Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Simen H. Ellingsdalen

Improving the Accuracy of a Five-Axis Machine Tool by Laser Tracker Calibration

Geometric Error Compensation

Master's thesis in Mechanical Engineering
Supervisor: Prof. Knut Sørby
June 2021

NTNU
Norwegian University of
Science and Technology

Simen H. Ellingsdalen

Improving the Accuracy of a Five-Axis Machine Tool by Laser Tracker Calibration

Geometric Error Compensation

Master's thesis in Mechanical Engineering
Supervisor: Prof. Knut Sørby
June 2021

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of
Science and Technology

Preface

This master's thesis in TPK4940 Manufacturing Technology is written in the spring semester of 2021 by Simen Ellingsdalen as the concluding work of a two-year master program in mechanical engineering at NTNU - the Norwegian University of Science and Technology. It builds upon the work done in the fall of 2020 in the specialization project, and makes up 30 of the 120 credits of the study program. Although the basic principles of the problem are easy to understand, the work has been demanding in its complexity. Working on the project has been challenging, fun and interesting. I would like to give thanks to my supervisor - Professor Knut Sørby for en excellent collaboration and smooth execution of the experimental work. I would also like to thank the staff of the workshop at Valgrinda, led by Arild Sæther, for accommodating me.

Abstract

A key attribute of a well functioning machine tool is its high repeatability and accuracy. As the number of axes in a machine tool increases, so does the potential for error. This work focuses on determining the geometrical errors of all axes of a five-axis machine tool by the use of a standard laser tracker. The measuring setup and the method of calculating the errors of the rotational axes are described in detail. The variability of the measured results is evaluated, and measures for reducing the measurement uncertainty of laser tracker measurements of machine tool axis motion is proposed. Based on the measured errors, a compensation strategy is developed. The method of compensation is based on reconstruction of the NC-code, and Python scripts are used to generate the compensated NC-data. The compensation strategy's effectiveness is verified for the translational errors of the linear axes, and the results are comparable to those achieved by other researchers.

Sammendrag

En av hovedegenskapene til en velfungerende verktøymaskin er dens repeterbarhet og nøyaktighet. Når antall akser i en verktøymaskin øker, øker også potensialet for feil. Denne oppgaven fokuserer på å finne alle de geometriske feilene til en fem-akse verktøymaskin ved å bruke en vanlig laser tracker. Måleoppsettet og metoden som er brukt for å kalkulere de geometriske feilene til rotasjonsaksene er beskrevet i detalj. Variasjonen i de målte fielene blir behandlet og vurdert, og måter å redusere måleusikkerheten ved måling av aksebevegelsene i verktøymaskiner med laser tracker blir foreslått. Basert på de målte feilene utvikles det en kompensasjonsstrategi. Kompensasjonsmetoden er basert på å endre NC-koden, og Python script blir brukt til å generere kompenserte NC-data. Effektiviteten til kompensasjonsstrategien blir vist på translasjonsfeilene til de lineære aksene, og resultatene er sammenlignbare med de som andre forskere har oppnådd.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

As machine tools transitioned from being entirely manual to being numerically controlled, opportunities for improving the geometrical accuracy expanded beyond mechanical optimization. The geometrical accuracy of machine tools can be improved by identifying the geometric errors of the machine tool and applying a compensation algorithm to eliminate them. The interest for numerical compensation based on software saw an increase in the 1970's. In 1977, Prof. R. Hocken received the CIRP Taylor Medal, for implementing this kind of error compensation on a Moore N.5 coordinate measuring machine (CMM) [1].

The incentive for implementing numerical compensation on machine tools is not the achievable precision in itself, as some companies achieved impressive sub-micron level accuracy through mechanical optimization alone. However, the investment in both time and manual labour was substantial [2]. Some of the benefits and limitations of numerical compensation is presented in the following section.

1.2 Why Perform Numerical Compensation of a Machine Tool?

Benefits of performing a numerical compensation of geometric errors in a machine tool:

- Compensation of geometric errors means that the movement of the machine axes are closer to the ideal intended movement. This will consequently result in a part geometry in closer conformance to the modeled geometry. This means higher part accuracy, and thereby higher part quality.
- By measuring the geometric errors of the machine tool with consequent re-compensation throughout the life time of the machine tool, volumetric accuracy is assured and maintained. Geometric errors are sure to "evolve" or change with time as a result of aging, wear, collisions, relocation of the machine tool, changes in the thermal environment or changes in the foundation making the need to repeat the compensation process evident [3].
- As D.C. Thompson states [4]: "The availability of modern computational tools makes the application of active and precalibrated error compensation an economical alternative to designing and building for absolute accuracy. Thus the mechanical accuracy of the machine need only be sufficient to allow error compensation to the desired level of accuracy..." What he means is that the opportunities available with modern computational tools, allows for machines to be produced with larger tolerances than before because the potential for numerical compensation is so great. This means that machine tools can be produced faster and cheaper while still meeting the high demand for accuracy in the industry.

However, numerical compensation is not the answer to every aspect of machine tool accuracy. The limitations of the principle are the following:

- Although the geometric errors of the machine tool can be compensated for, stability cannot be compensated into a machine tool. Long term stability is a feature which must be built into the machine from the beginning.
- Geometric changes arising from thermo-elastic deformations may still be present.
- The degree to which the machine tool is able to reproduce a movement remains the limit for the achievable accuracy.
- Numerical compensation of a machine tool may involve having to drive several axes simultaneously during cutting that otherwise would be stationary. This may result in reduced stiffness and may introduce additional errors if the driven axes have significant reversal error, limited least increment step or other positioning accuracy characteristics that vary with the direction of motion.
- A compensation method which includes compensation in the functional orientation ideally requires three orthogonal rotational axes to compensate for all angular errors in all axes. However, few machine tool are configured this way. Typically, with certain axis-positions, the spindle of the machine tool can end up being parallel with the rotational axis. In this singular configuration, the motion needed to compensate for angular errors may not be available to the machine control system. This can lead to very rapid motion in machine axes and consequently large elastic deformations in the machine structure due to large acceleration forces. If the machine is cutting, this might lead to rough surface quality or other part errors. The rapid motion may also increase the power consumption of the axis drives which again may lead to increased thermo-elastic deformation in the machine structure. Compensation of the functional orientation in close proximity to singular configurations of machine axes should therefore be handled with great care.
- As stated in the last point on benefits of numerical compensation, the geometrical requirements of the machine tool components and assembly may be relaxed to a certain degree if numerical compensation is added. However, the tolerances of the machine tool parts and assembly may also greatly impact the stiffness and repeatability of the motion of the machine axes. Misalignment of the spindle may also impact tool wear. This should be taken into account when designing machine tools to be numerically compensated [3].

1.3 Problem Description

When machining large parts on five-axis machines, geometrical errors, especially those associated with the rotational axes, may propagate as relatively large deviations in both position and inclination of geometrical features. Before tracking interferometers were invented, measuring and compensating for the geometrical errors of CNC machine tools and CMMs was a slow and tedious process which could go on for several days. With the entry of high accuracy tracking laser interferometers, the measuring and compensation process has become significantly faster, and may be executed in a matter of a few hours. The commercial systems utilize specialized tracking interferometers made only for measuring the axis-motions of machine tools and CMMs. Although these systems offer unprecedented measuring precision, using a more versatile conventional laser tracker may be more appropriate in some cases.

- Can acceptable measurement uncertainty be achieved using a versatile laser tracker?
- Based on the measured geometrical errors of the machine tool; what is the achievable accuracy improvement using a self-developed compensation strategy?

1.4 Project Scope

A method for measuring the geometrical errors of the linear axes was implemented successfully in the specialization project in the fall of 2020 (see appendix B). However, the measured errors were the result of only one round of measurement, and the variability of the measured errors could therefore not be evaluated. The measurement of the linear axes will be repeated in this work using the same method as in the specialization project report. The variability of the resulting errors will be evaluated and discussed. The geometric errors of the rotational axes will be measured using a similar method to that for the linear axes. A compensation strategy will be developed to compensate for the measured geometrical errors.

Rigid body motion is assumed, meaning that the measured errors are assumed not to be influenced by any deformations due to tool mass and/or workpiece mass. The measured errors of one axis are also assumed independent of the position of the other axes.

1.5 Thesis Structure

The thesis is structured around the IMRaD (Introduction, Method, Results and Discussion) framework, however, it also includes a theory chapter.

The theory chapter starts by introducing machine tool kinematics with focus on five-axis machines. The kinematics of the machine tool used in this work is also presented. Chapter 2.2.1 aims to explain how the motion of the machine axes can be represented mathematically, and builds the foundation for the kinematic model of a machine tool. Chapter 2.3 and 2.4 introduce the errors that exist in the linear and rotational axes respectively, according to the international standards. Based on the mathematical representation of rigid body motion and the error definition, chapter 2.5 presents the quasi-static error models of the linear and rotational axes. Chapters 2.6 and 2.7 presents ways of measuring geometric errors in machine tools and introduce the basic principle of measurement uncertainty. In chapter 2.8, some of the sources of uncertainty specific to the measuring instrument used in this work are discussed. The final part of the theory chapter is dedicated to numerical compensation. Chapter 2.9 gives an overview of some of the basic principles of numerical compensation, while chapter 2.9.1 introduces different types of geometric compensation.

The method chapter usually only presents the method used to obtain the results. In this work, the results are strictly defined as the measured errors of the machine axis movement before and after compensation, and are presented in chapter 4. Although the principle of the measurement method and error calculation was not developed during the project period, the implementation and execution can be considered part of the results. Thus, chapter 3 also include part of the project results and applies to the curve fitting and compensation procedure as well. Some of the scripts produced and used in the project are presented in chapter 3, and the rest can be found in the appendix.

As previously mentioned, chapter 4 contains the measured errors of the machine tool axis motion before and after compensation. The results are presented, and discussed briefly. The variability of the results and the uncertainty of the measurement is discussed in depth in chapter 5. Finally, in chapter 6, a conclusion is made and further work suggested.

The contents of an appendix are mostly for reference. Although that is true for most of the appendix in this thesis, I would like to highlight the conference paper in appendix A. The paper was submitted to CIRP (College International pour la Recherche en Productique) on May 4th as a proposed contribution to the 15th edition of the conference on Intelligent Computation in Manufacturing Engineering (ICME). the conference was to be held on 14-16 July 2021 in the Gulf of Naples, Italy, but will be held digitally due to Covid-19. The paper was accepted on May 26th, and will be included in the Technical Programme of the CIRP ICME '21 as well as in the Proceedings.

Chapter 2

Theory

2.1 Machine Tool Kinematics

Machine tools can be configured in a number of different ways and may have anywhere from a single axis to a double digit number of axes. An axis represents a degree of freedom, DOF, and the number varies depending on the number of the axes the machine tool has. The movements required to achieve the desired size and shape of the workspace leads to typical kinematics solutions and machine configurations. An example is the production of cylindrical parts or other parts of rotational symmetry. For this kind of production, a cylindrical workspace is appropriate which may be achieved using two linear axes. Machine tools of this type are called turning lathes. Milling machines generally have a minimum of three linear axes making up a workspace with a square or rectangular cross-section. The position of the linear axes vary. In some machine tools, all of the linear motion is made at the tool end while the workpiece is stationary. In other machines, the X- and Y-axes may be located at the workpiece end, while the Z-axis is at the tool end of the machine tool.

As previously mentioned, machine tools may also have more than just two or three axes. Machine tools with five axes have become very popular because of their great versatility. Five axes allows the tool to not only be positioned at any position in the working volume, but the orientation of the tool relative to the workpiece may also be varied. This allows for several faces of a workpiece to be machined in a single setup, and allows greater freedom regarding tooling. An endmill might for instance be used, oriented at a 45° angle relative to the workpiece, to make a chamfer instead of using a dedicated chamfering tool. The configuration of the five axes vary, and many different machine configurations exist. A common way of characterizing the configuration of the different axes is by a sequence of capital letters. Starting from the left, the first letter is the axis closest to the workpiece. R represents a rotational axis, and L a linear axis. The letter all the way to the right is the axis closest to the tool. LLLRR, RRLLL and RLLLR are three examples of configurations of three linear and two rotary axes, which is the most common in five-axis machine tools. A short description of each of these configurations follows [5]:

- LLLRR: A spindle head with two axes of rotation is mounted at the end of the X-, Y- and Z-axes. One of the axes of rotation twists the spindle head, while the other tilts it. This designation does not define whether the X-, Y- and Z-motions are made at the workpiece end of the machine or at the tool end. In Figure 2.1(a), the linear motion is made at the tool end. This machine configuration is typically used to machine large moulds and dies, and are referred to as gantry machine tools.
- RRLLL: This configuration has a worktable with two rotary axes fitted to it. One rotates the table, while the other tilts it. The tilt motion can be compared to a cradle motion. The three linear motions are typically made at the tool end of the machine, but the worktable may also be provided with one of the linear axes (see Figure 2.1(b)).
- RLLLR: The worktable is provided with one rotational degree of freedom giving it the ability to revolve around its own axis. At the tool end, the spindle is able to tilt or swivel. These types of machines are especially suited for tall cylindrical workpieces with various geometrical feauters along their perimiter (see Figure 2.1(c)).

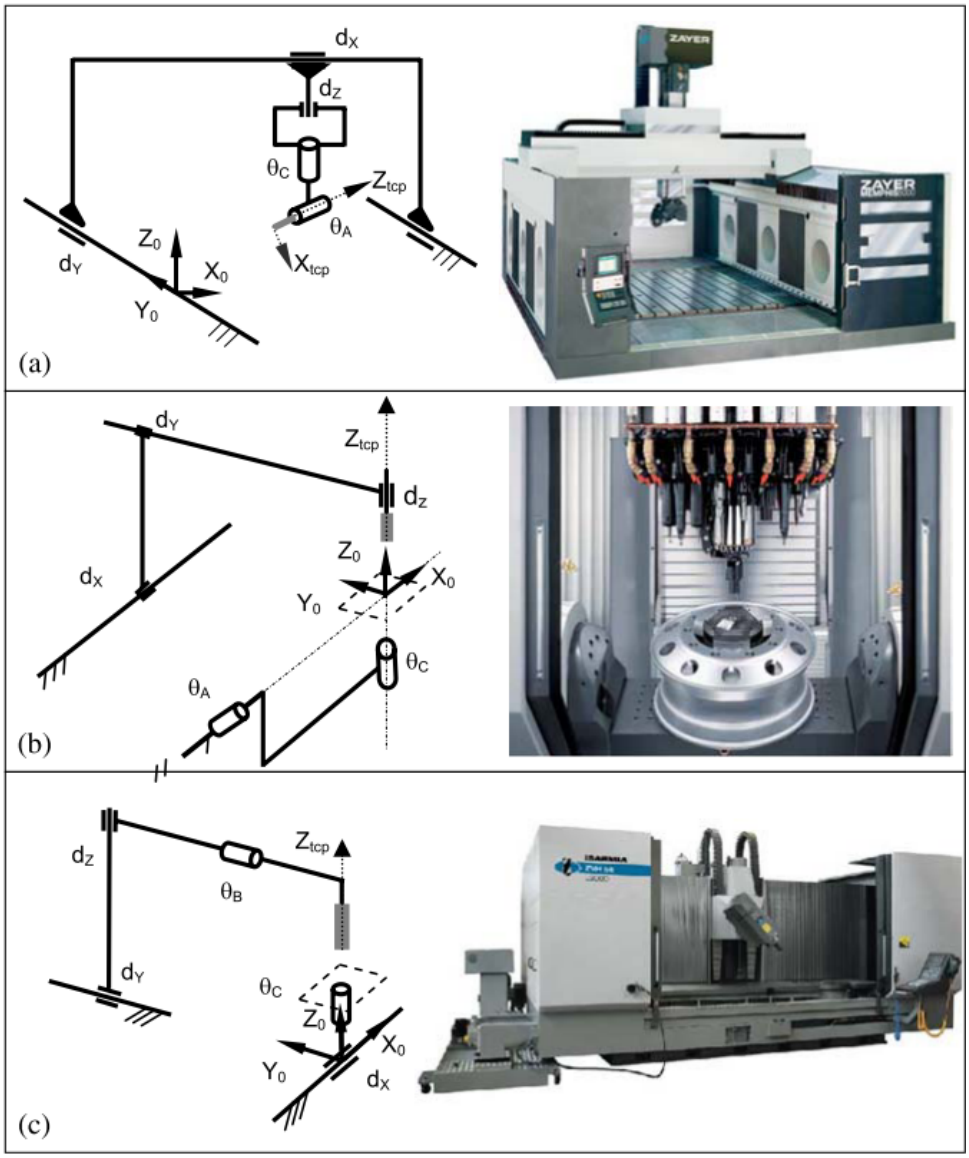


Figure 2.1: Three different configurations of five-axis machine tools along with their kinematic chains - **(a)**: LLLRR, **(b)**: RRLLL, **(c)**: RLLLR [5]

2.1.1 Kinematic Chain

When talking about the structure of a machine tool, and the configuration of the axes, the term *kinematic chain* is often mentioned. By making a quick web-search and navigating to the Wikipedia page - "Kinematic chain" one can read that "In mechanical engineering, a kinematic chain is an assembly of rigid bodies connected by joints to provide constrained (or desired) motion that is the mathematical model for a mechanical system" [6]. This describes it really well. Although nothing is entirely rigid, the components making up a machine tool may be assumed to be rigid bodies. The joints connecting these rigid bodies limit their relative motion, and may have one or more DOFs. Ken Waldron and Jim Schmiedleler describe a kinematic joint as "a connection between two bodies that constrains their relative motion" [7]. In *higher pair joints* contact happens at points or along lines, while contact in *lower pair joints* occurs over surfaces. The advantage with lower pair joints is the low wear and friction which can be achieved by distributing the load over a large contact area and trapping a thin layer of lubricant in the small gap between surfaces.

There are six different types of lower pair joints, and two of them are largely used in machine tools. The joint types in question are *prismatic* and *revolute* joints. Both only have one DOF. A Revolute joint consists of two surfaces cylindrical in shape, one internal surface, and one external. The joint is constructed so that rotation is only permitted on one of the bodies relative to the other. A prismatic joint is similar to the revolute joint in the sense that it has an internal and an external surface and allows only one degree of freedom. However, the degree of freedom of a prismatic joint is along the direction of extrusion of the two contacting surfaces. On the left side of figure 2.1, the kinematic chains of the machines on the right is displayed. In the figure, prismatic joints are denoted with a lower case d, and revolute joints with the greek letter θ.

2.2 Deckel Maho DMU 50 eVolution

The Deckel Maho DMU 50 eVolution (hereafter referred to as just "the DMU 50") is an example of a machine tool of the RRLLL configuration, but unlike the machine in figure 2.1(b), the two rotational axes in the DMU 50 are not orthogonal to each other. According to the ISO standard number 841 on coordinate system and motion nomenclature, "A, B and C define rotary axes about linear axes X, Y and Z respectively" [8]. The C-axis of the DMU 50 follows this convention, but the B-axis is wrongly termed according to the standard since this rotational axis does not coincide with the Y-axis or any of the translational axes of the machine tool. The configuration of the axes in the kinematic chain is schematically visualized in Figure 2.2 (right) and in Figure 2.3.

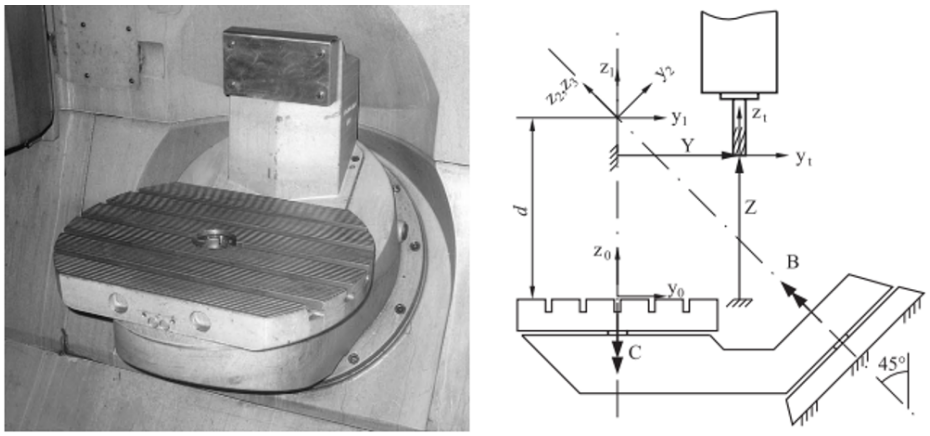


Figure 2.2: Deckel Maho DMU 50 eVolution axis configuration [9]

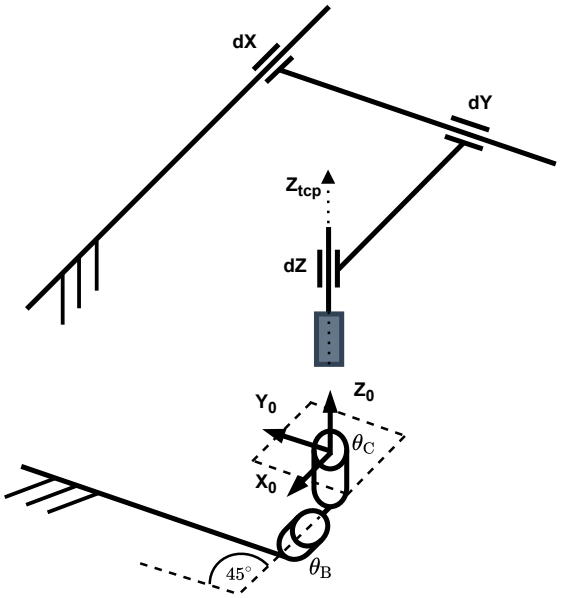


Figure 2.3: Kinematic chain of the Deckel Maho DMU 50 eVolution

2.2.1 Mathematical Representation of Machine Motion

As we now know the configuration of the axes and the way in which they are connected and move relative to each other, let us have a quick look at how the motions of the machine tool can be represented mathematically. The following section is based on the book "Modern Robotics - Mechanics, Planning And Control" by Kevin M. Lynch and Frank C. Park [10].

In order to completely describe a rigid body's position and orientation in three dimensional space, a minimum of six numbers is needed. A common way of representing a rigid body's position and orientation is to attach a reference frame to the body. The position and orientation of the body in relation to a fixed reference frame is then described using a 4 x 4 matrix called a homogeneous transformation matrix (HTM). A homogeneous transformation matrix takes on the form

$$T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_1 \\ r_{21} & r_{22} & r_{23} & p_2 \\ r_{31} & r_{32} & r_{33} & p_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad R \in SO(3), p \in \mathbb{R}^3 \tag{2.1}$$

The set of all transformation matrices are called the special eucledian group, SE(3), and they inhibit some properties worth noting:

Every transformation matrix in SE(3) has an inverse matrix of the form

$$T^{-1} = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R^T & -R^T p \\ 0 & 1 \end{bmatrix} \in SE(3). \tag{2.2}$$

T_1T_2, the product of two transformation matrices is also a transformation matrix in SE(3), transformation matrices are associative, meaning that $(T_1T_2)T_3$ is equal to $T_1(T_2T_3)$, they are generally not commutative, meaning that T_1T_2 is not equal to T_2T_1.

Not only can the 4 x 4 matrix describe the position and orientation of the rigid body, but it can also translate or rotate a vector or a frame in addition to change the representation of a vector or a frame

from coordinates in one frame to coordinates in another frame. Perhaps the main reason for using 4 x 4 matrices in this application is that these operations can be executed using simple linear algebra [10].

Any position and orientation of a rigid body in three dimensional space can be described by a position vector p $\in \mathbb{R}^3$, and a rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$. If we consider a space frame $\{s\}$ and a body frame $\{b\}$, the position of the rigid body can be described as

$$p = p_1\hat{x}_s + p_2\hat{y}_s + p_3\hat{z}_s \tag{2.3}$$

And the axes of the body frame $\{b\}$ can be described as

$$\hat{x}_b = r_{11}\hat{x}_s + r_{21}\hat{y}_s + r_{31}\hat{z}_s \tag{2.4}$$
$$\hat{y}_b = r_{12}\hat{x}_s + r_{22}\hat{y}_s + r_{32}\hat{z}_s \tag{2.5}$$
$$\hat{z}_b = r_{13}\hat{x}_s + r_{23}\hat{y}_s + r_{33}\hat{z}_s \tag{2.6}$$

which in matrix form is

$$p = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}, R = [\hat{x}_b, \hat{y}_b, \hat{z}_b] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2.7}$$

A rigid body only has three degrees of freedom in rotation, and therefore only three of the nine elements in the rotation matrix can be chosen independently. As such, the position and orientation of the rigid body can be described by six individual parameters. The six constraints on the rotation matrix can be described as follows:

1. The three column vectors of the rotation matrix represent the unit axes of the body frame in space frame coordinates, hence all three column vectors must be unit vectors and satisfy:

$$r_{11}^2 + r_{12}^2 + r_{13}^2 = 1 \tag{2.8}$$
$$r_{21}^2 + r_{22}^2 + r_{23}^2 = 1 \tag{2.9}$$
$$r_{31}^2 + r_{32}^2 + r_{33}^2 = 1 \tag{2.10}$$

2. The three column vectors of the rotation matrix must be orthogonal to each other, which is the same as the dot product equalling to zero: $\hat{x}_b \cdot \hat{y}_b = \hat{x}_b \cdot \hat{z}_b = \hat{y}_b \cdot \hat{z}_b = 0$ or:

$$r_{11}r_{12} + r_{21}r_{22} + r_{31}r_{32} = 0 \tag{2.11}$$
$$r_{12}r_{13} + r_{22}r_{23} + r_{32}r_{33} = 0 \tag{2.12}$$
$$r_{11}r_{13} + r_{21}r_{23} + r_{31}r_{33} = 0 \tag{2.13}$$

   Which can be written more compactly as a single set of constraints on the rotation matrix R,

$$R^T R = I, \tag{2.14}$$

   where R^{T} denotes the transpose of R and I denotes the identity matrix.

To follow convention and make sure that the coordinate frame is right-handed, which would mean that $\hat{x}_b \times \hat{y}_b = \hat{z}_b$, one last constraint must be placed on the rotation matrix. This is also equivalent to the determinant of the matrix being equal to one. The formula for calculating the determinant of a 3×3 matrix is given as

$$detM = a^T(b \times c) = c^T(a \times b) = b^T(c \times a). \tag{2.15}$$

By substituting the columns of R into the formula, the last constraint is made

$$detR = 1 \tag{2.16}$$

Had we allowed for left-handed coordinate systems as well, the constraint would have been $detR = \pm 1$. The set of 3×3 rotation matrices that follow the constraints stated above are called the special orthogonal group SO(3). A group in mathematics consists of a set of elements and an operation on two elements (matrix multiplication for SO(n)) such that for all A, B in the group, the properties listed below are satisfied:

- **Closure:** AB is also in the group
- **Associativity:** (AB)C = A(BC)
- **Identity element existence:** There exists an element A^{-1} such that $\mathrm{A}^{-1} = \mathrm{A}^{-}1\mathrm{A} = I$

Angular Velocity

Considering a coordinate frame with unit axes $\{\hat{x}, \hat{y}, \hat{z}\}$ attached to a rotating body, the time derivatives of the unit axes can be determined. The frame is examined at times t and t + Δt. The change in orientation can be expressed as $\Delta\theta$ - the rotation angle about some unit rotation axis passing through the center of the coordinate frame $\hat{w}$. The axis is coordinate free - not yet represented in any particular reference frame. As Δt approaches zero, the ratio $\Delta\theta/\Delta t$ becomes the rate of rotation $\dot{\theta}$, and $\hat{w}$ can be regarded as the instantaneous axis of rotation. The rotation axis $\hat{w}$ and rate of rotation $\dot{\theta}$ can be combined to define the angular velocity w as follows:

$$w = \hat{w}\dot{\theta} \tag{2.17}$$

The time derivative of each unit axis becomes:

$$\dot{\hat{x}} = w \times \hat{x}, \tag{2.18}$$
$$\dot{\hat{y}} = w \times \hat{y}, \tag{2.19}$$
$$\dot{\hat{z}} = w \times \hat{z}. \tag{2.20}$$

In order to represent the equations above in coordinates, a frame of reference in which to represent ω must be chosen. Considering a fixed-frame $\{s\}$ and a body frame $\{b\}$, the orientation of the body frame in fixed-frame coordinates can be expressed at time t by the rotation matrix $R(t)$. The first column of $R(t)$, $r_1(t)$, describes $\hat{x}$ in fixed-frame coordinates. Similarly, the second and third columns of $R(t)$, $r_2(t)$ and $r_3(t)$, express $\hat{y}$ and $\hat{z}$ respectively. At time t, let $\omega_s \in \mathbb{R}^3$ be the angular velocity expressed in fixed-frame coordinates, equations 2.18-2.20 can be expressed in fixed-frame coordinates as:

$$\dot{r_i} = \omega_s \times r_i, \qquad i = 1, 2, 3 \tag{2.21}$$

which may be rearranged into the following single 3×3 matrix equation:

$$\dot{R} = \begin{bmatrix} \omega_s \times r_1 & \omega_s \times r_2 & \omega_s \times r_3 \end{bmatrix} = \omega_s \times R \tag{2.22}$$

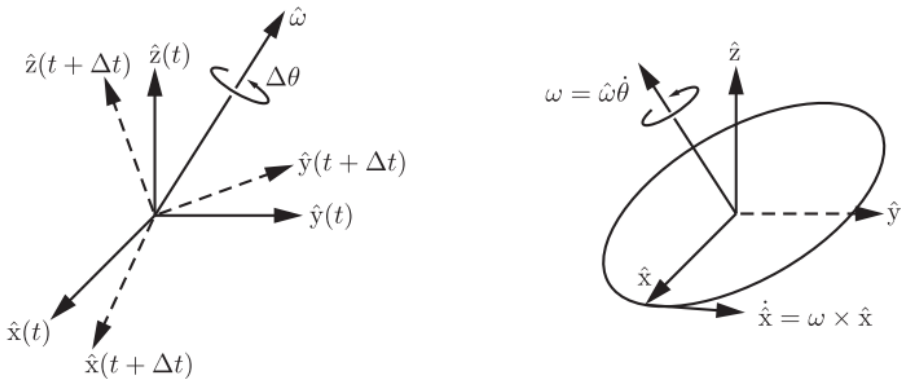


Figure 2.4: Left: The instantaneous angular velocity vector. Right: Calculating the angular velocity of the unit axis $\hat{x}$ [10].

To eliminate the cross product on the right hand side of equation 2.22, $\omega_s \times \mathrm{R}$ may be rewritten as $[\omega_s]\mathrm{R}$, where $[\omega_s]$ is the 3×3 skew-symmetric representation of $\omega_s \in \mathbb{R}^3$. Given a vector $x = [x_1 x_2 x_3]^T \in \mathbb{R}$, the skew-symmetric representation of the vector x is

$$[x] = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}. \tag{2.23}$$

$[x]$ being skew-symmetric means that

$$[x] = -[x]^T. \tag{2.24}$$

The set of all 3×3 real skew-symmetric matrices is called so(3) and is the lie algebra of the lie group SO(3).

A property of rotation matrices and skew-symmetric matrices is that given any $\omega \in \mathbb{R}^3$ and $R \in SO(3)$, the following is always true

$$R[\omega]R^T = [R\omega]. \tag{2.25}$$

With skew-symmetric notation, equation 2.22 can be rewritten as

$$[\omega_s]R = \dot{R}. \tag{2.26}$$

Post-multiplying by R^{-1} yields

$$[\omega_s] = \dot{R}R^{-1} \tag{2.27}$$

By writing the rotation matrices in explicit form with subscripts, ω_b can be obtained by ω_s using the subscript cancellation rule. In summary

$$\dot{R}R^{-1} = [\omega_s], \tag{2.28}$$
$$R^{-1}\dot{R} = [\omega_b] \tag{2.29}$$

relate the angular velocity ω represented in two different coordinates frames together.

Exponential Coordinate Representation of Rotation

Exponential coordinates for rotation parametrize a rotation matrix in terms of a rotation axis (represented by a unit vector $\hat{\omega}$) and an angle of rotation θ about that axis; the vector $\hat{\omega}\theta \in \mathbb{R}^3$ then serves as the three-parameter exponential coordinate representation of the rotation. If $\hat{\omega}$ and θ is written individually, the representation is termed the axis-angle representation of a rotation. The exponential coordinate representation can be interpreted in any of the following three ways equivalently:

- the axis $\hat{\omega}$ and rotation angle θ such that, if a frame initially coincident with $\{s\}$ were rotated by θ about $\hat{\omega}$, its final orientation relative to $\{s\}$ would be expressed by R
- the angular velocity $\hat{\omega}\theta$ expressed in $\{s\}$ such that, if a frame initially coincident with $\{s\}$ followed $\hat{\omega}$ for one unit of time (i.e., $\hat{\omega}\theta$ is integrated over this time interval), its final orientation would be expressed by R
- the angular velocity $\hat{\omega}\theta$ expressed in $\{s\}$ such that, if a frame initially coincident with {s} followed $\hat{\omega}$ for θ units of time (i.e., $\hat{\omega}$ is integrated over this time interval), its final orientation would be expressed by R

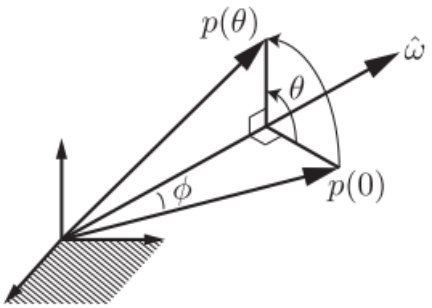


Figure 2.5: Vector $p(0)$ rotated by an angle θ about the rotation axis $\hat{\omega}$ to $p(\theta)$ [10].

In figure 2.5, the three dimensional vector p_0 is rotated by θ about $\hat{\omega}$ to $p(\theta)$. Since the rotation axis, $\hat{\omega}$ is of unit length, the rotation can be considered achieved by the vector $p(0)$ rotating at a constant rate of 1 rad/s from $t = 0$ to $t = \theta$. With $p(t)$ defined as the path traced by the tip of $p(0)$ as it rotates, the velocity of $p(t)$, denoted $\dot{p}$, is then given by

$$\dot{p} = \hat{\omega} \times p. \tag{2.30}$$

The angle Φ is considered constant. Then, the tip of the vector $p(t)$ traces out a circle with radius $||p||sin\Phi$ about the $\hat{\omega}$-axis. $\dot{p}$ is tangent to the circle path with magnitude $||p||sin\Phi$ according to eq. 2.30. The differential equation can be expressed as

$$\dot{p} = [\hat{\omega}]p \tag{2.31}$$

using the skew-symmetric representation of the rotation axis $\hat{\omega}$. The initial condition of the differential equation is $p(0) = 0$. The solution to this linear differential equation is given by

$$p(t) = e^{[\hat{\omega}]t}p(0) \tag{2.32}$$

and since t and θ are equivalent in this case, it may be written

$$p(t) = e^{[\hat{\omega}]\theta}p(0) \tag{2.33}$$

Now, by expanding the matrix exponential, $e^{[\hat{\omega}]\theta}$, in the series form and simplifying, we end up with Rodrigues' formula:

$$Rot(\hat{\omega}, \theta) = e^{[\hat{\omega}]\theta} = I + sin\theta[\hat{\omega}] + (1 - cos\theta)[\hat{\omega}]^2 \in SO(3). \tag{2.34}$$

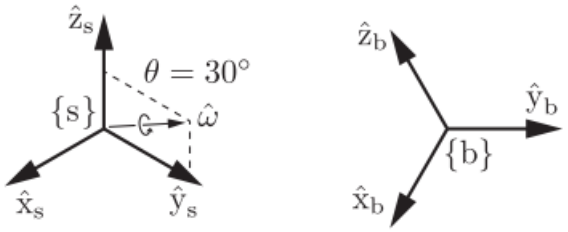


Figure 2.6: Body frame $\{b\}$ obtained by rotating a coordinate frame originally coincident with the fixed frame $\{s\}$ by an angle $\theta_1 = 30°$ about the rotation axis $\hat{\omega} = (0, 0.866, 0.5)$ [10].

As an example, considering the rotation angle of $\theta_1 = 30° = 0.524$ rad about the rotation axis $\hat{\omega}$ in figure 2.6 to obtain the orientation of body frame $\{b\}$ in fixed-frame, $\{s\}$ coordinates. The unit axis $\hat{\omega}$ is given as (0, 0.866, 0.5). The rotation matrix representation of the body frame $\{b\}$ can be calculated as

$$R = e^{[\hat{\omega}_1]\theta_1} \tag{2.35}$$

$$= I + sin\theta_1[\hat{\omega_1}] + (1 - cos\theta_1)[\hat{\omega}_1]^2 \tag{2.36}$$

$$= I + 0.5 \begin{bmatrix} 0 & -0.5 & 0.866 \\ 0.5 & 0 & 0 \\ -0.866 & 0 & 0 \end{bmatrix} + 0.134 \begin{bmatrix} 0 & -0.5 & 0.866 \\ 0.5 & 0 & 0 \\ -0.866 & 0 & 0 \end{bmatrix}^2 \tag{2.37}$$

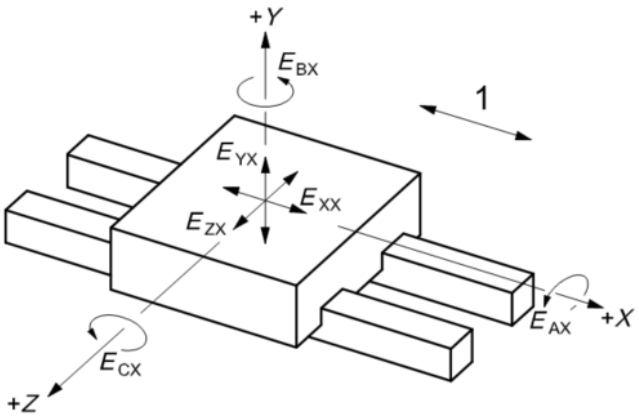
$$= \begin{bmatrix} 0.866 & -0.250 & 0.433 \\ 0.250 & 0.967 & 0.058 \\ -0.433 & 0.058 & 0.899 \end{bmatrix}. \tag{2.38}$$

The orientation of the frame $\{b\}$ can be represented by R or by the unit axis $\hat{\omega} = (0, 0.866, 0.5)$ and the angle $\theta_1 = 0.524$ rad , i.e., the exponential coordinates $\hat{\omega}_1\theta_1 = (0, 0.453, 0.262)$ [10].

2.3 Geometric Errors of Linear Axes

Each linear axis has six individual errors. Three translational and three angular errors. The translational errors are further divided into the linear positioning errors, and straightness errors. According to the naming convention in ISO 230-1 on Geometric accuracy of machines operating under no-load or quasi-static conditions [11], the geometric errors are denoted by a capital E, followed by a subscript where the first letter denotes the direction of the deviation and the second letter denotes the name of the axis. For example, E_{YX} denotes the translational deviation in Y-direction when the X-axis is moved (see figure 2.7). This type of error is also called a straightness error. If the letters in the subscript are the same, i.e. E_{XX}, the error is called the linear positioning error. This error is different from the straightness errors as it arises from errors in the control of the numerically controlled axis as well, rather than the purely geometric nature of the origin of the straightness errors. "The straightness and angular errors are considered pure geometric errors, whereas the linear displacement (positioning) errors are a function of both geometry and the axis drive system characteristics" [11].

As for the angular errors, they follow the same naming convention as the translational errors only differing in the first letter of the subscript. This first letter now represents the axis about which the angular deviation occurs. The rotational axes about the X- Y- and Z-axes are termed the A-, B- and C-axes respectively, as defined in ISO 841 [8]. As an example, the angular error E_{AX} is the angular deviation about the A-axis as the X-axis is moved (see figure 2.7). Still considering the X-axis, the angular errors E_{AX}, E_{BX} and E_{CX} are also referred to as the roll- yaw- and pitch angles of the axis respectively [11].



Key

1	X-axis commanded linear motion
E_{AX}	angular error motion around A-axis (roll)
E_{BX}	angular error motion around B-axis (yaw)
E_{CX}	angular error motion around C-axis (pitch)
E_{XX}	linear positioning error motion of X-axis; positioning deviations of X-axis
E_{YX}	straightness error motion in Y-axis direction
E_{ZX}	straightness error motion in Z-axis direction

Figure 2.7: "Angular and translational error motions of a component commanded to move along a (nominal) straight line trajectory parallel to the X-axis" [11]

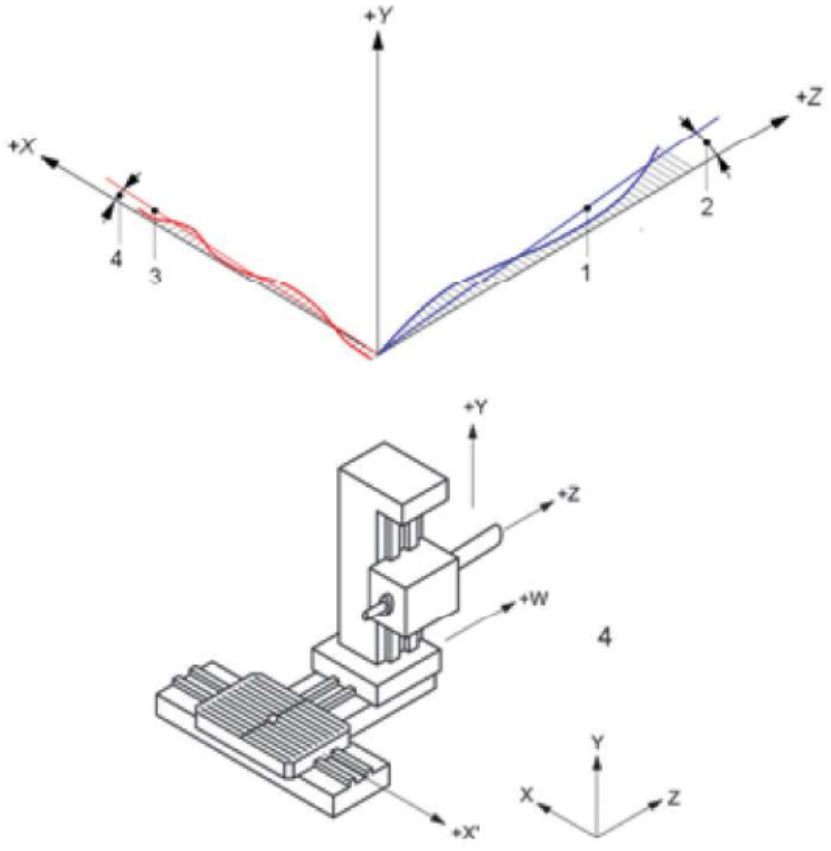
2.3.1 Squareness Errors Between Axes of Linear Motion

In ISO 230-1 [11], squareness error between two axes of linear motion is defined as "the difference between the inclination of the reference straight line of the trajectory of the functional point of a linear moving component with respect to its corresponding principal axis of linear motion and (in relation to) the inclination of the reference straight line of the trajectory of the functional point of another linear moving component with respect to its corresponding principal axis of linear motion".

In other words, the squareness error between two axes of linear motion is calculated by the slopes of the two straightness errors associated with the two axes. Taking the X- and Z-axes as an example: When the straightness errors E_{ZX} and E_{XZ} have been measured, two reference straight lines are fitted to the data using e.g. the least squares method. The angles between the reference straight lines and the nominal X- and Z-axis, $\theta_{\mathrm{X,ZX}}$ and $\theta_{\mathrm{Z,ZX}}$ (see figure 2.8), may then be calculated [12]. The squareness error between the X- and Z-axes may then be calculated as [11]:

$$E_{\mathrm{B(0Z)X}} = E_{\mathrm{B0X}} = \theta_{X,ZX} - \theta_{Z,ZX} \tag{2.39}$$

A positive squareness error means that the angle between the two axes considered is larger than 90°, and conversely a negative squareness error means that the angle is smaller than 90°. In a machine tool with three mutually orthogonal axes, there are three squareness errors[13].



Key

1 reference straight line associated to E_{XZ}
2 E_{XZ} reference straight line inclination; $\theta_{\mathrm{Z,ZX}}$ (positive value, as shown)
3 reference straight line associated to E_{ZX}
4 E_{ZX} reference straight line inclination; $\theta_{\mathrm{X,ZX}}$ (negative value, as shown)

Figure 2.8: Example of squareness error between X- and Z-axis of linear motion [11]

2.4 Geometric Errors of Rotational Axes

A machine tool usually has at least one axis of rotation. This may be a tool holding spindle unit (in the case of the machine tool being a milling machine), or a work holding spindle unit (lathe). A spindle unit consists of a spindle housing, or stator, a bearing and a spindle - the rotating element also called rotor. It is not uncommon for a lathe machine tool to have two work holding spindle units rotating about the same axis of rotation opposite of each other. This setup makes machining both ends of a workpiece in one operation possible. Milling machines typically just have one tool holding spindle unit, although production machines with a master and a slave tool holding spindle unit is not uncommon. Machining centers with both tool holding and work holding spindle units are also popular machines for prototyping and complex machining operations in general.

Aside from the two types of spindle units previously mentioned, machine tools may have one or more numerically controlled axes of rotation. These may be rotary (swivelling) tables or rotary (swivelling) heads. A rotary table usually holds a workpiece and has the capability to angularly position it in the workspace. A rotary head usually holds a tool, and can position it angularly in the workspace.

As for the linear axes, the geometrical errors associated with a rotary axis can be divided into "error motions of axis of rotation" and "position and orientation errors (axis shift) of axis average line". Figure 2.9 (left) shows the error motions of an axis of rotation and Figure 2.9 (right) shows the location and orientation errors of the axis average line.

As is the case with the linear axes, the rotational axes also have six individual errors. Referring to Figure 2.9 (left), these are the radial error motions in X- and Y-direction (denoted by E_{XC} and E_{YC}), the axial error motion (E_{ZC}), the two tilt error motions (E_{AC} and E_{BC}) and finally the angular positioning error motion (E_{CC}). The angular error motion E_{CC} is similar to the linear positioning error of the linear axes with regard to the origin of the error being affected by the numerical control of the axis [11].

In addition to the errors of the axis, there are four more errors regarding the location and orientation of the axis itself. These errors are denoted the same way as the other errors but with the addition of a zero between the two letters in the subscript. Referring to Figure 2.9 (right), E_{X0C} and E_{Y0C} are the errors of the position of C in X- and Y-axis directions respectively, and E_{A0C} and E_{B0C} are the error of the orientation of C in A- and B-axis directions respectively. The last two errors can also be considered as the squareness error of C to X and Y respectively [11].

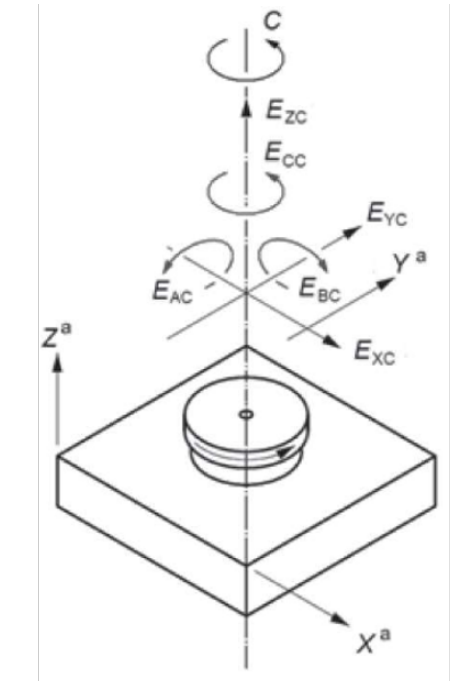
2.5 Kinematic Model

Error Representation

As explained in section 2.2.1, the position and location of a rigid body in three dimensional space can be described by a position vector p describing the position of a coordinate frame attached to the rigid body (the body frame $\{b\}$), and a 3×3-matrix R describing the orientation of the unit axes of the body frame. The orientation may be represented by a regular rotation matrix, on exponential coordinate form as a the skew-symmetric matrix $[\hat{\omega}]\theta$ or in axis-angle form as the skew-symmetric matrix representation of the unit rotation axis $[\hat{\omega}]$ multiplied by the rotation angle θ.

Infinitesimal Rotations

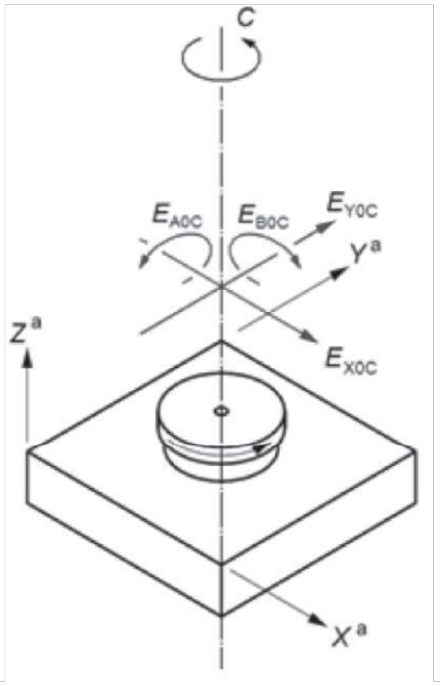
As previously mentioned, the set of all 3×3 skew-symmetric matrices is called so(3) or the lie algebra of the lie group SO(3). These matrices can be used to represent rotations on axis-angle or exponential coordinate form, but "they are not themselves rotations: the skew-symmetric matrices are derivatives" [14]. An infinitesimal rotation matrix has the form



Key

E_{XC} radial error motion of C in X-direction
E_{YC} radial error motion of C in Y-direction
E_{ZC} axial error motion of C
E_{AC} tilt error motion of C around X-axis
E_{BC} tilt error motion of C around Y-axis
E_{CC} angular positioning error motion of C; measured angular positioning deviations of C-axis

[a] Reference axis.



Key

E_{X0C} error of the position of C in X-axis direction
E_{Y0C} error of the position of C in Y-axis direction
E_{A0C} error of the orientation of C in the A-axis direction; squareness of C to Y
E_{B0C} error of the orientation of C in the A-axis direction; squareness of C to X

[a] Reference axis.

Figure 2.9: **Left**: Error motions of an axis of rotation. **Right**: Location and orientation errors of axis average line (adapted from [11])

$$I + Ad\theta, \tag{2.40}$$

where $d\theta$ is vanishingly small and $A \in so(n)$, for instance with $A = L_x$,

$$dL_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -d\theta \\ 0 & d\theta & 1 \end{bmatrix}. \tag{2.41}$$

An advantage to infinitesimal rotations is that the order in which they are applied is irrelevant, meaning that infinitesimal rotation matrices to the first order are commutative [14].

Quasi-static Error Model of Linear Axes

The angular and translational errors of a linear axis are described in the quasi-static error model of the axis. The error model was presented and used to measure and calculate the errors of all three linear axes in the specialization project in the fall of 2020 (see appendix B, section 2.5).

The error model is a homogeneous transformation matrix where the angular errors are represented on exponential coordinate form as a 3×3 skew-symmetric matrix, and the translational errors in X-, Y-, and Z-direction is represented as the fourth column vector. The following matrix equation is the quasi-static

error model of the X-axis, where the homogeneous transformation matrix (4×4) represents the axis motion including the angular and translational errors assuming small angular errors.

$$\begin{bmatrix} \Delta_x(x) \\ \Delta_y(x) \\ \Delta_z(x) \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & -E_{\mathrm{CX}}(x) & E_{\mathrm{BX}}(x) & E_{\mathrm{XX}}(x) + x \\ E_{\mathrm{CX}}(x) & 1 & -E_{\mathrm{AX}}(x) & E_{\mathrm{YX}}(x) \\ -E_{\mathrm{BX}}(x) & E_{\mathrm{AX}}(x) & 1 & E_{\mathrm{ZX}}(x) \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_x \\ y_x \\ z_x \\ 1 \end{bmatrix}$$

The matrix equation solves the deviation in X-, Y-, and Z-direction of an X-axis displacement compared to the ideal X-axis displacement with no errors. From the matrix equation, the following equations can be derived:

$$\begin{aligned} \Delta_x(x) &= -y_x E_{\mathrm{CX}}(x) + z_x E_{\mathrm{BX}}(x) + E_{\mathrm{XX}} \\ \Delta_y(x) &= x_x E_{\mathrm{CX}}(x) - z_x E_{\mathrm{AX}}(x) + E_{\mathrm{YX}} \\ \Delta_z(x) &= -x_x E_{\mathrm{BX}}(x) + y_x E_{\mathrm{AX}}(x) + E_{\mathrm{ZX}} \end{aligned}$$

Quasi-Static Error Model of Rotational Axes

The quasi-static error model of a rotational axis, here taking the C-axis as an example, can be described by the following transformation using homogeneous transformation matrices:

$$T_{\mathrm{CS}} = T_{\mathrm{OS}} T_{\mathrm{SS}} T_{\mathrm{ES}} T_{\mathrm{CC}}, \tag{2.42}$$

where T_{OS} is the position of the origin of the body coordinate frame $\{C\}$

$$T_{\mathrm{OS}} = \begin{bmatrix} 1 & 0 & 0 & O_{\mathrm{XC}} \\ 0 & 1 & 0 & O_{\mathrm{YC}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

T_{SS} is the squareness error of the C-axis

$$T_{\mathrm{SS}} = \begin{bmatrix} 1 & 0 & S_{\mathrm{YC}} & 0 \\ 0 & 1 & -S_{\mathrm{XC}} & 0 \\ -S_{\mathrm{YC}} & S_{\mathrm{XC}} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

T_{EC} is the translational and angular errors as a function of the C-axis position

$$T_{\mathrm{ES}} = \begin{bmatrix} 1 & -E_{\mathrm{CC}} & E_{\mathrm{BC}} & E_{\mathrm{XC}} \\ E_{\mathrm{CC}} & 1 & -E_{\mathrm{AC}} & E_{\mathrm{YC}} \\ -E_{\mathrm{BC}} & E_{\mathrm{AC}} & 1 & E_{\mathrm{ZC}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and T_{CC} is the nominal angular position of the C-axis [15].

$$T_{\mathrm{CS}} = \begin{bmatrix} cosC & -sinC & 0 & 0 \\ sinC & -cosC & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Neglecting the squareness errors for now, the volumetric error components as a function of the C-axis position can be described by the following matrix equation:

$$\begin{bmatrix} \Delta_x(C) \\ \Delta_y(C) \\ \Delta_z(C) \\ 0 \end{bmatrix} = \begin{bmatrix} -E_{\mathrm{CC}}(C)sin\theta & -E_{\mathrm{CC}}(C)cos\theta & E_{\mathrm{BC}}(C) & E_{\mathrm{XC}}(C) \\ E_{\mathrm{CC}}(C)cos\theta & -E_{\mathrm{CC}}(C)sin\theta & -E_{\mathrm{XC}}(C) & E_{\mathrm{YC}}(C) \\ -E_{\mathrm{BC}}(C)cos\theta + E_{\mathrm{AC}}(C)sin\theta & E_{\mathrm{BC}}(C)sin\theta + E_{\mathrm{AC}}(C)cos\theta & 0 & E_{\mathrm{ZC}}(C) \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_C \\ y_C \\ z_C \\ 1 \end{bmatrix},$$

which yields the following three equations [16]

$$\begin{aligned} \Delta_x(C) &= -E_{\mathrm{CC}}(C)(x_C sinC + y_C cosC) + E_{\mathrm{BC}}(C)z_C + E_{\mathrm{XC}}(C) \\ \Delta_y(C) &= -E_{\mathrm{CC}}(C)(x_C cosC - y_C sinC) - E_{\mathrm{AC}}(C)z_C + E_{\mathrm{YC}}(C) \\ \Delta_z(C) &= [-E_{\mathrm{BC}}(C)cosC + E_{\mathrm{AC}}(C)sinC]x_C + [E_{\mathrm{BC}}sinC + E_{\mathrm{AC}}(C)cosC]y_C + E_{\mathrm{ZC}}(C). \end{aligned}$$

2.6 Measuring Geometric Errors

The methods used for obtaining the errors of a machine tool may be divided into two main categories: Direct and indirect error measurement. In this thesis, the definitions of direct and indirect measurements follows that of Schwenke et. al. in [17]. Direct measurement means measuring each error individually, while indirect measurements measures the effects of several errors superimposed on each other. In the indirect error measurement method, the collected data must be treated extensively in order to extract the individual errors.

2.6.1 Direct Measurement

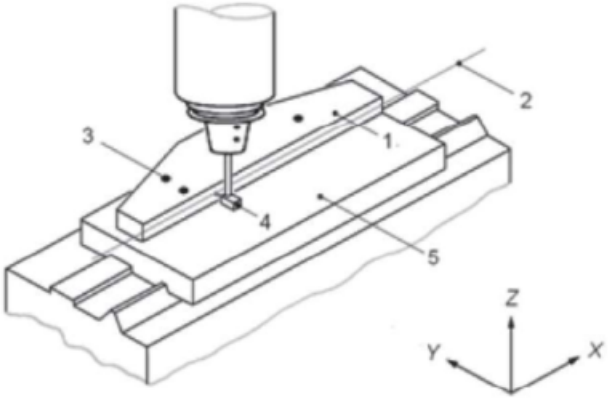
Schwenke et. al. [17] further divide the direct measurements into three groups based on the instruments used and their metrological reference. The first group contain the methods where artefacts such as straightedges, line scales or step gauges are used. These methods are covered by ISO 230-1 [11]. In 8.2.2.1, the measurement of straightness error motions using a straightedge is described. The straightedge is used as the straightness reference, and a linear displacement sensor is used to observe the relative deviation between the tool holding and the work holding side of the machine tool as it is moved along the travel of the concerned axis. The linear displacement sensor shall be placed as close to the functional point on the spindle as possible. The straightness errors of the straightedge should also be taken into account. If they are unknown, they can be identified by performing measurements in a reversed setup (see figure 2.10). In this setup, the straightedge and linear displacement sensor is turned 180° and the measurements repeated. In this way, it is possible to separate the straightness error of the straightedge from the straightness error of the linear axis. The following equations apply:

$$M(X) = \frac{[E_1(X) + E_2(X)]}{2} \tag{2.43}$$

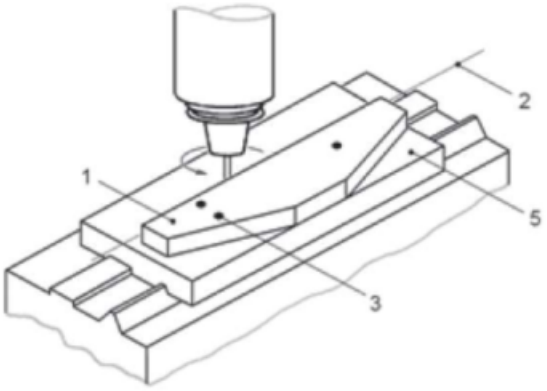
$$S(X) = \frac{[E_1(X) - E_2(X)]}{2} \tag{2.44}$$

where $M(X)$ is the straightness deviation of the reference surface of the straightedge at a given measurement position X, $S(X)$ is the straightness deviation of the axis of motion at a given measurement position X and $E_1(X)$ and $E_2(X)$ are the measurement data obtained from the normal and reversed setup respectively (see Figure 2.11)[11].

Other methods in this category are the ones using step gauges ore line scales. Next comes the methods using laser light's linear propagation and wavelength as reference. The third and final group measures errors in reference to earths gravity field. The direction of the gravity vector then becomes the reference to which geometric errors are measured [17].



a) Normal setup

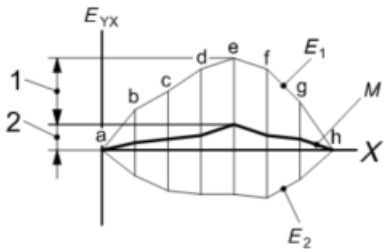


b) Reversed setup

Key

1	straightedge
2	measurement line
3	straightedge support points (3) both sides
4	linear displacement sensor
5	machine table

Figure 2.10: Measuring the straightness error using a straightedge [11]



Key

X	X-axis positions
E_{YX}	straightness deviations of X in Y-axis direction
1	straightness error of X in Y-axis direction $[S(X)_{max}]$
2	straightness error of the straightedge
a to h	measurement positions
E_1	plot of readings from normal setup
E_2	plot of readings from reverse setup
M	mean of E_1 and E_2

Figure 2.11: Separating the straightness errors of the straightedge and the axis motion [11]

2.6.2 Indirect Measurement

Indirect measurements usually means measuring deviations which are the result of several different individual errors superimposed on each other. This measuring method often requires multi-axis movement of the machine tool, and may involve the use of artefacts in different forms. A way of performing such an indirect measurement is to measure a machined test piece in a CMM. However, many factors contribute to the uncertainty of measurements made in this way. Clamping factors, tool wear, and other machining parameters are examples of such factors.

Contour measurements use simultaneous movement of two or more axes to produce either a straight line motion, or circular motions within the working volume of the machine tool. Deviations from the programmed paths are then measured and analyzed either by special equipment such as a double ball bar [18], or linear displacement probes [19]. The last example may be considered a whole separate method of error measurement. In this method, a probe in the form of a precision sphere fixed to the spindle of the machine tool, is moved together with the rotational axes in such a way that nominally, no relative movement between the two should occur. The deviation, the unintended relative movement between spindle and work table, is measured by three displacement sensors configured in a specific way (see Figure 2.12).

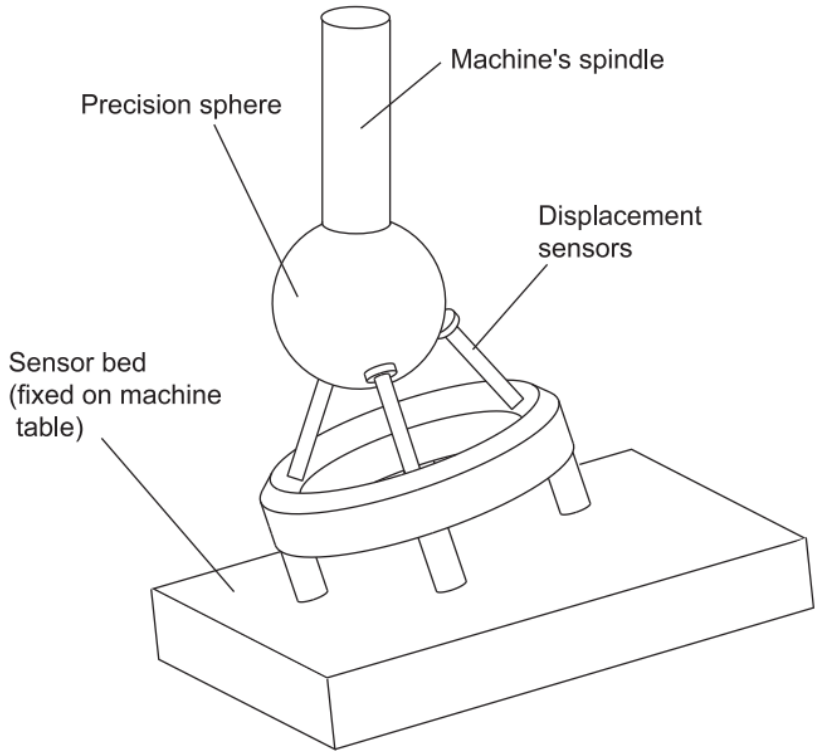


Figure 2.12: The setup of the probe (precision sphere) and the displacement sensors called R-test [19]

Equipment used to evaluate errors directly, may also be used in the indirect measuring methods. An example of this is the use of laser interferometry which traditionally is used to measure the errors of a single axis at a time to determine the errors of several axes simultaneously. By using the principle of multilateration, the length measurements obtained by an interferometer may be used to determine the three dimensional coordinates of points on the spindle/work table of a machine tool from which several errors may be calculated. However in practice, since the use of a conventional interferometer requires manual interaction in order to change the measurement direction, this method is quite unfeasible. As an alternative to this, a double ball bar possesses the ability to measure in every direction. However this measuring instrument is somewhat limited in its usable stroke. An instrument with the ability to provide

precise length measurements in every direction with a large range is the Lasertracer. This instrument automatically tracks and follows the target reflector and in this way enables the measurement of a large number of points in the machine working volume in a reasonable amount of time. The precision of the Lasertracer is higher compared to conventional laser trackers because the length measurement is made directly in reference to a stationary sphere. This helps to reduce the radial measurement uncertainty [17].

On machine tools such as parallel kinematic machines and hybrid machines, direct measurement of geometric errors may not be possible at all, or may only function for a few errors. In these cases indirect measurement methods are the only ones which can provide a complete picture of the geometric errors of the machine tool [17].

2.7 Measurement Uncertainty

Measurement uncertainty is related to the observation of the effects of other phenomena than the ones you intend to observe. This might be for instance the change of the refractive index of air which leads to a change in a laser distance measurement, or the thermal expansion of a gauge block measured using a micrometer when the goal of the measurement is to determine an estimate of the true value of the length dimension of the gauge block.

When the result of a measurement of some physical quantity is given, there should always follow a number indicating how precise the measurement result is. This is of great importance to those who depend on, use or in other ways are interested in the results of measurements. If no such indication is given, comparing different measurement results is difficult. An easy to understand, commonly accepted procedure for evaluating the quality of measurements is therefore necessary for determining and expressing the measurement's uncertainty [20].

Uncertainty, which means "doubt", is in the context of measurement a sort of doubt as to how accurately the result of a measurement represents the true value of the measurand (the quantity subject to measurement). Uncertainty in measurement is defined as "parameter, associated with the result of a measurement, that characterizes the dispersion of the values that could reasonably be attributed to the measurand" [20].

In assessment of the uncertainty in measurement the approaches generally fit into either what is known as Type A evaluation, or Type B. Type A evaluation of uncertainty is defined as the "method of evaluation of uncertainty by the statistical analysis of series of observations", and Type B the "method of evaluation of uncertainty by means other than the statistical analysis of series of observations" [20].

Based on repeated measurements or observations, the expected value is often best estimated by the arithmetic mean or average of the observations

$$\bar{q} = \frac{1}{n}\sum_{k=1}^{n} q_k. \tag{2.45}$$

The observed values of the measurand will vary because of random effects or random variations in the quantities affecting the value of the measurand. Two parameters often used to estimate the variance of the probability distribution and describe the dispersion of the observed values about their mean are the experimental variance of the observations calculated as

$$s^2(q_k) = \frac{1}{n-1}\sum_{j=1}^{n}(q_j - \bar{q})^2, \tag{2.46}$$

and its positive square root - the experimental standard deviation

$$s(q_k) = \sqrt{s^2(q_k)} = \sqrt{\frac{1}{n-1}\sum_{j=1}^{n}(q_j - \bar{q})^2}. \tag{2.47}$$

The best estimate of the the variance of the mean is given by

$$s^2(\bar{q}) = \frac{s^2(q_k)}{n}, \tag{2.48}$$

and the estimated standard deviation of the mean is given as the positive square root of the estimated variance of the mean

$$s(\bar{q}) = \sqrt{s^2(\bar{q})} = \sqrt{\frac{s^2(q_k)}{n}} \tag{2.49}$$

2.8 Laser Tracker

The instrument used for obtaining the measurements used in this work is the Leica Absolute Tracker AT960 MR. For details on the working principles of laser trackers I refer to section 2.6 of the specialization project (see appendix B).

Sources of Uncertainty in Laser Tracker Measurement

Huo & Cheng [21] classifies the uncertainty errors of laser tracker measurement into four categories:

1. Static or quasi-static uncertainty sources
2. Dynamic uncertainty sources
3. Fitting and evaluation algorithm related uncertainty sources
4. Measuring strategy/sequence related uncertainty sources

In their article, they provide the following figure as an overview of the sources of uncertainty associated with a general laser tracker measurement:

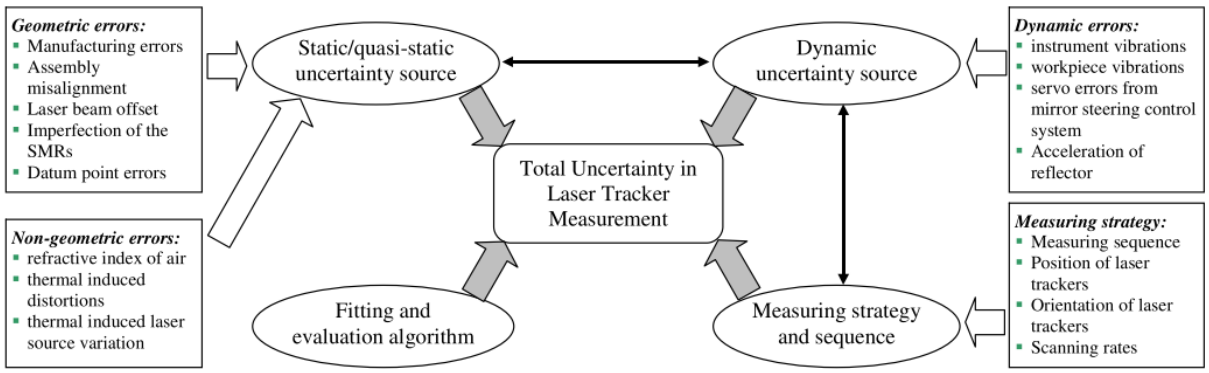


Figure 2.13: Overview of sources of uncertainty in laser tracker measurement [21]

As seen in the figure, the static/quasi-static sources of uncertainty can further be divided into geometric errors and non-geometric errors. The geometric sources of uncertainty include inaccuracies in the rotational axes of the tracker, mirror center offset, imperfections of the spherically mounted retroreflector (SMR) and datum point errors. These are the most significant sources of uncertainty, and are subject to

error compensation by the manufacturers who all have their own error correction algorithms and calibration processes. Non-geometric sources of uncertainty are those related to variations in the wavelength of the laser light and thermally induced distortions of the laser light source and other mechanical and optical components in the tracker structure. Variations in the refractive index of the medium the laser light travels in influences the wavelength of the light and thereby the calculation of the measured distance. The sensitivity of a laser distance measurement to environmental factors influencing the refractive index of air is summarized in Table 2.1.

Table 2.1: Sensitivity of laser distance measurements to environmental parameters [17]

Condition	Uncertainty	Resulting uncertainty
Air temperature	1° C	1 μm/m
Air Pressure	1 hPa	0.3 μm/m
Humidity	10 % RH	0.1 μm/m
CO_2 content	100 ppm	14 nm/m

The dynamic source of uncertainty stems from dynamic errors such as instrument- or workpiece vibrations, servo errors in the mirror steering control system and acceleration of the reflector. The latter is an issue when acquiring measurements in scanning mode (high continuous sampling rate). While these error types are present in a laser tracker measurement system, some of them are considered negligible in magnitude and difficult to evaluate. The uncertainty of laser tracker measurements are highly non-uniform in space, which gives rise to uncertainty related to the measurement strategy and the sequence in which points are measured. The last source of uncertainty is that of the fitting and evaluation algorithm used on the discrete points measured.

2.8.1 Hexagon Inspire measuring software

Inspire is a versatile measuring software made for any portable measuring arm or laser tracker for probing and scanning. It features tools for constructing and fitting geometry to measured points, and evaluation of measured points and features according to the common standards on geometric dimensioning and tolerancing. Inspire also offers automation functionality for repetitive measurement operations [22].

2.9 Numerical Compensation

In this thesis, numerical compensation is used as the name of the process of compensating for geometric errors on numerically controlled machine tools. As stated in ISO/TR 16907 [3], by numerically compensating for the geometric errors of a machine tool, the accuracy of parts produced on the machine may be increased, the cost of manufacturing and assembling machines may be decreased and the maintenance cost during the life cycle of the machine may be lowered.

One can say there are two stages to numerically compensate the geometric errors in a machine tool. One is to compensate the spatial coordinate of the *functional point* to eliminate translational errors. The functional point of a machine tool is defined as "the cutting tool centre point or point associated with a component on the machine tool where the cutting tool would contact the part for the purposes of material removal". The other is to compensate the *functional orientation*. The functional orientation is the "relative orientation between the component of the machine tool that carries the cutting tool and the component of the machine tool that carries the workpiece".

A volumetric error model is a model describing the resulting errors in both the machine tool's functional point and functional orientation. These errors may arise from individual error motions in addition to orientation errors of the machine tool axes. This model also includes axis-positions and other *structural loop* variables like tool lengths and tool offsets. The structural loop is defined as the "assembly of

components, which maintains the relative position between two specified objects". An example of this would be everything connecting the workpiece and the cutting tool in a milling machine such as spindle bearings, slideways, machine frame and work table. Based on the volumetric error model, a volumetric compensation can be executed. Just the functional point of the machine can be compensated, making it a voulumetric compensation of the functional. The compensation can include the functional orientation as well, which is called a volumetric compensation of functional point and functional orientation. An interesting thing to note is that since the compensations are done at the functional point, i.e. at the tip of the cutting tool, in the case that a spherical cutting tool is used, such as a ball nose endmill, a compensation of the functional point actually represents a full volumetric compensation for that particular case. The reason for this being that errors in functional orientation does not affect the geometry of the workpiece when a spherical cutting tool is used.

The volumetric error model may be a kinematic error model, or a spatial error grid. A kinematic error model is a mathematical error model describing the structural loop of a machine tool as a kinematic chain and describes the errors that are taken into account. A kinematic error model may be represented in a number of different ways such as e.g. homogeneous transformation matrices. A kinematic error model may include errors from elastic deformation as a result of loads from acceleration/deceleration or other mechanical loads such as heavy tools, heavy workpieces etc. called quasi-rigid body behavior. A specific type of kinematic error model is the rigid body kinematic error model. In this model, all the elements of the structural loop is considered as entirely rigid. This means that the effects of external loads are neglected. Also, the the errors of one axis is assumed independent from the position of other axes. Compensation for the errors represented in the rigid body kinematic error model is called a rigid body kinematic compensation. A statement on which errors are included in the error model is recommended.

Another way of representing the volumetric error model is in the form of an error table. An error table is a table containing the values of each of the errors associated with an axis at several different linear or angular command positions in each axis. For a linear axis, the error table usually contains both the translational error motions (positioning and straightness errors) and the angular error motions (roll, pitch and yaw). Similarly for the rotational axes, both the translational error motions (axial and radial error) in addition to the angular error motions (tilt error and angular positioning errors). Reversing the sign of the errors in the error table results in a compensation table - the amount by which the motion must be corrected in order to obtain a geometrically correct movement.

A spatial error grid is a three dimensional representation of the errors of the machine tool. While the error *table* represents the error in a single axis, the error *grid* represents the resulting error in position and orientation from the errors of all the axes influencing the position of the functional point at each sampling point. The values in the error grid are the superposition of the effects of geometric errors in multiple axes. Just as reversing the sign of the error table results in the compensation table, reversing the sign of the error grid results in the compensation grid.

In the error grid as well as in the error table or any other representation of the geometric errors in a machine tool, the errors are reported in a certain set of sampling points, a certain set of positions of the linear and rotational axes. This means the errors are represented in a discrete way, with error values "jumping" from one value to another. To completely represent the errors of the entire working volume of a machine tool, an infinite number of sampling points would have to be evaluated. This is obviously not feasible. Therefore, for points in the working volume not equal to the sampling points, geometric error values are interpolated from the neighbouring sampling points to estimate the error value to a sufficient degree of accuracy [3].

2.9.1 Types of Geometric Compensation

Error compensation can be divided into two different types depending on the repeatability of the machine tool - 'pre-calibrated error compensation' and 'active error compensation' In 'pre-calibrated error compensation', the errors are measured on one occasion an then used for later compensation of machine motion. This presupposes that the process of measuring and the errors themselves are highly repeatable. 'active error compensation' on the other hand continuously evaluates the errors while the machining process takes place. This method does not require the same repeatability as the pre-calibrated one [23].

ISO/TR 16907 [3] defines and standardises different forms of numerical compensation in machine tools. The following part aims to summarize these types of compensations.

Compensation for Positioning Errors of Linear Axes Along Specific Lines, L-POS

This compensation method only compensates for the positioning error along specific lines in the working volume. It will not reduce the straightness error of axes, and it will not compensate for angular errors except on those specific lines. Taking the X-axis of a machine tool as an example: The linear positioning error E_{XX} is compensated and brought as close to the nominal position as possible. This includes compensating for the effect of angular errors in the axis, but not compensating the angular error itself. The straightness errors, E_{YX} and E_{ZX} are left uncompensated. The compensation (and measurement) is typically performed in the centre of travel of the other axes or the most frequently used machine tool volume.

Compensation for Straightness errors of Linear Axes Along Specific Lines, L-STR

The straightness errors are compensated along specific lines in the working volume. With the application of this compensation method alone, the positioning error of axes will not be compensated, and angular errors will affect the straightness of lines parallel to the compensated line.

Compensation for Squareness Error Between Axes of Linear Motion at Specific Lines, L-SQU

Errors in squareness between linear axes are compensated, but only along specific lines in the working volume. Angular error motion of the axes will affect squareness error at other measurement lines, and the positioning error of axes are not compensated.

Compensation for the Angular Error Motions of Linear Axes on 3-D Position of Functional Point in the Working Volume, L-ANG

The angular error motion of the machine tool is compensated by adjusting the three-dimensional position of the functional point in the working volume. The remaining errors will be entirely a result of positioning and straightness errors. Errors in the functional orientation will remain unchanged, and machining with large diameter cutting tools will reveal these angular errors. Cutting operations using tools with spherical tips can be fully compensated using this method.

Physical Compensation for Errors in Functional Orientation, FOR

Complete compensation of errors in the functional orientation of a machine tool require three numerically controlled orthogonal rotary axes where a minimum of two of these can not be parallel to the spindle axis. The functional orientation errors arising from errors in both the rotary axes and the linear axes are compensated using this method. This method should be implemented in combination with the L-ANG

ang R-ANG methods. If not, unintended X-, Y- and Z-movements should be compensated by other means.

Volumetric Compensation of Linear Axes, L-VOL

This method has the potential to completely compensate for the effects of all geometric errors in the linear axes of a machine tool. The method combines the L-POS, L-STR, L-SQU and L-ANG compensations. Since the rotational axes are not used to compensate for angular error in the linear axes, direct measurement of angular errors will still show the uncompensated error motions. Cutting with tools having a large contact area will also reveal the effects of the angular errors. Spherical tool cutting will be completely compensated for all geometric errors of the linear axes.

Volumetric Compensation of Linear Axes Including Functional Orientation, L-VOL+

This compensation is a combination of L-VOL and FOR, meaning that the requirement regarding the rotary axes remains the same.

Compensation for Positioning Errors of Rotary Axes, R-POS

Similarly as for the linear axes, this compensation reduces the rotational positioning errors of the rotary axes.

Compensation for Position and Orientation Errors of Rotary Axes, R-RAX

Compensation for radial errors in a rotary axis at a specific elevation above the table surface, or at a specific tool length in the case of a rotating tool. At other elevations or tool lengths, tilt errors will affect the radial error of the axis.

Compensation for Position and Orientation Errors of Rotary Axes, R-POR

Both positioning and orientation errors of the rotary axes are compensated in this method, not to be confused with the angular positioning errors as covered by the R-POS method. An axis average line of the rotation is typically defined.

Compensation for the Tilt Error Motions of Rotary Axes on 3-D Position of Functional Point in the Working Volume, R-ANG

This compensation method reduces the effects of tilt error motions in the rotary axes on the functional point. Functional orientation, however, remains uncompensated. As for L-ANG, cutting with spherical tools will be completely compensated for the effects of angular error motions in the rotary axes, while cutting with large diameter cutting tools will still be affected by the angular error motions.

Volumetric Compensation for Rotary Axis Errors, R-VOL

R-VOL is a combination of several compensations - R-POS, R-RAX, R-POR and R-ANG. Functional orientation remains uncompensated.

Volumetric Compensation for Rotary Axis Errors Including Functional Orientation, R-VOL+

The same combination of compensations as R-VOL with the addition of FOR - compensation of functional orientation.

In addition to these standard compensations, machine tool manufacturers may define their own specific error compensation strategies for the linear and rotary axes termed L-SPEC and R-SPEC in ISO 16907 [3].

2.9.2 Where is Numerical Compensation Applied?

Numerical compensation may be applied on-line or off-line, meaning that the compensation may be applied directly on the machine controller, or off-line as alterations to the NC-data fed to the machine controller (Figure 2.14). Some machines have compensations built-in, such as backlash compensation, which means that compensation also can be applied both on- and off-line in combination.

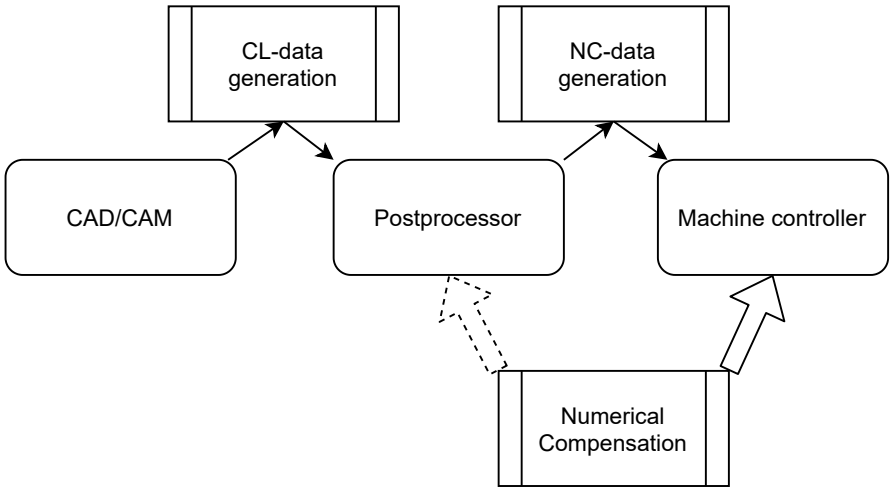


Figure 2.14: Numerical compensation may be applied either off-line, on-line or as a combination of both

2.9.3 Other Influences on the Application of Numerical Compensation of Geometric Errors

Temperature

Temperature may affect both the determination of the geometric errors along with the geometric performance of the machine. Testing the influence of thermal effects on the machine tool according to ISO 230-3 prior to determining geometric errors, compensation and validation would be good practice. The machine tool's stability with regard to temperature variation and the errors this brings will set limitations as to what precision is attainable through other compensations.

Repeatablility

Several things affect the repeatability of the machine tool. Mechanical play in machine parts can cause reversal error in machine axes and thereby a reduced repeatability. Friction and wear, thermal effects of machine tool components and plastic deformation of components or foundations over time are other examples of things that may affect repeatability.

This also sets an upper limit to what precision is achievable through compensation. The effects of some short-term repeatability issues can be reduced to a certain degree by averaging at least over measurements in different directions if geometric errors are taken as an example.

Machine Tool Least Increment Step

The least increment the machine axes can be moved will directly affect the least amount of compensation that can be applied.

Workpiece- and Tool Mass

Typically, the geometric accuracy of machine tools are evaluated in quasi-static no-load conditions. In some cases though, it might be more appropriate to evaluate errors including the mass of the workpiece or the tool.

Chapter 3

Method

3.1 Repeating the Measurements of the Linear Axes

In the specialization project report written in the fall of 2020 (see appendix B), the geometric errors of the linear axes was measured once. In this work, a new configuration of the points on the spindle is chosen and measured four times in order to evaluate the variability of the results (see Figure 3.1).

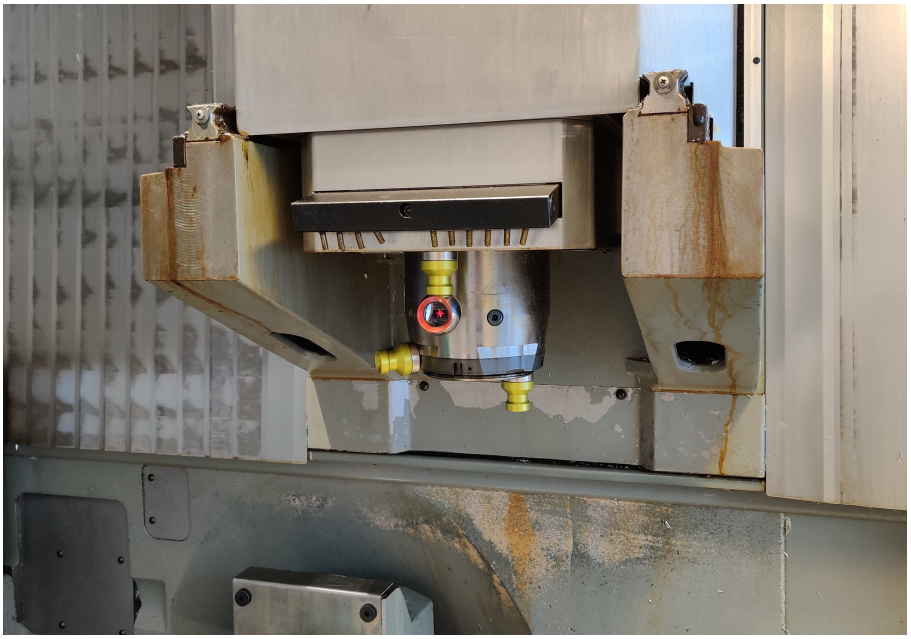


Figure 3.1: Configuration of the measured points on the spindle

The measuring procedure followed is essentially the same as the one in the specialization project. The position of the three points fixed to the machine spindle assembly is acquired by sequentially measuring the position of the Red Ring Reflector (RRR) on each of the three positions on the spindle. Sequentially measuring means that the movement of the axis is made with the RRR fixed to a single magnet holder, and the position measured at each increment along the axis movement. The machine axis is then reset, the RRR moved to the next magnet holder, and the process repeated. The measurements are acquired using the stable points mode in Inspire. This means that the software detects when the machine is stationary, and then executes the measurement. This procedure is carried out four times per linear axis of the machine (see Figure 3.2).

After acquiring the measurements of the points along each linear axis, the origin and axes of the fixed

coordinate frame of the machine must be established. Firstly, the magnet holder fixed to the spindle is used to measure points along the rotational motion of the spindle. The spindle is unlocked and manually indexed during this measuring process. From the measured points along the spindle rotation motion, a circle is constructed and fitted by the least squares method in Inspire. A line representing the direction of the Z-axis is made, and another line is constructed to represent the direction of the X-axis. The measurement data of one of the points on the spindle is used as input for these lines. The coordinate frame is aligned to the X-axis line, the Z-axis line and the constructed circle feature. The constructed features and the coordinate frame alignment can be seen in Figure 3.3.

The raw measurement data of every point is then exported to a csv-file, before it is used to calculate the errors using the same approach as in the specialization project. The magnitude of each error is averaged over the four rounds of measurements made, and the standard deviation of the error in each measured point is calculated. The Python scripts used to calculate the errors of the linear axes from the raw measurement data can be read in Appendix C.1 and C.2.

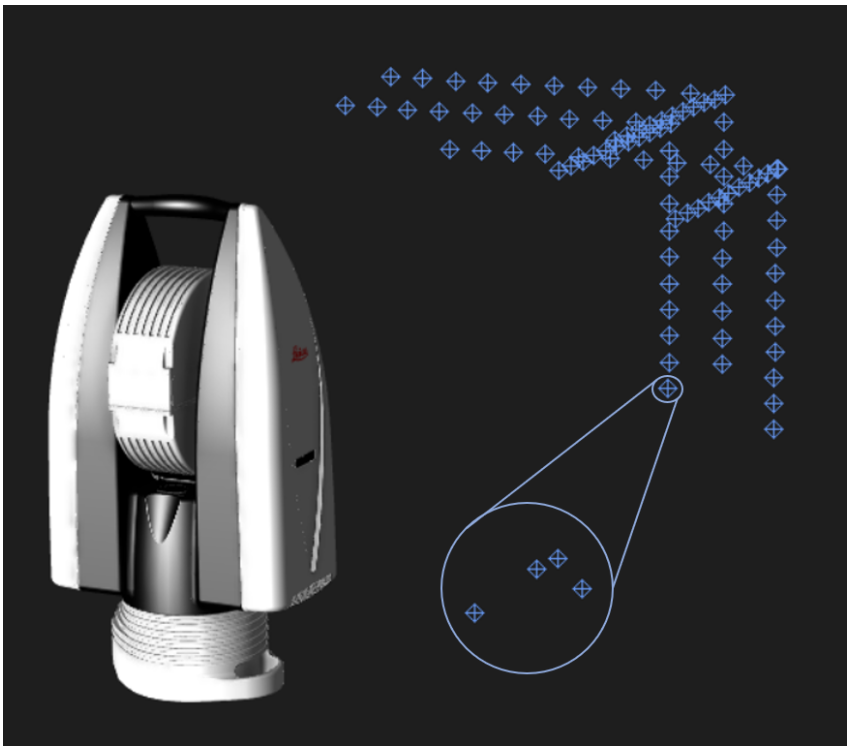


Figure 3.2: Acquisition of the four rounds of measurements for each of the linear axes of the DMU 50

3.2 Determining the Errors of the Rotational Axes

Three non-colinear points are chosen on the machine table as shown in Figure 3.4. The setup also ensures that the height of the measured point relative to the table surface is different for each point when the axes are in the zero-position. The three points, K, P, Q have coordinates (x_{CK}, y_{CK}, z_{CK}), (x_{CP}, y_{CP}, z_{CP}) and (x_{CQ}, y_{CQ}, z_{CQ}). oxyz is the reference coordinate frame, and $o_C x_C y_C z_C$ is the C-axis coordinate frame. The homogeneous coordinates of o_C is $\begin{bmatrix} x_o & y_o & z_o & 1 \end{bmatrix}^T$ in the reference coordinate frame before the axes are moved. If there are no errors in the motion of the rotational axes, the homogeneous coordinates

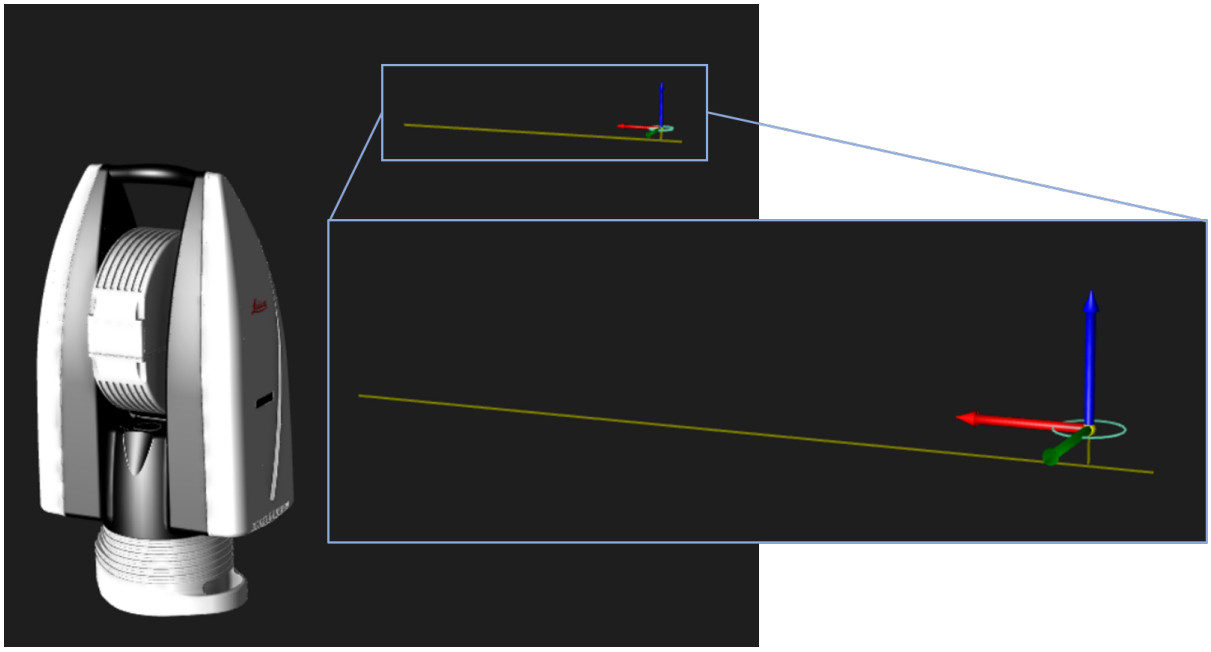


Figure 3.3: Constructed features and alignment of the coordinate frame in Inspire

of o_c at time t will be

$$\begin{bmatrix} x_{ot} \\ y_{ot} \\ z_{ot} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_o \\ y_o \\ z_o \\ 1 \end{bmatrix} = \begin{bmatrix} x_o \\ y_o \\ z_o \\ 1 \end{bmatrix} \tag{3.1}$$

and the homogeneous coordinates of point P are

$$\begin{bmatrix} x_{Pt} \\ y_{Pt} \\ z_{Pt} \\ 1 \end{bmatrix} = \begin{bmatrix} cosC & -sinC & 0 & x_{ot} \\ sinC & cosC & 0 & y_{ot} \\ 0 & 0 & 1 & z_{ot} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{CP} \\ y_{CP} \\ z_{CP} \\ 1 \end{bmatrix} \tag{3.2}$$

The coordinates of point P $(x'_{Pt}, y'_{Pt}, z'_{Pt})$ are then measured using the laser tracker. The three volumetric error components at the point P is then

$$\begin{bmatrix} \Delta_{Pxt} \\ \Delta_{Pyt} \\ \Delta_{Pzt} \\ 0 \end{bmatrix} = \begin{bmatrix} x'_{Pt} \\ y'_{Pt} \\ z'_{Pt} \\ 1 \end{bmatrix} - \begin{bmatrix} x_{Pt} \\ y_{Pt} \\ z_{Pt} \\ 1 \end{bmatrix} \tag{3.3}$$

which can be substituted into the error model from chapter 2 to obtain the following three equations

$$\begin{aligned} x'_{Pt} - x_{CP}cosC + y_{CP}sinC - x_o &= -E_{\mathrm{CCt}}(C)(x_{CP}sinC + y_CPcosC) + E_{\mathrm{BCt}}(C)z_{CP} + E_{\mathrm{XCt}}(C) \\ y'_{Pt} - x_{CP}sinC - y_{CP}cosC - y_o &= E_{\mathrm{CCt}}(C)(x_{CP}cosC - y_{CP}sinC) - E_{\mathrm{ACt}}(C)z_{CP} + E_{\mathrm{YCt}} \\ z'_{Pt} - z_{CP} - z_o &= [E_{\mathrm{BCt}}(C)cosC + E_{\mathrm{ACt}}(C)]x_{CP} + [E_{\mathrm{BCt}}(C)sinC + E_{\mathrm{ACt}}(C)cosC]y_{CP} + E_{\mathrm{ZC}}(C). \end{aligned} \tag{3.4}$$

Performing the same procedure on points K and Q yields the following set of equations written on matrix form

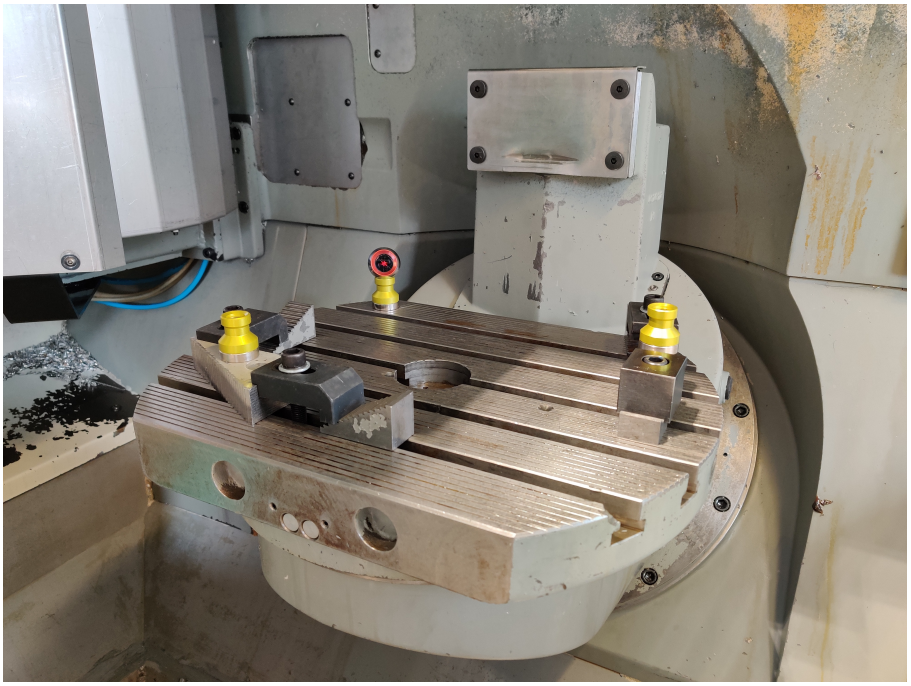


Figure 3.4: The setup for measuring the errors of the B- and C-axis of the Deckel Maho DMU 50 eVolution.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & z_{CP} & -\hat{y}_{CP} \\ 0 & 1 & 0 & -z_{CP} & 0 & \hat{x}_{xP} \\ 0 & 0 & 1 & \hat{y}_{CP} & -\hat{x}_{CP} & 0 \\ 1 & 0 & 0 & 0 & z_{CQ} & -\hat{y}_{xQ} \\ 0 & 1 & 0 & -z_{CQ} & 0 & \hat{x}_{CQ} \\ 0 & 0 & 1 & \hat{y}_{CQ} & -\hat{x}_{xQ} & 0 \\ 1 & 0 & 0 & 0 & z_{CK} & -\hat{y}_{CK} \\ 0 & 1 & 0 & -z_{CK} & 0 & \hat{x}_{CK} \\ 0 & 0 & 1 & \hat{y}_{xK} & -\hat{x}_{xK} & 0 \end{bmatrix} \begin{bmatrix} E_{\mathrm{XC}t}(C) \\ E_{\mathrm{YC}t}(C) \\ E_{\mathrm{ZC}t}(C) \\ E_{\mathrm{AC}t}(C) \\ E_{\mathrm{BC}t}(C) \\ E_{\mathrm{CC}t}(C) \end{bmatrix} = \begin{bmatrix} \Delta_{Kxt} \\ \Delta_{Kyt} \\ \Delta_{Kzt} \\ \Delta_{Pxt} \\ \Delta_{Pyt} \\ \Delta_{Pzt} \\ \Delta_{Qxt} \\ \Delta_{Qyt} \\ \Delta_{Qzt} \end{bmatrix} \tag{3.5}$$

where Δ_{ijt} (i = K, P or Q, j = x, y or z) denotes the volumetric error component at point i, in the direction of the j-axis at the tth step. $\Delta_{\mathrm{ixt}} = \mathrm{x}'_{\mathrm{it}} - \hat{\mathrm{x}}_{\mathrm{Ci}} - \mathrm{x}_{\mathrm{o}}$, $\Delta_{\mathrm{iyt}} = \mathrm{y}'_{\mathrm{it}} - \hat{\mathrm{y}}_{\mathrm{Ci}} - \mathrm{y}_{\mathrm{o}}$ and $\Delta_{\mathrm{izt}} = \mathrm{z}'_{\mathrm{it}} - \hat{\mathrm{z}}_{\mathrm{Ci}} - \mathrm{z}_{\mathrm{o}}$, where $\hat{\mathrm{x}}_{\mathrm{Ci}} = \mathrm{x}_{\mathrm{Ci}}\mathrm{cosC} - \mathrm{y}_{\mathrm{Ci}}\mathrm{sinC}$ and $\hat{\mathrm{y}}_{\mathrm{Ci}} = \mathrm{y}_{\mathrm{Ci}}\mathrm{cosC} + \mathrm{x}_{\mathrm{Ci}}\mathrm{sinC}$, (i = K, P or Q) [16].

Six equations can be extracted from this overdetermined system of equations to form a new system:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & z_{CP} & -\hat{y}_{CP} \\ 0 & 1 & 0 & -z_{CP} & 0 & \hat{x}_{xP} \\ 1 & 0 & 0 & 0 & z_{CQ} & -\hat{y}_{xQ} \\ 0 & 0 & 1 & \hat{y}_{CQ} & -\hat{x}_{xQ} & 0 \\ 0 & 1 & 0 & -z_{CK} & 0 & \hat{x}_{CK} \\ 0 & 0 & 1 & \hat{y}_{xK} & -\hat{x}_{xK} & 0 \end{bmatrix} \begin{bmatrix} E_{\mathrm{XC}t}(C) \\ E_{\mathrm{YC}t}(C) \\ E_{\mathrm{ZC}t}(C) \\ E_{\mathrm{AC}t}(C) \\ E_{\mathrm{BC}t}(C) \\ E_{\mathrm{CC}t}(C) \end{bmatrix} = \begin{bmatrix} \Delta_{Pxt} \\ \Delta_{Pyt} \\ \Delta_{Qxt} \\ \Delta_{Qzt} \\ \Delta_{Kyt} \\ \Delta_{Kzt} \end{bmatrix} \tag{3.6}$$

Care has to be taken to select the correct equations. Zhenjiu Zhang and Hong Hu explain this in detail in their articles on geometric error identification in machine tools [24][16].

Solving this system of equations yields the following expressions for the errors of the C-axis [16]:

$$
\begin{aligned}
E_{\mathrm{XC}}(C) &= \Delta_{Pxt} + \frac{M}{N}\left[\hat{y}_{CP} - \frac{z_{CP}(\hat{y}_{CQ} - \hat{y}_{CP})}{z_{CQ} - z_{CP}}\right] - \frac{z_{CP}(\Delta_{Qxt} - \Delta_{Pxt})}{z_{CQ} - z_{CP}} \\
E_{\mathrm{YC}}(C) &= \Delta_{Pyt} - \frac{M}{N}\left[\hat{x}_{CP} - \frac{z_{CP}(\hat{x}_{CP} - \hat{x}_{CK})}{z_{CP} - z_{CK}}\right] - \frac{z_{CP}(\Delta_{Pyt} - \Delta_{Kyt})}{z_{CP} - z_{CK}} \\
E_{\mathrm{ZC}}(C) &= \Delta_{Qzt} + \frac{M}{N}\left[\frac{\hat{y}_{CQ}(\hat{x}_{CK} - \hat{x}_{CP})}{z_{CP} - z_{CK}} - \frac{\hat{x}_{CQ}(\hat{y}_{CP} - \hat{y}_{CQ})}{z_{CQ} - z_{CP}}\right] - \frac{\hat{y}_{CQ}(\Delta_{Kyt} - \Delta_{Pyt})}{z_{CP} - z_{CK}} + \frac{\hat{x}_{CQ}(\Delta_{Qxt} - \Delta_{Pxt})}{z_{CQ} - z_{CP}}
\end{aligned} \tag{3.7}
$$

$$
\begin{aligned}
E_{\mathrm{AC}}(C) &= \frac{\Delta_{Kyt} - \Delta_{Pyt}}{z_{CP} - z_{CK}} + \frac{\hat{x}_{CP} - \hat{x}_{CK}}{z_{CP} - z_{CK}} \cdot \frac{M}{N} \\
E_{\mathrm{BC}}(C) &= \frac{\Delta_{Qxt} - \Delta_{Pxt}}{z_{CQ} - z_{CP}} + \frac{\hat{y}_{CQ} - \hat{y}_{CP}}{z_{CQ} - z_{CP}} \cdot \frac{M}{N} \\
E_{\mathrm{CC}}(C) &= \frac{M}{N}
\end{aligned} \tag{3.8}
$$

where

$$
\begin{aligned}
M &= (\Delta_{Qxt} - \Delta_{Pxt})(\hat{x}_{CK} - \hat{x}_{CQ})(z_{CP} - z_{CK}) + [(\Delta_{Pyt} - \Delta_{Kyt})(\hat{y}_{CK} - \hat{y}_{CQ}) + (\Delta_{Kzt} - \Delta_{Qzt})(z_{CP} - \\
&\quad z_{CK})](z_{CQ} - z_{CP}) \\
N &= (\hat{x}_{CK} - \hat{x}_{CQ})(\hat{y}_{CP} - \hat{y}_{CQ})(z_{CP} - z_{CK})(\hat{x}_{CK} - \hat{x}_{CP})(\hat{y}_{CK} - \hat{y}_{CQ})(z_{CQ} - z_{CP})
\end{aligned}
$$

The position of the three points located on the machine table are measured sequentially four times (see Figure 3.5(right)). An increment of 20 degrees was chosen for the C-axis, and 10 degrees for the B-axis. A new alignment of the coordinate frame was made for each of the two rounds of measurements. A circle was constructed for each of the two rotational axes with the measured positions of one of the points located on the machine table as input. A line representing the direction of the Z-axis perpendicular to the machine table when C = 0° is constructed. The line representing the direction of the X-axis is constructed the same way as for the linear axes, but the line is projected onto the plane spanned out by the circle features for each rotational axis (see Figure 3.5(left)). The Python scripts used to calculate the errors of the rotational axes from the raw measurement data can be read in Appendix D.1 and D.2.

3.3 Volumetric Error Model

Based on the 21 measured and modelled geometric errors of the machine tool, a model containing all error motions of the machine tool may be made [12]. By joining the error models of each of the linear axes, the volumetric error in the working volume may be calculated. The error model of a linear axis can be represented by a Homogeneous Transformation Matrix (HTM). With the assumption of small angular errors, the error model of the X-axis takes on the form:

$$
\begin{bmatrix}
1 & -E_{\mathrm{CX}}(x) & E_{\mathrm{BX}}(x) & E_{\mathrm{XX}}(x) + x \\
E_{\mathrm{CX}}(x) & 1 & -E_{\mathrm{AX}}(x) & E_{\mathrm{YX}}(x) \\
-E_{\mathrm{BX}}(x) & E_{\mathrm{AX}}(x) & 1 & E_{\mathrm{ZX}}(x) \\
0 & 0 & 0 & 1
\end{bmatrix} \tag{3.9}
$$

Although this error model provides a complete description of the error in tool center position, a more common way of representing straightness error is by extracting the inclination of the straightness errors of each individual axis, and then synthesize a squareness error.

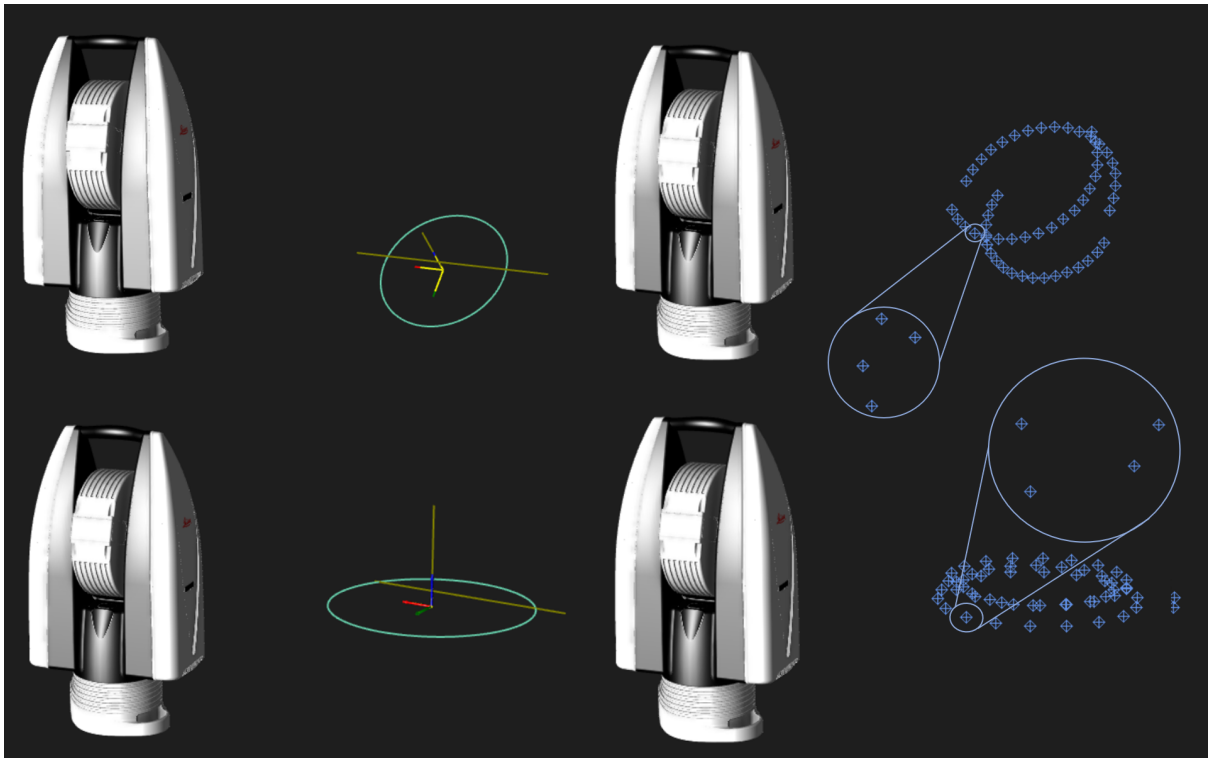


Figure 3.5: **Left:** Constructed features and alignment of the coordinate frame in Inspire. **Right:** The measured points for the B-axis (top) and C-axis (bottom).

3.3.1 Squareness Errors of Translational Axes

As a linear axis moves, straightness errors affect the movement of the functional point on the moving component traveling along the axis. A squareness error describes the deviation of a reference straight line fitted to the trajectory of the functional point as the axis moves, to the nominal direction of the axis.

For modeling of translation using HTMs, a local coordinate system is assigned to each axis. When the axes are in their zero position, the origins of the three local coordinate systems are defined as being coincident with the reference coordinate system of the machine. The local coordinate system's axes are also defined coaxial with the reference coordinate system's axes. Considering the movement of one of these axes affected only by squareness error. The squareness error makes the actual movement line of the axis coordinate system inclined compared to the nominal axis direction, and no longer parallel or coaxial. This is well illustrated in Figure 3.6.

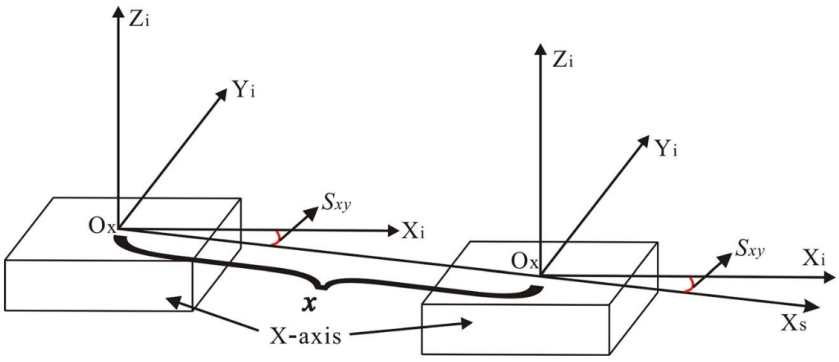


Figure 3.6: The motion of the X-axis affected by squareness error [13]

The translational transformation matrix along an arbitrary direction can be represented by the following transformation matrix:

$$\begin{bmatrix} 1 & 0 & 0 & m \cdot k_x \\ 0 & 1 & 0 & m \cdot k_y \\ 0 & 0 & 1 & m \cdot k_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

here, $\mathbf{k} = [k_x, k_y, k_z, 1]^T$ represents the unit direction vector of the line along which the translation takes place, and m the nominal displacement along the axis. Knowing the squareness error of the axis in question, the unit vector is established using equation 3.11.

$$n_X = [cos(E_{\mathrm{C(0X)Y}}), sin(E_{\mathrm{C(0X)Y}}), 0] \tag{3.11}$$

Using the unit vector obtained using eq. 3.11, the squareness error model of the X-axis is represented as:

$$\begin{bmatrix} 1 & 0 & 0 & xcos(E_{\mathrm{C(0X)Y}}) \\ 0 & 1 & 0 & xsin(E_{\mathrm{C(0X)Y}}) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$

Then, considering the Z-axis with squareness errors of the Z-axis to the X-axis ($\mathrm{E_{B(0X)Z}}$) and squareness error of the Z-axis to the Y-axis ($\mathrm{E_{A(0Y)Z}}$). The unit vector of the actual translation direction can be obtained using the equation:

$$n_Z = [sin(E_{\mathrm{B(0X)Z}}), cos(E_{\mathrm{B(0X)Z}})sin(E_{\mathrm{A(0Y)Z}}), cos(E_{\mathrm{B(0X)Z}})cos(E_{\mathrm{A(0Y)Z}})] \tag{3.13}$$

Making the squareness error model of the Z-axis:

$$\begin{bmatrix} 1 & 0 & 0 & zsin(E_{\mathrm{B(0X)Z}}) \\ 0 & 1 & 0 & zcos(E_{\mathrm{B(0X)Z}})sin(E_{\mathrm{A(0Y)Z}}) \\ 0 & 0 & 1 & zcos(E_{\mathrm{B(0X)Z}})cos(E_{\mathrm{A(0Y)Z}}) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

Applying the assumption of small angular errors provides the following squareness error models of the X- and Z-axis respectively [13]:

$$\begin{bmatrix} 1 & 0 & 0 & xE_{\mathrm{C(0X)Y}} \\ 0 & 1 & 0 & xE_{\mathrm{C(0X)Y}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} and \begin{bmatrix} 1 & 0 & 0 & zE_{\mathrm{B(0X)Z}} \\ 0 & 1 & 0 & zE_{\mathrm{A(0Y)Z}} \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.15}$$

3.3.2 Integrating the Squareness Errors

As described in the theory chapter, squareness errors of translational axes are the sum of the angular deviation of the reference straight line fitted to the measured straightness error data to the nominal axis of the two axes in question. This means that the squareness errors are synthesized from two angles, one for each of the concerned axes. When compensating for squareness errors, this is usually done in only one of the concerned axes. In the following, the X-axis is the datum axis and the squareness errors are compensated in the Y- and Z-axis.

Integrating the squareness errors, the three HTMs describing the motion of the X-, Y- and Z-axis are:

$$\begin{bmatrix} 1 & -E_{\mathrm{CX}}(x) & E_{\mathrm{BX}}(x) & E_{\mathrm{XX}}(x)+x \\ E_{\mathrm{CX}}(x) & 1 & -E_{\mathrm{AX}}(x) & E_{\mathrm{YX}}(x) \\ -E_{\mathrm{BX}}(x) & E_{\mathrm{AX}}(x) & 1 & E_{\mathrm{ZX}}(x) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.16}$$

$$\begin{bmatrix} 1 & -E_{\mathrm{CY}}(y) & E_{\mathrm{BY}}(y) & E_{\mathrm{XY}}(y)+yE_{\mathrm{C(0X)Y}} \\ E_{\mathrm{CY}}(y) & 1 & -E_{\mathrm{AY}}(y) & E_{\mathrm{YY}}(y)+y \\ -E_{\mathrm{BY}}(y) & E_{\mathrm{AY}}(y) & 1 & E_{\mathrm{ZY}}(y) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.17}$$

$$\begin{bmatrix} 1 & -E_{\mathrm{CZ}}(z) & E_{\mathrm{BZ}}(z) & E_{\mathrm{XZ}}(z)+zE_{\mathrm{B(0X)Z}} \\ E_{\mathrm{CZ}}(z) & 1 & -E_{\mathrm{AZ}}(z) & E_{\mathrm{YZ}}(z)+zE_{\mathrm{A(0Y)Z}} \\ -E_{\mathrm{BZ}}(z) & E_{\mathrm{AZ}}(z) & 1 & E_{\mathrm{ZZ}}(z)+z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.18}$$

3.4 Curve Fitting of Measured Errors

To be able to compensate for errors in every position along each of the axes on the machine, one approach is to make an analytical model of the discrete error data measured. This may be done by curve fitting a polynomial to the measured error data using the least squares method. The degree of polynomial suitable to represent the measured data depends on the shape of the measured errors. If a clear linear relationship can be seen between the axis travel and the error, a straight line may provide an accurate representation. However, if the error fluctuates along the travel of the axis, then a higher order polynomial may be more appropriate. In this section, the process of developing an analytical model to continuously represent the errors along the travel of each axis will be shown in detail.

The function "curve_fit" was made (Listing 3.1), which takes the name of the error and the degree of polynomial to be fitted to the measured data as inputs. The first part of the function (line 5-12) opens the text file containing the measured errors. The data is separated into two arrays, one containing the error values and the other containing the displacement along the axis. The next part of the script (line 20-62) constructs the reference straight line of the straightness errors of each axis, and calculates the angle between this line and the nominal axis. The first if-statement checks if the error is a straightness error or a linear positioning error. In the case of a linear positioning error (e.g. $\mathrm{E_{YY}}$), no reference straight line is made. If the error is a straightness error (e.g. $\mathrm{E_{XY}}$), a reference straight line is fitted using the numpy.polyfit function once again. The slope of the reference straight line in relation to the nominal axis is calculated based on the dot product of the directional vectors of the two. The algorithm then decides if the angle is positive or negative according to ISO 841 ([8]).

Before curve fitting, the straightness deviation stemming from the angular deviation of the reference straight line is subtracted from the straightness error (see Figure 3.7(right)) (line 66). numpy.polyfit is then used to fit a polynomial of the specified degree to the errors (see Figure 3.7(left)) (line 70). The function uses the least squares method of fitting the function. The function outputs a list of coefficients for the polynomial, along with the residual of the fitted curve. This is stored in the variable "curve_error". The residual is the sum of the squared deviations of the curve fit and is assigned to the local variable "residual_error" (line 73). Dividing the residual by the number of data points yields the mean squared error of the curve fit assigned to the local variable "MSE_error" which is returned by the function (line 74). Next, an array of displacement values for plotting the fitted polynomial is made. An if-statement checks the axis-direction and constructs the array accordingly (line 78-81). The polynomial is then made from the array of coefficients returned by the numpy.polyfit function. An if-statement checks if the polynomial is linear or of a higher degree before the polynomial is assigned to "error_model" (line 83-88).

Listing 3.1: The curve_fit function

```
def curve_fit(error, degree):

    # Treating the text file containing the error values

    with open(f'{error}.txt') as file:
        d0 = file.read()
        d1 = d0.split('\n')
        d2 = [i.split(',') for i in d1]
        d3 = [d2[i][0].split(' ') for i in range(len(d2))]

        Axis = np.zeros(len(d3)-1)
        error_parameter = np.zeros(len(d3)-1)

    for i in range(len(d3)-1):
        Axis[i] = float(d3[i][0])
        error_parameter[i] = float(d3[i][1])

    # Checking if the error is a straightness error

    if error[-2] != error[-1] and error[-2] != 'A' and error[-2] != 'B'\
        and error[-2] != 'C':

    # Constructing the reference straight line

        straight_line_fit = np.polyfit(Axis, error_parameter, 1)
        straight_line_fit[1] = 0.
        ref_straight_line = straight_line_fit[0] * Axis +\
            straight_line_fit[1]

    # Constructing vectors for the reference straight line and the nominal\
    # axis. Then calculating the angle based on the dot product

        a = np.array([Axis[-1] * 1e3 - 0, ref_straight_line[-1] - \
                      ref_straight_line[0]])
        b = np.array([Axis[-1]* 1e3 - 0, 0 - 0])
        len_a = np.sqrt(a[0]**2 + a[1]**2)
        len_b = np.sqrt(b[0]**2 + b[1]**2)
        angle = np.arccos((a @ b)/(len_a * len_b))

    # Determining the sign of the angle

        if error[-1] == 'X':
            if error[-2] == 'Y':
                if ref_straight_line[-1] < ref_straight_line[0]:
                    angle = -1 * angle
            if error[-2] == 'Z':
                if ref_straight_line[-1] > ref_straight_line[0]:
                    angle = -1 * angle
        if error[-1] == 'Y':
            if error[-2] == 'X':
                if ref_straight_line[-1] > ref_straight_line[0]:
                    angle = -1 * angle
            if error[-2] == 'Z':
                if ref_straight_line[-1] < ref_straight_line[0]:
                    angle = -1 * angle
        if error[-1] == 'Z':
            if error[-2] == 'X':
                if ref_straight_line[-1] > ref_straight_line[0]:
                    angle = -1 * angle
            if error[-2] == 'Y':
                if ref_straight_line[-1] < ref_straight_line[0]:
                    angle = -1 * angle
```

```
63
64      # Subtracting the angular deviation from the straightness deviation
65
66          error_parameter = error_parameter - ref_straight_line
67
68      # Fitting the polynomial to the measured data
69
70      curve_error = np.polyfit(Axis, error_parameter, degree, full = True, \
71                               cov = False)
72      if curve_error[1].size > 0:
73          residual_error = curve_error[1][0]
74          MSE_error = residual_error/len(error_parameter)
75
76      # Constructing the array of axis-displacements
77
78      if Axis[-1] < 0:
79          axis = np.arange(0, Axis[-1] - 1, -1.)
80      else:
81          axis = np.arange(0, Axis[-1] + 1, 1.)
82
83      error_model = np.array(curve_error[0][0] * axis**degree)
84      if degree == 1:
85          error_model = error_model + curve_error[0][1]
86      else:
87          for i in range(1, degree):
88              error_model = error_model + curve_error[0][i] * axis**(degree - i)
89
90      if error[-2] != error[-1] and error[-2] != 'A' and error[-2] != 'B'\
91          and error[-2] != 'C':
92
93          return Axis, error_parameter, axis, error_model, MSE_error, \
94                 ref_straight_line, angle
95
96      return Axis, error_parameter, axis, error_model, MSE_error
```

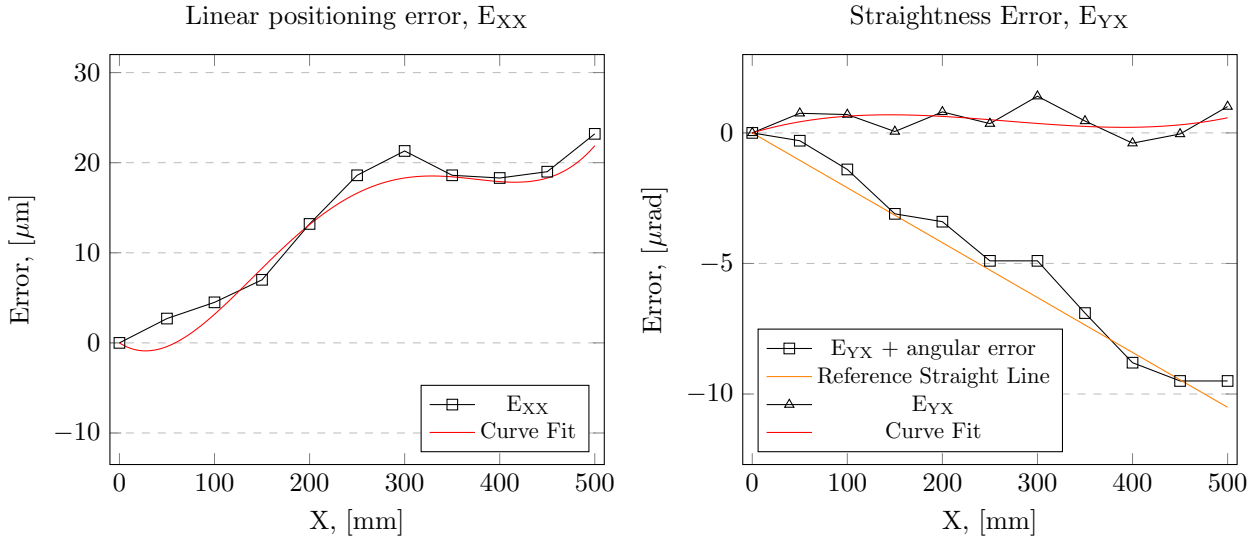


Figure 3.7: Example of fitted polynomials to the measured data. **Left:** The linear positioning error of the X-axis with the fitted polynomial. **Right:** The straightness error in Y-direction for the X-axis before and after subtracting the angular deviation. The reference straight line is also plotted.

A second function called "squareness" (Listing 3.2) was made to calculate the squareness error of axes. The function takes two inputs: axis1 and axis2. The function then uses the returned angle values from the curve-fit function and adds these together. The squareness error between axis1 and axis2 is returned.

Listing 3.2: The squareness function

```
def squareness(axis1, axis2):
    squareness_error = curve_fit(f'E_{axis2}{axis1}', 1)[6] + \
                       curve_fit(f'E_{axis1}{axis2}', 1)[6]
    return squareness_error
```

3.5 Compensation

The position of the tool-tip relative to the workpiece is described through the forward kinematics of the machine tool as the position (x_W, y_W, z_W) and the orientation (i_W, j_W, k_W). Knut Sørby thoroughly described the kinematics of the Deckel Maho in his article "Inverse kinematics near singular configurations" [9] and the following formulas were derived:

$$i = \frac{1}{2}(\sqrt{2}c_C s_B - s_C c_B + s_C), \tag{3.19}$$

$$j = \frac{1}{2}(\sqrt{2}s_C s_B + c_C c_B - c_C), \tag{3.20}$$

$$k = \frac{1}{2} + \frac{1}{2}c_B, \tag{3.21}$$

$$x = \frac{\sqrt{2}}{2}[Xs_C + (Y + Z - d)c_C]s_B + \frac{1}{2}[-Y + Z - d - (Y + Z - d)c_B]s_C + Xc_C c_B, \tag{3.22}$$

$$y = \frac{\sqrt{2}}{2}[(Y + Z - d)s_C - Xc_C]s_B + Xs_C c_B + \frac{1}{2}[(Y + Z - d)c_C c_B + (Y - Z + d)c_C], \tag{3.23}$$

$$z = \frac{1}{2}[-\sqrt{2}Xs_B + (Y + Z - d)c_B - Y + Z + d]. \tag{3.24}$$

The formulas describing the inverse relationship were derived and established as:

$$B = arccos(2k - 1), \tag{3.25}$$

$$C = arctan\Big[(1 - k)i + \sqrt{2(k - k^2)}j, \sqrt{2(k - k^2)}i + (k - 1)j\Big], \tag{3.26}$$

$$X = \Big[- y\sqrt{2(k - k^2)} - x + 2xk\Big]cosC + \Big[x\sqrt{2(k - k^2)} + 2yk - y\Big]sinC + (d - z)\sqrt{2(k - k^2)}, \tag{3.27}$$

$$Y = \Big[x\sqrt{2(k - k^2)} + yk\Big]cosC + \Big[y\sqrt{2(k - k^2)} - xk\Big]sinC - z + d - dk + zk, \tag{3.28}$$

$$Z = \Big[x\sqrt{2(k - k^2)} + yk - y\Big]cosC + \Big[y\sqrt{2(k - k^2)} - xk + x\Big]sinC + d - dk + zk. \tag{3.29}$$

These formulas are based on the ideal kinematics of the machine tool. All machines are subject to errors of different sources and of different magnitudes. In this work, an error model for the geometric errors of a five axis machine tool is presented. Integrating this error model into the model of the ideal machine kinematics is done in the following way:

Referring to Figure 3.8, the following transformations in the form of HTMs can be established:

$$T_0^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.30}$$

$$T_1^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{45^\circ} & -s_{45^\circ} & 0 \\ 0 & s_{45^\circ} & c_{45^\circ} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.31}$$

$$T_2^3 = \begin{bmatrix} c_B & s_B & 0 & 0 \\ s_B & c_B & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.32}$$

$$T_3^4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{45^\circ} & s_{45^\circ} & 0 \\ 0 & -s_{45^\circ} & c_{45^\circ} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.33}$$

$$T_4^5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.34}$$

$$T_5^w = \begin{bmatrix} c_C & s_C & 0 & 0 \\ -s_C & c_C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.35}$$

$$T_0^t = \begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.36}$$

$$T_w^t = T_w^5 T_5^4 T_4^3 T_3^2 T_2^1 T_1^0 T_0^t \tag{3.37}$$

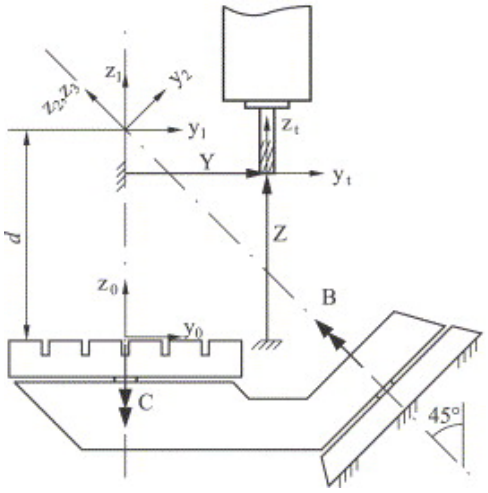


Figure 3.8: Side view illustration of the DMU 50 [9]

Including the error models of the different axes presented in this work as well as in the specialization project, the ideal transformations can be "updated" to include the geometric errors:

$$T_0^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_1^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{45^\circ} & -s_{45^\circ} & 0 \\ 0 & s_{45^\circ} & c_{45^\circ} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^{3,e} = \begin{bmatrix} c_B & s_B & 0 & 0 \\ s_B & c_B & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -E_{\mathrm{CB}} & E_{\mathrm{BB}} & E_{\mathrm{XB}} \\ E_{\mathrm{CB}} & 1 & -E_{\mathrm{AB}} & E_{\mathrm{YB}} \\ -E_{\mathrm{BB}} & E_{\mathrm{AB}} & 1 & E_{\mathrm{ZB}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{45^\circ} & s_{45^\circ} & 0 \\ 0 & -s_{45^\circ} & c_{45^\circ} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{5,e}^w = \begin{bmatrix} c_C & s_C & 0 & 0 \\ -s_C & c_C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -E_{\mathrm{CC}} & E_{\mathrm{BC}} & E_{\mathrm{XC}} \\ E_{\mathrm{CC}} & 1 & -E_{\mathrm{AC}} & E_{\mathrm{YC}} \\ -E_{\mathrm{BC}} & E_{\mathrm{AC}} & 1 & E_{\mathrm{ZC}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_0^{t,e} = \begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -E_{\mathrm{CX}} & E_{\mathrm{BX}} & E_{\mathrm{XX}} \\ E_{\mathrm{CX}} & 1 & -E_{\mathrm{AX}} & E_{\mathrm{YX}} \\ -E_{\mathrm{BX}} & E_{\mathrm{AX}} & 1 & E_{\mathrm{ZX}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -E_{\mathrm{CY}} & E_{\mathrm{BY}} & E_{\mathrm{XY}} \\ E_{\mathrm{CY}} & 1 & -E_{\mathrm{AY}} & E_{\mathrm{YY}} \\ -E_{\mathrm{BY}} & E_{\mathrm{AY}} & 1 & E_{\mathrm{ZY}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -E_{\mathrm{CZ}} & E_{\mathrm{BZ}} & E_{\mathrm{XZ}} \\ E_{\mathrm{CZ}} & 1 & -E_{\mathrm{AZ}} & E_{\mathrm{YZ}} \\ -E_{\mathrm{BZ}} & E_{\mathrm{AZ}} & 1 & E_{\mathrm{ZZ}} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_w^{t,e} = T_w^{5,e} T_5^4 T_4^3 T_3^{2,e} T_2^1 T_1^0 T_0^{t,e} \tag{3.38}$$

Huang et al. [25] derived formulas for the tool axis orientation vector and tool center position including the geometric errors. Ignoring the higher-order terms, the formulas are short and simple. However, the derivation of similar formulas for the DMU 50 results in expressions spanning several pages. A compensation strategy using the explicit analytical formulas for the forward kinematics was discarded.

A compensation strategy based on the deviation between the ideal kinematics and the actual kinematics was developed. The strategy compensates for the translational errors of the linear axes, and the translational errors arising from both the translational and the angular errors of the rotational axes. The angular errors of the linear axes are neglected in this work. The functional orientation of the tool remains uncompensated in this compensation strategy. Referring to section 2.9.1, the compensation strategy combines L-POS, L-STR, L-SQU and R-ANG.

Let P_{ideal} be the position of the tool based on the ideal machine kinematics when only the linear axes are moved:

$$P_{ideal,\ linear} = T_0^t P \tag{3.39}$$

The actual position reached by the tool P_{actual} can be calculated by including the error models of each linear axis:

$$P_{actual,\ linear} = {}^{e}T_0^X\, {}^{e}T_X^Y\, {}^{e}T_Y^Z P \tag{3.40}$$

In equation 3.40, the angular errors are neglected.

The volumetric deviation e_{linear} between $P_{actual,\ linear}$ and $P_{ideal,\ linear}$ is found by subtracting the actual position by the ideal position. The compensation value c_{linear} is the deviation with reversed sign.

The compensation values for the rotational axes are found in a similar way. The desired position of the rotational axes is read from the NC-code generated by the post-processor. Considering a coordinate system located in the center of the machine table, the ideal position of a point on the machine table $P_{ideal,\ rotational}$ is then calculated using the transformation based on the forward kinematics. The actual position reached by the point on the machine table $P_{actual,\ rotational}$ is calculated by reading the errors from the polynomials fitted to the measured and calculated error data. The coordinate system is rotated by the amount of each angular error. It is then translated by the amount of the translational errors before it is rotated according to the desired position of the B- and C-axis. The deviation $e_{rotational}$ is calculated as the difference between the actual and the ideal position, as for the linear axes.

In practice, the compensation values arising from the errors in the rotational axes $c_{rotational}$ are calculated first and added to the desired X-, Y- and Z-coordinates. The compensation values based on the errors of the linear axes are then calculated based on these coordinates. The compensation strategy is summarized in Figure 3.9.

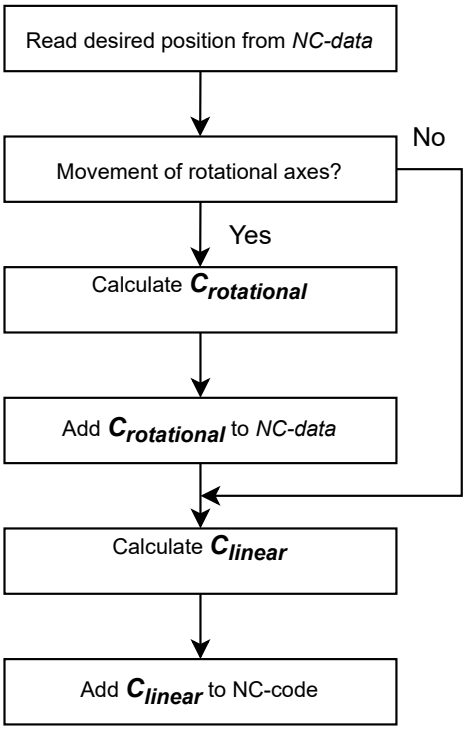


Figure 3.9: The compensation strategy

3.6 Compensation Procedure

Based on the volumetric error model, including the continuously modelled errors, a compensation function is made. It compensates for the deviation in tool center position from the nominal position.

Referring to the flow chart of the compensation strategy in Figure 3.9, the compensation values based on the errors of the rotational axes needs to be calculated first. A Python script was made to calculate the position of a point on the work table of the DMU 50 after uncompensated rotation of both rotational axes. The script can be read in Listing 3.3. First, the packages needed in the script are imported. In this case, only *numpy* and the self-made function *curve_fit* is needed. The curve fit function for the rotational axes is similar to that for the linear axes in Listing 3.1, and can be read in Appendix E. The parameter d (vertical distance from machine tool coordinate frame to the rotation center of the B- and C-axis) is then defined as 154.996 mm, the value given by the machine manufacturer. The transformation-function *T_r* based on the ideal machine tool kinematics is then defined (line 6-48), taking the rotation angle of the B- and C-axis in degrees as arguments. The B- and C-angle is converted to radians, and the sine and cosine to these angles are defined. Sine and cosine of 45° is also defined. The HTMs defined in the code that follows are recognized as the ideal forward kinematics of the DMU 50 as previously described (eq 3.30 - 3.35). The product of the matrix multiplication of all the defined HTMs are returned as T.

Another four separate functions are then defined (line 50-94). The three first functions are simple rotation matrices about the X-, Y- and Z-axes, and the fourth is a translation-matrix.

The following function *T_er* (line 96-128) returns a transformation matrix similar to that returned by *T_r*, but including the errors of B- and C-axis. The errors of the rotational axes are first defined using the average values calculated from the four repetitions of measurements (line 98-103 and line 113-118). The rotation matrices are then constructed based on the measured errors, before the translation matrices are constructed (line 105-111 and line 120-126). The rotation matrices constructed from the measured errors are then multiplied first, before the translation matrices are multiplied. Lastly, the transformation based on the ideal forward kinematics is multiplied with the product of all the previous matrices (line 128).

The last few lines of the script (line 131-133) demonstrates how the deviation between the transformation based on the ideal kinematics and the one including the errors is found. The output of the three print commands is shown in Figure 3.10. By reversing the signs of the deviation in X-, Y-, and Z-direction, the volumetric compensations values to compensate for the translational and angular errors of the rotational axes are derived.

Listing 3.3: Script to calculate deviation in uncompensated rotation motion

```
1  import numpy as np
2  from CurveFittingFunction_rotational import curve_fit as cf_rot
3
4  d = 154.996
5
6  def T_r(B, C):
7
8      B = np.deg2rad(B)
9      C = np.deg2rad(C)
10
11     s_B, c_B = np.sin(B), np.cos(B)
12     s_C, c_C = np.sin(C), np.cos(C)
13     s_45, c_45 = np.sin(np.pi/4), np.cos(np.pi/4)
14
15     T_d = np.array([[1, 0, 0, 0],
16                     [0, 1, 0, 0],
17                     [0, 0, 1, d],
18                     [0, 0, 0, 1]])
19
20     T_45 = np.array([[1,   0,      0,    0],
```

```
                        [0, c_45, -s_45, 0],
                        [0, s_45,  c_45, 0],
                        [0,   0,     0,  1]])

    T_B = np.array([[c_B, -s_B, 0, 0],
                    [s_B,  c_B, 0, 0],
                    [ 0,    0,  1, 0],
                    [ 0,    0,  0, 1]])

    T_n45 = np.array([[1,    0,     0,    0],
                      [0,  c_45, s_45,  0],
                      [0, -s_45, c_45,  0],
                      [0,    0,     0,    1]])

    T_nd = np.array([[1, 0, 0,  0],
                     [0, 1, 0,  0],
                     [0, 0, 1, -d],
                     [0, 0, 0,  1]])

    T_C = np.array([[ c_C, s_C, 0, 0],
                    [-s_C, c_C, 0, 0],
                    [  0,   0,  1, 0],
                    [  0,   0,  0, 1]])


    T =  T_d @ T_45 @ T_B @ T_n45 @ T_nd @ T_C

    return T

def rot_x(angle):

    c, s = np.cos(angle), np.sin(angle)

    T_x = np.array([[1, 0,  0, 0],
                    [0, c, -s, 0],
                    [0, s,  c, 0],
                    [0, 0,  0, 1]])

    return T_x

def rot_y(angle):

    c, s = np.cos(angle), np.sin(angle)

    T_y = np.array([[c, 0, -s, 0],
                    [0, 1,  0, 0],
                    [s, 0,  c, 0],
                    [0, 0,  0, 1]])

    return T_y

def rot_z(angle):

    c, s = np.cos(angle), np.sin(angle)

    T_z = np.array([[c, -s, 0, 0],
                    [s,  c, 0, 0],
                    [0,  0, 1, 0],
                    [0,  0, 0, 1]])

    return T_z

def translate(displacement):

```

```
85      X = displacement[0]
86      Y = displacement[1]
87      Z = displacement[2]
88
89      T = np.array([[1, 0, 0, X],
90                    [0, 1, 0, Y],
91                    [0, 0, 1, Z],
92                    [0, 0, 0, 1]])
93
94      return T
95
96  def T_er(B, C):
97
98      E_XB = cf_rot('E_XB_avg.', 4)[3][round(B)] * 1e-3
99      E_YB = cf_rot('E_YB_avg.', 4)[3][round(B)] * 1e-3
100     E_ZB = cf_rot('E_ZB_avg.', 8)[3][round(B)] * 1e-3
101     E_AB = cf_rot('E_AB_avg.', 7)[3][round(B)] * 1e-6
102     E_BB = cf_rot('E_BB_avg.', 7)[3][round(B)] * 1e-6
103     E_CB = cf_rot('E_BB_avg.', 7)[3][round(B)] * 1e-6
104
105     T_E_XB = translate([E_XB, 0, 0])
106     T_E_YB = translate([0, E_YB, 0])
107     T_E_ZB = translate([0, 0, E_ZB])
108
109     T_E_AB = rot_x(E_AB)
110     T_E_BB = rot_y(E_BB)
111     T_E_CB = rot_z(E_CB)
112
113     E_XC = cf_rot('E_XC_avg.', 8)[3][round(C)] * 1e-3
114     E_YC = cf_rot('E_YC_avg.', 8)[3][round(C)] * 1e-3
115     E_ZC = cf_rot('E_ZC_avg.', 8)[3][round(C)] * 1e-3
116     E_AC = cf_rot('E_AC_avg.', 7)[3][round(C)] * 1e-6
117     E_BC = cf_rot('E_BC_avg.', 7)[3][round(C)] * 1e-6
118     E_CC = cf_rot('E_BC_avg.', 7)[3][round(C)] * 1e-6
119
120     T_E_XC = translate([E_XC, 0, 0])
121     T_E_YC = translate([0, E_YC, 0])
122     T_E_ZC = translate([0, 0, E_ZC])
123
124     T_E_AC = rot_x(E_AC)
125     T_E_BC = rot_y(E_BC)
126     T_E_CC = rot_z(E_CC)
127
128     return T_E_AB @ T_E_BB @ T_E_CB @ T_E_AC @ T_E_BC @ T_E_CC @ T_E_XB @ \
129         T_E_YB @ T_E_ZB @ T_E_XC @ T_E_YC @ T_E_ZC @ T_r(B, C)
130
131 print(T_r(90, 90) @ np.transpose([179.009, 94.545, 143.533, 1]))
132 print(T_er(90, 90) @ np.transpose([179.009, 94.545, 143.533, 1]))
133 print((T_r(90, 90) @ np.transpose([179.009, 94.545, 143.533, 1]))\
134       - (T_er(90, 90) @ np.transpose([179.009, 94.545, 143.533, 1])))
```

```
[134.68404283 -16.91958937 305.62241063   1.        ]
[134.72796186 -17.02465949 305.64018776   1.        ]
[-0.04391904  0.10507011 -0.01777713  0.        ]
```

Figure 3.10: Output of the three print commands in Listing 3.3. The first line shows the coordinates of the point after transformation using the ideal kinematics, the second shows the coordinates of the point after transformation including the errors of the rotational axes and finally the third line shows the deviation between the two transformations. The output is given in homogeneous coordinates (four columns).

The compensation values obtained from the script in Listing 3.3 should then be added to the ideal NC-data generated by the postprocessor.

The NC-data including the compensation values for the errors of the rotational axes can then be used as input in the script in Listing 3.4. The script imports the curve_fit function for the linear axes (Listing 3.1) in addition to the squareness function (Listing 3.2). The transformation function for the linear axes using ideal kinematics is then defined as T (line 5-28), which takes an array with displacement values in X-, Y-, and Z-direction as input. The transformation matrix is returned.

T_e (line 30-75) takes two arguments, one being the desired displacement, and the other being the the desired three dimensional position. The coordinates of the desired position (line 36-38) is used to define the three translational errors in each linear axis by reading the value of the fitted polynomial at the desired positio (line 40-50). The squareness errors are then defined (line 53-55) before the transformation matrices for each of the three axes are constructed (line 58-68) and the product of the multiplication returned as T_e (line 73-75).

As a proof-of-concept, the same displacements and positions as the ones used for measuring the geometrical errors are used as the NC-data. *compensation* takes the NC-data as input (line 90), and constructs an empty array for the compensated NC-data (line 91-92). For every row in the array of NC-data, the desired position is read as the next row in NC-data (line 94). The displacement equals the desired position (line 95). The nominal position is then calculated as the product of the initial coordinates and the ideal transformation (line 96). The actual position is calculated using the transformation matrix including the geometrical errors (line 97). The deviation between them is calculated as a simple difference (line 99), and the compensation values are generated and added to the nominal NC-data (line 100-101). The array of compensated NC-data is returned.

Figure 3.11 shows the two arrays of NC-data. On the left is the ideal NC-data, and on the right is the NC-data compensated for the translational errors of the linear axes. The compensated NC-data was applied on the DMU 50, and the geometric errors were measured another four times. The results are presented in the following chapter.

Listing 3.4: Script to generate compensated NC-data

```
1  import numpy as np
2  from CurveFittingFunction_linear import curve_fit as cf
3  from squareness import squareness as s
4
5  def T(displacement):
6
7      X = displacement[0]
8      Y = displacement[1]
9      Z = displacement[2]
10
11     T_X = np.array([[1, 0, 0, X],
12                     [0, 1, 0, 0],
13                     [0, 0, 1, 0],
14                     [0, 0, 0, 1]])
15
16     T_Y = np.array([[1, 0, 0, 0],
17                     [0, 1, 0, Y],
18                     [0, 0, 1, 0],
19                     [0, 0, 0, 1]])
20
21     T_Z = np.array([[1, 0, 0, 0],
22                     [0, 1, 0, 0],
23                     [0, 0, 1, Z],
24                     [0, 0, 0, 1]])
25
26     T = T_X @ T_Y @ T_Z
27
28     return T
```

```
29
30 def T_e(displacement, position):
31
32     X = displacement[0]
33     Y = displacement[1]
34     Z = displacement[2]
35
36     pos_X = position[0]
37     pos_Y = position[1]
38     pos_Z = position[2]
39
40     E_XX = cf('E_XX_avg.', 4)[3][round(pos_X)] * 1e-3
41     E_YX = cf('E_YX_avg.', 5)[3][round(pos_X)] * 1e-3
42     E_ZX = cf('E_ZX_avg.', 8)[3][round(pos_X)] * 1e-3
43
44     E_XY = cf('E_XY_avg.', 8)[3][round(pos_Y)] * 1e-3
45     E_YY = cf('E_YY_avg.', 6)[3][round(pos_Y)] * 1e-3
46     E_ZY = cf('E_ZY_avg.', 4)[3][round(pos_Y)] * 1e-3
47
48     E_XZ = cf('E_XZ_avg.', 8)[3][round(pos_Z)] * 1e-3
49     E_YZ = cf('E_YZ_avg.', 7)[3][round(pos_Z)] * 1e-3
50     E_ZZ = cf('E_ZZ_avg.', 6)[3][round(pos_Z)] * 1e-3
51
52
53     S_YX = s('Y', 'X')
54     S_ZY = s('Z', 'Y')
55     S_ZX = s('Z', 'X')
56
57
58     T_e_X = np.array([[1, 0, 0,   E_XX + X],
59                       [0, 1, 0,     E_YX  ],
60                       [0, 0, 1,     E_ZX  ],
61                       [0, 0, 0,       1   ]])
62
63     T_e_Y = np.array([[1, 0, 0,   E_XY + Y * S_YX],
64                       [0, 1, 0,      E_YY + Y    ],
65                       [0, 0, 1,        E_ZY      ],
66                       [0, 0, 0,         1        ]])
67
68     T_e_Z = np.array([[1, 0, 0, E_XZ + (-Z) * S_ZX],
69                       [0, 1, 0, E_YZ + (-Z) * S_ZY],
70                       [0, 0, 1,      E_ZZ + Z     ],
71                       [0, 0, 0,          1        ]])
72
73     T_e = T_e_X @ T_e_Y @ T_e_Z
74
75     return T_e
76
77 X = cf('E_XX_avg.', 4)[0]
78 #Y = cf('E_YY_avg.', 6)[0]
79 #Z = cf('E_ZZ_avg.', 8)[0]
80 NC_data = np.zeros((len(X), 4))
81 #NC_data = np.zeros((len(Y), 4))
82 #NC_data = np.zeros((len(Z), 4))
83 for i in range(len(NC_data)):
84     NC_data[i] = [0., 0., 0., 1]
85     NC_data[i][0] = X[i]
86     #NC_data[i][1] = Y[i]
87     #NC_data[i][2] = Z[i]
88
89
90 def compensation(NC_data):
91     NC_data_comp = np.zeros(((len(NC_data)), 4))
92     NC_data_comp[0] = [0., 0., 0., 1]
```

```
    for i in range(1, len(NC_data)):
        desired_position = NC_data[i]
        displacement = desired_position
        nominal_position = T(displacement) @ np.transpose([0, 0, 0, 1])
        actual_position = T_e(displacement, desired_position)\
        @ np.transpose([0, 0, 0, 1])
        deviation = actual_position - nominal_position
        compensation_values = -1 * deviation
        NC_data_comp[i] = NC_data[i] + compensation_values

    return NC_data_comp

Compensated_NC_data = compensation(NC_data)
```

	0	1	2	3
0	0	0	0	1
1	50	0	0	1
2	100	0	0	1
3	150	0	0	1
4	200	0	0	1
5	250	0	0	1
6	300	0	0	1
7	350	0	0	1
8	400	0	0	1
9	450	0	0	1
10	500	0	0	1

	0	1	2	3
0	0	0	0	1
1	49.9956	-0.000949353	-0.00156385	1
2	99.9937	-0.000753776	0.000895042	1
3	149.993	-0.000362045	-0.00238055	1
4	199.992	-0.000215857	-0.00110904	1
5	249.99	-0.000387325	0.00149589	1
6	299.987	-0.000716486	0.000389921	1
7	349.984	-0.000948794	-0.000920075	1
8	399.981	-0.000872622	0.00182009	1
9	449.979	-0.000456766	0.000298728	1
10	499.981	1.20629e-05	-0.00292907	1

Figure 3.11: The NC-data generated based on the ideal kinematics (**left**) and the NC-data compensated for the translational errors of the linear axes (**right**). Both arrays are expressed in homogeneous coordinates.

Chapter 4

Results

4.1 Uncompensated Error Motion

Following are the resulting average errors calculated from the measured points on the linear as well as on the rotational axes. For each calculated point, the average standard deviation of the four rounds of measurement is indicated in the form of error bars.

The linear positioning error, E_{ij} when $i = j$, is the largest translational error in all three linear axes. The peak magnitudes are approximately 20.5 μm for the X-axis, 40.8 μm for the Y-axis and -16.5 μm for the Z-axis. The straightness errors of the linear axes have much lower peak magnitudes.

4.1.1 X-axis

The linear positioning error E_{XX} shows a clear increasing trend although the measurement uncertainty indicated by the error bars is quite high. The two straightness errors E_{YX} and E_{ZX} are much lower. E_{YX} Shows a clear decreasing trend, while no clear trend can be seen for E_{ZX}. The measurement uncertainty is also much higher for the latter than for E_{YX}.

The angular errors have relatively low magnitudes compared to the magnitude of the measurement uncertainty. It is therefore difficult to draw any conclusions from the presented data.

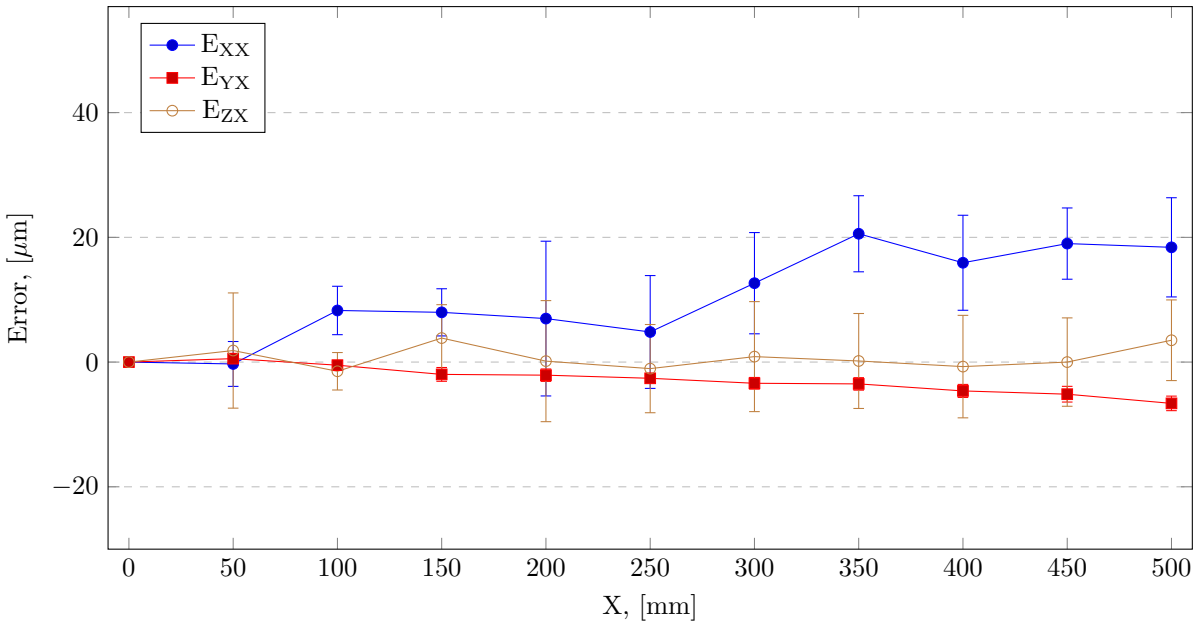


Figure 4.1: Plot of the average translational error of the X-axis

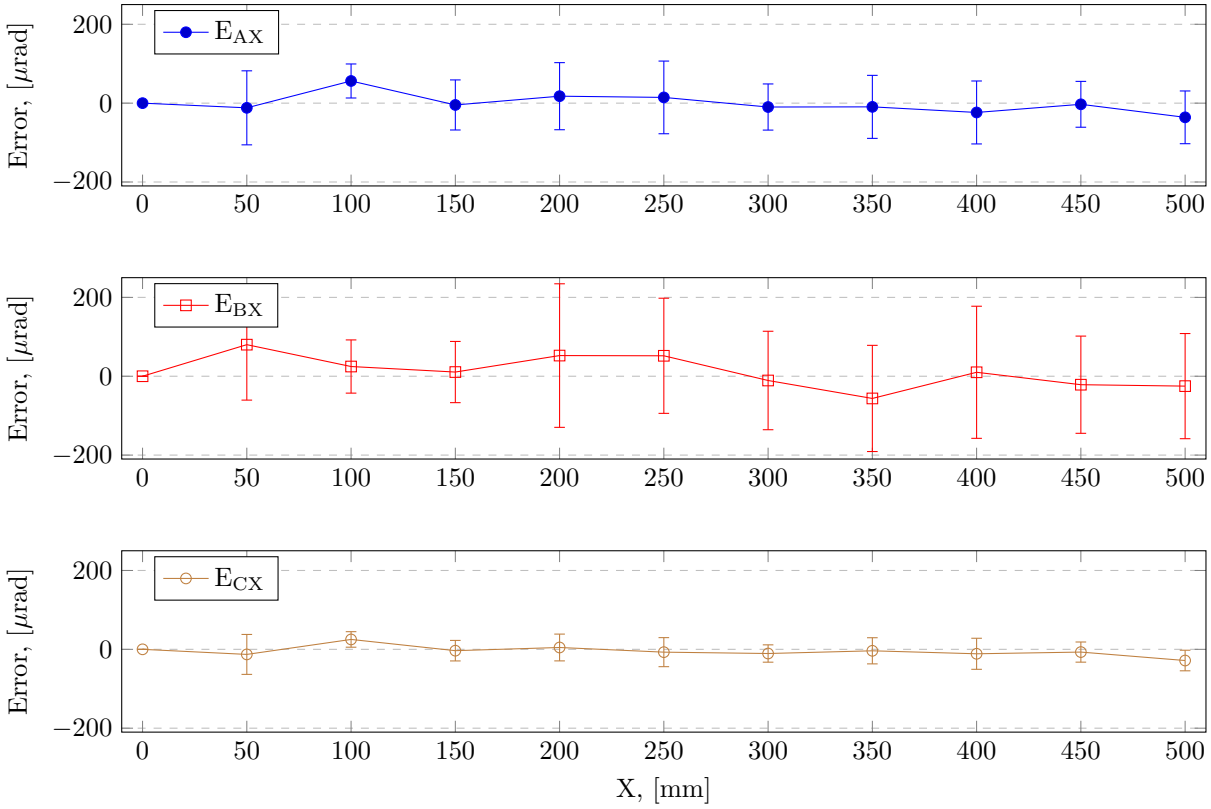


Figure 4.2: Plot of the average angular error of the X-axis

4.1.2 Y-axis

The linear positioning error is almost double the magnitude as that of the X-axis, topping out at 48 μm. The measurement uncertainty varies from one point to another, but appears to be somewhat lower than for the linear positioning error of the X-axis. The two straightness errors are relatively low, especially E_{XY} having a peak absolute value of just over 5 μm. The measurement uncertainty for E_{XY} is also quite low on average compared to the other errors of the Y-axis.

There is also great variation in the measurement uncertainty of the angular errors. As is the case for the X-axis, drawing any conclusions based on the measured data proves difficult also for the Y-axis.

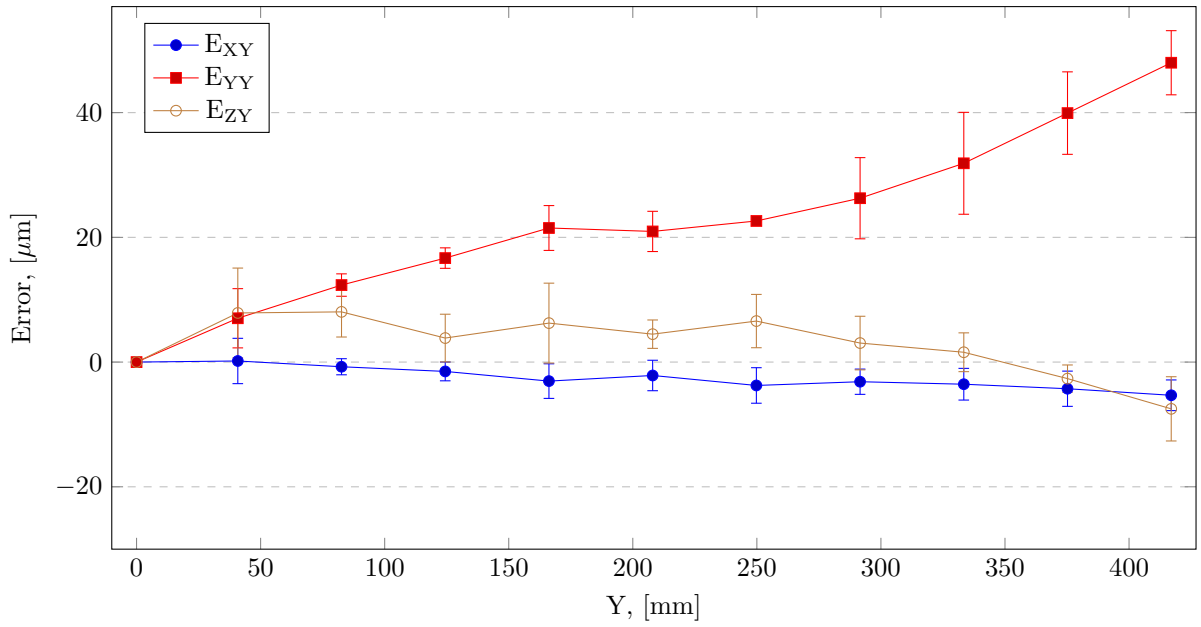


Figure 4.3: Plot of the average translational error of the Y-axis

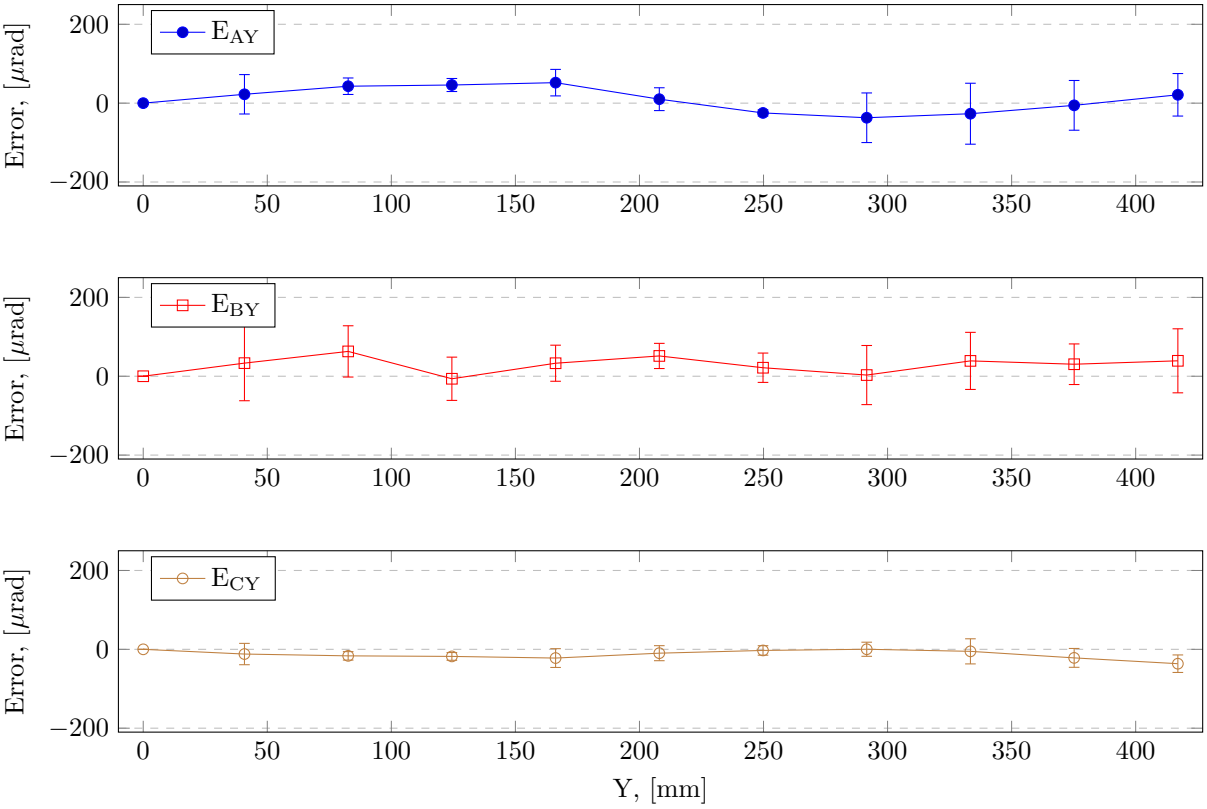


Figure 4.4: Plot of the average angular error of the Y-axis

4.1.3 Z-axis

Again the linear positioning error is the dominant error. It also has the largest measurement uncertainty of the three translational errors of the Z-axis. The straightness errors are about half the magnitude of the linear positioning error. E_{YZ} shows significantly lower measurement uncertainty then the other two errors.

The angular error in A-axis direction shows larger magnitude than the other two angular errors, but the measurement uncertainty is also larger, making the result inconclusive.

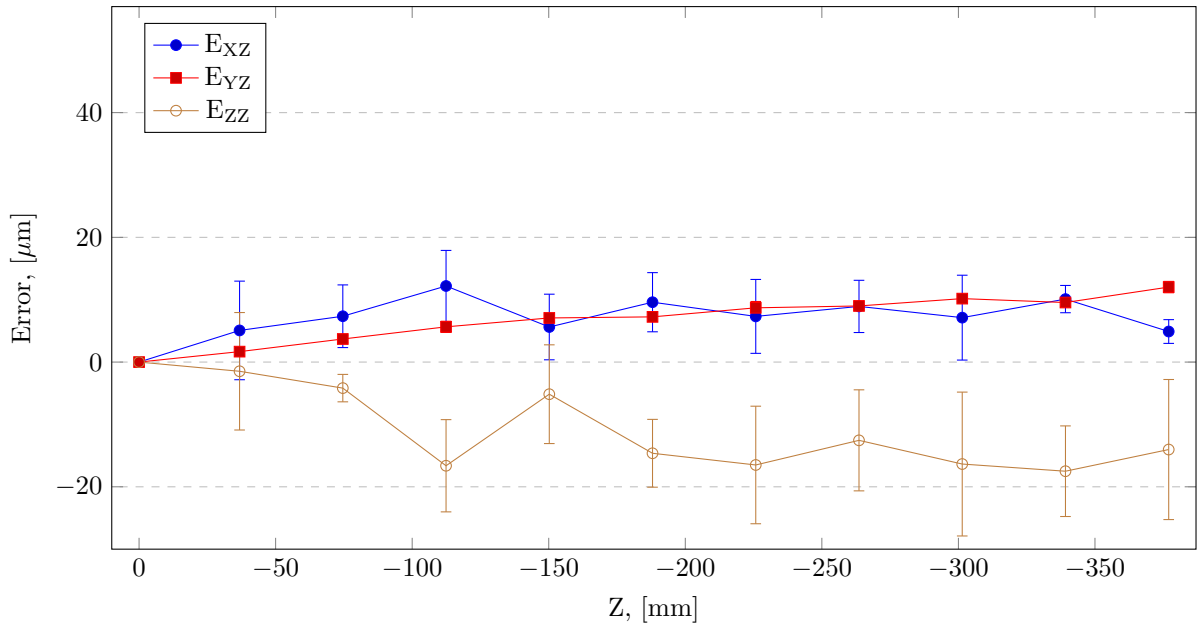


Figure 4.5: Plot of the average translational error of the Z-axis

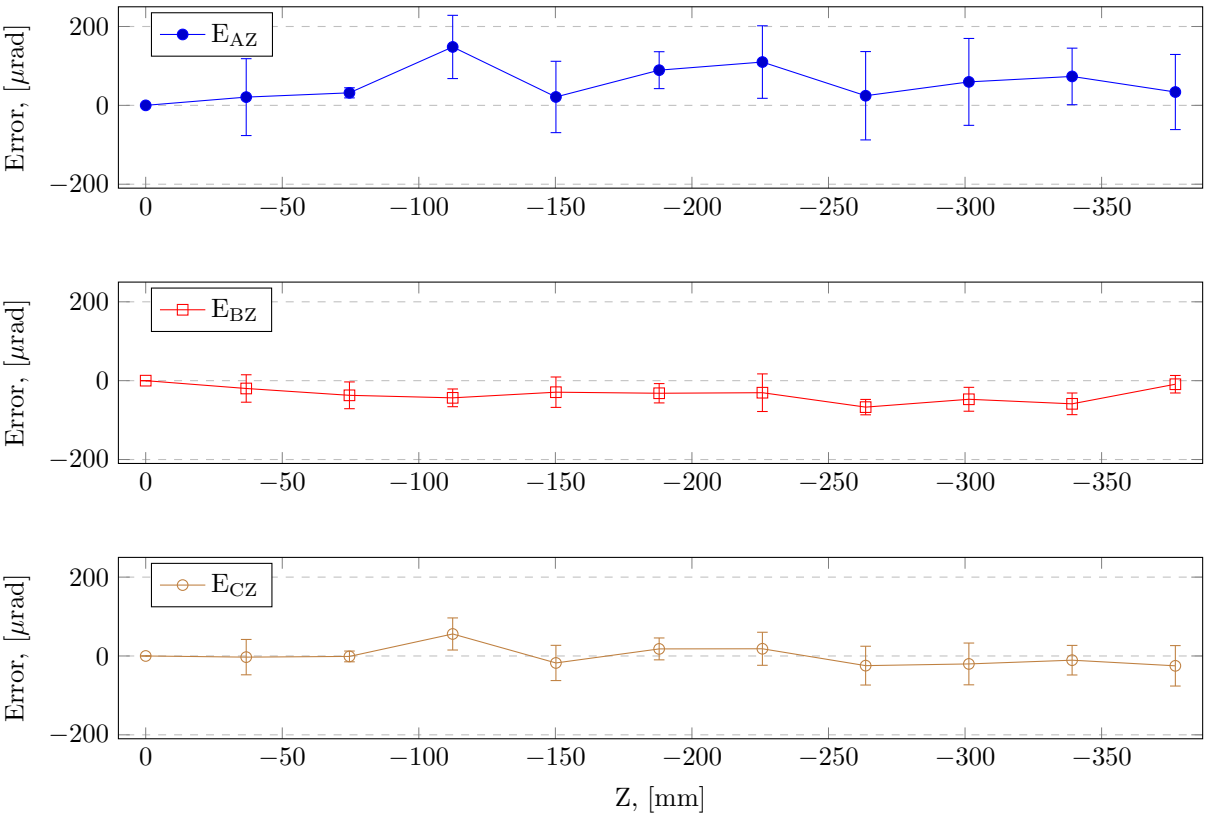


Figure 4.6: Plot of the average angular error of the Z-axis

4.1.4 B-axis

What immediately stands out about the translational errors of the B-axis is the magnitude of the measurement uncertainty, which is quite large. The magnitude of the measured errors is at the same time quite low, which makes it hard to interpret the data.

The angular errors of the B- and C-axes are represented with the same Y-scale, and compared to those of the C-axis, the angular errors of the B-axis are negligable.

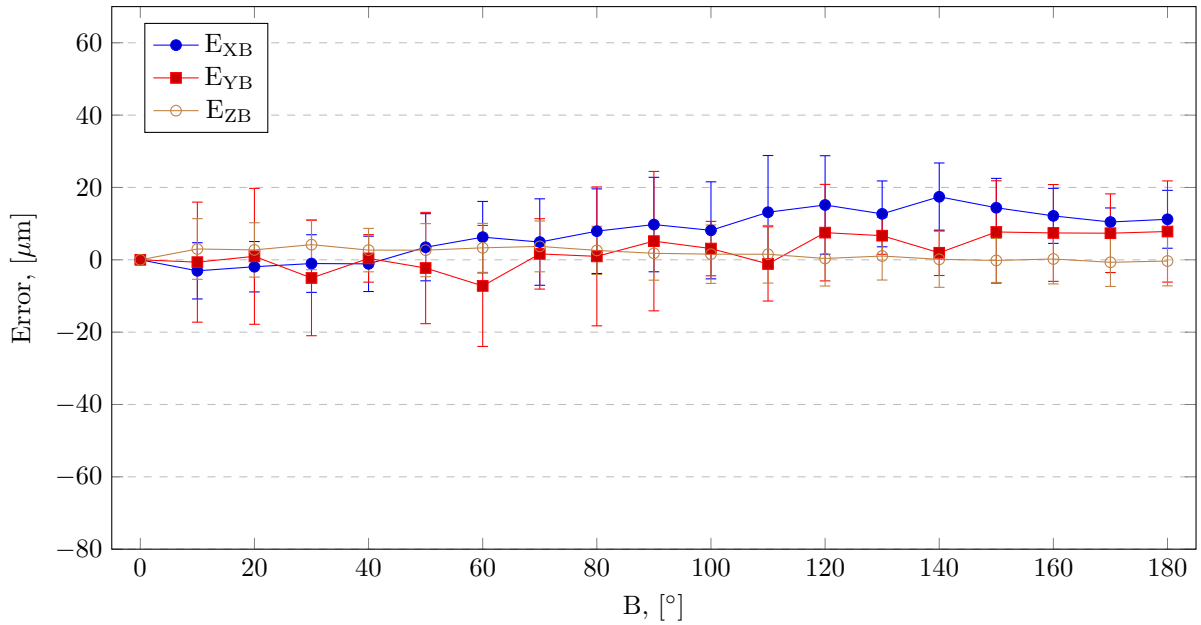


Figure 4.7: Plot of the average translational error of the B-axis

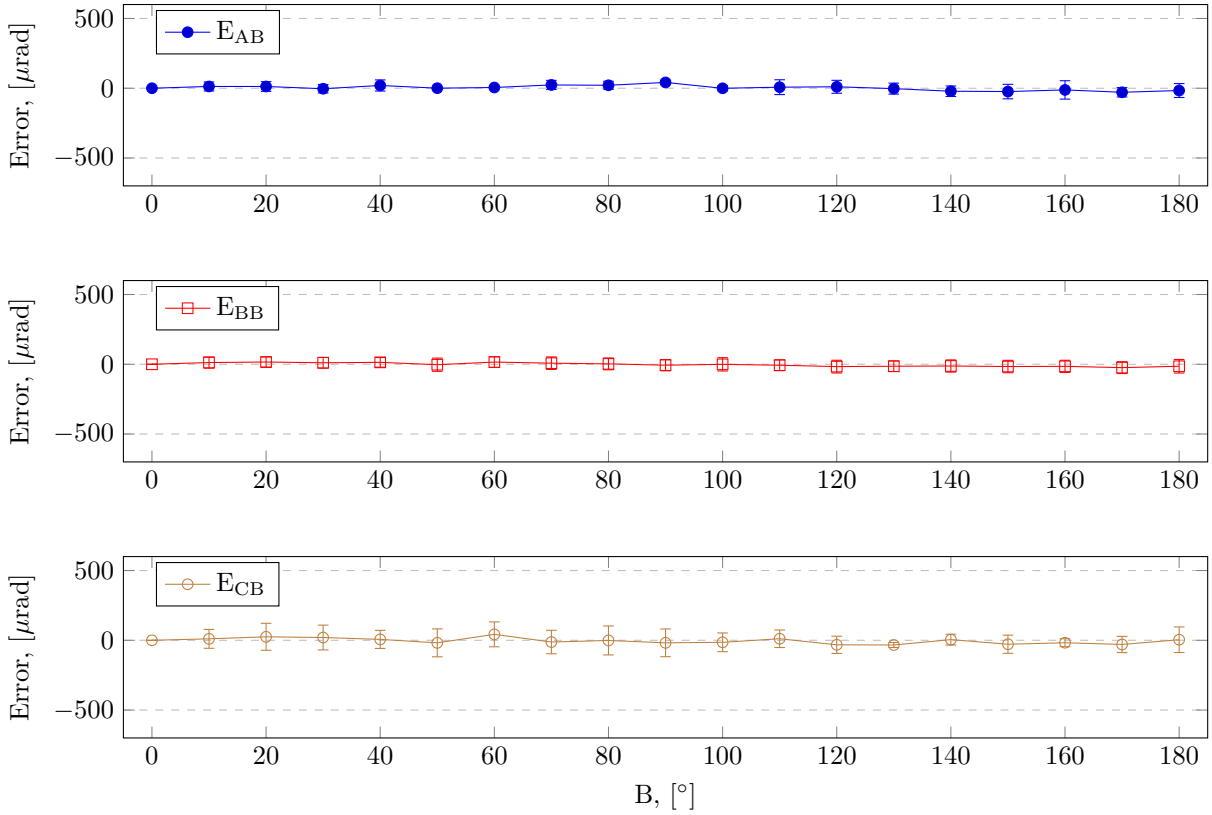


Figure 4.8: Plot of the average angular error of the B-axis

4.1.5 C-axis

The translational errors of the C-axis show somewhat clearer characteristics than those of the B-axis, although the measurement uncertainty is still high compared to the error magnitude. E_{ZC} stands out as the largest error. It also appears in a cyclic manner.

E_{AC} shows the clearest tendency out of the three angular errors, but the measurement uncertainty is higher than for the other two angular errors which also have much lower error magnitudes.

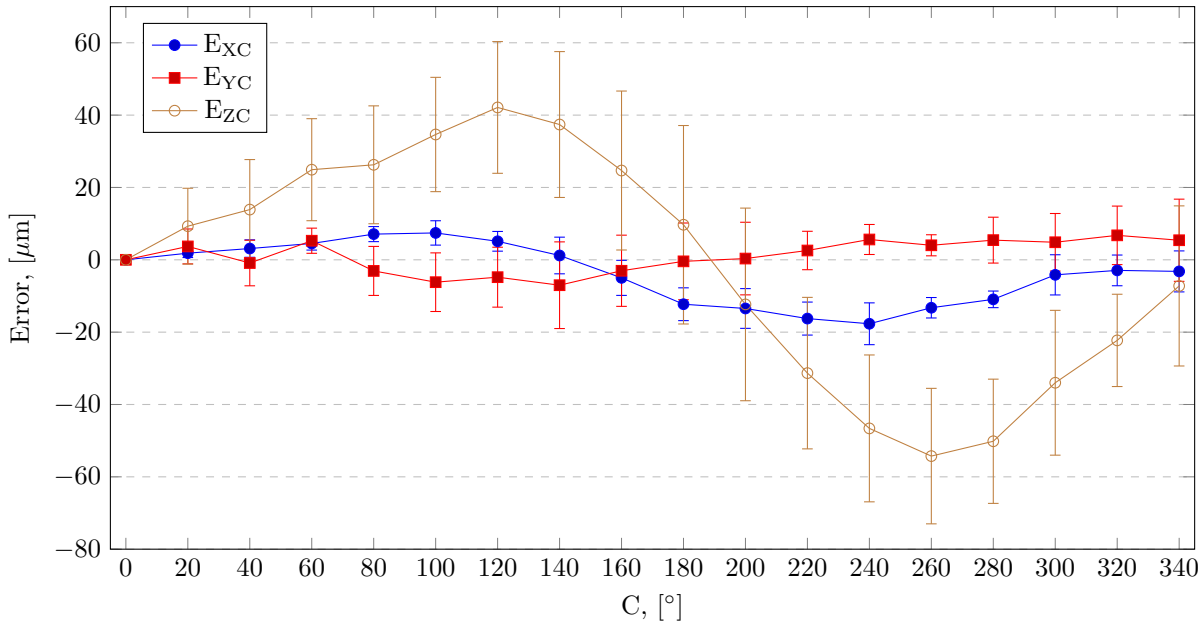


Figure 4.9: Plot of the average translational error of the C-axis

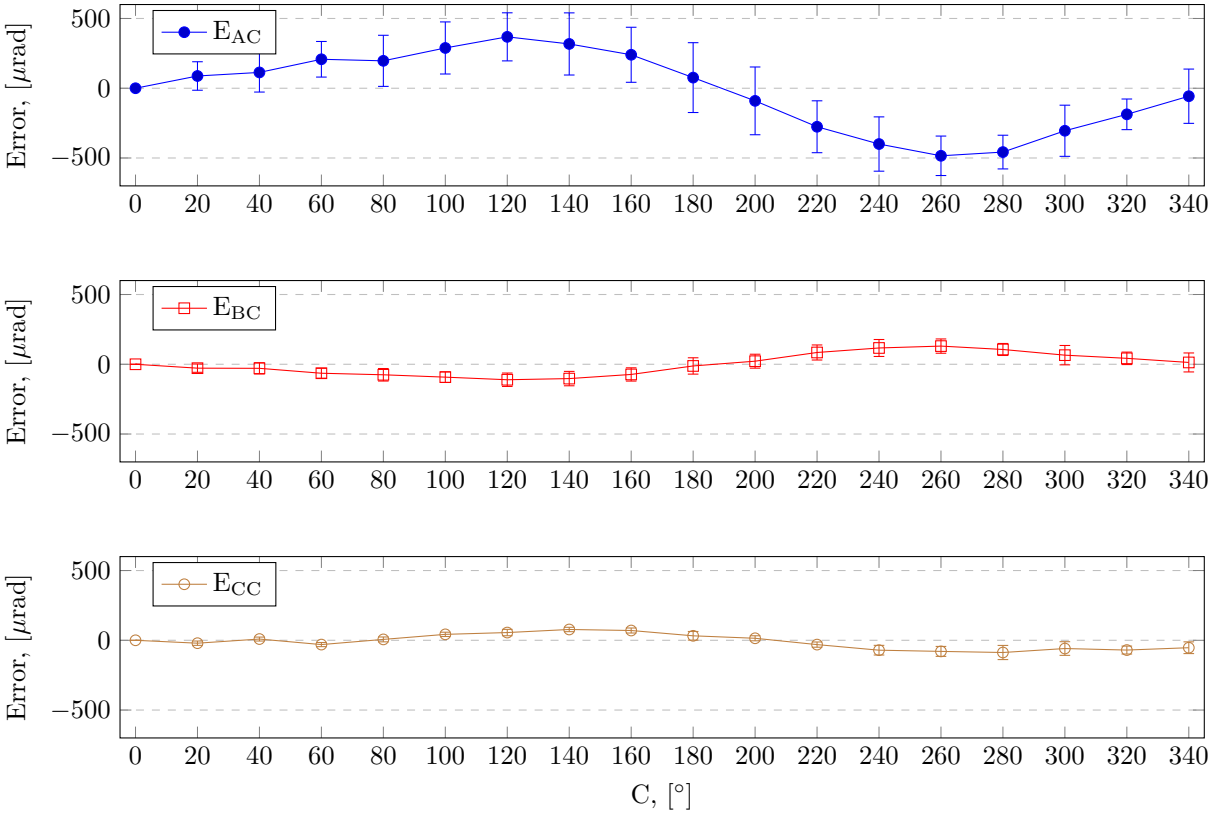


Figure 4.10: Plot of the average angular error of the C-axis

4.2 Compensated Error Motion

Following are the results of the measurements made on the DMU 50 after applying the compensated NC-data generated by the compensation algorithm implemented in the Python-script (Listing 3.4). Only the results of the translational errors are included, as the angular errors of the linear axes remain uncompensated.

4.2.1 X-axis

The maximum magnitude of E_{XX} is almost halved from 20.6 μm to 10.4 μm, while the minimum magnitude only differs by 1 μm. E_{ZX} was initially low, and remains so also after compensation. The maximum magnitude of the error shows an increase after compensation. The variability is high compared to the error magnitude. In the case of E_{YX}, the variability is low compared to the variability of the other two translational errors. As the error shows a nearly linear relationship with the axis travel, the squareness error dominates. By comparing the plots pre and post compensation, it can clearly be observed that the squareness error compensation has improved the axis motion and reduced the overall error. The maximum absolute error magnitude was improved from 6.6 μm before compensation, to 1.3 μm after compensation.

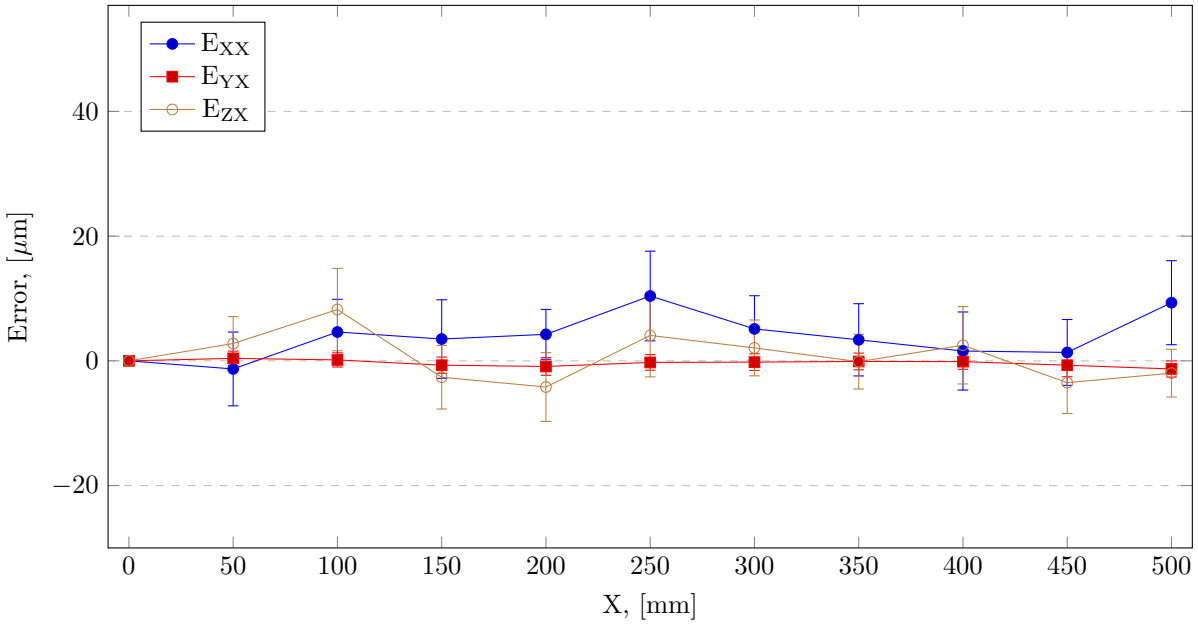


Figure 4.11: Plot of the average compensated translational error of the X-axis

4.2.2 Y-axis

A clear improvement in the linear positioning error is observed. The pre-compensation peak magnitude was 40.8 μm, and has been reduced to approximately 10 μm. The two straightness errors have seemingly increased in magnitude. The assumed reason for this increase in error magnitude is the way the coordinate system was constructed in relation to the measurement data. This will be discussed in depth in chapter 5.

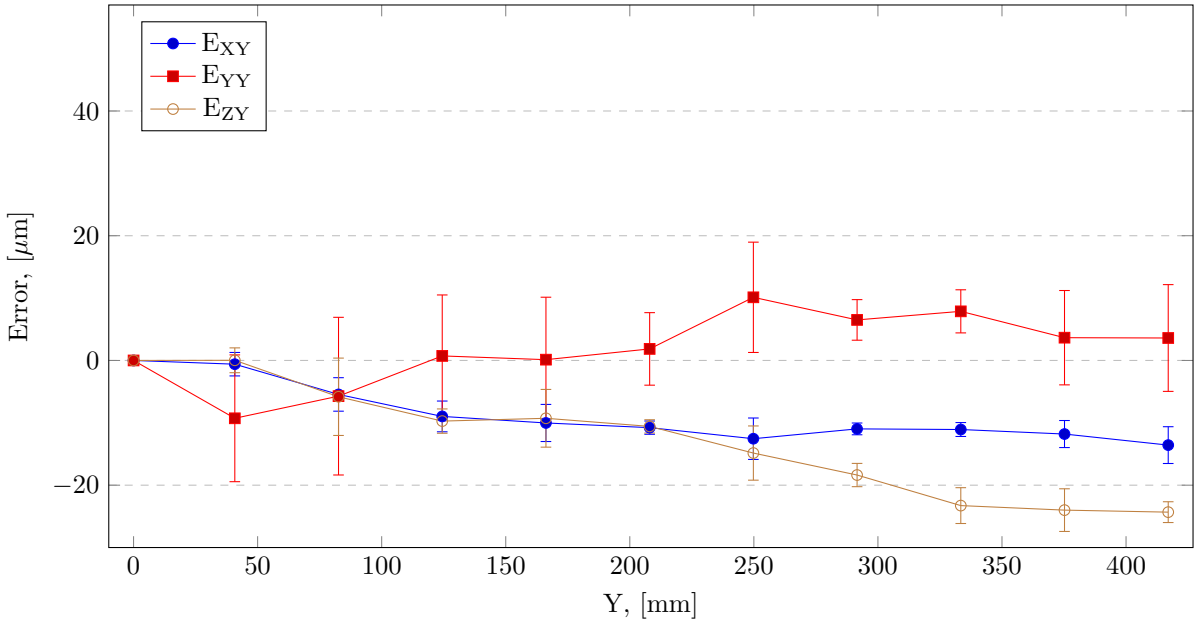


Figure 4.12: Plot of the average compensated translational error of the Y-axis

4.2.3 Z-axis

Although the peak magnitudes of the linear positioning error of the Z-axis have increased slightly from the pre-compensated motion, most of the measured points have been brought closer to zero. The variability of the measured results is relatively large also in this error. The straightness error E_{XZ} remains largely the same as the pre-calibration values, while E_{YZ} has been reduced substantially.

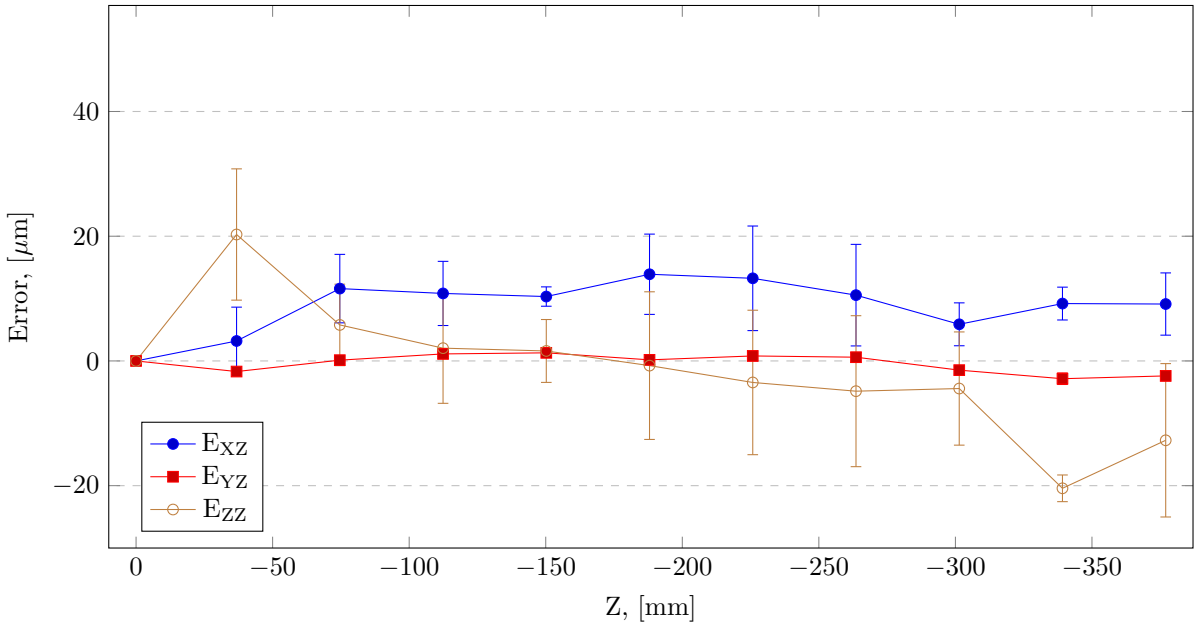


Figure 4.13: Plot of the average compensated translational error of the Z-axis

Chapter 5

Discussion

5.1 Variability of the Measured Errors

In the process of measuring the error motions of a CNC machine tool, there are numerous sources of error which can impact the measurements. By *sources of error*, factors influencing the value of the measured parameter other than the factors that were intended to be measured is meant. Examples of such sources of error for laser tracker measurement of CNC machine tool error motion are:

- Thermal expansion
- Uncertainty in laser length measurement
- Uncertainty in laser tracker angle measurement
- Other uncertainties related to the laser tracker measurement (see chapter 2.8)
- Vibration

All of these error sources, and more, contribute to the variability of the measured errors. However, some sources of error contribute more than others. A quantitative evaluation of the contribution of each is near impossible, but in some cases an error source dominates the uncertainty budget.

5.1.1 Measurement Uncertainty

A pattern can be recognized in the measured errors where some of the translational errors show much less variability than others. Referring to chapter 4, it can be observed that errors E_{YX} and E_{YZ} have very short error bars compared to the other errors, indicating a higher measurement quality. A possible explanation to this is proposed based on the assumed influence of the movement of the machine axis and the angular measurement of the laser tracker. Generally speaking, the linear positioning repeatability is lower than the repeatiability of the position in the transverse directions. Depending on the position of the laser tracker in relation to the axis movements of the machine, the influence of the angle performance on the evaluation of the error varies. With the setup used in this work, the errors in Y-direction have a low sensititvity to the angle performance of the laser tracker. The influence of the machine axis movements and the influence of the angle performance of the laser tracker is summarized in Table 5.1. For E_{YX} and E_{YZ}, the influence of both are low. This coincides well with the observed variability of the measurement results.

Based on the proposed relation between the position of the laser tracker and the variability in the measured results, a new and improved method of measuring the geometric errors of the linear axes can be developed. As presented in Table 5.1, two of the in total six straightness errors had low influence both

Table 5.1: Influence of the machine tool and the laser tracker on the variability of the measured translational errors of the linear axes

Influence on variability (X-axis)	E_{XX}	E_{YX}	E_{ZX}
Machine tool performance	High	Low	Low
Laser tracker performance	High	Low	High
Influence on variability (Y-axis)	E_{XY}	E_{YY}	E_{ZY}
Machine tool performance	Low	High	Low
Laser tracker performance	High	Low	High
Influence on variability (Z-axis)	E_{XZ}	E_{YZ}	E_{ZZ}
Machine tool performance	Low	Low	High
Laser tracker performance	High	Low	High

from the axis motion of the machine tool and from the angle performance of the laser tracker. Referring to Figure 5.1, the setup used to measure the errors in this work is similar to that illustrated in Pos. 1. By moving the laser tracker (referred to as *LT* in Figure 5.1) to the positions indicated in Pos. 2 and Pos. 3, a reduction in measurement uncertainty may be achieved. In Pos. 2, E_{XY} would have low influence from both the machine axis motion and the angle performance of the laser tracker. The same goes for E_{XZ}. In Pos. 3, E_{ZX} and E_{ZY} would also have lower measurement uncertainty than in Pos. 1. The linear positioning errors E_{XX} and E_{ZZ} are both highly influenced by both machine axis motion and laser tracker angle performance when measured in with the setup in Pos. 1. The repeatability of the machine axis motion is considered constant in this work, and hence will always influence the measurement result of the linear positioning errors of the machine tool. However, the influence of the angle performance of the laser tracker can be made lower by positioning the laser tracker in line with the axis motion. This means that the influence of the angle performance on E_{XX} is lower in Pos. 2, and in Pos. 3 for E_{ZZ}. Position 1 and 2 are practically feasible. The horizontal orientation of the laser tracker in position 3 however, is outside of the recommended use of the laser tracker.

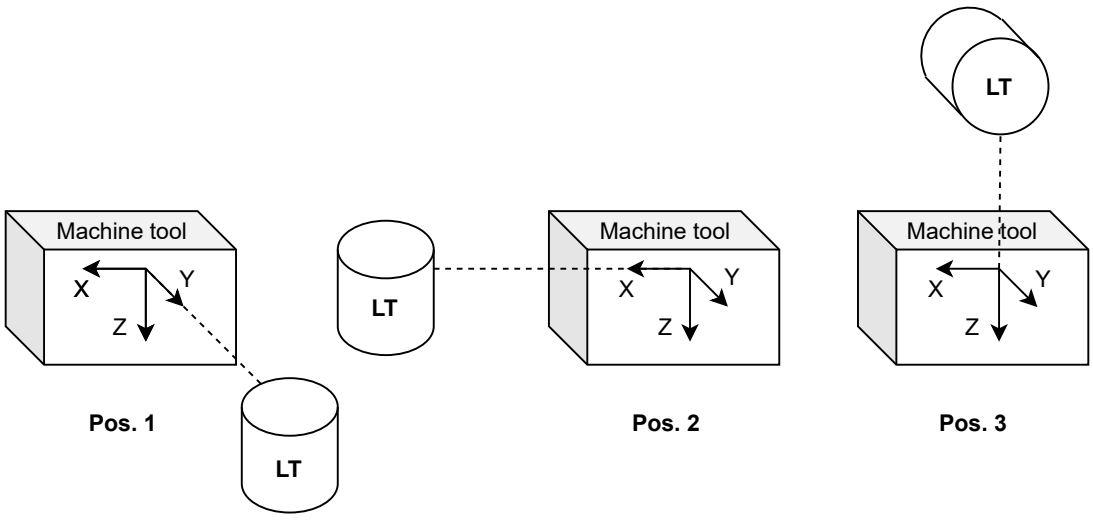


Figure 5.1: Three proposed positions of the laser tracker in order to minimize measurement uncertainty

Table 5.2 shows the assumed influences of machine axis motion and laser tracker angle performance using the three alternative measuring setups illustrated in Figure 5.1.

The lower measurement uncertainty achieved by having multiple measurement setups presupposes linear

Table 5.2: Influence of the machine tool and the laser tracker on the variability of the measured translational errors of the linear axes with alternative positions of the laser tracker

Influence on variability (X-axis)	E_{XX}	E_{YX}	E_{ZX}
Machine tool performance	High	Low	Low
Laser tracker performance	Low	Low	Low
Influence on variability (Y-axis)	E_{XY}	E_{YY}	E_{ZY}
Machine tool performance	Low	High	Low
Laser tracker performance	Low	Low	Low
Influence on variability (Z-axis)	E_{XZ}	E_{YZ}	E_{ZZ}
Machine tool performance	Low	Low	High
Laser tracker performance	Low	Low	Low

axis movement and is hence only valid for the translational errors of the linear axes. Both the translational and rotational movement of the rotational axes depends heavily on the angle performance of the laser tracker, and the measurement uncertainty will always suffer as a result of this dependency. This can easily be observed particularly for the translational errors of the B-axis and the axial error of the C-axis, where the variability of the measured results are quite large.

5.1.2 Y-axis Straightness Errors

The measured straightness errors of the Y-axis after compensation seem to indicate that the errors have increased in magnitude as opposed to the straightness errors of the X- and Z-axis where the straightness error magnitudes have decreased. A proposed cause of this inconsistency is the alignment of the coordinate system which the axis movements are measured in reference to. The uncompensated axis motion was first measured in reference to a coordinate system fitted to the uncompensated X- and Z-axis motion. Then compensation was applied, and a new coordinate system alignment was made to the compensated motion of the X- and Z-axis. Ideally, the same coordinate frame alignment should be used in both measurement setups to best be able to directly compare the results. However, since the measurements were made at different times, the setup had to be taken down after measuring the uncompensated motion and reset for measuring the axis-motion after compensation was applied. This means a new frame alignment had to be made for measuring the compensated axis-motion, which may slightly affect how the errors are represented. The new frame alignment will mainly affect the straightness error representation. A small change in the angle of the axes in the reference coordinate system will have little influence on the representation of the linear positioning errors.

An improvement to the measurement and compensation procedure is proposed where the measurement of the uncompensated axis motion is measured and the errors determined in the same measurement setup, and with the same reference coordinate system frame alignment, as the compensated axis motion is measured. This will ensure that the errors are compared to the same reference coordinate system.

Chapter 6

Conclusion and Further Work

The uncertainty in the error measurement is quite high for some of the translational errors of the linear axes, which makes determining the correct compensation values difficult. On the other hand, a few of the errors show very little variability between subsequent measurements - low measurement uncertainty, which forms a good base on which to determine compensation values. By analysis of the factors contributing to the measurement uncertainty in laser tracker measurement, a new measuring procedure aimed at minimizing the measurement uncertainty based on repositioning the laser tracker was proposed. The validity of the proposed measuring procedure presupposes linear axis-movement, and will therefore not be effective for the rotational axes of the machine tool.

A compensation strategy was developed which first uses the measured errors of the rotational axes to determine compensation values in the linear axes. These compensation values are added to the compensation values of the linear axes to construct compensated NC-code. The effectiveness of the compensation method on the linear axes was verified by repeating the error measurement using compensated NC-data. The results showed improvement in the linear positioning error of all three linear axes of the machine tool comparable to that achieved by Liu et. al [26] and Cui et. al [27]. The straightness errors of the X- and Z-axis was also improved, while the Y-axis showed slight deterioration. An explanation for this discrepancy as the result of using different reference coordinate frame alignments was proposed.

Further Work

The measuring and compensation procedure should be repeated, and the effectiveness re-verified including the proposed changes to the measurement procedure. The compensation strategy needs further refinement and could be integrated in the post-processor of a CAD-CAM system, or a stand-alone post-processor.

The errors of the linear axes were already established in the work on the specialization project report. In this work, the measurement procedure was repeated, and the variability of the measured errors evaluated. A similar method for determining the errors of the two rotational axes was implemented, and the same approach to evaluate the variability was used.

Bibliography

[1] S. Sartori and G. X. Zhang. "Geometric Error Measurement and Compensation of Machines". In: *CIRP Annals - Manufacturing Technology* 44.2 (Jan. 1995), pp. 599–609. ISSN: 17260604. DOI: 10.1016/S0007-8506(07)60507-1.

[2] H. Schwenke et al. "Error mapping of CMMs and machine tools by a single tracking interferometer". In: *CIRP Annals - Manufacturing Technology* 54.1 (Jan. 2005), pp. 475–478. ISSN: 00078506. DOI: 10.1016/S0007-8506(07)60148-6.

[3] International Standardisation Organization. *ISO/TR 16907, Machine tools - Numerical compensation of geometric erros*. 2015.

[4] D.C. Thompson and P.A. McKeown. "The Design of an Ultra-Precision CNC Measuring Machine". In: *CIRP Annals* 38.1 (1989), pp. 501–504. ISSN: 0007-8506. DOI: https://doi.org/10.1016/S0007-8506(07)62755-3. URL: http://www.sciencedirect.com/science/article/pii/S0007850607627553.

[5] L Norberto de Lacalle and Aitzol Lamikiz. "Machine Tools for Removal Processes: A General View". In: *Machine Tools for High Performance Machining*. Ed. by L N de Lacalle and A Lamikiz. London: Springer London, 2009, pp. 1–45. ISBN: 978-1-84800-380-4. DOI: 10.1007/978-1-84800-380-4{_}1. URL: https://doi.org/10.1007/978-1-84800-380-4_1.

[6] *Kinematic chain - Wikipedia*. URL: https://en.wikipedia.org/wiki/Kinematic_chain.

[7] Kenneth Waldron and James Schmiedeler. "Kinematics". In: *Springer Handbook of Robotics*. Ed. by Bruno Siciliano and Oussama Khatib. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 9–33. ISBN: 978-3-540-30301-5. DOI: 10.1007/978-3-540-30301-5{_}2. URL: https://doi.org/10.1007/978-3-540-30301-5_2.

[8] International Standardisation Organization. *ISO 841:1974, Industrial automation systems and integration — Numerical control of machines — Coordinate system and motion nomenclature*. 2001.

[9] Knut Sørby. "Inverse kinematics of five-axis machines near singular configurations". In: *International Journal of Machine Tools and Manufacture* 47.2 (2007), pp. 299–306. ISSN: 0890-6955. DOI: https://doi.org/10.1016/j.ijmachtools.2006.03.011. URL: http://www.sciencedirect.com/science/article/pii/S0890695506001027.

[10] K M Lynch and F C Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017. ISBN: 9781316609842. URL: https://books.google.no/books?id=8uS3AQAACAAJ.

[11] International Standardisation Organization. *ISO 230-1:2012, Test code for machine tools — Part 1: Geometric accuracy of machines operating under no-load or quasi-static conditions*. 2012.

[12] Jae Ha Lee, Yu Liu, and Seung Han Yang. "Accuracy improvement of miniaturized machine tool: Geometric error modeling and compensation". In: *International Journal of Machine Tools and Manufacture* 46.12-13 (Oct. 2006), pp. 1508–1516. ISSN: 08906955. DOI: 10.1016/j.ijmachtools.2005.09.004.

[13] Guoqiang Fu et al. “Squareness error modeling for multi-axis machine tools via synthesizing the motion of the axes”. In: *International Journal of Advanced Manufacturing Technology* 89.9-12 (Apr. 2017), pp. 2993–3008. ISSN: 14333015. DOI: 10.1007/s00170-016-9259-z. URL: https://link.springer.com/article/10.1007/s00170-016-9259-z.

[14] *Rotation matrix - Wikipedia*. URL: https://en.wikipedia.org/wiki/Rotation_matrix.

[15] Kwang-Il Lee, Dong-Mok Lee, and Seung-Han Yang. “Parametric modeling and estimation of geometric errors for a rotary axis using double ball-bar”. In: (). DOI: 10.1007/s00170-011-3834-0.

[16] Zhenjiu Zhang and Hong Hu. “A general strategy for geometric error identification of multi-axis machine tools based on point measurement”. In: *International Journal of Advanced Manufacturing Technology* 69.5-8 (June 2013), pp. 1483–1497. ISSN: 02683768. DOI: 10.1007/s00170-013-5094-7. URL: https://link.springer.com/article/10.1007/s00170-013-5094-7.

[17] H. Schwenke et al. “Geometric error measurement and compensation of machines-An update”. In: *CIRP Annals - Manufacturing Technology* 57.2 (Jan. 2008), pp. 660–675. ISSN: 00078506. DOI: 10.1016/j.cirp.2008.09.008.

[18] R.M.D. Mahbubur et al. “Positioning accuracy improvement in five-axis milling by post processing”. In: *International Journal of Machine Tools and Manufacture* 37.2 (1997), pp. 223–236. ISSN: 0890-6955. DOI: https://doi.org/10.1016/0890-6955(95)00091-7. URL: http://www.sciencedirect.com/science/article/pii/0890695595000917.

[19] Soichi Ibaraki, Chiaki Oyama, and Hisashi Otsubo. “Construction of an error map of rotary axes on a five-axis machining center by static R-test”. In: *International Journal of Machine Tools and Manufacture* 51.3 (Mar. 2011), pp. 190–200. ISSN: 08906955. DOI: 10.1016/j.ijmachtools.2010.11.011.

[20] International Standardisation Organization. *Uncertainty of measurement - Part 3: Guide to the expression of uncertainty in measurement (GUM:1995)*. 1995.

[21] Dehong Huo, Paul G. Maropoulos, and Chun Hung Cheng. “The framework of the virtual laser tracker - A systematic approach to the assessment of error sources and uncertainty in laser tracker measurement”. In: *Advances in Intelligent and Soft Computing*. Vol. 66 AISC. Springer Verlag, 2010, pp. 507–523. ISBN: 9783642104299. DOI: 10.1007/978-3-642-10430-5{_}39. URL: https://link.springer.com/chapter/10.1007/978-3-642-10430-5_39.

[22] *What is Inspire? — Hexagon MI*. URL: https://www.hexagonmi.com/products/metrology-software/inspire/what-is-inspire.

[23] R. Ramesh, M. A. Mannan, and A. N. Poo. “Error compensation in machine tools - a review. Part I: Geometric, cutting-force induced and fixture-dependent errors”. In: *International Journal of Machine Tools and Manufacture* 40.9 (July 2000), pp. 1235–1256. ISSN: 08906955. DOI: 10.1016/S0890-6955(00)00009-2.

[24] Zhenjiu Zhang and Hong Hu. “Measuring geometrical errors of linear axis of machine tools based on the laser tracker”. In: *2011 IEEE International Conference on Robotics and Biomimetics, ROBIO 2011*. 2011, pp. 2276–2281. ISBN: 9781457721373. DOI: 10.1109/ROBIO.2011.6181637.

[25] Nuodi Huang et al. “Integrated post-processor for 5-axis machine tools with geometric errors compensation”. In: *International Journal of Machine Tools and Manufacture* 94 (July 2015), pp. 65–73. ISSN: 08906955. DOI: 10.1016/j.ijmachtools.2015.04.005.

[26] Hongwei Liu et al. “Geometric error measurement and compensation for NC machine tools”. In: *ACM International Conference Proceeding Series*. New York, New York, USA: Association for Computing Machinery, Oct. 2018, pp. 10–15. ISBN: 9781450365666. DOI: 10.1145/3293688.3293692. URL: http://dl.acm.org/citation.cfm?doid=3293688.3293692.

[27] Gangwei Cui et al. “Geometric error compensation software system for CNC machine tools based on NC program reconstructing”. In: *International Journal of Advanced Manufacturing Technology* 63.1-4 (Nov. 2012), pp. 169–180. ISSN: 02683768. DOI: 10.1007/s00170-011-3895-0.

Appendices

Appendix A

Conference Paper

This page is left intentionally blank.

Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 00 (2021) 000–000

www.elsevier.com/locate/procedia

14th CIRP Conference on Intelligent Computation in Manufacturing Engineering, Gulf of Naples, Italy

Geometrical testing and accuracy improvement of five-axis machines by use of laser tracker

Simen H. Ellingsdalen, Knut Sørby*

Norwegian University of Science and Technology, Department of Mechanical and Industrial Engineering, NO-7491 Trondheim, Norway
* Corresponding author. Tel.: +4773590374. *E-mail address:* knut.sorby@ntnu.no

Abstract

In this paper, a compensation strategy for the volumetric errors of both the linear and the rotational axes of a five-axis machine is presented. The strategy compensates the translational error motions of the linear axes, as well as the volumetric error arising from the translational and the angular errors of the rotational axes. The angular errors of the linear axes are neglected, and the functional orientation of the tool axis remains uncompensated. The errors of each axis are measured using a Leica AT960 laser tracker. The measurement uncertainty of laser tracker measurement of machine tools is addressed, and the observed variability in the measured errors is discussed.

Peer-review under responsibility of the scientific committee of the 15th CIRP Conference on Intelligent Computation in Manufacturing Engineering.

Keywords: Five-axis machines; Error compensation; Laser tracker

1. Introduction

In numerical compensation of machine tools the geometrical errors of the machine must be identified, and a compensation algorithm for eliminating the errors must be applied. Numerical compensation has been around for decades, and the interest for numerical compensation based on software saw an increase in the 1970's. In 1977, Prof. R. Hocken received the CIRP Taylor Medal for implementing error compensation on a coordinate measuring machine [1]. Before numerical compensation entered the industry in the 1980's, accuracy of machine tools was ensured exclusively through mechanical optimization. While impressive sub-micron level accuracy was reached by some companies, the investment in both time and manual labour was substantial [2]. As both the power and availability of computational tools increased, the possibility for applying active pre-calibrated error compensation became a reality and a way of saving cost compared to manufacturing machine tools for absolute accuracy [3]. This concept justifies a change in how machine tools are manufactured in the sense that instead of manufacturing machine tools with very high mechanical accuracy, the machine tool only needs to be built with the required precision to be compensated to the correct level of accuracy. The ISO technical report on numerical compensation of machine tools, ISO/TR 16907:2015 [4] lists this as one of the major potential benefits of numerical compensation, reducing the overall cost of machine tool production.

Five-axis machines are geometrically complex machine tools with high versatility. With the addition of two rotational axes to the common three-axis milling machine, complex three-dimensional shapes can be machined with the tool perpendicular to the workpiece surface, ensuring optimal tool-workpiece interaction. Being able to machine inclined surfaces and chamfers using a standard face milling tool or an endmill may reduce the investment necessary in either time spent on several workpiece setups or in specialized tooling. Part accuracy may also be enhanced as the need for several workpiece setups is reduced as several sides of the same workpiece may be machined in the same setup.

Peer-review under responsibility of the scientific committee of the 15th CIRP Conference on Intelligent Computation in Manufacturing Engineering.

However, as the opportunities in terms of part complexity, efficiency enhancement and convenience increase with the addition of the rotational axes, so does the potential for errors. Each additional axis introduces its own set of error parameters influencing part accuracy and quality. These errors may have different sources, and include [4]: Component imperfections, alignment errors, elastic deformation of components, thermo-mechanical errors, loads and load variations, interpolation errors, errors in motion control and control software and errors in compensation. The error model of the machine tool also grows in complexity with the addition of the rotational axes.

Nomenclature

Error parameters of machine axes according to ISO 230-1:2012 [5]:

E_{ij}	Where *E* indicates "error", i indicates direction of error motion (i = X, Y, Z, B or C). In the case where i = j, the error is called the linear positioning error (linear axis) or angular positioning error (rotational axis). When i = A, B or C and i ≠ j, the error is called angular error motion (linear axis) or tilt error motion (rotational axis). If i = X, Y or Z and i ≠ j, the error is called straightness error motion (linear axis). If i = X or Y and j = B or C, the error is called a radial error motion and if i = Z the error is called axial error motion (rotational axis).
$E_{i(0j)k}$	Squareness error in direction of i-axis of k-axis in reference to the j-axis.
HTM	Homogeneous transformation matrix
TCP	Tool Center Position

2. Machine Kinematics

In a previous work done by Sørby [6], the forward kinematics of the Deckel Maho DMU 50 eVolution, hereafter referred to simply as the DMU 50, was developed. A number of coordinate frames were defined and the transformations between them described by a series of homogeneous transformation matrices. In Figure 1, the coordinate frames are drawn onto a schematic side view of the worktable and spindle of the DMU 50.

The following HTMs describe the transformations between the coordinate frames using the shorthand notation $\sin\varphi = s_\varphi$ and $\cos\varphi = c_\varphi$:

$$T_0^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{1.1}$$

$$T_1^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{45°} & -s_{45°} & 0 \\ 0 & s_{45°} & c_{45°} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{1.2}$$

$$T_2^3 = \begin{bmatrix} c_B & -s_B & 0 & 0 \\ s_B & c_B & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.3}$$

$$T_3^4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{45°} & s_{45°} & 0 \\ 0 & -s_{45°} & c_{45°} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.4}$$

$$T_0^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.5}$$

$$T_2^3 = \begin{bmatrix} c_C & -s_C & 0 & 0 \\ s_C & c_C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.6}$$

$$T_0^t = \begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.7}$$

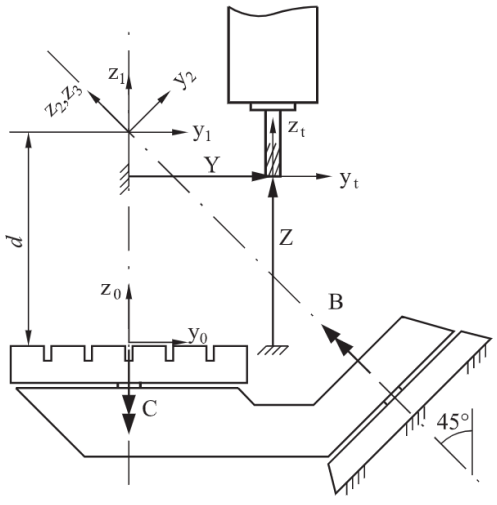


Figure 1: Side view of a five-axis machine with non-orthogonal rotary axes. Shown with the B-axis in B = 0º position [6].

The coordinate transformation from the milling tool's coordinate frame to the workpiece coordinate frame can be expressed by multiplications of transformation matrices:

$$T_w^{\,t} = (T_5^{\,w})^{-1}(T_4^{\,5})^{-1}(T_3^{\,4})^{-1}(T_2^{\,3})^{-1}(T_1^{\,2})^{-1}(T_0^{\,1})^{-1}T_0^{\,t} \tag{1.8}$$

3. Error Model

3.1 Linear Axes

Each axis of a machine tool has its own associated errors. Each axis has six position dependent geometric errors – three of them translational and three of them angular. It is also common to separate the straightness error motions of the linear axes into straightness errors and squareness errors. This means there exist three additional error parameters for the linear axes and another two per rotational axis. Lastly, location errors of the rotational axes may be included in the error model, but they will be neglected in this work.

The error model of a linear axis based on HTMs can be represented in the following way taking the X-axis of the DMU 50 as an example:

$${}^{e}T_0^{\,X} = \begin{bmatrix} 1 & -E_{CX} & E_{BX} & E_{XX}+X \\ E_{CX} & 1 & -E_{AX} & E_{YX} \\ -E_{BX} & E_{AX} & 1 & E_{ZX} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.9}$$

Similarly, the error model for the Y- and Z-axis including squareness errors becomes:

$${}^{e}T_X^{\,Y} = \begin{bmatrix} 1 & -E_{CY} & E_{BY} & E_{XY}+yE_{C(0X)Y} \\ E_{CY} & 1 & -E_{AY} & E_{YY}+y \\ -E_{BY} & E_{AY} & 1 & E_{ZY} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.10}$$

$${}^{e}T_Y^{\,Z} = \begin{bmatrix} 1 & -E_{CZ} & E_{BZ} & E_{XZ}+zE_{B(0X)Z} \\ E_{CZ} & 1 & -E_{AZ} & E_{YZ}+zE_{A(0Y)Z} \\ -E_{BZ} & E_{AZ} & 1 & E_{ZZ}+Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.11}$$

3.2 Rotational Axes

Similar to the error model for the linear axes, the error model for the rotational axes takes on the form:

$${}^{e}T_0^{\,B} = T_{OB}T_{SB}T_{EB}T_{BB} \tag{1.12}$$

where,

$$T_{OB} = \begin{bmatrix} 1 & 0 & 0 & O_{XB} \\ 0 & 1 & 0 & O_{YB} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.13}$$

$$T_{SB} = \begin{bmatrix} 1 & 0 & E_{B0B} & 0 \\ 0 & 1 & -E_{A0B} & 0 \\ -E_{B0B} & E_{A0B} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.14}$$

$$T_{EB} = \begin{bmatrix} 1 & -E_{CB} & E_{BB} & E_{XB} \\ E_{CB} & 1 & -E_{AB} & E_{YB} \\ -E_{BB} & E_{AB} & 1 & E_{ZB} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.15}$$

$$T_{BB} = \begin{bmatrix} c_B & -s_B & 0 & 0 \\ s_B & c_B & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.16}$$

Neglecting the squareness errors, the error model for the B-axis is [7]:

$${}^{e}T_0^{\,B} = \begin{bmatrix} -E_{CB}(B)s_B & -E_{CB}(B)c_B & -E_{BB}(B) & E_{XB}(B) \\ -E_{CB}(B)c_B & -E_{CB}(B)s_B & -E_{XB}(B) & E_{YB}(B) \\ -E_{BB}(B)c_B + E_{AB}(B)s_B & -E_{BB}(B)s_B + E_{AB}(B)c_B & 0 & E_{ZB}(B) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.17}$$

4. Error Measurement

Compensation of machine tools requires quantitative knowledge of the errors of the machine tool. In this work, a Leica Absolute Tracker AT960 MR was utilized to collect the three-dimensional coordinates of the retro reflector in order to calculate the errors of the linear as well as the rotational axes. The laser tracker uses a laser length measurement in addition to two angular measurements to calculate the position of the retro reflector in a three-dimensional spherical coordinate system.

4.2 Measuring Procedure

The measuring and calculation procedure is based on the method described by Zhang & Hu [8]. By measuring three points on the spindle and on the worktable at several axis-displacements evenly distributed throughout the travel of the axes, all position dependent geometric errors can be calculated.

Three reflector positions were chosen on the spindle of the DMU 50 (see Figure 2). The three positions on the spindle was measured sequentially through 11 points distributed evenly over the travel of the linear axes. The measuring procedure was carried out four times (see Figure 3).

Figure 2: The three reflector positions chosen on the spindle of the DMU 50

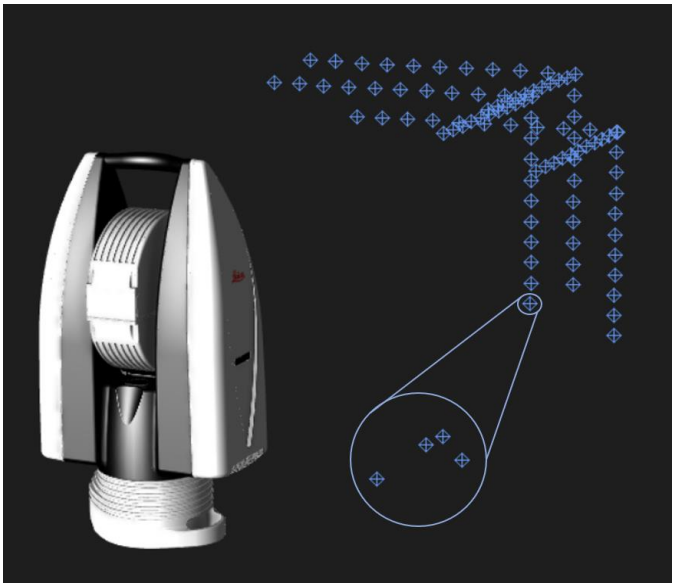


Figure 3: The four repetitions of measured points along each linear axis.

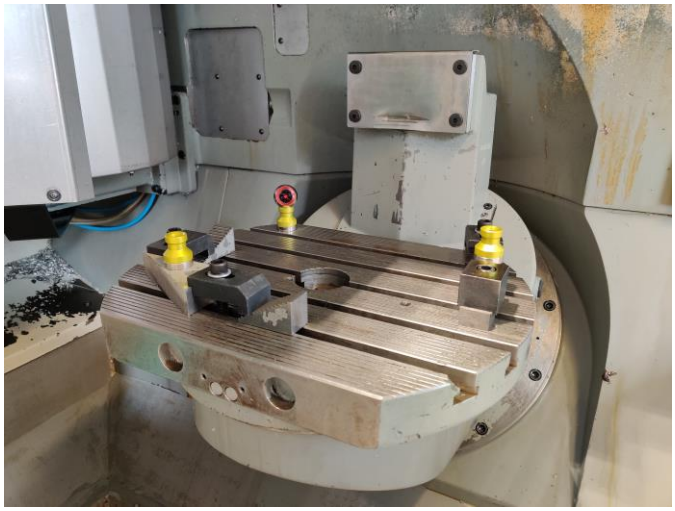


Figure 4: The three reflector positions chosen on the worktable of the DMU 50

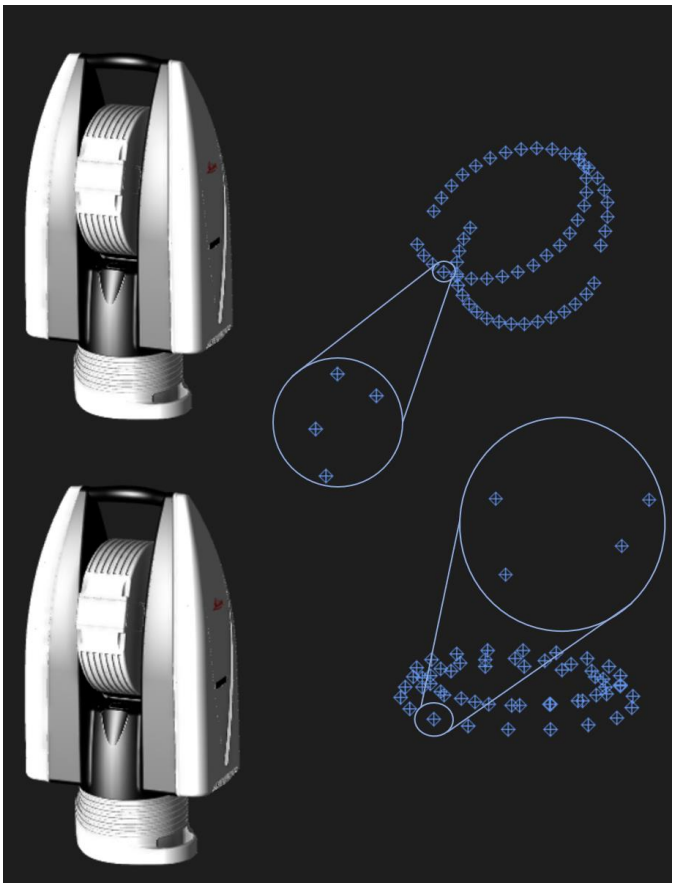


Figure 5: The four repetitions of measured points along each rotational axis.

The rotational axes were also measured a total of four times, with 19 and 18 points distributed along the travel of the B- and C-axis, respectively (see Figure 4 and Figure 5). The resulting errors are shown in Figure 6 and Figure 7.

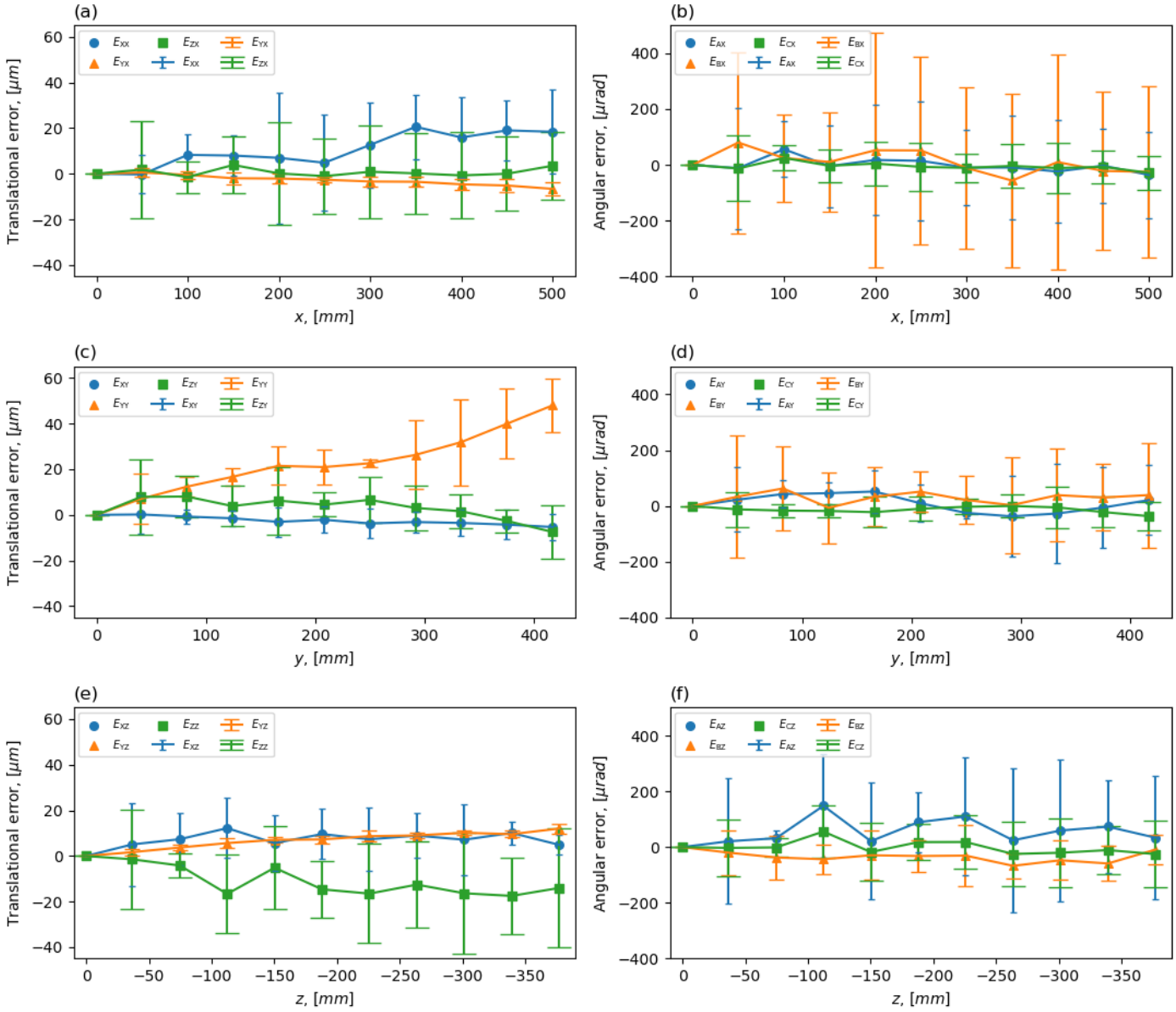


Figure 6: Errors of the linear axes. **(a)**: Translational errors of the X-axis. **(b)**: Angular errors of the X-axis. **(c)**: Translational errors of the Y-axis. **(d)**: Angular errors of the Y-axis. **(e)**: Translational errors of the Z-axis. **(f)**: Angular errors of the Z-axis.

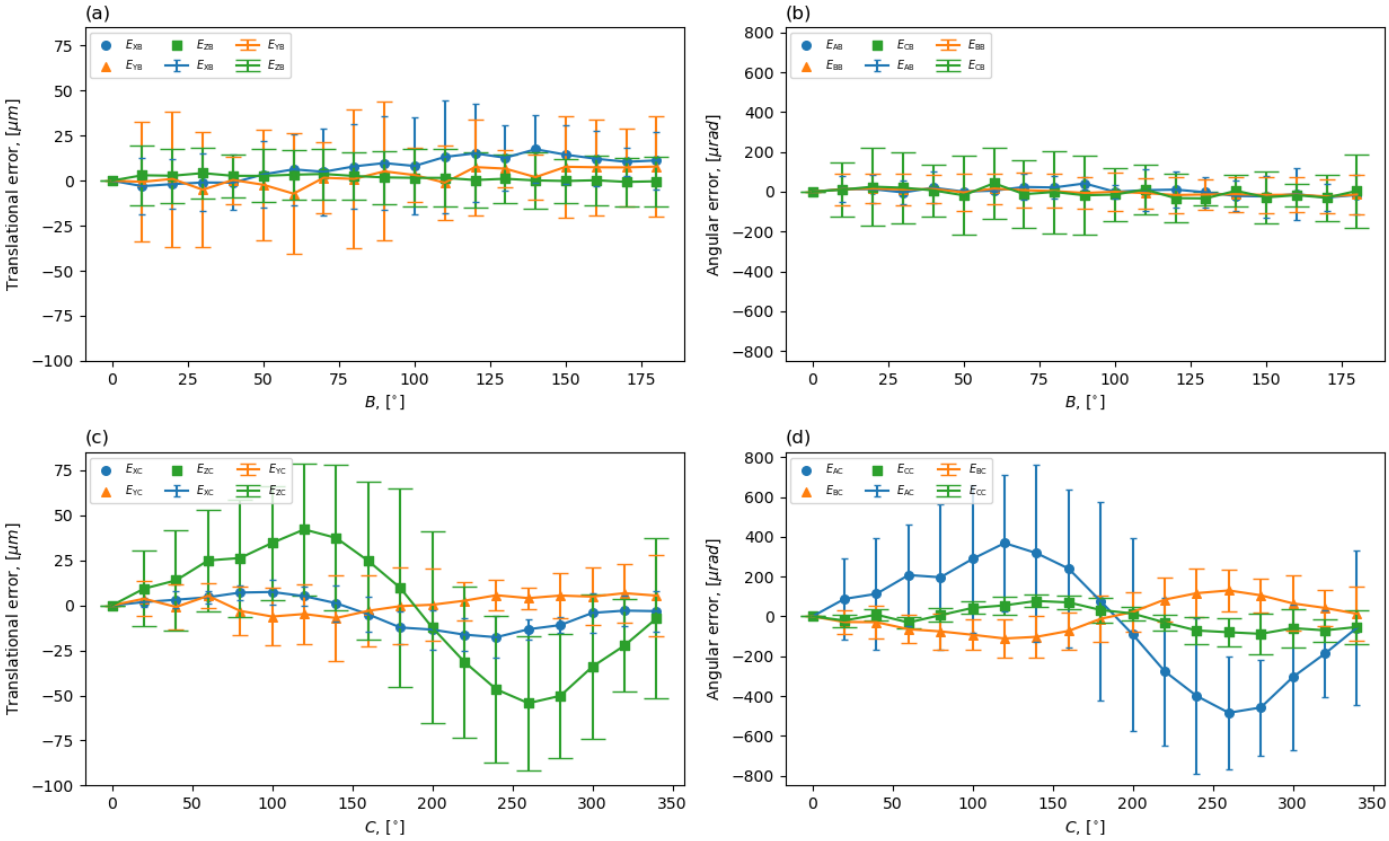


Figure 7: Errors of the rotational axes. **(a)**: Translational errors of the B-axis. **(b)**: Angular errors of the B-axis. **(c)**: Translational errors of the C-axis. **(d)**: Angular errors of the C-axis.

5. Error Compensation

Based on the ideal forward kinematics of the DMU 50, formulas for the TCP and the inclination of the *Z*-axis can be found [6]. In theory, by substituting the ideal machine kinematics with the error models for each axis, similar formulas of the forward kinematics including all error parameters could be found. However, in practice the expressions of the kinematics including all error parameters are complex and difficult to use in a compensation algorithm.

Similar to the work of Lee et. al [9], polynomials were fitted to the error data calculated from the measurement data. A reference straight line is also fitted to the calculated error data in order to determine the squareness errors of the linear axes (see Figure 8).

A compensation strategy was developed to compensate for the translational errors of the linear axes and the translational errors arising from both the translational and angular errors of the rotational axes. (The angular errors of the linear axes are neglected in this work.) Compensated NC-data is generated based on the ideal NC-data generated by the postprocessor. Based on the error models of all axes, the deviation between the position reached with ideal kinematics, and the position reached with the error models substituted into the ideal kinematics, is calculated. The compensation value (deviation with reversed sign) is then added to the nominal NC-code to generate the compensated NC-code.

Let P_{ideal} be the position of the tool based on the ideal machine kinematics when only the linear axes are moved:

$$P_{ideal,linear} = T_0^t P \tag{1.18}$$

The actual position reached by the tool P_{actual} can be calculated by including the error models of each linear axis:

$$P_{actual,linear} = {}^{e}T_0^{X}\,{}^{e}T_X^{Y}\,{}^{e}T_Y^{Z}\,P \tag{1.19}$$

In eq. 1.19, the angular error terms are neglected.

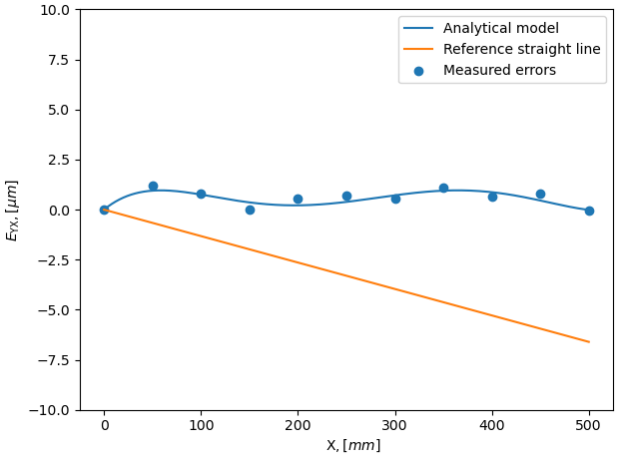


Figure 8: A fifth degree polynomial fitted to the calculated straightness error E_{YX} and the reference straight line fitted to the error parameter

The volumetric deviation e_{linear} between $P_{actual,linear}$ and $P_{ideal,linear}$ is found by simply subtracting the actual position by the ideal position. The compensation value c_{linear} is the deviation with reversed sign.

The compensation values for the rotational axes are found in a similar way. The desired position of the rotational axes is read from the NC-code generated by the post-processor. Considering a coordinate system located in the center of the machine table, the ideal position of a point on the machine table $P_{ideal,rotational}$ is then calculated using the transformation based on the forward kinematics. The actual position reached by the point on the machine table $P_{actual,rotational}$ is calculated by reading the errors from the polynomials fitted to the measured and calculated error data. The coordinate system is rotated by the amount of each angular error, then it is translated by the amount of the translational errors before it is rotated according to the desired position of the B- and C-axis. The deviation $e_{rotational}$ is calculated as the difference between the actual and the ideal position as for the linear axes.

In practice, the compensation values arising from the errors in the rotational axes $c_{rotational}$ is calculated first and added to the desired X-, Y- and Z-coordinates. The compensation values based on the errors of the linear axes are then calculated based on these coordinates. The compensation strategy is summarized in Figure 9**.**

The result of compensation of the linear axes is shown in Figure 10. The maximum error value was reduced by 49 % for E_{XX}, and by 78 % for E_{YY}.

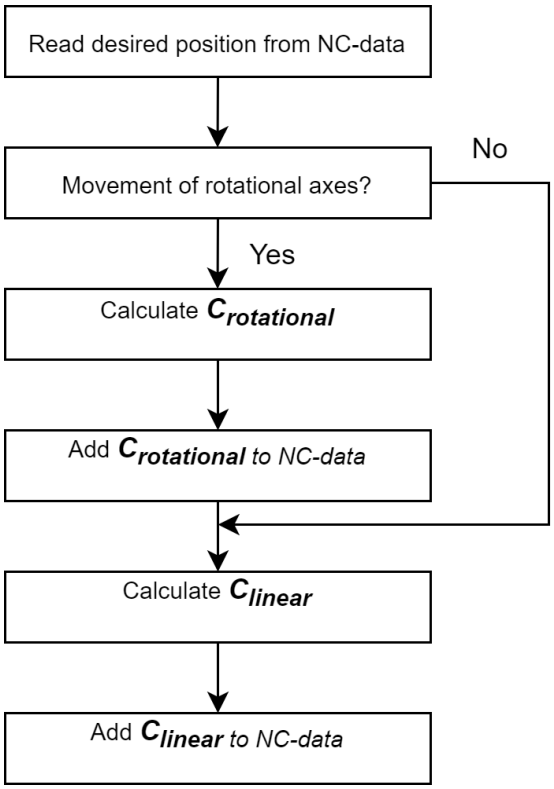


Figure 9: Compensation strategy

6. Measurement Uncertainty

When a number is assigned to any physical quantity as the result of a measurement, some quantitative indication of the quality of the measurement should be given. This indication serves as the foundation on which the users of the measurement evaluate its reliability [10].

A *Type A* evaluation of uncertainty was done on the measured and calculated errors displayed in Figure 6 and Figure 7. Based on four repetitions of measurements for each axis, the measurement uncertainty was estimated as the standard deviation of the measured data. The measurement uncertainty is indicated in the plots as error bars.

There are numerous sources of uncertainty in a measurement made by a laser tracker. One of these sources is the repeatability of the angular encoders of the laser tracker. The *Maximum Permissible Error* related to the angle performance, i.e. length measurements perpendicular to the direction of the laser beam, is *MPE* = ± 15 µm + 6 µm/m [11]. This is a relatively large uncertainty compared to that of the absolute distance measurement where the given *MPE* is ± 10 µm. However, measurements in the direction of the laser beam made without breaking the laser beam, will depend mostly on the interferometer uncertainty, with an *MPE* of 0.4 µm + 0.3 µm/m. Therefore a pattern can be recognized in the measured errors where some of the translational errors show much less variability than others. E_{YX} and E_{YZ} stands out as having much higher repeatability than the other error measurements. An explanation to this observation is proposed based on the assumed influence of the movement of the machine axis and the angular measurement of the laser tracker. Generally speaking, the linear positioning repeatability of a machine tool is lower than the repeatability of the position in the transverse directions. Depending on the position of the laser tracker in relation to the axis movements of the machine, the influence of the angle performance on the evaluation of the error varies. With the setup used in this work (see Figure 3), the errors in Y-direction have a low sensitivity to the angle performance of the laser tracker. Referring to Table 1, the influence of machine axis movement and that of the angular measurements of the laser tracker are both low. This coincides well with the observed variability of the measurement results.

Table 1: Influence of the machine tool and the laser tracker on the variability of the measured linear errors of the linear axes.

Influence on variability (X-axis)	E_{XX}	E_{YX}	E_{ZX}
Machine tool performance	High	Low	Low
Laser tracker performance	High	Low	High

Influence on variability (Y-axis)	E_{XY}	E_{YY}	E_{ZY}
Machine tool performance	Low	High	Low
Laser tracker performance	High	Low	High

Influence on variability (Z-axis)	E_{XZ}	E_{YZ}	E_{ZZ}
Machine tool performance	Low	Low	High
Laser tracker performance	High	Low	High

7. Conclusion

This paper presents a compensation strategy for all axes of a five-axis machine tool. Compensation values are calculated based on the deviation between the ideal forward kinematics of the machine tool and the forward kinematics including the errors of each axis. Error measurement is achieved using a conventional laser tracker using laser distance measurement and two angular measurements in a spherical coordinate system. The effectiveness of the compensation strategy on the translational errors of the linear axes is demonstrated using the same method of error measurement.

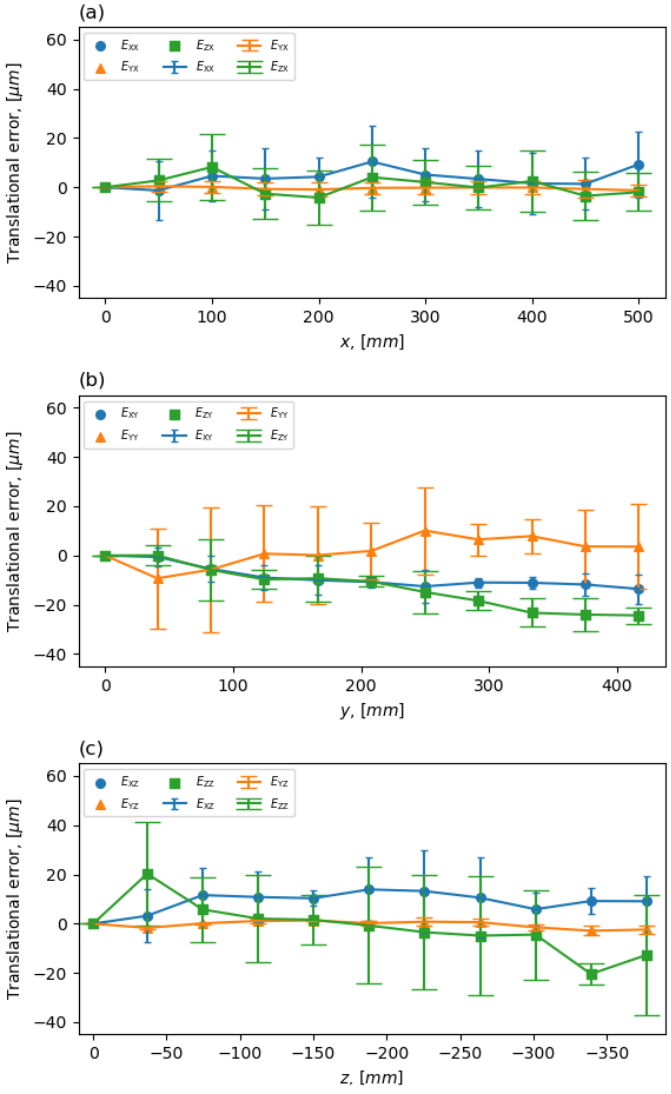


Figure 10: The result of compensation of the translational errors of the linear axes. **(a)**: Translational errors of the X-axis. **(b)**: Translational error parameters of the Y-axis. **(c)**: Translational errors of the Z-axis.

References

[1] S. Sartori and G. X. Zhang, "Geometric Error Measurement and Compensation of Machines," *CIRP Ann. - Manuf. Technol.*, vol. 44, no. 2, pp. 599–609, Jan. 1995, doi: 10.1016/S0007-8506(07)60507-1.

[2] H. Schwenke, M. Franke, J. Hannaford, and H. Kunzmann, "Error mapping of CMMs and machine tools by a single tracking interferometer," *CIRP Ann. - Manuf. Technol.*, vol. 54, no. 1, pp. 475–478, Jan. 2005, doi: 10.1016/S0007-8506(07)60148-6.

[3] D. C. Thompson and P. A. McKeown, "The Design of an Ultra-Precision CNC Measuring Machine," *CIRP Ann. - Manuf. Technol.*, vol. 38, no. 1, pp. 501–504, Jan. 1989, doi: 10.1016/S0007-8506(07)62755-3.

[4] "ISO/TR 16907:2015, Machine tools — Numerical compensation of geometric errors." International Standardization Organisation, 2011.

[5] "ISO 230-1:2012, Test code for machine tools — Part 1: Geometric accuracy of machines operating under no-load or quasi-static conditions," no. 2221. International Standardization Organisation, 1999.

[6] K. Sørby, "Inverse kinematics of five-axis machines near singular configurations," *Int. J. Mach. Tools Manuf.*, vol. 47, no. 2, pp. 299–306, Feb. 2007, doi: 10.1016/j.ijmachtools.2006.03.011.

[7] S. Xiang, M. Deng, H. Li, Z. Du, and J. Yang, "Volumetric error compensation model for five-axis machine tools considering effects of rotation tool center point," doi: 10.1007/s00170-019-03497-5.

[8] Z. Zhang and H. Hu, "A general strategy for geometric error identification of multi-axis machine tools based on point measurement," *Int. J. Adv. Manuf. Technol.*, vol. 69, no. 5–8, pp. 1483–1497, Jun. 2013, doi: 10.1007/s00170-013-5094-7.

[9] J. H. Lee, Y. Liu, and S. H. Yang, "Accuracy improvement of miniaturized machine tool: Geometric error modeling and compensation," *Int. J. Mach. Tools Manuf.*, vol. 46, no. 12–13, pp. 1508–1516, Oct. 2006, doi: 10.1016/j.ijmachtools.2005.09.004.

[10] GUM, "Evaluation of measurement data — Guide to the expression of uncertainty in measurement," *Int. Organ. Stand. Geneva ISBN*, vol. 50, no. September, p. 134, 2008, [Online]. Available: http://www.bipm.org/en/publications/guides/gum.html.

[11] Hexagon, "Leica Absolute Tracker AT960 ASME Specifications," no. mm, 2016, [Online]. Available: https://www.hexagonmi.com/en-GB/products/laser-tracker-systems/leica-absolute-tracker-at960.

Appendix B

Specialization Project, Fall 2020

This page is left intentionally blank.

NTNU
Faculty of Engineering
Department of Mechanical
and Industrial Engineering

TPK4540 - Manufacturing Technnology, Specilization Project

Improving the Accuracy of a Five-Axis Machine Tool by Laser Tracker Calibration

-

Mapping the Geometrical Error Parameters of the linear Axes

Simen Ellingsdalen

Supervisor: Prof. Knut Sørby

December 16, 2020

Abstract

A prerequisite for improving the positioning accuracy of any machine tool is understanding the build-up of the machine and how the different error parameters associated with each machine axis affect tool position and inclination. The first step then, is defining the kinematic model of the machine tool and determine the error parameters associated with each link in the kinematic chain resulting in the volumetric error. In this project, the kinematic model of the Deckel Maho DMU 50 eVolution is reviewed and the geometrical error parameters of the linear axes are defined. The mathematical error model is established based on homogeneous transformation matrices, and this is used to create a system of equations which can be solved to obtain an equation for each of the six error parameters of a linear axis. The working principle of the laser tracker is briefly presented, and the uncertainty of measurements made with the laser tracker is touched on. By measuring three points on the spindle of the machine tool traveling along each of the three linear axes, the error parameters are obtained through the mathematical error model. The measuring procedure is described in detail.

Contents

1 Introduction

1.1 Background and Motivation

A huge advantage with five-axis machine tools are their versatility. With two rotational axes in addition to the three linear axes, complex three-dimensional shapes can be machined with the tool perpendicular to the workpiece surface, ensuring optimal tool-workpiece interaction. The spindle axis may also be inclined to accomodate for instance side milling a chamfer using an end mill instead of a chamfering tool. Aside from machining complex shapes such as dies and fixtures, being able to machine several different faces of a part in one operation is a big advantage and can lead to both significant time savings and improved part accuracy. However, with two rotational axes and more complicated machine geometry and kinematics, comes more potential for geometrical errors.

Until the 1980s, there was no mathematical compensation of machine tools. Precision was ensured by expensive mechanical optimization by precision machining and manual craftsmanship. Sub-micron precision could be achieved by this method, but the time- and resource-investment was considerable. D. C. Thompson [1] states that: "The availability of modern computational tools makes the application of active and precalibrated error compensation an economical alternative to designing and building for absolute accuracy. Thus the mechanical accuracy of the machine need only be sufficient to allow error compensation to the desired level of accuracy..."

In the past decade, several of the large machine tool manufacturers have commercialized systems for automatically measuring and compensating for volumetric errors in the machine geometry [2]. This is usually done by measuring an artefact of known dimension, such as a sphere, with a digital touch trigger probe at several different rotation- and tilt angles of the rotational axes [3, 4]. On many CNC machine tools, a numerical compensation system for linear positioning errors is already in use. The cause of these errors are typically deviation from the specified pitch of the ball screw or a systematic error in a linear encoder. Some machines also have the capability to compensate for straightness deviations or squareness errors. Volumetric error compensation is a further development of these simpler compensations. The goal of volumetric error compensation is to tune the commanded x-, y- and z-coordinates along with the commanded rotations of the rotational axes to eliminate error in the tool center position and inclination [2].

There has been done a substantional amount of research into the field of geometric error mapping and mathematical compensation of CMMs and machine tools over the years. In 1995, Mahburbur, Lappalainen and Karjalainen [5] used a double ball bar to measure and identify the geometric errors of a CNC machine tool. They presented two different numerical methods for compensating for the measured geometric errors in the post processor: The Newton-Raphson iteration method and a method based on redefinition of task points. Comparison of the two methods showed that the Newton-Raphson method was more converging than than the other method, but they both successfully decreased the tool position error substantially after only one iteration.

Ten years later, Schwenke et. al. [6] used a self-developed tracking interferometer, of improved precision compared to commercial tracking interferometers, to measure the geometric errors of machine tools by the multilateration method. Monte-Carlo-Techniques was used in order to evaluate the uncertainties associated with the measurements. The measurement method was verified through experiments on a high accuracy CMM, a large horizontal arm CMM and a machine tool.

1.2 Problem Description

When machining large parts on five-axis machines, geometrical errors, especially those associated with the rotational axes, may manifest themselves as relatively large deviations in both position and inclination of geometrical features. Before tracking interferometers were invented, measuring and compensating for the geometrical errors of CNC machine tools and CMMs was a slow and tedious process wich could go on for several days. With the entry of high accuracy tracking laser interferometers, the measuring and compensation process has become significantly faster, and may be executed in a matter of a few hours. The commercial systems utilize specialized tracking interferometers made only for measuring the axis-motions of machine tools and CMMs. Although these systems offer unprecedented measuring precision, using a more versatile conventional laser tracker may be more appropriate in some cases. The question that needs answering then, is if sufficient error compensation and volumetric accuracy can be achieved using a versatile commercial laser tracker and simple measuring and compensation methods?

1.3 Project Scope

The goal of this work is to determine the error parameters of a specific machine tool, the Deckel Maho DMU 50 eVolution. The kinematics of the machine has been thoroughly defined and described by Knut Sørby in his article "Inverse kinematics of five-axis machines near singular configurations" [7]. This work will focus on acquiring reliable and accurate measurement data using the Leica Absolute Tracker AT960 laser tracker. The geometric errors that exist will be defined and evaluated, focusing firstly on the error parameters associated with the linear axes.

2 Theory

2.1 CNC Machine Tools and Their Geometry

There are many different types of machine tools in use in the industry. They all utilize a subtractive cutting process, whether it be turning, milling, grinding or a combination of these. According to the authors of "CIRP Encyclopedia of Production Engineering", "...it can be said that a machine tool is used to directly or indirectly make every modern man-made object. All manufacturing machines and objects are made and operate due to components made on machine tools including other machine tools. As such, machine tools are often called "mother machines."".

The simplest form of machine tool has only two axes. This type of machine tool is used to execute what is called a turning process. In this process the workpiece is clamped to the rotating element of the machine, the spindle. Outside of the spindle, the tool is mounted on a tool support which has linear movement perpendicular to the spindle axis. This axis is termed the X-axis. The Z-axis runs co-linear to the spindle axis. Conventional machines used for milling operations utilizes a minimum of three linear axes. In milling operations the workpiece is held stationary while the tool is clamped to the rotating spindle. The machine has two linear axes perpendicular to each other. The one with the longest travel is usually labeled as the X-axis. The Z-axis runs perpendicular to the XY-plane defined by the X- and Y-axes [8].

The two setups mentioned above are the most basic ways to set up the axes. However, many more

combinations of axes and spindles exist. In recent years, machines with five axes and machines combining turning and milling have become popular. Five-axis machines can have different setups, but most have three linear axes and two rotational axes. This enables the machine to tilt the tool in relation to the workpiece. This generally makes machining complex three dimensional shapes faster and more accurate [8]. The principal difference between how 3D shapes are machined on three- and five-axis machines is shown in figure 1. When making 3D-profiles on a three-axis machine, a milling tool with a spherical shape must be used, while on a five-axis machine, the tool axis can be indexed to be perpendicular with the workpiece surface at any given position. This allows for inclined surfaces to be machined with regular flat end mills for instance.

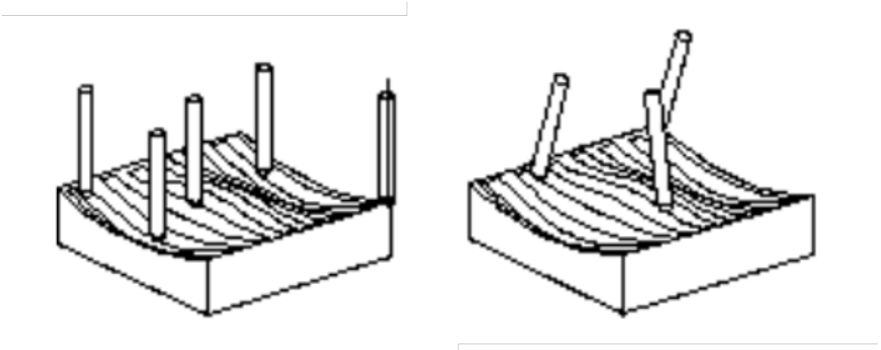


Figure 1: three-(left) and five-axis (right) machining of 3D shape

2.2 Deckel Maho DMU 50 eVolution

The Deckel Maho DMU 50 eVolution is a five-axis milling machine. The machine geometry is unusual in the way that the two rotational axes are not perpendicular to each other. Generally the naming convention on rotational axes is that the rotational axis A rotates about the linear X-axis, similarly B rotates about Y and C about Z [9]. The B axis in this machine does not rotate about the Y-axis, but about an axis inclined 45 degrees in relation to the Y- and Z-axes. The C and B axes intersect in a point a distance d from the fixed coordinate frame $x_0y_0z_0$ (see figure 2). With a working range of the C-axis of $(-\infty, \infty)$ and a B-axis range of $[0°, 180°]$ the machine is able to obtain a tool orientation perpendicular to every point on the surface of a half-sphere mounted on the work table.

2.3 Kinematics

In the following, the work done by Professor Knut Sørby [7] on defining and describing the kinematics of the Deckel Maho DMU 50 eVolution is summarized.

The kinematics of the machine describe the motions of the machine without considering masses or accelerations. The forward kinematics is a set of transformations used to obtain the cutter location from the axis variables X, Y, Z, B and C. Eight coordinate frames are used to obtain the kinematic equations for the machine:

1. $x_0y_0z_0$ is the base coordinate frame of the machine. It is located at the surface of the work table when the B- and C-axis are both in the 0° position.

2. $x_1y_1z_1$ is a translation of $x_0y_0z_0$ a distance d along the z_0-axis to the intersection point between the B- and C-axis.

3. $x_2y_2z_2$ is a rotation of $x_1y_1z_1$ at an angle $+45°$ about x_1 to align with the B-axis.

   These three coordinate frames are fixed, and do not move with the motion of the machine axes.

4. $x_3y_3z_3$ is a rotation of $x_2y_2z_2$ at an angle B around z_2.

5. $x_4y_4z_4$ rotates $x_3y_3z_3$ back $-45°$ about x_3.

6. $x_5y_5z_5$ translates $x_4y_4z_4$ a distance $-d$ back to the surface of the work table. This coordinate frame is always centered in the work table, also after the B-axis has been rotated.

7. $x_wy_wz_w$ rotates $x_5y_5z_5$ an angle $-C$ about z_5 to obtain the workpiece coordinate frame.

8. $x_ty_tz_t$ ia a coordinate frame fixed to the milling tool with the origin at the tip of the tool.

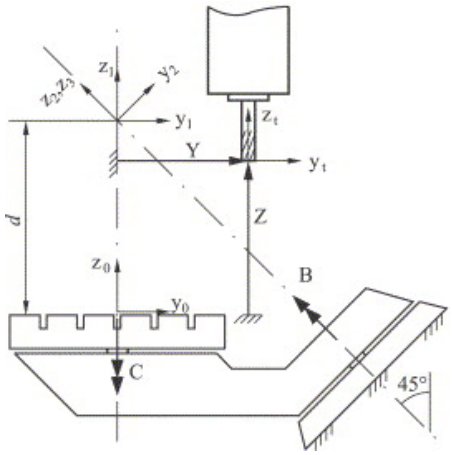


Figure 2: The geometry of the Deckel Maho DMU 50 eVolution [7]

By using homogeneous coordinates (representing a 3-vector (x,y,z) as a 4-vector (x,y,z,1)), translation can be expressed by matrix multiplication. The transformation from the milling tool's coordinate frame to the workpiece coordinate frame can be represented by the transformation matrices below. s_ϕ and c_ϕ is a shorthand notation for $sin\phi$ and $cos\phi$ respectively:

$$T_0^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$T_1^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{45^\circ} & -s_{45^\circ} & 0 \\ 0 & s_{45^\circ} & c_{45^\circ} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$T_2^3 = \begin{bmatrix} c_B & s_B & 0 & 0 \\ s_B & c_B & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$T_3^4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{45^\circ} & s_{45^\circ} & 0 \\ 0 & -s_{45^\circ} & c_{45^\circ} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$T_4^5 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$T_5^{\mathrm{w}} = \begin{bmatrix} c_C & s_C & 0 & 0 \\ -s_C & c_C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

$$T_0^{\mathrm{t}} = \begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

$$T_w^t = T_w^5 T_5^4 T_4^3 T_3^2 T_2^1 T_1^0 T_0^t \tag{8}$$

The resulting transformation matrix T_w^t contains four column vectors. Column one and two represents the inclination of the tool in the X- and Y-direction respectively. Column three represents the tool inclination in the Z-direction, and column four contains the tool center position (TCP). Together, column three and four make up the cutter location data commonly referred to as CL-data. The CL-data is a function of the machine axis variables X, Y, Z, B and C. For a given set of machine axis parameters, the CL-data can be found using the following equation:

$$\begin{bmatrix} i & x \\ j & y \\ k & z \\ 0 & 1 \end{bmatrix} = T_w^t \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{9}$$

From equation 9, the following equations for the forward kinematics are found:

$$i = \frac{1}{2}(\sqrt{2}c_C s_B - s_C c_B + s_C), \tag{10}$$

$$j = \frac{1}{2}(\sqrt{2}s_C s_B + c_C c_B - c_C), \tag{11}$$

$$k = \frac{1}{2} + \frac{1}{2}c_B, \tag{12}$$

$$x = \frac{\sqrt{2}}{2}[Xs_C + (Y + Z - d)c_C]s_B + \frac{1}{2}[-Y + Z - d - (Y + Z - d)c_B]s_C + Xc_C c_B, \tag{13}$$

$$y = \frac{\sqrt{2}}{2}[(Y + Z - d)s_C - Xc_C]s_B + Xs_C c_B + \frac{1}{2}[(Y + Z - d)c_C c_B + (Y - Z + d)c_C], \tag{14}$$

$$z = \frac{1}{2}[-\sqrt{2}Xs_B + (Y + Z - d)c_B - Y + Z + d]. \tag{15}$$

The forward kinematics are, as previously mentioned used to find the CL-data from the machine axis variables. However, we usually go the other way, wanting to find the axis variables corresponding to a certain cutter location. In other words we need the inverse kinematics of the machine tool. The inverse kinematics can be found analytically and exact, or numerically to a certain predefined precision. By using equation 9 and the relation

$$(T_w^t)^{-1} = T_t^w = T_t^0 T_0^1 T_1^2 T_2^3 T_3^4 T_4^5 T_5^w, \tag{16}$$

a set of equations are derived, which are used for finding the analytical solution to the inverse kinematics problem:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \\ a_{41} & a_{42} \end{bmatrix} = T_t^w \begin{bmatrix} i & x \\ j & y \\ k & z \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{17}$$

The B-axis inverse solution can be found directly from equation 12. For the other inverse solutions, eq 17 is used:

$$B = arccos(2k - 1), \tag{18}$$

$$C = arctan\Big[(1 - k)i + \sqrt{2(k - k^2)}j, \sqrt{2(k - k^2)}i + (k - 1)j\Big], \tag{19}$$

$$X = \Big[- y\sqrt{2(k - k^2)} - x + 2xk\Big]cosC + \Big[x\sqrt{2(k - k^2)} + 2yk - y\Big]sinC + (d - z)\sqrt{2(k - k^2)}, \tag{20}$$

$$Y = \Big[x\sqrt{2(k - k^2)} + yk\Big]cosC + \Big[y\sqrt{2(k - k^2)} - xk\Big]sinC - z + d - dk + zk, \tag{21}$$

$$Z = \Big[x\sqrt{2(k - k^2)} + yk - y\Big]cosC + \Big[y\sqrt{2(k - k^2)} - xk + x\Big]sinC + d - dk + zk. \tag{22}$$

Further details on the kinematics of the Deckel Maho DMU 50 eVolution is found in "Inverse kinematics of five-axis machines near singular configurations" by Professor Knut Sørby [7].

2.4 Geometrical Errors

A five-axis machine tool usually has three linear and two rotational axes, each with their own associated error parameters. Each linear axis has six error parameters - three translational error parameters ($\mathbf{E_{XX}}$, $\mathbf{E_{YX}}$ and $\mathbf{E_{ZX}}$) in addition to three rotational error parameters ($\mathbf{E_{AX}}$, $\mathbf{E_{BX}}$ and $\mathbf{E_{CX}}$) [10] (naming according to ISO 230-1, see figure 3). To completely describe the geometric error motions of the linear axes, the three squareness errors between each couple of axes, E_{C0X}, E_{A0Z} and E_{B0Z} are also considered [11] (figure 4). The linear error motion, $\mathbf{E_{XX}}$, is along the nominal motion direction of the X-axis, while the other two linear error parameters, $\mathbf{E_{YX}}$ and $\mathbf{E_{ZX}}$, are in the directions of the Y- and Z-axes, orthogonal to the X-axis. $\mathbf{E_{XX}}$, $\mathbf{E_{YY}}$ and $\mathbf{E_{ZZ}}$ are called the linear positioning error motions and are defined in ISO 230-1 as "unwanted motion along the direction of motion that results in the actual local position reached by the moving component at the functional point differing from the local commanded position along the direction of motion" [12]. If the position of the measured point on the moving component differs from the target position, we have a linear positioning deviation. This is the deviation in any given point along the motion of the component. "The straightness and angular errors are considered pure geometric errors, whereas the linear displacement (positioning) errors are a function of both geometry and the axis drive system characteristics" [12].

Key

1	X-axis commanded linear motion
E_{AX}	angular error motion around X-axis (Roll)
E_{BX}	angular error motion around Y-axis (Yaw)
E_{CX}	angular error motion around Z-axis (Pitch)
E_{XX}	linear positioning error motion of X-axis; positioning deviations of X-axis
E_{YX}	straightness error motion in Y-axis direction
E_{ZX}	straightness error motion in Z-axis direction

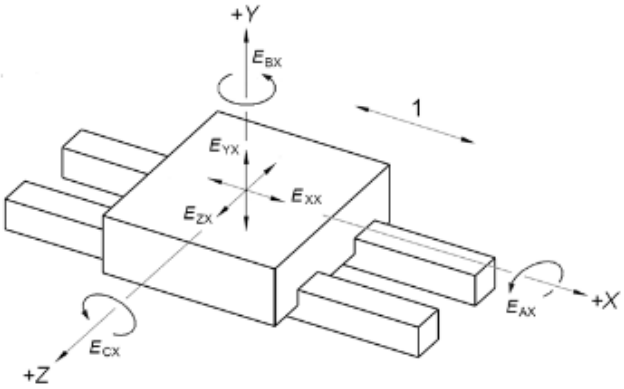
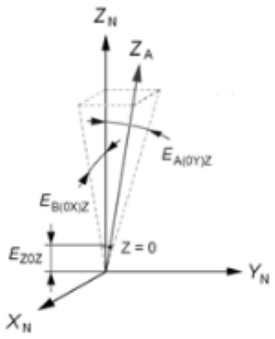


Figure 3: Error motions of a linear axis [13]



Key

X_N	nominal X-axis
Y_N	nominal Y-axis
Z_N	nominal Z-axis
Z_A	actual reference straight line of the component along Z-direction
E_{Z0Z}	zero position error of Z
$E_{A(0Y)Z}$	squareness error of Z to Y
$E_{B(0X)Z}$	squareness error of Z to X

NOTE In general, errors of the zero positions of linear axes (e.g. E_{Z0Z}) can be set to zero when checking geometric accuracy of a machine tool. Any change of E_{Z0Z} may cause errors in the workpiece.

Figure 4: Position and orientation errors of a linear axis [13]

Straightness deviations are measured against a reference straight line fitted to the measurement data. The reference line can be constructed in either of three ways according to ISO 230-1:

1. **Mean Minimum Zone Reference Straight Line**
   Arithmetic mean of two parallel straight lines in the straightness plane enclosing the measured straightness deviations and having the least separation.
2. **Least Squares Reference Straight Line**
   Straight line, where the sum of the squares of the measured straightness deviations is minimum.
3. **End Point Reference Straight Line**
   Straight line connecting the first and the last point of the measured straightness deviations.

Note that the reference straight line is computed from the measured deviations in two orthogonal planes withing the boundary of the measurement being performed.

Although this is the international standard, in the literature it is common to see the error parameters defined as zero in the initial position. For this reason the geometrical error parameters obtained in this project are all set to zero in the initial position of the axes.

2.5 The Error Model of a Linear Axis

As stated in the previous section, there are six geometrical errors associated with the motion of a linear axis, one for each degree of freedom. Considering the reference coordinate frame $oxyz$ and the spindle coordinate frame moving in the x-direction $o_x x_x y_x z_x$ The relationship between these coordinate frames can be described by the following equation:

$$\begin{bmatrix} \Delta_x(x) \\ \Delta_y(x) \\ \Delta_z(x) \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & -E_{CX}(x) & E_{BX}(x) & E_{XX}(x) \\ E_{CX}(x) & 0 & -E_{AX}(x) & E_{YX}(x) \\ -E_{BX}(x) & E_{AX}(x) & 0 & E_{ZX}(x) \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_x \\ y_x \\ z_x \\ 1 \end{bmatrix} \tag{23}$$

From equation 23 we get:

$$\begin{aligned} \Delta_x(x) &= -y_x E_{CX}(x) + z_x E_{BX}(x) + E_{XX} \\ \Delta_y(x) &= x_x E_{CX}(x) - z_x E_{AX}(x) + E_{YX} \\ \Delta_z(x) &= -x_x E_{BX}(x) + y_x E_{AX}(x) + E_{ZX} \end{aligned} \tag{24}$$

Which is the error model of the X-axis.

2.6 Laser Tracker

The laser tracker used in this project is a Leica Absolute Tracker AT960. It is a highly precise and versatile measuring instrument which is able to quickly and efficiently obtain the three dimensional coordinates of any point as long as there is a clear line of sight between the retro reflector and the tracker. It uses a combination of absolute distance measurement and interferometric distance measurement to obtain the distance between the tracker and the retro reflector to a very high degree of precision. Interferometric distance measurements works by analyzing a light wave sent from the interferometer and the returning wave reflected from a retro reflector. A superposition wave is made by these two waves, and every time the two light waves are in phase, the superposition wave peaks. By counting the number of peaks as the retro reflector is moved, the relative distance can be measured to a precision equal to half of the wavelength of the light wave. In the case of the laser light used in the Leica AT960, this equates to 0.32 μm. This type of measurement is also almost instantaneous. However a limitation of this measuring principle is that it only measures relative distance. There exists methods which measure absolute distance with high precision, however they require that the reflector is absolutely stationary as the measurement is executed and the distance calculated. The reason for this is that a "wobble measurement" is used to determine the lowest point on the wave. This requires that the light wave is stable for some amount of time, and is obviously a big disadvantage in dynamic measuring operations. The waveform does not change though, it only changes its relative position. Leica was the first manufacturer to combine the two measuring methods into what they call "AIFM", "Absolute Interferometer". This technology uses the unrivaled speed of the interferometer to monitor the relative position of the retro reflector as the distance is calculated using the absolute distance measurement technology. In combination, these two technologies provide an extremely precise length measurement while allowing the laser

beam to be broken without having to return the reflector to a point of known distance from the the interferometer [14].

With the distance to the retro reflector precisely measured by the AIFM, the spatial coordinate of the reflector can be determined by combining the length measurement with measurements from two angular encoders. The angular encoders measure two angles often referred to as azimuth and elevation. Combined, these three measurements form a spherical coordinate system, and the coordinates of the reflector can be determined by

$$\begin{aligned} x &= rsin\theta cos\phi \\ y &= rsin\theta sin\phi \\ z &= rcos\theta \end{aligned} \tag{25}$$

where (x, y, z) is the spatial coordinate, θ and ϕ are the azimuth and elevation angles and r is the distance from the tracker to the reflector [15].

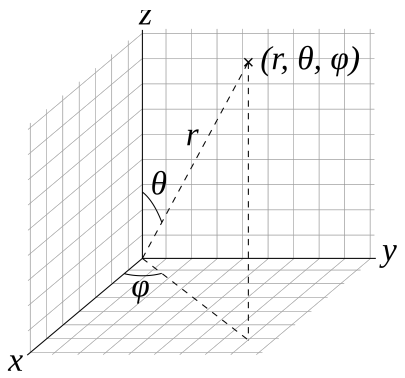


Figure 5: The spherical coordinate system of the laser tracker [16]

2.7 Multilateration

Multilateration is a method in which two or three dimensional coordinates are calculated from length measurements only. In simple terms: "If you know the distance of a point in space to at least three other known points, then it's coordinates can be determined" [17]. the length measurements are made from several fixed points to a single target position. This method is the same used in GPS (Global Positioning System) where the length measurements are made based on time of transmission (TOT) of signals from several satellites to the target. This measuring method determines the coordinates through the use of a non-linear least squares algorithm. The algorithm is used to fit all the measurement variables to the system variables which are:

Coordinates of the measuring stations: (X_i, Y_i, Z_i)

"Dead path length" for each measuring station: L_i

The coordinates of the target: (x_j, y_j, z_j)

The measured variables are the distances from the zero point to the target: l_{ij}

The deviation of the system variables to the following equation are minimized:

$$e_{ij} = \sqrt{(x_j - X_i)^2 + (y_j - Y_i)^2 + (z_j - Z_i)^2} - (l_{ij} - L_i), \tag{26}$$

where e_{ij} is the residual for the i-th measurement and the j-th target point [18].

With regards to determining geometrical errors in machine tools and coordinate measuring machines, multilateration is the method of choice when using a laser interferometer such as the Etalon LaserTRACER-NG. Unlike conventional laser trackers which employ distance measurement in the form of incremental interferometric measurments and absolute distance measurement in addition to two angular measurements (azimuth and elevation) to determine the position of the retro reflector, the LaserTRACER only makes use of the highly precise length measurement. This does not allow for the determination of the retro reflector's three dimensional position straight away, but as mentioned previously, by repeating the measurement of the distance from the LaserTRACER to the retro reflector a minimum of two times, the position can be calculated using multilateration.

2.8 Measurement Uncertainty

The yield of using the LaserTRACER and the multilateration method is unrivaled measuring precision. The LaserTRACER utilizes the accurate geometry of a highly precise ball with form errors below 50 nm as it's center of rotation and the inherent high resolution of laser interferometry to achieve a measurement uncertainty on the sub-micron level. Etalon states that the expanded measurement uncertainty of the LaserTRACER-NG is:

$$U = (0.2 + 0.3 \cdot L/1000)\mu m,$$

where L is the measurement length in mm [19].

The Leica Absolute Tracker AT960 has slightly higher interferometric meassurement uncertainty, but in the same order of magnitude:

$$MPE = (0.4 + 0.3 \cdot L/1000)\mu m,$$

where L is the measurement length in mm.

The Leica AT960 uses, in addition to the highly precise length measurement, two additional angle measurements (azimuth and elevation) to determine the retro reflector's coordinates in a spherical coordinate system. The uncertainty of the angle measurements are an order of magnitude larger than that of the interferometric length measurements with Leica's stated maximum permissible error being:

$$MPE = (15 + 6 \cdot L/1000)\mu m,$$

where L is the measurement length in mm [20].

Clearly, by determining the coordinates of the retro reflector using the spherical coordinate system with the measured angles from the angular encoders leads to increased measurement uncertainty.

3 Method

The following chapter will describe the method of obtaining the six error parameters associated with a linear axis by measuring three points on the spindle at different axis-displacements. The method is based on the work done by Zhenjiu Zhang and Hong Hu [15].

3.1 Extracting Geometrical Errors of a Linear Axis

Taking the X-axis of the Deckel Maho DMU 50 eVolution as an example, three points P, Q and K on the spindle head is measured making sure these points do not lie on a straight line to each other. A magnet holder is used to ensure the points stay in the same location relative to the spindle head throughout the data acquisition. The coordinates of points P, Q and K are (x_P, y_P, z_P), (x_Q, y_Q, z_Q) and (x_K, y_K, z_K) $oxyz$ is the reference coordinate frame and $o_x x_x y_x z_x$ is the coordinate frame of the spindle head moving along the X-axis. Taking point P as an example the six geometrical errors of the X-axis can be related to the volumetric errors of points on the spindle head. Let the homogeneous coordinates of o_x be $[x_o, y_o, z_o, 1]^T$ in the reference coordinate frame. If there exists no geometrical errors while the spindle head moves along the X-axis, the homogeneous coordinates of o_x $[x_{ot}, y_{ot}, z_{ot}, 1]^T$ at time t with a displacement of the X-axis equal to x can be obtained by performing the following translation:

$$\begin{bmatrix} x_{ot} \\ y_{ot} \\ z_{ot} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_o \\ y_o \\ z_o \\ 1 \end{bmatrix} \tag{27}$$

Then at this time, the coordinates of the point P in the reference coordinate frame can be found by:

$$\begin{bmatrix} x_{Pt} \\ y_{Pt} \\ z_{Pt} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_{ot} \\ 0 & 1 & 0 & y_{ot} \\ 0 & 0 & 1 & z_{ot} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{xP} \\ y_{xP} \\ z_{xP} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_o + x \\ 0 & 1 & 0 & y_o \\ 0 & 0 & 1 & z_o \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{xP} \\ y_{xP} \\ z_{xP} \\ 1 \end{bmatrix} \tag{28}$$

The real position of point P with spatial coordinates $(x'_{Pt}, y'_{Pt}, z'_{Pt})$ and homogeneous coordinates $[x'_{Pt}, y'_{Pt}, z'_{Pt}, 1]^T$ is then obtained by laser tracker measurement. The spatial coordinates are obtained using the laser tracker's spherical coordinate system. The position of P is influenced by the geometrical errors of the axis, and will deviate from the theoretical position. The three components of volumetric error $\Delta(x)$ at point P is $\Delta_x(P), \Delta_y(P)$ and $\Delta_z(P)$. The following relation can then be established:

$$\begin{bmatrix} \Delta_x(P) \\ \Delta_y(P) \\ \Delta_z(P) \\ 0 \end{bmatrix} = \begin{bmatrix} x'_{Pt} \\ y'_{Pt} \\ z'_{Pt} \\ 1 \end{bmatrix} - \begin{bmatrix} x_{Pt} \\ y_{Pt} \\ z_{Pt} \\ 1 \end{bmatrix} = \begin{bmatrix} x'_{Pt} \\ y'_{Pt} \\ z'_{Pt} \\ 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & x_o + x \\ 0 & 1 & 0 & y_o \\ 0 & 0 & 1 & z_o \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{xP} \\ y_{xP} \\ z_{xP} \\ 1 \end{bmatrix} = \begin{bmatrix} x'_{Pt} - x_{xP} - x_o - x \\ y'_{Pt} - y_{xP} - y_o \\ z'_{Pt} - z_{xP} - z_o \\ 0 \end{bmatrix} \tag{29}$$

If eq. (29) is substituted into the error model of the X-axis, we get:

$$\begin{aligned}
\Delta_x(P) &= x'_{Pt} - x_{xP} - x_o - x = -y_{xP}E_{CX}(x) + z_{xP}E_{BX}(x) + E_{XX} \\
\Delta_y(P) &= y'_{Pt} - y_{xP} - y_o = x_{xP}E_{CX}(x) - z_{xP}E_{AX}(x) + E_{YX} \\
\Delta_z(P) &= z'_{Pt} - z_{xP} - z_o = -x_{xP}E_{BX}(x) + y_{xP}E_{AX}(x) + E_{ZX}
\end{aligned} \tag{30}$$

By repeating the same procedure on points Q and K, we get:

$$\begin{aligned}
\Delta_x(Q) &= x'_{Qt} - x_{xQ} - x_o - x = -y_{xQ}E_{CX}(x) + z_{xQ}E_{BX}(x) + E_{XX} \\
\Delta_y(Q) &= y'_{Qt} - y_{xQ} - y_o = x_{xQ}E_{CX}(x) - z_{xQ}E_{AX}(x) + E_{YX} \\
\Delta_z(Q) &= z'_{Qt} - z_{xQ} - z_o = -x_{xQ}E_{BX}(x) + y_{xQ}E_{AX}(x) + E_{ZX}
\end{aligned}$$

$$\begin{aligned}
\Delta_x(K) &= x'_{Kt} - x_{xK} - x_o - x = -y_{xK}E_{CX}(x) + z_{xK}E_{BX}(x) + E_{XX} \\
\Delta_y(K) &= y'_{Kt} - y_{xK} - y_o = x_{xK}E_{CX}(x) - z_{xK}E_{AX}(x) + E_{YX} \\
\Delta_z(K) &= z'_{Kt} - z_{xK} - z_o = -x_{xK}E_{BX}(x) + y_{xK}E_{AX}(x) + E_{ZX}
\end{aligned}$$

Which can be rewritten as:

$$\begin{bmatrix}
1 & 0 & 0 & 0 & z_{xP} & -y_{xP} \\
0 & 1 & 0 & -z_{xP} & 0 & x_{xP} \\
0 & 0 & 1 & y_{xP} & -x_{xP} & 0 \\
1 & 0 & 0 & 0 & z_{xQ} & -y_{xQ} \\
0 & 1 & 0 & -z_{xQ} & 0 & x_{xQ} \\
0 & 0 & 1 & y_{xQ} & -x_{xQ} & 0 \\
1 & 0 & 0 & 0 & z_{xK} & -y_{xK} \\
0 & 1 & 0 & -z_{xK} & 0 & x_{xK} \\
0 & 0 & 1 & y_{xK} & -x_{xK} & 0
\end{bmatrix}
\begin{bmatrix}
E_{XX}(x) \\ E_{YX}(x) \\ E_{ZX}(x) \\ E_{AX}(x) \\ E_{BX}(x) \\ E_{CX}(x)
\end{bmatrix} =
\begin{bmatrix}
x'_{Pt} - x_{xP} - x_o - x \\ y'_{Pt} - y_{xP} - y_o \\ z'_{Pt} - z_{xP} - z_o \\
x'_{Qt} - x_{xQ} - x_o - x \\ y'_{Qt} - y_{xQ} - y_o \\ z'_{Qt} - z_{xQ} - z_o \\
x'_{Kt} - x_{xK} - x_o - x \\ y'_{Kt} - y_{xK} - y_o \\ z'_{Kt} - z_{xK} - z_o
\end{bmatrix} \tag{31}$$

As we see, there are nine equations, and only six unknown variables. As long as points P, Q and K don't lie on the same line, the rank of the coefficient matrix is six. This means that six equations from eq. (31) can be chosen to form a new system of equations with a unique solution. Zhang and Hu [15] explain thoroughy the selection process of the equations from (31) to form the new system:

$$\begin{bmatrix}
1 & 0 & 0 & 0 & z_{xP} & -y_{xP} \\
0 & 1 & 0 & -z_{xP} & 0 & x_{xP} \\
1 & 0 & 0 & 0 & z_{xQ} & -y_{xQ} \\
0 & 0 & 1 & y_{xQ} & -x_{xQ} & 0 \\
0 & 1 & 0 & -z_{xK} & 0 & x_{xK} \\
0 & 0 & 1 & y_{xK} & -x_{xK} & 0
\end{bmatrix}
\begin{bmatrix}
E_{XX}(x) \\ E_{YX}(x) \\ E_{ZX}(x) \\ E_{AX}(x) \\ E_{BX}(x) \\ E_{CX}(x)
\end{bmatrix} =
\begin{bmatrix}
x'_{Pt} - x_{xP} - x_o - x \\ y'_{Pt} - y_{xP} - y_o \\
x'_{Qt} - x_{xQ} - x_o - x \\ z'_{Qt} - z_{xQ} - z_o \\ y'_{Kt} - y_{xK} - y_o \\ z'_{Kt} - z_{xK} - z_o
\end{bmatrix} \tag{32}$$

Solving this system of equations yields the six error parameters as a function of the volumetric error of the points P, Q, K, the position of the origin o of the reference coordinate system and the displacement x:

$$
\begin{aligned}
\mathbf{E_{XX}}(\mathbf{x}) &= d_{xP} + \frac{M}{N}\left[y_{xP} - \frac{z_{xP(y_{xQ}-y_{xP})}}{z_{xQ}-z_{xP}}\right] - \frac{z_{xP}(d_{xQ}-d_{xP})}{z_{xQ}-z_{xP}} \\
\mathbf{E_{YX}}(\mathbf{x}) &= d_{yP} - \frac{M}{N}\left[x_{xP} - \frac{z_{xP(x_{xP}-x_{xK})}}{z_{xP}-z_{xK}}\right] - \frac{z_{xP}(d_{yP}-d_{yK})}{z_{xP}-z_{xK}} \\
\mathbf{E_{ZX}}(\mathbf{x}) &= d_{zQ} + \frac{M}{N}\left[\frac{y_{xQ}(x_{xK}-x_{xP})}{z_{xP}-z_{xK}} - \frac{x_{xQ}(y_{xP}-y_{xQ})}{z_{xQ}-z_{xP}}\right] - \frac{y_{xQ}(d_{yK}-d_{yP})}{z_{xP}-z_{xK}} + \frac{x_{xQ}(d_{xQ}-d_{xP})}{z_{xQ}-z_{xP}}
\end{aligned}
\tag{33}
$$

$$
\begin{aligned}
\mathbf{E_{AX}}(\mathbf{x}) &= \frac{d_{yK}-d_{yP}}{z_{xP}-z_{xK}} + \frac{x_{xP}-x_{xK}}{z_{xP}-z_{xK}} \cdot \frac{M}{N} \\
\mathbf{E_{BX}}(\mathbf{x}) &= \frac{d_{xQ}-d_{xP}}{z_{xQ}-z_{xP}} + \frac{y_{xQ}-y_{xP}}{z_{xQ}-z_{xP}} \cdot \frac{M}{N} \\
\mathbf{E_{CX}}(\mathbf{x}) &= \frac{M}{N},
\end{aligned}
$$

where

$$
\begin{aligned}
M &= (d_{xQ}-d_{xP})(x_{xK}-x_{xQ})(z_{xP}-z_{xK}) + [(d_{yP}-d_{yK})(y_{xK}-y_{xQ}) + (d_{zK}-d_{zQ})(z_{xP}-z_{xK})](z_{xQ}-z_{xP}) \\
N &= (x_{xK}-x_{xQ})(y_{xP}-y_{xQ})(z_{xP}-z_{xK}) - (x_{xK}-x_{xP})(y_{xK}-y_{xQ})(z_{xQ}-z_{xP}) \\
d_{xP} &= x'_{Pt} - x_{xP} - x_o - x \\
d_{yP} &= y'_{Pt} - y_{xP} - y_o \\
d_{xQ} &= x'_{Qt} - x_{xQ} - x_o - x \\
d_{zQ} &= z'_{Qt} - z_{xQ} - z_o \\
d_{yK} &= y'_{Kt} - y_{xK} - y_o \\
d_{zK} &= z'_{Kt} - z_{xK} - z_o
\end{aligned}
\tag{34}
$$

The same approach is taken to set up a the system of equations for the Y- and Z-axes. By using the same points on the spindle, the unreduced system of equations for the Y-axis is:

$$
\begin{bmatrix}
1 & 0 & 0 & 0 & z_{yP} & -y_{yP} \\
0 & 1 & 0 & -z_{yP} & 0 & x_{yP} \\
0 & 0 & 1 & y_{yP} & -x_{yP} & 0 \\
1 & 0 & 0 & 0 & z_{yQ} & -y_{yQ} \\
0 & 1 & 0 & -z_{yQ} & 0 & x_{yQ} \\
0 & 0 & 1 & y_{yQ} & -x_{yQ} & 0 \\
1 & 0 & 0 & 0 & z_{yK} & -y_{yK} \\
0 & 1 & 0 & -z_{yK} & 0 & x_{yK} \\
0 & 0 & 1 & y_{yK} & -x_{yK} & 0
\end{bmatrix}
\begin{bmatrix}
E_{XY}(y) \\ E_{YY}(y) \\ E_{ZY}(y) \\ E_{AY}(y) \\ E_{BY}(y) \\ E_{CY}(y)
\end{bmatrix}
=
\begin{bmatrix}
x'_{Pt} - x_{yP} - x_o \\
y'_{Pt} - y_{yP} - y_o - y \\
z'_{Pt} - z_{yP} - z_o \\
x'_{Qt} - x_{yQ} - x_o \\
y'_{Qt} - y_{yQ} - y_o - y \\
z'_{Qt} - z_{yQ} - z_o \\
x'_{Kt} - x_{yK} - x_o \\
y'_{Kt} - y_{yK} - y_o - y \\
z'_{Kt} - z_{yK} - z_o
\end{bmatrix}
\tag{35}
$$

And by using the same strategy as for the x-axis system of equations, the reduced system of equations is:

$$\begin{bmatrix} 0 & 1 & 0 & -z_{yP} & 0 & x_{yP} \\ 0 & 0 & 1 & y_{yP} & -x_{yP} & 0 \\ 0 & 1 & 0 & -z_{yQ} & 0 & x_{yQ} \\ 1 & 0 & 0 & 0 & z_{yK} & -y_{yK} \\ 0 & 0 & 1 & y_{yK} & -x_{yK} & 0 \\ 1 & 0 & 0 & 0 & z_{yP} & -y_{yP} \end{bmatrix} \begin{bmatrix} E_{XY}(y) \\ E_{YY}(y) \\ E_{ZY}(y) \\ E_{AY}(y) \\ E_{BY}(y) \\ E_{CY}(y) \end{bmatrix} = \begin{bmatrix} y'_{Pt} - y_{yP} - y_o - y \\ z'_{Pt} - z_{yP} - z_o \\ y'_{Qt} - y_{yQ} - y_o - y \\ x'_{Kt} - x_{yK} - x_o \\ z'_{Kt} - z_{yK} - z_o \\ x'_{Pt} - x_{yP} - x_o \end{bmatrix} \tag{36}$$

Similarly we get the reduced system of equations for the z-axis:

$$\begin{bmatrix} 0 & 0 & 1 & y_{zP} & -x_{zP} & 0 \\ 1 & 0 & 0 & 0 & z_{zQ} & -y_{zQ} \\ 0 & 0 & 1 & y_{zQ} & -x_{zQ} & 0 \\ 0 & 1 & 0 & -z_{zK} & 0 & x_{zK} \\ 1 & 0 & 0 & 0 & z_{zP} & -y_{zP} \\ 0 & 1 & 0 & -z_{zP} & 0 & x_{zP} \end{bmatrix} \begin{bmatrix} E_{XZ}(z) \\ E_{YZ}(z) \\ E_{ZZ}(z) \\ E_{AZ}(z) \\ E_{BZ}(z) \\ E_{CZ}(z) \end{bmatrix} = \begin{bmatrix} z'_{Pt} - z_{zP} - z_o - z \\ x'_{Qt} - x_{zQ} - x_o \\ z'_{Qt} - z_{zQ} - z_o - z \\ y'_{Kt} - y_{zK} - y_o \\ x'_{Pt} - x_{zP} - x_o \\ y'_{Pt} - y_{zP} - y_o \end{bmatrix} \tag{37}$$

The solution of the six error parameters for the z-axis on the extended form is shown in the appendix.

3.2 Measuring Procedure

Firstly, the base coordinate system is set up. This is done using Hexagon's metrology software - Inspire. The retro reflector is placed on the spindle, and a circle is constructed from its rotation (figure 6). The center point of the constructed circle is the origin, and the plane fitted to the measured points makes up the XY-plane. Then, points are measured along the travel of the X-axis and a line constructed and fitted to the measured points. Lastly, the Z-axis is constructed as a line perpendicular to the XY-plane. All subsequent measurements regarding the linear axes are in reference to this coordinate system.

Three points, P, Q and K, are then chosen on the spindle head of the machine, making sure these points are not co-linear and that they do not make the value of the denominator, N, zero (figure 7). The spatial coordinates of these three points are then measured with the Leica Absolute Tracker AT960 at 11 displacements evenly distributed along the travel of the axes (figure 8).

Figure 6: Setting up the coordinate system

Figure 7: Choosing the three points on the spindle

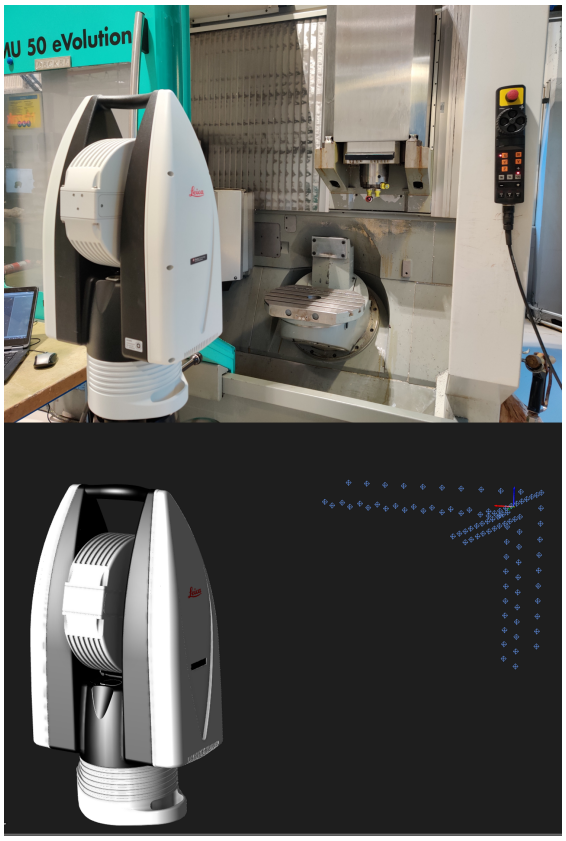


Figure 8: Measuring points along the displacement of the axes

4 Results

Figure 9, 10 and 11 show the calculated geometrical error parameters for the X-, Y- and Z-axis respectively. As can be seen in figure 9, the largest linear error of the X-axis is in the direction of travel, topping out at just over 23 μm at 500 mm traveled. The straightness errors E_{YX} and E_{ZX} are significantly lower. The largest angular error, E_{AX}, is approximately -250 μrad at 300 mm traveled. The remaining two angular errors are both within an absolute angle of 200 μrad for the entire travel of the axis.

Similar error magnitudes can be seen for the Y- and Z-axes, differing slightly with larger magnitude of the linear error E_{YY} and larger angular errors in the Z-axis travel in general and E_{AZ} in particular.

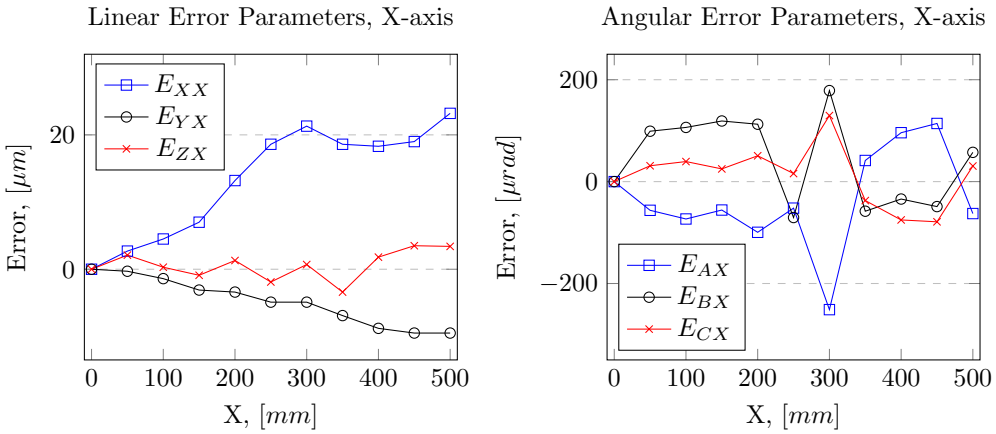


Figure 9: Plot of the linear (left) and angluar (right) error parameters of the x-axis

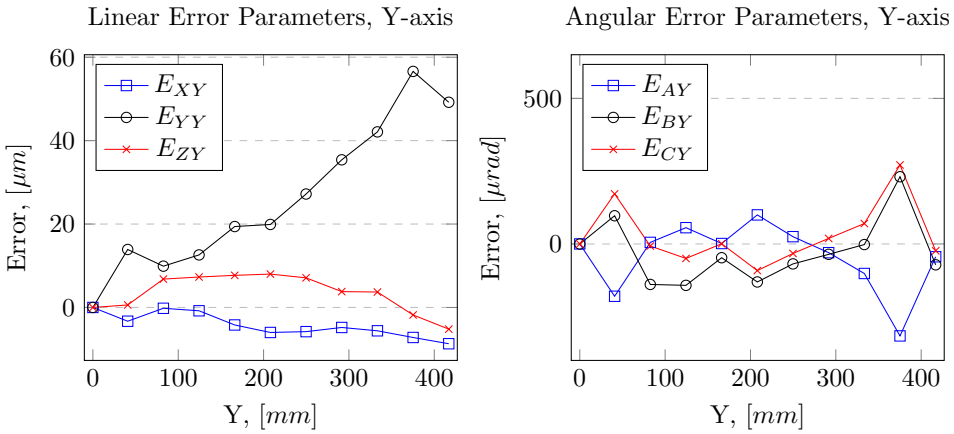


Figure 10: Plot of the linear (left) and angluar (right) error parameters of the y-axis

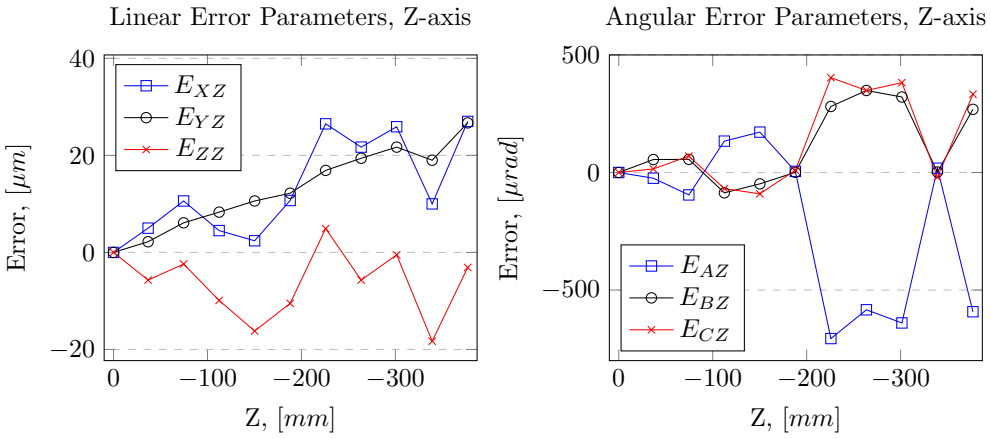


Figure 11: Plot of the linear (left) and angluar (right) error parameters of the z-axis

5 Discussion

The obtained error parameters have not been validated by another measuring method, and are the result of a single round of measurements. Although some of the error parameters are a bit higher than expected, they do seem probable, and within the order of magnitude that errors of machine tools may be in. The magnitude of the linear error parameter E_{YY} is certainly quite high at nearly 60 μm, but not completely improbable. The same goes for the peak values of the angular error parameters of all three axes, and E_{AZ} in particular with a peak of just over 700 μrad.

The coordinates of points P, Q and K are obtained by Laser Tracker measurements combining the highly precise length measurement with the less precise angular measurements of the two angular encoders in the tracker. A more comprehensive measuring method using the multilateration method would yield higher measuring precision by obtaining the coordinates of the points without using the less precise angular measurements of the laser tracker.

6 Future Work

Further validation of the measured error parameters through repeating the measurement process with a different placement of points P, Q and K could be useful. Measuring a larger set of points per length of axis travel might also be useful to better observe the evolution of the error parameters throughout the axis travel. Validating the measurement results obtained with the laser tracker by conventional measurement methods using a laser interferometer would be interesting.

In this work, the six geometrical error parameters associated with each linear axis was determined. In order to complete the error model of the three linear axes, the three squareness errors need to be determined. And to attain the full error model of the Deckel Maho DMU 50 eVolution, the error parameters associated with the two rotational axes must be determined and validated.

When all error parameters of the machine tool are determined and validated, the next step in the process of increasing the positioning accuracy of the machine is to apply a method of compensating

for the error motions of all axes. The overall goal of the error compensation will be to reduce both the spatial volumetric positioning error along with the spindle inclination error as much as possible.

7 Conclusion

A method for determining the geometrical error parameters of the linear axes of the Deckel Maho DMU 50 eVolution using the versatile and highly precise Leica Absolute Tracker AT960 laser tracker is used in this paper as the first step in improving the positioning accuracy of the machine tool. The method requires very little set up, and measurements can be made quickly and with a high degree of precision. All six geometrical error parameters of a linear axis can be determined by measuring three points on the spindle in a single set up. Although methods exist for determining the error parameters with less measurement uncertainty, this method may be sufficiently precise for certain machine tools and applications.

References

[1] D.C. Thompson and P.A. McKeown. "The Design of an Ultra-Precision CNC Measuring Machine". In: *CIRP Annals* 38.1 (1989), pp. 501–504. ISSN: 0007-8506. DOI: https://doi.org/10.1016/S0007-8506(07)62755-3. URL: http://www.sciencedirect.com/science/article/pii/S0007850607627553.

[2] Soichi Ibaraki et al. "Estimation of three-dimensional volumetric errors of machining centers by a tracking interferometer". In: *Precision Engineering* 39 (2015), pp. 179–186. ISSN: 0141-6359. DOI: https://doi.org/10.1016/j.precisioneng.2014.08.007. URL: http://www.sciencedirect.com/science/article/pii/S0141635914001457.

[3] OKUMA Corporation. *5-Axis Auto Tuning System*. 2016. URL: https://www.okuma.co.jp/english/onlyone/fivetuning/index.html.

[4] DMG MORI. *3D quickSET*. 2020. URL: https://no.dmgmori.com/products/digitization/integrated-digitization/production/technology-cycles/3d-quickset.

[5] R.M.D. Mahbubur et al. "Positioning accuracy improvement in five-axis milling by post processing". In: *International Journal of Machine Tools and Manufacture* 37.2 (1997), pp. 223–236. ISSN: 0890-6955. DOI: https://doi.org/10.1016/0890-6955(95)00091-7. URL: http://www.sciencedirect.com/science/article/pii/0890695595000917.

[6] H. Schwenke et al. "Error mapping of CMMs and machine tools by a single tracking interferometer". In: *CIRP Annals* 54.1 (2005), pp. 475–478. ISSN: 0007-8506. DOI: https://doi.org/10.1016/S0007-8506(07)60148-6. URL: http://www.sciencedirect.com/science/article/pii/S0007850607601486.

[7] Knut Sørby. "Inverse kinematics of five-axis machines near singular configurations". In: *International Journal of Machine Tools and Manufacture* 47.2 (2007), pp. 299–306. ISSN: 0890-6955. DOI: https://doi.org/10.1016/j.ijmachtools.2006.03.011. URL: http://www.sciencedirect.com/science/article/pii/S0890695506001027.

[8] Masahiko Mori, Adam Hansel, and Makoto Fujishima. "Machine Tool". In: *CIRP Encyclopedia of Production Engineering*. Ed. by Luc Laperrière and Gunther Reinhart. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 792–801. ISBN: 978-3-642-20617-7. DOI: 10.1007/978-3-642-20617-7_6533. URL: https://doi.org/10.1007/978-3-642-20617-7_6533.

[9] International Standardisation Organization. *ISO 841:1974, Industrial automation systems and integration — Numerical control of machines — Coordinate system and motion nomenclature*. 2001.

[10] Yl Lin and Y Shen. "Modelling of five-axis machine tool metrology models using the matrix summation approach". In: *The International Journal of Advanced Manufacturing Technology* 21.4 (2003), pp. 243–248.

[11] Martin Camboulives et al. "Calibration of a 3D working space by multilateration". In: *Precision Engineering* 44 (2016), pp. 163–170. ISSN: 0141-6359. DOI: https://doi.org/10.1016/j.precisioneng.2015.11.005. URL: http://www.sciencedirect.com/science/article/pii/S0141635915002214.

[12] International Standardisation Organization. *ISO 230-1:2012, Test code for machine tools — Part 1: Geometric accuracy of machines operating under no-load or quasi-static conditions*. 2012.

[13] International Standardisation Organization. *ISO/TR 16907, Machine tools - Numerical compensation of geometric erros*. 2015.

[14] Hexagon Metrology. Part of Hexagon. *lEICA ABSOLUTE INTERFEROMETER - A New Approach to Laser Tracker Absolute Distance Meters*. 2012. URL: https://www.hexagonmi.com/en-IN/solutions/technical-resources/technical-articles/the-leica-absolute-interferometer.

[15] Z. Zhang and H. Hu. "Measuring geometrical errors of linear axis of machine tools based on the laser tracker". In: *2011 IEEE International Conference on Robotics and Biomimetics*. 2011, pp. 2276–2281. DOI: 10.1109/ROBIO.2011.6181637.

[16] Wikipedia. *Spherical coordinate system*. URL: https://en.wikipedia.org/wiki/Spherical_coordinate_system.

[17] Etalon - part of Hexagon. *Multilateration*. 2020. URL: https://www.etalonproducts.com/en/technology/multilateration/.

[18] Jerzy Sładek et al. "Virtual coordinate measuring machine built using Laser Tracer system and spherical standard". In: *Metrology and Measurement Systems* 20.1 (2013), pp. 77–86.

[19] Etalon - part of Hexagon. *ETALON LASERTRACER-NG*. 2020. URL: https://www.etalonproducts.com/en/products/lasertracer/.

[20] Hexagon AB. *LEICA ABSOLUTE TRACKER AT960 ASME B89.4.19-2006 Specifications*. 2020. URL: https://www.hexagonmi.com/products/laser-tracker-systems/leica-absolute-tracker-at960.

Appendix

The solution of the geometrical error parameters related to the Z-axis on the extended form:

$\mathbf{E_{XZ}(z)} = \frac{1}{(y_{zP}-y_{zQ})((-x_{zK}+x_{zQ})z_{zP}+(x_{zK}-x_{zP})z_{zQ}+z_{zK}(x_{zP}-x_{zQ}))}(z_{zP}(y`_{Kt}-y`_{Pt}-y_{zK}+y_{zP})y_{zQ}^2+((-z_{zQ}-z_{zP})y_{zP}^2-(y`_{Kt}-y`_{Pt}-y_{zK})(z_{zQ}+z_{zP})y_{zP}-z_{zP}^3+(z_{zQ}+z`_{Pt}-z`_{Qt}+z_{zK})z_{zP}^2+(-z_{zK}z_{zQ}+(-z`_{Pt}+z`_{Qt})z_{zK}-x_{zP}^2+(x`_{Pt}-x`_{Qt}+2x_{zQ})x_{zP}+(x_o-x`_{Pt}-x_{zK})x_{zQ}-x_{zK}(x_o-x`_{Qt}))z_{zP}+(x_o-x`_{Pt}+x_{zP})((x_{zK}-x_{zP})z_{zQ}+z_{zK}(x_{zP}-x_{zQ})))y_{zQ}-(-y_{zP}^2z_{zQ}-z_{zQ}(y`_{Kt}-y`_{Pt}-y_{zK})y_{zP}-z_{zP}^2z_{zQ}+(z_{zQ}^2+(z`_{Pt}-z`_{Qt}+z_{zK})z_{zQ}-(x_{zK}-x_{zQ})(x_o-x`_{Qt}+x_{zQ}))z_{zP}-z_{zK}z_{zQ}^2+((-z`_{Pt}+z`_{Qt})z_{zK}+(x_{zK}-x_{zP})(x_o-x`_{Pt}+x_{zP}))z_{zQ}+z_{zK}(x_{zP}-x_{zQ})(x_o-x`_{Qt}+x_{zQ}))y_{zP})$

$\mathbf{E_{YZ}(z)} = \frac{1}{(y_{zp}-y_{zQ})((-z_{zQ}+z_{zK})x_{zp}+(-x_{zK}+x_{zQ})z_{zp}+x_{zK}z_{zQ}-x_{zQ}z_{zK})}(-x_{zp}^3z_{zK}+(x_{zK}z_{zp}+z_{zK}(x`_{Pt}-x`_{Qt}+2x_{zQ}))x_{zp}^2+(-z_{zp}^2z_{zK}+((-x`_{Pt}+x`_{Qt}-2x_{zQ})x_{zK}+2z_{zK}(z_{zQ}+z`_{Pt}/2-z`_{Qt}/2))z_{zp}+(-y_{zp}^2+(-y_o+y`_{Pt}+y_{zQ})y_{zp}-x_{zQ}^2+(-x`_{Pt}+x`_{Qt})x_{zQ}+(y_o-y`_{Pt})y_{zQ}-z_{zQ}(z_{zQ}+z`_{Pt}-z`_{Qt}))z_{zK}+z_{zQ}(y_{zp}-y_{zQ})(y_o-y`_{Kt}+y_{zK}))x_{zp}+x_{zK}z_{zp}^3-2x_{zK}(z_{zQ}+z`_{Pt}/2-z`_{Qt}/2)z_{zp}^2+((y_{zp}^2+(y_o-y`_{Pt}-y_{zQ})y_{zp}+x_{zQ}^2+(x`_{Pt}-x`_{Qt})x_{zQ}+(-y_o+y`_{Pt})y_{zQ}+z_{zQ}(z_{zQ}+z`_{Pt}-z`_{Qt}))x_{zK}-x_{zQ}(y_{zp}-y_{zQ})(y_o-y`_{Kt}+y_{zK}))z_{zp}-(y_{zp}-y_{zQ})(x_{zK}z_{zQ}-x_{zQ}z_{zK})(y_o-y`_{Pt}+y_{zp}))$

$\mathbf{E_{ZZ}(z)} = \frac{1}{(y_{zP}-y_{zQ})((z_{zQ}-z_{zK})x_{zP}+(-z_{zP}+z_{zK})x_{zQ}+x_{zK}(z_{zP}-z_{zQ}))}(x_{zP}(y`_{Kt}-y`_{Pt}-y_{zK}+y_{zP})y_{zQ}^2+((-x_{zP}-x_{zQ})y_{zP}^2-(x_{zP}+x_{zQ})(y`_{Kt}-y`_{Pt}-y_{zK})y_{zP}-x_{zP}^3+(x`_{Pt}-x`_{Qt}+x_{zK}+x_{zQ})x_{zP}^2+(-x_{zK}x_{zQ}+(-x`_{Pt}+x`_{Qt})x_{zK}-z_{zP}^2+(2z_{zQ}+z`_{Pt}-z`_{Qt})z_{zP}+(z-z`_{Pt}+z_o-z_{zK})z_{zQ}-z_{zK}(z-z`_{Qt}+z_o))x_{zP}+(z+z_{zP}-z`_{Pt}+z_o)((-z_{zP}+z_{zK})x_{zQ}+x_{zK}(z_{zP}-z_{zQ})))y_{zQ}-(-x_{zQ}y_{zP}^2-(y`_{Kt}-y`_{Pt}-y_{zK})x_{zQ}y_{zP}-x_{zP}^2x_{zQ}+(x_{zQ}^2+(x`_{Pt}-x`_{Qt}+x_{zK})x_{zQ}+(z_{zQ}-z_{zK})(z_{zQ}+z-z`_{Qt}+z_o))x_{zP}-x_{zK}x_{zQ}^2+((-x`_{Pt}+x`_{Qt})x_{zK}-(z_{zP}-z_{zK})(z+z_{zP}-z`_{Pt}+z_o))x_{zQ}+x_{zK}(z_{zP}-z_{zQ})(z_{zQ}+z-z`_{Qt}+z_o))y_{zP})$

$\mathbf{E_{AZ}(z)} = \frac{1}{(y_{zP}-y_{zQ})((z_{zQ}-z_{zK})x_{zP}+(-z_{zP}+z_{zK})x_{zQ}+x_{zK}(z_{zP}-z_{zQ}))}(x_{zP}^3+(-x_{zK}-x`_{Pt}+x`_{Qt}-2x_{zQ})x_{zP}^2+(x_{zQ}^2+(2x_{zK}+x`_{Pt}-x`_{Qt})x_{zQ}+(x`_{Pt}-x`_{Qt})x_{zK}+y_{zP}^2+(y`_{Kt}-y`_{Pt}-y_{zK}-y_{zQ})y_{zP}+(-y`_{Kt}+y`_{Pt}+y_{zK})y_{zQ}-(z_{zP}-z_{zQ})(-z_{zP}+z`_{Pt}-z`_{Qt}+z_{zQ}))x_{zP}-x_{zK}x_{zQ}^2+((-x`_{Pt}+x`_{Qt})x_{zK}-(y_{zP}-y_{zQ})(y`_{Kt}-y`_{Pt}-y_{zK}+y_{zP}))x_{zQ}+x_{zK}(z_{zP}-z_{zQ})(-z_{zP}+z`_{Pt}-z`_{Qt}+z_{zQ}))$

$\mathbf{E_{BZ}(z)} = \frac{1}{(z_{zQ}-z_{zK})x_{zP}+(x_{zK}-x_{zQ})z_{zP}-x_{zK}z_{zQ}+x_{zQ}z_{zK}}(x_{zP}^2+(-x_{zK}-x`_{Pt}+x`_{Qt}-x_{zQ})x_{zP}+z_{zP}^2+(-z`_{Pt}+z`_{Qt}-z_{zK}-z_{zQ})z_{zP}+(x`_{Pt}-x`_{Qt}+x_{zQ})x_{zK}+(z_{zQ}+z`_{Pt}-z`_{Qt})z_{zK}+(y_{zP}-y_{zQ})(y`_{Kt}-y`_{Pt}-y_{zK}+y_{zP}))$

$\mathbf{E_{CZ}(z)} = \frac{1}{(y_{zP}-y_{zQ})((x_{zK}-x_{zQ})z_{zP}+(-x_{zK}+x_{zP})z_{zQ}-z_{zK}(x_{zP}-x_{zQ}))}(z_{zP}^3+(-z`_{Pt}+z`_{Qt}-z_{zK}-2z_{zQ})z_{zP}^2+(z_{zQ}^2+(z`_{Pt}-z`_{Qt}+2z_{zK})z_{zQ}+(z`_{Pt}-z`_{Qt})z_{zK}+y_{zP}^2+(y`_{Kt}-y`_{Pt}-y_{zK}-y_{zQ})y_{zP}+(-y`_{Kt}+y`_{Pt}+y_{zK})y_{zQ}-(x_{zP}-x_{zQ})(-x_{zP}+x`_{Pt}-x`_{Qt}+x_{zQ}))z_{zP}-z_{zK}z_{zQ}^2+((-z`_{Pt}+z`_{Qt})z_{zK}-(y_{zP}-y_{zQ})(y`_{Kt}-y`_{Pt}-y_{zK}+y_{zP}))z_{zQ}+z_{zK}(x_{zP}-x_{zQ})(-x_{zP}+x`_{Pt}-x`_{Qt}+x_{zQ}))$

Appendix C

Python Scripts - Linear Axes

C.1 Python Script For Calculating Errors of the Linear Axes

The script for calculating the errors of the X-axis is included as an example. The scripts used for the Y- and Z-axes are in essence the same. Note that "\" is a line continuation character.

```
# ============================================================
# Error Parameters Associated with the X-axis
# ============================================================

import numpy as np

# Defining the origin

Origin = np.array([0, 0, 0])
x_o, y_o, z_o = Origin[0], Origin[1], Origin[2]

# Importing the files containing the coordinates of the measured points,
# and arranging the data appropriately

with open('Round 1.csv') as file:
        d0 = file.read()
        d1 = d0.split('\n')
        d1.pop(99)
        d2 = [i.split(',') for i in d1]

        K_x = np.array([d2[0][1], d2[0][2], d2[0][3], 1])
        K_act_t = np.zeros((11, 4))
        for i in range(11):
            K_act_t[i][0:3] = d2[i][1:]
            K_act_t[i][3] = 1

        P_x = np.array([d2[11][1], d2[11][2], d2[11][3], 1])
        P_act_t = np.zeros((11, 4))
        for i in range(11, 22):
            P_act_t[i-11][0:3] = d2[i][1:]
            P_act_t[i-11][3] = 1

        Q_x = np.array([d2[22][1], d2[22][2], d2[22][3], 1])
        Q_act_t = np.zeros((11, 4))
        for i in range(22, 33):
            Q_act_t[i-22][0:3] = d2[i][1:]
            Q_act_t[i-22][3] = 1

# Defining the coordinates of the points P, Q and K in the reference frame

```

```
x_xP, y_xP, z_xP = float(P_x[0]), float(P_x[1]), float(P_x[2])
x_xQ, y_xQ, z_xQ = float(Q_x[0]), float(Q_x[1]), float(Q_x[2])
x_xK, y_xK, z_xK = float(K_x[0]), float(K_x[1]), float(K_x[2])

# Defining the nominal displacements along the axis of movement

x = np.array(np.linspace(0, 500, num = 11))

# Deviding the coordinates into separate arrays

x_actPt, y_actPt, z_actPt = P_act_t[:,0], P_act_t[:,1], P_act_t[:,2]
x_actQt, y_actQt, z_actQt = Q_act_t[:,0], Q_act_t[:,1], Q_act_t[:,2]
x_actKt, y_actKt, z_actKt = K_act_t[:,0], K_act_t[:,1], K_act_t[:,2]

# Defining the geometrical error parameter equations

def delta_x(x_o, x_actPt, x_actQt, x_xK, x_xP, x_xQ, y_actKt,y_actPt, y_xK, \
            y_xP, y_xQ, z_actKt, z_actQt, z_xK, z_xP, z_xQ):
    E_XX = (-(x_xK*y_xK*z_xQ - x_xK*y_xQ*z_xK + x_xK*y_xQ*z_xP - \
x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xQ + x_xQ*y_xQ*z_xK - \
x_xQ*y_xQ*z_xP)*(x_actPt - x_xP - x_o - x)/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - \
x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + \
x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - \
x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + \
x_xQ*y_xQ*z_xP) + (y_xK - y_xQ)*(y_xP*z_xQ - y_xQ*z_xP)*(y_actPt - y_xP - y_o)\
/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + \
x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + \
x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - \
x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) + (y_xK - y_xP)*\
(y_xP*z_xQ - y_xQ*z_xP)*(y_actQt - y_xQ - y_o)/(x_xK*y_xK*z_xP - \
x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - \
2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + \
x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - \
x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) - (y_xK - y_xQ)*(y_xP*z_xQ - y_xQ*z_xP)*\
(y_actKt - y_xK - y_o)/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + \
x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - \
x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + \
x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) - \
(z_xK - z_xP)*(y_xP*z_xQ - y_xQ*z_xP)*(z_actKt - z_xK - z_o)/\
(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + \
x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + \
x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - \
x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP)) * 1e3
    return E_XX

def delta_y(x_actPt, x_actQt, x_xK, x_xP, x_xQ, y_o, y_actKt, y_actPt, y_xK, \
            y_xP, y_xQ, z_actKt, z_actQt, z_xK, z_xP, z_xQ):
    E_YX = ((x_xK - x_xQ)*(x_xK*z_xP - x_xP*z_xK)*(x_actPt - x_xP - x_o - x)/\
(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + \
x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + \
x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - \
x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) + (x_xK*y_xK*z_xP - \
x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - \
2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + \
x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - \
x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP)*(y_actPt - y_xP - y_o)/\
(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + \
x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + \
x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - \
x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) - (x_xK - x_xQ)*\
```

```
(x_xK*z_xP - x_xP*z_xK)*(x_actQt - x_xQ - x_o - x)/(x_xK*y_xK*z_xP - \
x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - \
2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + \
x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - \
x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) - (-z_xQ + z_xP)*(x_xK*z_xP - x_xP*z_xK)*\
(z_actQt - z_xQ - z_o)/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + \
x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - \
x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + \
x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) + \
(x_xK*y_xP*z_xP - x_xK*y_xQ*z_xP - x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + \
x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ - x_xQ*y_xP*z_xP + x_xQ*y_xQ*z_xP)*\
(y_actKt - y_xK - y_o)/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + \
x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - \
x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + \
x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) + \
(-z_xQ + z_xP)*(x_xK*z_xP - x_xP*z_xK)*(z_actKt - z_xK - z_o)/(x_xK*y_xK*z_xP \
- x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - \
2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + \
x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - \
x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP)) * 1e3
    return E_YX

def delta_z(x_actPt, x_actQt, x_xK, x_xP, x_xQ, y_actKt, y_actPt, y_xK, y_xP, \
            y_xQ, z_o, z_actKt, z_actQt, z_xK, z_xP, z_xQ):
    E_ZX = (-(x_xK**2*y_xQ - x_xK*x_xP*y_xQ - x_xK*x_xQ*y_xK + x_xP*x_xQ*y_xK)\
*(x_actPt - x_xP - x_o - x)/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK \
+ x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - \
x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + \
x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) + \
(x_xK*y_xP*y_xQ - x_xK*y_xQ**2 - x_xQ*y_xK*y_xP + x_xQ*y_xK*y_xQ)*\
(y_actPt - y_xP - y_o)/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + \
x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - \
x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + \
x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) + \
(x_xK**2*y_xQ - x_xK*x_xP*y_xQ - x_xK*x_xQ*y_xK + x_xP*x_xQ*y_xK)*\
(x_actQt - x_xQ - x_o - x)/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK +\
x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - \
x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + \
x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) + \
(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + \
x_xK*y_xQ*z_xK - x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xQ + \
x_xP*y_xQ*z_xK - x_xP*y_xQ*z_xP - x_xQ*y_xP*z_xK + x_xQ*y_xP*z_xP)*\
(z_actQt - z_xQ - z_o)/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + \
x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - \
x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + \
x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) - \
(x_xK*y_xP*y_xQ - x_xK*y_xQ**2 - x_xQ*y_xK*y_xP + x_xQ*y_xK*y_xQ)*\
(y_actKt - y_xK - y_o)/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + \
x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - \
x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + \
x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) - \
(x_xK*y_xQ*z_xP - x_xK*y_xQ*z_xQ - x_xP*y_xQ*z_xP + x_xP*y_xQ*z_xQ - \
x_xQ*y_xP*z_xK + x_xQ*y_xP*z_xP + x_xQ*y_xQ*z_xK - x_xQ*y_xQ*z_xP)*\
(z_actKt - z_xK - z_o)/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + \
x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - \
x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + \
x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP)) * 1e3
    return E_ZX

def epsilon_x(x_actPt, x_actQt, x_xK, x_xP, x_xQ, y_actKt, y_actPt, y_xK, \
              y_xP, y_xQ, z_actKt, z_actQt, z_xK, z_xP, z_xQ):
    E_AX = ((x_xK - x_xQ)*(x_xK - x_xP)*(x_actPt - x_xP - x_o - x)/\
(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + \
x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + \
x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - \
```

```
x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) - (x_xK*y_xP - x_xK*y_xQ - \
x_xQ*y_xP + x_xQ*y_xQ)*(y_actPt - y_xP - y_o)/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ\
- x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + \
x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - \
x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + \
x_xQ*y_xQ*z_xP) - (x_xK - x_xQ)*(x_xK - x_xP)*(x_actQt - x_xQ - x_o - x)/\
(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + \
x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + \
x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - \
x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) - (-z_xQ + z_xP)*\
(x_xK - x_xP)*(z_actQt - z_xQ - z_o)/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - \
x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + \
x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - \
x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + \
x_xQ*y_xQ*z_xP) + (x_xK*y_xP - x_xK*y_xQ - x_xQ*y_xP + x_xQ*y_xQ)*\
(y_actKt - y_xK - y_o)/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + \
x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - \
x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + \
x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) + \
(-z_xQ + z_xP)*(x_xK - x_xP)*(z_actKt - z_xK - z_o)/(x_xK*y_xK*z_xP - \
x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - \
2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + \
x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - \
x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP)) * 1e6
    return E_AX

def epsilon_y(x_actPt, x_actQt, x_xK, x_xP, x_xQ, y_actKt, y_actPt, y_xK, \
              y_xP, y_xQ, z_actKt, z_actQt, z_xK, z_xP, z_xQ):
    E_BX = ((x_xK*y_xK - x_xK*y_xQ - x_xP*y_xK + x_xP*y_xQ)*(x_actPt - x_xP - \
x_o - x)/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + \
x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + \
x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - \
x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) - (y_xK - y_xQ)*\
(-y_xQ + y_xP)*(y_actPt - y_xP - y_o)/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - \
x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + \
x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - \
x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + \
x_xQ*y_xQ*z_xP) - (x_xK*y_xK - x_xK*y_xQ - x_xP*y_xK + x_xP*y_xQ)*\
(x_actQt - x_xQ - x_o - x)/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK +\
x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - \
x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + \
x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) - \
(z_xK - z_xP)*(-y_xQ + y_xP)*(z_actQt - z_xQ - z_o)/(x_xK*y_xK*z_xP - \
x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - \
2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + \
x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - \
x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) + (y_xK - y_xQ)*(-y_xQ + y_xP)*\
(y_actKt - y_xK - y_o)/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + \
x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - \
x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + \
x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) + \
(z_xK - z_xP)*(-y_xQ + y_xP)*(z_actKt - z_xK - z_o)/(x_xK*y_xK*z_xP - \
x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - \
2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + \
x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - \
x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP)) * 1e6
    return E_BX

def epsilon_z(x_actPt, x_actQt, x_xK, x_xP, x_xQ, y_actKt, y_actPt, y_xK, \
              y_xP, y_xQ, z_actKt, z_actQt, z_xK, z_xP, z_xQ):
    E_CX = ((x_xK - x_xQ)*(z_xK - z_xP)*(x_actPt - x_xP - x_o - x)/\
(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + \
x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + \
x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - \
```

```
x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) - (y_xK - y_xQ)*\
(-z_xQ + z_xP)*(y_actPt - y_xP - y_o)/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - \
x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + \
x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - \
x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + \
x_xQ*y_xQ*z_xP) - (x_xK - x_xQ)*(z_xK - z_xP)*(x_actQt - x_xQ - x_o - x)/\
(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + \
x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + \
x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - \
x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) - (z_xK - z_xP)*\
(-z_xQ + z_xP)*(z_actQt - z_xQ - z_o)/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - \
x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + \
x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - \
x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + \
x_xQ*y_xQ*z_xP) + (y_xK - y_xQ)*(-z_xQ + z_xP)*(y_actKt - y_xK - y_o)/\
(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + \
x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + \
x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - \
x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + x_xQ*y_xQ*z_xP) + (z_xK - z_xP)*\
(-z_xQ + z_xP)*(z_actKt - z_xK - z_o)/(x_xK*y_xK*z_xP - x_xK*y_xK*z_xQ - \
x_xK*y_xP*z_xK + x_xK*y_xP*z_xP + x_xK*y_xQ*z_xK - 2*x_xK*y_xQ*z_xP + \
x_xK*y_xQ*z_xQ - x_xP*y_xK*z_xP + x_xP*y_xK*z_xQ + x_xP*y_xQ*z_xP - \
x_xP*y_xQ*z_xQ + x_xQ*y_xP*z_xK - x_xQ*y_xP*z_xP - x_xQ*y_xQ*z_xK + \
x_xQ*y_xQ*z_xP)) * 1e6
    return E_CX

# Plotting the geometrical error parameters

import matplotlib.pyplot as plt

fig, axs = plt.subplots(3, 2)

axs[0, 0].plot(x, delta_x(x_o, x_actPt, x_actQt, x_xK, x_xP, x_xQ, y_actKt, \
               y_actPt, y_xK, y_xP, y_xQ, z_actKt, z_actQt, z_xK, z_xP, z_xQ))
axs[0, 0].set(xlabel = r'$x$, $[mm]$', ylabel = r'$E_{XX}$, $[\mu m]$')
axs[0, 0].scatter(x, delta_x(x_o, x_actPt, x_actQt, x_xK, x_xP, x_xQ, y_actKt,\
               y_actPt, y_xK, y_xP, y_xQ, z_actKt, z_actQt, z_xK, z_xP, z_xQ))

axs[1, 0].plot(x, delta_y(x_actPt, x_actQt, x_xK, x_xP, x_xQ, y_o, y_actKt, \
               y_actPt, y_xK, y_xP, y_xQ, z_actKt, z_actQt, z_xK, z_xP, z_xQ))
axs[1, 0].set(xlabel = r'$x$, $[mm]$', ylabel = r'$E_{YX}$, $[\mu m]$')
axs[1, 0].scatter(x, delta_y(x_actPt, x_actQt, x_xK, x_xP, x_xQ, y_o, y_actKt,\
               y_actPt, y_xK, y_xP, y_xQ, z_actKt, z_actQt, z_xK, z_xP, z_xQ))

axs[2, 0].plot(x, delta_z(x_actPt, x_actQt, x_xK, x_xP, x_xQ, y_actKt, \
          y_actPt, y_xK, y_xP, y_xQ, z_o, z_actKt, z_actQt, z_xK, z_xP, z_xQ))
axs[2, 0].set(xlabel = r'$x$, $[mm]$', ylabel = r'$E_{ZX}$, $[\mu m]$')
axs[2, 0].scatter(x, delta_z(x_actPt, x_actQt, x_xK, x_xP, x_xQ, y_actKt, \
          y_actPt, y_xK, y_xP, y_xQ, z_o, z_actKt, z_actQt, z_xK, z_xP, z_xQ))

axs[0, 1].plot(x, epsilon_x(x_actPt, x_actQt, x_xK, x_xP, x_xQ, y_actKt, \
               y_actPt, y_xK, y_xP, y_xQ, z_actKt, z_actQt, z_xK, z_xP, z_xQ))
axs[0, 1].set(xlabel = r'$x$, $[mm]$', ylabel = r'$E_{AX}$, $[\mu rad]$')
axs[0, 1].scatter(x, epsilon_x(x_actPt, x_actQt, x_xK, x_xP, x_xQ, y_actKt, \
               y_actPt, y_xK, y_xP, y_xQ, z_actKt, z_actQt, z_xK, z_xP, z_xQ))

axs[1, 1].plot(x, epsilon_y(x_actPt, x_actQt, x_xK, x_xP, x_xQ, y_actKt, \
               y_actPt, y_xK, y_xP, y_xQ, z_actKt, z_actQt, z_xK, z_xP, z_xQ))
axs[1, 1].set(xlabel = r'$x$, $[mm]$', ylabel = r'$E_{BX}$, $[\mu rad]$')
axs[1, 1].scatter(x, epsilon_y(x_actPt, x_actQt, x_xK, x_xP, x_xQ, y_actKt, \
              y_actPt, y_xK, y_xP, y_xQ, z_actKt, z_actQt,  z_xK, z_xP, z_xQ))

axs[2, 1].plot(x, epsilon_z(x_actPt, x_actQt, x_xK, x_xP, x_xQ, y_actKt, \
               y_actPt, y_xK, y_xP, y_xQ, z_actKt, z_actQt, z_xK, z_xP, z_xQ))
```

```
297 axs[2, 1].set(xlabel = r'$x$, $[mm]$', ylabel = r'$E_{CX}$, $[\mu rad]$')
298 axs[2, 1].scatter(x, epsilon_z(x_actPt, x_actQt, x_xK, x_xP, x_xQ, y_actKt, \
299                 y_actPt, y_xK, y_xP, y_xQ, z_actKt, z_actQt, z_xK, z_xP, z_xQ))
300
301 # Saving the data to text files
302
303 np.savetxt('E_XX.txt',(np.vstack((x, delta_x(x_o, x_actPt, x_actQt, x_xK, \
304 x_xP, x_xQ, y_actKt, y_actPt, y_xK, y_xP, y_xQ, z_actKt, z_actQt, z_xK, z_xP, \
305 z_xQ))).T), delimiter = ',', fmt = '%1.1f %1.1f')
306
307 np.savetxt('E_YX.txt',(np.vstack((x, delta_y(x_actPt, x_actQt, x_xK, x_xP, \
308 x_xQ, y_o, y_actKt, y_actPt, y_xK, y_xP, y_xQ, z_actKt, z_actQt, z_xK, z_xP, \
309 z_xQ))).T), delimiter = ',', fmt = '%1.1f %1.1f')
310
311 np.savetxt('E_ZX.txt',(np.vstack((x, delta_z(x_actPt, x_actQt, x_xK, x_xP, \
312 x_xQ, y_actKt, y_actPt, y_xK, y_xP, y_xQ, z_o, z_actKt, z_actQt, z_xK, z_xP, \
313 z_xQ))).T), delimiter = ',', fmt = '%1.1f %1.1f')
314
315 np.savetxt('E_AX.txt',(np.vstack((x, epsilon_x(x_actPt, x_actQt, x_xK, x_xP, \
316 x_xQ, y_actKt, y_actPt, y_xK, y_xP, y_xQ, z_actKt, z_actQt, z_xK, z_xP, \
317 z_xQ))).T), delimiter = ',', fmt = '%1.1f %1.3f')
318
319 np.savetxt('E_BX.txt',(np.vstack((x, epsilon_y(x_actPt, x_actQt, x_xK, x_xP, \
320 x_xQ, y_actKt, y_actPt, y_xK, y_xP, y_xQ, z_actKt, z_actQt, z_xK, z_xP, \
321 z_xQ))).T), delimiter = ',', fmt = '%1.1f %1.3f')
322
323 np.savetxt('E_CX.txt',(np.vstack((x, epsilon_z(x_actPt, x_actQt, x_xK, x_xP, \
324 x_xQ, y_actKt, y_actPt, y_xK, y_xP, y_xQ, z_actKt, z_actQt, z_xK, z_xP, \
325 z_xQ))).T), delimiter = ',', fmt = '%1.1f %1.3f')
```

C.2 Python Script for Calculating the Average Error and Measurement Variability of the Linear Axes

```
import numpy as np
import matplotlib.pyplot as plt


def average(parameter):
    Error_values = []
    Average_values = np.zeros(11)
    Error = np.zeros((2, 11))
    Std_dev = np.zeros(11)
    Std_dev_percent = np.zeros(11)
    for i in range(4):
        with open(f'Round {i+2}\{parameter}.txt') as file:
            d0 = file.read()
            d1 = d0.split('\n')
            d1.pop(-1)
            d2 = [i.split(' ') for i in d1]
            d2 = np.array(d2)
            d2 = d2.astype(np.float64)

            command_pos = np.array([d2[j][0] for j in range(len(d2))])
            Error_values.append(d2[:, 1])
    Error_values = np.array(Error_values)
    Error_values = np.concatenate((Error_values[0], Error_values[1], \
                                   Error_values[2], Error_values[3]))
    Error_values = np.reshape(Error_values, (4, 11))
    for i in range(11):
        Average_values[i] = np.sum(Error_values[:, i])/4
        Error[0, i] = abs(Error_values.max(axis = 0)[i] - Average_values[i])
        Error[1, i] = abs(Error_values.min(axis = 0)[i] - Average_values[i])
    for i in range (4):
        for j in range(11):
            Error_values_minus_average = Error_values - Average_values
            Squared = Error_values_minus_average**2
            Std_dev[j] = np.sqrt(np.sum(Squared, axis = 0)[j]/3)
            Avg_std_dev = np.sum(Std_dev)/len(Std_dev)
            Max_std_dev = np.max(Std_dev)
            if Average_values[j] == 0:
                Std_dev_percent[j] = 0
            else:
                Std_dev_percent[j] = abs(Std_dev[j]/Average_values[j]) * 100
    Avg_std_dev_percent = np.sum(abs(Std_dev))/np.sum(abs(Average_values)) * 100
    return command_pos, Error_values, Average_values, Error, Std_dev, \
        Max_std_dev, Avg_std_dev, Std_dev_percent, Avg_std_dev_percent

fig, axs = plt.subplots(3, 2)

axs[0, 0].errorbar(average('E_XX')[0], average('E_XX')[2], \
                   yerr = average('E_XX')[4] * 2, ecolor = 'k', capsize = 3)
axs[0, 0].set(xlabel = r'$x$, $[mm]$', ylabel = r'$E_{XX}$, $[\mu m]$', \
              ylim = (-25, 35))
axs[0, 0].scatter(average('E_XX')[0], average('E_XX')[2])

axs[1, 0].errorbar(average('E_YX')[0], average('E_YX')[2], \
                   yerr = average('E_YX')[4] * 2, ecolor = 'k', capsize = 3)
axs[1, 0].set(xlabel = r'$x$, $[mm]$', ylabel = r'$E_{YX}$, $[\mu m]$', \
              ylim = (-25, 35))
axs[1, 0].scatter(average('E_YX')[0], average('E_YX')[2])

axs[2, 0].errorbar(average('E_ZX')[0], average('E_ZX')[2], \
```

```
                    yerr = average('E_ZX')[4] * 2, ecolor = 'k', capsize = 3)
axs[2, 0].set(xlabel = r'$x$, $[mm]$', ylabel = r'$E_{ZX}$, $[\mu m]$', \
              ylim = (-25, 35))
axs[2, 0].scatter(average('E_ZX')[0], average('E_ZX')[2])

axs[0, 1].errorbar(average('E_AX')[0], average('E_AX')[2], \
                    yerr = average('E_AX')[4] * 2, ecolor = 'k', capsize = 3)
axs[0, 1].set(xlabel = r'$x$, $[mm]$', ylabel = r'$E_{AX}$, $[\mu rad]$')
axs[0, 1].scatter(average('E_AX')[0], average('E_AX')[2])

axs[1, 1].errorbar(average('E_BX')[0], average('E_BX')[2], \
                    yerr = average('E_BX')[4] * 2, ecolor = 'k', capsize = 3)
axs[1, 1].set(xlabel = r'$x$, $[mm]$', ylabel = r'$E_{BX}$, $[\mu rad]$')
axs[1, 1].scatter(average('E_BX')[0], average('E_BX')[2])

axs[2, 1].errorbar(average('E_CX')[0], average('E_CX')[2], \
                    yerr = average('E_CX')[4] * 2, ecolor = 'k', capsize = 3)
axs[2, 1].set(xlabel = r'$x$, $[mm]$', ylabel = r'$E_{CX}$, $[\mu rad]$')
axs[2, 1].scatter(average('E_CX')[0], average('E_CX')[2])



np.savetxt('E_XX_avg..txt',(np.vstack((average('E_XX')[0], average('E_XX')[2],\
average('E_XX')[4]))).T), delimiter = ',', fmt = '%1.1f %1.3f %1.3f')

np.savetxt('E_YX_avg..txt',(np.vstack((average('E_YX')[0], average('E_YX')[2],\
average('E_YX')[4]))).T), delimiter = ',', fmt = '%1.1f %1.3f %1.3f')

np.savetxt('E_ZX_avg..txt',(np.vstack((average('E_ZX')[0], average('E_ZX')[2],\
average('E_ZX')[4]))).T), delimiter = ',', fmt = '%1.1f %1.3f %1.3f')
np.savetxt('E_AX_avg..txt',(np.vstack((average('E_AX')[0], average('E_AX')[2],\
average('E_AX')[4]))).T), delimiter = ',', fmt = '%1.1f %1.3f %1.3f')

np.savetxt('E_BX_avg..txt',(np.vstack((average('E_BX')[0], average('E_BX')[2],\
average('E_BX')[4]))).T), delimiter = ',', fmt = '%1.1f %1.3f %1.3f')

np.savetxt('E_CX_avg..txt',(np.vstack((average('E_CX')[0], average('E_CX')[2],\
average('E_CX')[4]))).T), delimiter = ',', fmt = '%1.1f %1.3f %1.3f')
np.savetxt('E_XX_std._dev..txt',(np.vstack((average('E_XX')[5], \
average('E_XX')[6], average('E_XX')[8])).T), delimiter = ',', \
fmt = '%1.1f %1.1f %1i')
```

Appendix D

Python Scripts - Rotational Axes

D.1 Python Script For Calculating Errors of the Rotational Axes

```
# ==========================================================================
# Error Parameters Associated with the B-axis
# ==========================================================================

import numpy as np

# Defining the origin

Origin = np.array([0, 0, 0])
x_o, y_o, z_o = Origin[0], Origin[1], Origin[2]

# Importing the files containing the coordinates of the measured points,
# and arranging the data appropriately

with open('B, Round 1.csv') as file:
        d0 = file.read()
        d1 = d0.split('\n')
        d1.pop(-1)
        d2 = [i.split(',') for i in d1]

        K_b = np.array([d2[0][1], d2[0][2], d2[0][3], 1])
        K_act_t = np.zeros((19, 4))
        for i in range(19):
            K_act_t[i][0:3] = d2[i][1:]
            K_act_t[i][3] = 1

        P_b = np.array([d2[19][1], d2[19][2], d2[19][3], 1])
        P_act_t = np.zeros((19, 4))
        for i in range(19, 38):
            P_act_t[i-19][0:3] = d2[i][1:]
            P_act_t[i-19][3] = 1

        Q_b = np.array([d2[38][1], d2[38][2], d2[38][3], 1])
        Q_act_t = np.zeros((19, 4))
        for i in range(38, 57):
            Q_act_t[i-38][0:3] = d2[i][1:]
            Q_act_t[i-38][3] = 1

# Defining the coordinates of the points P, Q and K in the reference frame

x_BP, y_BP, z_BP = float(P_b[0]), float(P_b[1]), float(P_b[2])
```

```
x_BQ, y_BQ, z_BQ = float(Q_b[0]), float(Q_b[1]), float(Q_b[2])
x_BK, y_BK, z_BK = float(K_b[0]), float(K_b[1]), float(K_b[2])

# Defining the nominal displacements along the axis of movement

theta = np.array(np.linspace(0, 180 * np.pi/180, num = 19))
theta_deg = np.array(np.linspace(0, 180, num = 19))

# Deviding the coordinates into separate arrays

x_actPt, y_actPt, z_actPt = P_act_t[:,0], P_act_t[:,1], P_act_t[:,2]
x_actQt, y_actQt, z_actQt = Q_act_t[:,0], Q_act_t[:,1], Q_act_t[:,2]
x_actKt, y_actKt, z_actKt = K_act_t[:,0], K_act_t[:,1], K_act_t[:,2]

# Defining the geometrical error parameter equations

def delta_x(theta, x_o, x_BK, x_BP, x_BQ, x_actPt, x_actQt, y_BK, y_BP, y_BQ, \
            y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, z_actQt):
    E_XB = (((z_BP*(y_BK - y_BP)*y_BQ**2 + ((y_BK*y_BP + (x_BP + x_BQ)*x_BK - \
2*x_BP*x_BQ - y_BK**2)*z_BP + z_BQ*y_BP**2 - y_BK*y_BP*z_BQ - ((x_BK - x_BP)*\
z_BQ + z_BK*(x_BK - x_BQ))*x_BP)*y_BQ - x_BQ*((x_BK - x_BQ)*y_BP + y_BK*\
(x_BK - x_BP))*z_BP - y_BK*y_BP**2*z_BQ + (z_BQ*y_BK**2 + x_BQ*z_BK*\
(x_BK - x_BQ))*y_BP + z_BQ*x_BP*y_BK*(x_BK - x_BP))*np.cos(theta) - z_BP*\
(y_actKt - y_actPt)*y_BQ**2 + ((-z_BQ - z_actKt + z_actQt + z_BK)*z_BP**2 + \
(z_BQ*z_BK + (2*x_o - x_actPt - x_actQt)*x_BK + (y_actKt - y_actPt)*y_BK - \
z_BK**2 + (z_actKt - z_actQt)*z_BK + (-x_o + x_actQt)*x_BP - x_BQ*\
(x_o - x_actPt))*z_BP + z_BQ*(y_actKt - y_actPt)*y_BP - ((x_BK - x_BP)*z_BQ + \
z_BK*(x_BK - x_BQ))*(x_o - x_actPt))*y_BQ + ((z_BQ**2 + (z_actKt - z_actQt - \
z_BK)*z_BQ - (x_BK - x_BQ)*(x_o - x_actQt))*y_BP - y_BK*(x_BK - x_BP)*\
(x_o - x_actQt))*z_BP + (-z_BK*z_BQ**2 + ((-y_actKt + y_actPt)*y_BK - z_BK*\
(z_actKt - z_actQt - z_BK))*z_BQ + z_BK*(x_BK - x_BQ)*(x_o - x_actQt))*y_BP + \
z_BQ*y_BK*(x_BK - x_BP)*(x_o - x_actPt))/(((-2*x_BK + x_BP + x_BQ)*z_BP + \
(x_BK - x_BP)*z_BQ + z_BK*(x_BK - x_BQ))*y_BQ + ((x_BK - x_BQ)*y_BP + y_BK*\
(x_BK - x_BP))*z_BP - z_BK*(x_BK - x_BQ)*y_BP - z_BQ*y_BK*(x_BK - x_BP))) * 1e3
    return E_XB

def delta_y(theta, x_BK, x_BP, x_BQ, x_actPt, x_actQt, y_o, y_BK, y_BP, y_BQ, \
            y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, z_actQt):
    E_YB = (((-z_BP*(x_BP - x_BQ)*x_BK**2 + ((x_BP*x_BQ + (y_BK + y_BP)*y_BQ -\
x_BQ**2 - 2*y_BK*y_BP)*z_BP + z_BK*x_BP**2 - x_BP*x_BQ*z_BK + ((y_BP - y_BQ)*\
z_BK + z_BQ*(y_BK - y_BQ))*y_BP)*x_BK + ((y_BK - y_BQ)*x_BP + x_BQ*(y_BP - \
y_BQ))*y_BK*z_BP - x_BP**2*x_BQ*z_BK + (x_BQ**2*z_BK - z_BQ*y_BK*(y_BK - \
y_BQ))*x_BP - x_BQ*y_BP*z_BK*(y_BP - y_BQ))*np.cos(theta) + z_BP*(x_actPt - \
x_actQt)*x_BK**2 + ((z_BQ + z_actKt - z_actQt - z_BK)*z_BP**2 + (z_BQ*z_BK + \
(2*y_o - y_actKt - y_actPt)*y_BQ + (-x_actPt + x_actQt)*x_BQ - z_BQ**2 + \
(-z_actKt + z_actQt)*z_BQ + (-y_o + y_actPt)*y_BK - y_BP*(y_o - y_actKt))*\
z_BP - z_BK*(x_actPt - x_actQt)*x_BP + (y_o - y_actPt)*((y_BP - y_BQ)*z_BK + \
z_BQ*(y_BK - y_BQ)))*x_BK + ((z_BK**2 + (-z_BQ - z_actKt + z_actQt)*z_BK + \
(y_BK - y_BQ)*(y_o - y_actKt))*x_BP + x_BQ*(y_BP - y_BQ)*(y_o - y_actKt))*\
z_BP + (-z_BK**2*z_BQ + ((x_actPt - x_actQt)*x_BQ + z_BQ*(z_BQ + z_actKt - \
z_actQt))*z_BK - z_BQ*(y_BK - y_BQ)*(y_o - y_actKt))*x_BP - x_BQ*z_BK*(y_BP - \
y_BQ)*(y_o - y_actPt))/(((y_BK + y_BP - 2*y_BQ)*z_BP + (-y_BP + y_BQ)*z_BK - \
z_BQ*(y_BK - y_BQ))*x_BK + ((-y_BK + y_BQ)*x_BP - x_BQ*(y_BP - y_BQ))*z_BP + \
z_BQ*(y_BK - y_BQ)*x_BP + x_BQ*z_BK*(y_BP - y_BQ))) * 1e3
    return E_YB

def delta_z(theta, x_BK, x_BP, x_BQ, x_actPt, x_actQt, y_BK, y_BP, y_BQ, \
            y_actKt, y_actPt, z_o, z_BK, z_BP, z_BQ, z_actKt, z_actQt):
    E_ZB = ((((x_BP - x_BQ)*x_BK + (-y_BK + y_BP)*y_BQ - x_BP**2 + x_BP*\
x_BQ + y_BK*y_BP - y_BP**2)*(x_BK*y_BQ - x_BQ*y_BK)*np.cos(theta) - y_BQ*\
(x_actPt - x_actQt)*x_BK**2 + ((y_actKt - y_actPt)*y_BQ**2 + ((2*z_o + z_BQ - \
z_actKt - z_actQt + z_BK)*z_BP + (x_actPt - x_actQt)*x_BP + (-y_actKt + \
y_actPt)*y_BP + (-z_o - 2*z_BQ + z_actQt)*z_BK - z_BQ*(z_o - z_actKt))*y_BQ + \
y_BK*(x_actPt - x_actQt)*x_BQ - ((z_BP - z_BQ)*y_BK + y_BP*(z_BP - z_BK))*\
```

```
(z_o + z_BQ - z_actQt))*x_BK + (((-y_actKt + y_actPt)*y_BK - (z_BP - z_BK)*\
(z_o - z_actKt + z_BK))*x_BQ - x_BP*(z_BP - z_BQ)*(z_o - z_actKt + z_BK))*\
y_BQ + (((-x_actPt + x_actQt)*x_BP + y_BP*(y_actKt - y_actPt))*y_BK + y_BP*\
(z_BP - z_BK)*(z_o - z_actKt + z_BK))*x_BQ + x_BP*y_BK*(z_BP - z_BQ)*\
(z_o + z_BQ - z_actQt))/(((-2*z_BP + z_BQ + z_BK)*y_BQ + (z_BP - z_BQ)*y_BK + \
y_BP*(z_BP - z_BK))*x_BK + ((z_BP - z_BK)*x_BQ + x_BP*(z_BP - z_BQ))*y_BQ - \
y_BP*(z_BP - z_BK)*x_BQ - x_BP*y_BK*(z_BP - z_BQ))) * 1e3
    return E_ZB

def epsilon_x(theta, x_BK, x_BP, x_BQ, x_actPt, x_actQt, y_BK, y_BP, y_BQ, \
              y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, z_actQt):
    E_AB = ((((x_BP - x_BQ)*x_BK - x_BP**2 + x_BP*x_BQ + (y_BP - y_BQ)*\
(y_BK - y_BP))*(x_BK - x_BQ)*np.cos(theta) + (-x_actPt + x_actQt)*x_BK**2 + \
((x_actPt - x_actQt)*x_BP + (x_actPt - x_actQt)*x_BQ + (y_actKt - y_actPt)*\
y_BQ + (-z_BQ - z_actKt + z_actQt + z_BK)*z_BP + (-y_actKt + y_actPt)*y_BP + \
z_BQ*(z_BQ + z_actKt - z_actQt - z_BK))*x_BK + ((-x_actPt + x_actQt)*x_BQ - \
(-z_BP + z_BQ)*(z_BQ + z_actKt - z_actQt - z_BK))*x_BP + x_BQ*(y_BP - y_BQ)*\
(y_actKt - y_actPt))/(((-z_BQ - z_BK + 2*z_BP)*y_BQ + (-y_BK - y_BP)*z_BP + \
y_BK*z_BQ + y_BP*z_BK)*x_BK - (y_BK - y_BQ)*(-z_BP + z_BQ)*x_BP - x_BQ*\
(z_BK - z_BP)*(y_BP - y_BQ))) * 1e6
    return E_AB

def epsilon_y(theta, x_BK, x_BP, x_BQ, x_actPt, x_actQt, y_BK, y_BP, y_BQ, \
              y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, z_actQt):
    E_BB = (((-y_BK + y_BP)*y_BQ + y_BK*y_BP - y_BP**2 + (x_BP - x_BQ)*\
(x_BK - x_BP))*(y_BK - y_BQ)*np.cos(theta) + (-y_actKt + y_actPt)*y_BQ**2 + \
((y_actKt - y_actPt)*y_BK + (y_actKt - y_actPt)*y_BP + (x_actPt - x_actQt)*\
x_BK + (-z_BQ - z_actKt + z_actQt + z_BK)*z_BP + (-x_actPt + x_actQt)*x_BP + \
z_BK*(z_BQ + z_actKt - z_actQt - z_BK))*y_BQ + ((-y_actKt + y_actPt)*y_BP - \
(x_BK - x_BP)*(x_actPt - x_actQt))*y_BK - y_BP*(z_BK - z_BP)*(z_BQ + z_actKt \
- z_actQt - z_BK))/(((-z_BQ - z_BK + 2*z_BP)*x_BK + (-x_BP - x_BQ)*z_BP + \
z_BQ*x_BP + x_BQ*z_BK)*y_BQ + (x_BK - x_BP)*(-z_BP + z_BQ)*y_BK + y_BP*\
(z_BK - z_BP)*(x_BK - x_BQ)) * 1e6
    return E_BB

def epsilon_z(theta, x_BK, x_BP, x_BQ, x_actPt, x_actQt, y_BK, y_BP, y_BQ, \
              y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, z_actQt):
    E_CB = ((((((-x_BP + x_BQ)*x_BK + (-y_BK + y_BP)*y_BQ - x_BQ**2 + \
x_BP*x_BQ + y_BK*(y_BK - y_BP))*z_BP + (x_BP - x_BQ)*(x_BK - x_BQ)*z_BK - \
z_BQ*(y_BK - y_BQ)*(y_BK - y_BP))*np.cos(theta) + (((y_BK + y_BP - 2*y_BQ)*\
x_BK + (x_BP + x_BQ)*y_BQ - x_BP*y_BK - x_BQ*y_BP)*z_BP - (y_BP - y_BQ)*\
(x_BK - x_BQ)*z_BK - z_BQ*(y_BK - y_BQ)*(x_BK - x_BP))*np.sin(theta) + \
(z_BQ + z_actKt - z_actQt - z_BK)*z_BP**2 + (z_BK**2 + (-z_actKt + z_actQt)*\
z_BK - z_BQ**2 + (-z_actKt + z_actQt)*z_BQ + (x_actPt - x_actQt)*x_BK + \
(y_actKt - y_actPt)*y_BQ + (-x_actPt + x_actQt)*x_BQ - (y_actKt - y_actPt)*\
y_BK)*z_BP - z_BK**2*z_BQ + (z_BQ**2 + (z_actKt - z_actQt)*z_BQ - \
(x_BK - x_BQ)*(x_actPt - x_actQt))*z_BK + z_BQ*(y_BK - y_BQ)*(y_actKt - \
y_actPt))/(((-y_BK - y_BP + 2*y_BQ)*x_BK + (-x_BP - x_BQ)*y_BQ + x_BP*y_BK + \
x_BQ*y_BP)*z_BP + (y_BP - y_BQ)*(x_BK - x_BQ)*z_BK + z_BQ*(y_BK - y_BQ)*\
(x_BK - x_BP))) * 1e6
    return E_CB


# Plotting the geometrical error parameters

import matplotlib.pyplot as plt

fig, axs = plt.subplots(3, 2, subplot_kw=dict(polar=True))

axs[0, 0].plot(theta, delta_x(theta, x_o, x_BK, x_BP, x_BQ, x_actPt, x_actQt, \
y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, z_actQt))
axs[0, 0].set(xlabel = r'$\mathrm{B}$, $[^{\circ}]$', \
              ylabel = r'$\mathrm{E_{XB}}$, $[\mathrm{\mu m}]$')
axs[0, 0].scatter(theta, delta_x(theta, x_o, x_BK, x_BP, x_BQ, x_actPt, \
```

```
x_actQt, y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, \
z_actQt))

axs[1, 0].plot(theta, delta_y(theta, x_BK, x_BP, x_BQ, x_actPt, x_actQt, y_o, \
y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, z_actQt))
axs[1, 0].set(xlabel = r'$\mathrm{B}$, $[^{\circ}]$', \
              ylabel = r'$\mathrm{E_{YB}}$, $[\mathrm{\mu m}]$')
axs[1, 0].scatter(theta, delta_y(theta, x_BK, x_BP, x_BQ, x_actPt, x_actQt, \
y_o, y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, z_actQt))

axs[2, 0].plot(theta, delta_z(theta, x_BK, x_BP, x_BQ, x_actPt, x_actQt, \
y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_o, z_BK, z_BP, z_BQ, z_actKt, z_actQt))
axs[2, 0].set(xlabel = r'$\mathrm{B}$, $[^{\circ}]$', \
              ylabel = r'$\mathrm{E_{ZB}}$, $[\mathrm{\mu m}]$')
axs[2, 0].scatter(theta, delta_z(theta, x_BK, x_BP, x_BQ, x_actPt, x_actQt, \
y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_o, z_BK, z_BP, z_BQ, z_actKt, z_actQt))

axs[0, 1].plot(theta, epsilon_x(theta, x_BK, x_BP, x_BQ, x_actPt, x_actQt, \
y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, z_actQt))
axs[0, 1].set(xlabel = r'$\mathrm{B}$, $[^{\circ}]$', \
              ylabel = r'$\mathrm{E_{AB}}$, $[\mathrm{\mu rad}]$')
axs[0, 1].scatter(theta, epsilon_x(theta, x_BK, x_BP, x_BQ, x_actPt, x_actQt, \
y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, z_actQt))

axs[1, 1].plot(theta, epsilon_y(theta, x_BK, x_BP, x_BQ, x_actPt, x_actQt, \
y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, z_actQt))
axs[1, 1].set(xlabel = r'$\mathrm{B}$, $[^{\circ}]$', \
              ylabel = r'$\mathrm{E_{BB}}$, $[\mathrm{\mu rad}]$')
axs[1, 1].scatter(theta, epsilon_y(theta, x_BK, x_BP, x_BQ, x_actPt, x_actQt, \
y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, z_actQt))

axs[2, 1].plot(theta, epsilon_z(theta, x_BK, x_BP, x_BQ, x_actPt, x_actQt, \
y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, z_actQt))
axs[2, 1].set(xlabel = r'$\mathrm{B}$, $[^{\circ}]$', \
              ylabel = r'$\mathrm{E_{CB}}$, $[\mathrm{\mu rad}]$')
axs[2, 1].scatter(theta, epsilon_z(theta, x_BK, x_BP, x_BQ, x_actPt, x_actQt, \
y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, z_actQt))

fig, axs = plt.subplots(3, 2)

axs[0, 0].plot(theta_deg, delta_x(theta, x_o, x_BK, x_BP, x_BQ, x_actPt, \
x_actQt, y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, z_actQt))
axs[0, 0].set(xlabel = r'$\mathrm{B}$, $[^{\circ}]$', \
              ylabel = r'$\mathrm{E_{XB}}$, $[\mathrm{\mu m}]$')
axs[0, 0].scatter(theta_deg, delta_x(theta, x_o, x_BK, x_BP, x_BQ, x_actPt, \
x_actQt, y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, \
z_actQt))

axs[1, 0].plot(theta_deg, delta_y(theta, x_BK, x_BP, x_BQ, x_actPt, x_actQt, \
y_o, y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, z_actQt))
axs[1, 0].set(xlabel = r'$\mathrm{B}$, $[^{\circ}]$', \
              ylabel = r'$\mathrm{E_{YB}}$, $[\mathrm{\mu m}]$')
axs[1, 0].scatter(theta_deg, delta_y(theta, x_BK, x_BP, x_BQ, x_actPt, \
x_actQt, y_o, y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, \
z_actQt))

axs[2, 0].plot(theta_deg, delta_z(theta, x_BK, x_BP, x_BQ, x_actPt, x_actQt, \
y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_o, z_BK, z_BP, z_BQ, z_actKt, z_actQt))
axs[2, 0].set(xlabel = r'$\mathrm{B}$, $[^{\circ}]$', \
              ylabel = r'$\mathrm{E_{ZB}}$, $[\mathrm{\mu m}]$')
axs[2, 0].scatter(theta_deg, delta_z(theta, x_BK, x_BP, x_BQ, x_actPt, \
x_actQt, y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_o, z_BK, z_BP, z_BQ, z_actKt, z_actQt))

axs[0, 1].plot(theta_deg, epsilon_x(theta, x_BK, x_BP, x_BQ, x_actPt, \
```

```
x_actQt, y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, \
z_actQt))
axs[0, 1].set(xlabel = r'$\mathrm{B}$, $[^{\circ}]$', \
              ylabel = r'$\mathrm{E_{AB}}$, $[\mathrm{\mu rad}]$')
axs[0, 1].scatter(theta_deg, epsilon_x(theta, x_BK, x_BP, x_BQ, x_actPt, \
x_actQt, y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, \
z_actQt))

axs[1, 1].plot(theta_deg, epsilon_y(theta, x_BK, x_BP, x_BQ, x_actPt, \
x_actQt, y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, \
z_actQt))
axs[1, 1].set(xlabel = r'$\mathrm{B}$, $[^{\circ}]$', \
              ylabel = r'$\mathrm{E_{BB}}$, $[\mathrm{\mu rad}]$')
axs[1, 1].scatter(theta_deg, epsilon_y(theta, x_BK, x_BP, x_BQ, x_actPt, \
x_actQt, y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, \
z_actQt))

axs[2, 1].plot(theta_deg, epsilon_z(theta, x_BK, x_BP, x_BQ, x_actPt, \
x_actQt, y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, \
z_actQt))
axs[2, 1].set(xlabel = r'$\mathrm{B}$, $[^{\circ}]$', \
              ylabel = r'$\mathrm{E_{CB}}$, $[\mathrm{\mu rad}]$')
axs[2, 1].scatter(theta_deg, epsilon_z(theta, x_BK, x_BP, x_BQ, x_actPt, \
x_actQt, y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, \
z_actQt))

# Saving the data to text files

np.savetxt('E_XB.txt',(np.vstack((theta_deg, delta_x(theta, x_o, x_BK, x_BP, x_BQ,
    x_actPt, x_actQt, y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt,
    z_actQt))).T), delimiter = ',', fmt = '%1.1f %1.3f')
np.savetxt('E_YB.txt',(np.vstack((theta_deg, delta_y(theta, x_BK, x_BP, x_BQ, x_actPt,
    x_actQt, y_o, y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, z_actQt
    ))).T), delimiter = ',', fmt = '%1.1f %1.3f')
np.savetxt('E_ZB.txt',(np.vstack((theta_deg, delta_z(theta, x_BK, x_BP, x_BQ, x_actPt,
    x_actQt, y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_o, z_BK, z_BP, z_BQ, z_actKt, z_actQt
    ))).T), delimiter = ',', fmt = '%1.1f %1.3f')
np.savetxt('E_AB.txt',(np.vstack((theta_deg, epsilon_x(theta, x_BK, x_BP, x_BQ, x_actPt,
     x_actQt, y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, z_actQt))).
    T), delimiter = ',', fmt = '%1.1f %1.3f')
np.savetxt('E_BB.txt',(np.vstack((theta_deg, epsilon_y(theta, x_BK, x_BP, x_BQ, x_actPt,
     x_actQt, y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, z_actQt))).
    T), delimiter = ',', fmt = '%1.1f %1.3f')
np.savetxt('E_CB.txt',(np.vstack((theta_deg, epsilon_z(theta, x_BK, x_BP, x_BQ, x_actPt,
     x_actQt, y_BK, y_BP, y_BQ, y_actKt, y_actPt, z_BK, z_BP, z_BQ, z_actKt, z_actQt))).
    T), delimiter = ',', fmt = '%1.1f %1.3f')
```

D.2 Python Script for Calculating the Average Error and Measurement Variability of the Rotational Axes

```
import numpy as np
import matplotlib.pyplot as plt


def average(parameter):
    Error_values = []
    Average_values = np.zeros(19)
    Error = np.zeros((2, 19))
    Std_dev = np.zeros(19)
    Std_dev_percent = np.zeros(19)
    for i in range(4):
        with open(f'Round {i+1}\{parameter}.txt') as file:
            d0 = file.read()
            d1 = d0.split('\n')
            d1.pop(-1)
            d2 = [i.split(' ') for i in d1]
            d2 = np.array(d2)
            d2 = d2.astype(np.float64)

            command_pos = np.array([d2[j][0] for j in range(len(d2))])
            Error_values.append(d2[:, 1])
    Error_values = np.array(Error_values)
    Error_values = np.concatenate((Error_values[0], Error_values[1], \
                                   Error_values[2], Error_values[3]))
    Error_values = np.reshape(Error_values, (4, 19))
    for i in range(19):
        Average_values[i] = np.sum(Error_values[:, i])/4
        Error[0, i] = abs(Error_values.max(axis = 0)[i] - Average_values[i])
        Error[1, i] = abs(Error_values.min(axis = 0)[i] - Average_values[i])
    for i in range (4):
        for j in range(19):
            Error_values_minus_average = Error_values - Average_values
            Squared = Error_values_minus_average**2
            Std_dev[j] = np.sqrt(np.sum(Squared, axis = 0)[j]/3)
            Avg_std_dev = np.sum(Std_dev)/len(Std_dev)
            Max_std_dev = np.max(Std_dev)
            if Average_values[j] == 0:
                Std_dev_percent[j] = 0
            else:
                Std_dev_percent[j] = abs(Std_dev[j]/Average_values[j]) * 100
    Avg_std_dev_percent = np.sum(abs(Std_dev))/np.sum(abs(Average_values)) * 100
    return command_pos, Error_values, Average_values, Error, Std_dev, \
        Max_std_dev, Avg_std_dev, Std_dev_percent, Avg_std_dev_percent

fig, axs = plt.subplots(3, 2)

axs[0, 0].errorbar(average('E_XB')[0], average('E_XB')[2], \
                   yerr = average('E_XB')[4] * 2, ecolor = 'k', capsize = 3)
axs[0, 0].set(xlabel = r'$y$, $[mm]$', ylabel = r'$E_{XB}$, $[\mu m]$')
axs[0, 0].scatter(average('E_XB')[0], average('E_XB')[2])

axs[1, 0].errorbar(average('E_YB')[0], average('E_YB')[2], \
                   yerr = average('E_YB')[4] * 2, ecolor = 'k', capsize = 3)
axs[1, 0].set(xlabel = r'$y$, $[mm]$', ylabel = r'$E_{YB}$, $[\mu m]$')
axs[1, 0].scatter(average('E_YB')[0], average('E_YB')[2])

axs[2, 0].errorbar(average('E_ZB')[0], average('E_ZB')[2], \
                   yerr = average('E_ZB')[4] * 2, ecolor = 'k', capsize = 3)
axs[2, 0].set(xlabel = r'$y$, $[mm]$', ylabel = r'$E_{ZB}$, $[\mu m]$')
```

```
60 axs[2, 0].scatter(average('E_ZB')[0], average('E_ZB')[2])
61
62 axs[0, 1].errorbar(average('E_AB')[0], average('E_AB')[2], \
63                    yerr = average('E_AB')[4] * 2, ecolor = 'k', capsize = 3)
64 axs[0, 1].set(xlabel = r'$y$, $[mm]$', ylabel = r'$E_{AB}$, $[\mu rad]$')
65 axs[0, 1].scatter(average('E_AB')[0], average('E_AB')[2])
66
67 axs[1, 1].errorbar(average('E_BB')[0], average('E_BB')[2], \
68                    yerr = average('E_BB')[4] * 2, ecolor = 'k', capsize = 3)
69 axs[1, 1].set(xlabel = r'$y$, $[mm]$', ylabel = r'$E_{BB}$, $[\mu rad]$')
70 axs[1, 1].scatter(average('E_BB')[0], average('E_BB')[2])
71
72 axs[2, 1].errorbar(average('E_CB')[0], average('E_CB')[2], \
73                    yerr = average('E_CB')[4] * 2, ecolor = 'k', capsize = 3)
74 axs[2, 1].set(xlabel = r'$y$, $[mm]$', ylabel = r'$E_{CB}$, $[\mu rad]$')
75 axs[2, 1].scatter(average('E_CB')[0], average('E_CB')[2])
76
77 np.savetxt('E_XB_avg..txt',(np.vstack((average('E_XB')[0], average('E_XB')[2],\
78 average('E_XB')[4])).T), delimiter = ',', fmt = '%1.1f %1.3f %1.3f')
79
80 np.savetxt('E_YB_avg..txt',(np.vstack((average('E_YB')[0], average('E_YB')[2],\
81 average('E_YB')[4])).T), delimiter = ',', fmt = '%1.1f %1.3f %1.3f')
82
83 np.savetxt('E_ZB_avg..txt',(np.vstack((average('E_ZB')[0], average('E_ZB')[2],\
84 average('E_ZB')[4])).T), delimiter = ',', fmt = '%1.1f %1.3f %1.3f')
85
86 np.savetxt('E_AB_avg..txt',(np.vstack((average('E_AB')[0], average('E_AB')[2],\
87 average('E_AB')[4])).T), delimiter = ',', fmt = '%1.1f %1.3f %1.3f')
88
89 np.savetxt('E_BB_avg..txt',(np.vstack((average('E_BB')[0], average('E_BB')[2],\
90 average('E_BB')[4])).T), delimiter = ',', fmt = '%1.1f %1.3f %1.3f')
91
92 np.savetxt('E_CB_avg..txt',(np.vstack((average('E_CB')[0], average('E_CB')[2],\
93 average('E_CB')[4])).T), delimiter = ',', fmt = '%1.1f %1.3f %1.3f')
```

Appendix E

Curve Fit Function for the Rotational Axes

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  # ================================================================
5  # "curve_fit" takes the name of the error parameter and the degree of the poly-
6  # nomial you want to fit to the measured data as inputs. It then uses numpy.po-
7  # lyfit to fit a polynomial of the desired degree to the measured data. The fu-
8  # nction also fits a reference straight line to the measured data and calcula-
9  # tes the angle between it and the nominal axis. The function returns an array
10 # containing the displacements along the axis where the measurements were made
11 # (Axis), the values of the error parameter (error_parameter), an array of axi-
12 # s-displacements in whole millimeters (axis), an array containing the values
13 # of the analytical error model for every value in "axis", the mean squared er-
14 # ror of the curve fit. If applicable, the function also returns an array con-
15 # aining the values of the reference straight line for every value in "Axis"
16 # and the angle between it and the nominal axis.
17 # ================================================================
18
19 def curve_fit(error, degree):
20
21     # Treating the text file containing the error parameters
22
23     with open(f'{error}.txt') as file:
24         d0 = file.read()
25         d1 = d0.split('\n')
26         d2 = [i.split(',') for i in d1]
27         d3 = [d2[i][0].split(' ') for i in range(len(d2))]
28
29         Axis = np.zeros(len(d3)-1)
30         error_parameter = np.zeros(len(d3)-1)
31
32     for i in range(len(d3)-1):
33         Axis[i] = float(d3[i][0])
34         error_parameter[i] = float(d3[i][1])
35
36     # Fitting the polynomial to the measured data
37
38     curve_error = np.polyfit(Axis, error_parameter, degree, full = True, \
39                              cov = False)
40     if curve_error[1].size > 0:
41         residual_error = curve_error[1][0]
42         MSE_error = residual_error/len(error_parameter)
43
```

```
    # Constructing the array of axis-displacements

    if Axis[-1] < 0:
        axis = np.arange(0, Axis[-1] - 1, -1.)
    else:
        axis = np.arange(0, Axis[-1] + 1, 1.)

    error_model = np.array(curve_error[0][0] * axis**degree)
    if degree == 1:
        error_model = error_model + curve_error[0][1]
    else:
        for i in range(1, degree):
            error_model = error_model + curve_error[0][i] * axis**(degree - i)

    return Axis, error_parameter, axis, error_model, MSE_error
```

Simen H. Ellingsdalen

Improving the Accuracy of a Five-Axis Machine Tool by Laser Tracker Calibration

NTNU
Norwegian University of
Science and Technology