Univerza v Mariboru

Fakulteta za elektrotehniko, računalništvo in informatiko

ROBOTIC ARM WITH TEN DEGREES OF FREEDOM

MARIA LUZ RASINES RODRIGUEZ

MARIA LUZ RASINES RODRIGUEZ
Robotic arm with ten degrees of freedom

MARIA LUZ RASINES RODRIGUEZ
Robotic arm with ten degrees of freedom

Student: Maria Luz Rasines Rodriguez

Student program: Engineering in industrial technologies, specialization in industrial electronics, Erasmus+ international exchange program.

Mentor: Prof. Dr. Riko Šafarič

Comentor: Kristijan Korez

MARIA LUZ RASINES RODRIGUEZ

Robotic arm with ten degrees of freedom

ABSTRACT

This bachellor thesis present a matlab code that makes a robotic arm to work. Controlling the arm means that all the movements of the arm are defined. The robotic arm is ten degrees of freedom. Inside this task we are going to use the code of the optimization algorithm based on genetic algorithms and the knowledge of direct kinematics equations to get the inverse kinematics model used to control the movement of a robotic arm. With this combination we got a good inverse kinematic model of 10 D.O.F. robot mechanism.

KEYWORDS

Robotic arm, genetic algorithm, 10 dregrees of freedom, kinematic.

INDEX

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Motivation and main idea

The use of robots is the latest trend in automating the different processes required in today's industry, such as welding, spray painting, conveying, material grinding, modelling, machine tools and many more.

Carrying out these objects requires the correct execution of the robot's movements and this is achieved by making use of the concepts that regulate the kinematics, dynamics and movement control of the robot.

A variety of applications that support the tasks of design, construction, simulation, communication and programming of robots are currently available.

The main idea of this project and what motivates it is realization is the idea of getting a Matlab code that perfectly defines the movement of a robotic arm.

We assume that a robotic arm is a programmable mechanical arm with functions similar to those of a human arm. We can find robotic arms that are the total of a mechanism or that are part of a more complex robot. The parts of a robotic arm are connected through joints that allow either rotational movement or translational movement or linear displacement.

This robotic arm will be made up of 10 degrees of freedom. The degrees of freedom are related to the joints of an arm. Each degree of freedom is a movement in one direction of each joint. Wrist, elbow and shoulder can move along x,y,z axes and can also rotate on these.

Based on the fact that the arm is in a certain position, we want, using the matlab code, to reach another position said by us.

To carry out this project we are going to focus on two important points that will be explained extensively in the following sections of this project. We will need to have a basis in robot arm direct mechanics and genetic algorithms. It will also be important to have a knowledge base on the use of Matlab, which is the main tool we will use. Once these two issues are well understood we will focus on understanding the codes that we will use as a basis for our project.

1.2 Objetives

In this project we can find the main objective, and what we could call the final objective with which the whole project is related. And related to this some secondary objectives that we are going to fulfil as we advance in the realization of the project. We will classify them below.

The main objetice of this bachellor tesis:

- To obtein a code that describes the complete functioning of a robotic arm. For this robotic arm we will take into account that it is composed of ten degrees of freedom. Using Matlab as the main tool.

This project requires skills from several disciplines: Mechanics, Robotics, computers and other subjects. During the work these disciplines interact so the secundary objetives that have been considered:

- To undestand robot arm kinematics, a branch of mechanics. Know the basics of forward or direct kinematics and inverse kinematic.

- To learn genetic algorithm as optimisation method. Know how to use this algorithm as a method to know the maximums and minimums of a function or its parameters.

- To know how to use the genitc algorithm of Kristijan Korez in Matlab.

- To know how to use the code of 6 grades of freedom created by Riko Šafarič in Matlab.

- To obtain a code of ten degreas of freedom that join the two previous codes.

1.3 Project plan

The development of this project has followed different steps. First we can find the learning steps, about the topic to be dealt with, in order to get a knowledge base. And finally the most practical steps, in which we develop our work. This steps are listed as follows:

1- The main characteristics that are involved on Robot arm kinematics are the forward kinematics and the inverse kinematics. This important topics must be learnt.

2- With regard to genetic algorithm we will need to know in depth how it is used as an optimization algorithm. The steps to follow to make a simple example of optimization are very important. From the choice of population, the conversión to binary, the division of chromosomes, the mutation, the cross-over, etc.

These two previous points were the basis for everything done in Matlab.

3- The next step was work with the genetic algorithm of matlab created by Kristijan Korez. This algotithm is based on the therycal genetic algothim. The code conteins everything about the genetic algorithm. The code use simple comands to obtein the solution, the global maximums or minimal of functions.

4- Matlab's 6 degrees of freedom code solution provided by Riko details the implementation of the robot arm control program. In it, we found a code with which we can know the position of the robotic arm. This code uses the command base that has Matlab of robotics. It uses the program's own commands related to forward kinematics and inverse kinetatics.

5- Once we have both Matlab codes, we'll put them together. We'll use the learning of the genetic algorithm to get the movement of the robotic arm defined. Combining both will give us an optimization process for our final result.

6- Once we got a Matlab program that combines both codes and once we know how to use it and understand what it does, we went on to try and turn it into a code that defines a robotic arm with 10 degrees of freedom, not 6 as we had defined. This change makes the movements of the robotic arm much more complete. Since when adding each degree of freedom we are referring to that our arm takes into account one more movement. As we have said before, each degree of freedom defines a movement either of rotation or of translation.

7- Finally we will have finished the practical part of our project.

1.3 Introduction to robotics and premises

In almost every age and culture, men have tried to build automatic machines that make their work easier, make their existence more comfortable, satisfy their curiosity and their desire to learn and investigate, or simply provide entertainment. At the end of the 18th century and also at the beginning of the 19th century, some machines were developed that are already worth mentioning for use in the textile industry, among which there were already some looms where, by means of punch cards, it was possible to choose the type of fabric to be woven. This was one of the first historical precedents of machines programmed by numerical control. A little later than in the textile industry, automatisms were incorporated in the mining and metallurgical industries, it was the time of the steam machines, which is one of the bases of the current industrial robots. The word robot was first used in 1920 and is derived from the Czech word "robotnik" and means, servant, servant or forced worker. It is in the 20th century when we start talking about robots and the development of these is linked to the development of microprocessors.

1954	From this date, the American George Devol, begins the construction of an articulated arm that performs a sequence of computer-programmable movements; this "arm" is considered to be the first industrial robot.
1956	Devol met Joseph Engelberger and together they founded the robot manufacturing company Unimation in 1960.
1961	A hydraulically driven Unimate robot is tested in a die-casting process at General Motors.

1968	Kawasaki joins Unimation and starts manufacturing and using industrial robots in Japan. In this year General Motors, employs robot batteries in the manufacturing process of the cars' rice paddies.
1973	The Swedish company ASEA, which manufactures the first completely electric robot, is the type of drive that has come to prevail, due to advances in the control of electric motors.
1974	The first industrial robot is introduced in Spain. This was also the year when the AL programming language began to be used, from which others would be derived for later use, such as the VAL (Victor's Assembly Language) of the PUMA robots, implemented in 1975 by Victor Scheinman, who, together with Devol and Engelberger, were pioneers in industrial robotics.
1978	The PUMA (Programmable Universal Machine for Assambly) robot from Unimatión begins to be used, which is one of the most widely used models. Its multi-articulated "arm" design is the basis for most of today's robots.
1981	Marketing of the SCARA (Selective Compliance Arm for Robotic Assambly) type robot begins in Japan.
1987	The International Federation of Robotics is formed with headquarters in Stockholm.

Table 1. Short introductios to robotics.

During these years,1954-1987, the foundations of industrial robotics were laid, and later on the use and sophistication of robots will increase with the increase in the performance of microprocessors and the possibilities of computing. The rise of robots, in addition to the development of electronics, is due to the industrial need to manufacture products with variations according to the tastes and needs of customers, which has made specific automatic manufacturing machines and devices to make a single product, only profitable for large series; with robotics and flexible automation, different products of the same family can be manufactured, with few or no structural changes in production lines, as these systems are adapted by program to the variable manufacturing conditions.

The robots, can be classified in several ways, a classification by type, can be the

next:

-Androids and Zoomorphic

-Mobile

-Service

-Telemanipulated or tele-operated

-Industrial

Almost all national and international associations or federations have their own definition of a robot, although with obvious analogies.

That of the International Federation of Robotics (IFR), is: "Industrial handling robot means an automatic, reprogrammable and multifunctional handling machine with three or more axes that can position and orient materials, parts, special tools or devices for carrying out various tasks at different stages of industrial production, whether in a fixed or moving position".

The definition of industrial robot of the I.S.O., comes from the "Robotic Industries Association (RIA) and is: "Reprogrammable multifunctional manipulator with various degrees of freedom, capable of handling materials, parts, tools or special devices according to variable programmed paths to perform various tasks". For the Japanese Industrial Robotics Association (JIRA): Robots are "devices capable of moving in a flexible manner analogous to that of living organisms, with or without intellectual functions, allowing operations in response to human commands. This definition is somewhat more ambiguous than the previous ones; in Japan the concept of robot is less restrictive than in other countries.

2. THEORETICAL BACKGROUND

2.2. Robot arm kinematics

The first Branch of which it is necessary to have a base and some knowledge is Robot arm kinematics. To get these insights I have helped myself from Peter Cork`s book. One of his chapters is about this. The most important facts we need to know are set out below.

In general, kinematics is a branch of mechanics that studies the movement of a body, without considering its mass or external forces. It focuses on the study and manipulation of mechanical links and joints.

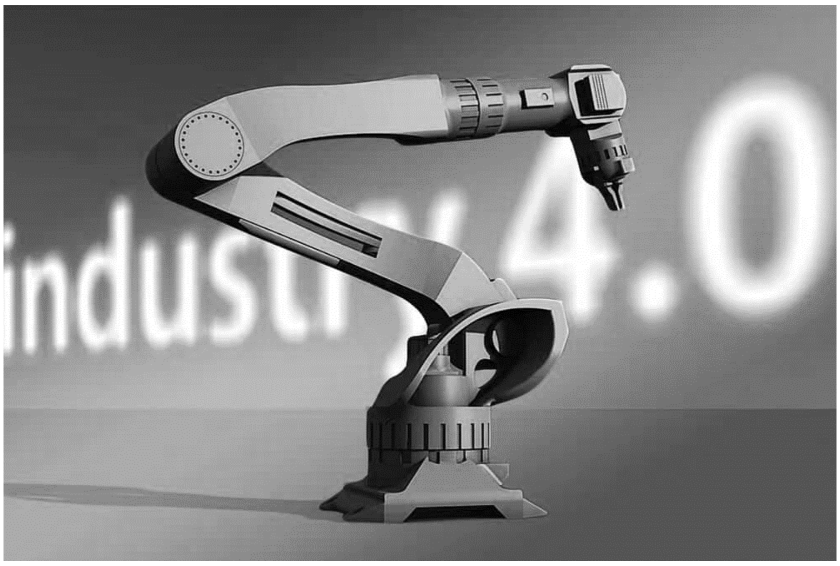


Figure 1: Robot arm

Especially when talking about robotic arms. These can be defined as a set of links (rigid body) that are joined in chain. This chain begins with a support base, generally fixed but sometimes it could be mounted on a mobile base, and ends at the other end, mobile and free. This is where the work tool is attached to the robot tip. It can be just a clamp, or a more complex tool like a drill, saw, wrench...

Their different parts are joined and connected to make their rotation and translation movements. With these movements the uses of the arms are extended. The arms can work with better speed and more displacement of the objects they transport. They can even perform more delicate tasks.

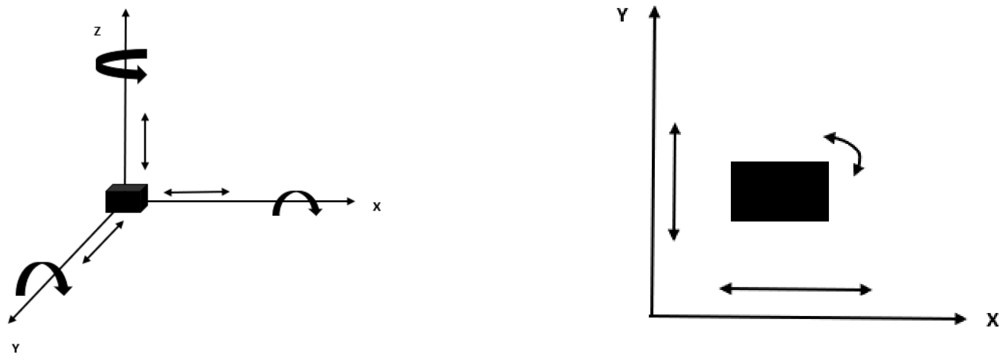


Figure 2: DOF of a rigid body. In space and in a plane.

As we have just said we can get a translational motion, a rotational motion or a combination of both. There are several types of joints, the two most commonly used by robots are the prismatic (P) and the rotating (R); both allow a single Degree of Freedom (DOF), or independent movement between the parts they join. Prismatics allows only a relative movement of displacement or translation in one direction or axis; rotation allows only a relative movement of rotation around a single axis; for this reason, we can say in general that the number of DOF in a robot is equal to the number of its joints or the number of axes. With three translations according to the respective X, Y or Z axis and three turns or rotations (yaw, pitch, roll) related to these same axes, we can position any element, object or tool in space.

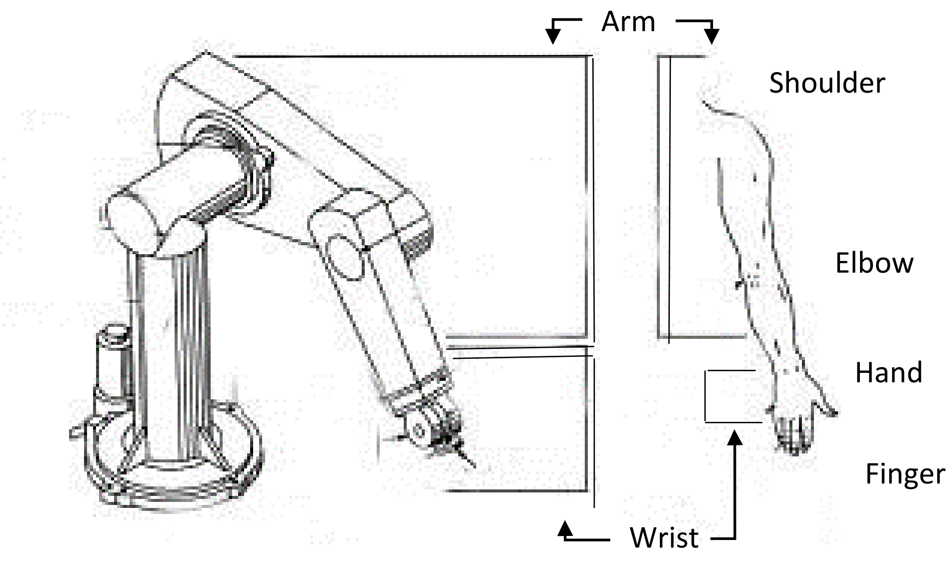


Figure 3. Correspondence between the robot and the real arm.

2.3. Denavit-Hartenberg notation

Our robotic arm will be composed of 10 degrees of freedom. This makes it defined by 10 joints. In other words each joint has one degree of freedom. At this point we have to introduce the Denavit-Hartenberg notation.

It is a systematic procedure to describe the kinematic structure of an articulated chain constituted by joints with only one degree of freedom.

For our 10 joint robot, from 1 to 10, we have 11 links, numbered from 0 to 10. The link 0 will be the base and the link 10 will be the end of our arm.

For this purpose, each joint is assigned a local reference system with origin in a point and orthogonal axes (X,Y,Z). Normally you start with a fixed and stationary reference system given by the axes (X_0, Y_0, Z_0) and an origin at a fixed point on the base on which the entire chain structure is mounted.

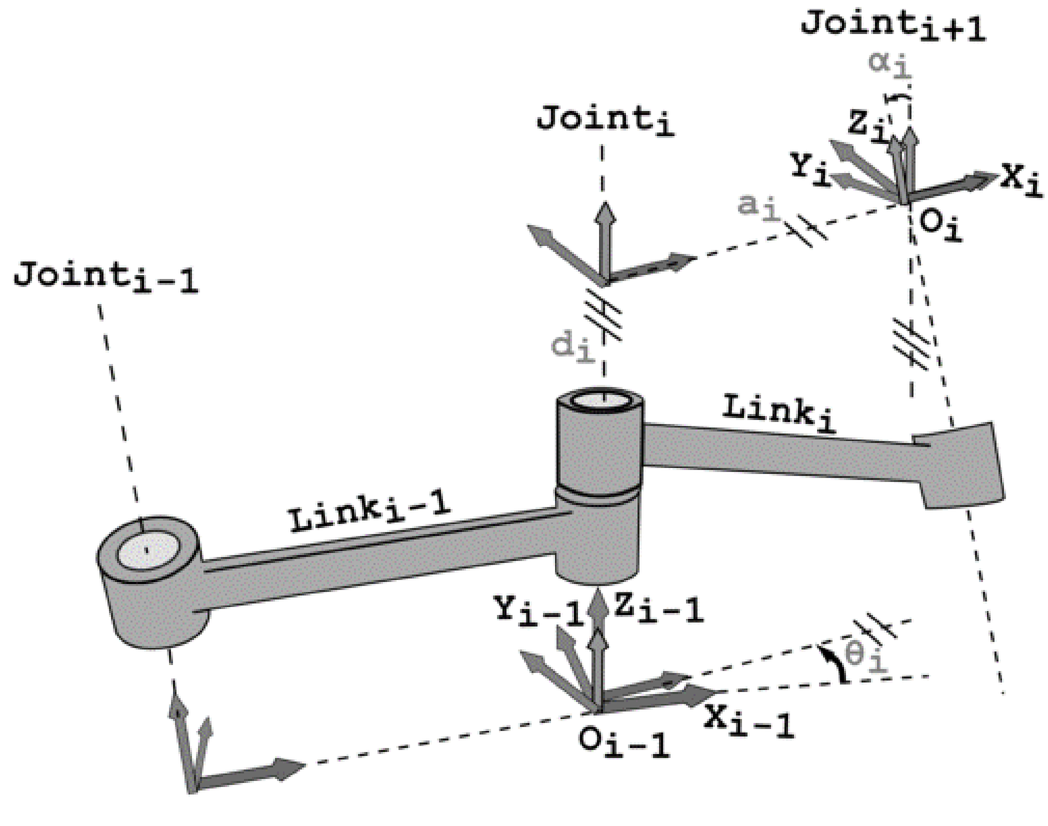


Figure 4. Joints and links.

Joint angle	θ_j	The angle between the x_{j-1} and x_j axes about the z_{j-1} axis.	Revolute joint variable
Link offset	d_j	The distance from thee origin of frame j-1 to the x_j axis along the z_{j-1} axis.	Prismatic joint variable
Link length	a_j	The distance between the z_{j-1} and z_j axis along the x_j axis; for intersecting axes is parallel to $\hat{z}_{j-1} \times \hat{z}_j$.	Constant
Link twist	α_j	The angle from the $z_{j-1}\ axis\ to\ the\ z_j$ axis about the z_j axis.	Constant
Joint type	σ_j	$\sigma = 0$ for a rebolut3e joint, $\sigma = 1$ for a prismatic joint.	Constant

Table 2: Denavit – Hartenberg parameters. [1]

2.4. Forward kinematics

Direct kinematics is a technique used in 3D computer graphics to solve and calculate the position of parts of an articulated structure from its fixed elements and the transformations caused by the structure's joints.

A typical example on which to perform this type of calculation is a robot, where a fixed reference system located at the base of the robot can be determined, and the location of each of the links can be described with respect to that reference system.

Determines the location of the end of the robot, respect to a reference coordinate system, when the values of the joints and of the geometrical parameters of the robot elements are known.

2.5. Inverse kinematics

Inverse kinematics is a problem that involves the calculation of all possible combinations of joint angles that could be used to achieve a certain position and orientation of the manipulator end effector.

The inverse kinematics problem is not as simple as the direct kinematics problem, because as kinematic equations are non-linear, their solutions are not always easy or even possible to find. Questions also arise about the existence of one solution, or multiple solutions.

I mean knowing the location of the robot determines what the configuration of the robot should be (joints and geometrical parameters).

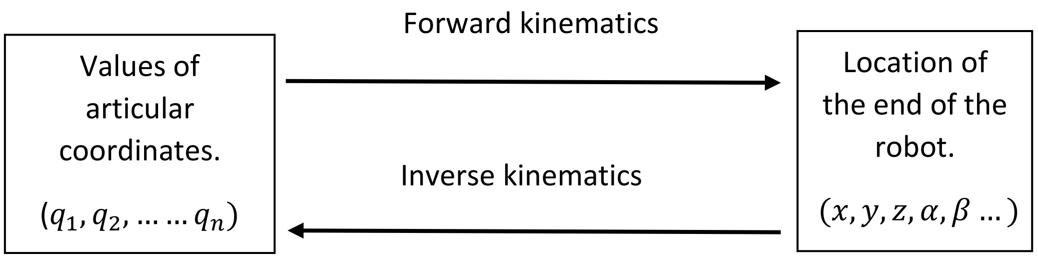

Forward kinematics
Values of articular coordinates.
$(q_1, q_2, \dots \dots q_n)$
Location of the end of the robot.
$(x, y, z, \alpha, \beta \dots)$
Inverse kinematics

Table 3: Relation between forward kinematics and inverse kinematics.

2.6. Genetic algorithm. Optimization and Algotithm.

The genetic algorithm is one of the best and more simple algorithms about the evolution. It has all the basic topics about the evolution.

We use this algorithm like a optimization method. In our case we are also going to use it like that. The genetic algorithm is one of the best methods, the one more effective, to find global maximus and minimums.

The dictionary defines algorithm like: "a process or set of rules to be followed in systematic operations (calculations) or other problem-solving operations, especially by a computer." This means a mathematical process that describes a task step by step. The steps are usually very simple, the complex task is designing the algorithms.

As we have said before the genetic algorithm is an optimization method. We can define optimization like "In mathematics and computer science, a method for determining the values of the variables involved in a process or system so that the result is the best possible. "So is the action of making the best or most effective use of a situation or resource. in these definitions say best possible solution because in a lot of case there are more than one solution, overcoat in the more complex cases. In this type of problems, we can find local or global solutions.

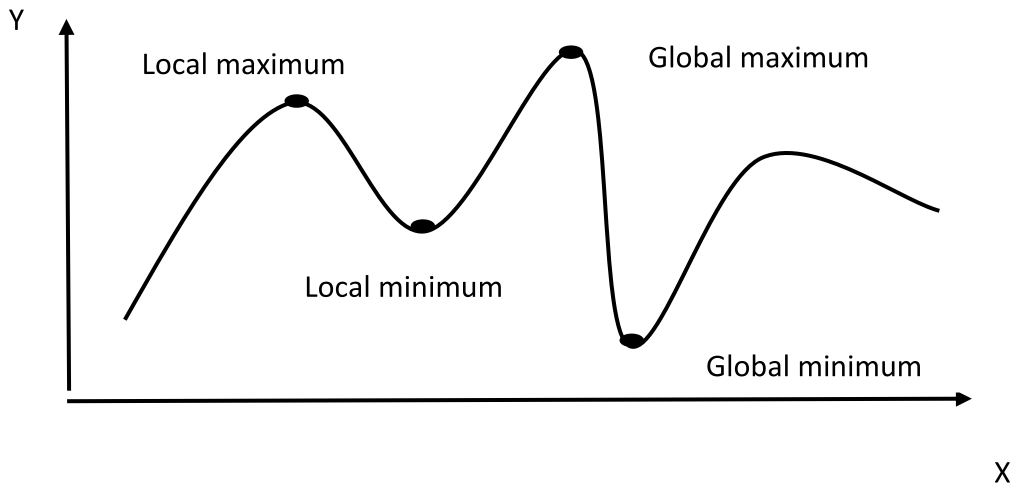


Figure 5. Minimun and maximun points.

2.7 Genetic algorithm

This algorithm follows the example of nature. It is based entirely on the method of evolution that we can find in nature genetic. The one we are going to use and to study, the genetic algorithm, is the most important and successful. This algorithm was presented the first time by J.H. Hollands. He defined the method trying to imitate totally to the nature, following its example.

For the work we want to do we need an algorithm the good enough for find the global optimum and not get stuck in a local optimum.

First, we can do a little summary of the most important characteristics:

-Use binary system

-Use the cross-over and the combination.

-Use the mutation

-The conditions to stop the loop.

In this figure we can see clearer the steps to follow.

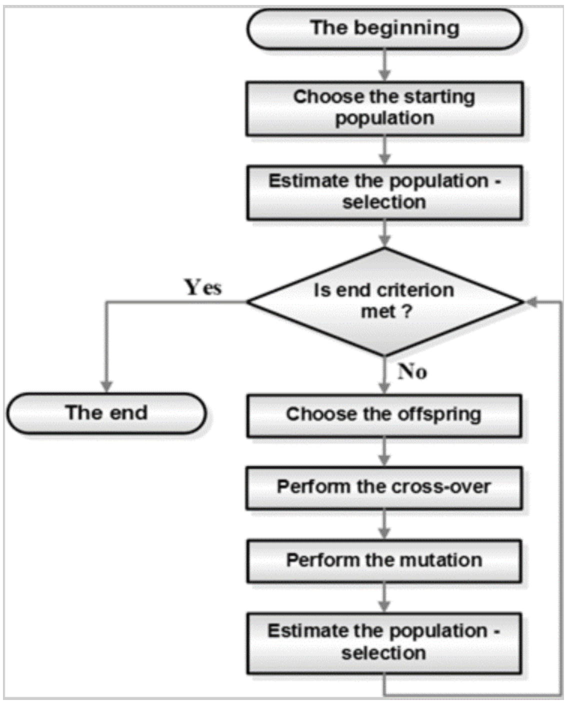


Table 4. Construction blocks. [2]

Second, we can explain more extended how the method works.

We start to work with a randomly selection of population. This population is composed of chromosomes, a series of genes. It gene is a bit, a zeros or a one. We must put this population in to binary system. Its genetic material is represented as a series of zeros and ones and we are going to work and to operate with this data.

From this data and helping us with the function which we are working with, we put a value on each data, quality value. We choose the ones that have the highest value, which will be the best.

Depending on this value the next generation will be created, the best has more copies in the next generation and the worst must be discarded. So those who have a higher quality value will have much advantage over those who have a lower value.

At this point, there comes an important moment when it is the turn to the step of the cross-over. The moment when we combine the genetic material that we have. The probability of the cross-over used to be between 0.6 to 1.0. As the first choice of population this combination will be random. If the chromosome is too long the cross-over can be done in more than one point. We start to work with a new population.

Into this step we find another essential one, the mutation. Important points of the mutation:

-It is less than 0.1.

-It is the change of a bit from 0 to 1 or reverse.

-It is the probability of change in this experiment. In each case it is different.

-It is not done in every individual.

-It is done randomly.

It is imperative that this algorithm works so we do not get stuck in a local solution.

In nature the mutation occurs now when the genetic material is passed to the next generation and a small part is lost. Nature regenerates it.

The important thing we must keep in mind is that the essential thing about these operations is that the result of the optimization is the improvement from generation to generation.

There are some parameters that defined the final of the loop to stop the optimisation. The moment to stop is always defined by the operator. The operator must define the number of allowed iterations that the algorithm has to do. He has to define the number of generations. This is important because in some cases if we do not stop it the algorithm will continue to the infinitive.

Our process also needs an error permitted. It is also a way to stop the optimization. We can give a minimum error.

Finally, when making simple examples applying this algorithm, we will have to follow the steps explained in this section of the project and we can help us from the following tables.

A series of number	Starting population	Value of the variable	F(x) (Value inside the function)	Offspring number fitness	Actual offspring number (rounded up)
Sum					
Average					
Maximum					
Minimum					

Table 5. [2]

First generation offspring	Randomly chosen cross-over partner	Cross-over point	New population	Value of the variable x	F(x)
Sum					
Average					
Maximum					
Minimum					

Table 6. [2]

3. TOOLS

MATLAB is a powerful software package for scientific computing, focusing on numerical computation, matrix operations, and especially scientific and engineering applications. It can be used as a simple matrix calculator, but its main interest lies in the hundreds of both general purpose and specialized functions it has, as well as its possibilities for graphical visualization. MATLAB also has its own programming language, very close to the usual ones in numerical calculation (Fortran, C,...) that allows the user to write his own scripts (set of commands written to a file, which can be executed with a single command) to solve a particular problem and also write new functions with, for example, their own algorithms. MATLAB also has many toolboxes, which add specialized functionality. Numerous contributions from its thousands of users worldwide can be found on The MathWorks Web site.

Figure 6. Matlab.

Inside MATLAB and investigating this web site we can find that MATLAB has a series of specific commands for the topic we are studying. There is a library (Robotics system toolbox) which we can work with. This library provides us with commands (tools and algorithms) to design and simulate robots. It has commands that define the generation and the planning of trajectories, the control of movements… It has a pretty number of commands that are very important for us. These commands are about the forward and the inverse kinematics.

The other important library we can find in MathWorks web site is the genetic algorithm toolbox.

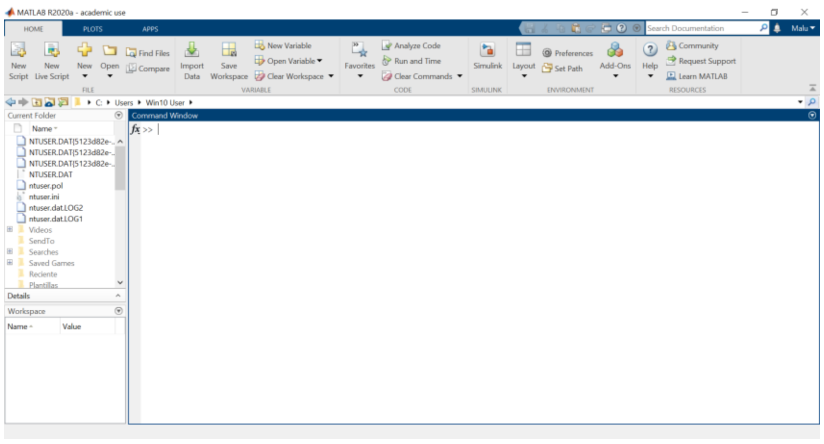


Figure 7: Matlab's workspace.

4. 10 DOF ROBOTIC ARM + GA

Once the area of work has been presented (MATLAB), with clear previous knowledge and with the objectives well defined we will move on to define this project.

As explained before, the main objective of this thesis is to obtain a MATLAB code that defines the movements of a ten-degree freedom robotic arm based on the genetic algorithm. We need to know how to define the robotic arm controlling all the possible parameters that can affect.

In a summarized way we are going to do through the code and randomly define the parameters of internal robot positions q1, q2, q3, q4, q5, q6, q7, q8, q9 and q10. Using the Denavit-Hartenberd notation and matrix we will define the HTM matrix. As we are going to play with the error, we will have already defined the correct HTM matrix, desired HTM. This matrix will be our input, the reference matrix.

Subtracting both we will know how close we have come. The process will be repeated as many times as necessary until the error is the one we have assigned, or the maximum iterations indicated are reached. In our case the process is repeated as long as the error is less than desired value (0.1). This is how we will carry out the optimization process and how we will find the best solution. We can explain very clear the process by a block diagram:

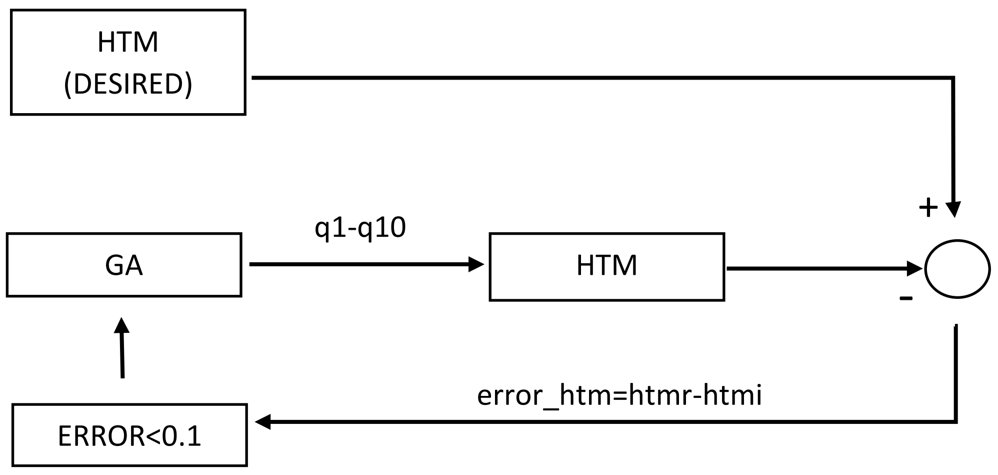


Figure 8: Block diagram. GA+HTM MATRIX

We will use the genetic algorithm created in MATLAB by Kristijan Korez which is composed of several functions. All of them together define the algorithm explained before (creation of the random population, the passage to binary, the cross-over, the mutation ...).

This algorithm is composed of the following codes and functions:

-FCN_ISKANA_FCN

-GA_GA

-GA_gene_v_osebke

-GA_generator_INT_ozje_generacije

-GA_izmerjono

-GA_KLIC_GA_FCN

-GA_krizaj

-GA_mutacija

-GA_oceni_popilacijo

-GA_selekciraj_v_gene

-ISKANI_PARAM

-TEST_UJEMANJA

By combining these functions with the Riko's code that defines in MATLAB a robotic arm of six degrees of freedom and making the necessary changes we find the solution to our project.

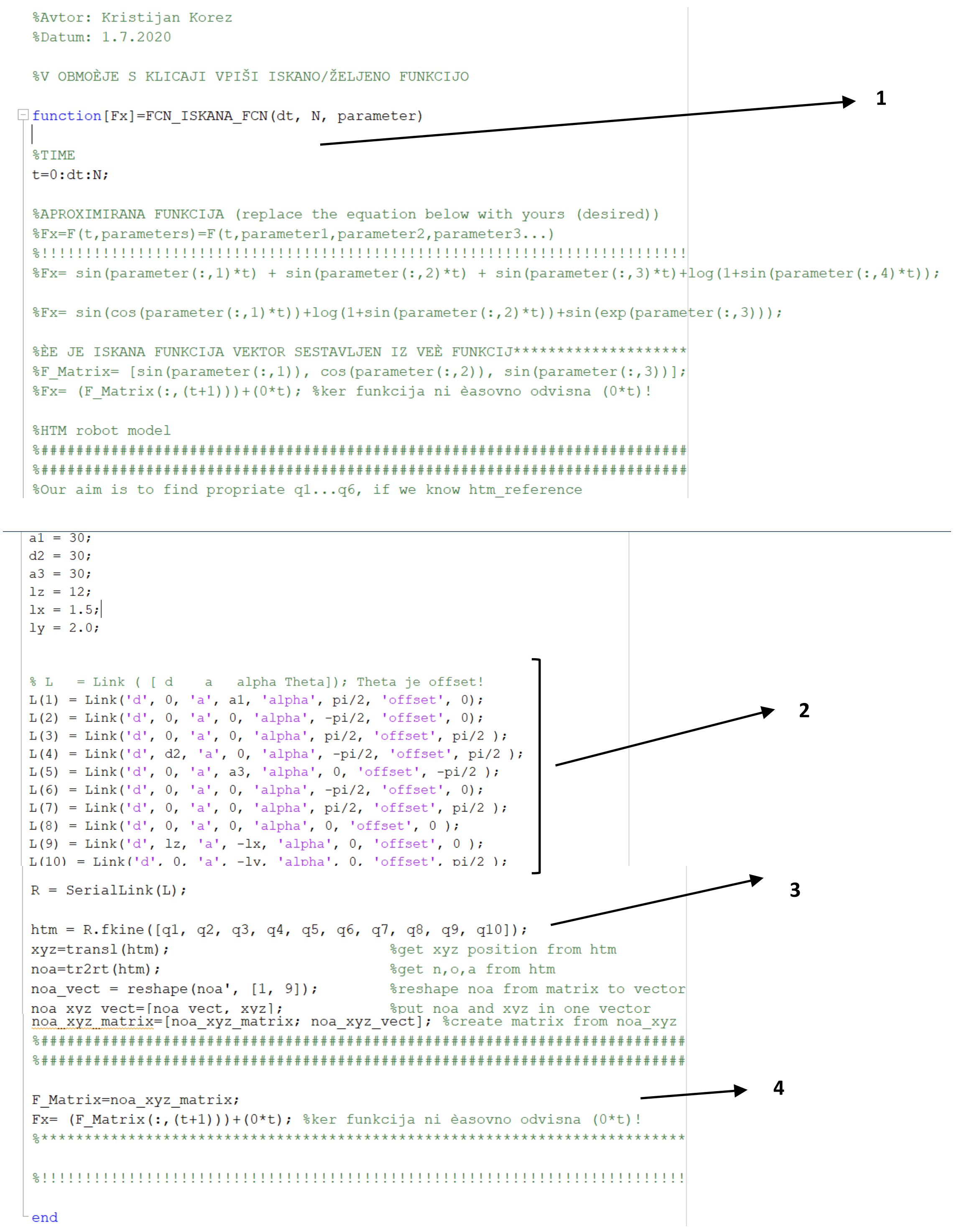
To use it we will follow these steps:

First of all, use the function FCN_ISKANA_FCN. Here we define the function we want to work with. In our case it will be the htm matrix. As we can see, we will randomly define the parameters from q1 to q10. We use these parameters to define our matrix.

As we can see there are some parts that are indicated in our code. These parts are the important ones. The first point is referred to the creation of the q values, we can see that we use a for, to create a loop that stop when the ten qs have a value. On the second point we can see the command Link. This command holds all information related to a robot joint and link such as kinematics parameters, rigid-body inertial parameters, motor and transmission parameters. We represent the robot link with this command where the input is a vector created by $\theta_j, d_j, a_j\ and\ \alpha_j$.The display values of the link object shows it is kinematics parameters as well as other status such the fact that it is a revolute joint and that the standard Denavit-Hartenberg parameter convention is used.

The third point is the creation of the htm matrix, a very important step in the code. We create the matrix with the command R.fkine (forward kinematics). As we have find in the matlab help: T=R.fkine(Q,options), if Q is a matrix the rows are interpreted as the generalized joint coordinates for a sequence of points along a trajectory.

The last part is where we definitely assign the value of the matrix to the function we are going to work with.



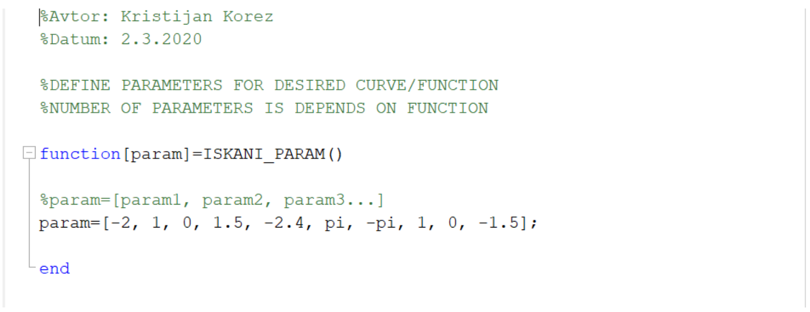
```
%Avtor: Kristijan Korez
%Datum: 1.7.2020

%V OBMOÈJE S KLICAJI VPIŠI ISKANO/ŽELJENO FUNKCIJO

function[Fx]=FCN_ISKANA_FCN(dt, N, parameter)

%TIME
t=0:dt:N;

%APROXIMIRANA FUNKCIJA (replace the equation below with yours (desired))
%Fx=F(t,parameters)=F(t,parameter1,parameter2,parameter3...)
%!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
%Fx= sin(parameter(:,1)*t) + sin(parameter(:,2)*t) + sin(parameter(:,3)*t)+log(1+sin(parameter(:,4)*t));

%Fx= sin(cos(parameter(:,1)*t))+log(1+sin(parameter(:,2)*t))+sin(exp(parameter(:,3)));

%ÈE JE ISKANA FUNKCIJA VEKTOR SESTAVLJEN IZ VEÈ FUNKCIJ*******************
%F_Matrix= [sin(parameter(:,1)), cos(parameter(:,2)), sin(parameter(:,3))];
%Fx= (F_Matrix(:,(t+1)))+(0*t); %ker funkcija ni èasovno odvisna (0*t)!

%HTM robot model
%##########################################################################
%##########################################################################
%Our aim is to find propriate q1...q6, if we know htm_reference
```

```
a1 = 30;
d2 = 30;
a3 = 30;
lz = 12;
lx = 1.5;
ly = 2.0;


% L   = Link ( [ d    a    alpha Theta]); Theta je offset!
L(1)  = Link('d', 0, 'a', a1, 'alpha', pi/2, 'offset', 0);
L(2)  = Link('d', 0, 'a', 0, 'alpha', -pi/2, 'offset', 0);
L(3)  = Link('d', 0, 'a', 0, 'alpha', pi/2, 'offset', pi/2 );
L(4)  = Link('d', d2, 'a', 0, 'alpha', -pi/2, 'offset', pi/2 );
L(5)  = Link('d', 0, 'a', a3, 'alpha', 0, 'offset', -pi/2 );
L(6)  = Link('d', 0, 'a', 0, 'alpha', -pi/2, 'offset', 0);
L(7)  = Link('d', 0, 'a', 0, 'alpha', pi/2, 'offset', pi/2 );
L(8)  = Link('d', 0, 'a', 0, 'alpha', 0, 'offset', 0 );
L(9)  = Link('d', lz, 'a', -lx, 'alpha', 0, 'offset', 0 );
L(10) = Link('d', 0, 'a', -ly, 'alpha', 0, 'offset', pi/2 );
```

```
R = SerialLink(L);

htm = R.fkine([q1, q2, q3, q4, q5, q6, q7, q8, q9, q10]);
xyz=transl(htm);                             %get xyz position from htm
noa=tr2rt(htm);                              %get n,o,a from htm
noa_vect = reshape(noa', [1, 9]);            %reshape noa from matrix to vector
noa_xyz_vect=[noa_vect, xyz];                %put noa and xyz in one vector
noa_xyz_matrix=[noa_xyz_matrix; noa_xyz_vect]; %create matrix from noa_xyz
%##########################################################################
%##########################################################################

F_Matrix=noa_xyz_matrix;
Fx= (F_Matrix(:,(t+1)))+(0*t); %ker funkcija ni èasovno odvisna (0*t)!
%**************************************************************************

%!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

end
```

Figure 9. FCN_INSKANA_FCN.

The next code we will use is ISKANI_PARAMETRI. Here is the desired parameter. So, the parameter Q1 to Q10. These parameters are solution, in theory we do not know these parameters, we only know the HTM. But for the simulation this is necessary. So here we chose desired parameters and later GA try to find the closest ones.



```
%Avtor: Kristijan Korez
%Datum: 2.3.2020

%DEFINE PARAMETERS FOR DESIRED CURVE/FUNCTION
%NUMBER OF PARAMETERS IS DEPENDS ON FUNCTION

function[param]=ISKANI_PARAM()

%param=[param1, param2, param3...]
param=[-2, 1, 0, 1.5, -2.4, pi, -pi, 1, 0, -1.5];

end
```

Figure 10: ISKANI_PARAM

We are now at the main code, GA_KLIC_GA_FCN. It is the code you can play with more. In it we have the possibility to change many parameters and define our optimization. Let's explain the most important details that we can change and what they do. As before, we will explain the important parts of the code indicating them by numbers. With the first signalling we find stev_osebkov, skalirnek and center. With this data we are going to make 'GA_generator_INT_populacije' generate the chromosomes. So, this part is very important. Depending on what we have entered in the previous function we will fill in this one. For each parameter above we will need a range in which it is, so if we call q to the defined parameter it will be in a range r. Depending on what we have introduced in the previous code explained, we will fill in this one. For each previous parameter we will need to define a range, so if we call q to the defined parameter it will be in a range r.

$$[r_1, r_2] = \frac{center \ \pm 1000}{skalirnik}$$

The reason for defining chromosomes on this way is to omit floating point numbers. It is faster to operate with integer number instead of floating-point ones.

Let's explain the following variables (number 2):

-dt: Sample time. In our case we have to leave this value equal 1, because we have time independent system.

-N: Observing interval (time dependent system) or number of elements in result matrix (time independent system) in our case 12-1. Our result matrix (HTM) consist of 16 elements. Last row is always same (0,0,0,1). So, we will omit last row in matrix (16-4=12 elements – 1 = 11).

-Num_of_int: Number of bits/genes for represent/describe each integer number/chromosome in generation. GA. algorithm converts person to chromosomes in genes (bits). It is very convenient if we use minimum number of bits, but enough for represent the highest number/ chromosome in population. For example: With 8 bits we can represent the highest person 256.

-Num_of_deci: Number of decimal numbers. If we operate with floating point numbers. Usually we operate with integer numbers and this variable must be 0.

-Dovoljena_napaka: Maximum allowed error. GA stop searching for better person when: HTM desired minus HTM is less than dovoljena_napaka.

-Max_interacij: Maximum number of execute while-loop. In case that GA. is not able to find better person. This parameter is usually bigger than 300.

-Verjetnost_mutacije: Possibility for mutation. Usually less than 0.1.

-Povecana_verjetnost: If there is no change in HTM desired minus HTM we increase possibility for mutation for “povecana_verjetnost”

-Povprecje_na: This is number of errors for averaging. Usually less than 10.

-Min_razl_povp: Difference in average error. Usually less than 0.5.

-N_best_brez_mut: Number of persons without possibility for mutation. Usually between 0 and 5.

-Nova_gen_na: Generate new population. If this is variable bigger than “max_interacij” new population will not be generated.

-Odmik: Closest of new population %. Usually between 5 and 20.

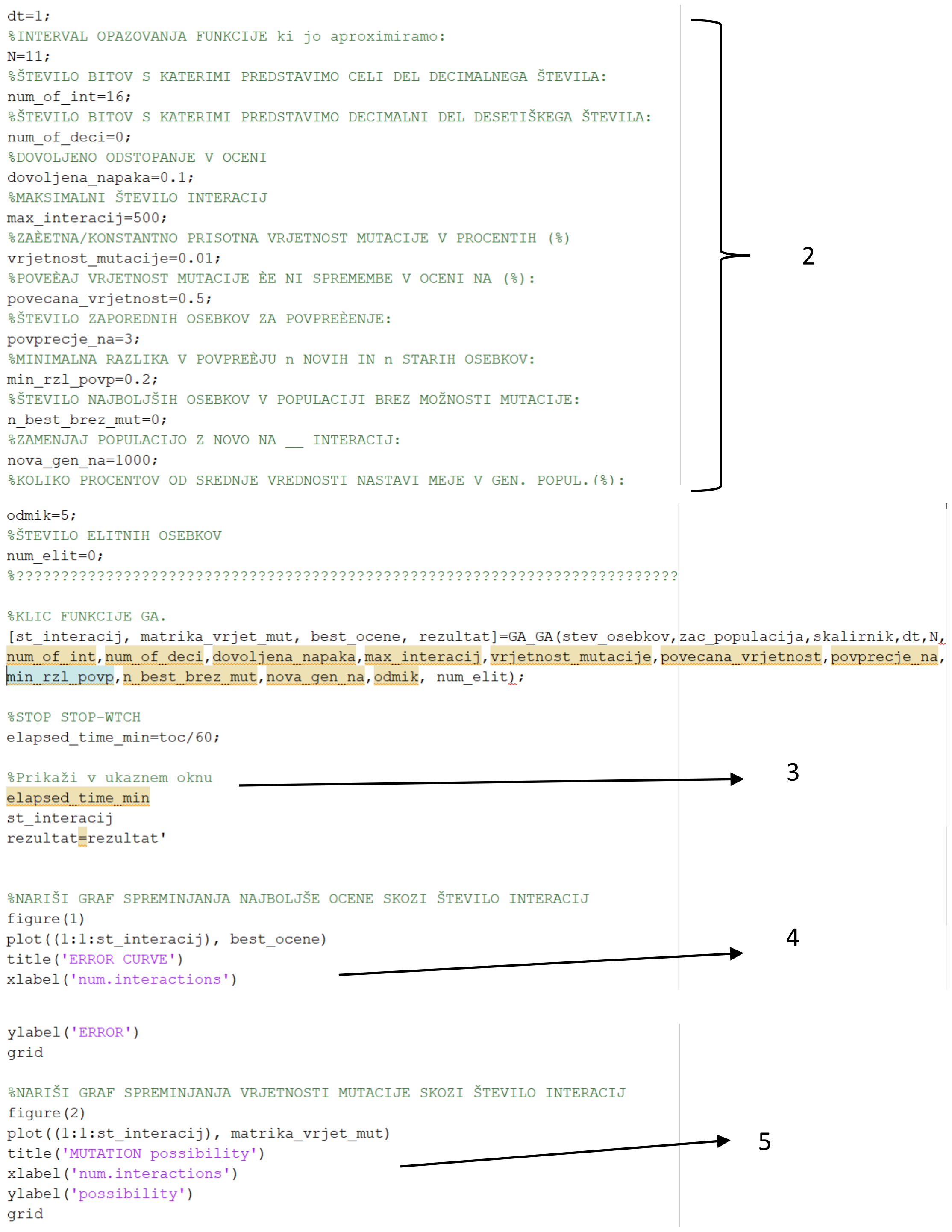
-Num_elit: This parameter is number of elite persons. Usually between 0 and 5.

The third number is indicating the thing that we are going to find in the command window. We can see the time it took, how may while-loops be executed and the final result. The result is going to be a vector with the first number indicating the best fitness function, the smallest error and the q vector. [q1, q2, q3, q4, q5, q6, q7, q8, q9, q10].

Then we have the two graphics we are going to find that are defined in the point 4 and 5. One is going to tell us the error, and the other one the mutation.

```
%Avtor: Kristijan Korez
%Datum: 2.3.2020

%V TEM PRIMERU Z GA. APROXIMIRAM VSE PARAMETRE ESC MODELA BATERIJE

%LASTNOSTI SPODAJ IPLEMENTIRANEGA GA:
%1. ZAŽETNO POPULACIJO LAHKO GENERIRAMO NAKLJUÈNO ZNOTRAJ DOLOÈENIH OKVIROV,
%KI SO ENAKI ZA VSE KROMOSOME ALI 2. UPORABIMO FUNKCIJO, KI GENERIRA
%NAKLJUÈNE KROMOSOME ZNOTRAJ DOLOÈENIH MEJ, OZ. V OKOLICI DOLOÈENE SREDNJE
%VREDNOSTI. KRIŽANJA SE DOGAJAJO NA ENEM MESTU ZNOTRAJ VSAKEGA KROMOSOMA.
%VRJETNOST MUTACIJE PRIÈNE NARAŠÈATI V PRIMERU KADAR SE POVPREÈJE
%DOLOÈENEGA ŠTEVILA OCEN NE SPREMENI ZA DOLOÈENO VREDNOST. MOŽNOST JE TUDI
%UPORABE ELITNIH OSEBKOV. PRAV TAKO JE VKLJUÈENA ZAMENJAVA CELOTNE
%POPULACIJE Z NOVO KO SE PREKORAÈI DOLOÈENO ŠTEVILO INTERACIJ. NOVA
%POPULACIJE JE GENERIRANA ZNOTRAJ MEJ NAJBOLJŠEGA OSEBKA VSEH TRENUTNIH
%INTERACIJ


%KRATKA NAVODILA ZA UPORABO:
%1. ODPREMO: FCN_ISKANA_FCN.m IN NADOMESTIMO OBSTOJEÈO FUNKCIJO S FUNKCIJO,
% KI JO ŽELIMO APROXIMIRATI
%3. GENERIRAMO SMISELNO POPULACIJO
%4. IZBEREMO SMISELNE PARAMETRE GA
%5. POŽENEMO SIMULACIJO

%IZBRIŠEM VSE SPREMENLJIVKE IN POÈISTIM UKAZNO OKNO
clear all
clc

%START STOP-WTCH
tic

%GENERIRAM ZAÈETNO POPULACIJO S POMOÈJO KLICA FUNKCIJE ZA GENERIRANJE POPU.
%center=[srednja vrednost1, max raztros1; srednja vrednost2, max raztros2]
%***************************************************************************
%ŽELJENI/ IZMERJENI PARAMETRI (èe jih poznam, da vem kako nastavim center)

stev_osebkov= 100;
skalirnik= [10000,10000,10000,10000,10000,10000,10000,10000,10000,10000];
center= [-19500,1000; 9500,1000; 0,1000; 14500,1000; -23500,1000; 31000,1000; -31000,1000; 19500,1000;
0,1000; -14500,1000;];

[zac_populacija]=GA_generator_INT_populacije(center, stev_osebkov);
%***************************************************************************


%VNOS PARAMETROV ???????????????????????????????????????????????????????????
%???????????????????????????????????????????????????????????????????????????
%KORAK SAMPLIRANJA:
```

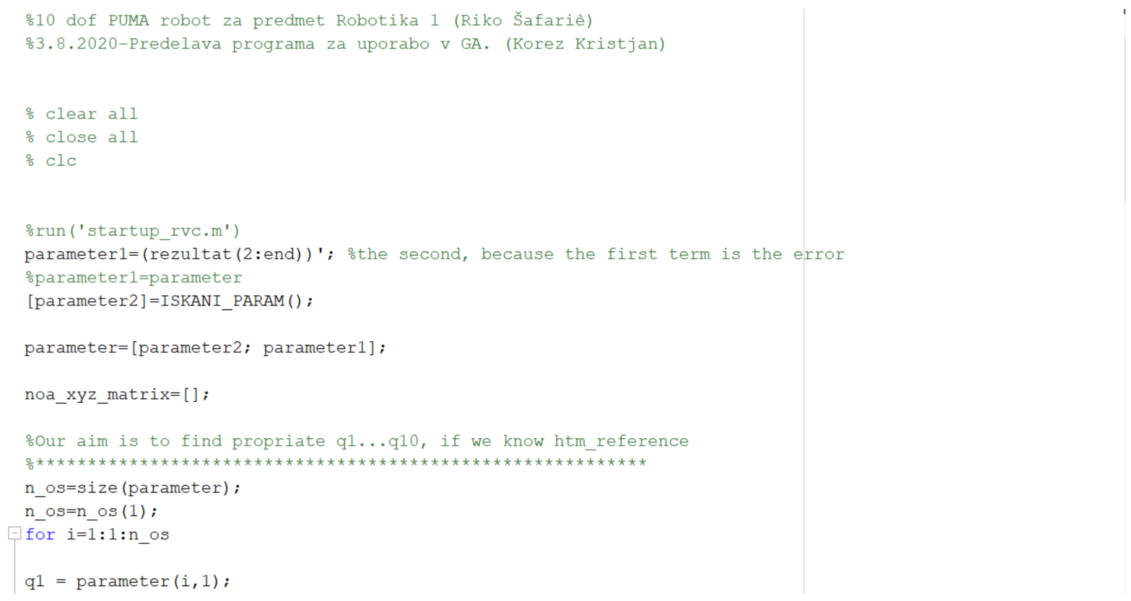
1



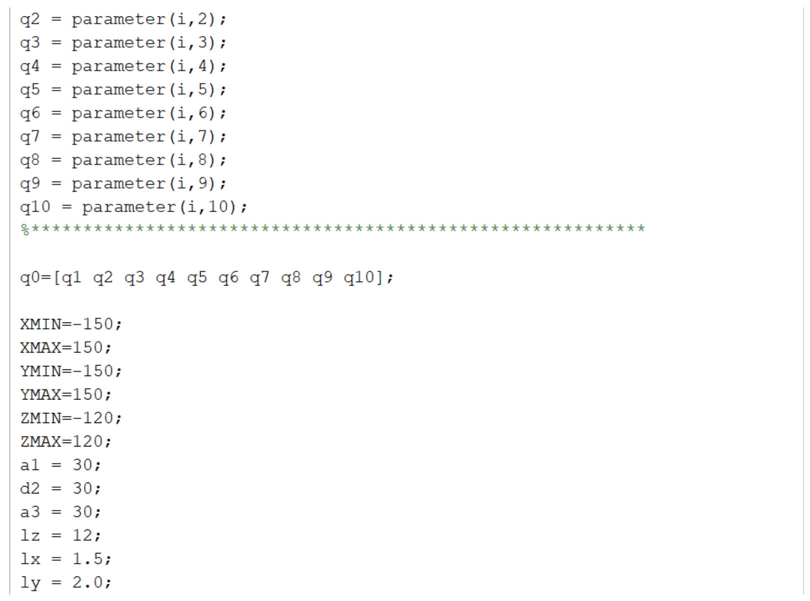
```
dt=1;
%INTERVAL OPAZOVANJA FUNKCIJE ki jo aproximiramo:
N=11;
%ŠTEVILO BITOV S KATERIMI PREDSTAVIMO CELI DEL DECIMALNEGA ŠTEVILA:
num_of_int=16;
%ŠTEVILO BITOV S KATERIMI PREDSTAVIMO DECIMALNI DEL DESETIŠKEGA ŠTEVILA:
num_of_deci=0;
%DOVOLJENO ODSTOPANJE V OCENI
dovoljena_napaka=0.1;
%MAKSIMALNI ŠTEVILO INTERACIJ
max_interacij=500;
%ZAÈETNA/KONSTANTNO PRISOTNA VRJETNOST MUTACIJE V PROCENTIH (%)
vrjetnost_mutacije=0.01;
%POVEÈAJ VRJETNOST MUTACIJE ÈE NI SPREMEMBE V OCENI NA (%):
povecana_vrjetnost=0.5;
%ŠTEVILO ZAPOREDNIH OSEBKOV ZA POVPREÈENJE:
povprecje_na=3;
%MINIMALNA RAZLIKA V POVPREÈJU n NOVIH IN n STARIH OSEBKOV:
min_rzl_povp=0.2;
%ŠTEVILO NAJBOLJŠIH OSEBKOV V POPULACIJI BREZ MOŽNOSTI MUTACIJE:
n_best_brez_mut=0;
%ZAMENJAJ POPULACIJO Z NOVO NA __ INTERACIJ:
nova_gen_na=1000;
%KOLIKO PROCENTOV OD SREDNJE VREDNOSTI NASTAVI MEJE V GEN. POPUL.(%):

odmik=5;
%ŠTEVILO ELITNIH OSEBKOV
num_elit=0;
%?????????????????????????????????????????????????????????????????????????

%KLIC FUNKCIJE GA.
[st_interacij, matrika_vrjet_mut, best_ocene, rezultat]=GA_GA(stev_osebkov,zac_populacija,skalirnik,dt,N,
num_of_int,num_of_deci,dovoljena_napaka,max_interacij,vrjetnost_mutacije,povecana_vrjetnost,povprecje_na,
min_rzl_povp,n_best_brez_mut,nova_gen_na,odmik, num_elit);

%STOP STOP-WTCH
elapsed_time_min=toc/60;

%Prikaži v ukaznem oknu
elapsed_time_min
st_interacij
rezultat=rezultat'


%NARIŠI GRAF SPREMINJANJA NAJBOLJŠE OCENE SKOZI ŠTEVILO INTERACIJ
figure(1)
plot((1:1:st_interacij), best_ocene)
title('ERROR CURVE')
xlabel('num.interactions')

ylabel('ERROR')
grid

%NARIŠI GRAF SPREMINJANJA VRJETNOSTI MUTACIJE SKOZI ŠTEVILO INTERACIJ
figure(2)
plot((1:1:st_interacij), matrika_vrjet_mut)
title('MUTATION possibility')
xlabel('num.interactions')
ylabel('possibility')
grid
```

Figure 11. GA_KLIC_GA_FCN

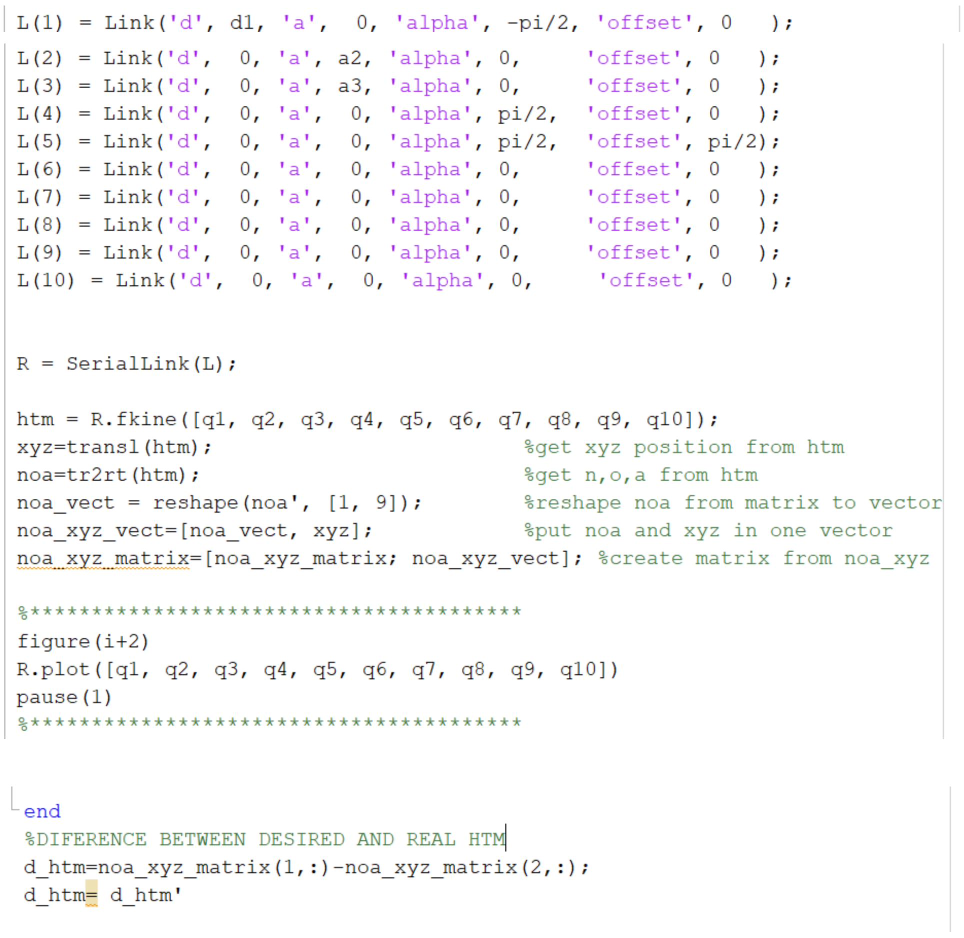
We will now explain the next and final code. Basically, in this code we are going to find a part more related to the robotics. It is where using all the previous data we obtain our final result. It is the last step of the whole program. It will show us the definitive results in a more visual way. The previous code already showed us the most important results that are the end of the algorithm, so the end of the optimization with the values of the qs, also showed us the minimum error obtained. This code has like a result the representation of the robotic arm directly. We can see the final position in 3D. Also now the command window show us d_htm, this is the difference between htm desired and htm aprox.



```
%10 dof PUMA robot za predmet Robotika 1 (Riko Šafariè)
%3.8.2020-Predelava programa za uporabo v GA. (Korez Kristjan)


% clear all
% close all
% clc


%run('startup_rvc.m')
parameter1=(rezultat(2:end))'; %the second, because the first term is the error
%parameter1=parameter
[parameter2]=ISKANI_PARAM();

parameter=[parameter2; parameter1];

noa_xyz_matrix=[];

%Our aim is to find propriate q1...q10, if we know htm_reference
%************************************************************
n_os=size(parameter);
n_os=n_os(1);
for i=1:1:n_os

q1 = parameter(i,1);
```



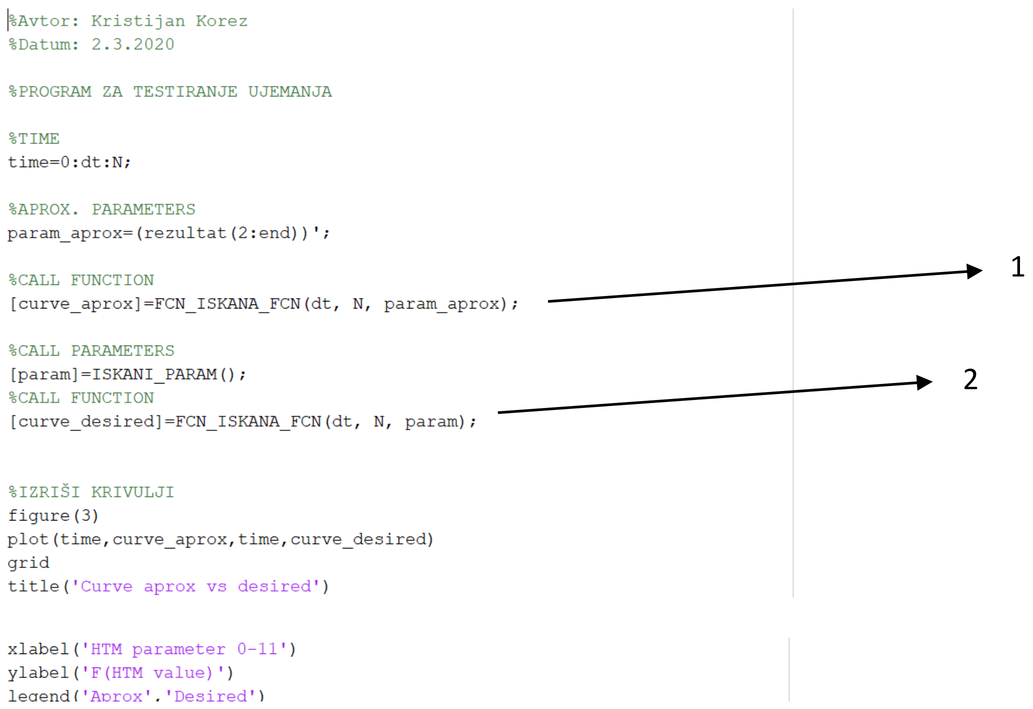
```
q2 = parameter(i,2);
q3 = parameter(i,3);
q4 = parameter(i,4);
q5 = parameter(i,5);
q6 = parameter(i,6);
q7 = parameter(i,7);
q8 = parameter(i,8);
q9 = parameter(i,9);
q10 = parameter(i,10);
%************************************************************

q0=[q1 q2 q3 q4 q5 q6 q7 q8 q9 q10];

XMIN=-150;
XMAX=150;
YMIN=-150;
YMAX=150;
ZMIN=-120;
ZMAX=120;
a1 = 30;
d2 = 30;
a3 = 30;
lz = 12;
lx = 1.5;
ly = 2.0;
```



```
L(1) = Link('d', d1, 'a',  0, 'alpha', -pi/2, 'offset', 0   );
L(2) = Link('d',  0, 'a', a2, 'alpha', 0,     'offset', 0   );
L(3) = Link('d',  0, 'a', a3, 'alpha', 0,     'offset', 0   );
L(4) = Link('d',  0, 'a',  0, 'alpha', pi/2,  'offset', 0   );
L(5) = Link('d',  0, 'a',  0, 'alpha', pi/2,  'offset', pi/2);
L(6) = Link('d',  0, 'a',  0, 'alpha', 0,     'offset', 0   );
L(7) = Link('d',  0, 'a',  0, 'alpha', 0,     'offset', 0   );
L(8) = Link('d',  0, 'a',  0, 'alpha', 0,     'offset', 0   );
L(9) = Link('d',  0, 'a',  0, 'alpha', 0,     'offset', 0   );
L(10) = Link('d',  0, 'a',  0, 'alpha', 0,     'offset', 0   );


R = SerialLink(L);

htm = R.fkine([q1, q2, q3, q4, q5, q6, q7, q8, q9, q10]);
xyz=transl(htm);                         %get xyz position from htm
noa=tr2rt(htm);                          %get n,o,a from htm
noa_vect = reshape(noa', [1, 9]);        %reshape noa from matrix to vector
noa_xyz_vect=[noa_vect, xyz];            %put noa and xyz in one vector
noa_xyz_matrix=[noa_xyz_matrix; noa_xyz_vect]; %create matrix from noa_xyz

%****************************************
figure(i+2)
R.plot([q1, q2, q3, q4, q5, q6, q7, q8, q9, q10])
pause(1)
%****************************************


end
%DIFERENCE BETWEEN DESIRED AND REAL HTM
d_htm=noa_xyz_matrix(1,:)-noa_xyz_matrix(2,:);
d_htm= d_htm'
```

Figure 12. ROB_10dof_puma

With the last step, which is not necessary since we already have the solutions we wanted, we see that among the codes that make up this program before we have also named the code 'TEST_UJEMANJA'. This program draw plot. This plot represents HTM parameters – the robot tip position and orientation (12 parameters). This plot compares two HTM (desired HTM and HTM as result of GA.)



```
%Avtor: Kristijan Korez
%Datum: 2.3.2020

%PROGRAM ZA TESTIRANJE UJEMANJA

%TIME
time=0:dt:N;

%APROX. PARAMETERS
param_aprox=(rezultat(2:end))';

%CALL FUNCTION
[curve_aprox]=FCN_ISKANA_FCN(dt, N, param_aprox);

%CALL PARAMETERS
[param]=ISKANI_PARAM();
%CALL FUNCTION
[curve_desired]=FCN_ISKANA_FCN(dt, N, param);


%IZRIŠI KRIVULJI
figure(3)
plot(time,curve_aprox,time,curve_desired)
grid
title('Curve aprox vs desired')


xlabel('HTM parameter 0-11')
ylabel('F(HTM value)')
legend('Aprox','Desired')
```

Figure 13: TEST_UJEMANJA

5. RESULTS AND DISCUSSION

Let's move on to work with the code and take a good look at the results and the graphics it generates. We follow the steps indicated in the previous section. In the first code we only have to make sure that it is correct and save it.

In the second one we will introduce the parameters that we want. In my case I have introduced a vector formed by 10 parameters, since we are working with a robot with 10 degrees of freedom. I have introduced: [2.3, -2, 1.5, 0, pi, -pi, 0, 1, -2.2, 1]. We save and close the function.

We go on to the most important part and where we have to be careful, we introduce the scaler value, in a vector also of 1x10. The scale value is 1000. And we go on to define the ranges of the parameters that we have just introduced. We will do it in the way previously explained.

2.3	$\frac{23000 \pm 1000}{10000} = [2.2, 2.4]$	-pi	$\frac{-31000 \pm 1000}{10000} = [-3.2, -3]$
-2	$\frac{-19500 \pm 1000}{10000} = [-2.05, -1.85]$	0	$\frac{0 \pm 1000}{10000} = [-0.1, 0.1]$
1.5	$\frac{14500 \pm 1000}{10000} = [1.35, 1.55]$	1	$\frac{9500 \pm 1000}{10000} = [0.85, 1.05]$
0	$\frac{0 \pm 1000}{10000} = [-0.1, 0.1]$	2.2	$\frac{22000 \pm 1000}{10000} = [2.1, 2.3]$
pi	$\frac{31000 \pm 1000}{10000} = [3, 3.2]$	1	$\frac{9500 \pm 1000}{10000} = [0.85, 1.05]$

Table 7. Ranges of the values of q.

After defining these ranges we will define the parameters of the algorithm. To the sample time we will assign a value of 1, to value of N we will put an 11, since it is the number of parameters that the matrix has minus one, we know that it consists of 16 but 4 of them are always the same and we do not count them. We are left with 12, which by subtracting one is 11.

The next parameters to define are num_of_int and num_of_deci. These parameters define the way to pass the number (member of the population) to binary. We are going to put 16 in the first parameter, so we are defining the number of bits for each number. We can represent the highest person like 2^{16}. The second paramet must be 0, because we are not going to operate with floating numbers.

The following parameters that we find are the ones that we can change the most. The error allowed will be set to 0.1 and the maximum of iterations to 500. This value does not define exactly the iterations that the program will do, but it defines the maximum ones. It serves to put a limit, but if the algorithm reaches an acceptable solution before, it will also stop. If the reason to stop is that the program has finish and has an error less than 0.1 (the stablished value) we will have a very good solution, if the program has stop because of the maximum iteration we did not find desired result. So, we can run it again and search for a better solution.

The results we get are as follows. The first prove give us:

```
elapsed_time_min =

   24.1838


iteration =

   158


result =

    0.0992
    2.3000
   -1.9948
    1.4939
    0.0336
    3.1411
   -3.1429
    0.0064
    0.9534
   -2.1779
    0.9972
```

Table 8: First results.

As we can see the program has needed 158 iterations, a very low number comparing it with the 500 interactions that could be done. So, the algorithm stop because it get an error less than 0.1.

Then we found the vector data. Let's analyse it a little. The first data is 0.0992, which is the exact error at the iteration 158.

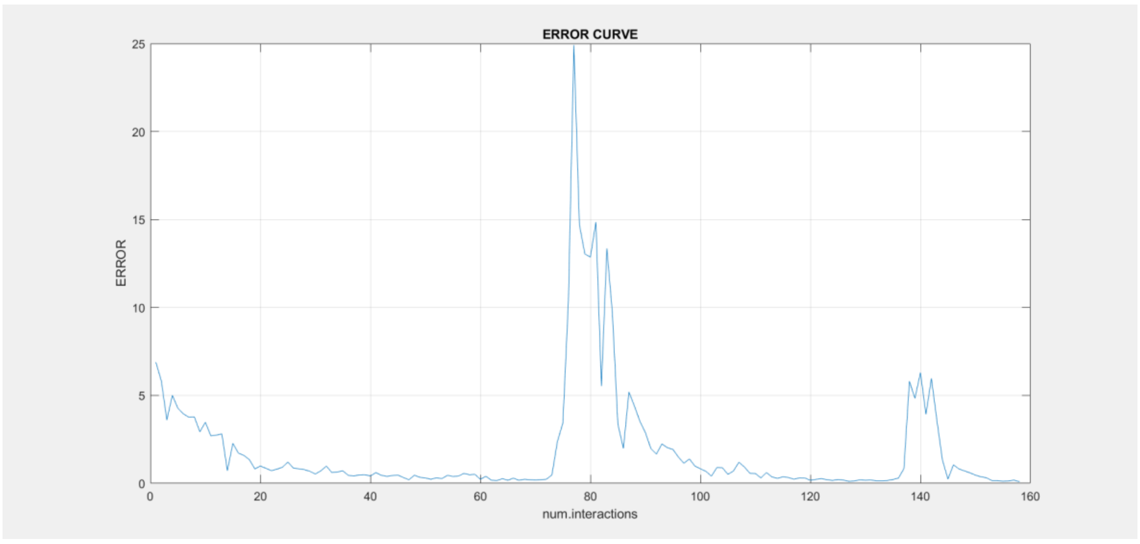


Figure 14: First results. Error curve.

We have a graph that represents the error that has been produced while the program was being computed. We see that it is represented by the error in the ordinate axis and the number of iterations in the abscissa axis.

We see that the curve ends up being very low. A value less than 0.1. Is the value that the previous vector of data gives us, the error in the last iteration is 0.09919. It is the lower point of the graph.

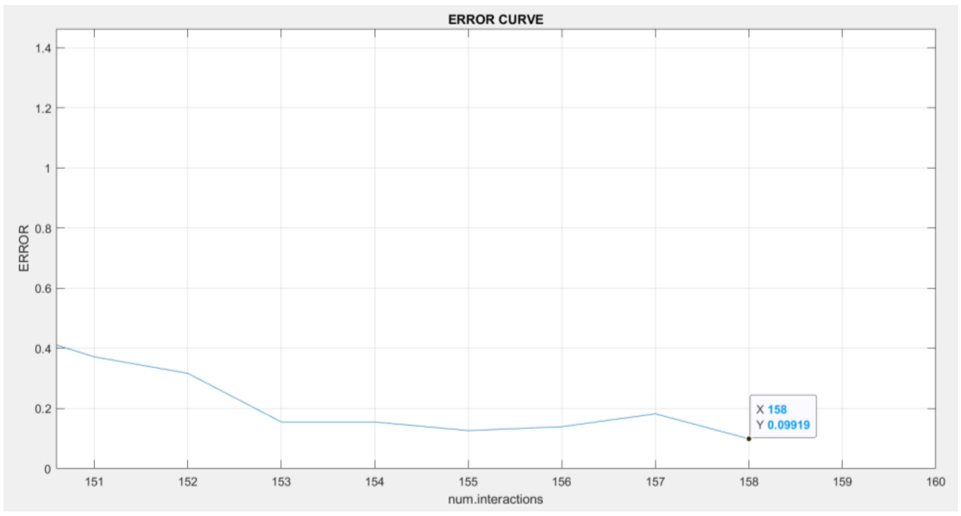


Figure 15. Final error of the last iteration.

Next, it also shows us a graph about the percentage of mutation that has been produced. In this case we have the percentage of mutation on the ordinate axis and as in the previous one in the number of iterations on the abscissa axis.

Previously we had set the percentage of mutation at 0.01. With this value established and during the optimization we have obtained a mutation curve like the one we see in the figure 15.

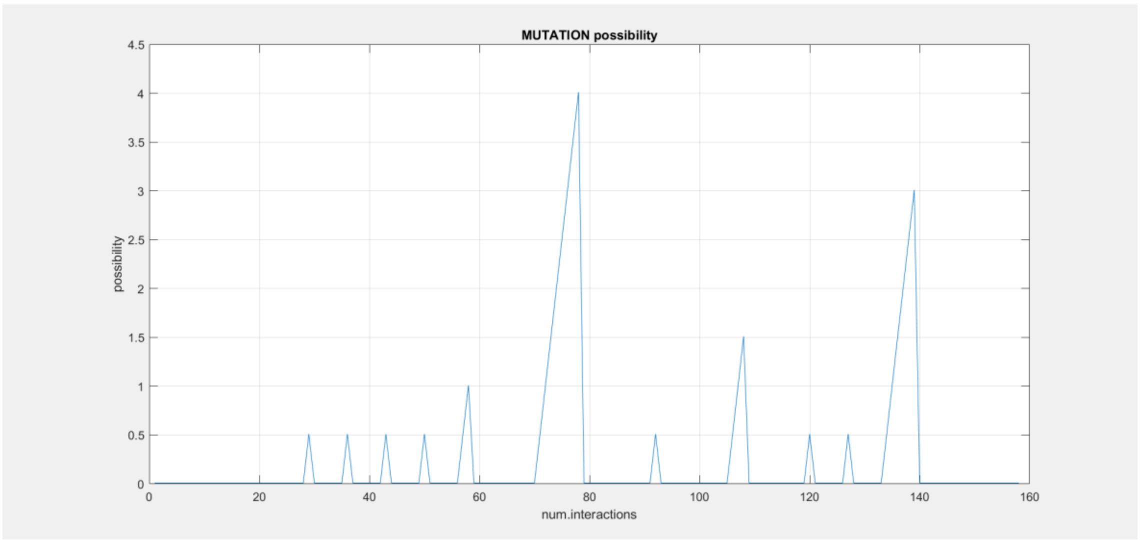
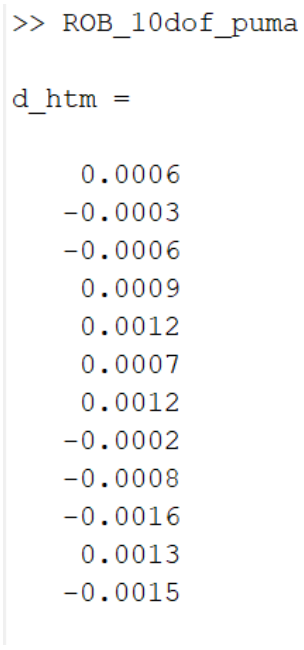


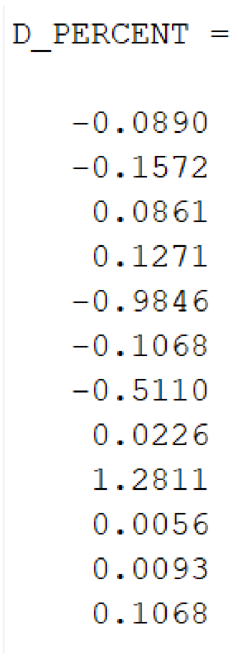
Figure 16. First results, Mutation possibility curve.

At this point, we will use the program ROB_10dof_puma. Computing this program we have already defined both htm matrices. And the subtraction of both gives us the following result. We will find it in Matlab's workspace. In this case we are obtaining very good results. All the values are very low. That's mean that the different between both htm matrix (the desired one and the initial one) are low, so they are quite similar. The algorithm made a good job.



```
>> ROB_10dof_puma

d_htm =

    0.0006
   -0.0003
   -0.0006
    0.0009
    0.0012
    0.0007
    0.0012
   -0.0002
   -0.0008
   -0.0016
    0.0013
   -0.0015
```

Table 9. Htm desided matrix minus htm aprox matrix.



```
D_PERCENT =

   -0.0890
   -0.1572
    0.0861
    0.1271
   -0.9846
   -0.1068
   -0.5110
    0.0226
    1.2811
    0.0056
    0.0093
    0.1068
```

Table 10. Htm desided matrix minus htm aprox matrix in percent.

Also in matlab workspace we can see this vector. The D_PERCENT vector indicates the same as the other vector but in percent (%). The values are quite accurate and correct. We can now analyse a little bit the other data that the code gives us when we compute it.

As a result of this code we also found represented in 3D we will see the final position that the robotic arm acquires. Thanks to Matlab's applications and functions we can rotate, zoom in and out of the image to better appreciate the position. This can be seen in figures 16 and 17.

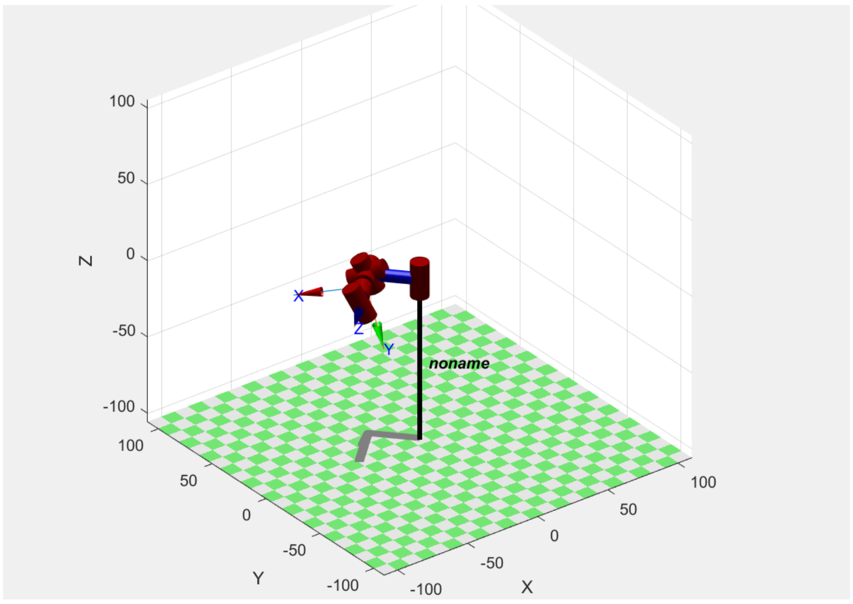


Figure 17. 3D final position os the robot arm.

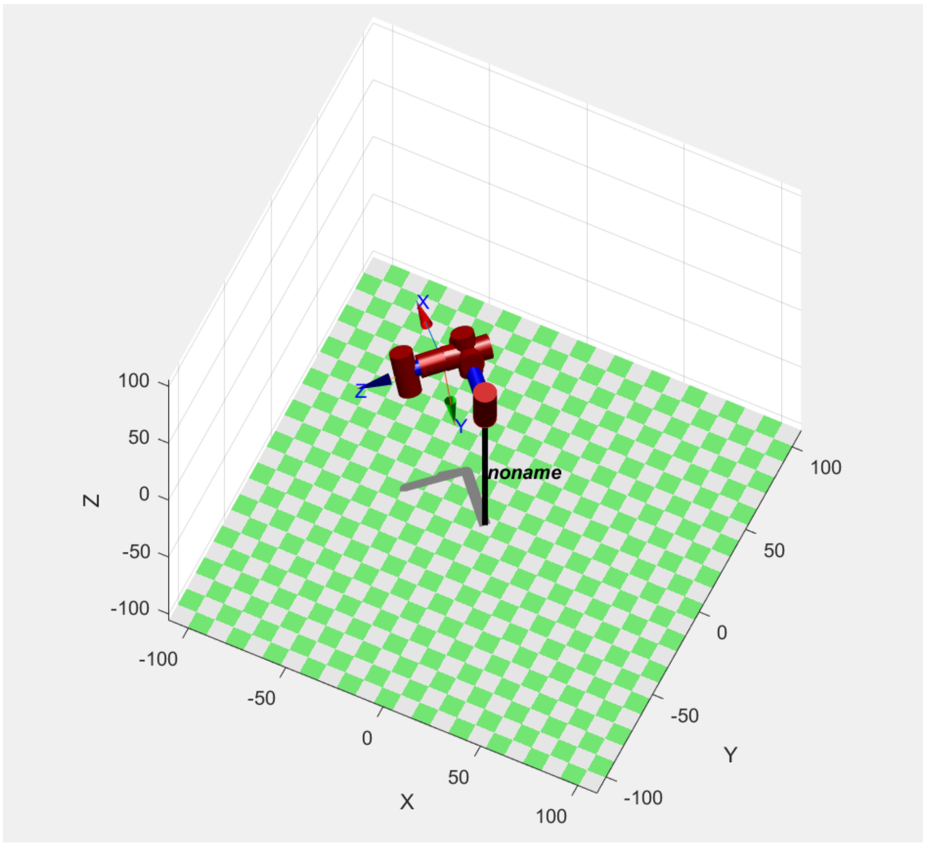


Figure 18. 3D robot arm.

In the last two figures we find below we will see two graphics. They are the same but the second one is the first ones with a very big zoom on it. We are talking about figures

17 and 18. As we see in the title of the graphs, it is the graphical representation (a curve) of the desired and the approximated function of the htm matrix. We see that they are practically the same. When we approach them, we can find that they are not 100% equal but that the error between them is minimal. We have obtained this graph by computing the code TEST_UJEMANJA.

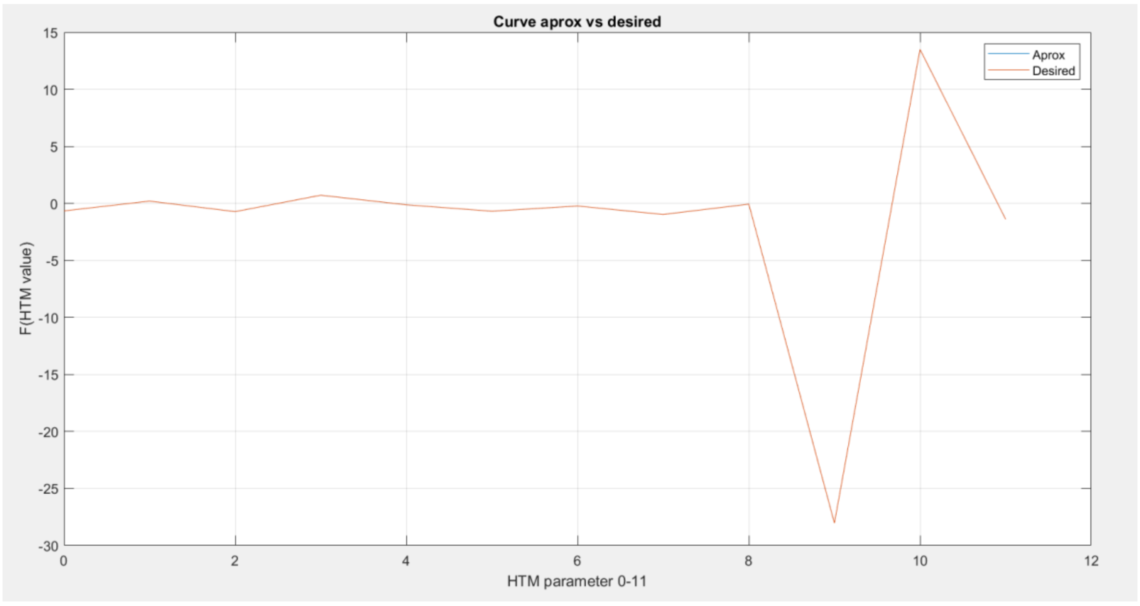


Figure 19. Comparation of htm aprox matix vs htm desired matrix.

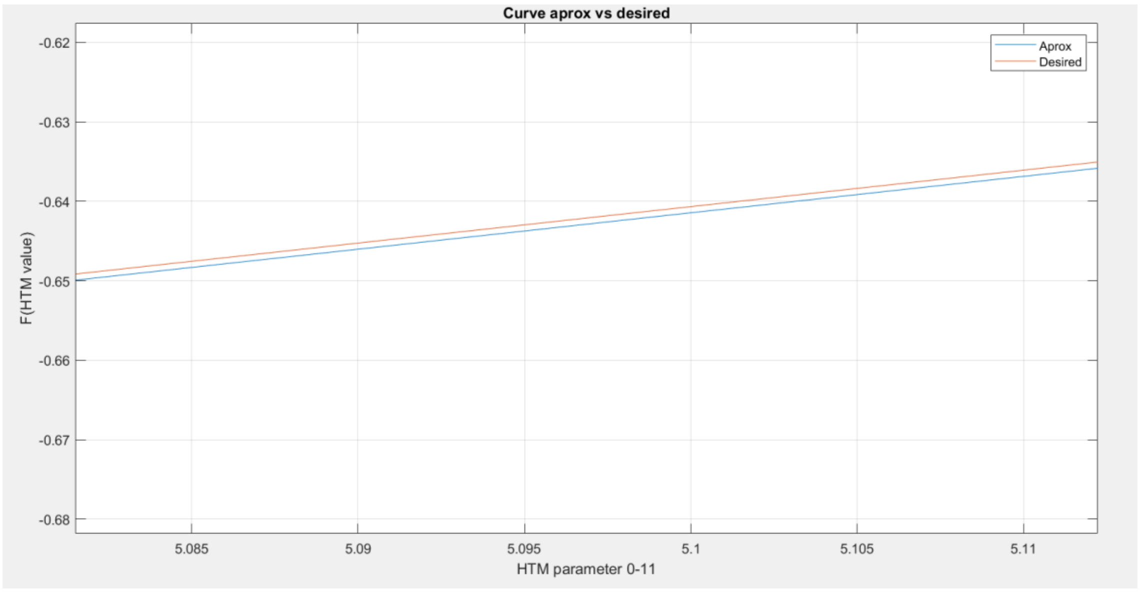


Figure 20. Amplification of the previous curve.

I am going to present a couple of examples to better explain the algorithm. I am going to kept working with others parameters in q ([-2.1, -1.2, 0, 1, 2, pi, -pi, 1, 1.5, 2]). With this data I kept using the code until the end. They are the next ones, with 500 iterations and all the other parameters with the values we have already said before. In this case we can see the program has made 281 iterations. It got an error of 0.0903. Also in the result vector we can see the parameter that have the q vector now. The robotic function is going to work with it.

```
elapsed_time_min =

   34.3138


iteration =

   281


result =

    0.0903
   -2.1128
   -1.2577
   -0.0225
    0.9427
    2.0090
   -3.1414
   -3.0756
    0.9509
    1.5463
    2.0423
```

Table 11. Seconds results.

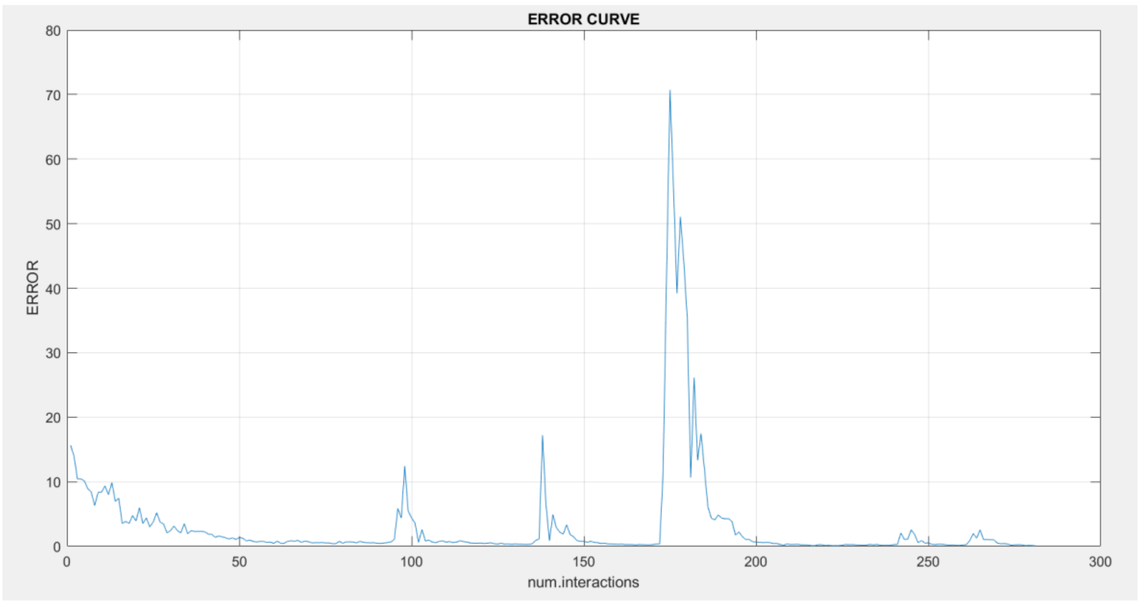


Figure 21. Error curve.

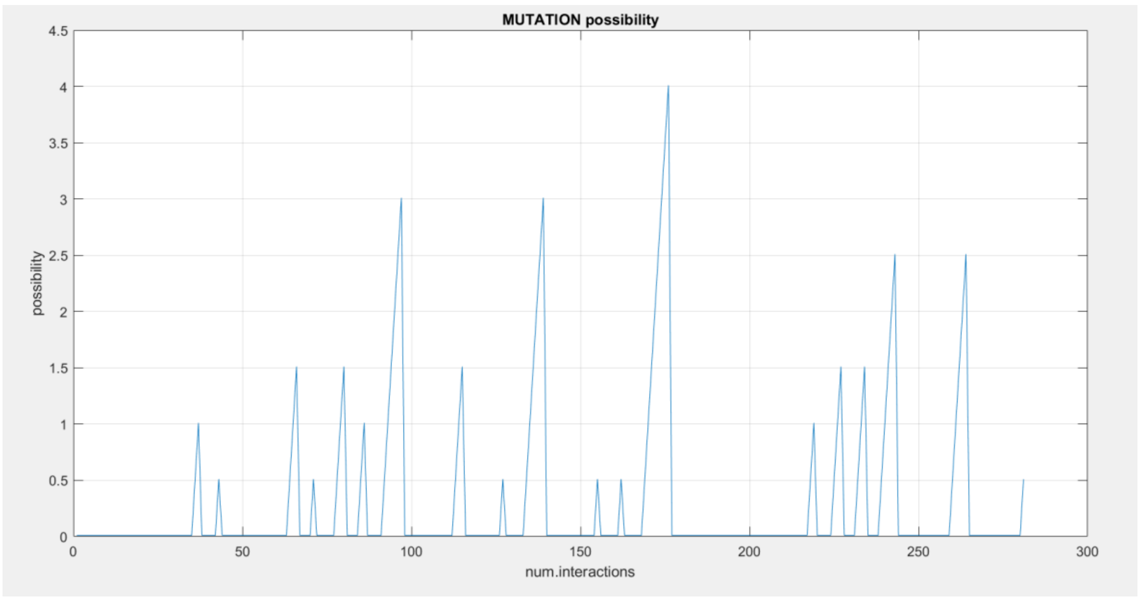


Figure 22. Mutation possibility curve.

In this case the different between the htm desired and the htm the approx. are the following ones. We have found very good results because the values are very low. We have to record that the htm matrix is only the tip position and the robot tip orientation, that's why the q values, the ones of the positions of the other parts of the robot, are different. So, the q parameters that we have introduce and q values the GA algorithm are different. The results are that we can have several poses of the robot arm which are all corrects, and all with the same final position of the tip.

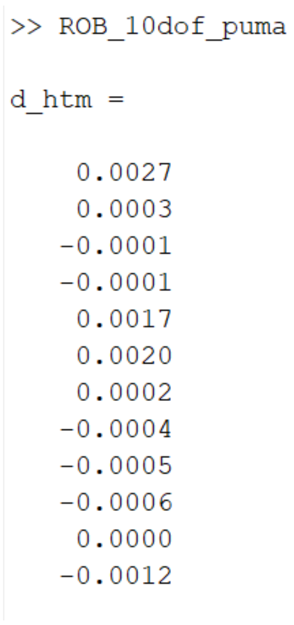
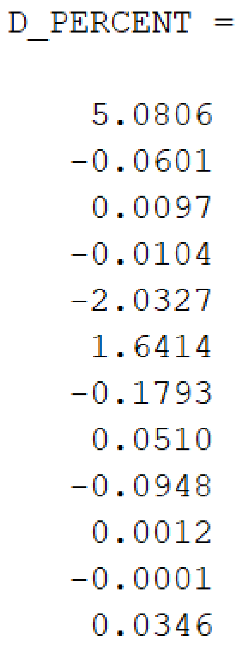

```
>> ROB_10dof_puma

d_htm =

    0.0027
    0.0003
   -0.0001
   -0.0001
    0.0017
    0.0020
    0.0002
   -0.0004
   -0.0005
   -0.0006
    0.0000
   -0.0012
```

Table 12. Htm desired matrix minus htm approximation matrix.



```
D_PERCENT =

    5.0806
   -0.0601
    0.0097
   -0.0104
   -2.0327
    1.6414
   -0.1793
    0.0510
   -0.0948
    0.0012
   -0.0001
    0.0346
```

Table 13. Htm desired matrix minus htm approximation matrix. In percent.

On the following figures, like in the case presented before we can see in 3D the robot arm. Each cylinder represents one joint. In the picture we need to have 10 cylinders.

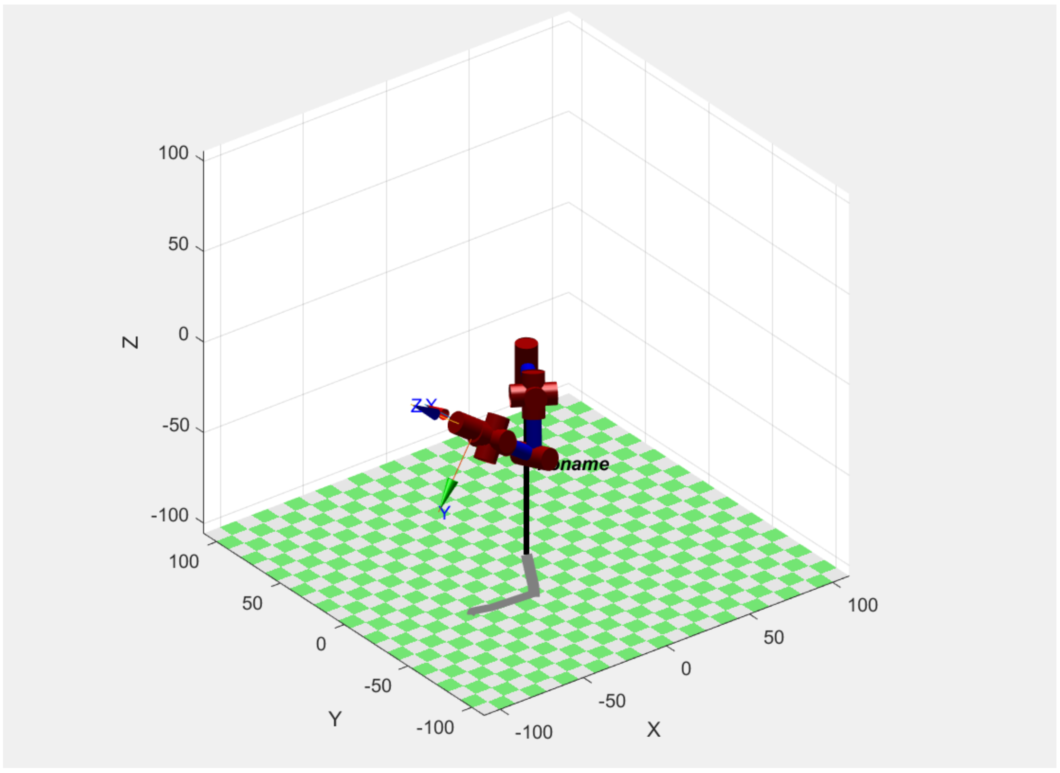


Figure 23. Position of the robotic arm in 3D.

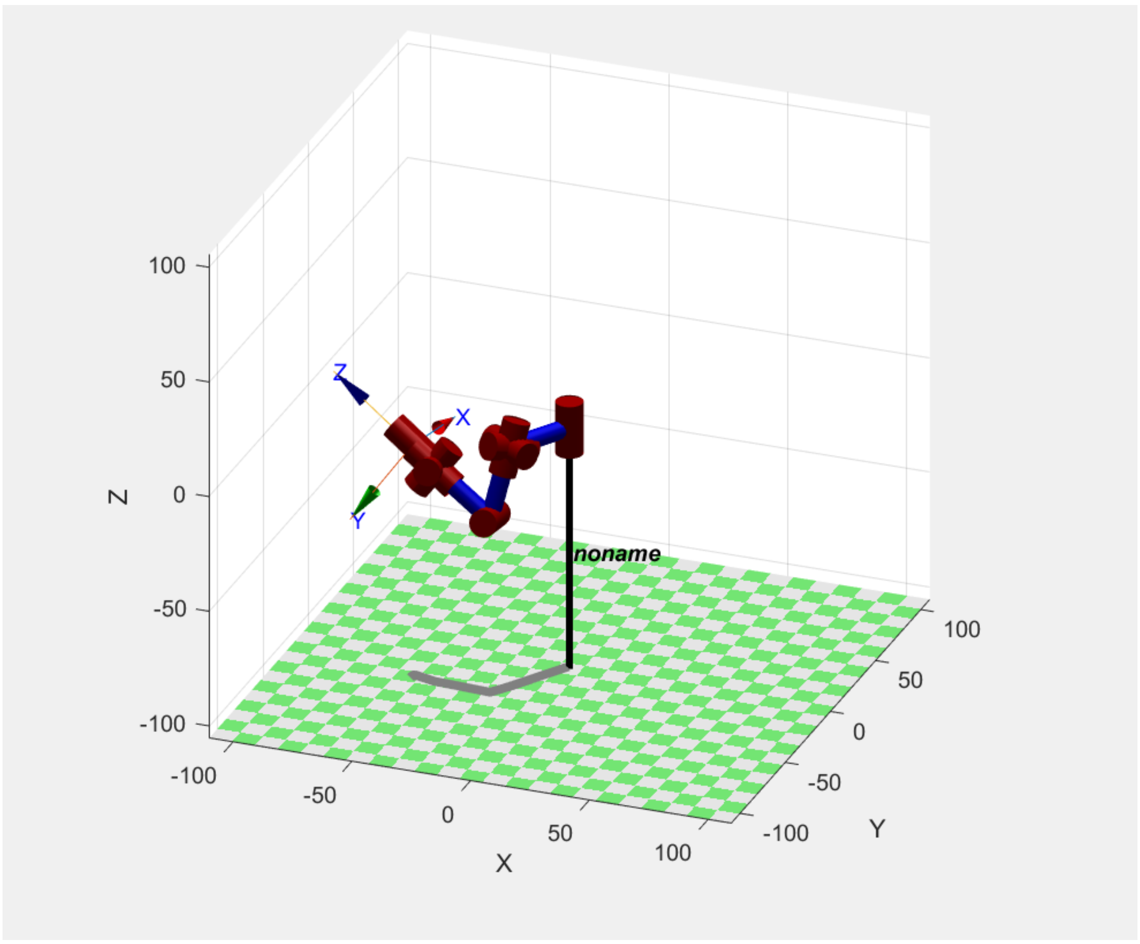


Figure 24. Position of the robotic arm in 3D.

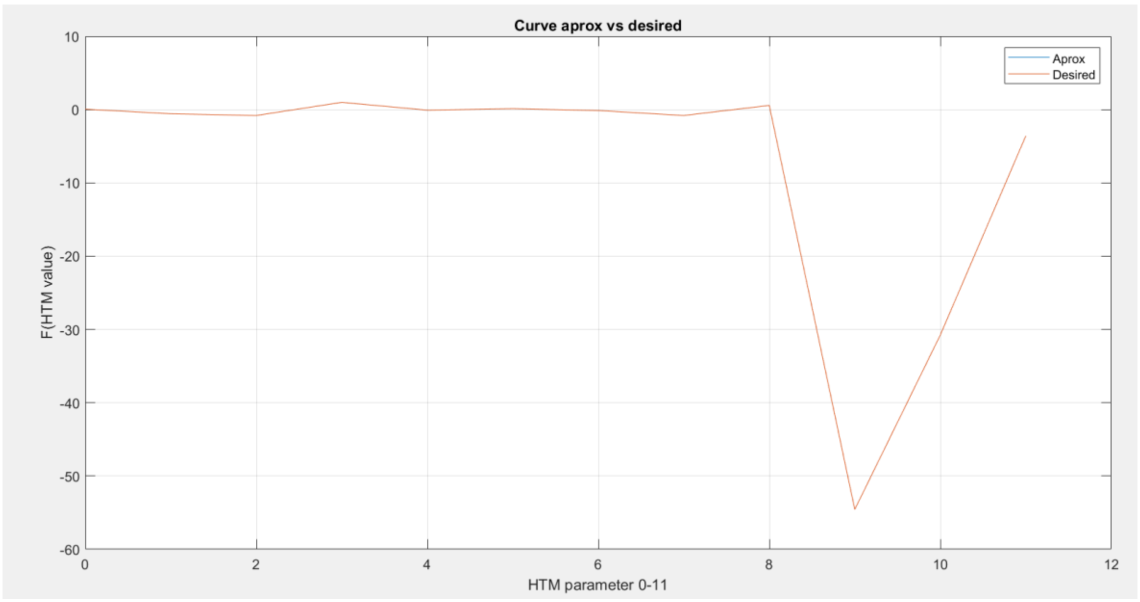


Figure 25. Curve aprox. vs desired.

As a last example to see the different cases that can happen to us using this optimisation method, we are going to show the following one. As we see in our first screenshot of the MATLAB workspace, we find that it has stopped the method after performing the 500 iterations. This indicates that we have not achieved the expected result. In a beginning we should run again the functions to do the GA algorithm I find us another solution. I have left all the steps of this method because in the last images where we find the robotic arm it is very clear that it is formed by 10 degrees of freedom. We can see very good thee ten red cylinders and their position. We can also analyse a little bit the data we get from the differences between the approximate matrix and the one we want. In the previous case we have seen that the percentage of error was very very low. And now we find slightly higher values. Normally the vector d_hmt has values in the order of 10^{-3} and now we can find values in the order of 10^{-2}. And the vector D_PERCENT usually has percentages of value 10^{-1} and in this example 10^{0}.

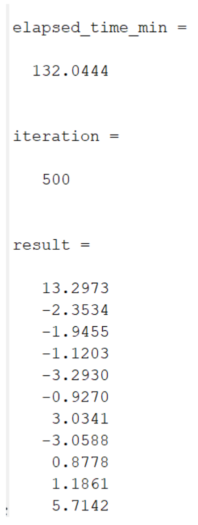

```
elapsed_time_min =

  132.0444


iteration =

   500


result =

   13.2973
   -2.3534
   -1.9455
   -1.1203
   -3.2930
   -0.9270
    3.0341
   -3.0588
    0.8778
    1.1861
    5.7142
```

Table 14. Third results.

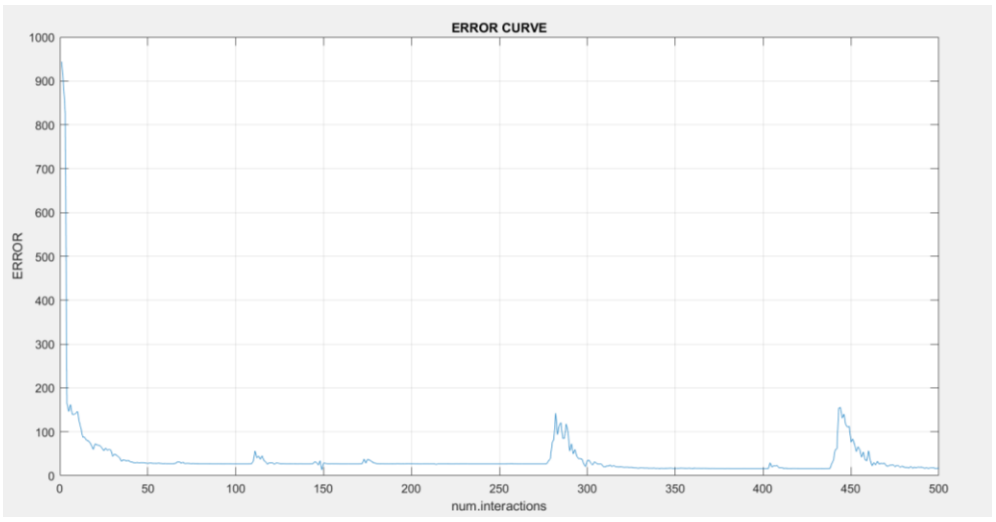


Figure 26. Error curve of the third results.

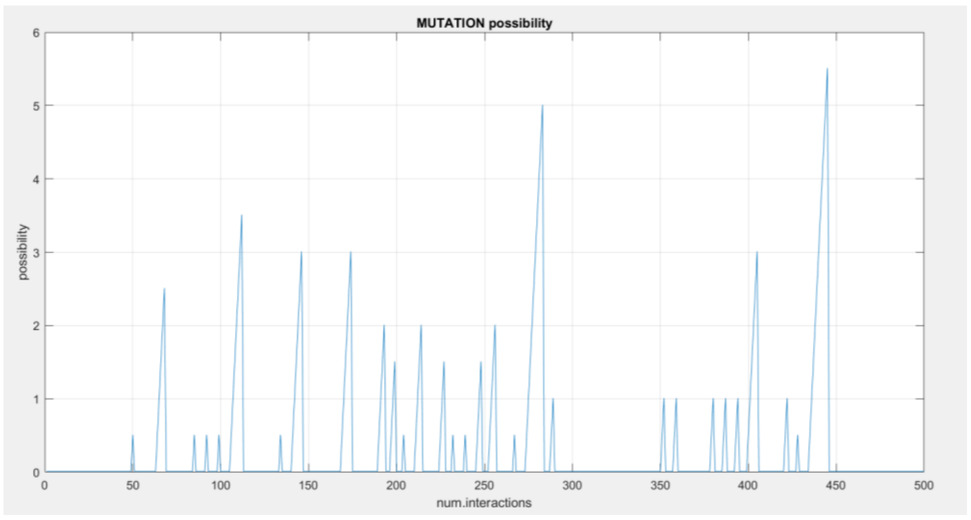


Figure 27. Mutation posibility curve of the third results.

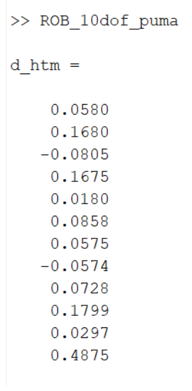

```
>> ROB_10dof_puma

d_htm =

    0.0580
    0.1680
   -0.0805
    0.1675
    0.0180
    0.0858
    0.0575
   -0.0574
    0.0728
    0.1799
    0.0297
    0.4875
```

Table 15. D_htm vector. (Third results.)

```
D_PERCENT =

    -9.3326
  -361.1173
    10.3004
   330.0293
     1.8133
   -86.2032
     7.3619
    56.5565
   -11.8302
    -0.2701
    -1.5405
    -1.1881
```

Table 16 D_PERCECNT vecctor.. (Third results.)

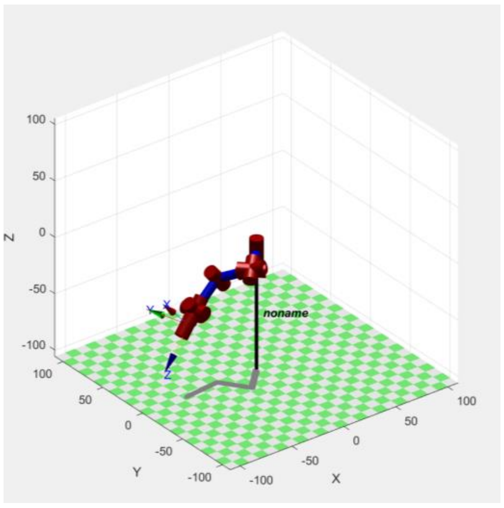


Figure 28. Position of the robotic arm. (Third results)

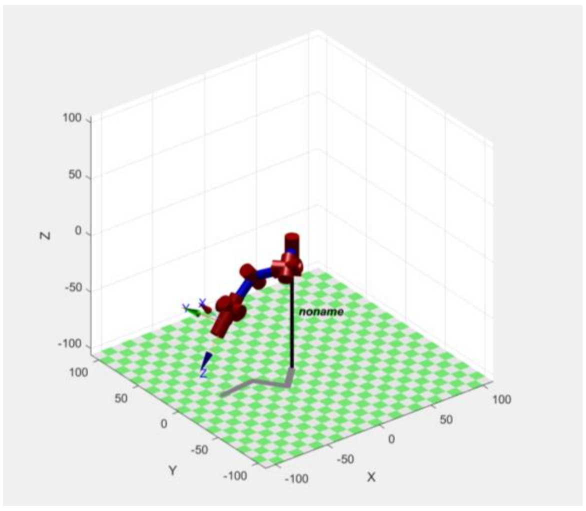


Figure 29. Position of the rorobitc arm. (Third results)

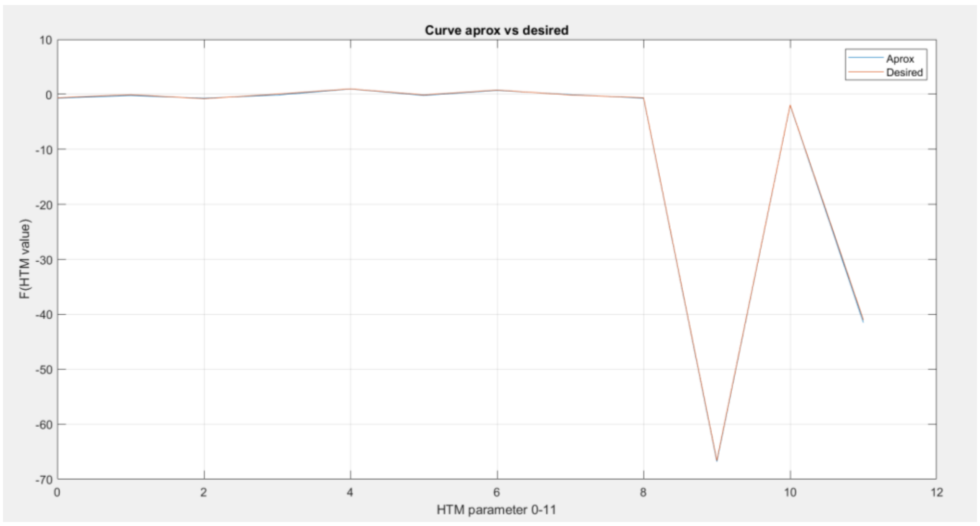


Figure 30. Curve aprox. vs desired. (Third results)

6. CONCLUSION

At the end we will take an overview of our work.

With the realization of this project we can say that we learn a lot about several different topics. Starting with simply robotics, focusing more on the robotic arms. And within this modality in the number of tools that MATLAB must define its operation and behaviour. We have investigated in the way of using this tool to learn how with a code we can completely define its position, as much in reference to a robotic arm of 6 degrees of freedom, like but advanced the project for one of ten degrees of freedom.

Having learned this part, we move on to the optimization method part. The genetic algorithm. We have seen how over time optimization methods based on the genetic algorithm have emerged and why these genetic processes are related to mathematics and want to copy them to perfection. At the beginning we have learned from their implementation on paper and pen, with simple examples. Little by little we have introduced more difficulties such as the percentage of mutation and the difficulty of functions and the choice of more complex population such as what may seem as simple as adding negative or decimal numbers. Controlled once the method on the paper, analytically we have happened to learn it by means of a whole set of programs that in their totality defined the optimization method to the perfection. We have known how to use this program and these Matlab codes and the use and the way to vary the most important and significant parameters.

Once these objectives have been achieved, it has been possible to learn how to use its fusion to define the position of a robotic arm with 10 degrees of freedom. Achieving the main objective of this project.

Currently we know how to use this method of optimizing functions to get to know the position of the robotic arm, and not only know it but define it ourselves. And we know how to interpret the results we get as we apply the different codes.

The results as we have seen above are very good, the error made in the result of the example we have exposed and explained is minimal.

So the final and summarized conclusion is that I have learned a lot by doing this project. I have had to make an effort and introduce myself into new fields that I have never worked on so thoroughly before. I have met a very interesting new branch of engineering and we have managed to reach the main objectives successfully.

7. REFERENCES

[1] Peter Cork, "Robotics, Vision and control, Fundamental algorithms in MATLAB".Springer. Star springer tracts in advanced robotics. First edition 2011, corrected second printing 2013.

[2] Peter Cork, "Matlab commands" http://www.petercorke.com/RTB/r9/html/index.html . 1990-2011

[3] Prof. Dr. Riko Šafarič and Asis. Dr. Andreja Rojko, University of Maribor, "Intelligent control techniques in Mechatronics – Genetic algorithm". Ebook.

[4] The MathWorks: www.mathworks.es

[5] Prof. Fernando Viadero and Prof. Pablo Garcia Fernandez, "Dynamics and cinematics of machines". Notes of the subject of the University of Cantabria. 2019.

[6] Prof. Ramon San Cibrián Herrera, "Kinematics of Machines and Mechanisms". University of Cantabria. 2019.

[7] http://www.etitudela.com/profesores/rpm/rpm/downloads/robotica.pdf

[8] Matlab help

[9] Jose Cortes Parejo, “La representación Denavit-Hartenberg”. 2008.

https://personal.us.es/jcortes/Material/Material_archivos/Articulos%20PDF/RepresentDH.pdf

[10] Barrientos, A, Peñín, L.F, Balaguer, C. & Aracil, R. “Fundamentos de Robótica 2ª”. Ed. McGraw-Hill, 2007

[11] Fu, K.S.; González, R.C. & Lee, C.S.G. “Robótica: Control, detección, visión e inteligencia”. McGraw-Hill, 1988