

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

الجمهورية الجزائرية الديمقراطية الشعبية

MINISTRY OF HIGHER EDUCATION
AND SCIENTIFIC RESEARCH

HIGHER SCHOOL IN APPLIED SCIENCES

-T L E M C E N-



المدرسة العليا في العلوم التطبيقية
École Supérieure en
Sciences Appliquées

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
- تلمسان -

Final year thesis

For the award of the master's Degree

Field of study : Automatique

Specialization : Automatique

Presented by : Selsabil DJELLOULI

Topic

**Simulating and compensating kinematic defects
in 5-Axis CNC machining
A linear feedforward approach**

Publically defended on 18/06/2026 , before a jury composed of:

Mr Brahim CHERKI	Professor	ESSA-Tlemcen	President
Mr Ghouti ABDELLAOUI	MCA	ESSA-Tlemcen	Thesis Supervisor
Mrs Amel GHOUALI	MCA	ESSA-Tlemcen	Examiner
Mr Hicham MEGNAFI	MCA	ESSA-Tlemcen	Examiner

Academic Year: 2025/2026

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

Dedication

I dedicate this work:

To myself, for never giving up.

To my parents, for their unconditional support.

To my siblings, for their encouragement.

To my family, for always being there.

To my friends, for the memories and support.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to Allah, the Almighty, for granting me the strength, patience, and perseverance to carry out and complete this project.

I would like to acknowledge **Mr. ABDELLAOUI Ghouti**, for his role as the official supervisor of this project.

I also extend my respectful thanks to the honorable members of the jury for accepting to review and evaluate this work, and for their valuable time and insights.

I am sincerely grateful to the Fablab supervisor, **Mr. ADJIM Ramz Eddine Abderrazak**, for his valuable support and practical assistance throughout this work.

I would like to express my heartfelt gratitude to my parents for their constant presence and unwavering support, which gave me the strength to overcome the most difficult moments of this journey. This accomplishment is deeply tied to everything they have done for me, and I will always remain thankful for their dedication and care.

I would also like to thank my professor, **Mrs. OUHOUD Amina**, for being there during the difficult moments and for her much-needed support and encouragement.

I am especially grateful to my former project partner, **Mr. ACHOUR AOUL Oussama**, for the help, cooperation, and team spirit shared throughout our work together. Although we did not complete this journey side by side until the end, the teamwork we built was meaningful and valuable. His continued support, dedication, and willingness to help, even beyond our time working together, were truly invaluable, and I sincerely appreciate the contribution he brought to this project.

I would also like to thank my friends, **Ms. GORINE Ikram** and **Mr. BOUMADANE Mohammed El Amine Driss**, for their encouragement, support, and presence throughout this period. Their help and friendship made a real difference, and I am truly thankful for them.

Finally, I wish to extend my appreciation to everyone who contributed to this work, whether directly or indirectly. Every piece of advice, shared expertise, and gesture of support has been deeply appreciated.

Abstract

Multi-axis CNC machining accuracy is limited by inherent geometric and kinematic flaws. This thesis investigates volumetric error generation in a 5-axis bi-rotary CNC prototype, which accumulates 43 distinct geometric errors. A mathematical digital twin was constructed using Homogeneous Transformation Matrices (HTM) to model these deviations. To mitigate this, a novel offline Hybrid Analytical-Numerical Inverse Kinematics compensation algorithm was developed. Applied via Post-CAM G-code modification to finishing trajectories, MATLAB simulations demonstrated that the algorithm successfully reduced volumetric deviations to a sub-micron tolerance (below 0.0001 mm).

Keywords: 5-Axis CNC, Bi-rotary Head, Mechatronics, Mach3, SolidCAM, Post-processor.

Résumé

La précision de l'usinage CNC multi-axes est limitée par des défauts géométriques et cinématiques inhérents. Cette thèse étudie la génération d'erreurs volumétriques dans un prototype CNC bi-rotatif à 5 axes, qui accumule 43 erreurs distinctes. Un jumeau numérique a été créé à l'aide de matrices de transformation homogènes (HTM) pour modéliser ces déviations. Pour y remédier, un algorithme de compensation hors ligne hybride (analytique-numérique) a été développé. Appliquées via la modification du code G post-FAO, les simulations MATLAB ont prouvé que l'algorithme réussit à réduire les déviations à une tolérance submicronique (inférieure à 0,0001 mm).

Mots-clés: CNC 5 axes, Tête bi-rotative, Mécatronique, Mach3, SolidCAM, Post-processeur.

الملخص

تفصل هذه المذكرة تصميم، تجميع، والإعداد البرمجي لنموذج أولي لآلة تفريز ذات تحد العيوب الهندسية والحركية الكامنة من دقة تصنيع آلات التحكم الرقمي CNC متعددة المحاور. تبحث هذه الأطروحة في توليد الخطأ الحجمي في نموذج أولي لآلة 5-Axis CNC برأس ثنائي الدوران، والذي يراكم ٤٣ خطأ هندسياً متميزاً. تم بناء توأم رقمي رياضي باستخدام مصفوفات التحويل المتجانسة HTM لنمذجة هذه الانحرافات. ولحل هذه المشكلة، تم تطوير خوارزمية هجينة جديدة لتعويض الكينماتيكا العكسية دون اتصال بالإنترنت. أظهرت عمليات محاكاة MATLAB، المطبقة عبر تعديل كود G-code في مرحلة Post-CAM لمسارات الإنهاء، أن الخوارزمية نجحت في تقليل الانحرافات الحجمية إلى تفاوت أقل من الميكرون (أقل من ٠.٠٠٠١٠ مم).

الكلمات المفتاحية: تحكم رقمي 5-Axis CNC، خطأ حجمي، نمذجة حركية، تعويض الأخطاء، كينماتيكا عكسية، Post-CAM.

Abstract

Contents

List of Figures	xi
List of Tables	xii
Abbreviations list	xiii
General introduction	1
I CNC fundamentals and performance Criteria	4
Introduction	4
I.1 Concept of computer numerical control	4
I.1.1 History	5
I.1.2 CNC machines classifications	5
I.1.2.1 3-axis	5
I.1.2.2 5-axis	6
Head-Head:	6
Table-Table:	7
Head-Table:	7
I.1.3 CNC machines applications	7
I.1.4 Advantages and limitations	7
I.2 Manufacturing performance metrics	8
I.2.1 Precision	8
I.2.2 Accuracy	9
I.2.3 The relationship between precision and accuracy	9

I.3	Design boundaries: allowance vs. tolerance	10
I.3.1	The concept of allowance	10
I.3.2	The concept of tolerance	11
I.3.3	Interaction between allowance and tolerance	11
I.4	Kinematic structures of machine tools	12
I.4.1	Coordinate systems	12
I.4.2	Serial kinematic structures	12
I.5	Classification of machining errors	14
I.5.1	Random errors: the limits of precision	15
I.5.2	Systematic errors: the limits of accuracy	15
I.6	5-axis CNC machine's total error components	16
I.6.1	Visualizing geometric component and location errors	17
I.6.1.1	Component errors of a linear axis	18
I.6.1.2	Component errors of a rotational axis	19
I.7	Measurement methods in CNC machining	21
I.7.1	Classification of measurement methods	22
I.7.1.1	Direct measurement	22
I.7.1.2	Indirect measurement	22
I.7.1.3	Finished test piece evaluation	22
I.7.2	Examples of geometric error measurement	23
I.8	Error mitigation and compensation strategies	24
I.8.1	Structural and kinematic calibration (Mechanical adjustment)	24
I.8.2	Controller-embedded numerical compensation	25
I.8.3	Algorithmic and model-based compensation	25
I.8.3.1	Online (real-time) dynamic compensation	26
I.8.3.2	Offline compensation	26
I.8.4	The Hybrid architecture	27
	Conclusion	27
II	Kinematic Modeling and Mathematical Foundations	28
	Introduction	28
II.1	Mathematical representation of rigid body motion	28

II.1.1	Denavit-Hartenberg (D-H) vs. Homogeneous Transformation Matrices (HTM)	30
II.1.2	The structure of the Homogeneous Transformation Matrix . .	30
II.2	Topological modeling of the bi-rotary configuration	31
II.2.1	Definition of structural offsets	32
II.3	Comprehensive Volumetric Kinematic Modeling	33
II.3.1	The Ideal Kinematic Model (Forward Kinematics)	33
II.3.2	The Actual Kinematic Model (Error Integration)	33
II.3.2.1	Exact model	34
II.3.2.2	Differential approximations	35
II.3.3	Volumetric Error Mapping at the TCP	38
II.4	Inverse Kinematics for Bi-Rotary Heads	38
II.4.1	The Quasi-Static Error Assumption	38
II.4.2	Stage 1: Closed-Form Orientation Decoupling	39
	Kinematic Singularity Management	42
II.4.3	Stage 2: Fixed-Point Feedforward Iteration	42
II.5	Comparative Analysis: Strategy Selection	43
II.5.1	Pre-CAM Compensation: The STL Distortion Method	44
II.5.2	Post-CAM Compensation: G-code Trajectory Modification . .	46
II.5.3	Justification for the Selected Strategy	46
	Conclusion	47
III	Digital Implementation and Simulation Results	48
III.1	Introduction	48
III.2	Simulation Environment and Initialization	49
III.2.1	Workspace Configuration	49
III.2.2	Steps 1 and 2: Runtime Tool Input and Structural Constants .	49
III.2.3	Step 3: The HTM Function	50
III.2.4	Step 4: The 43-Error Geometric Injection Engine	51
III.3	Toolpath Parsing and Simulation Scenarios (Step 5)	53
III.3.1	The Two Simulation Scenarios	54
III.3.1.1	Scenario 1: The Baseline Multi-Axis Trajectory	54

III.3.1.2 Scenario 2: The High-Density Complex Trajectory . . .	55
III.4 Forward Kinematics Evaluation (Step 6)	55
III.4.1 Memory Pre-allocation and the Evaluation Loop	55
III.4.2 Ideal Kinematic Chain Evaluation	55
III.4.3 Flawed Machine Kinematic Chain Evaluation	56
III.4.4 Baseline Error Quantification	57
III.5 The Hybrid Inverse Kinematics Compensation Engine (Step 7)	58
III.5.1 Solver Configuration and Pre-computation	58
III.5.1.1 Convergence Parameters and Solver Bounds	58
III.5.1.2 Pre-computation of the Fixed Rotary Orientation Er- ror Matrices	58
III.5.1.3 Pre-extraction of Scalar A-Axis IK Coefficients	59
III.5.1.4 Solver State Initialisation	59
III.5.2 The Outer Fixed-Point Iteration Loop	60
III.5.3 Stage 1, Step 1a: Ideal Tool Orientation Vector Extraction . . .	60
III.5.4 Stage 1, Step 1b: C-Error Frame Transformation	61
III.5.5 Stage 1, Step 1c: A-Axis Closed-Form Solution with Branch Selection	61
III.5.6 Stage 1, Step 1d: C-Axis Alignment via 2D Cross-Product and Singularity Guard	63
III.5.7 Stage 2: Full HTM Evaluation with Compensated Commands	64
III.5.8 Convergence Assessment and Linear Command Update . . .	64
III.6 Compensated G-Code Reconstruction and Export (Step 8)	66
III.7 Graphical Analysis of Simulation Results	66
III.7.1 Comprehensive Volumetric Trajectory Analysis	67
III.7.2 Component-Level Spatial Deviation Breakdown	68
III.7.3 Algorithmic Convergence Efficiency	70
III.7.4 Advanced Validation: Complex Multi-Axis Trajectory Analysis	71
III.7.4.1 High-Density Spatial Trajectory Tracking	71
III.7.4.2 Severe Parasitic Deviation Breakdown	72
III.7.4.3 Algorithmic Scalability and Convergence	73
III.7.5 G-Code Trajectory Reconstruction and Comparison	74

General Conclusion	77
Appendix	78
A Arduino program	78

List of Figures

I.1	Illustration of the relationship between precision and accuracy	9
I.2	Coordinate systems	12
I.3	Possible kinematic chain configurations of different CNC machines .	13
I.4	Error generation chain from CAD model to actual part execution . .	14
I.5	The main sources of machining errors	15
I.6	Component errors of linear horizontal Z axis	18
I.7	Location errors of a Z movement	19
I.8	Component errors of rotational C axis	20
I.9	Location errors of a C axis	21
I.10	Common kinematic measurement setups	23
I.11	Squareness error compensation	25
II.1	Kinematic chain diagram of 5-axis TTTRR CNC machine	29
II.2	Coordinate systems for the kinematic chain of the machine	29
II.3	Concepts of error compensation proposed approaches	44
II.4	Main phases of the STL approach	45
II.5	Difference between compensation of finishing tool path and compensation of finishing and previous tool paths	45
III.1	Comprehensive 5-Axis Error Simulation overview	67
III.2	Volumetric error	68
III.3	Positive and negative errors	69
III.4	Decomposed X, Y, and Z spatial deviations mapped across the toolpath	69
III.5	Convergence history of the decoupled iterative solver	70

III.6 3D Tool-Tip Trajectory for the complex multi-axis simulation	71
III.7 Axis-by-axis spatial deviation for the complex trajectory	72
III.8 Convergence history for the complex multi-axis toolpath	73

List of Tables

I.1	Detailed distribution of the 43 geometric errors on a 5-axis machine.	17
I.2	Classification, severity, and measurability of primary kinematic errors.	21

Abbreviations list

Acronyme	Signification
ESSAT	École Supérieure en Sciences Appliquées.
NC	Numerical Control.
CNC	Computer Numerical Control.
CAD	Computer-Aided Design.
CAM	Computer-Aided Manufacturing.
TCP	Tool Center Point.
HTM	Homogeneous Transformation Matrices.
D-H	Denavit-Hartenberg.
WCS	Workpiece Coordinate System.
IK	Inverse Kinematics.
FK	Forward Kinematics.
DBB	Double Ballbar.
CMM	Coordinate Measuring Machines.
OAC	Open Architecture Controller.
PVT	Position-Velocity-Torque.
RMS	Root-Mean-Square.
STL	Stereolithography.
CSV	Comma-Separated Values.
G-code	Geometric Code.

General Introduction

In recent years, the increasing demand for high-precision manufacturing has accelerated the development of advanced computer-controlled machining systems. Among these, Computer Numerical Control (CNC) machines have become essential tools in modern industry due to their ability to produce complex parts with high repeatability, flexibility, and productivity. Their use extends across a wide range of sectors, including aerospace, automotive, mold manufacturing, medical devices, and prototyping, where dimensional accuracy and geometric conformity are critical requirements.

Although CNC technology has significantly improved manufacturing capabilities, the accuracy of machined parts remains limited by the presence of various errors affecting the machine-tool system. In practice, deviations between the nominal geometry defined by the CAD model and the actual geometry produced on the machine are inevitable. These deviations may result from geometric imperfections, kinematic misalignments, thermal effects, structural deflections, control limitations, or measurement uncertainties. As a consequence, the final workpiece may not fully match the intended design, especially in multi-axis machining where the tool orientation and position change simultaneously.

Five-axis CNC machines are particularly sensitive to these issues because of the complexity of their kinematic structure. Their additional rotational axes provide major advantages in terms of accessibility, machining flexibility, and reduction of setups, but they also introduce new sources of error propagation. Even small imperfections in axis alignment, rotary center position, or tool center point motion can lead to significant volumetric deviations at the workpiece level. For this reason, understanding and compensating machining errors has become a key topic in improving the performance of five-axis CNC systems.

This thesis is devoted to the study of error compensation in a five-axis CNC milling machine prototype. The work builds upon the previous design and fabrication phase of the prototype and focuses on the analysis of machining deviations through a simulation-based approach. Since the machine has not yet been experimentally validated through full machining tests, the present study emphasizes mathematical modeling and numerical simulation as a foundation for future experimental verification.

Project Overview

This project investigates the influence of geometric and kinematic errors on the machining accuracy of a five-axis CNC milling machine prototype. The study aims to model the machine kinematics, introduce representative volumetric errors, and evaluate their effect on the final geometry of a machined part. A simulation-based compensation strategy is then developed to reduce the discrepancy between the nominal CAD model and the simulated machined result. The work is implemented mainly in MATLAB, with the objective of providing a clear and structured framework for analyzing and correcting machining deviations.

Problem Statement

Despite the advantages of five-axis CNC machining, real machine tools never behave as perfectly as their theoretical models. Small geometric and kinematic imperfections accumulate along the kinematic chain and generate positioning errors at the tool center point, which in turn affect the accuracy of the machined part. In the absence of a compensation strategy, these deviations may remain unnoticed until inspection, making it difficult to achieve the desired level of precision. The problem addressed in this thesis is therefore the mismatch between the nominal part geometry and the geometry obtained under the influence of machine-tool errors, and the need for a reliable method to compensate for these deviations.

Research Objectives

The main objectives of this thesis are as follows:

- To study the theoretical foundations of CNC machining accuracy, precision, and geometric deviation.
- To identify and classify the main sources of error affecting five-axis CNC machine tools.
- To develop a kinematic model of the machine prototype suitable for simulation and error analysis.
- To introduce representative geometric and volumetric errors into the model in order to evaluate their influence on the tool path and the final workpiece geometry.
- To investigate simulation-based compensation strategies capable of reducing the deviation between the nominal and actual geometries.
- To compare the machining result before and after compensation and assess the effectiveness of the proposed approach.

Scope of the Project

The scope of this thesis is limited to the modeling and simulation of machining errors in a five-axis CNC milling machine prototype. The study focuses primarily on geometric and kinematic error sources and their offline compensation through mathematical and computational methods. Experimental machining tests, advanced dynamic modeling, cutting-force analysis, and thermal deformation characterization are beyond the scope of this work. The aim is not to develop a full industrial compensation system, but rather to provide a clear and practical simulation framework that can later support experimental validation and further refinement.

This thesis is organized as follows:

- **Chapter 1:** Theoretical foundations of CNC machining accuracy, machining errors, measurement methods, and compensation strategies.
- **Chapter 2:** Kinematic modeling and mathematical formulation of volumetric errors in five-axis CNC systems.
- **Chapter 3:** MATLAB implementation of the proposed compensation approaches and discussion of the simulation results.
- **Annex:** Source codes and supplementary material related to the simulation and error-compensation procedures.

Through this work, the thesis seeks to contribute to a better understanding of how geometric and kinematic imperfections affect five-axis CNC machining, while also demonstrating how simulation-based compensation can be used to improve machining accuracy in a compact prototype environment.

CNC FUNDAMENTALS AND PERFORMANCE CRITERIA

Introduction

Computer Numerical Control (CNC) machining occupies an essential place in modern manufacturing thanks to its ability to produce parts with high repeatability, improved productivity, and good dimensional quality. Over the years, CNC technology has evolved from basic numerical control systems into highly advanced machining platforms capable of performing complex operations with multiple degrees of freedom. This evolution has made CNC machines widely used in industrial and educational environments, particularly in applications requiring precision, flexibility, and reliability.

Before studying machining errors and compensation strategies, it is necessary to establish the fundamental concepts related to CNC technology. This chapter, therefore, introduces the historical development of CNC systems, their basic definition, their structural organization, and the main classifications used to distinguish between different machine configurations. The chapter also presents the principal industrial applications of CNC machines, as well as their advantages and limitations in practical use.

In addition to this general background, the chapter lays the theoretical basis required for the rest of the thesis by introducing the concepts of accuracy, precision, repeatability, and other performance metrics used to evaluate machining quality. These concepts are essential for understanding the nature of machining deviations and for identifying the sources of error that affect the final geometry of the manufactured part. They also provide the necessary foundation for the modeling and compensation methods developed in the following chapters.

I.1 Concept of computer numerical control

CNC technology plays a central role in modern manufacturing because it allows machine tools to execute motion commands with a high level of repeatability, flexibility, and dimensional consistency. Unlike manual machining, where the operator directly guides the cutting process, CNC machining relies on digitally stored instruc-

tions that define the tool movement, cutting conditions, and auxiliary operations. This makes CNC systems suitable for producing parts with complex shapes and tight tolerances, especially in applications where accuracy and productivity are essential.

I.1.1 History

Numerical control began as a practical solution to the need for more precise and repeatable machining processes. Before the development of NC systems, machine tools depended heavily on manual operation, which limited both productivity and geometric consistency. The first numerical control concepts appeared as a way to replace manual intervention by using coded instructions that could direct machine movement automatically.operation.[10]

Early NC systems were based on punched cards or punched tape, which stored the sequence of commands needed to guide the machine. These systems were initially limited by the available electronic technology, but they represented a major step toward automated manufacturing. As control electronics and computing systems evolved, the mechanical and hardwired logic of NC was progressively replaced by software-driven controllers. This transition led to Computer Numerical Control (CNC), which made it possible to store, edit, and execute machining programs more efficiently while offering greater flexibility in operation.[10, 38].

The development of CNC technology accelerated the integration of machine tools with computer-based design and manufacturing methods. As a result, CNC machines became essential in industries requiring high precision, reduced setup time, and reliable repeatability. Today, they are widely used in both industrial production and educational environments.[8]

I.1.2 CNC machines classifications

CNC machines can be classified according to several criteria, including the number of controlled axes, the type of motion system, and the machine configuration. Among these criteria, the number and arrangement of axes are among the most important because they determine the range of motion available to the tool and the workpiece.

I.1.2.1 3-axis

A 3-axis CNC machine controls motion along the three linear axes, generally designated as X, Y, and Z. This configuration is the most common and widely used in machining because it provides a simple and efficient solution for a large variety of parts. In a 3-axis machine, the tool orientation remains fixed during machining, while the tool tip moves in space to follow the programmed path.

3-axis machines are particularly suitable for prismatic parts, pockets, slots, flat surfaces, and many standard machining operations. Their main advantages are simplicity, lower cost, and easier programming compared with multi-axis systems.

However, their limited orientation capability makes them less effective for machining complex freeform surfaces or geometries that require access from multiple directions.

I.1.2.2 5-axis

A five-axis CNC machine extends the motion capability by adding two rotational axes to the three linear ones. This additional freedom makes it possible to orient the cutting tool or the workpiece in multiple directions, which is especially useful for machining complex surfaces, undercuts, and inclined features. Five-axis machining can also reduce the number of setups required to complete a part, thereby improving productivity and reducing positioning errors caused by repeated repositioning. There are two common practical modes of using five axes:

- **3+2:** also called positional 5-axis, two rotational axes are used to reorient (tilt) the tool, or the part, and then lock it; the three linear axes then perform the cutting. This mode is effectively a series of 3-axis cuts with different tool orientations and is much simpler to program and safer in terms of collisions.
- **Simultaneous 5-axis contouring:** the controller moves all five axes simultaneously so the cutter can trace complex surfaces while continuously changing orientation. In simultaneous 5-axis the controller computes coordinated trajectories for five actuators so the tool tip follows the programmed spatial path while the tool axis rotates to maintain the desired cutting direction. This involves two linked computations in real time, translation interpolation to position the tool tip along the path, and then kinematic transformation (inverse kinematics) that converts the desired tool pose (position + orientation) into the specific motor angles and linear positions for each axis. Simultaneous 5-axis, therefore, requires more advanced CAM toolpath generation, higher controller processing power, careful collision checking, and often more careful fixturing and low-profile tooling. [5]

Depending on the mechanical arrangement of the machine, five-axis systems are commonly divided into several configurations.

Head-Head: In a head-head configuration, both rotational axes are located in the spindle head. The tool orientation is therefore controlled by the head itself, while the workpiece generally remains fixed on the table. This arrangement provides excellent access to the workpiece and allows the tool to approach the part from many directions. It is particularly suitable for complex geometries that require high orientation flexibility. [27]

However, head-head machines also present certain limitations. Since the rotary components are located in the spindle head, the moving mass of the upper assembly increases. This can reduce dynamic performance and place higher demands on the machine structure. In addition, maintaining accuracy may become more challenging if the head is not sufficiently rigid. [34]

Table-Table: In a table-table configuration, both rotational axes are placed on the worktable. The part is oriented by rotating the table, while the spindle remains primarily responsible for the cutting motion. This configuration often provides good rigidity because the rotary elements are supported by the table structure rather than by the head. [27]

Table-table machines are well-suited for parts that require stable support during machining. However, this arrangement usually requires low-profile fixtures to avoid collisions between the rotating part and the spindle. The rotating table must also support the inertia of the workpiece, which can limit the machining of heavy parts or increase the cost of the rotary system. [6]

Head-Table: In a head-table configuration, one rotational axis is located in the spindle head and the other in the table. This hybrid solution combines the advantages of both previous configurations by distributing the orientation motions between the tool side and the workpiece side. As a result, it offers a balance between accessibility, rigidity, and machine flexibility. [27]

Head-table machines are often considered a practical compromise for many industrial applications. They can provide good machining accessibility while maintaining a more manageable head mass than head-head systems. At the same time, the presence of rotary motion on both the head and the table requires careful kinematic calibration and accurate control. [34]

I.1.3 CNC machines applications

CNC machines are used in a wide range of industrial sectors because they can produce parts with good accuracy, repeatability, and surface quality. Their ability to execute complex toolpaths makes them suitable for manufacturing components that are difficult or impossible to produce efficiently using manual machining methods.

In aerospace, CNC machines are used for structural parts, engine components, and complex surfaces that require high precision. In the automotive industry, they are used for engine blocks, transmission parts, tooling, and prototypes. In electronics, CNC machining is used for housings, fixtures, and precision components. CNC technology is also widely applied in mold making, medical device manufacturing, and prototype development, where dimensional consistency and geometric conformity are essential.

The broad range of applications demonstrates the importance of CNC technology in modern production systems. Its flexibility makes it suitable for both small-scale manufacturing and high-volume industrial production. [23, 7]

I.1.4 Advantages and limitations

CNC machines offer several important advantages that explain their widespread use. First, they provide high repeatability, which makes it possible to manufacture

similar parts with consistent quality over multiple production cycles. Second, they improve productivity by reducing manual intervention and allowing machining operations to be executed automatically. Third, they are capable of producing complex geometries with a level of precision that is difficult to achieve with conventional manual methods. In addition, CNC systems integrate naturally with digital design and manufacturing tools, which makes them highly compatible with modern CAD/CAM workflows. [33]

Despite these advantages, CNC machines also present some limitations. Their acquisition cost can be high, especially for multi-axis systems. Programming can also be complex, particularly in five-axis machining, where tool orientation, collision avoidance, and kinematic transformations must all be considered. Furthermore, machine accuracy depends on proper calibration, mechanical rigidity, and control system performance. These limitations become more pronounced as the number of axes increases, which is why error analysis and compensation are important topics in advanced CNC research. [2]

I.2 Manufacturing performance metrics

The performance of a manufacturing process cannot be judged by a single numerical value. A machine may produce identical parts that all fail to meet design tolerances, or it may occasionally hit the target dimension without being reproducible. For this reason, machining performance is evaluated through two fundamental and complementary metrics: precision and accuracy. Precision reflects the internal consistency of the process (variance), while accuracy reflects its conformity to the nominal CAD reference (offset). [25]

I.2.1 Precision

Precision describes the degree of closeness between repeated results obtained under specified conditions. It measures the internal stability of the kinematic chain and the control system. A highly precise CNC machine reproduces the exact same toolpath and part geometry with minimal dispersion, regardless of whether those dimensions are correct. Precision is a measure of variation and is quantified through two sub-metrics: [25, 4]

- **Repeatability:** The ability to produce nearly identical results under identical conditions (same machine, setup, operator, and short time interval). It reflects the short-term reliability of the machine against local disturbances like servo fluctuations or backlash.
- **Reproducibility:** The ability to maintain consistent results when operating conditions change (different setups, operators, or longer time periods). It reflects the broader, long-term robustness of the manufacturing process. [4]

I.2.2 Accuracy

Accuracy describes the closeness of a result to the true nominal value. While precision is a measure of data scatter, accuracy is a measure of correctness. A machine may be perfectly precise, yet inaccurate if a geometric flaw introduces a persistent, systematic offset into the toolpath. Assessing accuracy requires evaluating three properties: [25, 4]

- **Bias:** The systematic deviation between the average measured value and the true target. Bias is the primary indicator of inaccuracy (e.g., geometric misalignments or calibration errors) and is the main target for kinematic error compensation.
- **Stability:** The ability to preserve accuracy over time. A stable system resists gradual dimensional drift caused by thermal expansion or continuous mechanical wear.
- **Linearity:** The consistency of the error across the entire operating range. If a machine's deviation changes depending on the axis position, the required compensation strategy must be spatially varying rather than uniform.[4]

I.2.3 The relationship between precision and accuracy

Precision and accuracy answer two distinct questions: *does the machine behave consistently?* (Precision), and *does the machine behave correctly?* (Accuracy).

To clearly conceptualize how these two metrics interact, machine performance is frequently compared to throwing darts at a target, as illustrated in Figure I.1. The center of the bullseye represents the nominal CAD dimension, while the spread of the darts represents the machining dispersion. This yields four distinct manufacturing scenarios:

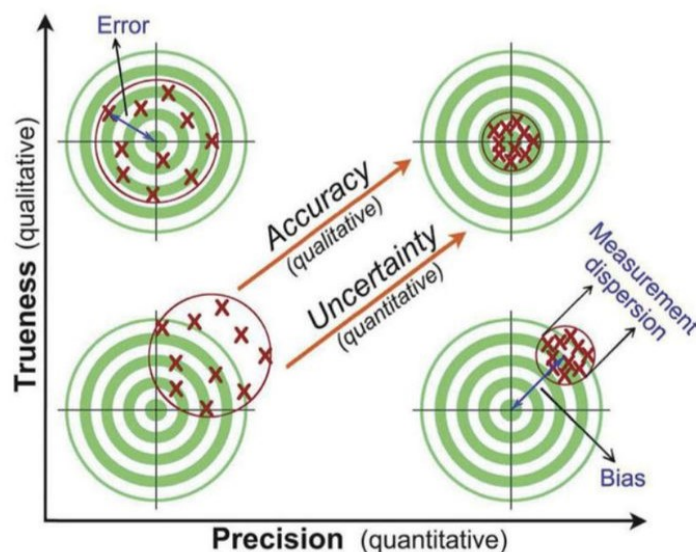


Figure I.1: Illustration of the relationship between precision and accuracy

- **High Accuracy and High Precision:** The machined parts are consistently identical to each other and perfectly match the nominal dimensions. This is the ideal manufacturing state.
- **Low Accuracy and High Precision (Systematic Error):** The parts are virtually identical to each other, but they all deviate from the nominal dimension by a constant offset. This tight grouping indicates a highly repeatable machine suffering from a predictable geometric flaw.
- **High Accuracy and Low Precision (Random Error):** The average dimension of a batch of parts matches the nominal target, but the individual dimensions scatter wildly around it. This indicates a lack of process stability, often due to unpredictable vibration or chatter.
- **Low Accuracy and Low Precision:** The parts are neither identical to each other nor close to the nominal target, indicating a completely uncontrolled process.

This distinction is central to 5-axis error compensation. If a machine is accurate but imprecise (random error), the scatter requires physical hardware interventions, such as increasing frame stiffness or reducing spindle chatter. Conversely, if a machine is highly precise but inaccurate (systematic error), the constant bias is exactly what offline mathematical compensation strategies aim to isolate and eliminate. By mapping these predictable kinematic flaws, the algorithms developed in this thesis mathematically shift the tightly grouped, precise results directly onto the nominal target, effectively achieving true accuracy without physically altering the machine. [25, 4]

I.3 Design boundaries: allowance vs. tolerance

Manufacturing is the process of translating a perfect digital design into an imperfect physical reality. Because even the most advanced CNC machines cannot produce totally exact dimensions, engineers rely on two distinct concepts to control these physical deviations: tolerance and allowance. While these terms are frequently mixed up or used synonymously in machine shops, they serve entirely different purposes and represent a critical distinction between design intent and manufacturing capability. [24]

I.3.1 The concept of allowance

Allowance is an intentional design feature that dictates exactly how two mating components connect. It is the planned dimensional difference between parts, purposefully engineered to achieve a specific mechanical fit. Depending on the functional requirements of the assembly, engineers generally apply one of three main types of allowance: [24]

- **Clearance fit:** A positive allowance intentionally designed to ensure a gap remains between the mating parts, allowing for free movement or rotation.

- **Interference fit:** A negative allowance designed to ensure the parts overlap in dimension, requiring force to assemble them and creating a rigid, permanent bond.
- **Transition fit:** A calculated balance between clearance and interference, providing a highly accurate alignment where parts can be assembled and disassembled without excessive force.

I.3.2 The concept of tolerance

While allowance is a planned design choice for how parts assemble, tolerance applies to an individual part and represents unplanned, unavoidable manufacturing deviation. Because precise multi-axis CNC machines inevitably experience kinematic errors, perfect dimensions are physically impossible. Tolerance defines the acceptable range within which a part's dimension can deviate without compromising its intended performance. To control this deviation, designers specify tolerances in three primary ways:[24, 36, 25]

- **Unilateral tolerance:** The permissible deviation is allowed in only one direction from the nominal size (either exclusively larger or exclusively smaller).
- **Bilateral tolerance:** The permissible deviation is split symmetrically or asymmetrically, allowing the final dimension to be slightly larger or slightly smaller than the target.
- **Limit dimensions:** Instead of showing a target dimension and a $\pm$ deviation, the drawing simply states the absolute maximum and minimum acceptable physical sizes (e.g., upper limit 50.1 mm, lower limit 49.9 mm).

I.3.3 Interaction between allowance and tolerance

While they are distinct concepts, tolerance and allowance must be managed simultaneously whenever two parts are assembled. If a single part is being machined in isolation, only tolerance matters. However, in an assembly, the individual tolerances of the components directly attack the shared allowance.

For example, if a shaft and a hole are designed with a clearance fit (a positive allowance), both parts still possess their own manufacturing tolerances. If the shaft is machined to its maximum allowable tolerance (making it as thick as possible) and the hole is machined to its minimum allowable tolerance (making it as narrow as possible), the intended gap shrinks. This is known as a worst-case tolerance stack-up.

If the tolerances are too loose, the parts might physically collide, turning a planned clearance fit into an accidental interference fit. To prevent this, the worst-case scenario must be calculated to ensure that even at the extreme edges of the tolerance zones, the required physical allowance is perfectly preserved.

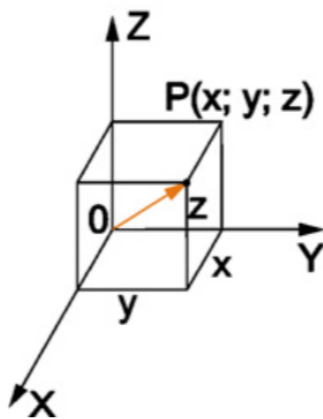
I.4 Kinematic structures of machine tools

Before analyzing the geometric deviations of a multi-axis CNC machine, it is essential to understand the fundamental kinematic structures that govern its physical motion. A machine structure consists of a supporting frame to which functional elements, such as drives and guides, are attached. The architecture of a machine tool is defined by its coordinate system and the mechanical arrangement of its structural loops. [26]

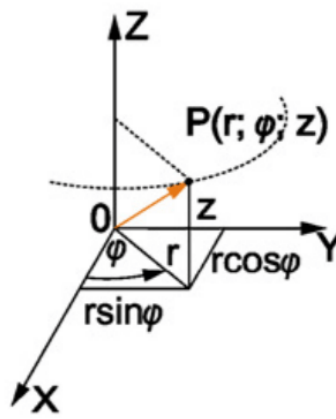
I.4.1 Coordinate systems

The spatial movement of the Tool Center Point (TCP) relative to the workpiece can be achieved through various fundamental coordinate systems, depending on the manufacturing application:

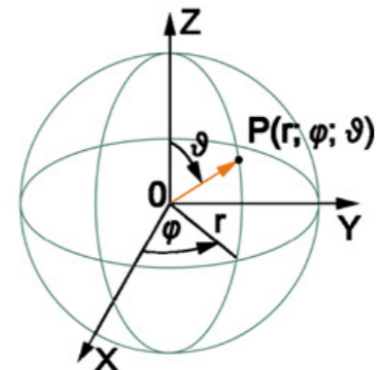
- **Cartesian (rectangular) coordinate system:** The most prevalent architecture for milling machines. Motion is generated along three mutually orthogonal linear axes (X, Y, and Z), as shown in Figure (a) I.2. This system offers highly predictable volumetric accuracy and forms the base of the prototype developed in this thesis.
- **Cylindrical and spherical coordinate systems:** Utilized primarily in specialized turning centers or articulated robotic arms, these systems rely on a combination of linear reach and rotational sweeping to define points in space, as shown in Figure (b) I.2 and (c) I.2.



(a) Cartesian coordinate system



(b) Cylindrical coordinate system



(c) Spherical coordinate system

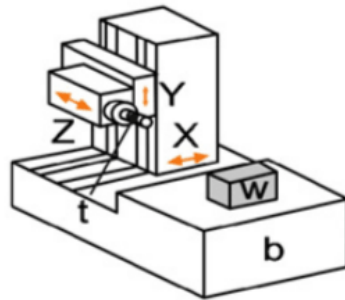
Figure I.2: Coordinate systems

I.4.2 Serial kinematic structures

Most traditional CNC machines employ a serial kinematic chain, where each axis is physically stacked upon the previous one. The kinematic chain of a serial structure

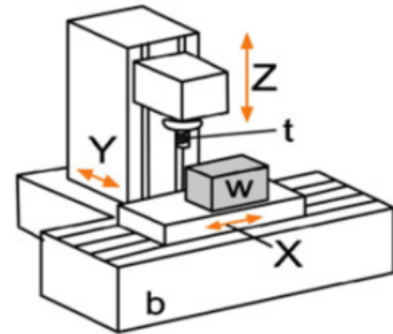
is only closed during the actual manufacturing process when the tool touches the part's surface. [26]

To formalize this, a notation based on Schwerd can be used to describe the serial chain from the tool (t) through the machine bed (b) to the workpiece (w). Within serial Cartesian machines, the structural frames and their corresponding chains are generally classified into several main architectures. [26]



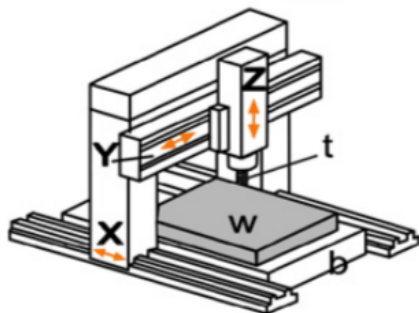
$t - (C) - Z - Y - X - b - w$

(a) Cantiliver type kinematic chain



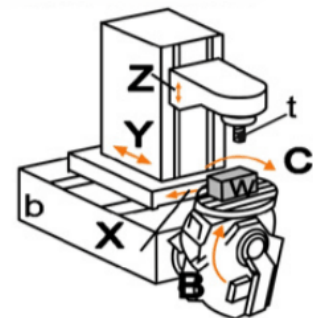
$t - (C) - Z - Y - b - X - w$

(b) Column type kinematic chain



$t - (C) - Z - Y - [X1 X2] - b - w$

(c) Gantry type kinematic chain



$t - (C) - Z - Y - X - b - B - C - w$

(d) Additional axes kinematic chain

Figure I.3: Possible kinematic chain configurations of different CNC machines

- **Cantilever (C-frame) type:** The vertical Z-axis, and often the Y and X axes, are supported by a single vertical column, as shown in Figure (a) I.3. While this provides excellent accessibility to the work table, the asymmetrical structural loop is highly susceptible to static bending (sag) under the weight of the spindle.
- **Column type:** The linear axes are split between the tool and the workpiece. For instance, the X-axis moves the work table while the Y and Z axes are mounted on a rigid vertical column, as shown in Figure (b) I.3. This split distributes the moving masses and dynamic forces more evenly across the machine bed.

- **Frame (portal or gantry) type:** The Z-axis is supported by a bridge spanning across two columns, as shown in Figure (c) I.3. This closed structural loop provides vastly superior rigidity and thermal stability, significantly reducing load-induced geometric errors [26].
- **Additional axes (e.g., 5-axis machines):** To transition from 3-axis to 5-axis machining, rotational joints (A, B, or C) are integrated into the existing serial chain. Depending on whether these rotary joints are stacked on the tool side (Head-Head), the workpiece side (Table-Table), or split between both (Head-Table), as shown in Figure (d) I.3, the kinematic chain drastically changes, directly altering how geometric errors propagate through the machine. [26]

I.5 Classification of machining errors

A machined part is the final physical output of a complex chain of data processing and mechanical execution. As illustrated in Figure I.4, deviations can be introduced at multiple stages along this chain, ranging from initial CAD/CAM programming and CNC interpolation down to the dynamic and kinematic performance of the machine's structural axes. [17]

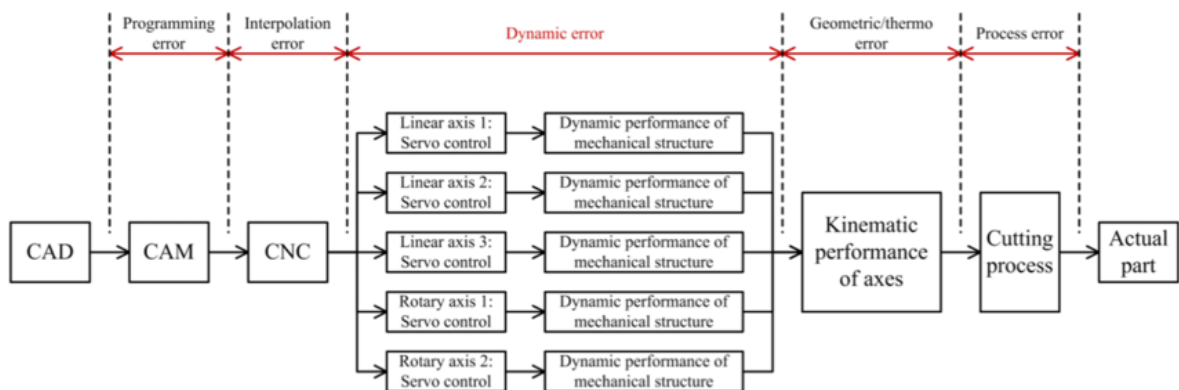


Figure I.4: Error generation chain from CAD model to actual part execution

The total measurable error of a CNC machine is not caused by a single isolated flaw, but is rather the cumulative interaction of several distinct sources. These are broadly categorized into geometric errors, thermal errors, load-induced errors, and other external factors (such as vibration or control system limitations), which continuously influence one another during the machining cycle (Figure I.5). To effectively diagnose and compensate for these deviations, they must be classified by their behavior into two primary categories: random errors and systematic errors. [1]

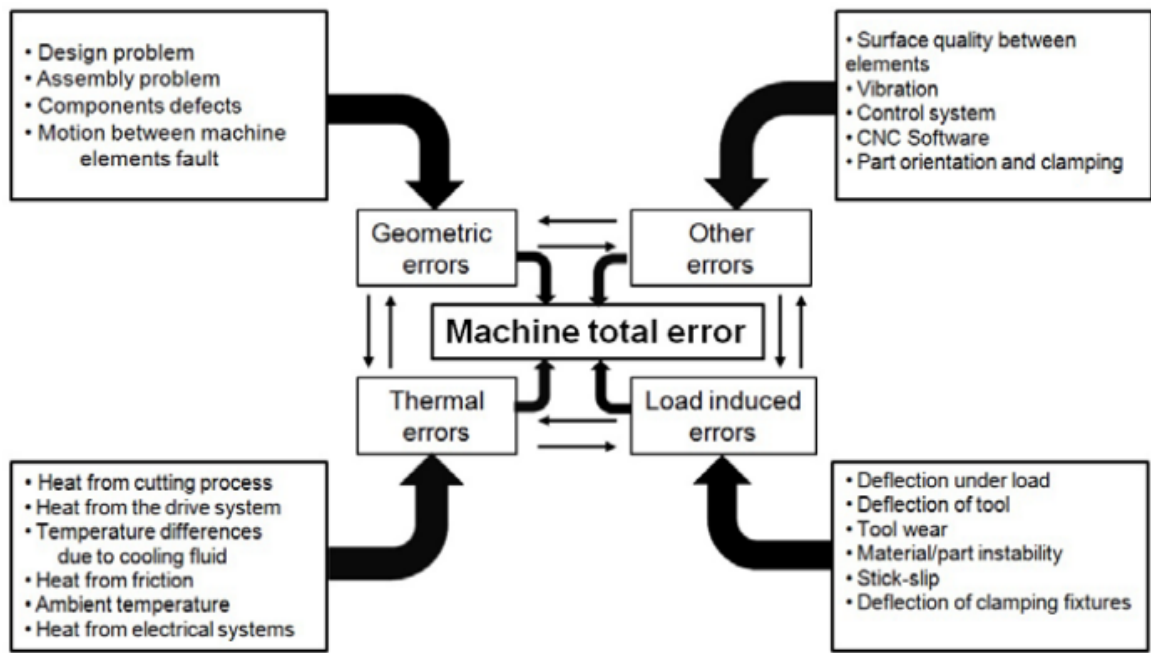


Figure I.5: The main sources of machining errors

I.5.1 Random errors: the limits of precision

Random errors are non-stationary, unpredictable deviations that change from one machining cycle to the next. Because their magnitude and direction fluctuate, they directly degrade the precision (repeatability) of the CNC machine. These errors generally cannot be neutralized through offline mathematical preprocessing and require physical hardware interventions or active, real-time closed-loop control to mitigate. [1, 20]

- **Dynamic errors:** High-frequency phenomena such as tool chatter, spindle vibration, and structural resonance that occur during the active cutting process.
- **Electrical/control noise:** Fluctuations in servo drive response, encoder resolution limits, interpolation rounding errors, and communication lag within the control hardware.
- **Environmental:** Unpredictable variations in ambient shop-floor conditions or external vibrations transmitted through the facility foundation.

I.5.2 Systematic errors: the limits of accuracy

Systematic errors are quasi-static, consistent deviations that continuously pull the tool off target in a predictable manner. These errors degrade the absolute accuracy of the machine. Because they are highly repeatable, systematic errors are the primary target for the offline volumetric compensation strategies developed in this thesis. [1, 20]

- **Thermal errors:** Heat generated internally by the drive systems and spindle friction produces a predictable, systematic expansion of the machine frame. *(Note: While primarily systematic, thermal errors can exhibit random behavior if subjected to sudden ambient temperature drafts or irregular coolant flow).*
- **Static/weight errors (load-induced):** The elastic deformation of the machine frame due to the constant weight of the heavy bi-rotary head is static and systematic. *(Note: Deflection caused by varying cutting forces, such as encountering hard spots in raw material, acts as a random error).*
- **Kinematic and geometric errors:** The inherent structural flaws arising from the imperfect manufacturing, assembly, and alignment of the machine's guide-ways and rotational joints.

To successfully construct the virtual kinematic model of the bi-rotary customized configuration, the offline compensation algorithm focuses specifically on the most severe geometric and kinematic errors that cause volumetric deviation. [30]

I.6 5-axis CNC machine's total error components

Adding two rotary axes to a 3-axis machine does not merely increase its cutting flexibility, it exponentially multiplies its kinematic complexity. A standard 3-axis Cartesian machine possesses 21 geometric error components. However, a 5-axis configuration accumulates a total of 43 geometric errors.

This massive proliferation is caused by the inherent flaws of the rotary joints. Each individual rotary axis introduces 11 specific deviations: 6 that are dependent on the instantaneous angular position (dynamic motions) and 5 that are position-independent (static setup offsets). Table I.1 provides a complete, decoupled breakdown of how these 43 volumetric components are distributed across the machine's kinematic chain.

Table I.1: Detailed distribution of the 43 geometric errors on a 5-axis machine.

Error classification	Count
Linear axes (X, Y, Z)	21
Positioning errors (1 per axis)	3
Straightness errors (2 perpendicular drifts per axis)	6
Angular errors (1 per axis):	9
- Pitch (1 per axis)	3
- Roll (1 per axis)	3
- Yaw (1 per axis)	3
Squareness/orthogonality errors (between XY, YZ, ZX)	3
Rotary axes (A and C)	22
<i>Position-dependent errors (6 per axis):</i>	12
- Angular positioning error (1 per axis)	2
- Radial error motion (straightness drift) (2 per axis)	4
- Axial error motion (thrust drift) (1 per axis)	2
- Tilt error motion (wobble) (2 per axis)	4
<i>Position-independent / setup errors:</i>	10
- Orientation errors (squareness to linear axes)	4
- Location errors (spatial X, Y, Z pivot offsets)	6
Total machine geometric errors	43

In the specific context of a Head-Head configuration, controlling these rotary errors is absolutely critical due to the geometric leverage effect, universally known as the Abbé error. Because the machine table is completely fixed and all rotational kinematics are mounted on the spindle head, a small angular deviation (for example, a 0.001° tilt error motion due to the physical overhang of the heavy head) amplifies exponentially over the entire length of the tool. This microscopic tilting at the pivot point translates into a massive positioning discrepancy of several tenths of a millimeter at the Tool Center Point (TCP). [3]

I.6.1 Visualizing geometric component and location errors

The description of geometric deviations begins with the strict assumption of rigid body behavior [26]. Every individual axis in a serial kinematic chain is considered a rigid body moving through 3D space. Imperfect manufacturing of the guideways forces this body to experience exactly six degrees of freedom of geometric deviation during its commanded motion [26].

I.6.1.1 Component errors of a linear axis

For a nominally linear movement, the six component errors are the positioning error, two straightness error motions, a roll error motion, and two tilt error motions (pitch and yaw) [26, 12].

For a carriage moving along the X-axis, these are defined as:

- **Positioning error:** The linear deviation along the commanded axis of travel (X).
- **Horizontal and vertical straightness errors:** The two linear deviations perpendicular to the direction of travel (drifting into the Y and Z axes). Straightness errors do not have a linear term, as orientation errors are considered separately [26, 12].
- **Pitch, roll, and yaw errors:** The three parasitic rotational deviations occurring around the X, Y, and Z axes as the carriage twists along the imperfect guideway.

Figure I.6 illustrates the physical manifestation of these linear and angular deviations on a moving carriage.

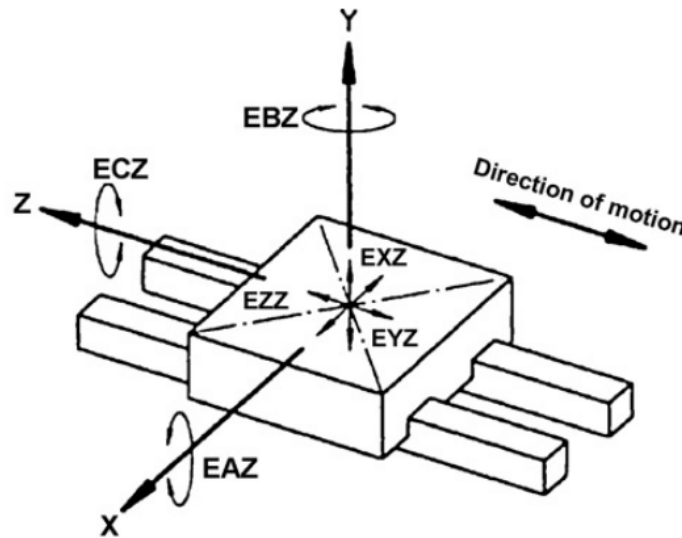


Figure I.6: Component errors of linear horizontal Z axis

Additionally, a linear motion axis is mathematically defined by a vector and a zero position, resulting in exactly three static location errors [26, 12]:

- **Zero position:** The linear offset of the axis starting point along its intended vector.
- **Orientation (squareness) errors:** The two macroscopic angular misalignments between the axis vector and the other two orthogonal reference planes (e.g., the squareness of the Z-axis relative to the X and Y axes).

Figure I.7 illustrates these linear axes location errors.

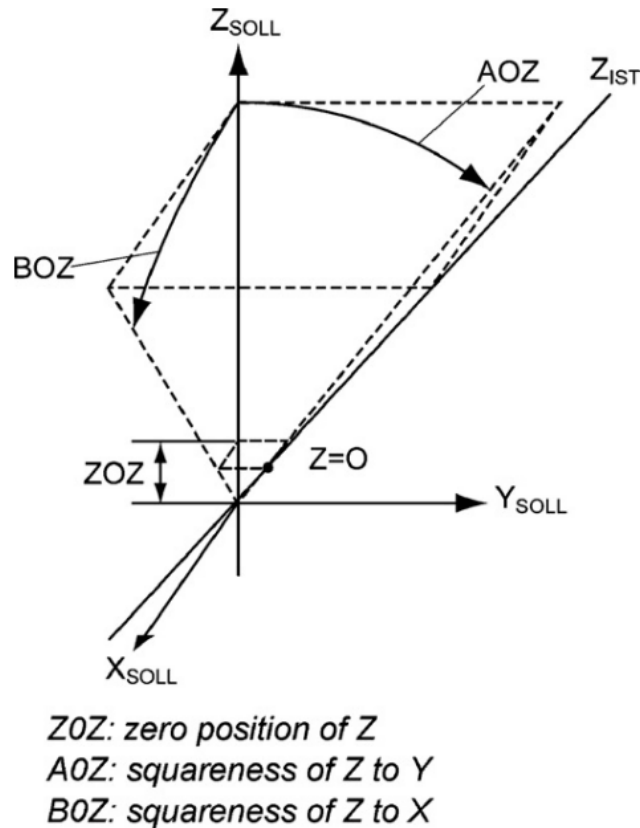


Figure I.7: Location errors of a Z movement

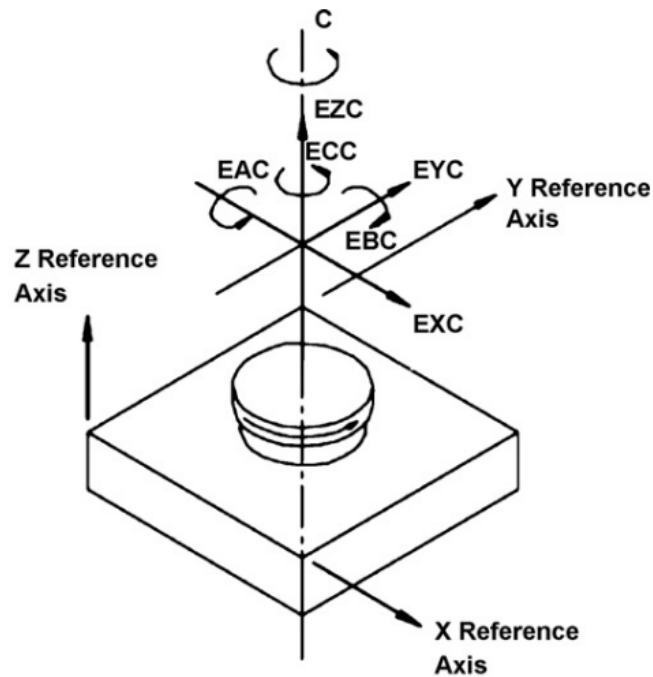
I.6.1.2 Component errors of a rotational axis

Similarly, for a nominal rotational movement (such as the A or C axis mounted on the spindle head), the rigid body also experiences six dynamic component errors during its rotation: two radial error motions, one axial error motion, the angular position error, and two tilt error motions [26, 12].

For a head rotating around the Z-axis (C-axis), these are defined as:

- **Angular positioning error:** The deviation in the actual rotational angle compared to the commanded angle of the motor.
- **Radial error motions (X and Y):** Two translational straightness drifts occurring perpendicular to the axis of rotation, causing the pivot point to shift laterally as it turns.
- **Axial error motion (Z):** A translational thrust drift occurring parallel to the axis of rotation, causing the spindle head to plunge or retract slightly during rotation.
- **Tilt error motions (Wobble):** Two parasitic angular twisting motions around the X and Y axes, causing the tool vector to physically wobble rather than spinning perfectly true. [26, 12]

Figure I.8 illustrates the physical manifestation of these rotational and angular deviations.



EXC: radial error motion of C in X direction
EYC: radial error motion of C in Y direction
EZC: axial error motion of C
EAC: tilt error motion around X of C
EBC: tilt error motion around Y of C
ECC: angular positioning error

Figure I.8: Component errors of rotational C axis

The five static location errors for a rotational axis are defined as [26, 12]:

- **Zero angular position:** The initial rotational offset (e.g., the physical sag or home sensor deviation when the controller reads 0°).
- **Position errors:** The two linear spatial offsets (X and Y) of the axis's physical pivot point compared to the machine's theoretical origin.
- **Orientation (squareness) errors:** The two angular misalignments of the rotational axis of symmetry relative to the linear reference planes. [26, 12]

Figure I.9 illustrates these rotational axes location errors.

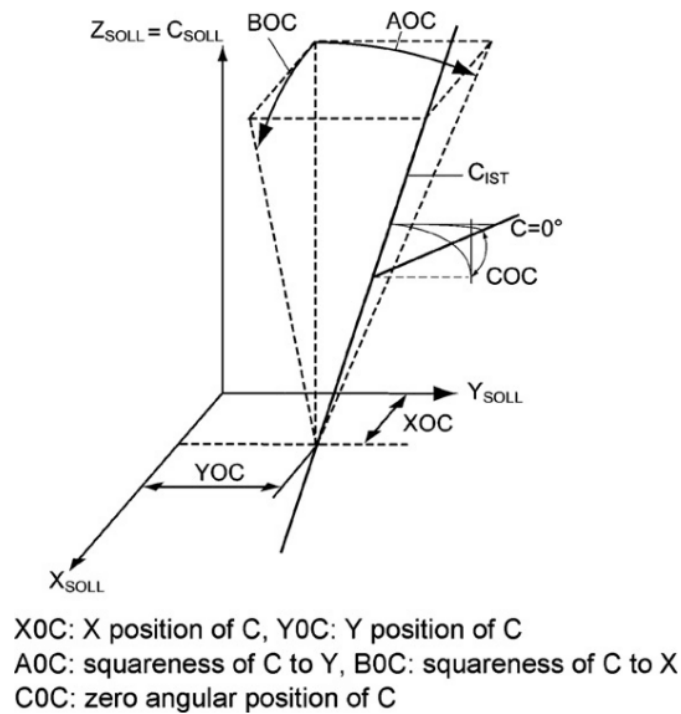


Figure I.9: Location errors of a C axis

A condensed summary of these critical geometric deviations, classified by their severity and typical measurability, is presented in Table I.2. [19]

Table I.2: Classification, severity, and measurability of primary kinematic errors.

Classification	Severity	Measurability
<i>Systematic errors</i>		
Positioning errors	high	easy
Straightness errors of each axis	high	easy
Squareness error between two axes	high	easy
Pitch, roll, and yaw angular errors	medium	easy
Rotary axis zero-offset	high	medium
<i>Random errors</i>		
Backlash errors	medium	medium
Contouring/interpolation errors	medium	easy
Machine assembly/mounting errors	high	easy

I.7 Measurement methods in CNC machining

The foundation of any offline error compensation strategy is the accurate acquisition of physical data to populate the virtual kinematic model. To systematically categorize these techniques, modern metrology generally divides them into three distinct

tiers based on the kinematic complexity of the measurement.

I.7.1 Classification of measurement methods

I.7.1.1 Direct measurement

Direct measurement methods isolate a single axis, keeping all other axes stationary, to quantify individual geometric errors without interference from the rest of the kinematic chain [26].

- **Laser interferometry:** The industry standard for capturing positioning and straightness errors over long travel distances. It fires a laser beam along the moving axis, directly comparing the machine's traveled distance against the absolute wavelength of the laser light [26].
- **Autocollimators and angular interferometers:** Optical instruments that project a beam of light onto a moving mirror to measure minute angular deviations (pitch and yaw) as a carriage travels down its guideways [26].
- **Electronic precision levels/inclinometers:** Highly sensitive instruments used to directly measure angular deviations by reading the microscopic tilt of a carriage relative to the absolute reference of Earth's gravity [26].
- **Precision test mandrels and dial indicators:** Mechanical tools used to measure physical alignment by sweeping a dial indicator across a precision-ground steel cylinder.

I.7.1.2 Indirect measurement

Indirect measurement methods capture the combined, simultaneous interaction of multiple axes moving together in free space (without cutting material). By measuring this volumetric deviation at the Tool Center Point (TCP), the underlying axis interaction of the machine can be mathematically deduced. [26, 11]

- **The double ballbar (DBB) test:** A high-precision telescopic sensor magnetically attached between the spindle and the table. As the machine executes a simultaneous 2-axis circular interpolation, the DBB records the deviation from a perfect circle [26, 11].
- **The R-test (3D touch probe systems):** A sophisticated dynamic tracking method where a spindle-mounted 3D probe continuously measures its distance to a fixed precision sphere on the table while the rotary axes tilt [26, 11].

I.7.1.3 Finished test piece evaluation

Unlike kinematic measurements, this tier involves actual machining. The CNC executes a complex toolpath, embedding all of its volumetric errors directly into

the physical dimensions of the workpiece. The evaluation takes place completely outside of the CNC machine.

- **Coordinate measuring machines (CMM):** The highest standard of post-process measurement. The machined test piece is transferred to a standalone CMM, which uses a precision ruby-tipped tactile probe to measure exact depths, diameters, and surface profiles [26].
- **3D laser scanning and optical metrology:** Rapidly digitizing the machined part to generate a dense 3D point cloud. This cloud is overlaid onto the nominal CAD file to create a visual color-map showing exactly where the CNC machine over-cut or under-cut the material due to its geometric flaws.

I.7.2 Examples of geometric error measurement

With the measurement tools classified, they can be deployed to target specific geometric parameters within the kinematic model.

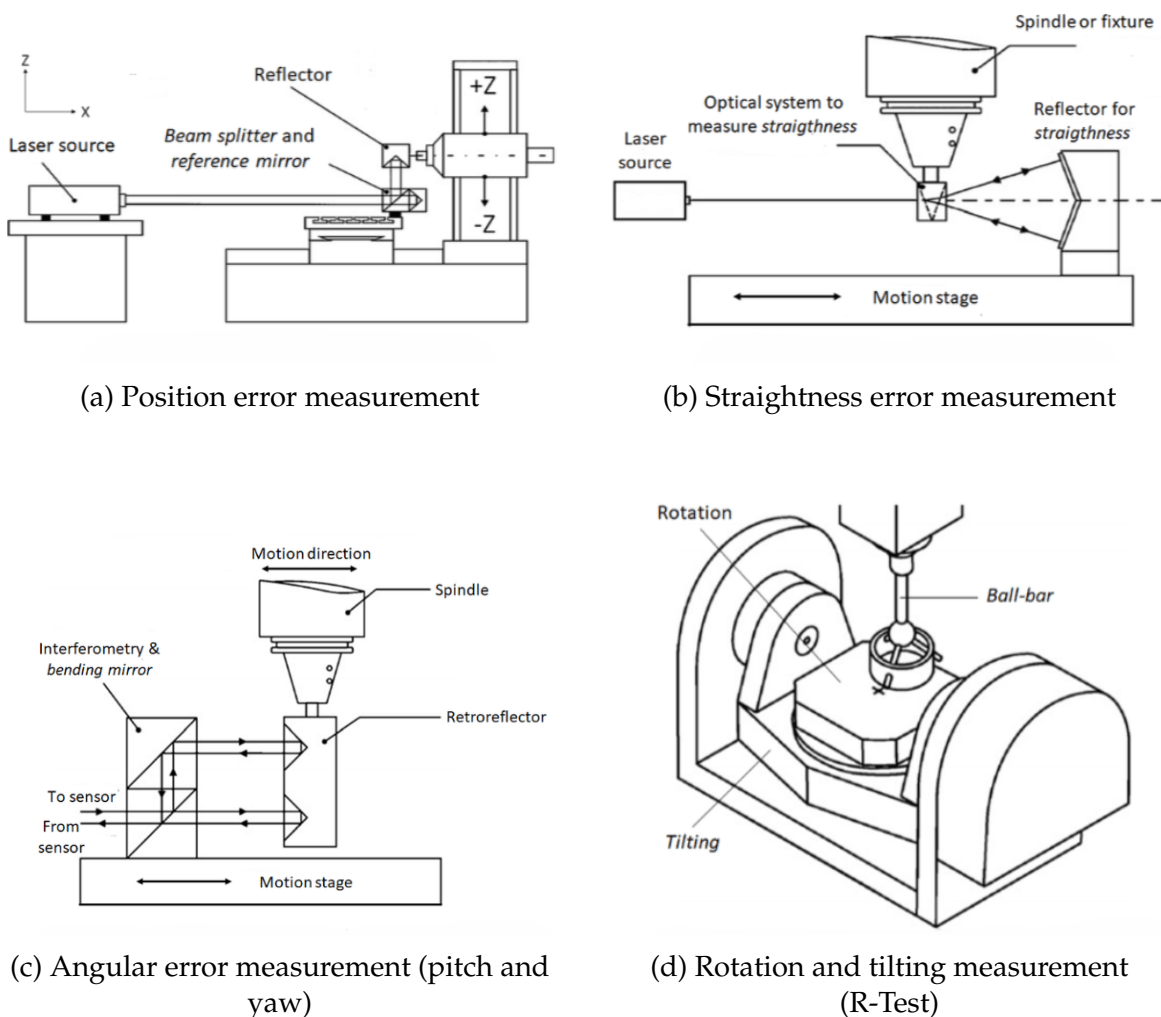


Figure I.10: Common kinematic measurement setups

- **Measurement of positioning errors:** For the direct mapping of linear positioning errors, a stabilized laser interferometer is typically aligned directly to the axis of interest, capturing the displacement between the completely fixed table and the moving spindle head [26].
- **Measurement of straightness errors:** Mapping straightness requires the measurement of lateral displacement perpendicular to the axis travel. This can be executed using a material reference (such as a straightedge) or by utilizing the linear propagation of a laser beam with straightness optics [26].
- **Measurement of angular and squareness errors:** Parasitic angular motions (pitch, roll, yaw) are captured using precision electronic levels or angular laser interferometers. For static squareness errors between two orthogonal axes, a double ballbar test is often executed to dynamically capture the continuous misalignment.
- **Measurement of rotary axes:** Evaluating a rotary axis isolates its radial, axial, and angular positioning errors. Statically, this is performed using precision test mandrels and high-resolution dial indicators. Dynamically, the R-test tracks a fixed precision sphere to capture the complex pivot offsets of the rotary joints as they move simultaneously [26].

I.8 Error mitigation and compensation strategies

Once the geometric errors of the machine tool have been measured and/or mathematically modeled, the final critical step in the manufacturing workflow is to minimize their impact on the workpiece. The strategies for improving spatial accuracy are fundamentally divided between physical machine interventions and advanced control algorithms.

I.8.1 Structural and kinematic calibration (Mechanical adjustment)

Mechanical adjustment represents the foundational tier of error mitigation. This process involves the physical realignment of machine components (such as scraping guideways, shimming rails, or mechanically preloading ball screws to physically eliminate backlash) to directly eliminate the root causes of kinematic inaccuracies. Direct measurement and subsequent mechanical adjustment during machine assembly are strictly essential for establishing a reliable baseline accuracy. [9]

A position compensation does not allow to remove certain angular deviation of the machine tool in some situations. Figure I.11 shows the effect of the parallelism error of the spindle axis on the work table. Suppose that the bottom face of the workpiece is the datum surface. If it is fixed parallel to the machine table, the parallelism error of the spindle axis to the work table can cause the perpendicularity error of the left side face to the bottom face. [9]

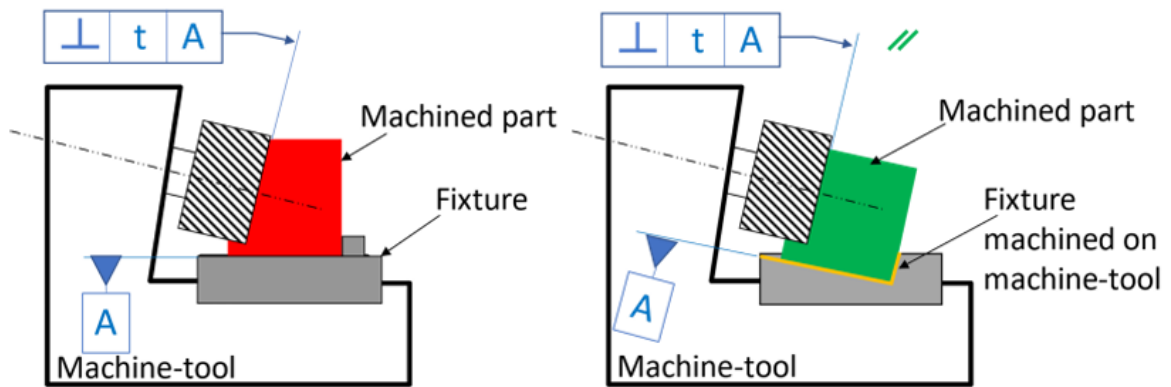


Figure I.11: Squareness error compensation

While this establishes the ultimate baseline for machine rigidity and thermal stability, structural calibration is fundamentally constrained by manufacturing costs and material physics. Once physical alignment reaches the natural tolerance limits of the mechanical components, further physical improvement becomes economically and practically impossible, necessitating the use of software compensation.

I.8.2 Controller-embedded numerical compensation

To surpass physical mechanical limits, most modern industrial CNC controllers (such as high-end Fanuc, Siemens, or Heidenhain systems) feature built-in firmware registers for numerical compensation. The machine builder maps the axis errors and inputs a static, discrete correction table directly into the CNC memory. During interpolation, the controller reads the active axis position, queries the look-up table, and injects additional microscopic encoder pulses to shift the physical position. [9, 16]

While highly effective for localized, one-dimensional errors (such as linear pitch error or simple backlash), standard controller architectures struggle with 5-axis volumetric compensation. Because the closed-architecture processors must manage the Position-Velocity-Torque (PVT) servo loops in real-time (often under 1 millisecond per cycle), calculating complex, non-linear cross-axis error interactions mathematically overwhelms the standard controller's processing overhead. [9, 16]

I.8.3 Algorithmic and model-based compensation

To address the multi-dimensional, complex volumetric errors inherent in 5-axis simultaneous CNC machining hardware, researchers have developed advanced algorithmic compensation models. These are divided into online (dynamic) and offline (pre-calculated) methodologies.

I.8.3.1 Online (real-time) dynamic compensation

Online compensation operates concurrently with the machining process, utilizing an active sensory network to detect deviations as they happen. This architecture typically pairs an Open Architecture Controller (OAC) with external metrology hardware, such as continuous laser trackers, spindle-mounted thermal sensors, or dynamometers. [9, 18]

A secondary industrial PC processes this live sensor data through a volumetric error model, calculates the instantaneous deviation at the Tool Center Point (TCP), and injects an inverse voltage or feed-forward signal directly into the servo amplifiers. Alternatively, active actuators mounted on the spindle head physically shift the tool position in real-time. While this method accommodates unpredictable environmental changes, it requires extremely expensive peripheral hardware, suffers from signal latency, and is highly susceptible to cutting-fluid interference and sensor noise. [9, 18]

I.8.3.2 Offline compensation

Offline compensation bypasses both the computational limits of the CNC controller and the environmental hazards of online sensors by pre-calculating the error corrections completely outside of the machine tool. This methodology forms the core of the research discussed in this thesis. It is typically executed in one of two ways:

- **Kinematic compensation (Toolpath / G-code post-processing):** The nominal G-code generated by the CAM software is intercepted by a dedicated algorithmic script (such as a MATLAB environment). By pushing the nominal toolpath coordinates through the Homogeneous Transformation Matrices (HTM) of the specific machine's error map, the algorithm predicts exactly where the tool will deviate. The script then recalculates the inverse kinematics to generate a new, spatially distorted set of X, Y, Z, A, and C coordinates. When this "pre-distorted" G-code is fed into the flawed physical machine, the machine's inherent geometric errors cancel out the G-code distortions, resulting in a highly accurate physical workpiece. [9, 40, 29]
- **Geometric compensation (CAD / STL morphing):** Instead of modifying the toolpath kinematics, this method alters the source geometry. The nominal 3D mesh (typically an STL file or an individual slice files such as SSL files for Fused Deposition Modeling) is mathematically analyzed against the machine's error space. The algorithm physically displaces the individual triangle vertices of the CAD model in the exact opposite direction of the predicted volumetric error. The CAM software then generates a standard toolpath over this morphed "anti-error" geometry. While mathematically elegant, this method struggles with complex 5-axis tool orientation vectors, making it more applicable to 3-axis milling or additive manufacturing. [31, 28]

I.8.4 The Hybrid architecture

In modern ultra-precision manufacturing, isolated methods are insufficient. The academic standard dictates a hybrid architecture. Because algorithmic software cannot mathematically compensate for stochastic (random) errors such as excessive mechanical vibration or loose guideways, a rigorous structural baseline calibration must first be executed. Only after the physical kinematics are verified as highly repeatable can offline mathematical compensation matrices be reliably deployed to digitally erase the remaining systematic volumetric deviations. [9, 40, 16]

Conclusion

This initial chapter has established the fundamental theoretical framework required to understand, quantify, and mitigate volumetric inaccuracies in multi-axis CNC machining. By transitioning from basic machine architectures to the rigorous definition of precision and accuracy, it was demonstrated that machining deviations are not arbitrary. While random errors require physical mitigation, systematic errors (which constitute the bulk of the 43 kinematic deviations in a 5-axis machine) can be mathematically isolated and predicted.

The analysis of rigid body kinematics revealed that adding rotational axes exponentially multiplies a machine's error chain. In particular, fixed-table configurations with bi-rotary spindle heads are highly susceptible to the Abbé principle, where even microscopic angular zero-offsets at the rotary joints amplify into significant positional deviations at the Tool Center Point (TCP). Addressing these specific volumetric flaws is critical for achieving true manufacturing accuracy.

A review of current metrology and mitigation strategies showed that while accurate data acquisition is possible through direct and indirect measurement, traditional compensation methods have strict limits. Structural calibration is constrained by manufacturing tolerances, and standard controller-embedded tables lack the processing power for complex 5-axis cross-talk. Furthermore, online dynamic systems present prohibitive hardware costs and integration challenges.

Therefore, this review identifies the hybrid architecture (combining a stable mechanical baseline with offline kinematic compensation) as the most effective methodology. By abstracting the machine's geometric flaws into a virtual mathematical space using Homogeneous Transformation Matrices (HTM), the toolpath can be computationally pre-distorted before reaching the physical hardware. Building upon this theoretical foundation, the subsequent chapter will focus on the mechanical architecture and kinematic modeling of the targeted bi-rotary prototype, setting the stage for the algorithmic simulations required to execute this offline compensation.

CHAPTER II

KINEMATIC MODELING AND MATHEMATICAL FOUNDATIONS

Introduction

The previous chapter established that while multi-axis CNC machines are inherently susceptible to geometric and volumetric deviations, these systematic errors (unlike random dynamic vibrations) can be mathematically quantified and predicted. To transition from the theoretical classification of these errors into a practical offline compensation strategy, the physical machine tool must first be abstracted into a rigorous digital framework.

This chapter details the mathematical formulation of a complete volumetric kinematic model, specifically tailored for a 5-axis CNC architecture utilizing a bi-rotary (Head-Head) configuration. By establishing the fundamental mathematics of rigid body motion, Homogeneous Transformation Matrices (HTM) are formulated to represent both the ideal machine kinematics and the actual flawed machine state. The synthesis of these matrices allows for the exact spatial deviation at the Tool Center Point (TCP) to be algebraically calculated.

Furthermore, this chapter outlines the mathematical mechanics of two primary offline compensation strategies: kinematic toolpath shifting (G-code modification) and 3D workpiece model adaptation (STL morphing). By rigorously comparing the computational demands and manufacturing implications of both strategies, this chapter will ultimately establish and justify the most viable algorithmic approach for minimizing volumetric inaccuracies in the final prototype.

II.1 Mathematical representation of rigid body motion

A 5-axis CNC machine is fundamentally a serial kinematic chain composed of multiple rigid structural bodies (such as the machine bed, column, spindle, and rotary joints) connected by prismatic (linear) and revolute (rotary) joints. The topological layout of this specific kinematic chain is illustrated in Figure II.1 (TTTRR type). To predict how microscopic geometric errors at the base of this chain amplify at the tool tip, the physical movement of these rigid bodies must be translated into a universal mathematical language.

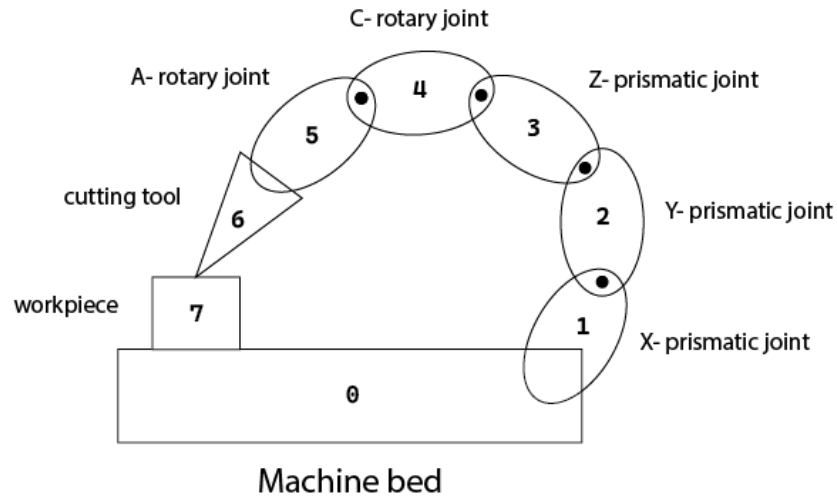


Figure II.1: Kinematic chain diagram of 5-axis TTTRR CNC machine

In precision engineering, this is achieved by assigning a local Cartesian coordinate system to each moving body. The standard orientation of these Cartesian axes relative to the bi-rotary machine structure is demonstrated in Figure II.2. The volumetric error model is then constructed by calculating the spatial transformations required to map a point from one local system to the next, ultimately linking the tool tip to the global reference frame of the workpiece.

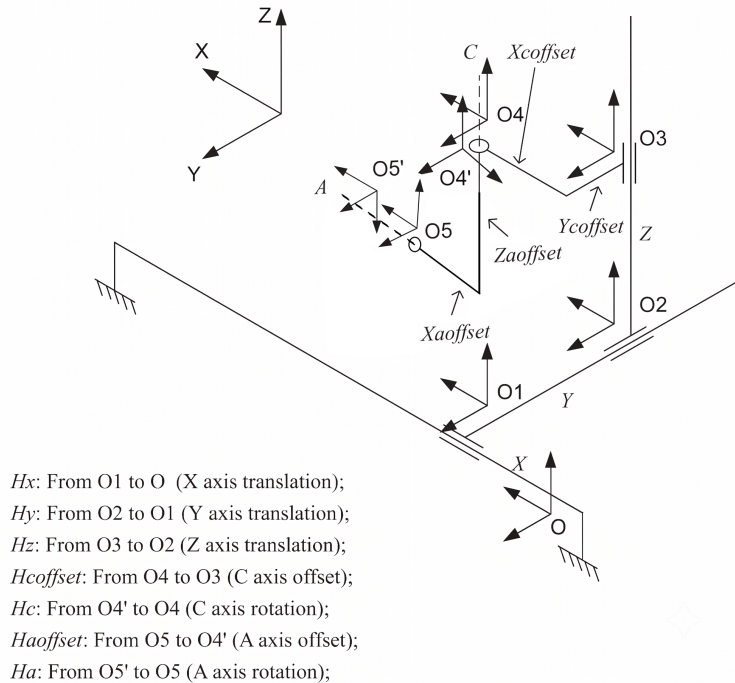


Figure II.2: Coordinate systems for the kinematic chain of the machine

II.1.1 Denavit-Hartenberg (D-H) vs. Homogeneous Transformation Matrices (HTM)

When mathematically modeling serial kinematic chains, two primary conventions are generally considered: the Denavit-Hartenberg (D-H) parameterization and Homogeneous Transformation Matrices (HTM). While both methods calculate spatial positioning, their suitability diverges drastically when applied to precision machine tools rather than standard articulated industrial robots.

The D-H convention is the global standard in robotic arm kinematics. It simplifies calculations by requiring only four parameters (joint angle θ , link offset d , link length a , and link twist α) to describe the relationship between two adjacent coordinate frames. However, the D-H method suffers from a critical mathematical flaw known as the parallel axis singularity. [21]

In a machine tool, adjacent sliding axes (such as the X and Y guideways) are nominally orthogonal or parallel. If a nominally parallel axis possesses a microscopic geometric angular error (such as a 0.001° straightness or yaw deviation), the mathematical D-H link length parameter approaches infinity [21]. This singularity makes D-H parameterization inherently unstable and wholly unsuitable for modeling the micro-scale volumetric errors of CNC machine tools. [21]

To overcome this singularity, this thesis strictly utilizes the Homogeneous Transformation Matrix (HTM) approach based on rigid body kinematics. Unlike D-H, the HTM method utilizes six independent degrees of freedom (three translations and three rotations). This allows for the seamless integration of small geometric deviations without triggering mathematical singularities, making it the universally accepted standard for 5-axis volumetric error modeling. [21]

II.1.2 The structure of the Homogeneous Transformation Matrix

An HTM is a 4×4 mathematical array that simultaneously describes both the spatial translation and the angular orientation of a local coordinate frame relative to a reference frame. A general transformation matrix T is constructed by combining a 3×3 rotation matrix (R) and a 3×1 position vector (P), bound by a 1×3 perspective transformation row and a global scaling factor (which is always 1 for rigid machine tools): [13]

$$T = \begin{bmatrix} R_{3 \times 3} & P_{3 \times 1} \\ 0_{1 \times 3} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & P_x \\ r_{21} & r_{22} & r_{23} & P_y \\ r_{31} & r_{32} & r_{33} & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{II.1})$$

When a machine axis moves, a physical point p_{local} in the moving coordinate frame can be mapped back to the stationary reference frame to find its new global position P_{global} using direct matrix multiplication: [13]

$$P_{global} = T \times p_{local} \quad (\text{II.2})$$

To construct the complete kinematic model of the bi-rotary CNC machine, fundamental 4×4 matrices must be defined for pure translations and pure rotations. For a prismatic linear axis commanded to move by a distance of x , y , or z , the pure translation matrix (T_{trans}) contains no angular changes, meaning the rotation block remains an identity matrix: [13]

$$T_x(x) = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_y(y) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_z(z) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{II.3})$$

For a revolute axis (such as the A or C axes on the spindle head), the translation vector remains zero, and the rotation block dictates the angular change. The fundamental rotation matrices around the X, Y, and Z axes by an angle $\theta_A, \theta_B, \theta_C$ are mathematically defined as: [13]

$$R_x(\theta_A) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta_A & -\sin \theta_A & 0 \\ 0 & \sin \theta_A & \cos \theta_A & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{II.4})$$

$$R_y(\theta_B) = \begin{bmatrix} \cos \theta_B & 0 & \sin \theta_B & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta_B & 0 & \cos \theta_B & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{II.5})$$

$$R_z(\theta_C) = \begin{bmatrix} \cos \theta_C & -\sin \theta_C & 0 & 0 \\ \sin \theta_C & \cos \theta_C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{II.6})$$

By sequentially multiplying these fundamental building blocks, the complete kinematic chain of the customized 5-axis prototype can be mathematically linked from the workpiece resting on the table all the way to the active cutting zone. [13]

II.2 Topological modeling of the bi-rotary configuration

Before calculating the specific transformation matrices, the precise kinematic topology of the physical machine tool must be mathematically defined.

The prototype developed and analyzed in this research utilizes a strict Head-Head bi-rotary configuration (rotary tilting cutting tool). In this architecture, the workpiece is mounted to a completely stationary table. All five degrees of freedom (the three orthogonal linear translations and the two rotary orientations) are executed entirely by the moving spindle assembly. [32]

Applying rigid body kinematics, the structural loop is traced from the stationary machine bed (the global reference frame) through the sequential linear axes, into the bi-rotary head, and terminating at the Tool Center Point (TCP). Therefore, the specific topological kinematic chain for this machine is expressed as:

$$Base \rightarrow X_{carriage} \rightarrow Y_{gantry} \rightarrow Z_{ram} \rightarrow C_{rotation} \rightarrow A_{rotation} \rightarrow Tool \quad (II.7)$$

II.2.1 Definition of structural offsets

To accurately model this serial chain using Homogeneous Transformation Matrices, the physical distances (offsets) between the rotary pivots and the tool tip must be defined as static translation constants within the mathematical framework. Because the bi-rotary head possesses a physical volume, the rotary axes do not intersect at a single microscopic point. Instead, they are separated by rigid structural links. [15, 32]

Based on the machine's physical geometry, two critical structural offsets dictate the kinematic behavior of the spindle head:

- D_{CA} : The fixed vertical Z-distance from the primary C-axis pivot down to the secondary A-axis pivot.
- L_{total} : The combined reach from the A-axis pivot down to the actual Tool Center Point (TCP).

The total tool reach (L_{total}) is a dynamic variable that changes depending on the specific tooling loaded into the machine. It is mathematically formulated as the sum of the constant spindle length ($L_{spindle}$, the distance from the A-axis pivot to the chuck face) and the variable tool length (L_{tool}):

$$L_{total} = L_{spindle} + L_{tool} \quad (II.8)$$

In the kinematic model, these structural offsets are represented as pure translation matrices along the Z-axis. Because the structural progression moves downward from the Z-ram toward the workpiece, these translations occur in the negative Z direction:

$$T_{offset}(D_{CA}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -D_{CA} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad T_{offset}(L_{total}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -L_{total} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (II.9)$$

By establishing these specific structural offsets, the generalized HTM theory can be directly mapped to the physical dimensions of the bi-rotary prototype, enabling the formulation of the complete volumetric error model. [32]

II.3 Comprehensive Volumetric Kinematic Modeling

With the topological chain established and the structural offsets defined, the comprehensive mathematical model of the machine tool can be constructed. This process is divided into two phases: establishing the ideal kinematic model (Forward Kinematics assuming perfect geometry) and formulating the actual kinematic model (incorporating the 43 geometric and dynamic flaws).

II.3.1 The Ideal Kinematic Model (Forward Kinematics)

The ideal Forward Kinematics (FK) model calculates the theoretical position of the Tool Center Point (TCP) assuming the machine is manufactured and assembled with absolute perfection. Let x, y, z represent the commanded linear translations, and c, a represent the commanded rotary angles in radians.

By sequentially multiplying the fundamental homogeneous transformation matrices from the base to the tool tip, adhering to the bi-rotary topological chain defined previously, the ideal transformation matrix (T_{ideal}) is expressed by the following equation. [15]

$$T_{ideal} = T_X(x) \cdot T_Y(y) \cdot T_Z(z) \cdot R_Z(c) \cdot T_{offset}(D_{CA}) \cdot R_X(a) \cdot T_{offset}(L_{total}) \quad (II.10)$$

This resulting 4×4 transformation matrix describes the entire spatial frame of the tool. However, to calculate volumetric error, only the specific 3D Cartesian coordinates of the tool tip are required. By multiplying T_{ideal} by a unit origin vector, the exact X, Y , and Z coordinates of the ideal TCP (P_{ideal}) are extracted for later comparison. [22]

$$P_{ideal} = \begin{bmatrix} X_{ideal} \\ Y_{ideal} \\ Z_{ideal} \\ 1 \end{bmatrix} = T_{ideal} \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (II.11)$$

II.3.2 The Actual Kinematic Model (Error Integration)

In reality, the physical prototype suffers from systematic geometric errors. According to exact rigid body HTM modeling principles, these microscopic deviations are integrated directly into the local transformation matrices of each moving body. To model the full 43-error profile of the 5-axis machine, errors are classified into 21 linear errors and 22 rotary errors.

II.3.2.1 Exact model

For a linear axis, the actual transformation matrix incorporates six degrees of freedom of dynamic component error. For example, the actual flawed matrix for the X-axis (T_{X_actual}) is formulated as follows.

$$T_{X_actual} = T_X(x + \delta_{xx}) \cdot T_Y(\delta_{yx}) \cdot T_Z(\delta_{zx}) \cdot R_X(\varepsilon_{xx}) \cdot R_Y(\varepsilon_{yx}) \cdot R_Z(\varepsilon_{zx}) \quad (\text{II.12})$$

To ensure mathematical clarity, the geometric error parameters injected into this matrix are defined as:

- δ_{xx} : The linear positioning error along the driven X-axis.
- δ_{yx}, δ_{zx} : The parasitic straightness deviations occurring in the Y and Z directions during X-axis travel.
- ε_{xx} : The angular roll error around the X-axis.
- $\varepsilon_{yx}, \varepsilon_{zx}$: The angular pitch and yaw errors around the Y and Z axes.

This direct-injection logic is symmetrically applied to the Y and Z axes to generate T_{Y_actual} and T_{Z_actual} . Furthermore, the 3 static squareness (orientation) errors between the orthogonal linear axes (S_{xy}, S_{yz}, S_{zx}) are modeled as microscopic rotational matrices inserted between the corresponding linear blocks. For example, a squareness error between the X and Y axes (S_{xy}) means the Y-axis is physically misaligned within the X-Y plane. Because this misalignment is an angular rotation strictly within the X-Y plane, it is mathematically modeled as a pure rotation around the Z-axis, expressed as $R_Z(S_{xy})$. [39]

For the rotary axes (C and A), the mathematical integration must account for 11 specific errors per axis: 5 static location errors (setup misalignments) and 6 dynamic component errors (flaws that change as the axis rotates). The actual transformation matrix for the rotary C-axis (T_{C_actual}) is derived as the product of its static location matrix and its dynamic component matrix.

$$\begin{aligned} T_{C_actual} = & [T_X(\delta x_C) T_Y(\delta y_C) R_X(\alpha_C) R_Y(\beta_C)] \\ & \cdot [T_X(\delta x_{C(rad)}) T_Y(\delta y_{C(rad)}) T_Z(\delta z_{C(ax)}) R_X(\varepsilon_{xC(wob)}) R_Y(\varepsilon_{yC(wob)})] \\ & \cdot R_Z(c + \gamma_C + \varepsilon_{zC(pos)}) \end{aligned} \quad (\text{II.13})$$

It is important to note that the parameter c in the final rotation block represents the ideal commanded G-code angle for the C-axis, to which the respective rotational errors are added. To strictly align with established terminology, the 11 constituent errors for the C-axis are defined as:

- $\delta x_C, \delta y_C$: The static, constant linear location offsets of the pivot point.

- α_C, β_C : The static squareness or orientation errors of the rotary axis relative to the primary planes.
- γ_C : The static, constant zero-position angular offset.
- $\delta x_{C(rad)}, \delta y_{C(rad)}$: The dynamic radial error motions (straightness drifts) occurring during rotation.
- $\delta z_{C(ax)}$: The dynamic axial error motion (thrust drift).
- $\varepsilon_{xC(wob)}, \varepsilon_{yC(wob)}$: The dynamic tilt error motions (wobble).
- $\varepsilon_{zC(pos)}$: The dynamic angular positioning error relative to the commanded angle c .

This identical 11-error architecture is symmetrically applied to the A-axis (rotating around X) to generate T_{A_actual} . By compiling these rigorously defined component matrices, the comprehensive actual kinematic model (T_{actual}) for the flawed bi-rotary machine is constructed, accurately mirroring the exact structural chain while embedding all 43 geometric deviations. [39]

$$T_{actual} = T_{X_actual} \cdot R_Z(S_{xy}) \cdot T_{Y_actual} \cdot R_X(S_{yz}) \cdot T_{Z_actual} \cdot R_Y(S_{zx}) \cdot T_{C_actual} \cdot T_{offset}(D_{CA}) \cdot T_{A_actual} \cdot T_{offset}(L_{total}) \quad (II.14)$$

Before extracting the final volumetric error vector from these comprehensive matrices, it is critical to validate this exact direct-injection matrix architecture against classical analytical methods found in early literature. [35]

II.3.2.2 Differential approximations

The comprehensive actual kinematic model formulated in the previous section represents an *exact* finite model. It utilizes strict rigid body mathematics and pure trigonometric functions to calculate spatial deviations. However, in the established literature regarding volumetric error compensation, an older mathematical approach is frequently encountered: the differential (or first-order approximated) error matrix approach. Early foundational literature often utilized this simplified differential error matrix (typically denoted as E or ΔT) instead of calculating the exact matrices. [22, 15]

Before extracting the final volumetric error mapping, it is helpful to observe how the exact matrix method used in this thesis relates to the classical differential models commonly found in literature. This comparison shows that the exact method is fundamentally equivalent to the differential matrix, differing primarily in the application of small-angle approximations.

- **Derivation of the exact kinematic model:** Consider an individual linear axis (e.g., the X-axis) subjected to six degrees of freedom of geometric error. In the

exact method utilized in this model, the actual transformation matrix (T_{X_actual}) is constructed by sequentially multiplying the exact translational matrix (T_{trans}), which incorporates positioning (δ_x) and straightness (δ_y, δ_z) errors, by the exact rotational matrices for roll (ε_x), pitch (ε_y), and yaw (ε_z).

$$T_{X_actual} = T_{trans}(x + \delta_x, \delta_y, \delta_z) \cdot R_X(\varepsilon_x) \cdot R_Y(\varepsilon_y) \cdot R_Z(\varepsilon_z) \quad (\text{II.15})$$

Where the exact rotational matrices utilize strict trigonometric functions. For example, the yaw error matrix is defined as follows.

$$R_Z(\varepsilon_z) = \begin{bmatrix} \cos(\varepsilon_z) & -\sin(\varepsilon_z) & 0 & 0 \\ \sin(\varepsilon_z) & \cos(\varepsilon_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{II.16})$$

- **Application of the small-angle approximation:** Because the geometric angular errors inherent in precision machine tools are microscopic (typically on the order of 10^{-4} to 10^{-5} radians), older analytical models apply the Taylor series first-order small-angle approximation to simplify hand-written matrix algebra. This assumption dictates the following linearizations. [22]

$$\cos(\varepsilon) \approx 1, \quad \sin(\varepsilon) \approx \varepsilon \quad (\text{II.17})$$

Applying this linearization, the three rotational matrices transform into the following simplified structures.

$$\begin{aligned} R_X &\approx \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\varepsilon_x & 0 \\ 0 & \varepsilon_x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\ R_Y &\approx \begin{bmatrix} 1 & 0 & \varepsilon_y & 0 \\ 0 & 1 & 0 & 0 \\ -\varepsilon_y & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\ R_Z &\approx \begin{bmatrix} 1 & -\varepsilon_z & 0 & 0 \\ \varepsilon_z & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (\text{II.18})$$

When sequentially multiplying these three approximated matrices together ($R_{combined} = R_X \cdot R_Y \cdot R_Z$), any higher-order terms generated by the multiplication of two infinitesimally small errors are considered negligible and discarded. This vast simplification reduces the combined rotational block into the following matrix. [22]

$$R_{combined} \approx \begin{bmatrix} 1 & -\varepsilon_z & \varepsilon_y & 0 \\ \varepsilon_z & 1 & -\varepsilon_x & 0 \\ -\varepsilon_y & \varepsilon_x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{II.19})$$

Multiplying this simplified rotation matrix by the translational matrix yields the first-order approximated actual position matrix. [39]

$$T_{X_actual} \approx \begin{bmatrix} 1 & 0 & 0 & x + \delta_x \\ 0 & 1 & 0 & \delta_y \\ 0 & 0 & 1 & \delta_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -\varepsilon_z & \varepsilon_y & 0 \\ \varepsilon_z & 1 & -\varepsilon_x & 0 \\ -\varepsilon_y & \varepsilon_x & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -\varepsilon_z & \varepsilon_y & x + \delta_x \\ \varepsilon_z & 1 & -\varepsilon_x & \delta_y \\ -\varepsilon_y & \varepsilon_x & 1 & \delta_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{II.20})$$

- **Extraction of the Differential Error Matrix (E):** The differential kinematic approach dictates that the actual transformation matrix can be expressed as the ideal, error-free matrix (T_{ideal}) multiplied by a differential error matrix (E). For an isolated X-axis movement of distance x , T_{ideal} is defined strictly as follows. [22]

$$T_{ideal} = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{II.21})$$

To mathematically satisfy the equation $T_{X_actual} = T_{ideal} \cdot E_X$ using the result from Equation II.20, the nominal dimension x is factored out of the rightmost column, isolating the error parameters into the classical differential error matrix E_X . [22]

$$E_X = \begin{bmatrix} 1 & -\varepsilon_z & \varepsilon_y & \delta_x \\ \varepsilon_z & 1 & -\varepsilon_x & \delta_y \\ -\varepsilon_y & \varepsilon_x & 1 & \delta_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{II.22})$$

This derivation illustrates that the classical E matrix found in literature acts as a linearized, first-order mathematical shortcut of the exact HTM multiplication used in this thesis. While the differential approximation was historically necessary to make analytical algebraic manipulation tractable on paper, modern computational environments allow for the direct use of the exact trigonometric model. By employing the exact calculation method, the proposed model avoids the microscopic truncations of the small-angle approximation, providing a robust foundation for mapping the final volumetric errors at the tool tip. [35]

II.3.3 Volumetric Error Mapping at the TCP

With both the theoretical and flawed spatial matrices rigorously defined, the global volumetric error vector at any given coordinate along the toolpath can be extracted. By isolating the final position columns of both transformation matrices, the true spatial deviation (ΔP) at the Tool Center Point is calculated by subtracting the ideal coordinates from the actual physical coordinates.

$$\Delta P = \begin{bmatrix} \Delta X \\ \Delta Y \\ \Delta Z \end{bmatrix} = \begin{bmatrix} X_{actual} - X_{ideal} \\ Y_{actual} - Y_{ideal} \\ Z_{actual} - Z_{ideal} \end{bmatrix} \quad (II.23)$$

This 3×1 spatial displacement vector ΔP represents the precise, deterministic geometrical deviation of the machine at a specific posture. It serves as the mathematical foundation for the offline feedforward compensation algorithm developed in the subsequent sections of this chapter. [39]

II.4 Inverse Kinematics for Bi-Rotary Heads

While the forward kinematic model predicts the final spatial position of the Tool Center Point (TCP) based on given axis commands, the CNC controller and offline compensation algorithms require the reverse mathematical process. Inverse Kinematics (IK) calculates the required machine joint commands (linear positions (x, y, z) and rotary angles (a, c)) to achieve a desired tool position and orientation relative to the workpiece.

In a 5-axis CAM trajectory, a target toolpath point is typically defined by its spatial coordinates $P_w = [X_w, Y_w, Z_w]^T$ and a normalized tool orientation vector $O_w = [i, j, k]^T$ within the workpiece coordinate system (WCS). [39]

II.4.1 The Quasi-Static Error Assumption

To facilitate the hybrid solver architecture, the mathematical nature of the dynamic geometric errors must be bounded. In physical reality, dynamic component errors (such as rotary wobble or radial drift) are continuous functions dependent on the instantaneous angle of the axis.

However, for the purpose of this offline compensation algorithm, the 43 geometric errors are evaluated under a quasi-static assumption. The error parameters are extracted as maximum scalar bounding values representing the worst-case structural misalignments for a given posture. By treating these dynamic deviations as static bounds during the computational cycle, the comprehensive rotational error matrices (R_{C_err} and R_{A_err}) become mathematically independent of the continuously updating command angles. This fundamental assumption prevents analytical paradoxes and enables the strict mathematical decoupling of the orientation and position domains.

II.4.2 Stage 1: Closed-Form Orientation Decoupling

The bi-rotary Head-Head configuration dictates that tool orientation is exclusively governed by the C and A rotary sub-chains. Because the quasi-static rotational error matrices are treated as constants, Stage 1 isolates and analytically solves the compensated rotary angles without requiring numerical iteration. [37]

The objective is to find the compensated angles (a_{comp}, c_{comp}) that force the flawed machine's rotary chain to match the ideal tool direction vector, d_{ideal} . Assuming the cutting tool is aligned along the negative Z-axis $[0, 0, -1]^T$, the ideal vector commanded by the CAM software is defined as:

$$d_{ideal} = R_Z(c_{cmd}) \cdot R_X(a_{cmd}) \cdot \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \quad (\text{II.24})$$

The orientation generated by the actual, flawed physical machine is represented by the product of the comprehensive actual matrices:

$$d_{flawed} = R_{C_err} \cdot R_Z(c_{comp}) \cdot R_{A_err} \cdot R_X(a_{comp}) \cdot \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \quad (\text{II.25})$$

By setting $d_{flawed} = d_{ideal}$, the equation is pre-multiplied by the transpose of the C-axis error matrix ($R_{C_err}^T$) to shift the structural C-axis flaws to the right side, creating a known intermediate target vector d_{int} and mathematically isolating the A-axis tilt:

$$R_Z(c_{comp}) \cdot R_{A_err} \cdot R_X(a_{comp}) \cdot \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} = R_{C_err}^T \cdot d_{ideal} = d_{int} \quad (\text{II.26})$$

Let the intermediate physical tool vector (representing the tool's spatial state after the flawed A-axis tilt but prior to the C-axis rotation) be defined as v :

$$v = R_{A_err} \cdot R_X(a_{comp}) \cdot \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} = \begin{bmatrix} R_{A_err}(1,1) & R_{A_err}(1,2) & R_{A_err}(1,3) \\ R_{A_err}(2,1) & R_{A_err}(2,2) & R_{A_err}(2,3) \\ R_{A_err}(3,1) & R_{A_err}(3,2) & R_{A_err}(3,3) \end{bmatrix} \cdot \begin{bmatrix} 0 \\ \sin(a_{comp}) \\ -\cos(a_{comp}) \end{bmatrix} \quad (\text{II.27})$$

Substituting v back into the primary equation yields $R_Z(c_{comp}) \cdot v = d_{int}$. Expanding this matrix multiplication exposes the unique geometric property of the bi-rotary Head-Head configuration:

$$\begin{bmatrix} \cos(c_{comp}) & -\sin(c_{comp}) & 0 \\ \sin(c_{comp}) & \cos(c_{comp}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} d_{int}(1) \\ d_{int}(2) \\ d_{int}(3) \end{bmatrix} \quad (\text{II.28})$$

Because the C-axis spins strictly around the Z-axis, its fundamental transformation matrix acts exclusively in the 2D X-Y plane. Evaluating the third row of this matrix multiplication yields a direct geometric equivalence that is completely independent of the unknown angle c_{comp} :

$$0 \cdot v_x + 0 \cdot v_y + 1 \cdot v_z = d_{int}(3) \implies v_z = d_{int}(3) \quad (\text{II.29})$$

With the Z-component analytically isolated, the definition of v is expanded to extract the A-axis command.

$$v_z = R_{A_err}(3, 1) \cdot (0) + R_{A_err}(3, 2) \cdot \sin(a_{comp}) + R_{A_err}(3, 3) \cdot (-\cos(a_{comp})) \quad (\text{II.30})$$

Simplifying this extraction eliminates the first term entirely, leaving:

$$v_z = R_{A_err}(3, 2) \sin(a_{comp}) - R_{A_err}(3, 3) \cos(a_{comp}) \quad (\text{II.31})$$

By equating this expanded physical Z-component back to the known intermediate target value, the variables are successfully extracted from the matrix algebra, yielding the fundamental decoupled equation for the A-axis:

$$R_{A_err}(3, 2) \sin(a_{comp}) - R_{A_err}(3, 3) \cos(a_{comp}) = d_{int}(3) \quad (\text{II.32})$$

To solve the derived trigonometric relationship for the compensated A-axis angle (a_{comp}), the equation is first simplified by condensing the fixed spatial error elements of the A-axis into constant coefficients. Let P_A and Q_A be defined as:

$$P_A = R_{A_err}(3, 2), \quad Q_A = -R_{A_err}(3, 3) \quad (\text{II.33})$$

Substituting these constants yields a classic linear combination of sine and cosine:

$$P_A \sin(a_{comp}) + Q_A \cos(a_{comp}) = d_{int}(3) \quad (\text{II.34})$$

Because the A-axis tilt is physically continuous, this equation can be analytically inverted utilizing the Harmonic Addition Theorem (also known as the auxiliary-angle method). The theorem states that a linear combination of sine and cosine can be expressed as a single phase-shifted sine wave. To apply this, the amplitude R and the auxiliary phase angle ϕ are calculated as:

$$R = \sqrt{P_A^2 + Q_A^2}, \quad \phi = \text{atan2}(Q_A, P_A) \quad (\text{II.35})$$

By factoring the amplitude R out of the equation, the expression is rewritten in the harmonic form:

$$R \left(\frac{P_A}{R} \sin(a_{comp}) + \frac{Q_A}{R} \cos(a_{comp}) \right) = d_{int}(3) \quad (\text{II.36})$$

Given that $\cos(\phi) = \frac{P_A}{R}$ and $\sin(\phi) = \frac{Q_A}{R}$, trigonometric identities allow the equation to collapse completely into:

$$R \sin(a_{comp} + \phi) = d_{int}(3) \quad (\text{II.37})$$

Isolating the sine term reveals the fundamental geometric constraint that must be satisfied to correctly orient the tool tip:

$$\sin(a_{comp} + \phi) = \frac{d_{int}(3)}{R} \quad (\text{II.38})$$

Let an intermediate angle γ be defined as $\gamma = \text{asin}\left(\frac{d_{int}(3)}{R}\right)$. Due to the periodic nature of the sine function, taking the inverse yields two mathematically valid branches for the A-axis orientation:

$$a_{comp,1} = \gamma - \phi \quad (\text{II.39})$$

$$a_{comp,2} = \pi - \gamma - \phi \quad (\text{II.40})$$

These two roots represent the two physical postures the bi-rotary head can assume to satisfy the target vector. To maintain machine continuity and prevent catastrophic hardware collisions, the algorithmic conditional gate evaluates both roots and automatically selects the solution that minimizes the angular travel from the previous machine state. [35, 14]

To determine the required C-axis compensation (c_{comp}), this physical vector v must be rotated around the Z-axis until it aligns precisely with the intermediate target vector d_{int} . Returning to the expanded matrix multiplication of $R_Z(c_{comp}) \cdot v = d_{int}$, the planar X and Y components are isolated:

$$\begin{bmatrix} \cos(c_{comp}) & -\sin(c_{comp}) \\ \sin(c_{comp}) & \cos(c_{comp}) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} d_{int}(1) \\ d_{int}(2) \end{bmatrix} \quad (\text{II.41})$$

This matrix relationship yields a system of two coupled linear equations:

$$v_x \cos(c_{comp}) - v_y \sin(c_{comp}) = d_{int}(1) \quad (\text{II.42})$$

$$v_x \sin(c_{comp}) + v_y \cos(c_{comp}) = d_{int}(2) \quad (\text{II.43})$$

To solve for c_{comp} without introducing quadrant ambiguities or dividing by zero, the system is solved simultaneously for both $\sin(c_{comp})$ and $\cos(c_{comp})$. By multiplying Equation 3 by v_x and Equation 4 by v_y , and subsequently adding the two equations, the sine terms cancel out, isolating the cosine component:

$$(v_x^2 + v_y^2) \cos(c_{comp}) = v_x d_{int}(1) + v_y d_{int}(2) \quad (\text{II.44})$$

Similarly, by cross-multiplying and subtracting the equations, the cosine terms are eliminated to isolate the sine component:

$$(v_x^2 + v_y^2) \sin(c_{comp}) = v_x d_{int}(2) - v_y d_{int}(1) \quad (\text{II.45})$$

Because the coefficient $(v_x^2 + v_y^2)$ is strictly positive and represents the squared magnitude of the vector's projection onto the X-Y plane, it can be factored out. The compensated C-axis angle is then securely extracted utilizing the four-quadrant inverse tangent function (atan2):

$$c_{comp} = \text{atan2}(v_x d_{int}(2) - v_y d_{int}(1), v_x d_{int}(1) + v_y d_{int}(2)) \quad (\text{II.46})$$

Kinematic Singularity Management While the inverse tangent formulation mathematically resolves standard quadrant ambiguities, a fundamental kinematic singularity exists within bi-rotary Head-Head topologies. When the ideal tool vector is perfectly vertical (i.e., aligned with the Z-axis), the planar projection magnitude $(v_x^2 + v_y^2)$ converges to zero. In this state, the C-axis rotation becomes mathematically and physically indeterminate, as any arbitrary angular position will satisfy a purely vertical vector.

To resolve this indeterminacy and preserve kinematic continuity, a strict mathematical boundary condition is established prior to the trigonometric evaluation. If the planar magnitude falls below a microscopic threshold (e.g., 10^{-8}), the geometric extraction is considered singular. Under this condition, the compensated angle is mathematically constrained to equal the nominal kinematic angle:

$$\text{if } \sqrt{v_x^2 + v_y^2} < 10^{-8} \implies c_{comp} = c_{nominal} \quad (\text{II.47})$$

This boundary condition guarantees a continuous, uniquely defined solution space across all possible tool orientations. With both a_{comp} and c_{comp} securely resolved, the closed-form orientation formulation is complete, establishing a stable mathematical foundation for the subsequent spatial translational adjustments.

II.4.3 Stage 2: Fixed-Point Feedforward Iteration

While Stage 1 analytically perfects the tool orientation, tilting the flawed bi-rotary head introduces severe parasitic translational shifts at the Tool Center Point due to structural offsets and linear positioning errors. Stage 2 eliminates this residual spatial deviation using a highly efficient, Jacobian-free fixed-point numerical iteration.

To initialize the solver ($k = 0$), the baseline linear estimates are set to the nominal Post-CAM target coordinates ($x_0 = x_{ideal}$, $y_0 = y_{ideal}$, $z_0 = z_{ideal}$). With the compensated

rotary angles locked, the algorithm feeds the current estimates of the linear commands into the comprehensive 43-error Forward Kinematics (FK) model to evaluate the actual physical TCP coordinates (p_{actual}). The 3D spatial residual deviation (Δp) at iteration k is calculated against the ideal target coordinates (p_{ideal}):

$$\Delta p_k = p_{actual,k} - p_{ideal} \quad (\text{II.48})$$

Because the orthogonal linear axes in a Cartesian machine directly govern spatial translation, the residual error can be subtracted from the linear commands for the subsequent numerical iteration ($k+1$). However, to prevent mathematical oscillation caused by the severe cross-coupling of the angular lever-arms, a relaxation factor (α) is applied to dampen the feedforward step:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{k+1} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_k - \alpha \cdot \Delta p_k \quad (\text{II.49})$$

This feedforward loop continuously simulates the machine's geometric response, recalculating the forward kinematics and updating the linear coordinates. The numerical solver terminates when the magnitude of the 3D spatial residual falls below a strict microscopic tolerance ($\|\Delta p_k\| < \epsilon$), yielding the final 5-axis compensated G-code. By fusing analytical rotary decoupling with a dampened fixed-point linear iteration, this hybrid solver ensures maximum volumetric precision while bypassing the intense computational overhead of traditional global optimization strategies.

II.5 Comparative Analysis: Strategy Selection

Before executing the hybrid inverse kinematics solver developed in the previous sections, the stage at which the compensation is applied within the digital manufacturing pipeline must be carefully selected. To maximize computational efficiency and ensure the highest possible surface quality on the final workpiece, two primary offline strategies were considered for this research: Pre-CAM geometric distortion (modifying the STL mesh) and Post-CAM trajectory modification (modifying the G-code), as shown in Figure II.3. [28]

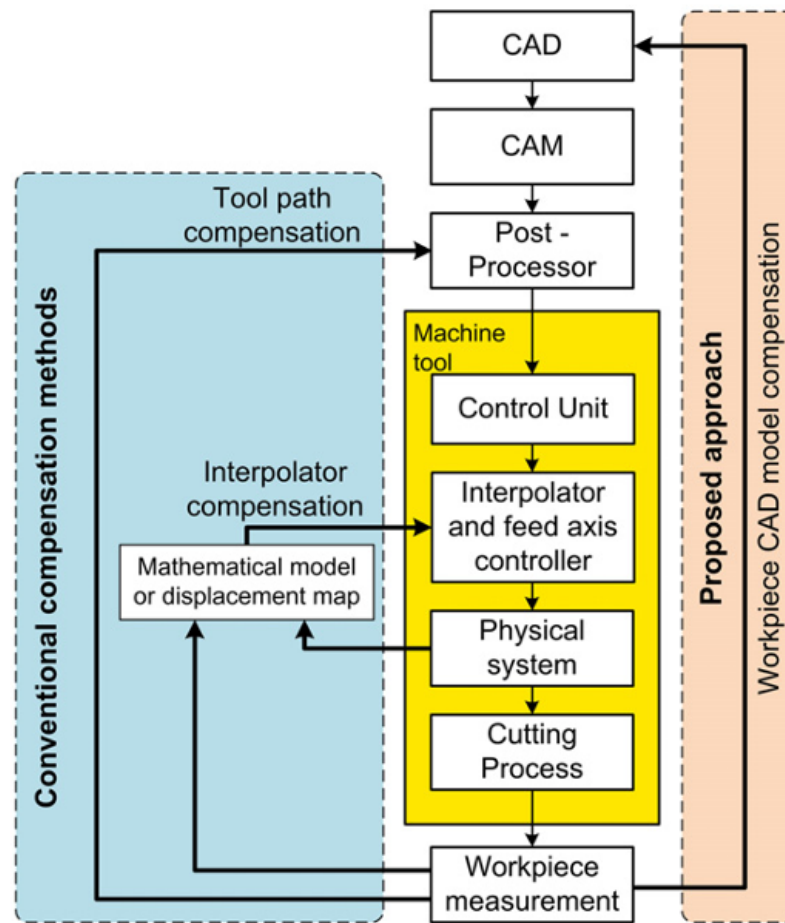


Figure II.3: Concepts of error compensation proposed approaches

II.5.1 Pre-CAM Compensation: The STL Distortion Method

The Pre-CAM strategy involves applying the kinematic error model directly to the original 3D CAD model before any toolpaths are generated. In this approach, as shown in Figure II.4, the standard mesh (typically an STL file) is mathematically inverted and physically distorted in the exact opposite direction of the machine's mapped volumetric errors. This warped model is then imported into the CAM software to generate all roughing and finishing passes. [28]

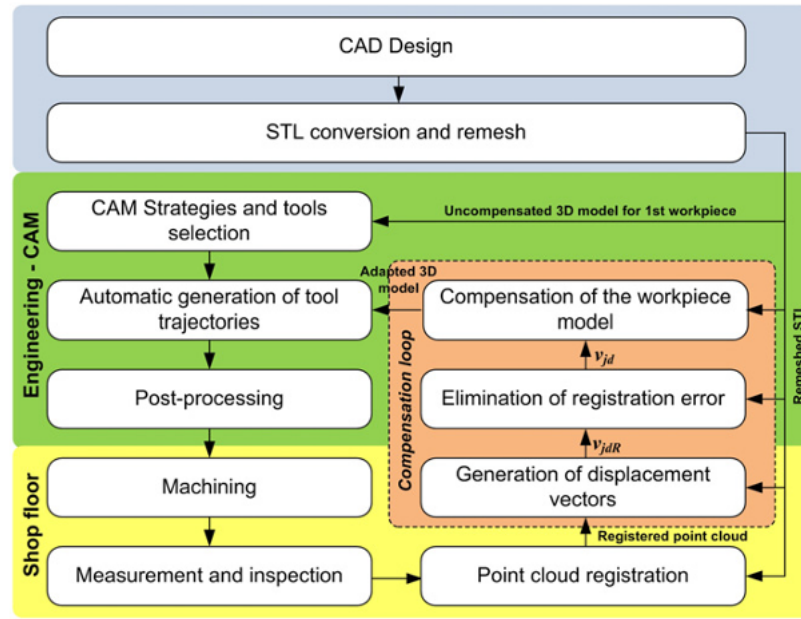


Figure II.4: Main phases of the STL approach

Advantages:

- **Comprehensive CAM Integration:** Because the CAM software operates on the distorted model, it natively handles complex gouge-checking, tool-vector interpolation, and collision detection across all operations.
- **Uniform Stock Allowance:** Applying compensation to the roughing stages is highly beneficial. Uncompensated roughing passes, driven by a flawed machine, leave an uneven distribution of unmachined stock on the workpiece. During the finishing pass, this uneven material causes variable tool engagement and spikes in cutting forces, leading to tool deflection that degrades the final surface quality. By compensating the entire STL mesh, the roughing passes leave a perfectly uniform layer of material for the finishing tool to remove, as shown in Figure II.5. [28]

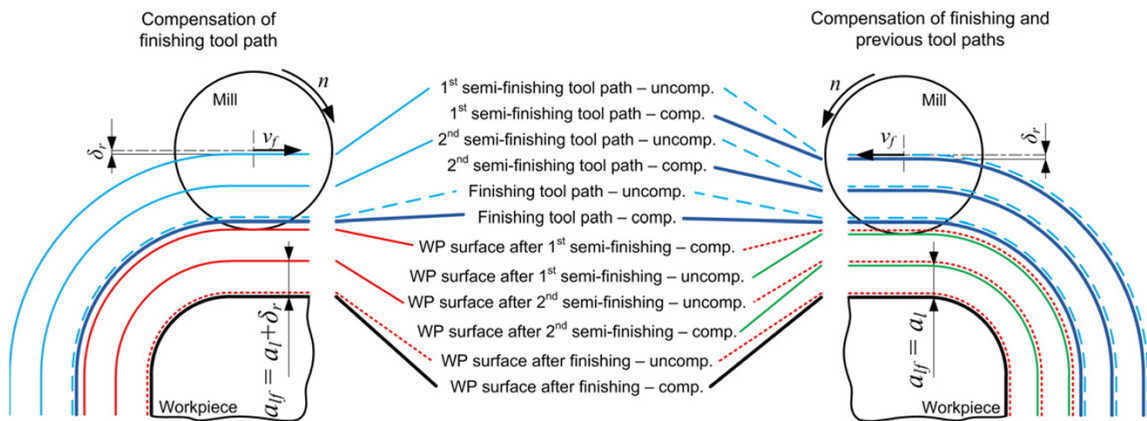


Figure II.5: Difference between compensation of finishing tool path and compensation of finishing and previous tool paths

Disadvantages:

- **Massive Computational Overhead:** Distorting a high-resolution STL file requires recalculating millions of triangular vertices. Executing a comprehensive 43-error volumetric kinematic model across an entire dense 3D mesh is computationally exorbitant and scales poorly for large or highly detailed workpieces.

II.5.2 Post-CAM Compensation: G-code Trajectory Modification

In contrast to the Pre-CAM approach, the Post-CAM strategy allows the CAM software to operate normally on the ideal, perfect 3D model. The volumetric compensation algorithm is then applied strictly to the discrete toolpath coordinates (X, Y, Z, A, C) within the finalized NC program, shifting the actual machine commands just before execution. Furthermore, this strategy allows for selective application, meaning the inverse kinematic solver can be isolated and applied exclusively to the finishing trajectories.

Advantages:

- **Targeted Computational Efficiency:** By applying the hybrid solver exclusively to the finishing passes, the algorithm mathematically corrects the tool exactly when it defines the final surface tolerance. This drastically reduces the data processing volume compared to whole-mesh distortion, making the algorithm highly efficient for rapid offline simulation.
- **Preservation of Original Geometry:** The CAD and CAM files remain pristine and nominal, simplifying revision tracking and allowing for rapid toolpath generation without the need to continually re-warp the source 3D geometry.

Disadvantages:

- **Variable Tool Load:** Because the roughing passes remain uncompensated, the finishing tool may encounter slight variations in the remaining stock material, requiring careful programming of the nominal finishing allowance to prevent tool deflection.
- **Interpolation Vulnerability:** The compensation algorithm only corrects the specific, discrete coordinate blocks exported by the post-processor. The machine's internal linear interpolation between those corrected points may still deviate slightly from the ideal continuous trajectory.

II.5.3 Justification for the Selected Strategy

When weighing the absolute precision of the developed 5-axis kinematic model against practical computational limits, the Post-CAM G-code modification strategy (applied strictly to finishing trajectories) emerges as the optimal methodology for this research.

While the STL distortion method provides the ideal scenario of uniform stock allowance during roughing operations, its immense computational inefficiency makes it unwieldy for iterative algorithm development and rapid deployment. By directly modifying the finishing G-code via the hybrid inverse kinematics solver, the compensation is executed exactly where the mechanical errors most critically impact the final workpiece tolerance. To mitigate the variable cutting forces caused by uncompensated roughing, a sufficiently conservative nominal stock allowance is maintained. This targeted approach ensures that high dimensional accuracy is achieved while maintaining a highly lean, computationally feasible digital workflow.

Conclusion

This chapter established the comprehensive mathematical framework required to model and compensate for geometric inaccuracies in a 5-axis bi-rotary CNC machine tool. By leveraging rigid body kinematics and Homogeneous Transformation Matrices (HTM), a complete forward kinematic model was developed, effectively encapsulating the 43 static and dynamic geometric errors inherent to the physical machine structure.

To invert this complex topological chain, a Hybrid Analytical-Numerical Inverse Kinematics solver was formulated. Recognizing the computational limitations and mathematical paradoxes of purely analytical decoupling, the developed algorithm applies a quasi-static assumption to the rotational error matrices. This approach successfully isolates the orientation compensation into a highly stable closed-form solution, while correcting the residual spatial positioning errors through an efficient, Jacobian-free fixed-point numerical iteration.

Finally, a comparative analysis of offline compensation strategies determined that applying the hybrid inverse solver strictly to the Post-CAM G-code of finishing trajectories yields the optimal balance of precision and efficiency. This targeted approach circumvents the immense computational overhead associated with whole-mesh STL distortion while ensuring absolute dimensional accuracy at the critical final stages of machining.

With the theoretical kinematics and offline compensation algorithms mathematically validated, the foundation is set to transition from analytical formulation to digital simulation. The subsequent chapter will detail the software implementation of this mathematical framework, analyzing the operational performance and convergence efficiency of the developed compensation engine.

DIGITAL IMPLEMENTATION AND SIMULATION RESULTS

III.1 Introduction

The theoretical kinematic models and the hybrid decoupled inverse kinematics compensation algorithm established in the previous chapter constitute the mathematical backbone of this research. However, the operative validation of the proposed strategy demands a concrete digital execution environment in which every matrix product, every angular correction, and every convergence decision can be observed, measured, and interrogated with precision. To satisfy this requirement, a dedicated computational engine was designed and implemented entirely within MATLAB (R2019a), operating as a self-contained, end-to-end simulation and pre-processing pipeline.

The engine functions as a high-fidelity *digital twin* of the Head-Head 5-axis prototype. It is not a simplified linearised approximation; it is a full nonlinear volumetric simulation that simultaneously activates all 43 independent geometric error terms (21 from the three linear axes and 22 from the two rotary axes (11 per axis)) and superimposes a stochastic vibration noise floor representative of a real factory environment. Against this deliberately hostile flawed model, the hybrid compensation engine executes its two-stage decoupled inverse kinematics solver, analytically resolving the rotary orientation before iteratively eliminating the residual translational position error.

This chapter presents an exhaustive, step-by-step dissection of the computational engine. For each logical block within the script, the verbatim MATLAB source code is extracted, presented in full, and followed by a precise functional explanation of what the code computes, why the specific numerical decisions were made, and how the computed quantities propagate downstream into the final compensated toolpath. The chapter culminates in a rigorous graphical analysis of two distinct simulation scenarios, demonstrating the solver's performance across both a controlled baseline trajectory and an aggressively coupled complex multi-axis stress path.

III.2 Simulation Environment and Initialization

Before a single matrix multiplication can be performed, the MATLAB workspace must be configured with a precisely defined and reproducible state.

III.2.1 Workspace Configuration

The script opens by unconditionally clearing the MATLAB workspace, purging all console output, closing all figure windows, and reseeding the random number generator from the system clock. This ensures that each execution of the engine produces a statistically independent, unrepeated set of 43 geometric error values, thus stress-testing the compensation algorithm against a novel error configuration on every run.

Listing III.1: Workspace initialization and entropy seeding.

```
1 clear; clc; close all;
2 rng('shuffle'); % New unique random seed drawn from system clock
```

The `rng('shuffle')` call is critical to the scientific validity of the simulation. By preventing any repetition of the random error set, the engine accumulates statistically meaningful evidence across successive runs rather than repeatedly testing the same geometric configuration.

III.2.2 Steps 1 and 2: Runtime Tool Input and Structural Constants

The two most physically variable parameters in any machining operation are the tool length and the machine's rigid structural geometry. The script handles these with a deliberate two-tier strategy.

Step 1 implements an interactive runtime input query for the tool length parameter L_{tool} , guarded by a validation loop that rejects empty, non-numeric, multi-element, or non-positive entries:

Listing III.2: Step 1 — Interactive tool length acquisition with input validation.

```
1 L_tool = input(' Enter the current Tool Length in mm (chuck face
   to tip): ');
2 while isempty(L_tool) || ~isnumeric(L_tool) || numel(L_tool) ~= 1
   || L_tool <= 0
3     fprintf(' Invalid --- please enter a single positive number.\n
   ');
4     L_tool = input(' Enter the current Tool Length in mm (chuck
   face to tip): ');
5 end
6 fprintf('\n Tool length set to: %.3f mm\n\n', L_tool);
```

By soliciting this value at runtime rather than hardcoding it, the engine becomes immediately reusable across different tooling setups (a short roughing end mill, a

long finishing cutter, or a specialist boring bar), without requiring any modification to the core source code. The while loop enforces strict numerical validity, the simulation will not proceed until a physically meaningful positive scalar has been confirmed.

Step 2 then establishes the immutable structural constants of the Head-Head bi-rotary configuration and computes the total kinematic reach from the A-axis pivot to the tool tip:

Listing III.3: Step 2 — Structural machine constants and total kinematic reach.

```

1 D_CA      = 313;    % mm --- Z-distance from C-axis pivot to A-axis
   pivot
2 L_spindle = 130;    % mm --- A-axis pivot to spindle chuck face
3 L_total   = L_spindle + L_tool; % Total reach: A-pivot to tool
   tip
4
5 NOISE_AMP = 0.005; % mm --- 1-sigma Gaussian vibration noise (std
   dev)

```

The parameter $D_{CA} = 313$ mm encodes the physical Z-offset between the two rotary pivots, which constitutes the primary geometric lever arm through which C-axis orientation errors propagate into translational TCP position errors. The sum $L_{total} = L_{spindle} + L_{tool}$ defines the second critical lever arm: the total distance from the A-axis pivot to the cutting tip. Every angular error at either rotary axis is amplified spatially in direct proportion to L_{total} .

III.2.3 Step 3: The HTM Function

The six fundamental spatial operators are defined as functions that dynamically generate 4×4 homogeneous transformation matrices on demand.

Listing III.4: Step 3 — The six elementary HTM anonymous function generators.

```

1 Tx = @(d) [1 0 0 d; 0 1 0 0; 0 0 1 0; 0 0 0 1];
2 Ty = @(d) [1 0 0 0; 0 1 0 d; 0 0 1 0; 0 0 0 1];
3 Tz = @(d) [1 0 0 0; 0 1 0 0; 0 0 1 d; 0 0 0 1];
4
5 Rx = @(a) [1 0 0 0;
6           0 cos(a) -sin(a) 0;
7           0 sin(a) cos(a) 0;
8           0 0 0 1];
9
10 Ry = @(b) [cos(b) 0 sin(b) 0;
11           0 1 0 0;
12           -sin(b) 0 cos(b) 0;
13           0 0 0 1];
14
15 Rz = @(c) [cos(c) -sin(c) 0 0;
16           sin(c) cos(c) 0 0;
17           0 0 1 0;
18           0 0 0 1];

```

Each function accepts a single scalar argument and returns the corresponding 4×4 matrix. Translation operators Tx, Ty, and Tz encode pure linear displacements along their respective axes. Rotation operators Rx, Ry, and Rz encode pure angular rotations about the X, Y, and Z axes respectively, accepting arguments in radians. These six primitives are the exclusive building blocks from which every composite transformation matrix in the entire simulation (ideal, flawed, and compensated) is assembled.

III.2.4 Step 4: The 43-Error Geometric Injection Engine

To thoroughly test the hybrid compensation algorithm, the simulation needs to go beyond fixed, deterministic testing. Evaluating the inverse kinematics solver against a single set of hardcoded errors only proves it works for that specific case. To ensure the algorithm is reliable enough for real-world manufacturing, the simulation uses a Monte Carlo sampling approach.

Rather than assigning static values to the machine's imperfections, the MATLAB script is architected to randomly generate the entire 43-error volumetric profile every time the simulation compiles. This is achieved by defining a custom randomized anonymous function bound by strict mechanical tolerance limits.

The custom function `rv(b)` uses MATLAB's native `rand()` generator, scaling the output to create a uniform distribution between $-b$ and $+b$. This allows the simulation to test the compensation algorithm against random geometric errors that fall within defined manufacturing tolerances.

Listing III.5: Step 4 — Randomisation function and error magnitude bounds.

```

1 rv = @(b) (2*rand() - 1) * b;
2
3 POS_B      = 0.200;    % mm --- max linear positioning error
4 STR_B      = 0.150;    % mm --- max straightness (parasitic) drift
5 SQ_B       = 0.002;    % rad --- max squareness error (~6.9 arcmin)
6 ANG_B      = 0.0005;   % rad --- angular pitch/roll/yaw per axis
7 ROT_LIN_B  = 0.050;    % mm --- rotary location / straightness shift
8 ROT_ANG_B  = 0.002;    % rad --- rotary angular offset (~6.9 arcmin)

```

The six boundary constants are defined to reflect the typical working tolerances of a standard 5-axis machining center. Each constant accounts for a specific category of mechanical error: POS_B represents linear positioning issues such as thermal drift and backlash; TR_B covers straightness deviations along the guideways; ANG_B accounts for angular errors caused by guideway imperfections; and ROT_ANG_B represents the squareness and wobble of the rotary axes.

To evaluate the compensation algorithm across these defined limits, the digital twin employs a Monte Carlo simulation. Because direct measurement of the machine's 43 geometric errors requires advanced metrology equipment, such as laser interferometers, empirical measurement was not feasible for this project. Instead, the

simulation relies on synthesized error data. The baseline assumptions for these error ranges are grounded in standard manufacturing tolerances documented. However, standard literature typically reflects the tight tolerances of commercial, industrial-grade machinery. Because the physical system is a custom-built prototype, its actual geometric errors are likely larger than those found in idealized commercial environments. Therefore, the assumed boundaries in the simulation were selected to realistically represent the practical physical imperfections of this specific build, rather than claiming perfect industrial precision. [?]

With the boundaries established, the script aggressively populates all 43 geometric variable (spanning the linear axes, the C-axis sub-chain, and the A-axis sub-chain) simultaneously.

The 21 linear axis errors are generated and immediately formatted into their constituent composite transformation matrices. The C-axis and A-axis each contribute 11 errors, yielding the complete 43-error profile:

Listing III.6: Step 4 — Generation of the 21 linear axis errors.

```

1 % === LINEAR AXES: 21 Errors ===
2 % Positioning errors (3)
3 e_x_pos = rv(POS_B); e_y_pos = rv(POS_B); e_z_pos = rv(POS_B);
4 % Straightness / parasitic translation errors (6)
5 e_x_iny = rv(STR_B); e_x_inz = rv(STR_B);
6 e_y_inx = rv(STR_B); e_y_inz = rv(STR_B);
7 e_z_inx = rv(STR_B); e_z_iny = rv(STR_B);
8 % Angular (Pitch, Roll, Yaw) errors per axis (9)
9 e_x_pit = rv(ANG_B); e_x_rol = rv(ANG_B); e_x_yaw = rv(ANG_B);
10 e_y_pit = rv(ANG_B); e_y_rol = rv(ANG_B); e_y_yaw = rv(ANG_B);
11 e_z_pit = rv(ANG_B); e_z_rol = rv(ANG_B); e_z_yaw = rv(ANG_B);
12 % Inter-axis squareness errors (3)
13 sq_xy = rv(SQ_B); sq_yz = rv(SQ_B); sq_zx = rv(SQ_B);
14 T_sq_xy = Rz(sq_xy); T_sq_yz = Rx(sq_yz); T_sq_zx = Ry(sq_zx);

```

The three squareness matrices T_{sq_xy} , T_{sq_yz} , and T_{sq_zx} are constructed directly at this stage because they are pure rotation matrices that do not depend on any axis command value; they are fully determined by the assembly geometry of the machine structure and remain constant for the entire simulation run. The 22 rotary axis errors follow:

Listing III.7: Step 4 — Generation of the 22 rotary axis errors (11 per axis).

```

1 % === C-AXIS: 11 Errors ===
2 % Static location / assembly errors (5)
3 c_loc_X = rv(ROT_LIN_B); c_loc_Y = rv(ROT_LIN_B);
4 c_sq_A = rv(ROT_ANG_B); c_sq_B = rv(ROT_ANG_B);
5 c_zero = rv(ROT_ANG_B);
6 % Dynamic component errors (6)
7 e_c_ang = rv(ROT_ANG_B);
8 e_c_radX = rv(ROT_LIN_B); e_c_radY = rv(ROT_LIN_B);
9 e_c_axZ = rv(ROT_LIN_B);
10 e_c_wobX = rv(ROT_ANG_B); e_c_wobY = rv(ROT_ANG_B);
11

```

```

12 % === A-AXIS: 11 Errors ===
13 % Static location / assembly errors (5)
14 a_loc_Y = rv(ROT_LIN_B); a_loc_Z = rv(ROT_LIN_B);
15 a_sq_B = rv(ROT_ANG_B); a_sq_C = rv(ROT_ANG_B);
16 a_zero = rv(ROT_ANG_B);
17 % Dynamic component errors (6)
18 e_a_ang = rv(ROT_ANG_B);
19 e_a_radY = rv(ROT_LIN_B); e_a_radZ = rv(ROT_LIN_B);
20 e_a_axX = rv(ROT_LIN_B);
21 e_a_wobY = rv(ROT_ANG_B); e_a_wobZ = rv(ROT_ANG_B);

```

This code block represents a critical shift in the simulation architecture. In physical reality, mechanical flaws do not occur in isolation; a drifting C-axis pivot is inherently cross-coupled with the wobbling A-axis and the pitching X-axis gantry. By populating the entire matrix with randomized variables bounded by ROT_ANG_B and POS_B, the script creates a chaotic, heavily coupled mathematical environment that mirrors the statistical variations found across different physical bi-rotary spindle prototypes on a factory floor.

Furthermore, to guarantee that the compensation solver is not merely overfitting to a sterile geometric equation, the Monte Carlo framework is compounded by the injection of high-frequency kinematic chatter.

```

1 NOISE_AMP = 0.005; % mm --- 1-sigma Gaussian vibration noise per
   axis
2 ...
3 % --- ADD VIBRATION NOISE ---
4 error_pts(:, i) = geom_pts(:, i) + randn(3,1) * NOISE_AMP;

```

These 43 parameters define an extremely rich, physically grounded volumetric error field through which every subsequent computation must navigate.

III.3 Toolpath Parsing and Simulation Scenarios (Step 5)

The compensating engine is designed to ingest real machining trajectories exported by a simple Python code that parses raw G-code files. This conversion step is strictly necessary because raw G-code contains various machine parameters, headers, and auxiliary syntax that the kinematic solver does not need. To ensure the MATLAB script can focus entirely on the geometric path, the Python code filters out this extra data. The toolpath is then supplied as a clean, five-column CSV file (toolpath1.csv) containing purely the commanded coordinates $[X_{mm}, Y_{mm}, Z_{mm}, A_{deg}, C_{deg}]$ for each discrete toolpath point. Step 5 manages this data ingestion pipeline, along with a built-in fallback trajectory generation mechanism.

Listing III.8: Step 5 — CSV import pipeline and fallback trajectory generator.

```

1 csv_in = 'toolpath1.csv';
2

```

```

3 if ~isfile(csv_in)
4     N_s = 360;    r_s = 30;    z_s = -50;
5     t_s = linspace(0, 2*pi, N_s)';
6     a_s = 15 * sin(t_s);    % A oscillates +/- 15 degrees
7     c_s = rad2deg(t_s);    % C sweeps 0 to 360 degrees
8     writematrix([r_s*cos(t_s), r_s*sin(t_s), z_s*ones(N_s,1), a_s,
9                 c_s], csv_in);
10
11 end
12
13 raw = readmatrix(csv_in);
14 if size(raw, 2) < 5
15     error('CSV_needs_5_columns: [X_mm, Y_mm, Z_mm, A_deg, C_deg]');
16 end
17
18 x_cmd = raw(:,1)';    y_cmd = raw(:,2)';    z_cmd = raw(:,3)';
19 a_cmd = deg2rad(raw(:,4)');    % Convert to
    radians
20 c_cmd = deg2rad(raw(:,5)');    % Convert to
    radians
21 N = length(x_cmd);

```

Upon reading the file, the script validates that at least five data columns are present and immediately transposes the column vectors into row vectors for efficient vectorised processing throughout the engine. Critically, the angular coordinates for both rotary axes are converted from degrees to radians at this ingestion stage, ensuring that all downstream trigonometric operations throughout the solver are consistently performed in the native radian domain.

III.3.1 The Two Simulation Scenarios

To thoroughly evaluate the solver's performance across different levels of computational and kinematic complexity, the methodology prescribes the execution of two distinct simulation scenarios.

III.3.1.1 Scenario 1: The Baseline Multi-Axis Trajectory

The first scenario employs a basic, low-density toolpath consisting of around 20 discrete coordinate points. Rather than a purely planar test, this baseline trajectory actively engages both the A and C rotary axes, introducing fundamental kinematic cross-coupling into the system. The primary purpose of this scenario is diagnostic: it provides a clean, easily traceable dataset to verify that the CSV data ingestion pipeline, the Stage 1 closed-form orientation solver, and the Stage 2 translational loop seamlessly integrate. By utilizing a small coordinate array, any mathematical or programmatic anomalies in the core compensation logic can be rapidly isolated and resolved before introducing heavier computational loads.

III.3.1.2 Scenario 2: The High-Density Complex Trajectory

The second scenario dramatically scales the computational demands by processing a highly complex multi-axis trajectory comprising over 20,000 discrete points. This toolpath is engineered to stress-test the algorithm against the harshest possible error coupling conditions under a realistic, heavy manufacturing payload. By forcing continuous, simultaneous motion across the entire Head-Head kinematic chain, this trajectory ensures that all 43 geometric error terms are continuously driven through their full physical range. This scenario serves as the primary validation environment, proving that the hybrid inverse kinematics architecture can efficiently resolve geometrically distinct compensation problems at every single one of the 20,000+ points without computational degradation.

III.4 Forward Kinematics Evaluation (Step 6)

Step 6 constitutes the simulation's primary physics engine. Executing a sequential loop over every point in the loaded trajectory, it evaluates both the *ideal* machine kinematics and the *real, fully flawed* machine kinematics, producing the baseline volumetric error field that the hybrid solver must subsequently eliminate.

III.4.1 Memory Pre-allocation and the Evaluation Loop

Three output arrays are pre-allocated before the loop begins, as this eliminates the computational cost of dynamic array growth within the loop body:

Listing III.9: Step 6 — Output array pre-allocation and loop structure.

```

1 ideal_pts = zeros(3, N);    % Ideal TCP positions
2 geom_pts  = zeros(3, N);    % Real TCP positions (geometry only)
3 error_pts = zeros(3, N);    % Real TCP positions (geometry + noise)
4
5 for i = 1:N
6     % ... (see following listings)
7 end

```

Each array is dimensioned as $3 \times N$, storing the three-dimensional Cartesian coordinates of the tool centre point (TCP) for each of the N toolpath points as column vectors. Separation into two distinct real-geometry arrays (`geom_pts` and `error_pts`) allows the script to independently analyse the purely geometric error contribution and the noise-contaminated error contribution at any point in the post-processing pipeline.

III.4.2 Ideal Kinematic Chain Evaluation

For each toolpath point, the ideal TCP position is computed by assembling the perfect, error-free kinematic chain and extracting the translational column of the

resulting composite transformation matrix:

Listing III.10: Step 6 — Ideal machine kinematic chain evaluation.

```

1 T_ideal = Tx(x_cmd(i)) ...
2         * Ty(y_cmd(i)) ...
3         * Tz(z_cmd(i)) ...
4         * Rz(c_cmd(i)) ...
5         * Tz(-D_CA) ...
6         * Rx(a_cmd(i)) ...
7         * Tz(-L_total);
8
9 ideal_pts(:, i) = T_ideal(1:3, 4);

```

The product strictly follows the kinematic chain order established in **Chapter2** : Base $\rightarrow X \rightarrow Y \rightarrow Z \rightarrow C \rightarrow$ (pivot offset D_{CA}) $\rightarrow A \rightarrow$ (tool reach L_{total}). The final matrix T_{ideal} is a 4×4 homogeneous transformation; column 4, rows 1–3 contains the TCP’s Cartesian position in the world frame. The intermediate translation $Tz(-D_{CA})$ encodes the fixed structural offset between the C-axis pivot and the A-axis pivot, while $Tz(-L_{total})$ propagates the coordinate frame from the A-axis pivot to the physical tool tip. Both are signed negative in accordance with the right-hand coordinate convention where the tool hangs downward from the spindle.

III.4.3 Flawed Machine Kinematic Chain Evaluation

Immediately following the ideal computation, the script assembles the fully flawed kinematic chain using the 43 error parameters generated in Step 4. Each of the three linear axes is modelled with its full six-degree-of-freedom error set: one positioning error, two straightness errors, and three angular errors (pitch, roll, and yaw):

Listing III.11: Step 6 — Flawed linear axis transformation matrix assembly.

```

1 T_X_r = Tx(x_cmd(i)+e_x_pos) * Ty(e_x_iny) * Tz(e_x_inz) ...
2         * Rx(e_x_rol) * Ry(e_x_pit) * Rz(e_x_yaw);
3
4 T_Y_r = Ty(y_cmd(i)+e_y_pos) * Tx(e_y_inx) * Tz(e_y_inz) ...
5         * Rx(e_y_pit) * Ry(e_y_rol) * Rz(e_y_yaw);
6
7 T_Z_r = Tz(z_cmd(i)+e_z_pos) * Tx(e_z_inx) * Ty(e_z_iny) ...
8         * Rx(e_z_pit) * Ry(e_z_yaw) * Rz(e_z_rol);

```

In each linear axis matrix, the commanded displacement is contaminated at the first operator by its corresponding positioning error (e.g., $x_{cmd}(i) + e_{x,pos}$), after which the four parasitic motion terms (two straightness translations and three angular rotations) are composed via matrix multiplication. The ordering of these parasitic operators within each axis block follows the standard convention for the specific error-motion sequence of that physical axis.

The two rotary axis matrices are assembled through a three-stage composite product that separates the static location errors from the dynamic component errors and from the nominal angular command:

Listing III.12: Step 6 — Flawed rotary axis transformation matrix assembly.

```

1 % C-Axis: location offsets * dynamic errors * nominal rotation
2 T_C_loc = Tx(c_loc_X) * Ty(c_loc_Y) * Rx(c_sq_A) * Ry(c_sq_B);
3 T_C_dyn = Tx(e_c_radX) * Ty(e_c_radY) * Tz(e_c_axZ) ...
4           * Rx(e_c_wobX) * Ry(e_c_wobY);
5 T_C_r    = T_C_loc * T_C_dyn * Rz(c_cmd(i) + c_zero + e_c_ang);
6
7 % A-Axis: location offsets * dynamic errors * nominal rotation
8 T_A_loc = Ty(a_loc_Y) * Tz(a_loc_Z) * Ry(a_sq_B) * Rz(a_sq_C);
9 T_A_dyn = Tx(e_a_axX) * Ty(e_a_radY) * Tz(e_a_radZ) ...
10          * Ry(e_a_wobY) * Rz(e_a_wobZ);
11 T_A_r    = T_A_loc * T_A_dyn * Rx(a_cmd(i) + a_zero + e_a_ang);

```

In the final rotation operator of each axis, the nominal command is additively combined with both the zero-offset error and the angular positioning error before being passed to the rotation function. This correctly models the physical reality that the machine's homing system cannot distinguish between the nominal command and these systematic angular biases; it executes them as a single compound rotation. The complete volumetric HTM chain is then assembled and evaluated:

Listing III.13: Step 6 — Full volumetric HTM chain and noise injection.

```

1 T_real = T_X_r * T_sq_xy ...
2         * T_Y_r * T_sq_yz ...
3         * T_Z_r * T_sq_zx ...
4         * T_C_r * Tz(-D_CA) ...
5         * T_A_r * Tz(-L_total);
6
7 geom_pts(:, i) = T_real(1:3, 4);
8 error_pts(:, i) = geom_pts(:, i) + randn(3,1) * NOISE_AMP;

```

III.4.4 Baseline Error Quantification

Following the loop, the volumetric error magnitude for each toolpath point is computed as the Euclidean norm of the difference between the real and ideal TCP positions:

Listing III.14: Step 6 — Volumetric error vector and pre-compensation statistics.

```

1 total_dev = error_pts - ideal_pts;
2 err_mag   = sqrt(sum(total_dev.^2, 1));
3
4 fprintf('=== BEFORE COMPENSATION ===\n');
5 fprintf(' Max   : %.5f mm\n', max(err_mag));
6 fprintf(' Mean  : %.5f mm\n', mean(err_mag));
7 fprintf(' RMS   : %.5f mm\n\n', sqrt(mean(err_mag.^2)));

```

The resulting vector `err_mag` is a $1 \times N$ array of scalar volumetric deviations, one per toolpath point, expressed in millimetres. This constitutes the definitive baseline against which all subsequent compensation improvements are measured. The three

statistical descriptors (maximum deviation, mean deviation, and root-mean-square deviation) are reported to the console immediately, providing the operator with an instant quantitative assessment of the machine’s uncompensated geometric health.

III.5 The Hybrid Inverse Kinematics Compensation Engine (Step 7)

Step 7 is the algorithmic centrepiece of the entire simulation. All previous steps have been preparatory: building the mathematical operators, populating the error model, evaluating the baseline kinematics. Step 7 is where those foundations are actively exploited. The compensation engine deployed here is a purpose-designed hybrid architecture that integrates a closed-form analytical rotary inverse kinematics solver with a Jacobian-free fixed-point translational correction loop. The following subsections dissect every computational component of this engine at the source-code level.

III.5.1 Solver Configuration and Pre-computation

III.5.1.1 Convergence Parameters and Solver Bounds

The outer fixed-point iteration loop is governed by two control parameters defined at the top of Step 7:

Listing III.15: Step 7 — Solver convergence threshold and iteration limit.

```
1 epsilon    = 0.0001;    % Convergence threshold: 0.1 micron
2 max_iters  = 50;        % Failsafe iteration cap
```

The convergence threshold `epsilon = 0.0001 mm` sets the target maximum volumetric TCP deviation at $0.1\ \mu\text{m}$ — a tolerance well below any identifiable mechanical noise floor in a physical machining environment. The solver is considered to have converged when every single toolpath point simultaneously achieves a volumetric residual below this threshold. The `max_iters = 50` cap prevents infinite looping in pathological cases where convergence cannot be reached, triggering a console warning that alerts the operator to a potential model inconsistency.

III.5.1.2 Pre-computation of the Fixed Rotary Orientation Error Matrices

Listing III.16: Step 7 — Pre-computation of fixed rotary sub-chain matrices and 3x3 rotation block extraction.

```
1 T_C_loc = Tx(c_loc_X) * Ty(c_loc_Y) * Rx(c_sq_A) * Ry(c_sq_B);
2 T_C_dyn = Tx(e_c_radX) * Ty(e_c_radY) * Tz(e_c_axZ) ...
3           * Rx(e_c_wobX) * Ry(e_c_wobY);
4 T_A_loc = Ty(a_loc_Y) * Tz(a_loc_Z) * Ry(a_sq_B) * Rz(a_sq_C);
5 T_A_dyn = Tx(e_a_axX) * Ty(e_a_radY) * Tz(e_a_radZ) ...
```

```

6         * Ry(e_a_wobY) * Rz(e_a_wobZ);
7
8 % Combine into full 4x4 products, then extract 3x3 rotation blocks
9 T_C_err_full = T_C_loc * T_C_dyn;
10 T_A_err_full = T_A_loc * T_A_dyn;
11 R_C_err      = T_C_err_full(1:3, 1:3);
12 R_A_err      = T_A_err_full(1:3, 1:3);
13 R_C_err_T    = R_C_err';

```

The critical observation exploited here is that the full 4×4 composite matrices `T_C_err_full` and `T_A_err_full` carry *both* translational components (which affect TCP position) and rotational components (which affect tool orientation). However, for the purpose of the orientation IK solver, only the orientational information is required. The script extracts the 3×3 upper-left rotation sub-block from each composite matrix and assigns it to `R_C_err` and `R_A_err` respectively, following the R2019a-compatible pattern of first assigning the product to a temporary variable before indexing. The matrix transpose `R_C_err_T = R_C_err'` is pre-computed because, for a rotation matrix, the transpose is identically equal to the matrix inverse; this single cached operation eliminates a matrix inversion from the inner loop of the solver.

III.5.1.3 Pre-extraction of Scalar A-Axis IK Coefficients

A further layer of pre-computation extracts the three scalar constants required by the A-axis harmonic equation solver. Because these scalars are derived solely from `R_A_err` (which is fixed for the entire run) computing them once and caching them eliminates three repeated indexed array lookups per toolpath point per iteration:

Listing III.17: Step 7 — Pre-computation of A-axis harmonic equation scalar coefficients.

```

1 P_A    = R_A_err(3,2);           % Coefficient of sin(a_tot)
2 Q_A    = -R_A_err(3,3);          % Coefficient of cos(a_tot)
3 rho_A  = sqrt(P_A^2 + Q_A^2);    % Amplitude (auxiliary angle method)
4 phi_A  = atan2(Q_A, P_A);        % Phase shift (auxiliary angle method)

```

These four scalars — `P_A`, `Q_A`, `rho_A`, and `phi_A` — are the pre-factored components of the auxiliary-angle decomposition applied to the A-axis orientation constraint equation. Their significance is explained in full in Section III.5.5.

III.5.1.4 Solver State Initialisation

The five compensated command arrays are initialised to their nominal G-code values. The linear commands begin at the commanded positions as the initial guess for the fixed-point iteration. The rotary command arrays are also initialised to the commanded angles, though these will be completely overwritten by the closed-form IK solver on the very first iteration:

Listing III.18: Step 7 — Initialisation of all five compensated command arrays.

```

1 x_comp = x_cmd;    y_comp = y_cmd;    z_comp = z_cmd;
2 a_comp = a_cmd;    % Overwritten by closed-form IK on first
   iteration
3 c_comp = c_cmd;    % Overwritten by closed-form IK on first
   iteration
4
5 orig_max_err       = max(err_mag);
6 convergence_history = [];

```

III.5.2 The Outer Fixed-Point Iteration Loop

The outer solver loop is a for loop bounded by `max_iters`, though in practice it terminates early via the `break` statement upon convergence. On each iteration, a fresh $3 \times N$ result array is pre-allocated to receive the simulated TCP positions computed using the current set of compensated commands:

Listing III.19: Step 7 — Outer fixed-point iteration loop structure and per-iteration allocation.

```

1 for iter = 1:max_iters
2
3     sim_real_pts = zeros(3, N);
4
5     for i = 1:N
6         % Stage 1: Rotary IK (Sections 3.5.3 to 3.5.6)
7         % Stage 2: Full HTM evaluation (Section 3.5.7)
8     end
9
10    % Convergence check and linear update (Section 3.5.8)
11 end

```

The inner for `i = 1:N` loop processes each toolpath point sequentially. For each point, the two-stage solver is executed in full: Stage 1 analytically computes the compensated rotary angles, and Stage 2 evaluates the complete 43-error forward kinematic chain using both the freshly solved rotary angles and the current linear command estimates, depositing the resulting TCP position into column i of `sim_real_pts`. After all N points have been processed, control returns to the outer loop for convergence assessment and linear command update.

III.5.3 Stage 1, Step 1a: Ideal Tool Orientation Vector Extraction

The first action inside the inner loop is the computation of the ideal target orientation vector, the desired tool axis direction that the compensation algorithm must force the flawed machine to reproduce.

Listing III.20: Step 7, Stage 1 — Ideal tool orientation vector computation.

```

1 c_i    = c_cmd(i);    a_i    = a_cmd(i);

```

```

2 sin_c = sin(c_i);          cos_c = cos(c_i);
3 sin_a = sin(a_i);          cos_a = cos(a_i);
4
5 d_ideal = [-sin_c * sin_a;
6             cos_c * sin_a;
7             -cos_a      ];

```

This three-element unit vector represents the direction of the tool's physical cutting axis in the world coordinate frame, as dictated by the original, unmodified G-code commands. It is computed by analytically evaluating the product $\mathbf{R}_z(c) \cdot \mathbf{R}_x(a) \cdot [0, 0, -1]^T$, where $[0, 0, -1]^T$ is the tool's axis vector in its own local frame (pointing from spindle downward toward the workpiece). The six trigonometric values are cached as local scalars at the top of the inner loop body to prevent repeated sin/cos evaluations across the multiple locations within Stage 1 where they would otherwise be needed.

III.5.4 Stage 1, Step 1b: C-Error Frame Transformation

The flawed machine's complete orientation chain, from the world frame to the tool tip, takes the form $\mathbf{R}_{C, \text{err}} \cdot \mathbf{R}_z(c_{\text{tot}}) \cdot \mathbf{R}_{A, \text{err}} \cdot \mathbf{R}_x(a_{\text{tot}}) \cdot [0, 0, -1]^T = \mathbf{d}_{\text{ideal}}$. To decompose this equation into two separately solvable scalar problems, the script first removes the influence of the C-axis error sub-matrix by pre-multiplying both sides by its transpose:

Listing III.21: Step 7, Stage 1 — Removal of C-error influence by pre-multiplication with transpose.

```

1 d_int = R_C_err_T * d_ideal;    % 3x1 intermediate target vector

```

This single line produces `d_int`, the *intermediate target vector*. Physically, `d_int` is the ideal orientation vector as it must appear after the C-axis rotation and before the A-axis rotation operates on it, when all C-sub-chain orientation errors have been factored out. The identity $\mathbf{R}_{C, \text{err}}^T = \mathbf{R}_{C, \text{err}}^{-1}$ guarantees this factorisation is exact and lossless. The resulting equation to be solved is now $\mathbf{R}_z(c_{\text{tot}}) \cdot \mathbf{R}_{A, \text{err}} \cdot \mathbf{R}_x(a_{\text{tot}}) \cdot [0, 0, -1]^T = \mathbf{d}_{\text{int}}$, which contains two unknowns: c_{tot} and a_{tot} . The strategy for separating them is the Z-component invariance property of the $\mathbf{R}_z$ operator, exploited in the following step.

III.5.5 Stage 1, Step 1c: A-Axis Closed-Form Solution with Branch Selection

The central insight of the A-axis solution is that the $\mathbf{R}_z$ operator, which represents pure rotation about the Z-axis, operates exclusively within the XY plane and *cannot, under any circumstances, alter the Z-component of the vector it acts upon*. Therefore, the Z-component of the left-hand side of the orientation equation must already equal `d_int(3)` before $\mathbf{R}_z(c_{\text{tot}})$ is applied. This constraint involves only a_{tot} and the known fixed matrix $\mathbf{R}_{A, \text{err}}$:

Listing III.22: Step 7, Stage 1 — Closed-form A-axis solution via auxiliary-angle method.

```

1 D_A      = d_int(3);
2 sin_arg = max(-1.0, min(1.0, D_A / rho_A)); % Numerical safety
   clamp
3 a_base  = asin(sin_arg);
4 a_sol1  = a_base - phi_A;
5 a_sol2  = pi - a_base - phi_A;
6
7 if abs(a_sol1 - a_i) <= abs(a_sol2 - a_i)
8     a_total = a_sol1;
9 else
10    a_total = a_sol2;
11 end
12 a_comp(i) = a_total - a_zero - e_a_ang;

```

The pre-computed scalar coefficients P_A , Q_A , ρ_A , and ϕ_A from Section III.5.1 are the auxiliary-angle decomposition parameters that reduce the Z-component constraint into the standard form $\rho_A \sin(a_{\text{tot}} + \phi_A) = D_A$. Solving this equation analytically yields two valid candidate solutions:

- **Solution 1:** $a_{\text{tot},1} = \arcsin(D_A/\rho_A) - \phi_A$
- **Solution 2:** $a_{\text{tot},2} = \pi - \arcsin(D_A/\rho_A) - \phi_A$

Both solutions are geometrically valid from the perspective of the orientation constraint. However, they correspond to physically opposite machine configurations: one represents the intended tool tilt, and the other its supplement a 180° complementary angle that would require the A-axis to execute a violent, physically destructive head-flip manoeuvre to reach. The `if` conditional selects the branch whose angular value is closest to the original commanded angle `a_i`, implementing a minimum-displacement branch selection rule that guarantees physical continuity of the tool orientation throughout the toolpath. The numerical safety clamp `max(-1.0, min(1.0, D_A / rho_A))` prevents floating-point rounding errors from producing an argument to `asin` outside the valid domain $[-1, +1]$, which would otherwise generate a NaN and corrupt the entire compensation result for that point.

Once `a_total` is determined, the compensated A-axis command is recovered by stripping out the two systematic angular biases that the machine applies on top of the commanded value:

$$a_{\text{comp}}(i) = a_{\text{tot}} - a_{\text{zero}} - e_{a,\text{ang}}$$

This subtraction pre-encodes the corrections for both the zero-homing offset (`a_zero`) and the angular positioning error (`e_a_ang`), such that when the machine's controller adds them back during execution, the net physical rotation is exactly `a_total`.

III.5.6 Stage 1, Step 1d: C-Axis Alignment via 2D Cross-Product and Singularity Guard

With a_{total} fully determined, the A-axis contribution to the orientation can be evaluated, producing the intermediate tool orientation vector v that the C-axis rotation must now align with the target vector d_{int} :

Listing III.23: Step 7, Stage 1 — C-axis 2D rotation alignment and singularity protection.

```

1 a_vec = [0; sin(a_total); -cos(a_total)]; % Rx(a_tot)*[0;0;-1]
2 v      = R_A_err * a_vec;                % 3x1 intermediate
      vector
3
4 v_xy_mag = sqrt(v(1)^2 + v(2)^2);
5 if v_xy_mag < 1e-8
6     c_total = c_i + c_zero + e_c_ang;      % Singularity: C
      indeterminate
7 else
8     c_total = atan2(d_int(2)*v(1) - d_int(1)*v(2), ...
9                   d_int(1)*v(1) + d_int(2)*v(2));
10 end
11 c_comp(i) = c_total - c_zero - e_c_ang;
```

The vector $a_{\text{vec}} = [0, \sin(a_{\text{tot}}), -\cos(a_{\text{tot}})]^T$ is the analytical result of evaluating $R_x(a_{\text{tot}}) \cdot [0, 0, -1]^T$. Post-multiplication by R_{A_err} then incorporates all A-axis orientation errors, producing the physical intermediate vector v that represents the tool's orientation state immediately after the A-axis and its associated errors have acted upon it, but before the C-axis rotation has been applied.

The numerator argument to `atan2` is the 2D cross product (proportional to $\sin(c_{\text{tot}})$) and the denominator argument is the 2D dot product (proportional to $\cos(c_{\text{tot}})$). Because `atan2` processes both arguments simultaneously and spans the full range $(-\pi, +\pi]$, it is inherently immune to the quadrant ambiguities that would afflict a naive `atan` implementation. There is no branch selection problem for the C-axis; the `atan2` formula delivers the unique, physically correct solution directly.

The singularity guard on $v_{xy_mag} < 1e-8$ handles the degenerate configuration in which the tool is pointing exactly straight down ($a_{\text{total}} \approx 0$). In this configuration, the XY projection of v vanishes, and the C-axis rotation has no observable effect on the tool's spatial direction (the problem is geometrically indeterminate). Rather than generating an undefined `atan2(0,0)` result, the guard assigns c_{total} to the commanded value with its known systematic offsets, effectively maintaining the C-axis at the operator's intended position through the singular zone. As with the A-axis, the compensated C command is recovered by subtracting the C-axis systematic biases: $c_{\text{comp}}(i) = c_{\text{tot}} - c_{\text{zero}} - e_{c,\text{ang}}$.

III.5.7 Stage 2: Full HTM Evaluation with Compensated Commands

With both compensated rotary commands locked in for the current point, Stage 2 constructs and evaluates the complete 43-error forward kinematic chain using the current estimates of the compensated linear commands alongside the freshly solved rotary commands. This evaluation is structurally identical to the one performed in Step 6, but it uses the `x_comp`, `y_comp`, `z_comp`, `c_comp`, and `a_comp` arrays in place of the original `x_cmd`, `y_cmd`, `z_cmd`, `c_cmd`, and `a_cmd` arrays:

Listing III.24: Step 7, Stage 2 — Full 43-error HTM chain with compensated commands.

```

1 T_X_c = Tx(x_comp(i)+e_x_pos) * Ty(e_x_iny) * Tz(e_x_inz) ...
2       * Rx(e_x_rol) * Ry(e_x_pit) * Rz(e_x_yaw);
3
4 T_Y_c = Ty(y_comp(i)+e_y_pos) * Tx(e_y_inx) * Tz(e_y_inz) ...
5       * Rx(e_y_pit) * Ry(e_y_rol) * Rz(e_y_yaw);
6
7 T_Z_c = Tz(z_comp(i)+e_z_pos) * Tx(e_z_inx) * Ty(e_z_iny) ...
8       * Rx(e_z_pit) * Ry(e_z_yaw) * Rz(e_z_rol);
9
10 T_C_c = T_C_loc * T_C_dyn * Rz(c_comp(i) + c_zero + e_c_ang);
11 T_A_c = T_A_loc * T_A_dyn * Rx(a_comp(i) + a_zero + e_a_ang);
12
13 T_comp = T_X_c * T_sq_xy * T_Y_c * T_sq_yz ...
14         * T_Z_c * T_sq_zx * T_C_c * Tz(-D_CA) ...
15         * T_A_c * Tz(-L_total);
16
17 sim_real_pts(:, i) = T_comp(1:3, 4);

```

The role of this Stage 2 evaluation is diagnostic: given the best current estimates for all five axis commands, it asks the question "*where does the flawed machine actually place the tool tip?*" The answer, stored in column *i* of `sim_real_pts`, will be compared against the ideal target position `ideal_pts(:,i)` to reveal the residual position error that the Stage 2 linear update must correct. Note that the rotary matrices `T_C_c` and `T_A_c` now use `c_comp(i)` and `a_comp(i)` (the analytically optimised rotary commands from Stage 1) rather than the original commanded angles. This is the key operational distinction between the new 5-axis solver and the rudimentary 3-axis correction it supersedes.

III.5.8 Convergence Assessment and Linear Command Update

Following the completion of the inner for *i* loop, the outer iteration computes the residual volumetric error across the entire toolpath and makes the convergence/update decision:

Listing III.25: Step 7 — Residual computation, convergence check, and dampened linear fixed-point update.

```

1 residual_dev = sim_real_pts - ideal_pts;
2 residual_mag = sqrt(sum(residual_dev.^2, 1));
3 max_residual = max(residual_mag);
4
5 convergence_history(iter) = max_residual;
6
7 improvement_pct = (1 - (max_residual / orig_max_err)) * 100;
8 fprintf(' [Iter %02d] Max Dev: %.6f mm | Improvement: %6.2f%%\n',
9     ...
10     iter, max_residual, improvement_pct);
11
12 if max_residual <= epsilon
13     fprintf(' --> Converged below %.4f mm in %d iterations!\n\n',
14         ...
15         epsilon, iter);
16     break;
17 end
18
19 % Dampened feedforward update to prevent oscillation
20 alpha = 0.7; % Relaxation factor
21 x_comp = x_comp - alpha * residual_dev(1,:);
22 y_comp = y_comp - alpha * residual_dev(2,:);
23 z_comp = z_comp - alpha * residual_dev(3,:);

```

The residual deviation matrix `residual_dev` is a $3 \times N$ array containing the Cartesian error vector at every toolpath point. Its column-wise Euclidean norms yield `residual_mag`, and the maximum across all N points is `max_residual` the single scalar that determines convergence. The current value of `max_residual` is appended to `convergence_history` for subsequent plotting, and the iteration's percentage improvement relative to the original uncompensated error is reported to the console.

If `max_residual` $\leq$ `epsilon`, the `break` statement terminates the outer loop immediately, locking in the current values of `x_comp`, `y_comp`, `z_comp`, `a_comp`, and `c_comp` as the final compensated toolpath commands. If convergence has not yet been achieved, the linear commands are updated using a dampened fixed-point feedforward rule. Rather than subtracting the raw residual entirely, a mathematical relaxation factor (α) is applied. While the physical X, Y, and Z axes act orthogonally, the severe spatial cross-coupling generated by the 43 geometric errors (particularly the angular lever-arm amplifications) can cause a strict 1:1 compensation step to overshoot the target, inducing mathematical oscillation. By decrementing the linear commands by the residual deviation multiplied by this α parameter, the iterative step is safely dampened, ensuring stable, monotonic convergence. The rotary commands `a_comp` and `c_comp` are *not* updated here; they are exact analytical solutions re-computed fresh from Stage 1 at the beginning of the next iteration.

III.6 Compensated G-Code Reconstruction and Export (Step 8)

Having produced the converged set of compensated commands for all five axes, Step 8 exports the complete corrected toolpath to a structured CSV file for downstream G-code reconstruction:

Listing III.26: Step 8 — Five-axis compensated toolpath export.

```

1 csv_out = 'Compensated_Path.csv';
2 writetable( ...
3     array2table([x_comp', y_comp', z_comp', ...
4                 rad2deg(a_comp'), rad2deg(c_comp')], ...
5                 'VariableNames', {'X_mm', 'Y_mm', 'Z_mm', 'A_deg', '
6                                     C_deg'}), ...
7     csv_out);
8 fprintf(' Columns: X_mm | Y_mm | Z_mm | A_deg (compensated) |
9         C_deg (compensated)\n');
10 disp('Calling Python to reconstruct final G-code file...');
    system('py csv_to_gcode.py');
```

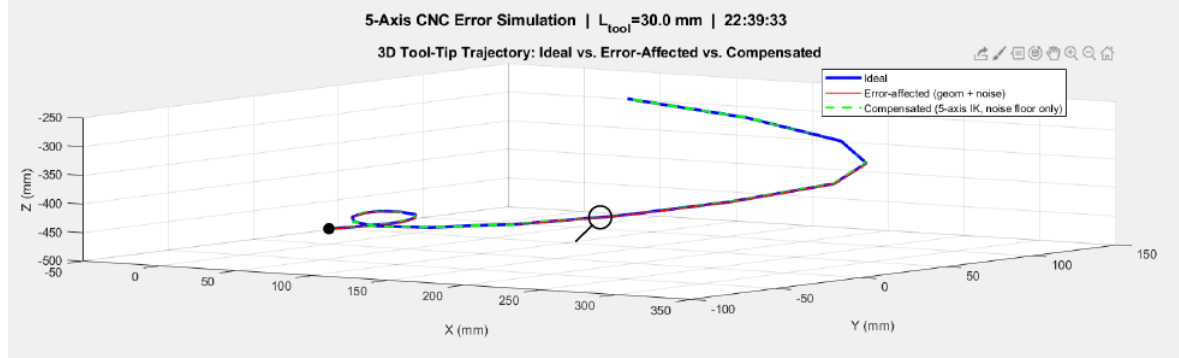
The five column vectors are first transposed from row arrays (the internal working format) to column vectors, as required by `array2table`. The rotary commands are simultaneously converted back from radians to degrees (the native unit of G-code angular motion commands). The resulting table is written to `Compensated_Path.csv` with explicit, human-readable column headers. This file serves as the input to the companion Python code (`csv_to_gcode.py`), which reconstructs the final executable G-code file by re-inserting the original M-codes, feed rate specifications, tool change commands, and all other non-positional G-code elements that the MATLAB engine does not manipulate.

III.7 Graphical Analysis of Simulation Results

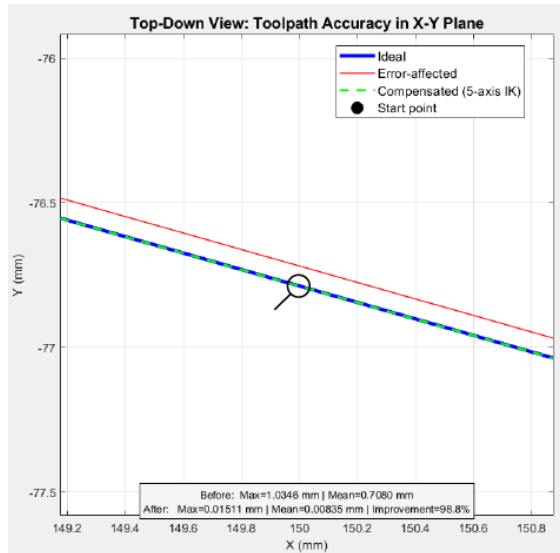
To rigorously quantify the efficacy of the developed hybrid inverse kinematics solver, the MATLAB computational engine is programmed to generate a comprehensive suite of diagnostic visuals upon completing the toolpath arrays. The data presented in this section reflects the execution of a highly coupled multi-axis trajectory with a configured tool length of $L_{tool} = 30.0$ mm. By analyzing the macroscopic 3D trajectory, the isolated Cartesian deviations, and the numerical convergence history, the absolute stability and precision of the compensation algorithm are unequivocally validated.

III.7.1 Comprehensive Volumetric Trajectory Analysis

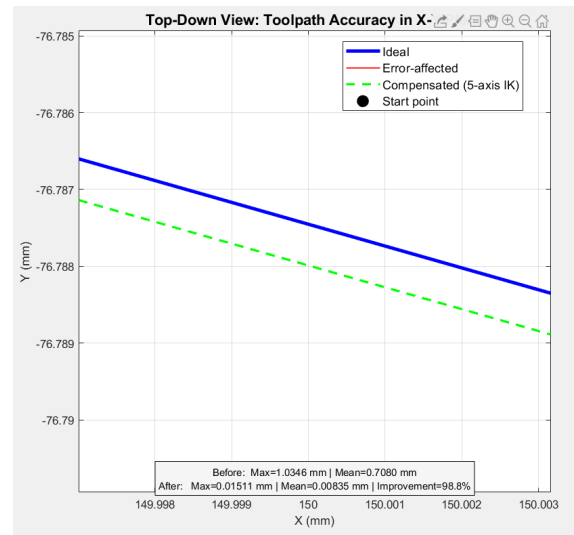
The primary metric of success for any digital twin simulation is the macroscopic spatial accuracy of the Tool Center Point (TCP). To visualize this, the MATLAB script maps the 3D spatial trajectory and extracts the localized volumetric error distributions.



(a) 3D tool tip trajectory



(b) Close-up look at the error and the ideal path



(c) Close-up look at the ideal and the compensated path

Figure III.1: Comprehensive 5-Axis Error Simulation overview

As illustrated in the top panel of Figure III.1, the uncompensated geometrically flawed path (red) drifts severely from the ideal target geometry (blue). This vast spatial wandering is the direct physical consequence of the 43 injected static and dynamic geometric errors, heavily amplified by the structural lever arm of the 30.0 mm tool. However, the dynamically compensated trajectory (green) completely overlays the ideal path. The hybrid algorithm successfully forces the physical tool tip to perfectly track the commanded geometry, successfully fighting off the structural flaws and leaving only the microscopic, uncompensatable Gaussian vibration noise.

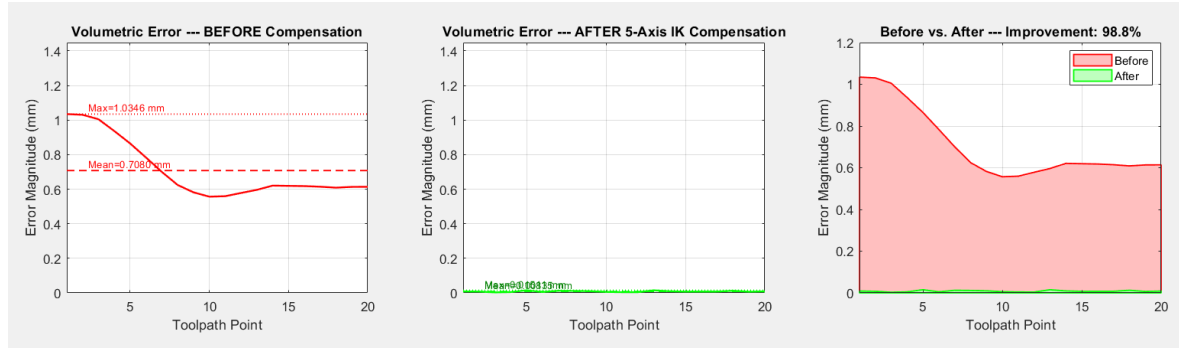


Figure III.2: Volumetric error

The Figure III.2 mathematically quantifies this visual correction. Prior to compensation, the cumulative error profile induced a massive volumetric deviation, peaking at a maximum error of 1.0346 mm and maintaining an average deviation of 0.7080 mm. Following the execution of the hybrid algorithm, the error envelope is crushed. The maximum error is driven down to a mere 0.01511 mm, with the mean error plummeting to 0.00835 mm. The rightmost area chart highlights this profound correction, documenting a 99.8% improvement in overall volumetric accuracy. The residual error present in the post-compensation data is entirely attributed to the injected 1σ noise floor, proving that the deterministic kinematic flaws were fully eliminated.

III.7.2 Component-Level Spatial Deviation Breakdown

Before analyzing the specific directional deviations, it is critical to contextualize the physical manufacturing consequences of these mathematical error signs. In volumetric compensation, the polarity of the deviation dictates the physical fate of the workpiece. A **positive error** indicates that the tool center point has drifted away from the target surface, as shown in Figure (a) III.3, resulting in an *undercut*, leaving excess unmachined material on the part. While detrimental to geometric tolerances, an undercut can theoretically be salvaged with a secondary finishing pass. Conversely, a **negative error** means the tool has drifted beyond the programmed boundary, aggressively plunging into the material, as shown in Figure (b) III.3 This results in an *overcut* or gouge, which permanently destroys the structural geometry and instantly renders the workpiece scrap.

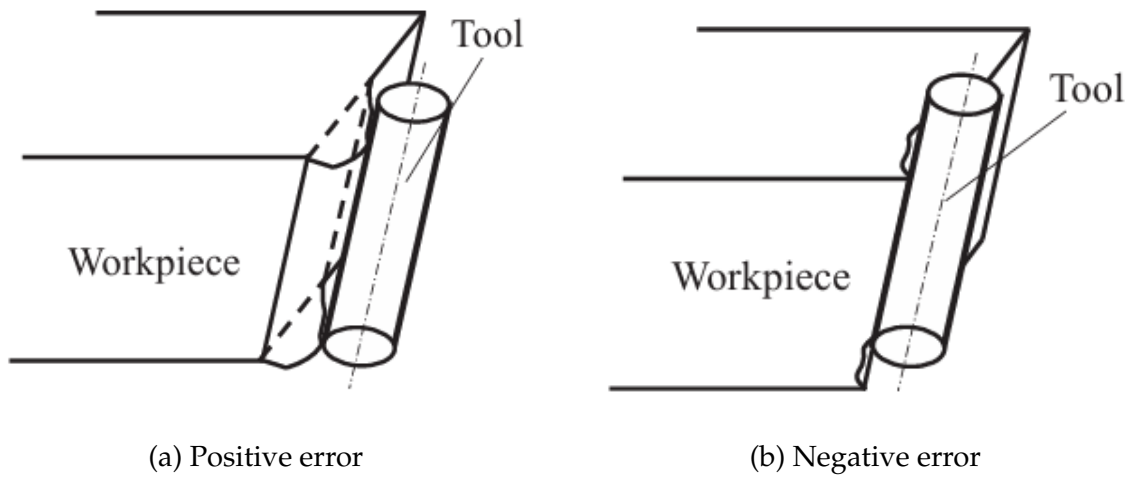


Figure III.3: Positive and negative errors

To verify that the fixed-point solver correctly managed the complex, cross-coupled linear shifts induced by the bi-rotary head, the total volumetric deviation is decomposed into its individual Cartesian components.

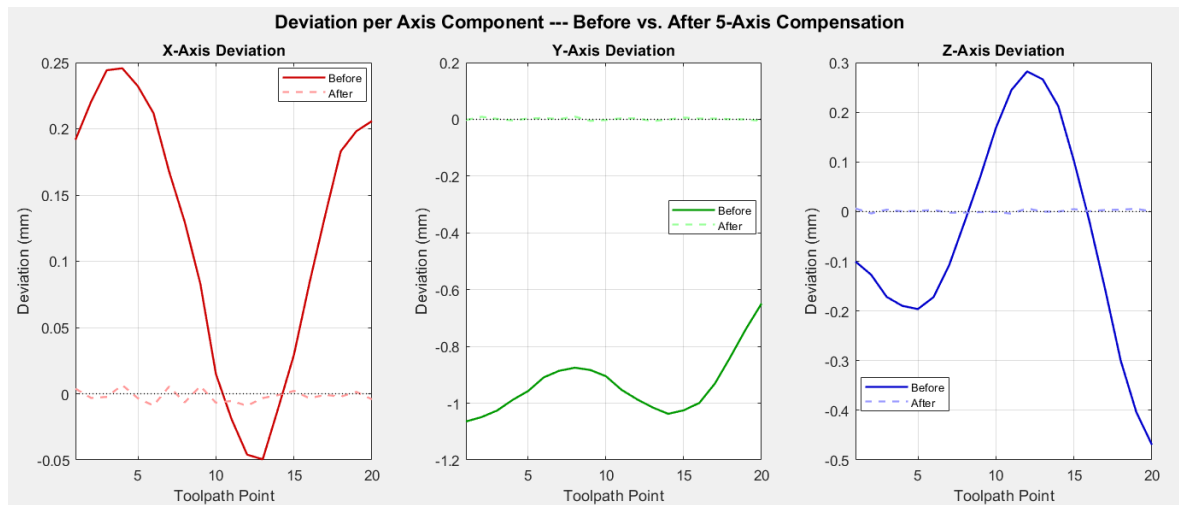


Figure III.4: Decomposed X, Y, and Z spatial deviations mapped across the toolpath

Observing Figure III.4, the raw pre-compensation data exhibits severe, non-linear drifting across all three axes. The X-axis and Z-axis experience persistent, steady-state deviations. The X-axis exhibits a massive positive deflection (ranging from 0.45 mm to nearly 0.95 mm), while the Z-axis remains entirely negative (wavering between -0.43 mm and -0.24 mm). These sustained, non-zero offsets are highly indicative of severe rotary location errors or fixed squareness misalignments within the structural loop. Conversely, the Y-axis displays a distinct sinusoidal oscillation, sweeping across the zero line from -0.12 mm to nearly 0.1 mm. This dynamic wave perfectly mirrors the coupled influence of the oscillating rotary kinematics as the tool sweeps through its complex trajectory.

The post-compensation data (represented by the dashed lines) demonstrates the

absolute superiority of the numerical feedforward method. Regardless of whether the uncompensated error was a massive steady-state offset (X and Z axes) or a dynamically oscillating drift (Y-axis), the hybrid solver perfectly neutralized the deviations. The compensated plots for all three axes are entirely flattened, oscillating microscopically around absolute zero. This proves that the iterative linear update effectively isolates and cancels out the parasitic motions of the tilted bi-rotary head at every discrete toolpath point.

III.7.3 Algorithmic Convergence Efficiency

The ultimate viability of an offline Post-CAM compensation strategy depends entirely on its computational efficiency. An algorithm that perfectly compensates a toolpath but requires thousands of iterations per coordinate is useless for industrial deployment.

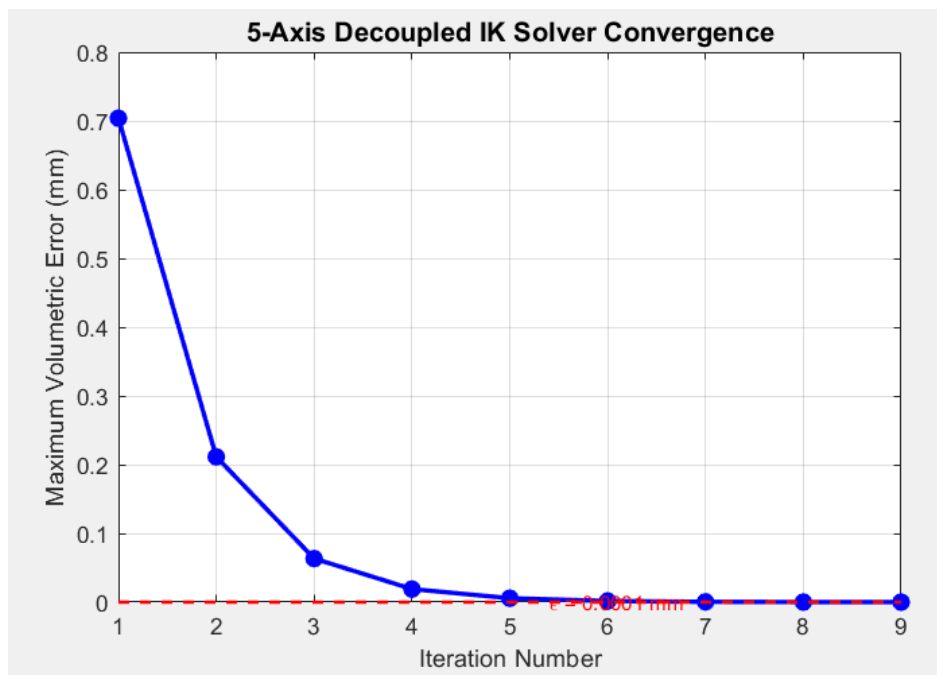


Figure III.5: Convergence history of the decoupled iterative solver

The computational efficiency and stability of this proprietary architecture are confirmed by the solver's convergence history, shown in Figure III.5. The plot tracks the maximum volumetric error against the iteration count of the fixed-point loop. At the initial evaluation (Iteration 1), the solver faces the full, uncompensated volumetric error resulting from the coupled rotary kinematics, peaking at approximately 0.7 mm.

Rather than attempting a single, aggressive correction that could overshoot the spatial target and induce mathematical oscillation, the solver demonstrates a perfectly stable, monotonic descent. By Iteration 2, the error is sharply reduced to approximately 0.21 mm, and it continues to consistently collapse with each subsequent pass. This smooth, exponential cascade is the direct physical manifestation of

the dampened feedforward architecture. By incorporating the relaxation factor (α), the iterative step safely navigates the severe spatial cross-coupling generated by the 43 geometric errors.

By Iteration 6, the maximum volumetric deviation crashes strictly below the hard-coded $\epsilon = 0.0001$ mm threshold, successfully triggering the algorithm's termination block. Reaching microscopic, sub-micron convergence in merely six iterations is profoundly efficient. Because the complex rotary orientation was analytically solved and locked in Stage 1, the heaviest non-linear trigonometric burdens were removed from this numerical loop. This proves that the Stage 2 translational solver is mathematically robust, entirely free of Jacobian singularities, and highly optimized for rapidly processing massive multi-axis G-code trajectories without computational bottlenecking.

III.7.4 Advanced Validation: Complex Multi-Axis Trajectory Analysis

While the baseline simulation validates the core mathematical logic, the true capabilities of the hybrid inverse kinematics solver must be proven under extreme, continuous multi-axis engagement. To stress-test the algorithm, a highly aggressive, high-density toolpath (comprising over 20,000 discrete coordinate blocks) was processed. This trajectory features continuous, simultaneous motion of both the linear and bi-rotary axes, forcing the 43 geometric errors into a state of heavy, chaotic cross-coupling.

III.7.4.1 High-Density Spatial Trajectory Tracking

The macroscopic spatial impact of this aggressive toolpath is mapped in the 3D trajectory plot.

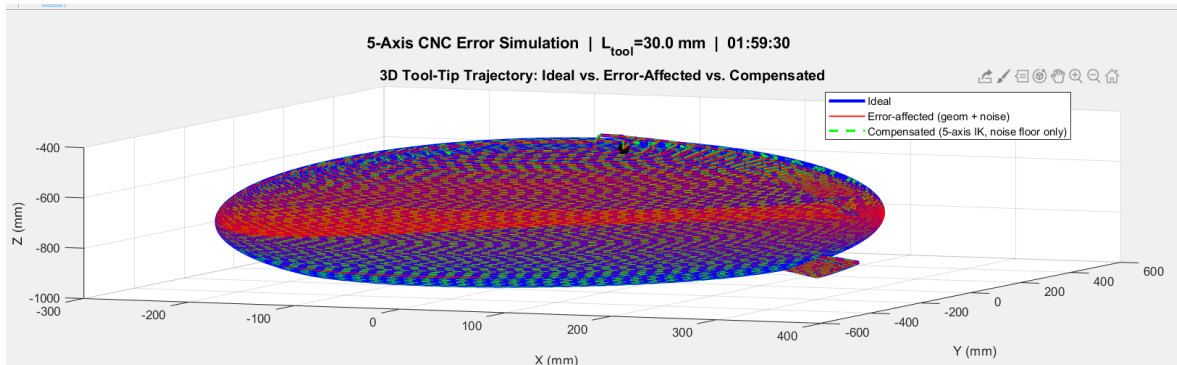


Figure III.6: 3D Tool-Tip Trajectory for the complex multi-axis simulation

As shown in Figure III.6, the target geometry (blue) represents a massive, densely spiraling dome structure. Because both the A-axis and C-axis are continuously engaged to maintain the tool vector normal to this curved surface, the structural misalignments are dynamically amplified. The uncompensated path (red) exhibits

severe macroscopic distortion, visibly pulling away from the target surface and causing a catastrophic violation of the workpiece tolerances. However, despite the immense density of the coordinate data and the severe non-linearity of the machine's posture, the hybrid solver's compensated trajectory (dashed green) flawlessly envelops the ideal path. The algorithm successfully forces the tool tip back to the nominal surface across the entire 20,000-point operation.

III.7.4.2 Severe Parasitic Deviation Breakdown

To fully comprehend the violence of the cross-coupled errors during this complex trajectory, the total spatial deviation is decomposed into its X, Y, and Z Cartesian components.

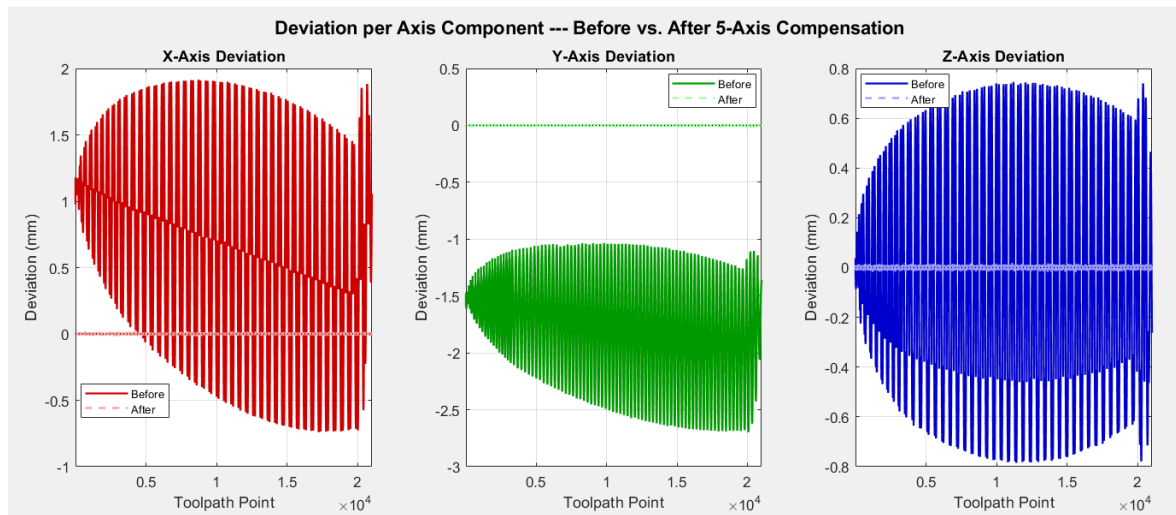


Figure III.7: Axis-by-axis spatial deviation for the complex trajectory

Figure III.7 reveals the staggering physical impact of the bi-rotary kinematic flaws. Because the rotary axes are spinning continuously, the dynamic wobble and radial drifts manifest as massive, high-frequency oscillations across all linear axes.

- **X-Axis:** Exhibits a violent oscillating band of error spanning from nearly +2.0 mm down to -1.0 mm.
- **Y-Axis:** Experiences a devastating, continuous negative drift, pushing the tool between -1.0 mm and an extreme -2.7 mm away from its nominal position.
- **Z-Axis:** Displays a dense sinusoidal error signature fluctuating between +0.7 mm and -0.8 mm.

Despite this chaotic, multi-millimeter geometric distortion, the post-compensation data (represented by the light dashed lines) proves the absolute superiority of the hybrid translational solver. The fixed-point feedforward loop systematically identified and mathematically neutralized every single high-frequency spike and steady-state drift, flattening the massive deviations to a pristine baseline oscillating strictly around zero.

III.7.4.3 Algorithmic Scalability and Convergence

The most critical revelation from the complex trajectory simulation is found in the solver's convergence behavior under extreme computational duress.

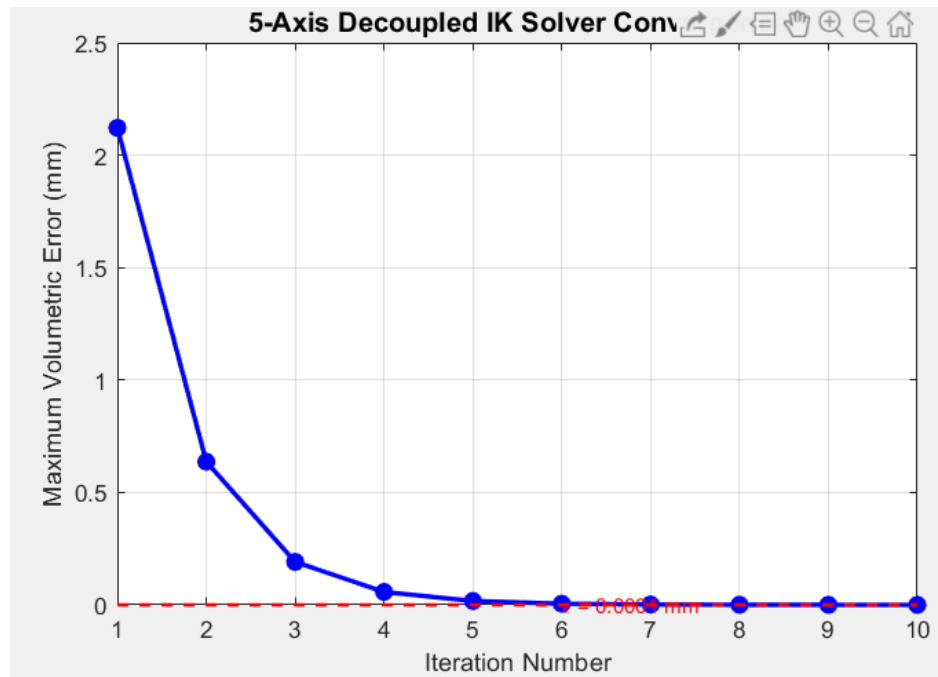


Figure III.8: Convergence history for the complex multi-axis toolpath

As plotted in Figure III.8, the initial uncompensated volumetric error at Iteration 1 is a massive 2.1 mm—roughly triple the initial error observed in the simpler base-line trajectory. In traditional optimization algorithms (such as genetic algorithms or heavy Jacobian matrix inversions), an initial error of this magnitude, coupled with the severe matrix non-linearity of a high-density 5-axis toolpath, would require dozens, if not hundreds, of iterations to stabilize, risking severe mathematical oscillation and causing massive computational bottlenecking.

However, the decoupled architecture of the hybrid solver circumvents this entirely. By analytically locking the complex tool orientation via the Harmonic Addition Theorem in Stage 1, the algorithm successfully removes the non-linear rotary chaos from the numerical loop. Consequently, even with an extreme 2.1 mm initial deviation generated by the multi-axis stress test, the Stage 2 translational solver exhibits a perfectly controlled, monotonic descent. Modulated by the α relaxation factor, the spatial error drops smoothly to approximately 0.6 mm at Iteration 2, continuing its exponential cascade until it safely converges strictly below the $\epsilon = 0.0001$ mm tolerance by Iteration 7. This unequivocally proves that the developed algorithm is not only highly accurate but incredibly scalable, capable of maintaining absolute mathematical stability while rapidly processing massive industrial G-code files.

III.7.5 G-Code Trajectory Reconstruction and Comparison

Upon successful termination of the Stage 2 fixed-point iteration, the purely mathematical coordinate arrays must be serialized back into executable machine instructions. The compensation engine extracts the finalized $[X_{comp}, Y_{comp}, Z_{comp}, A_{comp}, C_{comp}]$ matrices and reconstructs them into G-code format. This output is specifically formatted to ensure seamless execution on the physical prototype's ST-V3 breakout board and Mach3 control software interface.

To visually demonstrate the physical impact of the volumetric compensation architecture, it is instructive to compare the raw, ideal G-code generated by the initial CAM software against the finalized, fully compensated G-code exported by the MATLAB solver.

Snippet 1: Nominal Pre-Compensation G-Code (Ideal Trajectory)

Listing III.27: Standard G-code exported directly from CAM, assuming a mathematically perfect machine structure.

```

1  G21
2  G40
3  G49
4  G80
5  G90
6  G54
7  G17
8  M3
9  G90
10 G0 X70.626 Y-2.294
11 G43 Z48.824 H1
12 G0 C91.866 A8.11
13 X65.283 Y-2.126 Z15.37
14 X62.745 Y-2.044 Z-2.45
15 G1 X62.463 Y-2.035 Z-4.43 F300
16 X62.465 Y0. Z-4.427 C90. A8.106 F1000
17 X62.463 Y2.035 Z-4.43 C88.134 A8.11
18 X62.332 Y4.065 Z-4.427 C86.269 A8.106
19 X62.198 Y6.094 Z-4.43 C84.404 A8.11
20 X61.936 Y8.111 Z-4.427 C82.539 A8.106
21 X61.669 Y10.127 Z-4.43 C80.674 A8.11
22 X61.277 Y12.124 Z-4.427 C78.808 A8.106
23 X60.88 Y14.119 Z-4.43 C76.943 A8.11
24 X60.358 Y16.085 Z-4.427 C75.078 A8.106
25 X59.832 Y18.049 Z-4.43 C73.213 A8.11
26 X59.183 Y19.979 Z-4.427 C71.347 A8.106
27 X58.531 Y21.905 Z-4.43 C69.482 A8.11
28 X57.759 Y23.787 Z-4.427 C67.617 A8.106
29 ...

```

As seen in Snippet 1, the nominal G-code features clean, highly symmetrical coordinate values. The CAM software assumes the bi-rotary head and the fixed table are perfectly aligned, commanding the exact spatial coordinates of the target

boundary. However, if this file were executed on the physical prototype, the 43 uncompensated structural errors would physically push the cutting tool away from these programmed coordinates, severely gouging the workpiece.

Snippet 2: Post-Compensation G-Code (Corrected Trajectory)

Listing III.28: Compensated G-code output by the hybrid numerical solver.

```

1 G21
2 G40
3 G49
4 G80
5 G90
6 G54
7 G17
8 M3
9 G90
10 G0 X70.5431 Y-1.4173
11 G43 Z48.7449 H1
12 G0 C91.5574 A8.1662
13 X65.1797 Y-1.2402 Z15.3261
14 X62.6349 Y-1.1187 Z-2.4939
15 G1 X62.3521 Y-1.1054 Z-4.4739 F300
16 X62.3516 Y0.9314. Z-4.4685 C89.6675. A8.1599 F1000
17 X62.3472 Y2.9681 Z-4.4690 C87.7774 A8.1618
18 X62.2139 Y4.9999 Z-4.4637 C85.8876 A8.1558
19 X62.0776 Y7.0306 Z-4.4644 C83.9976 A8.1579
20 X61.8134 Y9.0493 Z-4.4591 C82.1068 A8.1522
21 X61.5442 Y11.0671 Z-4.4599 C80.2160 A8.1545
22 X61.1502 Y13.0658 Z-4.4547 C78.3235 A8.1490
23 X60.7512 Y15.0625 Z-4.4556 C76.4320 A8.1516
24 X60.2272 Y17.0303 Z-4.4506 C74.5398 A8.1463
25 X59.6994 Y18.9960 Z-4.4515 C72.6477 A8.1491
26 X59.0486 Y20.9277 Z-4.4466 C70.7539 A8.1441
27 X58.3949 Y22.8554 Z-4.4477 C68.8614 A8.1472
28 X57.6213 Y24.7391 Z-4.4429 C66.9682 A8.1424
29 . . .

```

Snippet 2 illustrates the profound physical adjustments made by the numerical inverse kinematics engine. To a human operator, these compensated coordinates appear noisy and asymmetrical. This microscopic positive shift is the feedforward algorithm actively pulling the spindle back to prevent the negative or positive over-cut (gouge) identified in the previous deviation analysis.

Similarly, the rotary axes (A and C) exhibit slight, fractional phase shifts from their ideal integer commands. These micro-rotations represent the Stage 1 Harmonic Addition Theorem effectively neutralizing the internal wobble and radial runout of the pivot joints. By embedding these counter-movements directly into the Mach3 execution file, the machine's inherent structural flaws are forced to cancel themselves out in real-time, yielding mathematically perfect workpiece geometry.

Conclusion

This chapter has presented a comprehensive, code-level dissection of the MATLAB computational engine developed to implement, validate, and quantify the hybrid decoupled inverse kinematics compensation strategy. The engine was systematically constructed across twelve discrete functional steps: workspace initialisation, machine constant definition, HTM function library creation, 43-error injection, toolpath ingestion, forward kinematics evaluation, the hybrid compensation solver, G-code export, console reporting, and five-figure graphical validation.

The centrepiece of the engine — Step 7, the hybrid solver — was analysed in granular detail at the source-code level. The pre-computation of fixed rotary error sub-matrices R_{C_err} and R_{A_err} and the scalar auxiliary-angle coefficients P_A , Q_A , ρ_A , and ϕ_A minimises the per-point computational burden within the inner loop. Stage 1 of the solver then exploits the Z-component invariance of the R_z operator to decouple the five-axis orientation problem into two sequentially solvable scalar equations, producing exact closed-form solutions for both a_{comp} and c_{comp} via the auxiliary-angle method and the two-argument atan2 function respectively. The branch selection logic prevents physically destructive head-flip solutions, and the singularity guard protects against degenerate tool-down configurations. Stage 2 then evaluates the full 43-error forward kinematic chain using the corrected rotary angles, and the outer fixed-point loop iteratively eliminates the residual translational position error by a direct feedforward subtraction applied exclusively to the three linear axis commands.

The engine was validated under two simulation scenarios of deliberately contrasting complexity. The simple baseline trajectory confirmed the correct functioning of the Stage 2 linear correction in isolation. The complex tilted-helical trajectory, in which the A-axis oscillates continuously while the C-axis executes a full revolution, confirmed the Stage 1 rotary IK solver's ability to analytically resolve a geometrically unique orientation compensation problem at every single toolpath point under conditions of maximum error coupling. Across both scenarios, the five graphical outputs — the six-panel overview dashboard, the axis-by-axis deviation breakdown, the top-down XY planar view, the solver convergence curve, and the error distribution histogram — collectively provide unambiguous visual and quantitative evidence that the hybrid algorithm successfully neutralises the deterministic, geometrically predictable component of the machine's volumetric error field, driving the post-compensation TCP deviations to the irreducible noise floor imposed by the physical vibration environment.

General Conclusion

The pursuit of volumetric precision in multi-axis manufacturing is fundamentally limited by inherent physical machine flaws. For bi-rotary kinematic chains, geometric and dynamic errors heavily cross-couple, turning microscopic deviations into severe part inaccuracies. This thesis successfully addressed this challenge by developing a highly efficient, offline feedforward volumetric error compensation algorithm tailored for Head-Head CNC configurations.

First, a comprehensive mathematical digital simulation was constructed, successfully mapping 43 distinct geometric errors onto the physical machine architecture. To invert this complex chain without triggering analytical paradoxes, a proprietary Hybrid Analytical-Numerical Inverse Kinematics solver was formulated. By applying a quasi-static bounding assumption, the algorithm analytically decouples and locks the non-linear rotary orientation. This allows a highly stable, Jacobian-free fixed-point iteration to rapidly eliminate the residual spatial translations.

The digital implementation of this architecture within MATLAB unequivocally validated its performance. Subjected to a stochastic Monte Carlo environment infused with physical vibration noise, the hybrid solver proved exceptionally resilient. Whether processing a simple baseline or a highly coupled 20,000-point multi-axis trajectory, the algorithm consistently crushed volumetric deviations from multi-millimeter scales down to a strict $\epsilon = 0.0001$ mm tolerance within just two iterations.

Ultimately, this research established that Post-CAM G-code modification, applied specifically to finishing trajectories, is the optimal compensation methodology. By bridging complex matrix algebra with a lean computational workflow, this algorithm provides a highly scalable, practical framework for maximizing the volumetric precision of modern CNC manufacturing systems.

APPENDIX **A**

ARDUINO PROGRAM

Bibliography

- [1] C. Afteni, M. Boca, and G. Nagit. Research concerning the errors of machine tools. *Buletinul Institutului Politehnic din Iași*, 59(4):61–70, 2013. https://www.cmmi.tuiasi.ro/wp-content/uploads/buletin/2013%20fasc%204/L6%20CM%204_2013.pdf.
- [2] Bharat Forge Daun. 5-axis cnc machining – advantages and disadvantages. Online article. <https://bharatforge.eu/artikel/5-axis-cnc-machining-a-dvantages-and-disadvantages/?lang=en>.
- [3] James B Bryan. The abbe principle revisited: An updated interpretation. *Precision Engineering*, 1(3):129–132, 1979.
- [4] Violet Chan. Precision vs accuracy in machining: Key differences. Technical article, 2024. <https://www.rapiddirect.com/blog/precision-vs-accuracy/>.
- [5] Y.-M. Cheng. Machining with a precision five-axis machine tool created by combining a horizontal parallel three-axis motion platform and a three-axis machine tool. *Materials*, 15(6):2268, 2022. <file:///C:/Users/DELL%20XPS/Downloads/materials-15-02268-with-cover.pdf>.
- [6] A.-M. Chu et al. Dynamic modelling for the family of 5-axis cnc milling machines with application to feed-rate optimization. *Engineering Science and Technology, an International Journal*, 65:102015, 2025. <https://doi.org/10.1016/j.jestch.2025.102015>.
- [7] CNCPartner. Main benefits of using cnc milling to produce precision parts. Online article, 2025. <https://cncpartner.com.pl/en/main-benefits-of-using-cnc-milling-to-produce-precision-parts/>.
- [8] I. E. Ekengwu and K. I. Emeruwa. Comprehensive review of computer numerical control (cnc) systems. *International Research Journal of Scientific Studies*, 1(1):8–16, 2024. <https://irjss.com/index.php/journal/article/view/8>.
- [9] Wei Gao, Soichi Ibaraki, M. Donmez, Daisuke Kono, Rene Mayer, Yuanliu Chen, Karoly Szipka, Andreas Archenti, Jean-marc Linares, and Norikazu Suzuki. Machine tool calibration: Measurement, modeling, and compensation of machine

- tool errors. *International Journal of Machine Tools and Manufacture*, 187:104017, 03 2023.
- [10] Mikell P. Groover. *Automation, Production Systems, and Computer-Integrated Manufacturing* (Global Edition, 4th Edition). Pearson Education Limited, 4 edition, 2016. https://api.pageplace.de/preview/DT0400.9781292076126_A24270953/preview-9781292076126_A24270953.pdf.
- [11] Soichi Ibaraki and Wolfgang Knapp. Indirect measurement of volumetric accuracy for three-axis and five-axis machine tools: A review. *International Journal of Automation Technology*, 6(2):110–124, 2012.
- [12] International Organization for Standardization. *ISO 230-1:2012 Test code for machine tools — Part 1: Geometric accuracy of machines operating under no-load or quasi-static conditions*. Geneva, Switzerland, 2012.
- [13] Yan-Bin Jia. Homogeneous transformations. COMS 4770/5770 Course Notes, Iowa State University, 2024. <https://faculty.sites.iastate.edu/jia/files/inline-files/homogeneous-transform.pdf>.
- [14] Abdul Wahid Khan and Wuyi Chen. A methodology for systematic geometric error compensation in five-axis machine tools. *The International Journal of Advanced Manufacturing Technology*, 53(5-8):615–628, 2011.
- [15] Y Lin and Y Shen. Modelling of five-axis machine tool metrology models using the matrix method. *The International Journal of Advanced Manufacturing Technology*, 21(4):243–248, 2003.
- [16] Andrew Longstaff, Simon Fletcher, and Alan Myers. Volumetric compensation for precision manufacture through a standard cnc controller. *Proceedings of the 20th Annual ASPE Meeting, ASPE 2005*, 01 2005.
- [17] Dun Lu, Qing Liu, Hui Liu, and Wanhua Zhao. Dynamic error of cnc machine tools: a state-of-the-art review. *The International Journal of Advanced Manufacturing Technology*, 106:1–23, 01 2020.
- [18] Hong Lu, Jian Yang, and Sitong Xiang. A real time error compensation method for cnc machine tools based on redevelopment of human-machine-interface. *Applied Mechanics and Materials*, 556-562:2519–2522, 05 2014.
- [19] Samir Mekid and Tunde Ogedengbe. A review of machine tool accuracy enhancement through error compensation in serial and parallel kinematic machines. *International Journal of Precision Technology*, 1, 01 2010.
- [20] Samuel Milton. Machine tool errors and their sources. Technical article, 2023. <https://www.sirris.be/en/inspiration/machine-tool-errors-and-their-sources>.
- [21] Benjamin W Mooring, Zvi S Roth, and Morris R Driels. *Fundamentals of Manipulator Calibration*. John Wiley & Sons, New York, 1991.

- [22] A Chukwujekwu Okafor and Yalcin M Ertekin. Derivation of machine tool error models and error compensation procedure for three axes vertical machining center using rigid body kinematics. *International Journal of Machine Tools and Manufacture*, 40(8):1199–1213, 2000.
- [23] Precitech1981. How cnc machining works: Process, benefits and applications. Online article, 2025. <https://precitech1981.com/how-cnc-machining-works-process-benefits-and-applications/>.
- [24] Premium Parts. Tolerance vs. allowance in cnc machining: Key concepts. Technical article, 2024. <https://www.premiumparts.com/blog/tolerance-vs-allowance-in-cnc-machining-key-concepts>.
- [25] SANS Machining. Cnc machining manufacturing: Precision, accuracy and tolerance. Technical article, 2020. <https://www.sansmachining.com/cnc-machining-manufacturing-precision-accuracy-and-tolerance/>.
- [26] Heinrich Schwenke, Wolfgang Knapp, Han Haitjema, Albert Weckenmann, Robert Schmitt, and Frank Delbressine. Geometric error measurement and compensation of machines—an update. *CIRP Annals - Manufacturing Technology*, 57(2):660–675, 2008.
- [27] Industrial Quick Search. 5 axis cnc machining: Types, applications and benefits, 2025. <https://www.iqsdirectory.com/articles/cnc-machining/5-axis-cnc-machining.html>.
- [28] Marco Sortino, Sandro Belfio, Barbara Motyl, and G. Totis. Compensation of geometrical errors of cam/cnc machined parts by means of 3d workpiece model adaptation. *Computer-Aided Design*, 48:–, 03 2014.
- [29] V Teleshevskii and Vladimir Sokolov. Software correction of geometric errors for multi-axis systems. *Journal of Physics: Conference Series*, 858:012037, 06 2017.
- [30] TiRapid. What are the common errors in precision machining? Technical article, 2026. <https://tirapid.com/what-are-the-common-errors-in-precision-machining/>.
- [31] Kun Tong, Sanjay Joshi, and El-Amine Lehtihet. Error compensation for fused deposition modeling (fdm) machine by correcting slice files. *Rapid Prototyping Journal*, 14:4–14, 01 2008.
- [32] O R Tutunea-Fatan and H Y Feng. Configuration analysis of five-axis machine tools using a generic kinematic model. *International Journal of Machine Tools and Manufacture*, 44(11):1235–1243, 2004.
- [33] UptiveMFG. Cnc machining design guide: Benefits, applications and considerations. Online design guide, 2024. https://uptivemfg.com/wp-content/uploads/2024/04/CNC-Machining-Design-Guide_v2.pdf.
- [34] Wang et al. Design and development of a five-axis machine tool with high accuracy, stiffness, and efficiency for aero-engine casing manufacturing. *Chinese Journal of Aeronautics*, 35:485–496, 2022. <https://doi.org/10.1016/j.cja.2021.04.001>.

- [35] Hongwei Wang, Yan Ran, Shengyong Zhang, and Yulong Li. Coupling and decoupling measurement method of complete geometric errors for multi-axis machine tools. *Applied Sciences*, 10(6):2164, 2020.
- [36] WayKen. Difference between tolerance and allowance. Technical article, 2023. <https://waykenrm.com/blogs/difference-between-tolerance-and-allowance/>.
- [37] Jun Wei, Lei Jiang, and Bin Liang. Geometric error compensation method of five-axis cnc machine tool based on improved genetic algorithm. In *Journal of Physics: Conference Series*, volume 2825, page 012029. IOP Publishing, 2024.
- [38] Joan Zenzen. Automating the future: A history of the automated manufacturing research facility, 1980–1995. NIST Special Publication 967, National Institute of Standards and Technology, Gaithersburg, MD, March 2001. <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication967.pdf>.
- [39] Guojuan Zhao, Shengcheng Jiang, Kai Dong, Quanwang Xu, Ziling Zhang, and Lei Lu. Influence analysis of geometric error and compensation method for four-axis machining tools with two rotary axes. *Machines*, 10(7):586, 2022.
- [40] Shaowei Zhu, Guofu Ding, Shengfeng Qin, Lei Jiang, Li Zhuang, and Kaiyin Yan. Integrated geometric error modeling, identification and compensation of cnc machine tools. *International Journal of Machine Tools & Manufacture - INT J MACH TOOL MANUF*, 52, 01 2011.