c=Bu-J2
a = sqrt((J3(1)-J2(1))^2+(J3(2)-J2(2))^2+(J3(3)-J2(3))^2);
b = sqrt((Bu(1)-J3(1))^2+(Bu(2)-J3(2))^2+(Bu(3)-J3(3))^2);
c = sqrt((J2(1)-Ba(1))^2+(J2(2)-Ba(2))^2+(J2(3)-Ba(3))^2);

V2 = Bu - J2; % Vector from Joint2 to Bucket
W2 = Ba - J2; % Vector from Joint2 to Ball
V2p = V2.*Jx + V2.*Jy; % The projection of V2 on the XY plane
W2p = W2.*Jx + W2.*Jy; % The projection of W2 on the XY plane
% Direction of rotation vector R2
R2 = cross([V2p(1); V2p(2); 0],[W2p(1); W2p(2); 0]);
V3_2 = J2 - J3; % Vector from Joint3 to Joint2
W3_2 = Bu - J3; % Vector from Joint3 to Bucket
Q3_2 = Ba - J3; % Vector from Joint3 to Ball
V3_2p = V3_2.*Jx + V3_2.*Jy; % The projection of V3_2 on the XY plane
W3_2p = W3_2.*Jx + W3_2.*Jy; % The projection of W3_2 on the XY plane
Q3_2p = Q3_2.*Jx + Q3_2.*Jy; % The projection of Q3_2 on the XY plane
% The angle of rotation between vectors V3_2 and W3_2
ANGLE2 = acosd(dot((V3_2p/norm(V3_2p)),(W3_2p/(norm(W3_2p)))));
% Direction of rotation vector R3
R3 = cross([W3_2p(1); W3_2p(2); 0],[Q3_2p(1); Q3_2p(2); 0]);
% Calculation of the angles of the triangle J2 - J3 - Bu
Phi = acosd(complex(-(c^2-a^2-b^2)/(2*a*b))); % Angle between a and b
Psi = acosd(complex(-(b^2-a^2-c^2)/(2*a*c))); % Angle between a and c
% The angle of rotation between vectors V2p and W2p
Theta = acosd(dot((V2p/norm(V2p)),(W2p/(norm(W2p)))));
% Angle change when rotating backwards
if (R2(3) < 0) && (Ba_X > J2_X)
    Theta = -Theta;
end
if (R2(3) > 0) && (Ba_X < J2_X)
    Theta = -Theta;
end
% Signal to the actuator 2 (BoomLift)
 if  (Theta > 1) && (ANGLE1 < 1) &&(ANGLE1 > -1)
    Power2 = -0.8;
 elseif  (Theta < -1) && (ANGLE1 < 1) && (ANGLE1 > -1)
     Power2 = 0.8;
 else
     Power2 = 0;
 end
% Angle change when rotating backwards
if (R3(3) < 0) && (Ba_X > J3_X)
    ANGLE2 = -ANGLE2; Phi = -Phi;
end
if (R3(3) > 0) && (Ba_X < J3_X)
    ANGLE2 = -ANGLE2; Phi = -Phi;
end
% Signal to the actuator 3 (DipperArm)
if  (ANGLE2 < Phi)  && (ANGLE1 < 1) && (ANGLE1 > -1)
   Power3 = 0.8;
 elseif  (ANGLE2 > Phi) && (ANGLE1 < 1) && (ANGLE1 > -1)
    Power3 = -0.8;
else
    Power3 = 0;
end
%% TRIANGLE J3 - J4 - Bu
% The sides of the triangle a=J3-J4, b=J4-Bu, c=Bu-J3
a1 = sqrt((J4(1)-J3(1))^2+(J4(2)-J3(2))^2+(J4(3)-J3(3))^2);
b1 = sqrt((Bu(1)-J4(1))^2+(Bu(2)-J4(2))^2+(Bu(3)-J4(3))^2);
c1 = sqrt((J3(1)-Ba(1))^2+(J3(2)-Ba(2))^2+(J3(3)-Ba(3))^2);

V4_2 = J3 - J4; % Vector from Joint4 to Joint3
W4_2 = Bu - J4; % Vector from Joint4 to Bucket
Q4_2 = Ba - J4; % Vector from Joint4 to Ball
V4_2p = V4_2.*Jx + V4_2.*Jy; % The projection of V4_2 on the XY plane
W4_2p = W4_2.*Jx + W4_2.*Jy; % The projection of W4_2 on the XY plane
Q4_2p = Q4_2.*Jx + Q4_2.*Jy; % The projection of Q4_2 on the XY plane
% The angle of rotation between vectors V4_2 and W4_2
ANGLE4 = acosd(dot((V4_2p/norm(V4_2p)),(W4_2p/(norm(W4_2p)))));
% Direction of rotation vector R4
R4 = cross([W4_2p(1); W4_2p(2); 0],[Q4_2p(1); Q4_2p(2); 0]);
Phi2 = acosd(complex(-(c1^2-a1^2-b1^2)/(2*a1*b1))); % Angle between a1 and b1
```

```
% Angle change when rotating backwards
if (R4(3) < 0) && (Ba_X > J4_X)
    ANGLE4 = -ANGLE4;
end
if (R4(3) > 0) && (Ba_X < J4_X)
    ANGLE4 = -ANGLE4;
end
% Signal to the actuator 4 (Bucket)
if  (ANGLE4 < Phi2)  && (ANGLE1 < 1) && (ANGLE1 > -1)
   Power4 = 0.8;
 elseif  (ANGLE4 > Phi2) && (ANGLE1 < 1) && (ANGLE1 > -1)
    Power4 = -0.8;
else
    Power4 = 0;
end
%% Distance between Bucket and Ball
dist = sqrt((Ba(1)-Bu(1))^2+(Ba(2)-Bu(2))^2+(Ba(3)-Bu(3))^2);
end
```

Listing 3: FABRIK algorithm code in the "CURRENT POSITIONS" block

```
function [Ba,J1,J2,J3,J4,Bu,ANGLE1,Power1,ANGLE2,ANGLE3,ANGLE4,dist]...
    = fcn(J1_X,J1_Y,J1_Z,J2_X,J2_Y,J2_Z,J3_X,J3_Y,J3_Z,...
    J4_X,J4_Y,J4_Z,Bu_X,Bu_Y,Bu_Z,Ba_X,Ba_Y,Ba_Z)
J1 = [J1_X; J1_Y; J1_Z]; J2 = [J2_X; J2_Y; J2_Z]; J3 = [J3_X; J3_Y; J3_Z];
J4 = [J4_X; J4_Y; J4_Z]; Bu = [Bu_X; Bu_Y; Bu_Z]; Ba = [Ba_X; Ba_Y; Ba_Z];
Jx = [1; 0; 0]; Jy = [0; 1; 0]; Jz = [0; 0; 1];
%% 1 joint
V1 = Bu - J1; % Vector from Joint1 to Bucket
W1 = Ba - J1; % Vector from Joint1 to Ball
V1p = V1.*Jx + V1.*Jz; % The projection of V1 on the XZ plane
W1p = W1.*Jx + W1.*Jz; % The projection of W1 on the XZ plane
% The angle of rotation between vectors V1 and W1
ANGLE1 = acosd(dot((V1p/norm(V1p)),(W1p/norm(W1p))));
% Direction of rotation vector R1
R1 = cross([V1p(1); 0; V1p(3)],[W1p(1); 0; W1p(3)]);
% Angle change when rotating backwards
 if R1(2) < 0
   ANGLE1 = -ANGLE1;
 end
% Signal to the actuator 1 (SLEW)
if   ANGLE1 > 1
   Power1 = -0.8;
elseif ANGLE1 < -1
    Power1 = 0.8;
else
    Power1 = 0;
end
%% ANGLES 2, 3, 4
% Calculations as in the CCD method.
V2 = Bu - J2; % Vector from Joint2 to Bucket
W2 = Ba - J2; % Vector from Joint2 to Ball
V2p = V2.*Jx + V2.*Jy; % The projection of V2 on the XY plane
W2p = W2.*Jx + W2.*Jy; % The projection of W2 on the XY plane
% The angle of rotation between vectors V2 and W2
ANGLE2 = acosd(dot((V2p/norm(V2p)),(W2p/norm(W2p))));
% Direction of rotation vector R2
R2 = cross(([V2p(1); V2p(2); 0]),([W2p(1); W2p(2); 0]));
% Angle change when rotating backwards
if (R2(3) < 0) && (Ba_X > J2_X)
    ANGLE2 = -ANGLE2;
end
if (R2(3) > 0) && (Ba_X < J2_X)
    ANGLE2 = -ANGLE2;
end
V3 = Bu - J3; % Vector from Joint3 to Bucket
W3 = Ba - J3; % Vector from Joint3 to Ball
V3p = V3.*Jx + V3.*Jy; % The projection of V3 on the XY plane
W3p = W3.*Jx + W3.*Jy; % The projection of W3 on the XY plane
% The angle of rotation between vectors V3 and W3
ANGLE3 = acosd(dot((V3p/norm(V3p)),(W3p/(norm(W3p)))));
% Direction of rotation vector R3
```

```
R3 = cross([V3p(1); V3p(2); 0],[W3p(1); W3p(2); 0]);
% Angle change when rotating backwards
if (R3(3) < 0) && (Ba_X > J3_X)
    ANGLE3 = -ANGLE3;
end
if (R3(3) > 0) && (Ba_X < J3_X)
    ANGLE3 = -ANGLE3;
end
V4 = Bu - J4; % Vector from Joint4 to Bucket
W4 = Ba - J4; % Vector from Joint4 to Ball
V4p = V4.*Jx + V4.*Jy; % The projection of V4 on the XY plane
W4p = W4.*Jx + W4.*Jy; % The projection of W4 on the XY plane
% The angle of rotation between vectors V4 and W4
ANGLE4 = acosd(dot((V4p/norm(V4p)),(W4p/(norm(W4p)))));
% Direction of rotation vector R4
R4 = cross([V4p(1); V4p(2); 0],[W4p(1); W4p(2); 0]);
% Angle change when rotating backwards
if (R4(3) < 0) && (Ba_X > J4_X)
    ANGLE4 = -ANGLE4;
end
if (R4(3) > 0) && (Ba_X < J4_X)
    ANGLE4 = -ANGLE4;
end
%% Distance between Bucket and Ball
dist = sqrt((Ba(1)-Bu(1))^2+(Ba(2)-Bu(2))^2+(Ba(3)-Bu(3))^2);
end
```

Listing 4: FABRIK algorithm code in the "SAVING LENGHTS" block

```
function [L1,L2,L3,L4]= fcn(Ba,F_J1,F_J2,F_J3,F_J4,F_Bu)
S_J1 = F_J1; S_J2 = F_J2; S_J3 = F_J3; S_J4 = F_J4; S_Bu = F_Bu
L1 = sqrt((Ba(1)-S_J2(1))^2+(Ba(2)-S_J2(2))^2+(Ba(3)-S_J2(3))^2);
L2 = sqrt((S_J3(1)-S_J2(1))^2+(S_J3(2)-S_J2(2))^2+(S_J3(3)-S_J2(3))^2);
L3 = sqrt((S_J4(1)-S_J3(1))^2+(S_J4(2)-S_J3(2))^2+(S_J4(3)-S_J3(3))^2);
L4 = sqrt((S_Bu(1)-S_J4(1))^2+(S_Bu(2)-S_J4(2))^2+(S_Bu(3)-S_J4(3))^2);
end
```

Listing 5: FABRIK algorithm code in the "CALCULATION OF POSITIONS" block

```
function [S_J1,S_J2,S_J3,S_J4,S_Bu,dist,...
    New_ANGLE2,New_ANGLE3,New_ANGLE4,Power2,Power3,Power4]...
    = fcn(


function [S_J1,S_J2,S_J3,S_J4,S_Bu,dist,...
    New_ANGLE2,New_ANGLE3,New_ANGLE4,Power2,Power3,Power4]...
    = fcn(Ba,F_J1,F_J2,F_J3,F_J4,F_Bu,L1,L2,L3,L4,ANGLE2,ANGLE3,ANGLE4)
Jx = [1; 0; 0]; Jy = [0; 1; 0];
S_J1 = F_J1; S_J2 = F_J2; S_J3 = F_J3; S_J4 = F_J4; S_Bu = F_Bu;
%% FABRIK
tol = 0.01; % Tolerance, accuracy
% Distance between new position of the Bucket and Ball
dist = sqrt((Ba(1)-S_Bu(1))^2+(Ba(2)-S_Bu(2))^2+(Ba(3)-S_Bu(3))^2);
% Implementation of the FABRIK algorithm
for N=1:100
    if (L1 < L2+L3+L4) && (dist > tol)
%% REACHABLE
%BACKWARD
S_Bu = Ba;
r4 = sqrt((S_Bu(1)-S_J4(1))^2+(S_Bu(2)-S_J4(2))^2+(S_Bu(3)-S_J4(3))^2);
Lam4 = L4/r4;
S_J4 = (1-Lam4)*S_Bu + Lam4.*S_J4;
r3 = sqrt((S_J4(1)-S_J3(1))^2+(S_J4(2)-S_J3(2))^2+(S_J4(3)-S_J3(3))^2);
Lam3 = L3/r3;
S_J3 = (1-Lam3)*S_J4 + Lam3.*S_J3;
% FORWARD
r2 = sqrt((S_J3(1)-S_J2(1))^2+(S_J3(2)-S_J2(2))^2+(S_J3(3)-S_J2(3))^2);
Lam2 = L2/r2;
S_J3 = (1-Lam2)*S_J2 + Lam2.*S_J3;
```

```
r3 = sqrt((S_J4(1)-S_J3(1))^2+(S_J4(2)-S_J3(2))^2+(S_J4(3)-S_J3(3))^2);
Lam3 = L3/r3;
S_J4 = (1-Lam3)*S_J3 + Lam2.*S_J4;
r4 = sqrt((S_Bu(1)-S_J4(1))^2+(S_Bu(2)-S_J4(2))^2+(S_Bu(3)-S_J4(3))^2);
Lam4 = L4/r4;
S_Bu = (1-Lam4)*S_J4 + Lam4.*S_Bu;
dist = sqrt((Ba(1)-S_Bu(1))^2+(Ba(2)-S_Bu(2))^2+(Ba(3)-S_Bu(3))^2);
    end
end
%% UNREACHABLE
if (L1 >= L2+L3+L4)
% FORWARD
r2 = sqrt((Ba(1)-S_J2(1))^2+(Ba(2)-S_J2(2))^2+(Ba(3)-S_J2(3))^2);
Lam2 = L2/r2;
S_J3 = (1-Lam2)*S_J2 + Lam2.*Ba;
r3 = sqrt((Ba(1)-S_J3(1))^2+(Ba(2)-S_J3(2))^2+(Ba(3)-S_J3(3))^2);
Lam3 = L3/r3;
S_J4 = (1-Lam3)*S_J3 + Lam3.*Ba;
r4 = sqrt((Ba(1)-S_J4(1))^2+(Ba(2)-S_J4(2))^2+(Ba(3)-S_J4(3))^2);
Lam4 = L4/r4;
S_Bu = (1-Lam4)*S_J4 + Lam4.*Ba;
% Distance between new position of the Bucket and Ball
dist = sqrt((Ba(1)-S_Bu(1))^2+(Ba(2)-S_Bu(2))^2+(Ba(3)-S_Bu(3))^2);
end
%% NEW ANGLES 2, 3, 4
% Calculation of the new angles of the calculated
% new positions of the joints (as in the CCD method)
V_2 = S_Bu - S_J2;
W_2 = Ba - S_J2;
V2_p = V_2.*Jx + V_2.*Jy;
W2_p = W_2.*Jx + W_2.*Jy;
New_ANGLE2 = acosd(complex(dot((V2_p/norm(V2_p)),(W2_p/norm(W2_p)))));
R_2 = cross(([V2_p(1); V2_p(2); 0]),([W2_p(1); W2_p(2); 0]));
if (R_2(3) < 0) && (Ba(1) > S_J2(1))
    New_ANGLE2 = -New_ANGLE2;
end
if (R_2(3) > 0) && (Ba(1) < S_J2(1))
    New_ANGLE2 = -New_ANGLE2;
end
V_3 = S_Bu - S_J3;
W_3 = Ba - S_J3;
V3_p = V_3.*Jx + V_3.*Jy;
W3_p = W_3.*Jx + W_3.*Jy;
New_ANGLE3 = acosd(complex(dot(V3_p/norm(V3_p),(W3_p/norm(W3_p)))));
R_3 = cross([V3_p(1); V3_p(2); 0],[W3_p(1); W3_p(2); 0]);
if (R_3(3) < 0) && (Ba(1) > S_J3(1))
    New_ANGLE3 = -New_ANGLE3;
end
if (R_3(3) > 0) && (Ba(1) < S_J3(1))
    New_ANGLE3 = -New_ANGLE3;
end
V_4 = S_Bu - S_J4;
W_4 = Ba - S_J4;
V4_p = V_4.*Jx + V_4.*Jy;
W4_p = W_4.*Jx + W_4.*Jy;
New_ANGLE4 = acosd(complex(dot((V4_p/norm(V4_p)),(W4_p/(norm(W4_p))))));
R_4 = cross([V4_p(1); V4_p(2); 0],[W4_p(1); W4_p(2); 0]);
if (R_4(3) < 0) && (Ba(1) > S_J4(1))
    New_ANGLE4 = -New_ANGLE4;
end
if (R_4(3) > 0) && (Ba(1) < S_J4(1))
    New_ANGLE4 = -New_ANGLE4;
end
%% SIGNALS ON ACTUATORS
% Signal to the actuator 2 (BoomLift)
if  (ANGLE2 > New_ANGLE2)
   Power2 = -0.8;
elseif  (ANGLE2 < New_ANGLE2)
    Power2 = 0.8;
else
    Power2 = 0.1;
end
% Signal to the actuator 3 (DipperArm)
if  (ANGLE3 > New_ANGLE3)
   Power3 = 0.8;
```

```
elseif  (ANGLE3 < New_ANGLE3)
    Power3 = -0.8;
else
    Power3 = 0;
end
% Signal to the actuator 4 (Bucket)
if  (ANGLE4 > New_ANGLE4)
   Power4 = 0.8;
elseif  (ANGLE4 < New_ANGLE4)
    Power4 = -0.8;
else
    Power4 = 0;
end
end
```

Appendix 2: Excavator bucket path in different methods

This appendix contains the images of the excavator path of the in space from its initial position to the target. The experiment was conducted in accordance with the zones indicated in Figure 22 and Table 8. In the figures below, all targets are at a reachable distance.

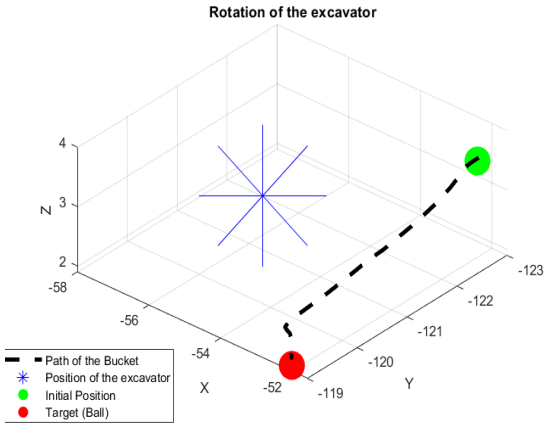


Figure 34. CCD. Zone 1. Down.

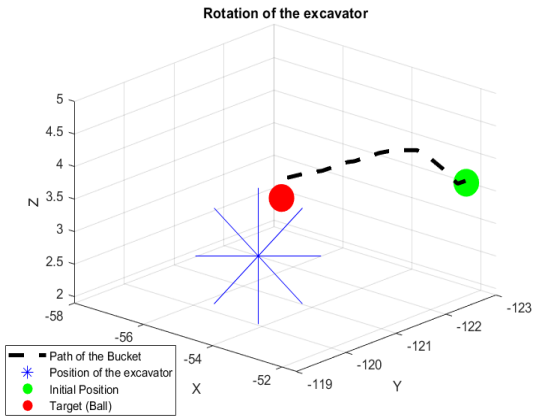


Figure 35. CCD. Zone 1. Up.

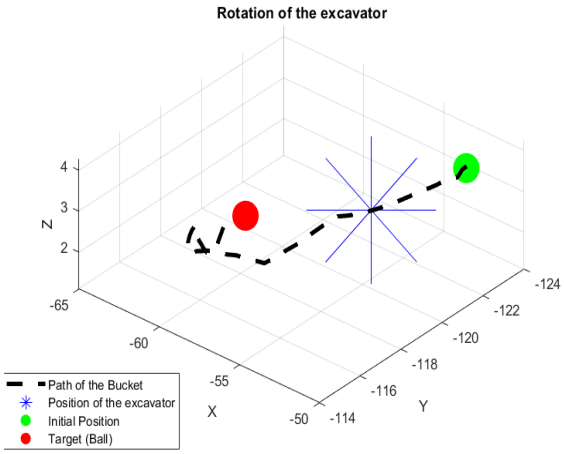


Figure 36. CCD. Zone 2. Down.

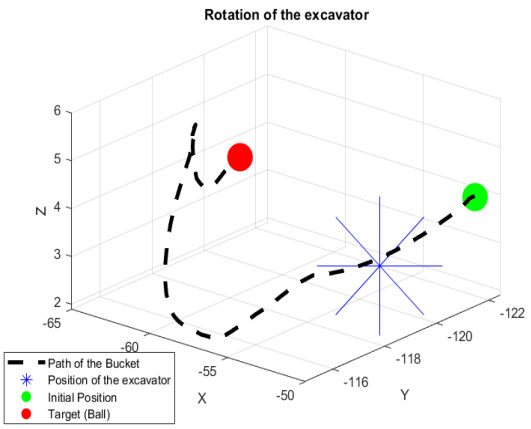


Figure 37. CCD. Zone 2. Up.

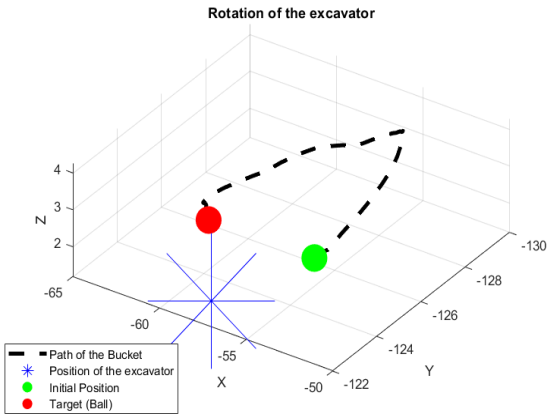


Figure 38. CCD. Zone 3. Down.

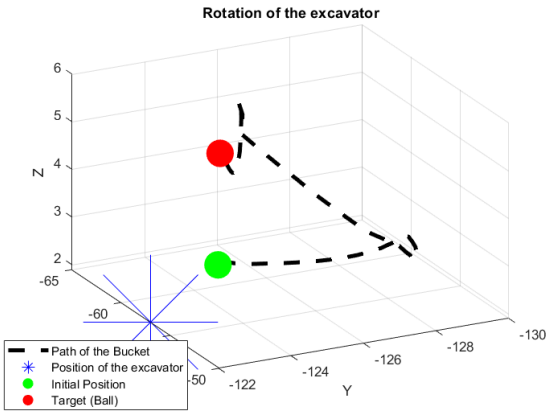


Figure 39. CCD. Zone 3. Up.

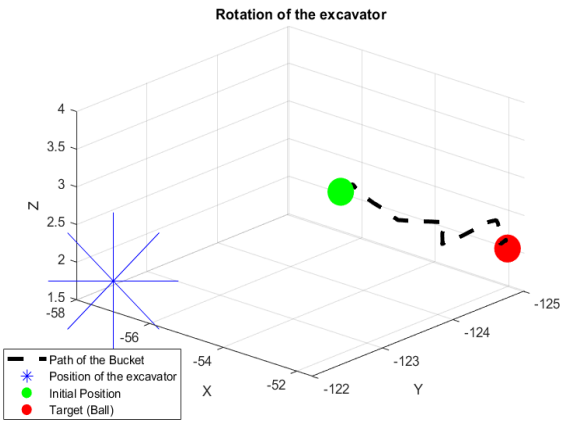


Figure 40. CCD. Zone 4. Down.

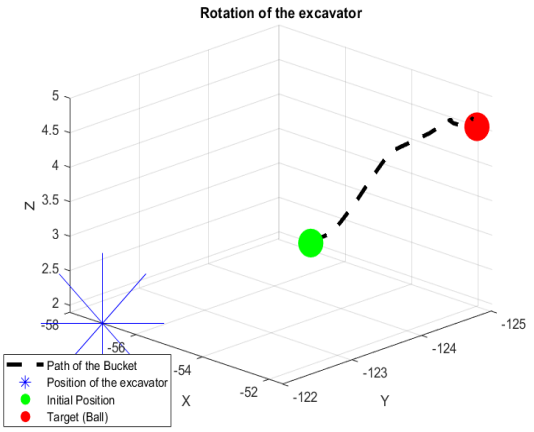


Figure 41. CCD. Zone 4. Up.

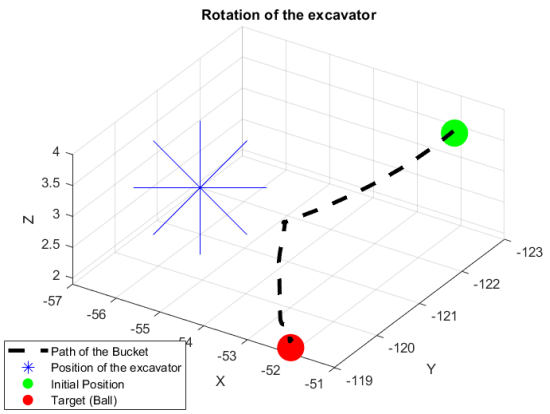


Figure 42. Target Triangle. Zone 1. Down.

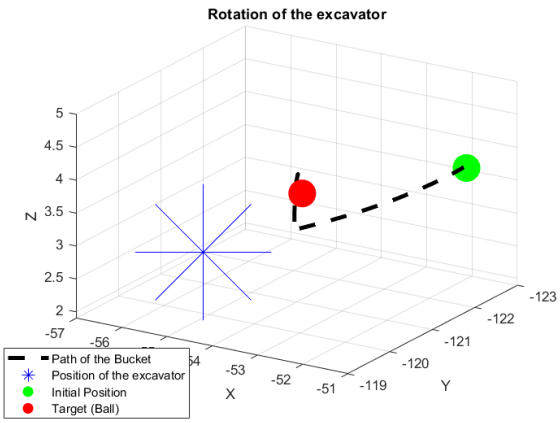
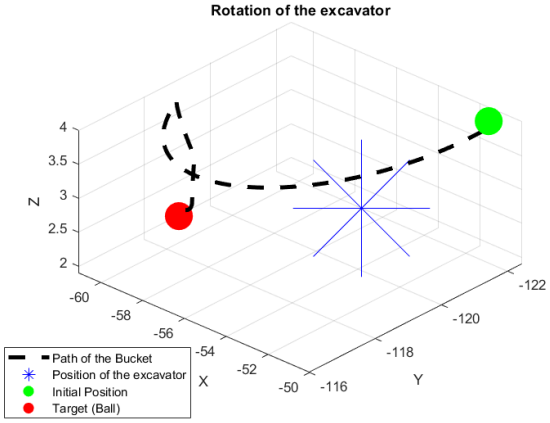


Figure 43. Target Triangle. Zone 1. Up.



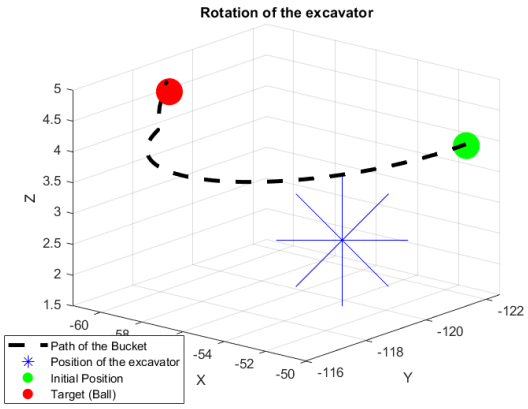
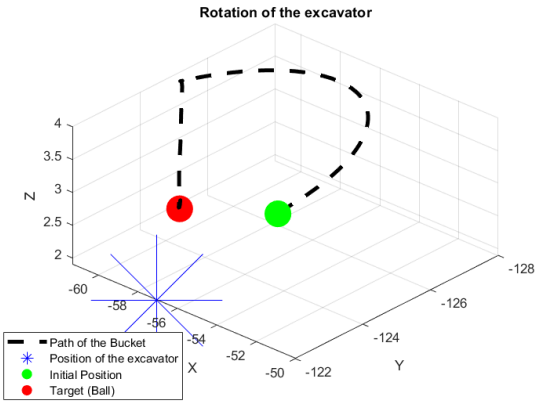


Figure 44. Target Triangle. Zone 2. Down. Figure 45. Target Triangle. Zone 2. Up.



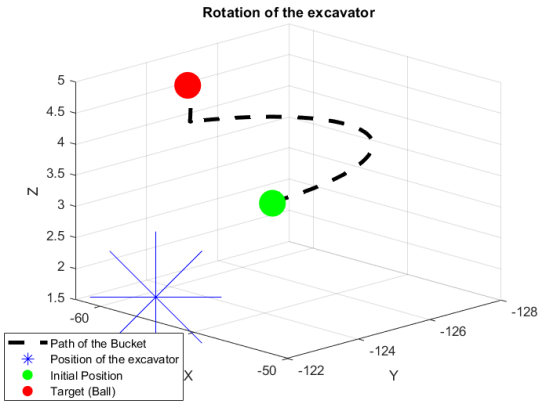
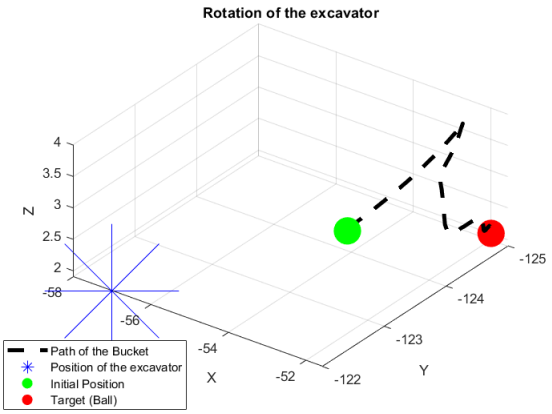


Figure 46. Target Triangle. Zone 3. Down. Figure 47. Target Triangle. Zone 3. Up.



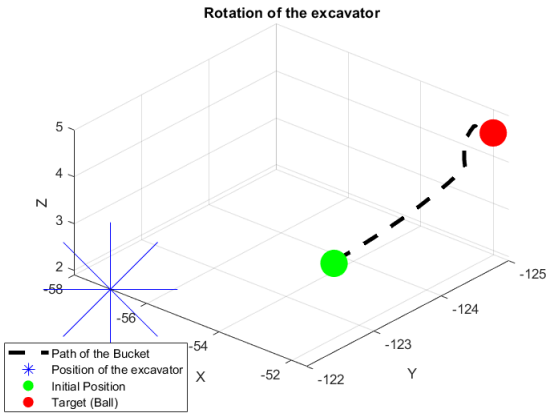


Figure 48. Target Triangle. Zone 4. Down. Figure 49. Target Triangle. Zone 4. Up.

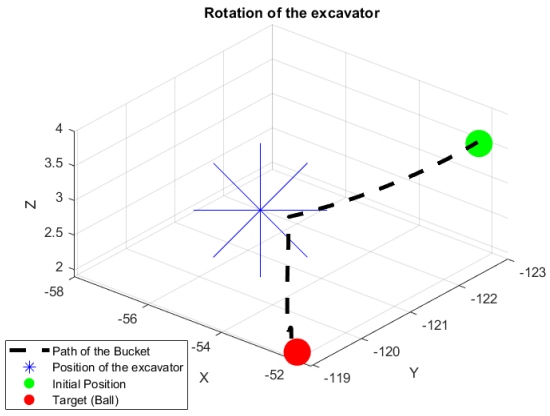


Figure 50. FABRIK. Zone 1. Down.

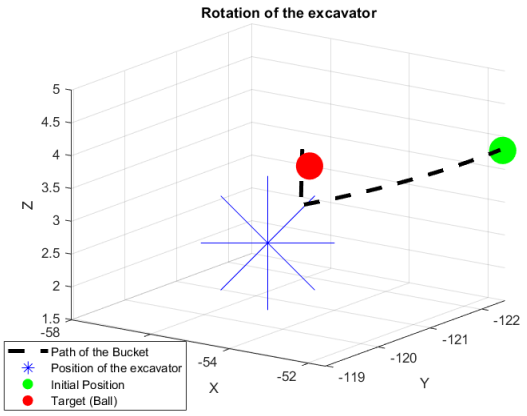


Figure 51. FABRIK. Zone 1. Up.

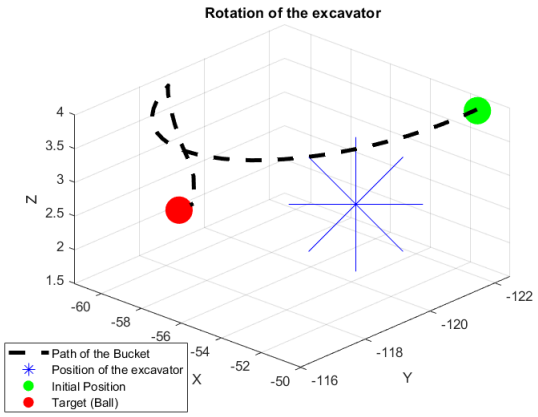


Figure 52. FABRIK. Zone 2. Down.

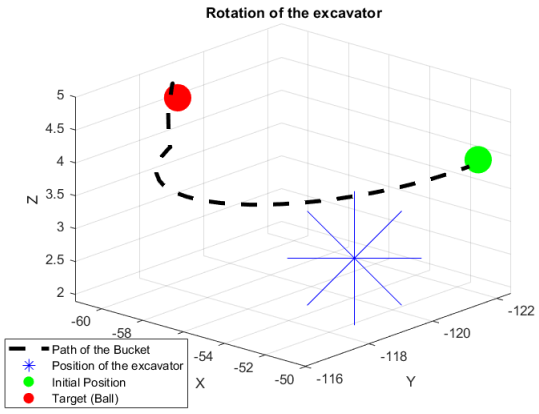


Figure 53. FABRIK. Zone 2. Up.

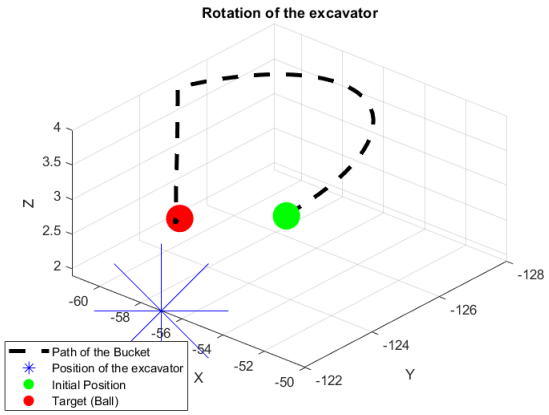


Figure 54. FABRIK. Zone 3. Down.

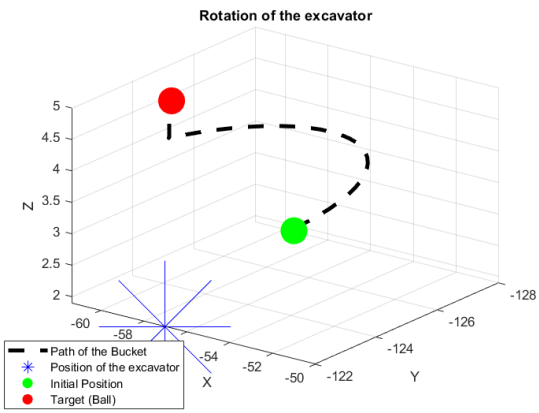


Figure 55. FABRIK. Zone 3. Up.

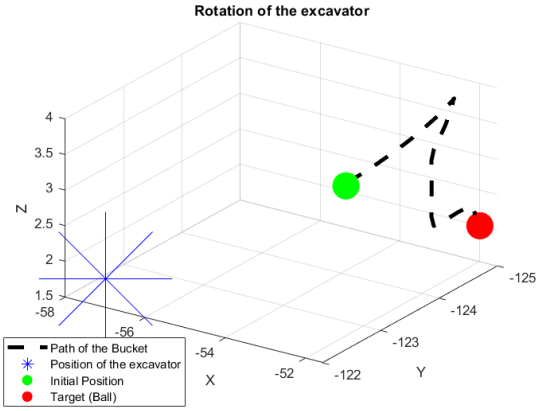


Figure 56. FABRIK. Zone 4. Down.

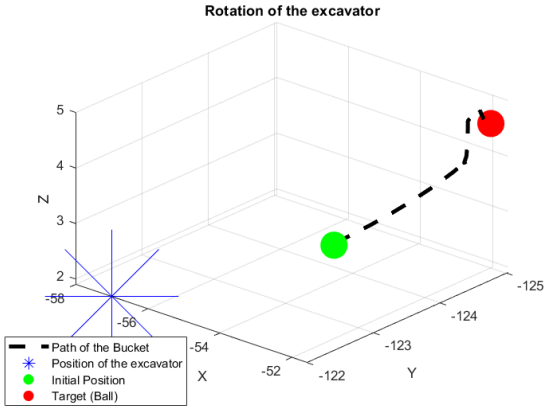


Figure 57. FABRIK. Zone 4. Up.

Appendix 3: Views of the excavator's end position

This appendix contains the screenshots of the final positions of the excavator in case of reachable targets.

Figure 58. CCD. Zone 1. Down.

Figure 59. CCD. Zone 1. Up.

Figure 60. CCD. Zone 2. Down.

Figure 61. CCD. Zone 2. Up.

Figure 62. CCD. Zone 3. Down.

Figure 63. CCD. Zone 3. Up.

Figure 64. CCD. Zone 4. Down.

Figure 65. CCD. Zone 4. Up.

Figure 66. Target Triangle. Zone 1. Down. Figure 67. Target Triangle. Zone 1. Up.

Figure 68. Target Triangle. Zone 2. Down. Figure 69. Target Triangle. Zone 2. Up.

Figure 70. Target Triangle. Zone 3. Down. Figure 71. Target Triangle. Zone 3. Up.

Figure 72. Target Triangle. Zone 4. Down. Figure 73. Target Triangle. Zone 4. Up.

Figure 74. FABRIK. Zone 1. Down.

Figure 75. FABRIK. Zone 1. Up.

Figure 76. FABRIK. Zone 2. Down.

Figure 77. FABRIK. Zone 2. Up.

Figure 78. FABRIK. Zone 3. Down.

Figure 79. FABRIK. Zone 3. Up.

Figure 80. FABRIK. Zone 4. Down.

Figure 81. FABRIK. Zone 4. Up.