UNIVERSITETET I AGDER

Error Kinematic Modeling and Calibration of the 3-DOF Gantry-Tau Parallel Kinematic Machine

by

Karam Mady

August 3, 2012

Supervisors:
Geir Hovland
Ilya Tyapin

Department of Engineering

Faculty of Technology and Science

Contents

Chapter 1

Introduction

1.1 Aknowledgement

I would like to thank, Geir Hovland and Ilya Tyapin of The University of Agder, Norway for their contributions to this work. In addition, I want to acknowledge the support from The University of Agder for providing the resources needed to optimize the Gantry-Tau presented in this Thesis.

1.2 Abstract

This thesis presents the results of an investigation into modeling the kinematic errors of the Gantry-Tau Parallel Kinematic Manipulator. The calculations has been done onto the Gantry-Tau which has been designed in University Of Agder. Though, all the diminsions used in the simulation was tacken out from it. The Gantry-Tau consist of 3 parallel arms, 6 links, 12 joints, and 3 base linear actuators. Each arm contains one link that allows a translation movement. see Figure 1.1.

In this thesis, the 3-DOF Inverse kinematic model is applied to the Gantry-Tau. The links is considered to be the error parameters. 3 error parameters account for errors in the 3 arm lengths. Deeply, 6 error parameters account for error in 6 links. Functionalities from two different types of measurement equipment; laser interferometers and linear encoders have been utilised to create the error calibration systems. Simulation measurment is replaced with laser interferometeres.
The optimisation function for this implementation is built upon the basic geometric premises of the Gantry-Tau's structure. The Complex Research method was used to optimise the error calibration. This method will mirror the worst result arround the centroid and produce the best result for calibration. In other words, the method tries to make the RMSE value equal to zero. The result is given as a function of the generation against the RMSE value.

1.3 Scope of Study

This thesis looks at Error Kinematic Modeling and Calibration of the 3-DOF Gantry-Tau Parallel Kinematic Machine, which was first presented in Brogardh et al. (2005). Triangular mounted links give several advantages: they enable a reconfiguration of the robot and a larger reach is obtained in the extremes of the workspace. The researcher used computer based software like Matlab to provide a mathematical model as well as calibrations methods for the 3 DOF Gantry- Tau Parallel kinematic machines. This thesis report presents the development of tools for selection of measurements points according to the chosen measurement device.
The error on links length of the Gantry-Tau were the place of study. The links length were found by applying the IK modelling on the Gantry-Tau. The laser tracker experimentaly will read the position of the robot head. New measured link length will be calculated. The error between the link lengths and the measured link lengths will be calculated. A suitable optimization method will be applied to calibrate and minimize the error.

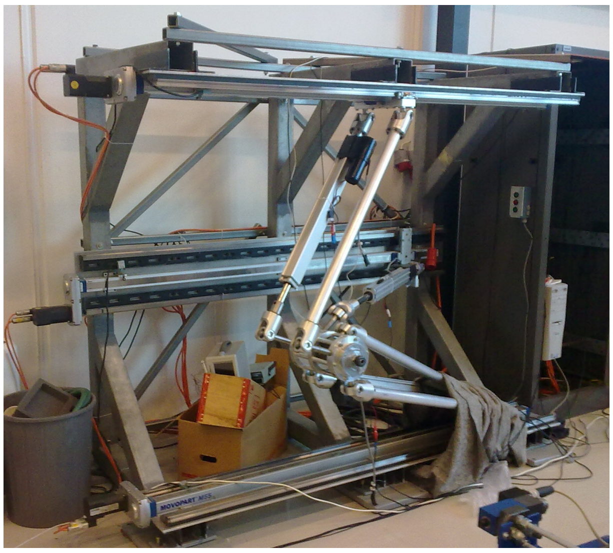


Figure 1.1: *Picture of the Gantry-Tau which used in this thesis.*

1.4 Motivation

The Error Kinematic Modeling and Calibration of The 3- DOF Gantry-Tau Parallel Kinematic Machine is a challenging area of study. Considering the area of optimization as well as calibration of the machine brings about the optimal performance. The modeling and calibration of the Gantry-Tau and other parallel manipulators is very important in application to its workspace.
The workspace of parallel manipulators with linear actuation beside the limits given by singularities and relative actuator placements is limited owing to three types of constraints: mechanical limit on the passive joints, interference between links and limited range in the length of linear actuators.

1.5 Organization of Thesis

The structure of this thesis is organized as follows:

- Chapter 1 contains aknowledgement, abstract, explain the scope of study, motivations and orginisation of the thesis.
- Chapter 2 includes a breif literature about Gantry-Tau and PKM.
- Chapter 3 provides a mathematical model of the 3- DOF Inverse Kinematics of the Gantry- Tau.
- Chapter 4 looks the identification methods, where computer aided software were used in calibrations methods in this case Matlab was extensively used. This chapter will produce mainly the Complex Search Method.
- In Chapter 5, the Cartesian paths -Linear and Circle path- for identification have been optimised.
- Chapters 6 and 7 look at the simulations, results, conclusion of the thesis, and suggested farther work.

Chapter 2

Gantry-Tau PKM

The Parallel kinematic manipulator (PKM) is a closed-loopkinematic chain mechanism whereby the end-effector is connected in parallel [8]. The PKM consists of a more number of actuators than the number of controlled degrees of freedom of the end-effector. Closed-kinematic chains.

Motion platforms with six degrees of freedom, also known as hexapods, are possibly the most popular robotic manipulators used in simulation technology. This parallel mechanism was first described by V. E. Gough [7],who constructed an octahedral hexapod to test the behavior of tyros subjected to forces created during airplane landings. The first document providing a detailed description of this structure used as an airplane cockpit simulator was published in 1965 by D. Stewart [4] (hence the name of the structure).The Stewart platform is a closed kinematic system with six degrees of freedom and six adjustable length links. Figure 2.2 and Figure 2.1 are two examples of the Gantry-Tau PKM. Compared to other similar structures, its main advantage is high rigidity and a high input-power-to-device-weight ratio.

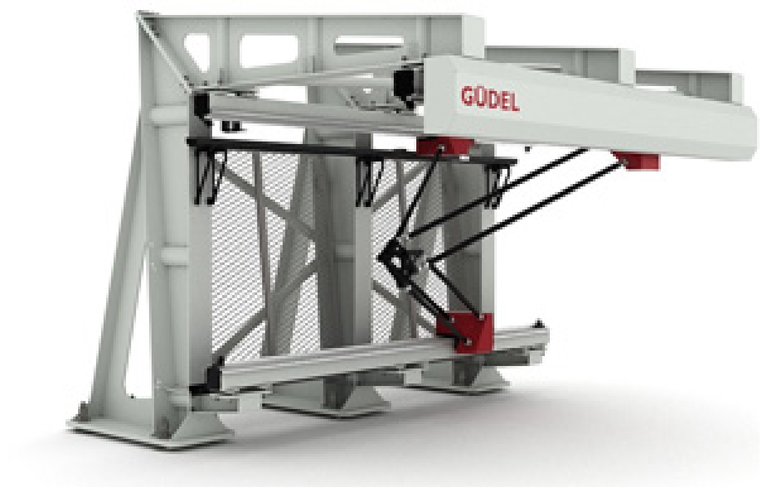


Figure 2.1: *Picture of the Gantry-Tau PKM.*

The traditional parallel kinematic manipulators (PKMs) have various applications in the industry due to its ability to occupy little space as compared to previous series kinematic manipulator.

The Tau family of PKMs was invented by ABB Robotics,[5] . The Gantry-Tau was designed to overcome the workspace limitations while retaining many advantages of PKMs such as low moving mass, high stiffness and no link twisting or bending moments. For a given Cartesian position of the robot each arm has two solutions for the inverse kinematics, referred to as the left- and right-handed configurations [3]. While operating the Gantry-Tau in both left- and right-handed configurations, the workspace will be significantly larger in comparison with both a serial Gantry-type robot and other PKMs with the same footprint and

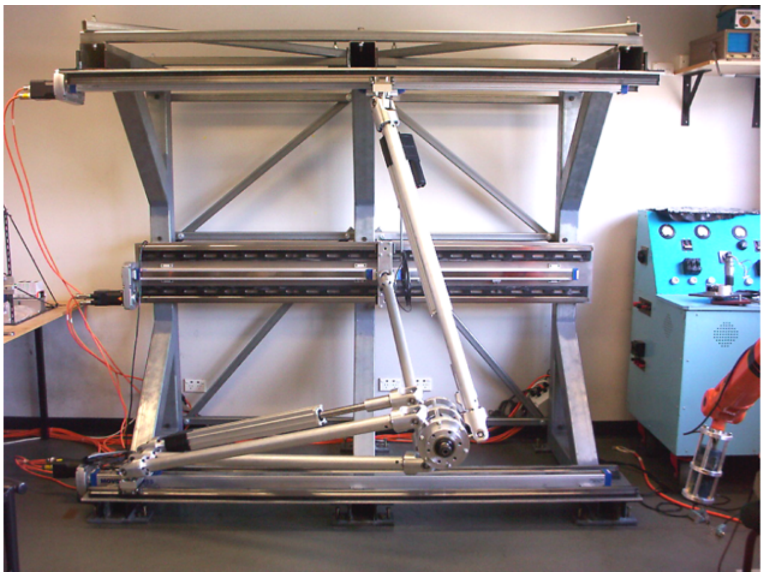


Figure 2.2: *An example of the Gantry-Tau PKM.*

with only axial forces in the links of the arms.

The intended application of the robot is for machining operations requiring a workspace equal to or larger than of a typical serial type robot, but with higher stiffness. However, the robot can also be designed for very fast material handling and assembly or for high precision processes such as laser cutting, water jet cutting and measurement. Currently, there exist large gaps between the stiffness and dynamic properties of serial-type robots and many of the developed PKM prototypes. An industrial serial-type robot typically has a Cartesian stiffness in the range 1-2 Newton's per micron ($N/\mu m$) and resonance frequencies below 10 Hz. A high-speed machining centre typically has stiffness larger than $50N/\mu m$ and resonance frequencies larger than 50 Hz. Between these specifications there exists a large range of untapped application areas for PKMs. With higher stiffness and resonance frequencies compared to serial robots, PKMs can provide benefits such as reduced oscillations and overshoots which in turn can lead to economic benefits gained from reduced cycle-times

The Gantry-Tau PKM has the potential to meet the needs the small and medium size enterprises. Robots are used in large industry today to optimized work tasks that could have taken many years, and money to complete. Some of these examples are welding, fabrication in the automobile industry, the process and chemical industry, and more and more in the oil and gas industries.

For the small and medium enterprises (SMEs) produces a lot, a robot with more flexibility to fit constantly changing tasks is needed. There are very few industries today that have adjustable geometry. It is for that matter that a new robotic concept should be developed to meet the demands of the SMEs.[11]. This industry needs robots that can reconfigure according to the various tasks. The Gantry-Tau PKM will be ideal for the SMEs, it could be designed in a modular way so that the framework can be adjusted to specific tasks. [6].

There are however some short falls for the SMEs, that is every reconfiguration robots requires a new kinematic calibration. Recalibration could prove difficult for non or un trained personnel at the SMEs. It will therefore be prudent to provide the adequate tools for the personnel to easily deal with the multiple kinematic calibrations. Robots are normally delivered as an optimized entity either for commercial or research purposes. The geometry is not changed in both cases, the kinematic calibration, which is needed is performed only once by the trained personnel for this purpose or the research personnel.[9].

The kinematic calibration of Gantry-Tau robot includes several manually performance steps which need a well trained personnel who can re do the calibrations of changed geometry , such as the choice of measurements poses or the analysis of measurements data. It cost SMEs some money to always bring in specialist to do these calibrations and recalibrations, so it will be prudent to give its own stuff a good understanding in dealing with these works.

Robot geometry: The Gantry-Tau PKM consists of three kinematic chains. Each chain includes a prismatic actuator, which is connected to the end-effector plate via a link cluster. The actuators are implemented as carts moving on tracks. The altogether six links are distributed to the link clusters in a 3-2-1 configuration and connected to carts and end-effector plate by spherical joints. The placement of the passive joints on plate and carts is such, that the links belonging to one cluster form parallelograms, which leads to a constant tool orientation. The robot has therefore three purely translational degrees of freedom (DOF).[2]

Chapter 3

3-DOF Inverse Kinematics (IK)

3.1 Introduction

Both forward and inverse kinematics can give a solution to understand the geometry of motion of the Gantry-Tau. Forward kinematics is a maping from joint space Q to Cartesian space W:

$$F(Q) \quad = \quad W \tag{3.1}$$

While inverse kinematics is a method to find the inverse maping from Cartesian space W to joint space Q:

$$Q \quad = \quad F^{-1}(W) \tag{3.2}$$

It is important to identify the mobility of the Gantry Tou before starting the kinematic formulation. The Gantry-Tau is considered to be 5-DOF [13]. By the same concept let us consider that the telescopic links $L2_2$ and $L3_3$ shown in Figure 3.1 have a fixed length, the configuration will be 3-DOF in this case. In this section a 3-DOF IK will be discussed as it has been publeshed in [10].

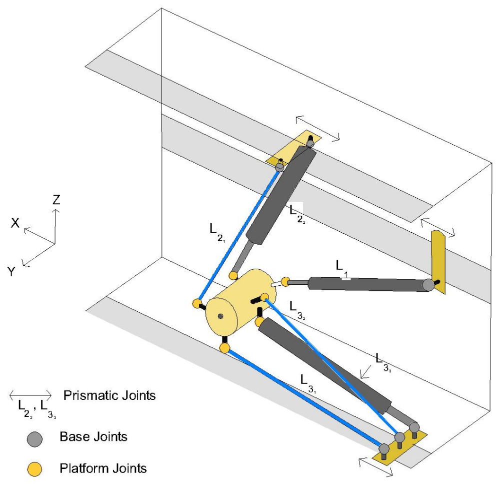


Figure 3.1: *The links and joints of the Gantry-Tau*

The 3-DOF Inverse Kinematics of the Gantry-Tau will be the best and simplest approach to use. It will not give any unknown variabels and will be relatively easy to programmed. 3-DOF IK is a simplefied model which reduces the geometric complexity of the structure. Assamption of fixed links $L2_2$ and $L3_3$ is taken in consideration, which can yeild to assamption of reduces the links from 6 links to 3 links sharing the same joint on the manipulated platform as shown in figure 3.2. The general case of 3-DOF configeration will be used on this work where 6 links is assumed which connect with speciefic offsets to the platform which shown in figure 3.3

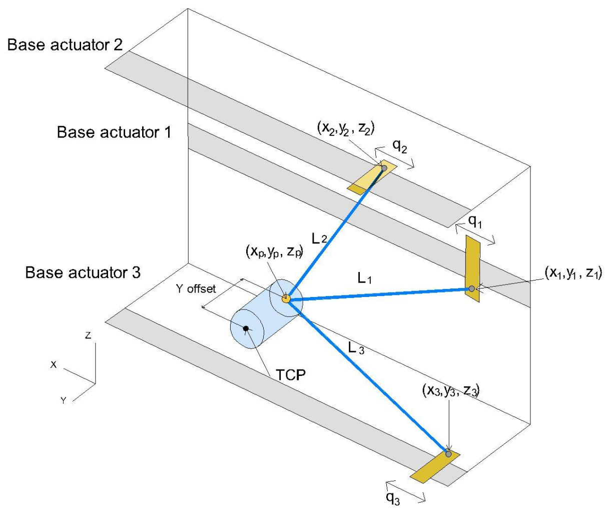


Figure 3.2: *Simplefied model of the Gantry-Tau*

3.2 Calculations of 3-DOF IK

To solve the IK for the Gantry-Tau, the calculations will be done from the Tool Centre Point (TCP) moving step by step toward the acuators.

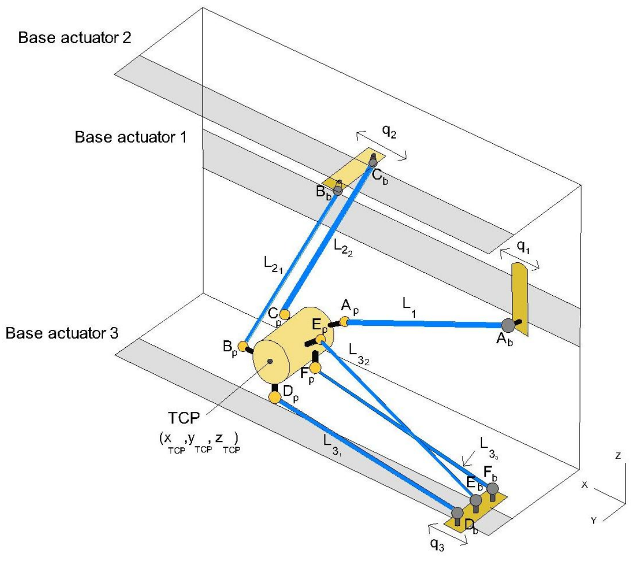


Figure 3.3: *The Gantry Tau for 3-DOF*

By looking to Figure 3.3, The first step is to calculate the link lengths which are:

$$(L_1)^2 = (X_{Ap} - X_{Ab})^2 + (Y_{Ab} - Y_{Ap})^2 + (Z_{Ab} - Z_{Ap})^2 \tag{3.3}$$
$$(L_{2_1})^2 = (X_{Bp} - X_{Bb})^2 + (Y_{Bb} - Y_{Bp})^2 + (Z_{Bb} - Z_{Bp})^2 \tag{3.4}$$
$$(L_{3_1})^2 = (X_{Dp} - X_{Db})^2 + (Y_{Db} - Y_{Dp})^2 + (Z_{Db} - Z_{Dp})^2 \tag{3.5}$$

X_{Ab}, X_{Bb}, X_{Db} are the acuaters positions which are equal to q_1, q_2, q_3 respectively. This leads to the following equation:

$$q_1 = \pm\sqrt{(L_1)^2 - (Y_{Ab} - Y_{Ap})^2 - (Z_{Ab} - Z_{Ap})^2} + X_{Ap} \tag{3.6}$$
$$q_2 = \pm\sqrt{(L_{2_1})^2 - (Y_{Bb} - Y_{Bp})^2 - (Z_{Bb} - Z_{Bp})^2} + X_{Bp} \tag{3.7}$$
$$q_3 = \pm\sqrt{(L_{3_1})^2 - (Y_{Db} - Y_{Dp})^2 - (Z_{Db} - Z_{Dp})^2} + X_{Dp} \tag{3.8}$$

Her it is noticed that for each actuater there are two possible solutions at a certine position of the TCP. In general we can say that every point can be reached by two positions of the actuator. To simplify this meaning it is called left-hand solution and right-hand solution. By practice this will give 8 possible solutions for every position of the TCP. This will be also a user input. The user should decide how can he reach a certine position. Figure 3.4 shows an example for the 8 possible solutions for a certine position.

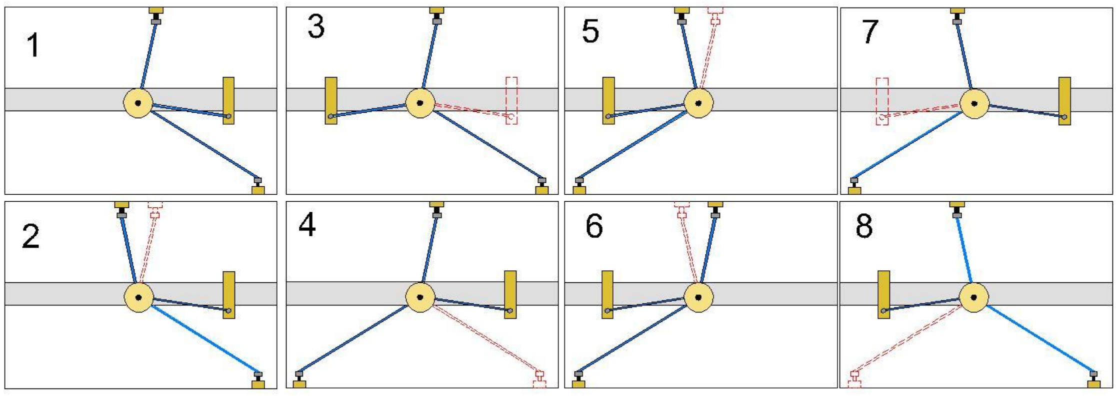


Figure 3.4: *The 8 possible configurations for the end-effector*

The links lengths L and the distances between the actuators and the joint of base connections of the links b are fixed and known. To determine the actuators positions, the joint connections between links and platforms p must be found depending on TCP position. In general it can be defined that:

$$[\ P\ ] = [\ TCP\ ] + [\ R_y\ ]\,[\ OFFSETT\ ] \tag{3.9}$$

This expression can be written in a clear way including the real variables:

$$\begin{bmatrix} X_{Ap} \\ Y_{Ap} \\ Z_{Ap} \end{bmatrix} = \begin{bmatrix} X_{TCP} \\ Y_{TCP} \\ Z_{TCP} \end{bmatrix} + [RotY] \begin{bmatrix} d_{AX} \\ d_{AY} \\ d_{AZ} \end{bmatrix} \tag{3.10}$$

$$\begin{bmatrix} X_{Bp} \\ Y_{Bp} \\ Z_{Bp} \end{bmatrix} = \begin{bmatrix} X_{TCP} \\ Y_{TCP} \\ Z_{TCP} \end{bmatrix} + [RotY] \begin{bmatrix} d_{BX} \\ d_{BY} \\ d_{BZ} \end{bmatrix} \tag{3.11}$$

$$\begin{bmatrix} X_{Dp} \\ Y_{Dp} \\ Z_{Dp} \end{bmatrix} = \begin{bmatrix} X_{TCP} \\ Y_{TCP} \\ Z_{TCP} \end{bmatrix} + [RotY] \begin{bmatrix} d_{DX} \\ d_{DY} \\ d_{DZ} \end{bmatrix} \tag{3.12}$$

Where TCP are the user input and $OFFSETT$ is a known distances. Rotation arround y-axis R_y must be found in order to know P. This guides us to the second step of IK which is foundation of the rotation arround y-axis. Let us immagine a rotation of β angle arround y-axis. which is shown in figure 3.5. An example is given to make it easier for understanding. The rotation is arround y-axis so y coloumb will be fixed and equal to:

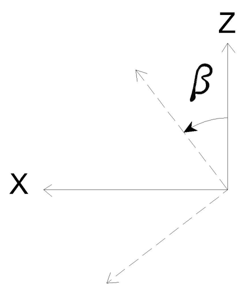


Figure 3.5: *Rotaion of β arround y-axis*

$$\begin{bmatrix} . & 0 & . \\ . & 1 & . \\ . & 0 & . \end{bmatrix} \tag{3.13}$$

Let us consider a rotation of β= 90 degree. Then X becomes -Z and Z becomes -X. This can easy replaces on the rotation array and it becomes:

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \tag{3.14}$$

It is know that $\sin 90 = 1$, $\cos 90 = 0$, and $-\sin 90 = -1$. The rotation array arround y-axis will be :

$$RotY = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \tag{3.15}$$

Equation 3.15 shows that we still need to know β. Here, thinking of the geometry of the robots will lead us to find a suitable expression which can be easily used for programming.

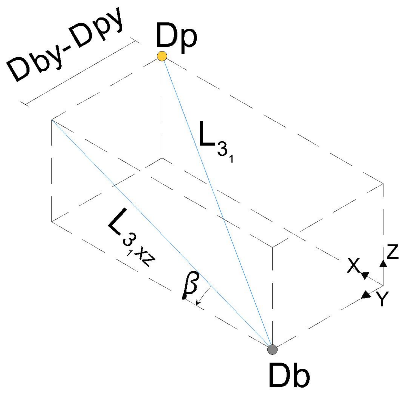


Figure 3.6: *Geometry of the third link in 3D.*

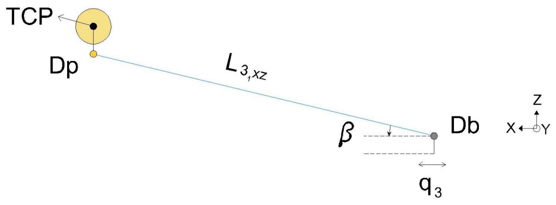


Figure 3.7: *Geometry of the third link in XZ-plane*

By looking to Figure 3.6 and Figure 3.7, L_{3_1} in xz-plane will be equal to:

$$L_{3_{1xz}} = \pm\sqrt{(L_{3_1})^2 - (Db_y - Dp_y)^2} \tag{3.16}$$

Then

$$\beta = \pm \text{acos} \frac{Dp_x - q_3}{L_{3_{1xz}}} \tag{3.17}$$

And finally

$$\beta = \pm \text{acos} \frac{Dp_x - q_3}{\sqrt{(L_{3_1})^2 - (Db_y - Dp_y)^2}} \tag{3.18}$$

It is noticed that β still dependent on q. This make the IK impossible to solve. β must be independent of q. A new concept for foundation of β which can be obtained from an auxiliary link L_{aux}. This auxiliary link is constructed from the triangular linkage formed between the platform and links of arm 3. The auxiliary link L_{aux} is the line starts from D_b and moves perbendicularly to the line connected between E_p and D_p.

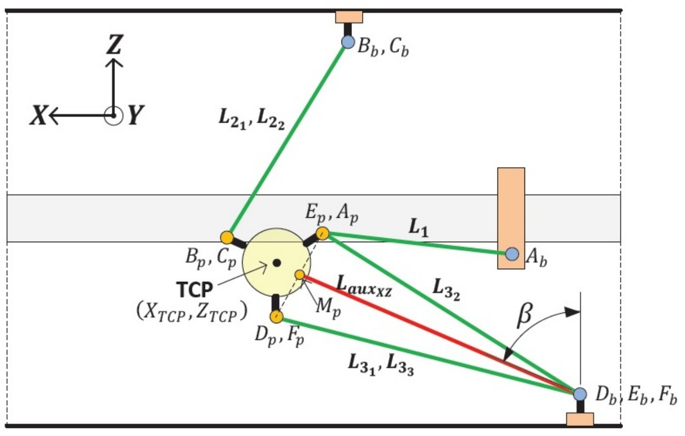


Figure 3.8: *Explaine of the auxiliary link in xz-plane L_{auxxz} [10]*

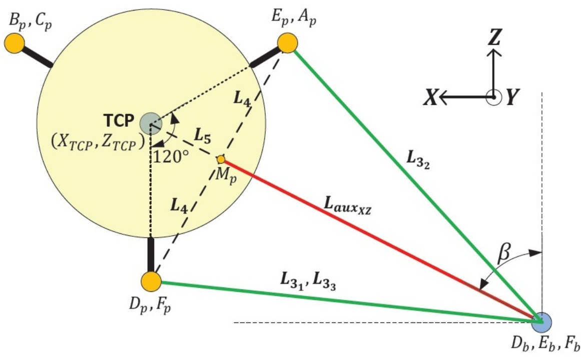


Figure 3.9: *The auxiliary link in xz-plane L_{auxxz} includs needed parameters [10]*

Figure 3.8 and 3.9 show the auxiliary link in xz-plane L_{auxxz}
By looking to figure 3.9:

$$\cos\beta = \frac{Z_{TCP} - Z_{Db}}{L_{auxXZ} + L_5} \tag{3.19}$$

And

$$L_{aux} = \sqrt{(L_{31})^2 + (L_4)^2} \tag{3.20}$$

L_4 and L_5 can be found by applying of the cosine theorem and fethaghors respectivly which equal to:

$$L_4 = \frac{1}{2}\sqrt{(Z_{Dp})^2 + (Z_{Ep})^2 - 2Z_{Dp}Z_{Ep}\cos(120)} \tag{3.21}$$

$$L_5 = \sqrt{(Z_{Dp})^2 - (L_4)^2} \tag{3.22}$$

(3.23)

And from figure 3.10, it can be found that:

$$L_{auxxz} = \sqrt{(L_{aux})^2 - (Y_{TCP} - Y_{Db})^2} \tag{3.24}$$

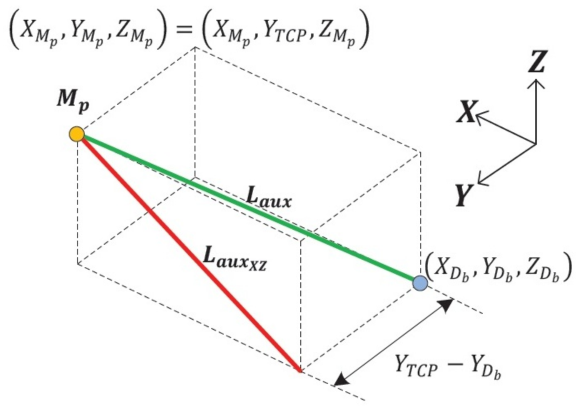

$(X_{M_p}, Y_{M_p}, Z_{M_p}) = (X_{M_p}, Y_{TCP}, Z_{M_p})$
M_p
Z
X
Y
L_{aux}
$L_{aux_{XZ}}$
$(X_{D_b}, Y_{D_b}, Z_{D_b})$
$Y_{TCP} - Y_{D_b}$

Figure 3.10: *The auxiliary link in xz-plane $L_{aux_{xz}}$ [10]*

Applying of equation 3.20 leads to:

$$L_{aux_{xz}} = \sqrt{(L_{31})^2 + (L_4)^2 - (Y_{TCP} - Y_{Db})^2} \tag{3.25}$$

Finally β can found by:

$$\beta = \pm \text{acos} \frac{Z - Z_{Db}}{L_{auxXZ} + L_5} \tag{3.26}$$

Where all variables are known.

Chapter 4

Identification Method

Calibration:
Identification method is the main part of this project. A deep study has been done to choose a good method which can be suitable for calibration error. A MatLap program should be established to achieve this goal. Calibration error can be found depending on the links length L_1-L_6 in the 5-DOF case and L_1 - L_3 in the 3-DOF case. The link lengths are known. By using the laser tracker in experiments, the TCP position can be found. The link lengths also can be found analytically by IK. Comparing of the theoretical link lengths to the real link length will lead to find the lengths error. Before going to laboratory and experiments, it will not be difficult to find the TCP position.

In the beginning it can be established a program to return back the theoretical link lengths L_1 - L_3 where X, Y, Z and C_1, C_2, C_3 are the user inputs and the actuators position q_1, q_2, q_3 have been found. The same program can be used when we use the laser tracker and experiments. After foundation of the error, a suitable method should be applied to achieve the calibration error. The IK for parallel robots leads to nonlinear equations. An optimization function such as FMINCON and LSQNONLIN is needed to achieve the calibration.

4.1 LSQNONLIN

LSQNONLIN is a good method to solve non-linear least squares problems. LSQNONLIN attempts to solve problems of the form:

$$X = \min \sum (FUN(X)^2) \tag{4.1}$$

where X and the values returned by FUN can be X vectors or matrices.

The IK function can be written in the form

$$\begin{bmatrix} Q_1 & Q_2 & Q_3 \end{bmatrix} = IK(X, Y, Z, C_1 - C_3, L_1 - L_6) \tag{4.2}$$

It should be noticed that the optimization will be done during a path of the robots. The path means that the robot will be moved depending on selected of different points. This will be explained on Chapter 5. Let us choose a path begining (X, Y, Z) and ending with (X^m, Y^m, Z^m) for $(C_1 - C_3)$ to $(C_1^m - C_3^m)$. The IK for the path will be:

$$\begin{bmatrix} Q_1 & Q_2 & Q_3 \\ \vdots & \vdots & \vdots \\ Q_1^n & Q_2^n & Q_3^n \end{bmatrix} = IK \begin{bmatrix} X & Y & Z & C_1 - C_3 & L_1 - L_6 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ X^n & Y^n & Z^n & C_1^n - C_3^n & L_1 - L_6 \end{bmatrix} \tag{4.3}$$

The LSQNONLIN optimisation function will be in the form:

$$[L_1 - L_6] \quad = \quad \min_{L_1 - L_6} \sum_{i=1}^{3} (Q_i - Q_i^m) \tag{4.4}$$

The optimization will start with some lenk lengths. The square diffrence between the theoritical Q_i and the measured Q_i^m will be counted. Minimizing the error is the aim of the optimization. The link lengths will be the factor for refering to the error in the system.

LSQNONLIN implements two different algorithms: trust region reflective and Levenberg-Marquardt. Choose one via the option Algorithm: for instance, to choose Levenberg-Marquardt, set

$$OPTIONS \quad = \quad optimset('Algorithm','levenberg - marquardt') \tag{4.5}$$

and then pass OPTIONS to LSQNONLIN as it explains in [1].

$$X \quad = \quad LSQNONLIN(FUN, X_0, LB, UB, OPTIONS) \tag{4.6}$$

Equation 4.6 is the generall form of LSQNONLIN in MatLab. The optimization starts at the matrix X_0 and finds a minimum X to the sum of squares of the functions in FUN. FUN accepts input X and returns a vector (or matrix) of function values F evaluated at X. It should be noted that FUN should return $FUN(X)$ and not the sum-of-squares $\sum(FUN(X)^2)$, where $(FUN(X)$ is summed and squared implicitly in the algorithm.

The lower bounds LB and upper bounds UB define a set of lower and upper bounds on the design variables X, so that the solution is in the range $LB \leq X \leq UB$. The options $OPTIONS$ minimizes with the default optimization parameters replaced by values in the structure OPTIONS, an argument created with the OPTIMSET function.

In our application, the results of LSQNONLIN are not acceptable. This is due to the LSQNONLIN find the first local minimum value of the function and return it as a minimum value. If LB and UB choosed to be so closed to the minimum value then the minimum value will be the same of the local minimum value and it will return a correct optimization value. In this case the result will not be enough for a good optimization. Increasing of the LB and UB will results many local minimum value and the optimization will return the first local minimum value.

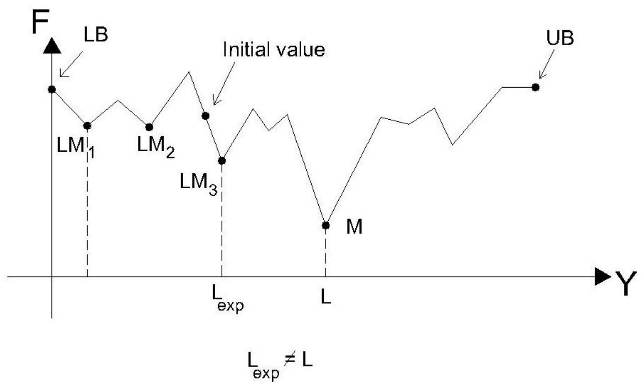


Figure 4.1: *Explaine of the princible work of LSQNONLIN - high range boundaries*

Changing of the OPTION parameter will help to gett a better optimization but not to solve the problem. For mor explain, let us see figure 4.1 and figure 4.2 which show the function by changing of Y while X is fixed. In figure 4.1 there is a high range between LB and UB which give a wrong optimization result. In figure 4.2 the LB and UB are too closed to the minimum value, This means that there is just one minimum value which will be the local and actually minimum value. At this case the result will be right.

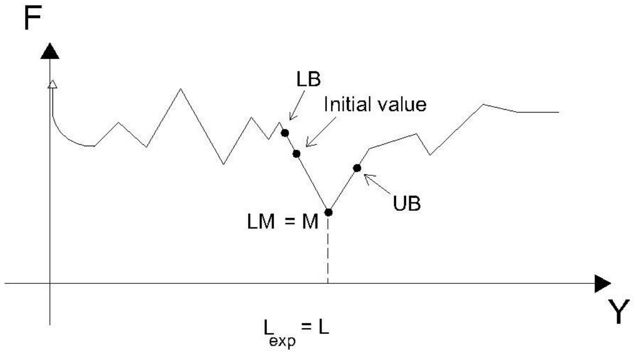


Figure 4.2: *Explaine of the princible work of LSQNONLIN - closed boundaries*

4.2 Complex Search Method

As it described before, the Gantary-Tau and other PKMs is difficult to calibrate the error manually. The Complex Search Algorithm are presented in this section. Refrence [12] explained the Complex Search Method in an easy way and will be mainlly used in this section.

4.2.1 Evolutionary Multi-Objective Optimisation

The method used in this thesis is based on evolutionary algorithm. Evolutionary algorithms are suitable for the Error Kinematic Modeling and Calibration because they deal simultaneously with a set of possible solutions that allow to find a set of candidates for the optimal result in a single run of the algorithm instead of having to perform a series of separate runs as it deals with the traditional mechanical programming method.

The complex search method is based on the evolutionary multi-objective optimisation concept. An example of multi-objective optimisation problem is given below:

$$\min \left[\; f_1(par), ..., f_k(par), f_{k+1}(par) \; \right] \tag{4.7}$$

subject to

$$g_i(par) \leq 0 \qquad (1 \leq i \leq r) \tag{4.8}$$
$$h_j(par) = 0 \qquad (1 \leq j \leq s) \tag{4.9}$$
$$par^L \leq par_p \leq par^U \quad (1 \leq N_{params} \leq N_{pop}) \tag{4.10}$$

Where k is the number of objective functions $f_k(par)$, par is a vector of N_{params} optimisation parameters (solutions), $g_i(par)$ and $h_j(par)$ are each of the r inequality and s equality problem constraints, par^L and par^U are lower and upper limits of the optimisation parameters. The constraints are considered as a new objective constraints handling function $f_{k+1}(par)$.
In multi-objective optimisation, objectives are not comparable with respect to their magnitude and value and may conflict, where some objectives can not be increased without a decreasing of others. The result of a multi-objective optimisation is a set of trade-off solutions which are considered to be suitable for all objectives. Due to the impossibility of achieving optimal values of all objectives simultaneously, multiple criteria decision making always involves a choice problem, for example the weight functions are used to define a level of importance of the objectives. The final solution represents a compromise between the different objectives and depends on the preferences of the decision maker.[12].

4.2.2 Concept and Stages

In this section, the concept of work of the Complex Search Method will be explained step by step. MatLap files for Complex Method and Fittness Function will be encluded in the appendix. These files show an explain for the mechanism of work of each step.

The Complex Search Method consists of several stages:

- Generete the the initial population N_{pop} from the number of the optimisation parameters N_{params}. The population N_{pop} is equal to $2^{N_{params}} + 1$.
- Found of root mean squared error. Then minimize the error by maximizing the function.
- Choose the boundaries upper and lower limits par^U and par^L and the number of times to evaluate the maximization of the error gen_{max}.
- Evaluate the objectives. The worst and the best results of the error are identified as x_{worst} and x_{best} respectively in each iteration.
- Calculate the centroid of the remaining error $x_{centroid}$ in each iteration.
- Mirror worst result of error x_{worst} through the centroid $x_{centroid}$.
- The new mirror result x_{new} will be checked to be not out of boundaries. Apply the maximization function and check the new mirror result. If it still worst, move it toward the best result by reset number of iterations k_r and increment it.
- Increment the generation gen and repeat the previos steps.
- Let the complex function initialize the maximization function and the population randomly.

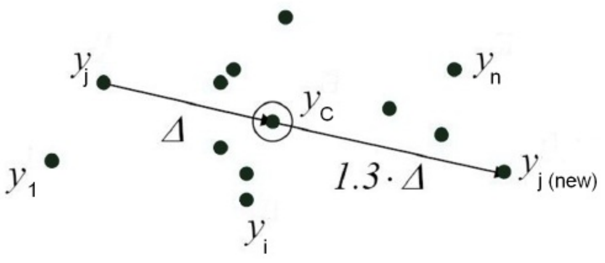


Figure 4.3: *Mirror of the worst result through the centroid.[12]*

To explain the implementation concept of the complex search method, consider Figure 4.3 where y_c is the centroid, y_j is the worst result, $y_{j(new)}$ is the mirror of the worst result, and n is the number of the results y. As it explained on the stages above, the centroid y_c will calculated and it is defined to:

$$y_c = \frac{\sum_{i\neq j}^{n} y_i}{n-1} \tag{4.11}$$

When the worst result y_j found, it will be mirrored through the centroid y_c. The new mirrored result $y_{j(new)}$ is equal to:

$$y_j(new) = 1.3 * (y_c - y_j) + y_c \tag{4.12}$$

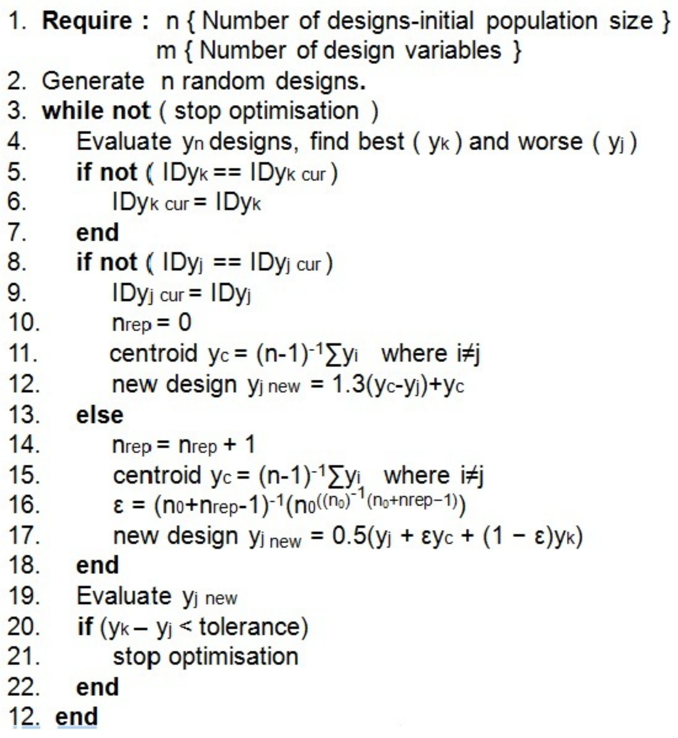
The new mirrord result $y_{j(new)}$ will move toward the carrent best result more or less strongly depending on following defintions:

$$y_{j(new)} = 0.5 * (y_j + \epsilon * y_c + (1-\epsilon) * y_k) \tag{4.13}$$

$$\tag{4.14}$$

$$\epsilon = \frac{n_0^{\frac{n_0+n_rep-1}{n_0}}}{n_0 + n_rep - 1} \tag{4.15}$$

Where the y_k is the best result, n_0 is a tuning parameter which is normally equal to 4 to 5, n_{rep} is the number of iterations. In the complex search algorithm the parameters n_0 and n_{rep} are used to switch between the worst and the best results. Figure 4.4 shows the algorithm of the complex method.



```
1.  **Require :** n { Number of designs-initial population size }
                  m { Number of design variables }
2.  Generate n random designs.
3.  **while not** ( stop optimisation )
4.      Evaluate $y_n$ designs, find best ( $y_K$ ) and worse ( $y_j$ )
5.      **if not** ( $IDy_K == IDy_{K\ cur}$ )
6.          $IDy_{K\ cur} = IDy_K$
7.      **end**
8.      **if not** ( $IDy_j == IDy_{j\ cur}$ )
9.          $IDy_{j\ cur} = IDy_j$
10.         $n_{rep} = 0$
11.         centroid $y_c = (n-1)^{-1}\sum y_i$   where $i \neq j$
12.         new design $y_{j\ new} = 1.3(y_c - y_j) + y_c$
13.     **else**
14.         $n_{rep} = n_{rep} + 1$
15.         centroid $y_c = (n-1)^{-1}\sum y_i$   where $i \neq j$
16.         $\varepsilon = (n_0 + n_{rep} - 1)^{-1}(n_0^{((n_0)^{-1}(n_0 + nrep - 1))})$
17.         new design $y_{j\ new} = 0.5(y_j + \varepsilon y_c + (1 - \varepsilon)y_K)$
18.     **end**
19.     Evaluate $y_{j\ new}$
20.     **if** ($y_K - y_j <$ tolerance)
21.         stop optimisation
22.     **end**
12. **end**
```

Figure 4.4: *The complex Search Algorithms.*[12]

Chapter 5

Cartesian Paths for Identification

The TCP of the Gantry-Tau will be moved from place to another dependening on the new user input cartesian point. A sett of points is called the path which will controll the movement of TCP from point to point by entering different a mount of points between them. The path should be optimised depending on the user requirement. There is many requiremets for creating a sufficient path. It should be considered the work area of the Gantry-Tau. In Chapter 3, 8 possible solution for each point has been explained. The optimal path should keep the actuaters in the same side-hand solution. This will make the motion of TCP during the path smooth and will avoid the rigidity when an actauter has to move long way to oblige the arm changing direction between right-hsnd solution to the left-hand solution or the converse. This unrequired distuirb can happen when the TCP want to reach a critical points. A critical points are those points that the Gantry-Tau can reach them with less than 8 possible solution. Some of them can be just one solution to reach them. It means that, if it is obligatory for the designer to reach those points, he should take in account to choose the wright solution for the beginning point. The optimisation will make it easy enough to avoid rigidity and will make it easy to choose the shape of movement between sending and reaching points.

5.1 Straight Line

A MatLab program has been established to creat a straight line efficient path. The program chooses a set of points with equal distances between a point and the next one. Applying the Inverse Kinematic Model will calculate the position of each base actuator. It will also check for the right-hand and left-hand solution for each arm. If there is a change on the direction, the point will be avoited.

The straight line path *op.tracker* is presented by TCP XYZ coordinates, actuator positions are presented by matrix *op.enc*, and the configurations given by matrix $op.config$ as shown below:

$$op.tracker = \begin{bmatrix} X & Y & Z \\ 2.1000 & 0 & 1.4500 \\ 1.9600 & 0.0600 & 1.3850 \\ 1.8200 & 0.1200 & 1.3200 \\ 1.6800 & 0.1800 & 1.2550 \\ 1.5400 & 0.2400 & 1.1900 \\ 1.4000 & 0.3000 & 1.1250 \\ 1.2600 & 0.3600 & 1.0600 \\ 1.1200 & 0.4200 & 0.9950 \end{bmatrix} \tag{5.1}$$

$$op.enc = \begin{bmatrix} Q_1 & Q_2 & Q_3 \\ 1.2316 & 1.2611 & 1.5972 \\ 1.0388 & 1.1325 & 1.2883 \\ 1.0231 & 0.9338 & 1.2141 \\ 0.7710 & 0.8978 & 0.8240 \\ 0.2959 & 0.6004 & 0.7293 \\ 0.4357 & 0.2118 & 0.8090 \\ 0.3537 & 0.6327 & 0.2256 \\ 0.0305 & 0.0271 & 0.3545 \end{bmatrix} \tag{5.2}$$

$$op.config = \begin{bmatrix} C_1 & C_2 & C_3 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{5.3}$$

5.2 Circle

By the same way used to find the straight line path on 5.1, A MatLab program has been established to creat a circle line efficient path. This path created to move on $XZ - plane$, Y is kept equal to zero. The radius of the circle choosed to be 0.5 m.
The circle path given from the result of the program are:

$$op.tracker = \begin{bmatrix} X & Y & Z \\ 0.7 & 0 & 0.8 \\ 1.2 & 0 & 1.3 \\ 1.7 & 0 & 0.8 \\ 1.2 & 0 & 0.3 \end{bmatrix} \tag{5.4}$$

$$op.config = \begin{bmatrix} C_1 & C_2 & C_3 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{5.5}$$

Chapter 6

Simulations and Results

In this chapter the simulation and the result of the error calibration of the Gantry-Tau PKM will be inserted. As it explained on the previous chapters, the 3-DOF inverse kinematic modell has been established. The error will be equaled to the root of squared mean of the diffrence between the original link length and the calculated from the laser tracker collected data. The error model will be in this form:

$$E = \sqrt{(\frac{Q - \grave{Q}}{2})^2} \tag{6.1}$$

Where E is the error, Q is the 3-DOF inverse kinematic model using the original link lengths, and $\grave{Q}$ is the 3-DOF inverse kinematic model using the original calculated link lengths from the tracker.

The error modell function EMF will minimize the error by maximizing the error E.

$$EMF = \sum_{1}^{n} E \tag{6.2}$$

Where n is the number of the points in the generation path which applied.
This error modell function will be used by the complex search algorithms and return the results.
The number of error parameters N_{params} is equal to 3. These error parameters are $L1$, $L2_1$, and $L3_1$. The number of population N_{pop} is defined as:

$$N_{pop} = 2^{N_{params}} + 1 \tag{6.3}$$
$$N_{pop} = 9 \tag{6.4}$$

Table 6.1: Simulation Results for straight line path by changing number of generation.

Results	Simulation 1	Simulation 2	Simulation 3
Lower bounds	[1.2200 1.0600 1.2200]	[1.2200 1.0600 1.2200]	[1.2200 1.0600 1.2200]
Higher bounds	[1.2800 1.1200 1.2800]	[1.2800 1.1200 1.2800]	[1.2800 1.1200 1.2800]
number of generation	200	2000	10000
Best result	[1.2503 1.0898 1.2500]	[1.2500 1.0900 1.2500]	[1.2500 1.0900 1.2500]
Final RMSE	0.000676	0.000000	0.000000
Total time (sec)	1.4677	13.9037	70.0079

The generated straight line path which has been discussed in 5.1 has been used to calibrate the error. By changing the boundaries and the generation times, the complex still show a good results. Table 6.1 shows that the complex search method can minimize the error to zero by changing the number of generation.

The second coloum in Table 6.1 shows that the number of generation is not enough to minimize the error to zero and get the same original link lengths, while increasing the number of generation make the result better.

Table 6.2 shows that the complex search method can minimize the error to zero by applying different boundary limits. A high range limits have been applied where all gives a good results and minimize the result to zero.

Table 6.2: Simulation Results for straight line path by differant boundary limits.

Innputs and Results	Simulation 4	Simulation 5	Simulation 6
Lower bounds	[1.2000 1.0400 1.2000]	[1.1800 1.0200 1.1800]	[1.1800 1.0200 1.1800]
Higher bounds	[1.3000 1.1400 1.3000]	[1.3200 1.1600 1.3200]	[1.3200 1.1600 1.3200]
number of generation	10000	10000	50000
Best result	[1.2500 1.0900 1.2500]	[1.2500 1.0900 1.2500]	[1.2500 1.0900 1.2500]
Final RMSE	0.000000	0.000000	0.000000
Total time (sec)	70.2287	72.1280	339.5003

The generated circle path which has been discussed in 5.2 has been used to calibrate the error. Table 6.3 shows that the complex search method can minimize the error to zero by changing the number of generation. The boundaries have been tested before and it found the the methods can work on a high range boundaries

Table 6.3: Simulation Results for circle path by differant number of generation.

Innputs and Results	Simulation 7	Simulation 8	Simulation 9
Lower bounds	[1.1600 1.0000 1.1600]	[1.1600 1.0000 1.1600]	[1.1600 1.0000 1.1600]
Higher bounds	[1.3400 1.1800 1.3400]	[1.3400 1.1800 1.3400]	[1.3400 1.1800 1.3400]
number of generation	1000	10000	50000
Best result	[1.2500 1.0900 1.2500]	[1.2500 1.0900 1.2500]	[1.2500 1.0900 1.2500]
Final RMSE	0.000000	0.000000	0.000000
Total time (sec)	7.2360	74.6405	348.3416

The results of simulions will be shown in the following graphs respectevly with their positions on Table 6.1, Table 6.1 and Table 6.3.

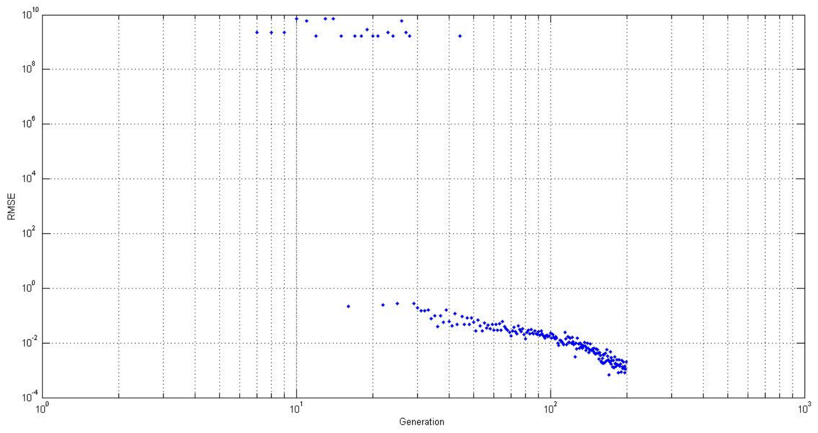


Figure 6.1: *Simulation 1*

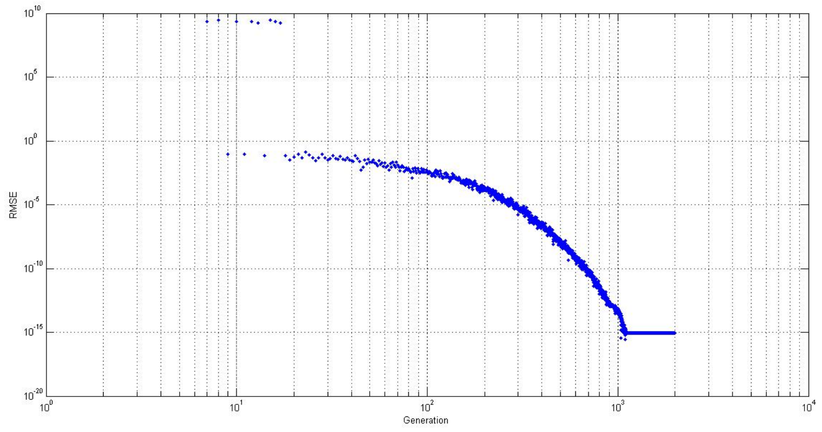


Figure 6.2: *Simulation 2*

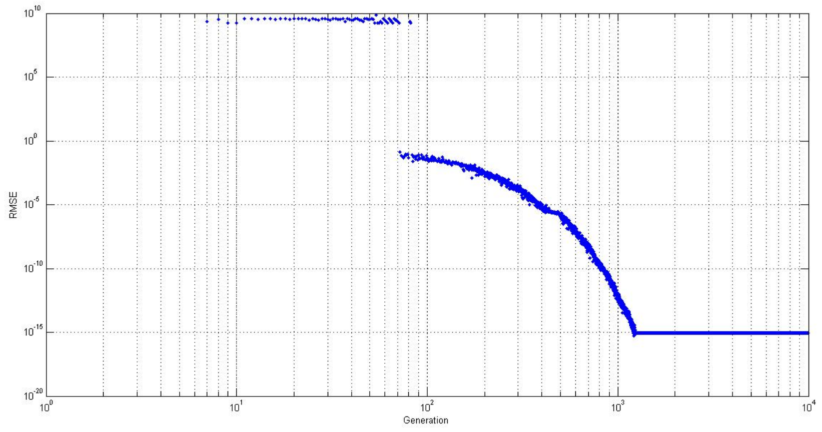


Figure 6.3: *Simulation 3*

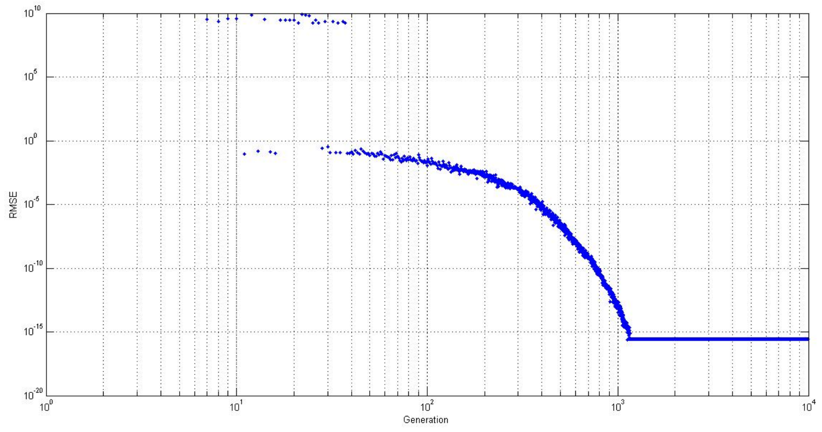


Figure 6.4: *Simulation 4*

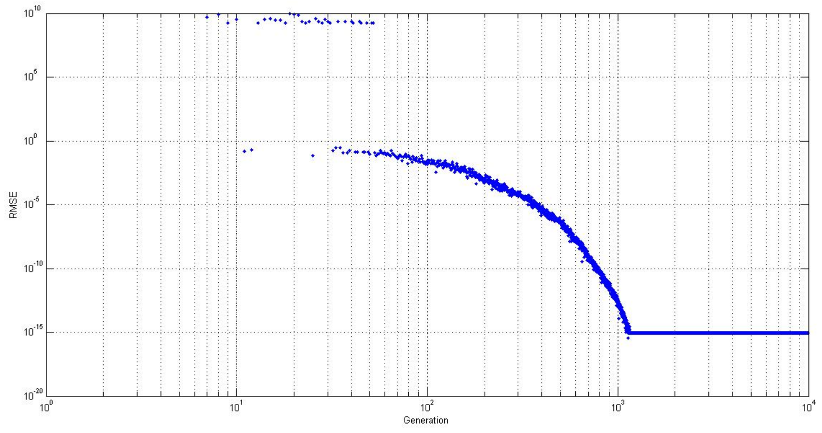


Figure 6.5: *Simulation 5*

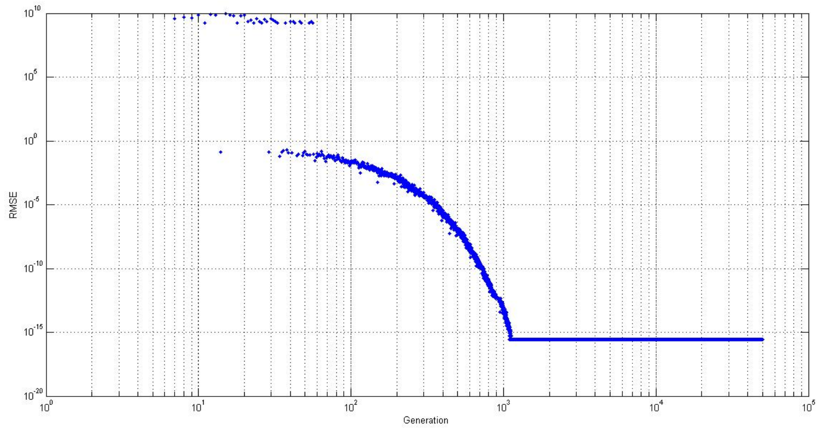


Figure 6.6: *Simulation 6*

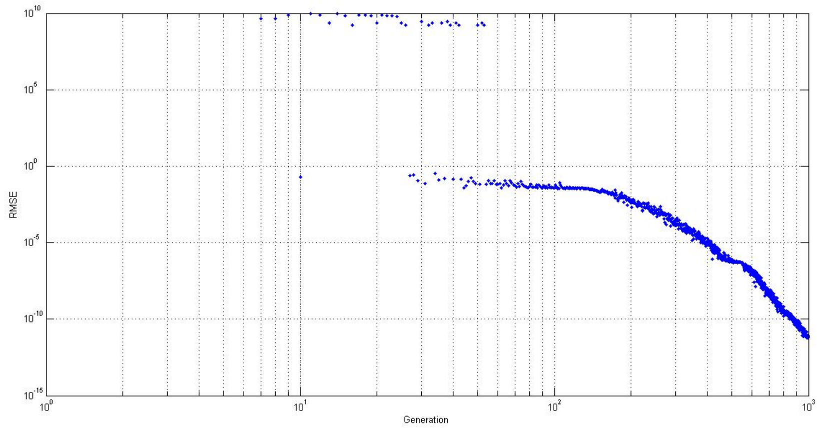


Figure 6.7: *Simulation 7*

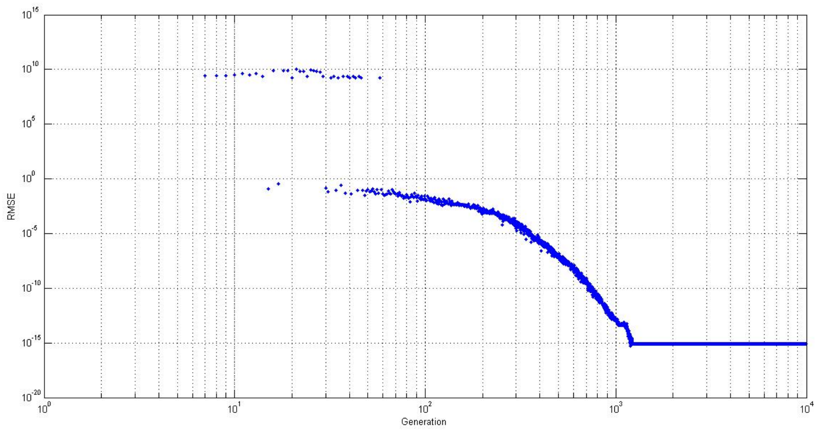


Figure 6.8: *Simulation 8*

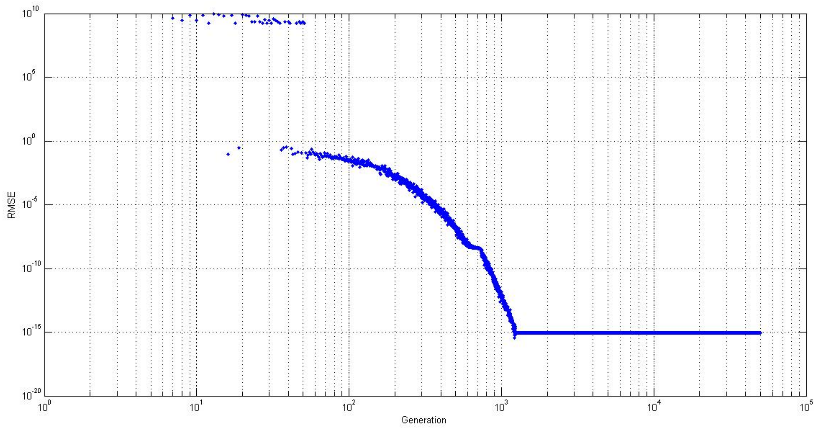


Figure 6.9: *Simulation 9*

Chapter 7

Conclusions and Suggested Further Work

7.1 Conclusions

The goal of this thesis was to apply asuitable technique for calibration error. This goal requires initially to model the kinematic errors of the Gantry-Tau PKM. The Gantry-Tau PKM consist of 6 links with 5-DOF. From the early stages of the project, it became clear that building a full kinematic model in both Forward and Inverse Kinematics with all error parameters is taken much time. Because of that, The Gantry-Tau considered to be 3-DOF and contains 3 arms. A 3-DOF inverse kinematic model have been established and the project became focussed on the Calibration error of three paremeters. These 3 parameters choosen to be the error on the 3 arms. Choosen of an efficient optimisation method was of importance.

LSQNONLIN gave unacceptable results and worked just for a very closed boundaries. The optimisation should be worked in a high range between the upper and lower limits and the method algorithm must be able to minimize the error in this high range.

The Complex Search Method was an efficient method. It has been applied for one point and gave a positive results. Then, cartisian paths are established using the inverse kinematic model. These paths were not highly accurate optimisation paths, but the complex method applied on these paths. Due to external constraints, there was no opportunity to collect experimental data. Experimentation would strengthen the merits of the project, but there is no enough time for generation of a highly accurate path and experimentation. Simulation by MatLab program was used instead of experiments. The results that have been achieved are highly relevant and enable the advancement of further works in the area of calibration error of Gantry-Tau PKM.

This thesis has been subjected to the complex search method to effectively calibrate the Gantry-Tau error, but the clearest conclusion to be found from this project is that there is an abundance of work to be done in order to achieve high strongly error calibration of the Gantry-Tau PKM. The Complex Search Method is an effiecient optimisation method which can apply perfectly on the calibration error of the PKM. 3-DOF Inverse Kinematic model is a consideration to make the work easy. The concentration were on the optimisation and identification method. Since the complex search method works well on the 3-DOF, it will be easy to apply for advanced 5-DOF inverse kinematic modell with more error parameters.

An advanced modelling with a higher number of error parameters can be established. These error parameters can be joint errors, platform errors and potentially opening arm lengths error parameters to form a full Kinematic Error Model for the Gantry-Tau. In the same time, collected data from experimental and laser tracker can be used instead of simulation. These two points can make the results strongly acceptable and more effective.

7.2 Suggested Further Work

In this section, it will be menstioned some points that can be suggested for those who are interested to work on the calibration error for Gantry-Tau.

- This thesis and the Matlab programing codes have been built in an easy way to replace the simulation with the experiment resuilts. The experiments is suggested to be done.
- Optemisation of an effective path is an important part to get the experiments work correctly. Choose an optimisation techniqe is suggested for different paths.
- Gantry-Tau calibration error can be extended and applies to a high number of error paremeters when the Gantry-Tau considered to be 5-DOF. Using of complex method technique for PKM kinematic error modell calibration in PKM is suggested.
- Using of LSGNONLIN technique for calibration error is not an immpossible idea. Changing of the options for this algorithm will not be enough to make it work correctly in PKM applications. The searcher should add some codes to this algorithm. These codes make the algorithm read the real minumum values and not the local minumum value. The Upper and Lower limits (boundaries) must not be closed to each other.

Bibliography

[1] MatLab 7.12.0 (2011a). *help LSQNONLIN.* MatLab-help, 2011.

[2] N. Andreff and I. Dressler. Closed-form calibration of the gantry-tau parallel robot. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, September.

[3] T. Brogardh and C. Gu. Parallel robot development at abb. In *Proc. 1st Intl. Coll. of the Collaborative Research Centre 562*, University of Braunschweig, 2002.

[4] T. Brogårdh. A device for relative movement of two elements. pages 1–15, 1996.

[5] T. Brogårdh. Design of high performance parallel arm robots for industrial applications. In *Proc. of the Symp. Commemorating the Legacy, Works, and Life of Sir Robert Stawell Ball Upon the 100th Anniversary of A Treatise on the Theory on The Screws.*, University of Cambridge, Trinity College, 2000.

[6] K. Nilsson R. Johansson A. Robertsson I. Dressler, M. Haage and T. Brogårdh. " configuration support and kinematics for reconfigurable gantry- tau manipulator". In *Proc. International Conference on Robotics and Automation (ICRA'07)*, Rome, 2007.

[7] V. Berbyuk L. Johannesson and T. Brogårdh. Gantry-tau - a new three degrees of freedom parallel kinematic robot. In *Proc. 4th Chemnitz Parallel Seminar*, pages 731–734, 2004.

[8] J.P Merlet. Parallel robots. Kluwer Academic Publishers, 2000.

[9] J. M Lavest P. Renaud, N. Andreff and M. Dhone. " simplifying the kinematic calibration of parallel mechanisms using vision-based metrology". *IEEE Transactions on Robotics*, 22(1):12–22.

[10] Victor F. Hernández R. *Development of a Model-Based Control System for the Gantry-Tau Parallel Kinematic Machine.* University of Agder, 2011.

[11] SME robot. *SME robot.* homepage, http://www.smerobot.org.

[12] Ilya Tyapin. *Multi-Objective Design Optimisation Of A Class Of Paralell Kinematic Machines.* University of Queensland, 2009.

[13] D. Zhang. Parallel robitic machine tools. *Modeling, Identification and Control*, pages 38–39, 2010.

List of Figures

List of Tables

Appendix

....
....
Gantry-Tau Parameters

function S=setup;
$S.L1$=1.25;   Length of link 1
$S.L2_1$=1.09;   Length of link 2
$S.L2_2$=1.09;   Length of link 3
$S.L3_1$=1.25;   Length of link 4...... real.
$S.L3_2$=1.25;   Length of link 5
$S.L3_3$=1.25;   Length of link 6

S.Ab.x = 0;   Pin A location on drive side
S.Ab.y = -0.572;
S.Ab.z = 0.775;

S.Ap.x = -0.1819;   Pin A location on platform side
S.Ap.y = -0.25;
S.Ap.z = -0.105;

S.Bb.x = 0;   Pin B location on drive side
S.Bb.y = 0.125;
S.Bb.z = 1.612;

S.Bp.x = 0;   Pin B location on platform side

S.Bp.y = 0;
S.Bp.z = 0.23;

S.Db.x = 0;   Pin D location on drive side
S.Db.y = 0.125;
S.Db.z = 0.188;

S.Dp.x = 0.1126;   Pin D location on platform side
S.Dp.y = 0;
S.Dp.z = -0.065;

S.Ep.z = -0.065;

....
....
Main Program (Calling the IK)

```
S=setup;
SS=setupp;       Tracker and encoder data
Q = invkin(S,X,Y,Z,C1,C2,C3)
```

....
....
Foundation of β Function

```
function beta=betafunction(S,Y,Z)
```
$L4 = 0.5 * (sqrt(S.Dp.z^2 + S.Ep.z^2 - 2 * S.Dp.z * S.Ep.z * cosd(120)));$
$L5 = sqrt(S.Dp.z^2 - L4^2);$
$LauxXZ = sqrt(S.L3_1^2 - L4^2 - (Y - S.Db.y)^2);$
```
if( isreal(LauxXZ))        real result of the length
LauxXZ=real(LauxXZ);
end;
beta = acosd((Z-S.Db.z)/(LauxXZ+L5));
```

....
....
Rotation Function

```
function R=RotY(beta)
This function takes the input beta (in radians about the Y-axis) and outputs the corresponding rotational
3x3 matrix R
Copyright Karam Mady (C) 2011
c=cos(beta);
s=sin(beta);
R = [c 0 s ; 0 1 0 ; -s 0 c];
```

....
....
Pinpoint Function

```
function new=pinpoint(tcp,pin0,beta)
Inputs:
beta : rotational angle about Y-axis in radians, scalar
pin0 : default platform pin position structure when beta=0, vector 3x1, in local coord
tcp : tcp location vector, 3x1
P=[pin0.x ; pin0.y ; pin0.z];
R = RotY(beta);
new=tcp+R*P;
```

....
....
The IK Function

```
function Q=invkin(S,X,Y,Z,C1,C2,C3)
```

C1,C2,C3 configuration 0/1 for each actuator
X,Y,Z are the TCP x,y,z-positions
S is a structure containing the kinematic parameters
Q is the output of actuator positions
beta = betafunction(S,Y,Z);
tcp = [X ; Y ; Z];
Ap = pinpoint(tcp,S.Ap,beta);
Bp = pinpoint(tcp,S.Bp,beta);
Dp = pinpoint(tcp,S.Dp,beta);

if C1==0
$Q(1) = sqrt((S.L1)^2 - (S.Ab.y - Ap(2))^2 - (S.Ab.z - Ap(3))^2) + Ap(1);$
else
$Q(1) = -sqrt((S.L1)^2 - (S.Ab.y - Ap(2))^2 - (S.Ab.z - Ap(3))^2) + Ap(1);$
end
if C2==0
$Q(2) = sqrt((S.L2_1)^2 - (S.Bb.y - Bp(2))^2 - (S.Bb.z - Bp(3))^2) + Bp(1);$
else
$Q(2) = -sqrt((S.L2_1)^2 - (S.Bb.y - Bp(2))^2 - (S.Bb.z - Bp(3))^2) + Bp(1);$
end
if C3==0
$Q(3) = sqrt((S.L3_1)^2 - (S.Db.y - Dp(2))^2 - (S.Db.z - Dp(3))^2) + Dp(1);$
else
$Q(3) = -sqrt((S.L3_1)^2 - (S.Db.y - Dp(2))^2 - (S.Db.z - Dp(3))^2) + Dp(1);$
end

for(i=1:3)   Q must be real to avoid system problems... Realize Q
if(isreal(Q(i)))
Q(i)=9e9;   If Q is not real, make it equal to big number
end;
end;

Here are the first working.. It has been replaced by a function for each step.

X,Y,Z are the TCP x,y,z-positions
S is a structure containing the kinematic parameters
Q is the output of actuator positions

Q(1)=sqrt((S.L1)$^2 - (y_Ab - y_Ap)^2 - (z_Ab - z_Ap)^2) + x_Ap;$
$Q(2) = sqrt((S.L2_1)^2 - (y_Bb - y_Bp)^2 - (z_Bb - z_Bp)^2) + x_Bp;$
$Q(3) = sqrt((S.L3_1)^2 - (y_Db - y_Dp)^2 - (z_Db - z_Dp)^2) + x_Dp;$

alfa=-acosd((x$_Dp - Q(3))/sqrt(L3_1^2 - (y_Db - y_Dp)^2));$

R= [cosd(alfa) 0 sind(alfa); 0 1 0; -sind(alfa) 0 cosd(alfa)];

[x$_Ap; y_Ap; z_Ap] = [X; Y; Z] + R * [x_Ap - X; y_Ap - Y; z_Ap - Z];$
$[x_Bp; y_Bp; z_Bp] = [X; Y; Z] + R * [x_Bp - X; y_Bp - Y; z_Bp - Z];$

$[x_Dp; y_Dp; z_Dp] = [X; Y; Z] + R * [x_Dp - X; y_Dp - Y; z_Dp - Z];$

Q(1)=sqrt((S.L1)$^2 - (S.Ab.y - Ap(2))^2 - (S.Ab.z - Ap(3))^2) + Ap(1);$
$Q(2) = sqrt((S.L2_1)^2 - (S.Bb.y - Bp(2))^2 - (S.Bb.z - Bp(3))^2) + Bp(1);$
$Q(3) = sqrt((S.L3_1)^2 - (S.Db.y - Dp(2))^2 - (S.Db.z - Dp(3))^2) + Dp(1);$

....
....
$LSQNONLINFunction$

Into lsqnonlin we will send measured X,Y,Z positions from laser tracker
and measured motor values q1,q2,q3
The optimisation should return L1-L6

X0 = [1.4]; Initial link lengths
LB=[1]; Lower limit
UB=[2]; Upper link length limit
OPTIONS=optimset('Display','on','TolFun',1e-100,'TolX',1e-100,'MaxFunEvals',1000);

P.X=X; TCP Measurements from laser tracker
P.Y=Y;
P.Z=Z;
P.Q=Q; We are pretending this is a measurement of motor positions
P.S=S;
P.C=[C1 C2 C3];

L_opt = LSQNONLIN(@(L) myfun(L,P),X0,LB,UB,OPTIONS)

....
....
Myfun Function

function F = myfun(L,P)
$P.S.L1 = L(1);$
$P.S.L2_1 = L(2);$
$P.S.L2_2 = L(2);$
$P.S.L3_1 = L(1);$
$P.S.L3_2 = L(1);$
$P.S.L3_3 = L(1);$

The Q below is estimated from measured TCP position
Q = invkin(P.S,P.X,P.Y,P.Z,P.C(1),P.C(2),P.C(3))

F(1)=P.Q(1)-Q(1);
F(2)=P.Q(2)-Q(2);
F(3)=P.Q(3)-Q(3);

Optimisation function: difference between measured P.Q and estimated Q

from P.X,Y,Z
F(1) = 1e1*(P.Q(1)-Q(1));
F(2) = 1e1*(P.Q(2)-Q(2));
F(3) = 1e1*(P.Q(3)-Q(3));

$[LF]$

....
....
Found the Links by IK of Tracker readings

function $L_exp = lcalculated(S, X, Y, Z, Q, X_exp, Y_exp, Z_exp)$;

This function will find the L1-L6 by given calculated Q1-Q3 and X,Y,Z
tested by the lazer machine... first test we use the same entered X,Y,Z by the user.

SS=setupp;
$X_exp = 1.49$;
$Y_exp = 0$;
$Z_exp = Z$;
$X_exp = 1.25$;
$Y_exp = Y$;
$Z_exp = 1.46$;

beta = betafunction(S,Y,Z);
$tcp_exp = [X_exp; Y_exp; Z_exp]$;
$Ap_exp = pinpointexp(tcp_exp, S.Ap, beta)$;
$Bp_exp = pinpointexp(tcp_exp, S.Bp, beta)$;
$Dp_exp = pinpointexp(tcp_exp, S.Dp, beta)$;

$L_exp(1) = sqrt((Q(1) - Ap_exp(1))^2 + (S.Ab.y - Ap_exp(2))^2 + (S.Ab.z - Ap_exp(3))^2)$;
$L_exp(2) = sqrt((Q(2) - Bp_exp(1))^2 + (S.Bb.y - Bp_exp(2))^2 + (S.Bb.z - Bp_exp(3))^2)$;
$L_exp(3) = L_exp(2)$;
$L_exp(4) = sqrt((Q(3) - Dp_exp(1))^2 + (S.Db.y - Dp_exp(2))^2 + (S.Db.z - Dp_exp(3))^2)$;
$L_exp(5) = L_exp(4)$;
$L_exp(6) = L_exp(4)$;
end

....
....
Reverse IK from Tracker readings TCP to Link Lengths

$functionQQ = invkin2(S, X, Y, Z, X_exp, Y_exp, Z_exp, C1, C2, C3, Q, L_exp)$
S=setup;

C1,C2,C3 configuration 0/1 for each actuator

$L_exp = lcalculated(S, X, Y, Z, Q, X_exp, Y_exp, Z_exp)$;
beta = betafunction(S,Y,Z)
$tcp_exp = [X_exp; Y_exp; Z_exp]$;
$Ap_exp = pinpointexp(tcp_exp, S.Ap, beta)$;
$Bp_exp = pinpointexp(tcp_exp, S.Bp, beta)$;

$Dp_exp = pinpoint(tcp_exp, S.Dp, beta);$

if C1==0
$QQ(1) = sqrt((L_exp(1))^2 - (S.Ab.y - Ap_exp(2))^2 - (S.Ab.z - Ap_exp(3))^2) + Ap_exp(1);$
else
$QQ(1) = -sqrt((L_exp(1))^2 - (S.Ab.y - Ap_exp(2))^2 - (S.Ab.z - Ap_exp(3))^2) + Ap_exp(1);$
end
if C2==0
$QQ(2) = sqrt((L_exp(2))^2 - (S.Bb.y - Bp_exp(2))^2 - (S.Bb.z - Bp_exp(3))^2) + Bp_exp(1);$
else
$QQ(2) = -sqrt((L_exp(2))^2 - (S.Bb.y - Bp_exp(2))^2 - (S.Bb.z - Bp_exp(3))^2) + Bp_exp(1);$
end
if C3==0
$QQ(3) = sqrt((L_exp(4))^2 - (S.Db.y - Dp_exp(2))^2 - (S.Db.z - Dp_exp(3))^2) + Dp_exp(1);$
else
$QQ(3) = -sqrt((L_exp(4))^2 - (S.Db.y - Dp_exp(2))^2 - (S.Db.z - Dp_exp(3))^2) + Dp_exp(1);$
end
....
....
Function For The Reverse IK

function $beta_exp = betaexpfunction(S, Y_exp, Z_exp);$

$L4 = 0.5 * (sqrt(S.Dp.z^2 + S.Ep.z^2 - 2 * S.Dp.z * S.Ep.z * cosd(120)));$
$L5 = sqrt(S.Dp.z^2 - L4^2);$
$LauxXZ_exp = sqrt(S.L3_1^2 - L4^2 - (Y_exp - S.Db.y)^2);$
$beta_exp = acosd((Z_exp - S.Db.z)/(LauxXZ_exp + L5));$

....
....
Foundation Of Optimal Path (Line Path) Main Program

function opts=setupp

S=setup;

XX=2.1;
Y=0.0;
Z=1.45;
C1=1;
C2=1;
C3=1;
N=30;
SS=optimalpath(XX,Y,Z,C1,C2,C3,S,N);

opts.S=S;
opts.SS=SS;

....
....
Foundation Of Optimal Path (Line Path)

function op=optimalpath(XX,Y,Z,C1,C2,C3,S,N)

```
for(i=1:N-1)      The number of rows, ie. the number of measurements
XX(i+1,:)=XX(i,:)- (XX(1,:)-0.7)/N;
Y(i+1,:)=Y(i,:)+ 0.6/N;
Z(i+1,:)=Z(i,:)- (Z(1,:)-0.8)/N;
end;
op.tracker = [XXYZ];

for(i=1:N)
C1=1;
C2=1;
C3=1;
op.enc(i,:)=invkin(S,XX(i,:),Y(i,:),Z(i,:),C1,C2,C3);
if op.enc(i,1)< 0
C1=0;
op.enc(i,:)=invkin(S,XX(i,:),Y(i,:),Z(i,:),C1,C2,C3);
end
if op.enc(i,2)< 0
C2=0;
op.enc(i,:)=invkin(S,XX(i,:),Y(i,:),Z(i,:),C1,C2,C3);
end
if op.enc(i,3)< 0
C3=0;
op.enc(i,:)=invkin(S,XX(i,:),Y(i,:),Z(i,:),C1,C2,C3);
end
op.C(i,:)=[C1 C2 C3];
end;

for(i=1:N)
if enc(i,1)>3
M(i)=N-1
elseif enc(i,2)>3
M(i)=N-1
elseif enc(i,3)>3
M(i)=N-1
else

for (j=1:M(i))
FINAL_TRACKER(j, :) = tracker(i, :)
FINAL_ENC(j, :) = enc(i, :)
FINAL_C(j, :) = C(i, :)
end
end
end

op.tracker=[2.1000       0       1.4500;
1.9600       0.0600      1.3850;
1.8200       0.1200      1.3200;
1.6800       0.1800      1.2550;
1.5400       0.2400      1.1900;
1.4000      0.3000       1.1250;
1.2600       0.3600      1.0600;
1.1200       0.4200      0.9950;
0.9800       0.4800      0.9300;
0.8400       0.5400      0.8650];

op.enc =[ 1.2316       1.2611       1.5972;
1.0388       1.1325      1.2883;
1.0231   0.9338   1.2141;
0.7710   0.8978   0.8240;
0.2959   0.6004   0.7293;
0.4357   0.2118   0.8090;
0.3537   0.6327   0.2256;
0.0305   0.0271   0.3545;
0.1878   0.5606   1.7909;
1.6422   1.5605   0.0081];

op.C =[ 1   1   1;
1   1   1;
1   1   1;
1   1   1;
1   1   1;
1   1   1;
1   1   1;
1   1   1;
1   1   0;
0   0   1];
```

....
....
Complex Search Method (Main Program)

Complex optimisation for L1-L6 of the Gantry-Tau

$N_params = 3;\quad numberofparameters : L1, L2_1, L3_1$
$N_pop = 2_p^N arams + 1;$

fcn$_name =' fitness_function';\quad nameoffunctiontomaximize$

bounds=[1.0*ones(1,N$_params); 1.3 * ones(1, N_params)];$

bounds = [1.23   1.07   1.23 ;
1.27   1.11   1.27];

$gen_max = 2000;$ number of times to evaluate fcn$_name;$

$x_start = [];$ let the complex function initialize the population randomly

$fit_start = [];$ let the complex funciton initialize the fitness;

$fcn_opts = setupp;$ When you have experiments, replace tracker values in setupp
....
crunch numbers!
tic;
$[x_best, fit_best, x_pop, fit_popstats] = complexmethod(fcn_name, bounds, gen_max, x_start, fit_start, fcn_opts);$
timtoc=toc;

$fprintf('Totaltime = fs.Timepergeneration = fsn', timtoc, timtoc/gen_max);$
$fprintf('FinalRMSE = fn', -fit_best)$

the rest is plotting results

figure(1)
loglog(-stats.trace$_fitness,'.')$
$xlabel('Generation')$
$ylabel('RMSE')$

x_best

....
....
Complex Search Method

function $[x_best, fit_best, x_pop, fit_popstats] =$
complexmethod(fcn_name,$bounds$,gen_max,x_start,fit_start,fcn_opts,$complex_opts$)
if nargin<2
error('Insufficient arguements')
end
if nargin<3
gen$_max = 1000;$
$elseifisempty($gen$_max)$
gen_max=1000;
end

if nargin<4
$x_start = []$;
end
if nargin<5
$fit_start = []$;
end
if nargin<6
$fcn_opts = []$;
end
if nargin<7
$complex_opts.alpha = 1.3$ constant determines how far to overshoot centroid
$complex_opts.n_r = 4$; constant for adjusting to repeated unimprovements
elseif isempty($complex_opts$)
$complex_opts.alpha = 1.3$; constant determines how far to overshoot centroid
$complex_opts.n_r = 4$; constant for adjusting to repeated unimprovements
end

$fcn_str = ['[fitness, x] =' fcn_name'(x, fcn_opts);']$; string to determine fitness
output of x allows function to change x if desired

$fit_best = -inf$; initial best fitness

$n_params = size(bounds, 2)$;

$gen = 1$; initialize number of generations

$alpha = complex_opts.alpha$; copy values
$n_r = complex_opts.n_r$;

if isempty(x_start)
$f_excess = 1.5$; factor to increase population over minimum (must be >=1)
$n_pop = round((n_params + 1) * f_excess)$;
$x_pop = ones(n_pop, 1) * bounds(1, :) + (ones(n_pop, 1) * (bounds(2, :) - bounds(1, :))). * rand(n_pop, n_params)$;
else

$x_pop = x_start;$
$n_pop = size(x_pop, 1);$

end

initialize fitness
if isempty((fit_start)
$fit_pop = zeros(1, n_pop);$
for $i = 1 : n_pop$
$x = x_pop(i, :);$
$eval(fcn_str);$ get fitness for each x
$x_pop(i, :) = x;$
$fit_pop(i) = fitness;$
gen=gen+1;
end
else
$fit_pop = fit_start;$
if numel$(fit_pop) \ = n_pop$
$error('Incorrectsizeoffit_pop');$
end
end

$stats.trace_fitness = zeros(1, gen_max);$
while $(gen < gen_max)$

$[fit_bestidx_best] = max(fit_pop);$ find best fitness
$x_best = x_pop(idx_best, :);$ find best params
$[fit_worstidx_worst] = min(fit_pop);$ find worst
$x_worst = x_pop(idx_worst, :);$ find best params

$x_centroid = (sum(x_pop) - x_worst)/(n_pop - 1);$ calc centroid of all but worst x

$x_new = x_centroid + alpha * (x_centroid - x_worst);$ "mirror" worst point through centroid

$idx = find(x_new < bounds(1, :));$ find params out of bounds
$x_new(idx) = bounds(1, idx);$
$idx = find(x_new > bounds(2, :));$
$x_new(idx) = bounds(2, idx);$

$x = x_new;$
$eval(fcn_str);$ get fitness for new x
$x_new = x;$
$fit_new = fitness;$
$stats.trace_fitness(gen) = fitness;$
$gen = gen + 1;$ increment generation

if $fit_new < fit_worst$ check if the new point is still worst
$k_r = 1;$ reset number of iterations
while $(fit_new < fit_worstgen < gen_max)$
$epsilon = (n_r/(n_r + k_r - 1))^{(}(n_r + k_r - 1)/n_r);$
$k_r = k_r + 1;$ increment number of iterations
R=rand;
$x_store = x_new;$ store previous x$_new$
x$_new = (x_new + epsilon * x_centroid + (1 - epsilon) * x_best)/2 + (x_centroid - x_best) * (1 - epsilon) * (2 * R - 1);$

$idx = find(x_new < bounds(1, :));$ find params out of bounds
$x_new(idx) = bounds(1, idx);$

$idx = find(x_new > bounds(2, :));$
$x_new(idx) = bounds(2, idx);$

$x = x_new;$
$eval(fcn_str);$ get fitness for new x
$x_new = x;$
$fit_new = fitness;$
$stats.trace_fitness(gen) = fitness;$
$gen = gen + 1;$ increment generation
end
end
$x_pop(idx_worst, :) = x_new;$ replace worst params with new params
$fit_pop(idx_worst) = fit_new;$
end

$[fit_bestidx_best] = max(fit_pop);$ find best fitness
$x_best = x_pop(idx_best, :);$ find best params

....
....
Fitness Function

$function[fitnessx] = fitness_function(x, opts)$
S=opts.S;
SS=opts.SS;
z=opts.SS.enc;

S.L1=x(1);
$S.L2_1 = x(2);$
$S.L3_1 = x(3);$
N=size(z);
for(i=1:N(1)) The number of rows, ie. the number of measurements
z$_hat(i, :)$=invkin(S,SS.tracker(i,1),SS.tracker(i,2),SS.tracker(i,3),SS.C(i,1),SS.C(i,2),SS.C(i,3)); Double-check if inverse kinematics give real numbers
for(j=1:3)
if(isreal(z$_hat(i, j)$))
$z_hat(i, j) = 9e9;$
$end;$
$end;$
$end;$
E=sqrt(mean((z-z$_hat).^2$)); root mean squared error
fitness=-sum(E); minimize error by maximizing fitness

....
....
Line Path

function UI=userinput;
exp=[1.5 0.0 1.45 ;
1.4 0.1 1.40 ;
1.3 0.2 1.35 ;
1.2 0.3 1.30]
UI.X=exp(:,1);
UI.Y=exp(:,2);
UI.Z=exp(:,3);
UI.C1 = 1;

UI.C2 = 1;
UI.C3 = 1;