UNIVERSITÀ DEGLI
STUDI DI PERUGIA

BACHELOR'S DEGREE IN ENGINEERING MANAGEMENT

The Significance of Quantitative Analysis and Monitoring Capability Indices in Process Quality Standardization and Control

Thesis Author: Simone Massinelli
Thesis Advisors: Gianluca Rossi, Alessio Troiani

Table of Contents

1 Introduction

1.1 My Internship Experience and the Company's Request

From July to October 2024, I completed an internship at Stanley Black & Decker DeWalt in Corciano, where I was part of the Quality team. During this period, I had the opportunity to learn and contribute to a variety of activities, primarily focused on maintaining and improving quality standards.

These activities included managing incoming goods, overseeing components and finished products, developing control plans, and supporting production processes.

I was also involved in verification and validation processes, performing tests and analyses to ensure compliance with established quality requirements. Additionally, I had the chance to gain valuable insights into the Toyota production methodology, which emphasizes continuous improvement and waste reduction, both principles that are crucial for sustaining high-quality standards in every phase of production.

Towards the end of my internship, I supported the team in the standardization of a gearbox, which will be discussed in detail later in this thesis. This component is produced at the Perugia plant and subsequently shipped to the Usti plant in the Czech Republic, where it is utilized in the production lines.

Due to this inter-plant collaboration, standardization became a mandatory requirement. Although both facilities belong to the same multinational corporation, each operates with its own financial accounts, functioning almost as independent entities. To prevent potential operational issues downstream, the standardization of this component was deemed essential.

During this specific standardization process, the company's primary request was for me to perform a statistical analysis of several parameters, with particular attention to critical ones (QC, "Quote Critiche" in Italian, "Critical Dimensions" in English). This was my main activity, although I was also tasked with conducting a Gage R&R study, managing sampling activities, and contributing to the overall quality control framework.

One of the key statistical parameters to monitor in order to ensure the capability of the machining equipment, particularly for the QC, is the Cpk index, which measures how well a process can produce output within specified limits by considering both the process mean and variability. A detailed analysis of the Cpk index will be provided later.

Currently, the company mandates this index to be calculated only during the standardization of a component. Afterward, sampling methods are applied according to the specific component's control plan, but the Cpk is no longer calculated over time.

The reason behind this approach is that the process begins with an assessment of the maximum number of cycles to determine the equipment's lifespan. For example, in the case of the gearbox molds, they are replaced once the maximum cycle limit is reached (100,000 cycles). However, during the equipment's lifespan, maintenance interventions are carried out if it is noticed that the mold is not performing optimally.

Thus, the current methodology is to calculate the Cpk at the start to verify machine capability, replace the machine or tool once its estimated life cycles are complete, and perform regular checks according to the control plan in the meantime. Calculating this index for the twelve QC of the gearbox, using Minitab software, was one of my main tasks during the standardization of this component.

This task played a vital role in the standardization process and set the foundation for the analyses presented in this work.

1.2 My Personal Goal

As previoulsy mentioned, during my internship, I was introduced to the company's standardization process and learned that the Cpk index is calculated only at the start of a component's production to assess the capability of the machining equipments.

While this approach is effective for the initial validation, it led me to consider whether calculating the index periodically during the equipment's lifecycle could provide additional benefits. I believed that a more frequent evaluation of the Cpk index might help identify issues earlier and support more informated decision-making.

Some aspects of the current plan made me raise some questions. One of the observations I made during my internship was about the company's reliance on estimates for tools replacement. For instance, tools replacements are based on the expected lifespan. However, this estimate does not account for variability and some tools might fail sooner, while others could last longer. By tracking the evolution of the Cpk index, it might be possible to optimize the timing of tools replacements, reducing waste and minimizing the risk of unexpected failures. This is just one of the potential advantages of implementing such a method, and I will analyze all the opportunities in detail later in the thesis.

I believed this approach could be especially valuable for a multinational corporation. Decisions based on assumptions, rather than accurate data, could lead to inefficiencies, while tracking the Cpk index over time could provide a clearer picture of tool performance, helping the company align its decisions with actual production conditions.

Motivated by this idea, I decided to go beyond the company's immediate requirements. After completing my internship, I began exploring methods to analyze how the Cpk index evolves as production progresses. To do so, I consulted academic research, textbooks, and professors, aiming to identify the best approach to conduct this study.

In fact, although the company did not request this specific analysis, I made it my goal to collect data and investigate patterns in the evolution of the Cpk index. Using data analysis methods, I aimed to uncover insights that could help improve production efficiency and support quality control.

This approach forms the foundation of the analyses presented in the second part of this thesis, where I discuss how I discovered patterns and the potential advantages for the company to implement a more dynamic approach, monitoring the Cpk index.

1.3 Objectives and Structure of the Thesis

This thesis is structured around a clear distinction between two main areas of work: the tasks carried out as part of the company's requirements during my internship and the research I independently pursued to explore new possibilities for process improvement. This distinction, already outlined in the preceding subsections 1.1 and 1.2, serves to clarify the dual nature of the work presented here.The first part focuses on the work done to satisfy the company's specific requests, while the second part focuses on my personal goal of developing an innovative approach to monitor the evolution of the Cpk index.

The first part of the thesis, developed in collaboration with my first advisor, Gianluca Rossi, examines the standardization process undertaken during my internship. It begins by explaining the broader context of standardization, including the role of ISO 9001 within the company and the importance and structure of the SPPAP document. Additionally, it discusses the sampling method for standardization and other methods used for the QC, and describes the steps involved in submitting and gaining approval for the standardization process.

This section also includes a detailed analysis of the component in question, focusing on the L-Shape Rotary Hammer and its gearbox.

Finally, the first part explains the tools and methods used during the standardization process. It describes the ZEISS Coordinate Measuring Machine (CMM) used for measurements and discusses the Gage R&R study conducted to assess the reliability of the measurement system.

The statistical tools applied in this phase, including the Cpk index and associated parameters, are also introduced, along with considerations regarding data normalization.

The second part of this thesis, developed with the support of my second advisor, Alessio Troiani, originates from the independent research activities undertaken after my internship. It begins by describing the methods used for data collection and the modifications introduced to the dataframe, aimed at enriching it with additional information useful for subsequent analyses.

A brief data cleaning phase followed, during which the dataset was checked to ensure sufficient accuracy and consistency. This preparatory step allowed for a more reliable analytical process.

The main focus of this research was then the development of analysis methods to better understand the behavior of the Cpk index over time. In particular, the work involved logically structuring the data, identifying trends and potential seasonal patterns, and monitoring the evolution of the Cpk index throughout the equipment's lifecycle. The goal was to move beyond isolated observations and reveal meaningful dynamics in the production process.

Furthermore, this section explores the practical benefits that such analyses could bring to production management. Specifically, it examines how dynamic monitoring could optimize tool replacement schedules, reduce material waste, and support decision-making with more targeted and timely information.

By integrating these analytical approaches into the traditional framework of quality control, this thesis seeks to offer a perspective that combines the consolidation of established methodologies with the introduction of innovative, data-driven practices for continuous improvement.

Part 1

The Standardization Process and a Practical Application within the Company

2 The Standardization Process

2.1 Role of ISO 9001 within the Company

ISO 9001 is an internationally recognized standard for quality management systems (QMS) that provides companies with a robust framework to enhance efficiency, improve product quality, and boost customers satisfaction. Its adoption is not mandatory but demonstrates a commitment to consistent processes, continuous improvement, and customer focus, which are all factors that are particularly valuable for global corporations like Stanley Black & Decker.

Although ISO 9001 is not a requirement for obtaining product certifications such as the CE mark, it significantly streamlines the processes needed to meet such standards. The CE mark certifies that a product meets EU safety, health, and environmental protection directives and is essential for market entry in the European Economic Area While ISO 9001 does not give the possibility to apply the CE mark directly, adherence to its quality management principles ensures that the production processes and documentation are compliant with the rigorous standards often required for certification. This makes the ISO 9001 framework a useful tool for achieving regulatory compliance and obtaining certifications that build consumer trust and facilitate market access.

Beyond regulatory benefits, ISO 9001 also enables companies to enhance their position as suppliers. In fact, compliance with its principles ensures that the components produced by a company meet the usually stringent quality expectations of external clients and partners. This certification is often a prerequisite for becoming an approved supplier to other organizations, as it guarantees a systematic and reliable approach to quality control throughout the production cycle.

Furthermore, ISO 9001 principles play a crucial role even within the company's internal supply chain. In the case of Stanley Black & Decker, the supplier-customer relationship often exists between plants of the same multinational corporation. For instance, as mentioned in the introduction, the standardized component analyzed in this thesis, is produced in the Corciano plant and suddenly shipped to the Usti plant in the Czech Republic for integration into final products.

Despite this internal relationship, the company still adheres to the standardized practices outlined in ISO 9001 to ensure consistency and reliability across its operations. To facilitate this, the corporation employs a document known as SPPAP (or simply PPAP), which will be discussed in the next subchapter of the thesis. The connection between this document and ISO 9001 is rooted in the IATF 16949:2016 regulation, strictly aligned with ISO 9001:2015 guidelines.

This structured approach highlights the strategic importance of ISO 9001 as a tool for achieving external certifications and as a foundation for maintaining high standards within a complex, multinational operational framework.

2.2 Description of the Document SPPAP

The Supplier Production Part Approval Process (SPPAP) originates from practices established within the automotive industry. Its primary purpose is to ensure that suppliers can consistently meet quality and manufacturing requirements for the parts and materials they provide. SPPAP is a critical methodology used to validate production processes, confirm that they are capable of producing components that meet specifications, and establish confidence in supplier capabilities.

At Stanley Black & Decker (SBD), SPPAP is used across all areas of the business for external suppliers of direct materials and subcontracted or special processes, like heat treatment or plating. It serves two main purposes: certifying suppliers who provide parts to SBD's manufacturing plants and certifying SBD itself as a qualified supplier for other customers.

There are two types of SPPAP utilized within SBD:

1. SPPAP Components: This applies to purchased parts or semi-finished products that require further processing within SBD plants. The goal is to ensure these parts meet SBD's manufacturing and quality requirements.
2. SPPAP Finished Goods (FG): This applies to finished products delivered directly to SBD or customers. It ensures that suppliers meet all manufacturing, testing, and quality requirements for the final goods.

SPPAP is applicable to both new and existing parts, especially when changes are introduced that require qualification. Such changes can include alterations to the component, the manufacturing process, or equipment, such as a mold change, that is also the reason behind the need of this specific standardization. In these cases, SPPAP helps verify that quality standards are maintained under the new conditions.

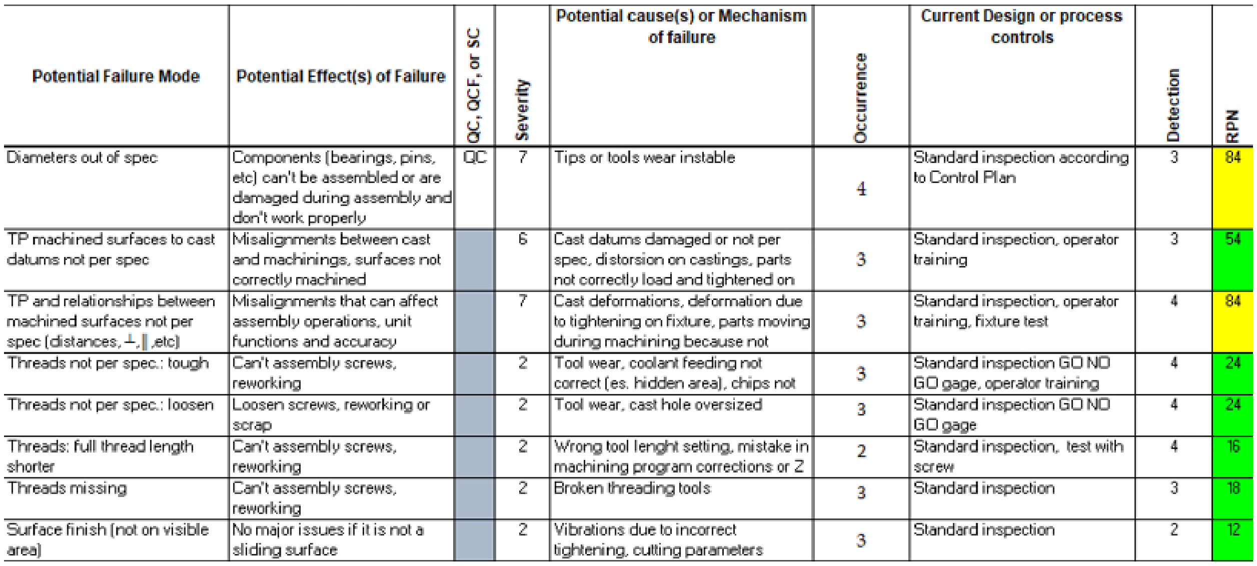
The SPPAP documentation ensures thorough analysis and validation of supplier processes. Key components include:

- Sampling Page: This section captures the outcomes of dimensional checks and test data for sampled components. Each parameter is reviewed and a code is given to indicate its status:
  - A (Approved): The parameter fully meets the required specifications.
  - B (Rejected): The parameter does not meet the required specifications and requires corrective action.
  - C (Conditional Approval): The parameter is acceptable under specific conditions but requires further monitoring or adjustments.
  - D (Recheck Required): The parameter needs to be re-evaluated to confirm its compliance.
  - E (Exception): The parameter deviates from standard specifications but is accepted due to special considerations or customer approval.

  For modifications like mold adjustments, sampling is mandatory for each mold, and separate records must be maintained for each mold too.
- Flow Diagram: Outlines the manufacturing steps, inspections, and transport activities for the component or product.
- PFMEA (Process Failure Modes and Effects Analysis): Evaluates potential risks in the manufacturing process by:
  - Identifying possible failure modes.
  - Assessing their impact through three parameters:
    - Severity: Indicates the seriousness of the failure's impact on functionality or safety.
    - Occurrence: Measures the likelihood of the failure happening.

∗ Detection: Evaluates the capability to detect the failure before it reaches production or the customer.

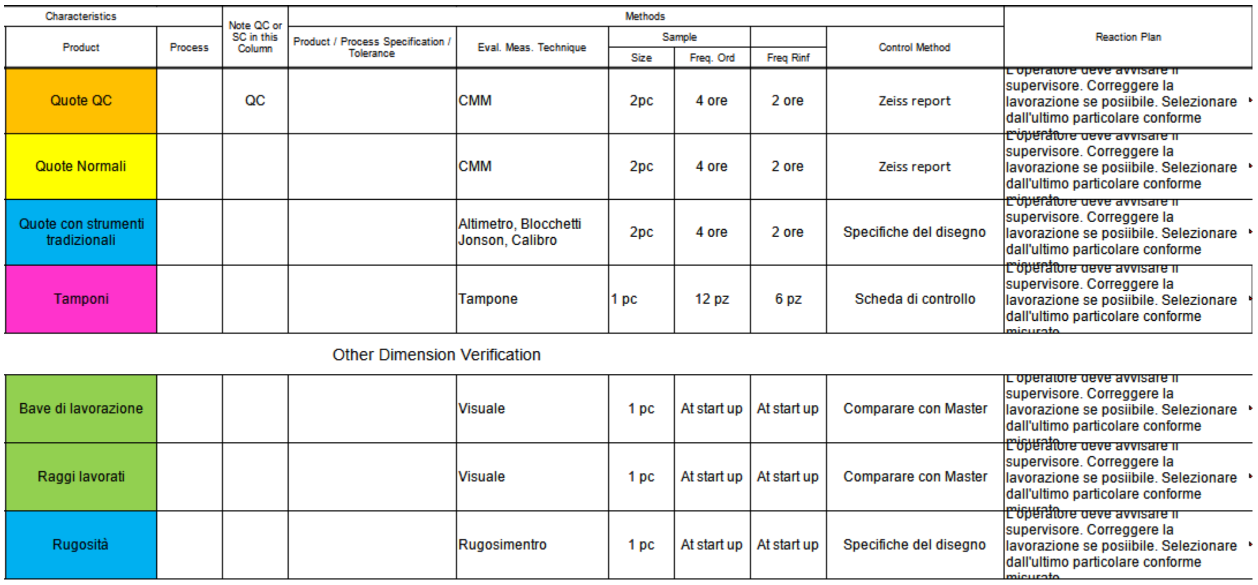
These factors are multiplied and contribute to calculating the Risk Priority Number (RPN), which prioritizes risks for corrective actions and has to be in a specific range to avoid production problems. If the risk factor is higher than a specific limit, the client can decide whether or not to accept the standardization.



Potential Failure Mode	Potential Effect(s) of Failure	QC, QCF, or SC	Severity	Potential cause(s) or Mechanism of failure	Occurrence	Current Design or process controls	Detection	RPN
Diameters out of spec	Components (bearings, pins, etc) can't be assembled or are damaged during assembly and don't work properly	QC	7	Tips or tools wear instable	4	Standard inspection according to Control Plan	3	84
TP machined surfaces to cast datums not per spec	Misalignments between cast and machinings, surfaces not correctly machined		6	Cast datums damaged or not per spec, distorsion on castings, parts not correctly load and tightened on	3	Standard inspection, operator training	3	54
TP and relationships between machined surfaces not per spec (distances, ⊥, ∥ ,etc)	Misalignments that can affect assembly operations, unit functions and accuracy		7	Cast deformations, deformation due to tightening on fixture, parts moving during machining because not	3	Standard inspection, operator training, fixture test	4	84
Threads not per spec.: tough	Can't assembly screws, reworking		2	Tool wear, coolant feeding not correct (es. hidden area), chips not	3	Standard inspection GO NO GO gage, operator training	4	24
Threads not per spec.: loosen	Loosen screws, reworking or scrap		2	Tool wear, cast hole oversized	3	Standard inspection GO NO GO gage	4	24
Threads: full thread length shorter	Can't assembly screws, reworking		2	Wrong tool lenght setting, mistake in machining program corrections or Z	2	Standard inspection, test with screw	4	16
Threads missing	Can't assembly screws, reworking		2	Broken threading tools	3	Standard inspection	3	18
Surface finish (not on visible area)	No major issues if it is not a sliding surface		2	Vibrations due to incorrect tightening, cutting parameters	3	Standard inspection	2	12

PFMEA of the Gearbox [1]

- Control Plan: Specifies how often parameters are inspected (e.g., once every four production hours) and the methods used for these inspections. It ensures consistency of parameters by setting clear schedules and techniques to control quality and react to non-conformities.



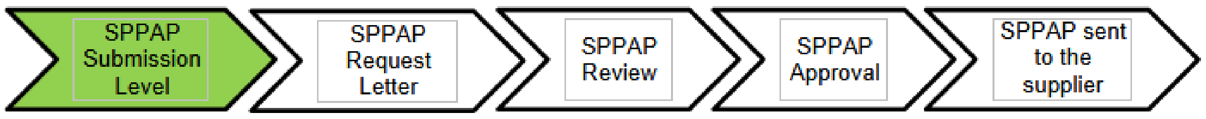
Characteristics		Note QC or SC in this Column	Methods						Reaction Plan
Product	Process		Product / Process Specification / Tolerance	Eval. Meas. Technique	Sample			Control Method	
					Size	Freq. Ord	Freq Rinf		
Quote QC		QC		CMM	2pc	4 ore	2 ore	Zeiss report	L'operatore deve avvisare il supervisore. Correggere la lavorazione se posiibile. Selezionare dall'ultimo particolare conforme misurato
Quote Normali				CMM	2pc	4 ore	2 ore	Zeiss report	L'operatore deve avvisare il supervisore. Correggere la lavorazione se posiibile. Selezionare dall'ultimo particolare conforme misurato
Quote con strumenti tradizionali				Altimetro, Blocchetti Jonson, Calibro	2pc	4 ore	2 ore	Specifiche del disegno	L'operatore deve avvisare il supervisore. Correggere la lavorazione se posiibile. Selezionare dall'ultimo particolare conforme misurato
Tamponi				Tampone	1 pc	12 pz	6 pz	Scheda di controllo	L'operatore deve avvisare il supervisore. Correggere la lavorazione se posiibile. Selezionare dall'ultimo particolare conforme misurato
			Other Dimension Verification						
Bave di lavorazione				Visuale	1 pc	At start up	At start up	Comparare con Master	L'operatore deve avvisare il supervisore. Correggere la lavorazione se posiibile. Selezionare dall'ultimo particolare conforme misurato
Raggi lavorati				Visuale	1 pc	At start up	At start up	Comparare con Master	L'operatore deve avvisare il supervisore. Correggere la lavorazione se posiibile. Selezionare dall'ultimo particolare conforme misurato
Rugosità				Rugosimentro	1 pc	At start up	At start up	Specifiche del disegno	L'operatore deve avvisare il supervisore. Correggere la lavorazione se posiibile. Selezionare dall'ultimo particolare conforme misurato

Control Plan of the Gearbox [2]

- Gage R&R Studies: Verifies the reliability of the measurement system through repeatability and reproducibility analysis, ensuring consistent data collection. This is a fundamental study, which will be further analyzed in Chapter 4.
- Statistical Studies: Tools such as process capability (e.g., Cpk) are computed for each critical dimension, confirming the process can produce parts within specifications. This section also has its own sub-chapter in Chapter 5.

These elements need to be inserted in the SPPAP document and collectively ensure that suppliers meet SBD's standards for quality and consistency.

2.3 Submission and Approval



SPPAP Process Flow [3]

The SPPAP submission process varies depending on the complexity and risk level associated with the part. Stanley Black & Decker (SBD) has defined five levels of submission to ensure the appropriate level of documentation and validation:

1. Level 1: Warrant Only
   Reserved for minor changes—such as part number or supplier name updates—that do not affect the process or design.
2. Level 2: Warrant with product samples and limited supporting data
   Used for low-to-medium risk parts or small material or revision changes.
3. Level 3: Warrant with product samples and complete supporting data
   This is the default submission level for most parts, including new components and any modifications affecting form, fit, or function.
4. Level 4: Warrant and other requirements defined by the customer
   Applied when only specific documents or checks are needed, typically for less critical adjustments.
5. Level 5: Warrant with product samples and full documentation review at the supplier's location
   The most comprehensive level, involving an on-site evaluation of all relevant materials and processes.

To initiate the process, a SPPAP Request Letter is prepared. This document outlines the submission level required and specifies which elements the supplier must include. It is the responsibility of the Supplier Quality Engineer (SQE) to define this level and clearly communicate it to the supplier.

Once the SPPAP package is submitted, the SQE performs a thorough review of all documentation to confirm compliance with the Request Letter. This review includes several critical components:

- Dimensional Reports: Evaluated by engineering to confirm that measurements meet design requirements.
- Functional and Mechanical/Electrical Test Reports: Reviewed to ensure the part functions correctly under expected conditions.
- Process Documentation: Includes the Control Plan, PFMEA, Process Flow Diagrams, and Capability Studies. The SQE checks each document for completeness, accuracy, and alignment with quality expectations.

After reviewing all materials, the SQE consolidates feedback and issues one of three possible outcomes:

1. Approved – The supplier is authorized to begin full production and delivery.
2. Rejected – The submission does not meet requirements; corrections are needed before resubmission.
3. Interim Approval – A conditional green light for limited production under specific constraints, such as quantity or time frame.

The final decision must be documented, signed by both the SQE and an engineering representative, and formally communicated back to the supplier. In cases of rejection, the supplier is not permitted to deliver any parts until all nonconformities are resolved.

SPPAP is a fundamental tool for ensuring quality and reducing production risk. By standardizing expectations and documentation, it supports transparency and consistency in supplier relationships. Within SBD, it is not just a compliance tool—it represents a structured approach to building trust and maintaining performance throughout the supply chain.

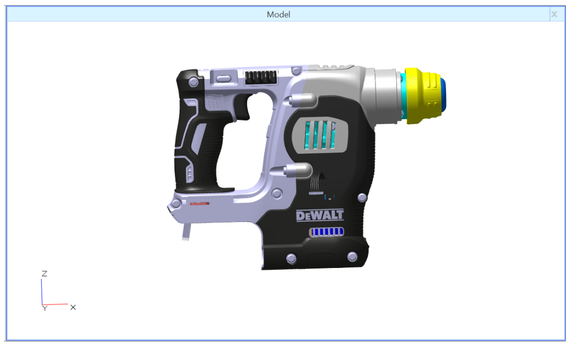
3 Description of the Standardized Component

3.1 The L-Shape Rotary Hammer

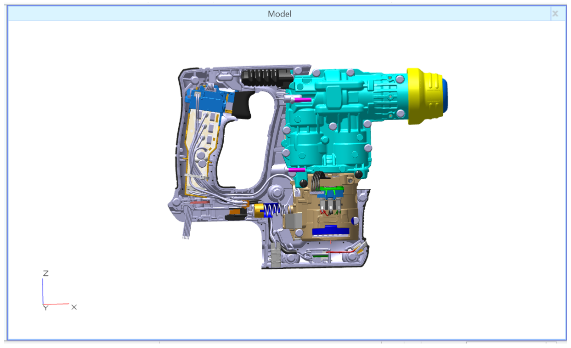
Brushless Cordless L-Shape Rotary Hammer [4]

The L Shape Rotary Hammer is the power tool into which the standardized gearbox discussed in this thesis is integrated. This advanced rotary hammer is designed for drilling applications in concrete and masonry, embodying high performance and durability.

I include here two images to help readers better understand the position of the gearbox within the tool. The first image shows the tool in its 'closed' state, while the second image displays the tool in its 'open' state, with internal components visible. In the 'open' state, the gearbox is highlighted in light blue for clarity.

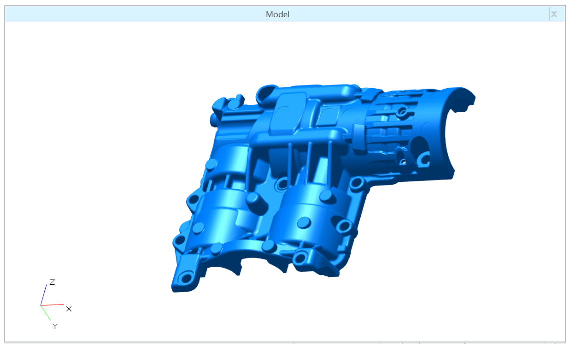


Windchill 3D Model 'closed' [5]



Windchill 3D Model 'open' [6]

3.2 Gearbox of the Tool



Windchill 3D Model of the Gearbox [7]

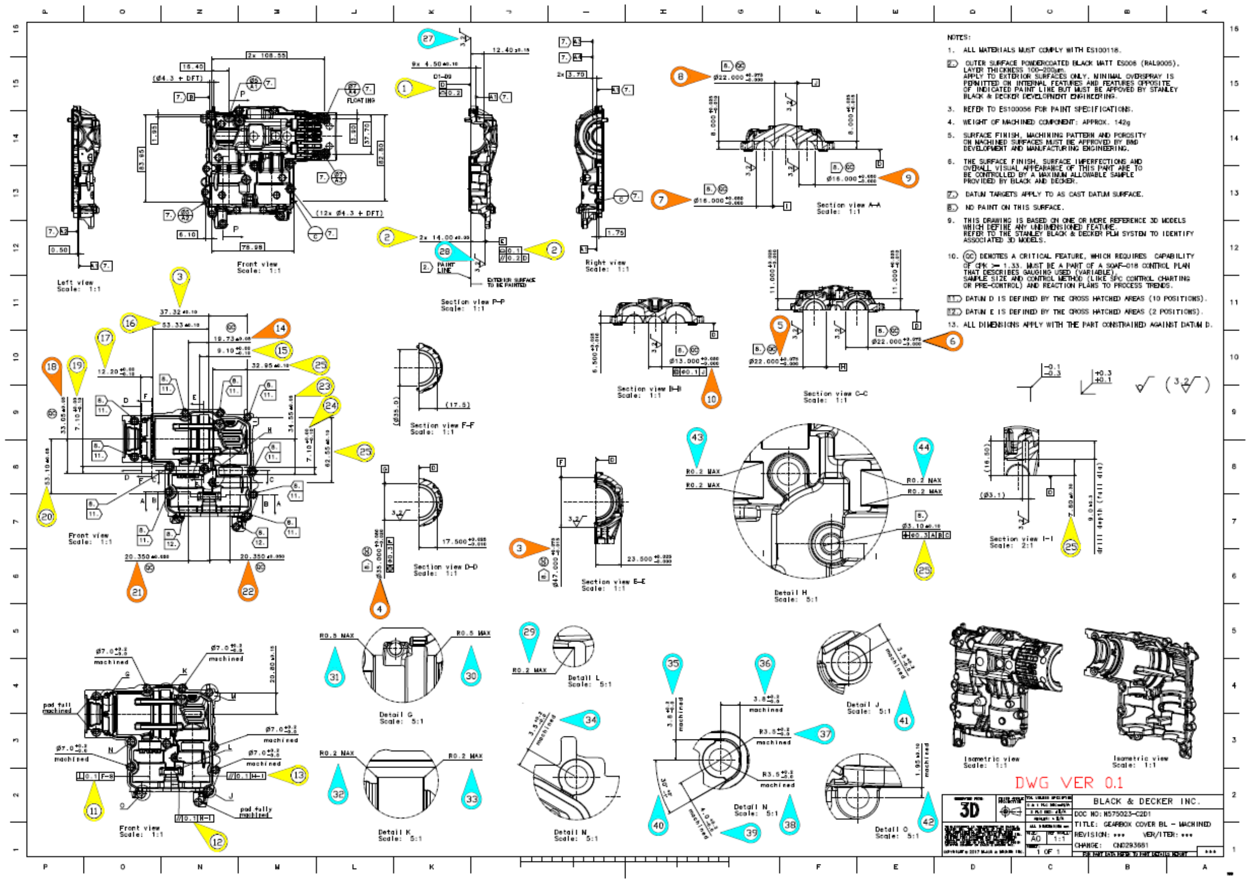
The gearbox is a critical component of the rotary hammer. It is responsible for transmitting power from the motor to the hammer mechanism and drill bit, ensuring efficient energy transfer and reliable tool performance. The gearbox also plays a vital role in converting rotational motion into the oscillating motion necessary for hammering action.

In the tool's design, the gearbox must interact seamlessly with other components to ensure proper assembly and functionality. Its shape and geometries are tailored to fit precisely within the housing, forming secure connections with adjacent parts such as the motor, hammer mechanism, and support brackets. This design ensures stability during operation while minimizing vibration and wear.

The gearbox is composed of 44 individual elements, as indicated in the next image [8]. However, not all points represent unique features; some reference multiple characteristics of the same element. This distinction means the actual number of dimensions and parameters requiring inspection rises to 60. Among these, 12 features are categorized as QC, which are essential for ensuring compliance with safety and performance standards.

The intricate design of the gearbox reflects the need to accommodate connections with other tool elements, such as alignment slots, fastening points, and contact surfaces for power transfer. These design considerations highlight the importance of precise manufacturing and thorough dimensional control to achieve optimal performance and durability.

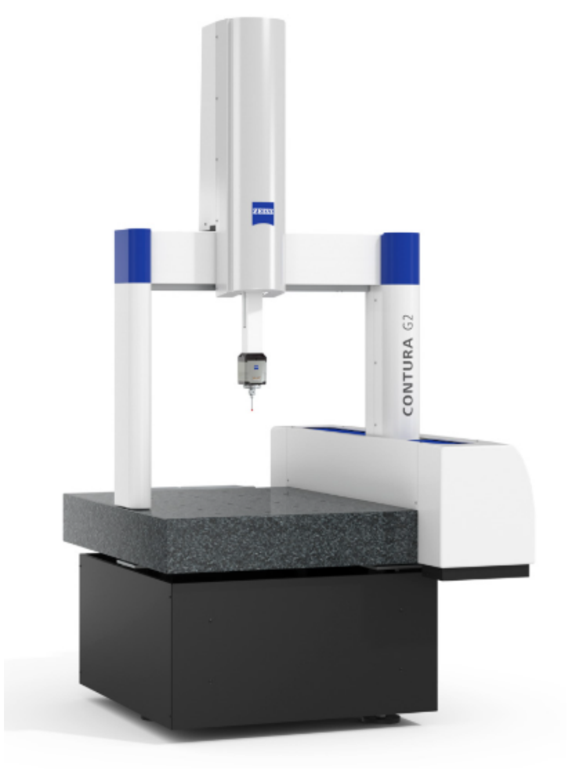
The performance of the tool strictly depends on the dimensions of this component, making it a key focus area for quality validation and process optimization. In the following sections, the methodology for controlling and verifying these critical features will be detailed.



Ballooned Technical Drawing [8]

4 The Measuring Instrument and the Gage R&R

4.1 ZEISS Contura CMM



ZEISS CONTURA [9]

Accurate dimensional measurements are essential, particularly when standardizing components, to ensure consistency, quality, and proper fit within an assembly. To achieve this, a highly reliable measurement system with minimal error margins is required. The ZEISS CONTURA Coordinate Measuring Machine (CMM) is the primary instrument used for dimensional verification, not only for the gearbox discussed in this thesis but also for all other components produced or inspected within the facility. Although a stereo vision scanner is occasionally used for specific types of quality control, the CMM remains the main tool for precision measurements.

The ZEISS CONTURA is a high-precision coordinate measuring machine designed for quality control in industrial production. It operates by using a probing system to collect measurement data from a component's surface, allowing for highly accurate dimensional verification. The machine is built to minimize external influences such as thermal expansion or vibrations, ensuring repeatability and precision in all measurements.

The system works by taking initial datum points (reference points) on the component, which serve as the foundation for the measurement process. After defining these references, additional points are captured, and planes, axes, and geometric features are constructed to create a coordinate framework in 3D space.

Once this spatial framework is established, the CMM collects all necessary measurements, including distances, depths, and geometric relationships between features.

VAST XT Probe [10]

The VAST XT probe head is utilized in the ZEISS CONTURA CMM, providing a significant advantage over conventional probes. While standard probes must detach from the component after each measurement to reposition for the next, the VAST XT can continuously scan along the surface without lifting, thanks to its pressure-sensing capability. This allows it to detect variations in surface pressure and generate an accurate 3D representation of the component being measured. By continuously moving along the surface, the VAST XT enables faster, smoother, and more precise measurements, reducing the time required for inspection and increasing accuracy. The acquired data is then compared to the ideal 3D model, verifying whether the parameters conform to specified tolerances. This method ensures that all components meet design requirements before being approved for production or assembly.

The ZEISS CONTURA CMM is operated through CALYPSO software, which provides a user-friendly interface for defining measurement programs. With CALYPSO, users can:

- Create measurement paths.
- Define specific parameters to be inspected.

- Select the appropriate probes and sensors for each measurement.
- Program the sequence of operations, ensuring an efficient and standardized workflow.

Once a measurement program is created, it is saved and reused for quality control of future production batches. For each component, the operator first manually controls the probe, reaching the datum points using the controller, after which the ZEISS CONTURA automatically follows the predefined measurement sequence. This automated process ensures repeatability and reliability in quality control while minimizing human error. The combination of precision hardware, advanced probe technology, and intelligent software makes the ZEISS CONTURA an essential tool in ensuring that all measured components, including the gearbox, comply with required specifications.

4.2 Gage R&R Process and Results

The Gage R&R (Repeatability and Reproducibility) study is an essential part of the SPC (Statistical Process Control) and plays a crucial role in ensuring the accuracy and reliability of measurement systems used for the standardization of mechanical components. Gage R&R helps assess the measurement system's ability to detect true variations in the parts, distinguishing between the variation introduced by the measuring instrument (repeatability) and that caused by different operators (reproducibility).

As introduced in the second chapter of this thesis, the Gage R&R study is a key element of the SPPAP and it's mandatory to send it to the customer who will receive the components. This study ensures that the measurement system, as the ZEISS CMM in our case, used for assessing the mechanical component's dimensions, is both reliable and consistent, which is critical when aiming for the standardization of the component.

This method identifies two primary sources of variation:

1. Repeatability: The variation observed when the same operator measures the same part multiple times using the same equipment.

2. Reproducibility: The variation observed when different operators measure the same part using the same equipment.

By calculating the percentage of total variation that is due to the measurement system, it helps determine if the system is sufficiently precise or if improvements are needed.

The capability of the measurement system is typically expressed using two statistical measures which refer to the "Can you adequately assess process performance?" (1) and "Can you sort good parts from bad?" (2) sections (see [11.2]). These are calculated as:

$$(1)\left(\frac{\sigma_{Gage}}{\sigma_{Total}}\right)\times 100\%$$

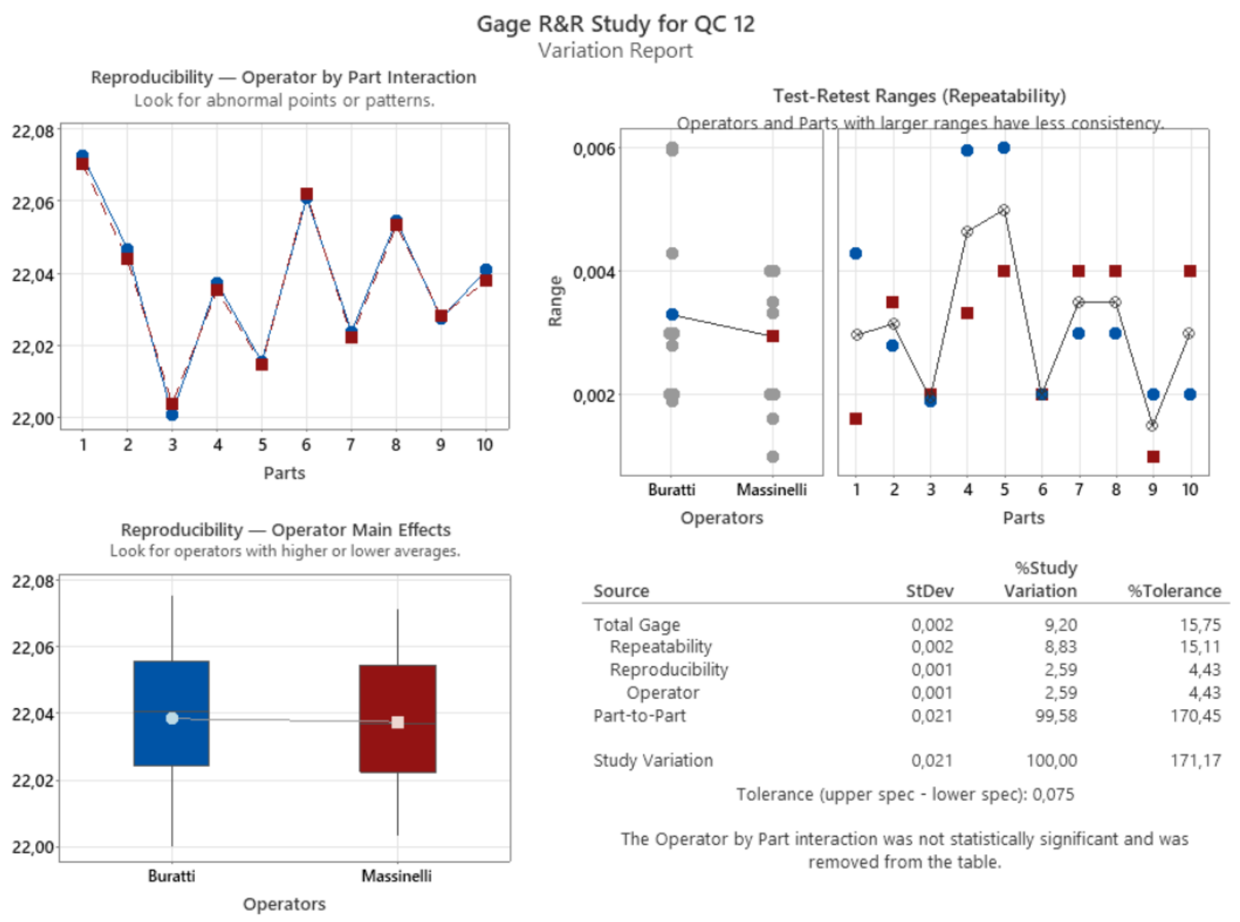
$$(2)\left(\frac{Tolerance_{Gage}}{Nominal\ Value}\right)\times 100\%$$

where:

- σ_{Gage} is the standard deviation of the measurement system, comprising both repeatability and reproducibility;
- σ_{Total} is the total variation observed in the process, comprising part-to-part variation;
- Nominal Value: The ideal or target dimension of a part;
- Tolerance$_{Gage}$: The range of error or variation introduced by the measuring instrument.

Lower values indicate a more reliable measurement system. Ideally, these values should be less than 30% to be considered acceptable.

In this study, a total of 10 mechanical components are selected for measurement. Measurements are taken by two different operators and 3 measurements of each component are made by each operator. The measurements taken by each operator were then analyzed to calculate the repeatability and reproducibility of the measurement system.



Source	StDev	%Study Variation	%Tolerance
Total Gage	0,002	9,20	15,75
Repeatability	0,002	8,83	15,11
Reproducibility	0,001	2,59	4,43
Operator	0,001	2,59	4,43
Part-to-Part	0,021	99,58	170,45
Study Variation	0,021	100,00	171,17

Tolerance (upper spec - lower spec): 0,075

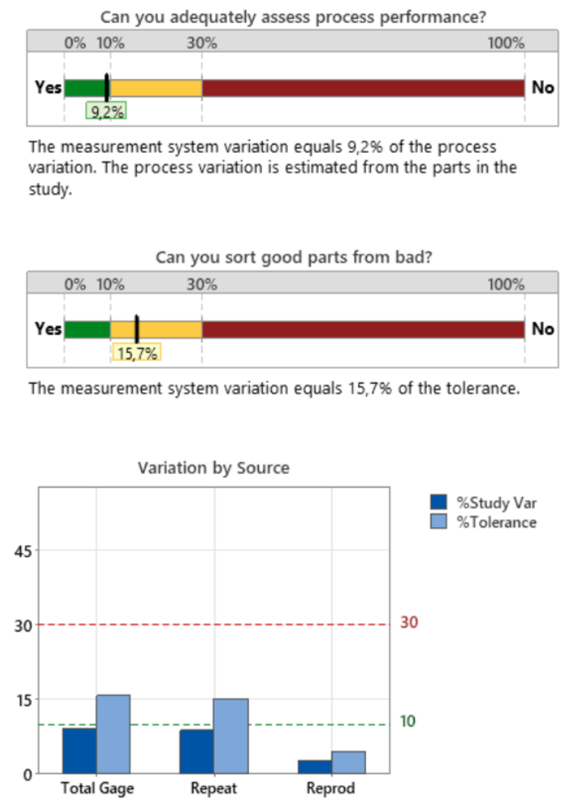
The Operator by Part interaction was not statistically significant and was removed from the table.

Gage R&R Results [11.1]

It is important to note that selecting components with a greater variance or those that fall outside the tolerance limits can provide more accurate insights into the measurement system's capability. If all components chosen for the study are very similar or perfectly within tolerance, the Gage R&R percentage could appear higher than it actually is. This is because the measurement system would likely show little variability, which might be attributed to the machine itself, even though the machine could be performing well. By choosing components with a wider range of measurements or more variation, the Gage R&R study can more effectively assess whether the measurement system is able to distinguish between genuine differences between parts, rather than only reflecting inherent measurement system noise.

Performing a Gage R&R study with more variable components also ensures that the measurement system is capable of detecting differences in part dimensions, providing a more accurate representation of its true performance.

A Gage R&R study is conducted for each QC and all of them should confirm that the measuring instrument can be used for the following standardization processes. After collecting the data and performing the necessary statistical analysis, the following results were observed for one of the QC:



The measurement system variation equals 9,2% of the process variation. The process variation is estimated from the parts in the study.

The measurement system variation equals 15,7% of the tolerance.

Gage R&R Results [11.2]

These results suggest that the measurement system is sufficiently precise for the purposes of standardizing the mechanical component. With an acceptable Gage R&R percentage, we can confidently proceed with the SPPAP procedures, knowing that the measurement system is reliable and will provide consistent results for component evaluation and quality control.

5 Cpk Index and Other Statistical Tools

5.1 Cpk Index

The Cpk Index (Process Capability Index) is a statistical measure used to evaluate a manufacturing process's ability to produce parts within specified tolerance limits. It quantifies how well the process mean aligns with the nominal value while considering the spread of measurements. A higher Cpk value indicates that the process is well-centered and exhibits minimal variation, ensuring product quality. For instance, internal rules of the company suggest that a process can be targeted as 'capable' if the Cpk index is equal or larger than 1.33, which is the acceptable value for the standardizations of the components.

The Cpk index is calculated using the following formula:

$$Cpk = \min\left(\frac{USL - \mu}{3\sigma}, \frac{\mu - LSL}{3\sigma}\right)$$

where:

- USL = Upper Specification Limit
- LSL = Lower Specification Limit
- μ = Process Mean
- σ = Process Standard Deviation

The Cpk index evaluates both accuracy (alignment of measurements with the target value) and precision (consistency of repeated measurements). A low Cpk can result from excessive variation in the process or from a well-controlled but off-centered process, leading to non-conforming parts. Therefore, both factors must be optimized to achieve a capable process.

For statistical reliability, a sufficiently large and homogeneous sample is necessary. As specified in the Six Sigma Study Guide [12], the minimum recommended sample size is 30 components to ensure the data follows a normal distribution and captures the natural variation in the process. Additionally, all sampled components must belong to the same production batch to minimize external influences such as material differences or machine calibration variations.

As discussed in previous chapters, the Cpk index is only calculated for dimensional QC and is applied exclusively during the standardization phase of the gearbox component. This ensures process capability before mass production begins.

The Cpk index calculation assumes that the distribution of measured values follows a normal distribution. If the data exhibits skewness or another non-normal pattern, preliminary statistical adjustments are necessary before proceeding with Cpk calculations. These alternative statistical analyses will be addressed in later section.

To ensure consistency and accuracy in calculating the Cpk index, the company follows a standardized procedure with the following steps:

1. Perform Gage R&R Study – This step verifies the measurement system's reliability before proceeding with further measurements.
2. Measure at least 30 components in production order – Measurements must be taken sequentially in the same order they were produced, ensuring that variations in process conditions are accurately reflected.
3. Record and collect measurement data – All dimensional QC are collected for statistical analysis.
4. Input data into Minitab software – The recorded values, including upper and lower limits, are entered into Minitab for analysis.
5. Execute Cpk calculation using Minitab's built-in function – Minitab processes the data and computes the Cpk index along with other statistical parameters.

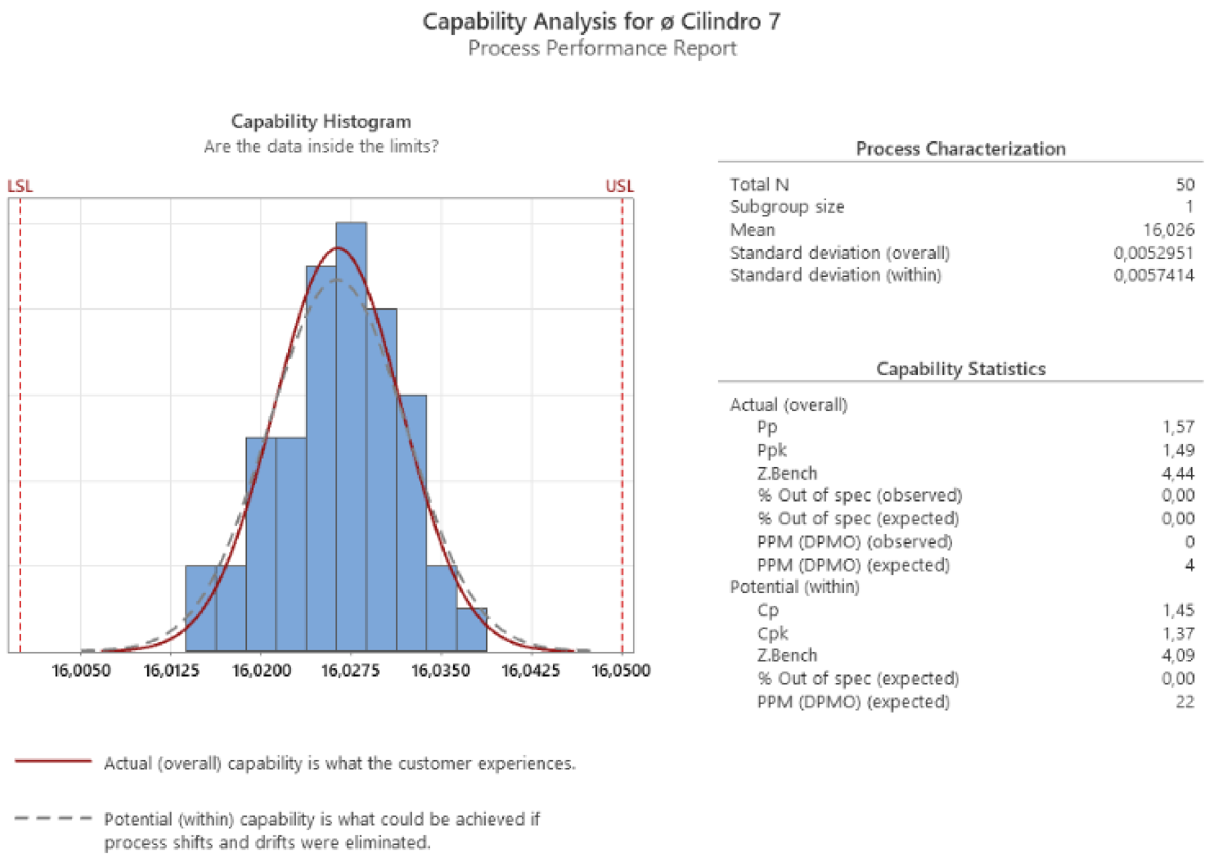
By following this structured approach, the company ensures that process capability is accurately assessed and maintained within acceptable limits, supporting the overall standardization efforts of the gearbox component.

One of the results of the study can be seen in the next image [13]. This graph represents the histogram of the collected values of a QC dimension of the gearbox.

In this specific case, the dimension is referred to the diameter of a cylinder, which has a nominal value of 16.025 mm, the UCL is 16.050 mm and the LCL is 16.000

mm. We can see from the image that the Cpk index is equal to 1.37 and thus confirm that the process to manufacture this specific dimension is capable.

Other parameters are reported but some of them are useless in the standardization phase, while the importance of Cp, Ppk, Z-Bench and PPM will be discussed in the next sub-chapter.



Process Characterization

Total N	50
Subgroup size	1
Mean	16,026
Standard deviation (overall)	0,0052951
Standard deviation (within)	0,0057414

Capability Statistics

Actual (overall)	
Pp	1,57
Ppk	1,49
Z.Bench	4,44
% Out of spec (observed)	0,00
% Out of spec (expected)	0,00
PPM (DPMO) (observed)	0
PPM (DPMO) (expected)	4
Potential (within)	
Cp	1,45
Cpk	1,37
Z.Bench	4,09
% Out of spec (expected)	0,00
PPM (DPMO) (expected)	22

Minitab Results [13]

5.2 Associated Parameters

The Cpk Index is closely related to several other statistical parameters that provide additional details about the capability and stability of the process. Among these, Cp, Ppk, and Z-Bench allow us to analyze how centered the process is, how it varies over time, and the probability of producing defective components (PPM).

The Cp (Process Capability Index without considering the shift) evaluates the theoretical capability of the process without considering its centering relative to the specification limits. The formula is:

$$Cp = \frac{USL - LSL}{6\sigma}$$

where σ represents the short-term standard deviation.

As already mentioned, Cpk also evaluates how much the process is shifted from the nominal value, so if Cp > Cpk, it means the process is off-center: even though the variability is acceptable, the mean is shifted towards one of the specification limits, increasing the risk of out-of-tolerance parts.

In the analysis table of the previous image, the values are:

- $Cp = 1.45$
- $Cpk = 1.37$

Since Cp is greater than Cpk, the process is not perfectly centered, but still good enough.

The Ppk (Process Performance Index) is calculated exactly as the Cpk, but it uses the overall standard deviation instead of the short-term standard deviation. Since σ_{overall} includes variations due to external factors (materials, environmental conditions, machine drift over time), Ppk is only useful in long-term analysis. However, since all components come from the same batch, Ppk is not relevant during standardization: the only parameter to consider is Cpk.

In this specific case, the values in the table are:

- $Ppk = 1.49$
- $Cpk = 1.37$

Here, Ppk is greater than Cpk, which may seem counterintuitive at first, but can actually occur when all measurements come from the same batch. That is because Minitab software calculates both Cpk and Ppk even in such cases. Ppk is computed using the overall standard deviation of the data points (σ_{overall}). Cpk, on the other hand, requires specifying subgroups, corresponding to different production lots, in order to estimate the within subgroup standard deviation (σ_{within}). Since all the data come from a single batch, there is no actual between-subgroup variation, so σ_{overall} and σ_{within} should ideally represent the same variability. However, due to different estimation formulas, their numerical values may slightly differ, which can lead to Ppk > Cpk. In such cases, the Ppk value provided by Minitab may actually be a better representation of the process capability than the Cpk value. Despite that, Cpk calculated by Minitab is conventionally used to assess process capability in standardization phases.

The Z-Bench represents the normalized Z-score that quantifies the distance of the process from the specification limits, expressing it in terms of standard deviations. The formula is:

$$Z\text{-Bench} = \min\left(\frac{USL - \mu}{\sigma}, \frac{\mu - LSL}{\sigma}\right)$$

In a perfectly normal and centered process, Z-Bench is approximately 3 times the Cpk, but this relationship is not always valid when the process does not follow a normal distribution.

In the table of the previous figure [13], there are two distinct Z-Bench :

- Z-Bench overall $= 4.44$
- Z-Bench potential $= 4.09$

Here, Z-Bench overall is computed using σ_{overall}, while Z-Bench potential is calculated with σ_{within}. When subgroups are defined, Z-Bench potential is the appropriate value to use for estimating the Z-Bench of the process, but as in the previous case, the Z-Bench overall can be considered a better estimation if all the measurements come from the same lot.

The PPM (Parts Per Million) indicates the estimated number of parts out of specification per million products. It is related to Z-Bench value with an inverse and exponential relation. If the process is centered :

Z-Bench	PPM (Defects per million)	Sigma Level	Cpk
6.00	0.002	Six Sigma	2.00
5.00	0.57	-	1.67
4.50	3.40	-	1.50
4.00	63	-	1.33
3.00	2,700	-	1.00
2.00	45,500	-	0.67

Relationship between Z-Score and other parameters [14]

In our case:

- Z-Bench potential = 4.09 $\rightarrow$ PPM potential = 22
- Z-Bench overall = 4.44 $\rightarrow$ PPM overall = 4

As previously analyzed, given that all the measurements come from the same production lot, PPM overall = 4 can better estimates the expected value of production wastes.

This last value is particularly relevant since it tells us how many errors we can expect from the working tool, also if the Cpk value is the one to be always checked during the standardization and a good value of it usually means that other parameters are good too, also because they are strictly connected between each other.

5.3 Normalization by Means of Transforms

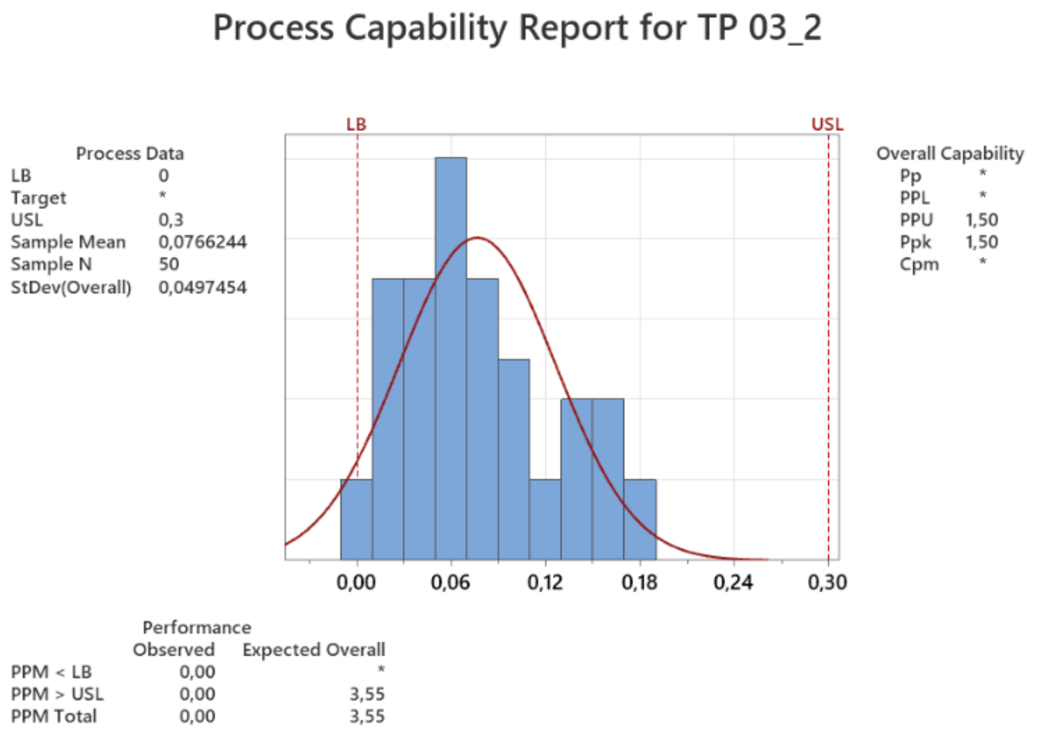
As already said, in statistical process control, the calculation of Cpk assumes that the collected data follows a normal distribution. However, certain parameters, such as True Positions, do not correspond to a specific magnitude but rather represent a distance from a reference point in space. These parameters frequently exhibit non-normal distributions, often resembling a Chi-squared distribution. While True Position parameters are rarely considered QC, in cases where Cpk analysis is required, normalization must be performed.

To assess whether a dataset follows a normal distribution, it is common practice to perform a normality test. These tests assume, as their null hypothesis, that the data are normally distributed. Once the test is performed, the p-value helps us decide whether or not to reject this assumption, since, a high p-value increases our confidence in the assumption of normality.

In general, if the p-value is below a chosen threshold (commonly 0.05), the normality hypothesis is rejected, and a transformation may be required to improve the data's adherence to a normal distribution.

The p-value also guides the selection of the most appropriate transformation. In Minitab, for example, twelve different transformation families are available, and the optimal one is typically chosen based on the resulting p-value of the goodness-of-fit test.

- The higher the p-value, the greater the confidence that the transformed data approximates a normal distribution.
- A p-value greater than 0.05 suggests that the transformed data does not significantly deviate from normality, making it acceptable for further capability analysis.



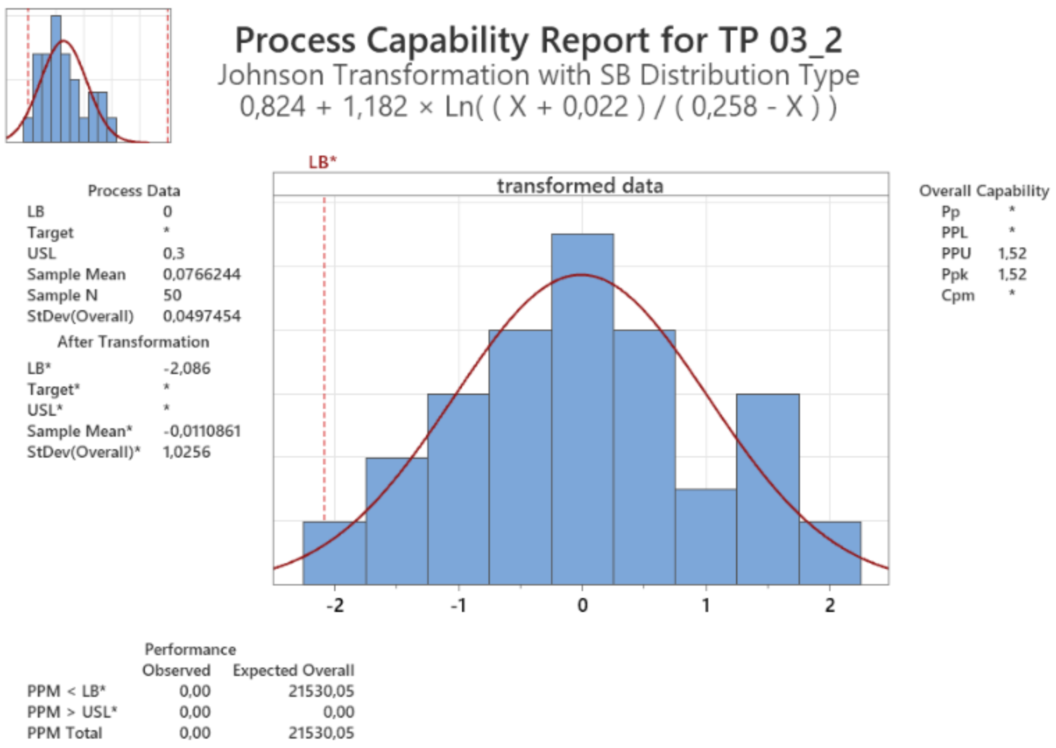
Process Capability Report for TP 03_2

Process Data	
LB	0
Target	*
USL	0,3
Sample Mean	0,0766244
Sample N	50
StDev(Overall)	0,0497454

Overall Capability	
Pp	*
PPL	*
PPU	1,50
Ppk	1,50
Cpm	*

Performance	Observed	Expected Overall
PPM < LB	0,00	*
PPM > USL	0,00	3,55
PPM Total	0,00	3,55

True Position Distribution [15]



Process Capability Report for TP 03_2
Johnson Transformation with SB Distribution Type
0,824 + 1,182 × Ln((X + 0,022) / (0,258 - X))

Process Data	
LB	0
Target	*
USL	0,3
Sample Mean	0,0766244
Sample N	50
StDev(Overall)	0,0497454
After Transformation	
LB*	-2,086
Target*	*
USL*	*
Sample Mean*	-0,0110861
StDev(Overall)*	1,0256

Overall Capability	
Pp	*
PPL	*
PPU	1,52
Ppk	1,52
Cpm	*

Performance	Observed	Expected Overall
PPM < LB*	0,00	21530,05
PPM > USL*	0,00	0,00
PPM Total	0,00	21530,05

Transformed Data [16]

In the provided case (transformation from [15] to [16]), the Johnson Transformation was selected, as it yielded the highest p-value (0.733), indicating a strong alignment with normality.

Once the transformation is applied, the new distribution follows a normal shape, effectively representing the original data in a way that allows capability indices such as Cpk to be computed reliably. The transformed histogram visually shows how the data behaves after the transformation.

Regarding capability indices, Ppk is used instead of Cpk in these cases because the software does not compute the Cpk index. However, as already discussed, when data come from a single lot, the difference between Ppk and Cpk is often minimal, and the Ppk may estimate the capability of the process even better than the Minitab's Cpk. Therefore, in these situations, monitoring Ppk is sufficient to determine whether the machine is capable.

In this case, the new Ppk value is 1.52, reflecting the capability of the process under normalized conditions. This transformation ensures that capability analysis remains valid, even for non-normally distributed parameters like True Position, thus maintaining accurate process evaluation.

Part 2

Statistical Analysis of the Evolution of the Cpk Index and its Benefits

6 Data Collection and Feature Engineering

6.1 Data Collection

At the beginning of the analysis, the first challenge I encountered was related to how data is collected within the company. The ideal way, to seriously analyze the trend of the Cpk index on any critical parameter, at least 30 measurements per batch would be required, and this would need to be done consistently over years. Clearly, such a data collection process would require long-term planning, which wasn't something that could have been proposed within the short time frame of my internship, which started and ended in the same academic year in which I wrote the thesis.

Because of that, I focused on understanding how the real data collection works: after the standardization phase, measurements are taken according to the control plans, and an example of these plans was provided in the first part of the thesis. This approach makes sense, especially since historically, monitoring the Cpk trend over time has never been a goal for the company, so there's no reason why additional measurements should be taken.

That said, there are still special cases, such as production problems or maintenance interventions, where more measurements than usual are taken. During those moments, valuable information might be gathered, but it is not tracked or noted in a structured way. I didn't have access to these details, but if they had been available, they would have been extremely useful for conducting a more precise analysis and gaining a better understanding of what happens at certain critical points.

For this project, I requested access to the individual measurements of the components produced over the last two years (January 2023 - January 2025). I was provided with all the individual measurements files, with recorded values for each part, which have been enough to start exploring some interesting patterns and understand how, even with few available data, something useful can be obtained.

Since it wasn't possible to perform an ”ideal” analysis, I had to make certain assumptions throughout the process to fill in the gaps in the data and ensure that the analysis could still provide meaningful insights.

6.2 Dataframe Construction

When I started looking at the data provided by the company, I quickly realized that the individual measurement values, in the format they were stored, weren't really usable for any kind of analysis. The data points were isolated, without any reference to the date of the measurement, and not even grouped by component type. All of these elements are absolutely necessary if you want to carry out a time series analysis.

As I mentioned earlier, a good analysis also needs a large amount of data collected over a long period. The component initially discussed in the thesis didn't have enough measurements to be useful in this sense. For that reason, I chose to switch focus to another component that had more than 600 measurements recorded over the past two years.

To prepare the data, I started by writing a Python script that could handle the original format of the files, which were stored in ZIP archives. The script allowed me to read these files, select only the ones related to the component I had chosen (based on their file names), and move them into a new folder for processing.

I also noticed that the file creation date matched the actual date the measurement was taken. That allowed me to use the file date as a reliable way to tag each measurement with its corresponding measurement date, a crucial detail for tracking how the values change over time.

Now, I will show the code used for this task. Before doing that, I would like to mention that I am also providing the script with all the necessary imports for the libraries used in the analysis. This ensures that the entire set of scripts can be run and reproduced directly within the context of this thesis.

```python
import os  # To manage file paths and operations
import zipfile  # To work with ZIP files
from datetime import datetime  # To handle and formatting dates
import fnmatch  # To filter file names
import plotly.express as px  # To create interactive charts
import plotly.graph_objects as go  # To add additional shapes to the chart
from plotly.subplots import make_subplots  # To make subplots
import plotly.io as pio  # To save plotly graphs as png
import pandas as pd  # To handle data in DataFrames
import numpy as np  # To do mathematical and statistical computations
from scipy.spatial.distance import mahalanobis  # To do the change detection
from astropy.timeseries import LombScargle  # To do the spectral analysis
from sklearn.metrics import mean_squared_error  # To find trends
from scipy.stats import gaussian_kde  # To check for normality
```

```
# Path to the ZIP folder containing the compressed data
zip_path = r'C:\Users\casam\Downloads\simone.zip'

# Path to the output folder where the modified files will be saved
output_folder = r'C:\Users\casam\OneDrive\Desktop\Simone\Tesi\MERGE\DATI 3'

# Open the ZIP file in read mode
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    # Retrieve the list of all files contained in the ZIP archive
    file_list = zip_ref.namelist()

    # Filter the files, only returning the ones with the specified name.
    # This needs to be done because we only want to select files which
    # refer to the same component
    chr_files = [f for f in file_list if fnmatch.fnmatch(f, 'chr/N739584*_chr.txt')]

    # Iterate over the filtered list of files
    for file in chr_files:
        # Retrieve file metadata to get the last modified date
        info = zip_ref.getinfo(file)

        # Convert the last modified date from tuple format to a human-readable string
        last_modified_date = datetime(*info.date_time).strftime('%Y-%m-%d %H:%M:%S')

        # Open each compressed file directly from the ZIP folder
        with zip_ref.open(file) as f:
            # Read the content of the file into a pandas DataFrame
            df = pd.read_csv(f, sep='\t', dtype=str, encoding='latin1')
        # Add a new column 'date' to store the last modified date of the original file
        df['date'] = last_modified_date

        # Construct the new file path with a modified name
        new_file = os.path.join(output_folder, os.path.basename(file).replace('_chr',
        '_modified.chr'))

        # Save the modified DataFrame as a new TSV file in the output folder
        df.to_csv(new_file, sep='\t', index=False)
```

Once the script was executed, I ended up with a folder full of .tsv files. Each of these files represented just one single measurement, and inside, every row was linked to one specific parameter. So basically, a full measurement included several rows, depending on how many parameters were checked.

The main issue at this point was that all this data was scattered across separate files. There was no central dataframe or table to work with. To make things easier, I used a company tool that let me pick a group of .tsv files and merge them into a single Excel file. This gave me one complete dataset with all the measurements together, in a much more practical format.

Another important thing is that the script I used earlier also added the correct date to each parameter row. After doing all this, the file was finally in a usable form and ready to be analyzed, which is exactly what I do in the next sections of the thesis.

7 Exploratory Data Analysis

7.1 Dataframe Inspection

Before starting the actual analysis, I performed an initial inspection of the dataset to make sure there were no inconsistencies, errors, or duplicated values that could affect the results.

The first step was converting the Excel file into a .pkl (pickle) file. This format is more efficient for data handling in Python: it allows faster reading and writing of large datasets, preserves data types more accurately, and integrates better with pandas for subsequent operations.

Below, I include the script used to convert the original file and the one used to read the new pickle file.

```python
def conversion(file_path, output_file):
    # Reads the ODS file and converts it into a DataFrame
    df = pd.read_excel(file_path, engine='odf')
    # Saves the DataFrame in pickle format
    df.to_pickle(output_file)

# Giving inputs to the function
conversion(r'C:\Users\casam\OneDrive\Desktop\Simone\Tesi\MERGE\data.3.ods',
 'file6.pkl')
```

As explained in the first part of this thesis, only QC parameters require the computation of the Cpk index. Therefore, to conduct this study, I selected a QC parameter corresponding to the diameter of a circular feature of the produced part. Since each row in the dataframe corresponds to a single parameter within one measurement, it was necessary to extract only the relevant rows corresponding to the QC parameter of interest.

To do this, I implemented a function that, given the complete dataframe and the row number of the parameter in the first measurement, returns only the rows and columns required for the analysis. This made it possible to isolate the data needed for a specific QC in a clean and systematic way. The function and its usage are shown below.

```
# Function to select and merge necessary rows and columns
def merge(read_file, qcoffset):

    df = pd.DataFrame(read_file)  # Transforms the input file in a DataFrame

    # Create a new 'row' column to index the rows of the DataFrame
    df['row'] = df.index + 2  # Adds an offset of +2 to match Excel row numbering

    # Select the first row containing the parameter we want to analyze
    first_row = df[df['row'] == qcoffset]

    # Select the name of the parameter
    id = first_row['id'].values[0]

    # Select all the rows of the DataFrame that we need to analyze the parameter
    filtered_df = df[df['id'] == id].copy()

    # We need to create a column containing all the dates,
    # which were first divided in two different 'date' columns
    # due to a formatting error in the dataframe. We fill the
    # empty rows of the firts columns with the dates contained in the second one
    filtered_df['date_c'] = filtered_df['date'].fillna(filtered_df['date.1'])

    # Select only the relevant columns to be returned
    needed_col = filtered_df[['row', 'deviation', 'uppertol',
     'lowertol', 'actual', 'nominal', 'date_c']]

    return needed_col  # Returns the relevant columns for the next analysis
```

Once the dataset was read and correctly structured, I moved on to a general inspection to verify its quality. The first step involved checking the structure of the dataset. Specifically, the data types of each column and the presence of null values. The code used for this task is also provided here.

The offset parameter has been set to a fixed value for the purpose of this specific analysis, but it can be easily modified to accept user input in order to support more general or customizable applications.

```
# Get the inputs to call the funcion
read_file = pd.read_pickle('file6.pkl')  # File containing data
qcoffset = 2 # Offset indicating the row of the QC to be analyzed

# Merge the data and get the info to understand the structure
qc_rows = merge(read_file, qcoffset)
qc_rows.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 618 entries, 0 to 32419
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   row        618 non-null    int64
 1   deviation  618 non-null    float64
 2   uppertol   618 non-null    float64
 3   lowertol   618 non-null    float64
 4   actual     618 non-null    float64
 5   nominal    618 non-null    float64
 6   date_c     618 non-null    object
dtypes: float64(5), int64(1), object(1)
memory usage: 38.6+ KB
```

Then, I used a second script to check for duplicate rows and to analyze the minimum and maximum values for all numerical columns. Again, the code for this step is included.

```
# Since the date_c column is filled with 'object', we convert to datetime values
qc_rows['date_c'] = pd.to_datetime(qc_rows['date_c'])
# Now we can sort our data by date
qc_rows = qc_rows.sort_values(by='date_c')
```

```
# Ensuring that there are no duplicates in our dataframe
print(qc_rows.duplicated().sum())

# Check if there are anomalous values
qc_rows_float = qc_rows.select_dtypes(include=['float64'])
min_values = qc_rows_float.min()
max_values = qc_rows_float.max()
summary = pd.DataFrame({
    'Min': min_values,
    'Max': max_values})

# Show the summary
summary
```

```
0
```

	Min	Max
deviation	-0.022031	0.019976
uppertol	0.025000	0.025000
lowertol	-0.025000	-0.025000
actual	37.947969	37.989976
nominal	37.970000	37.970000

The first 0 in the output refers to the number of duplicates, while the rows of the table actually represents minimum and maximum values for these parameters :

- Deviation – the difference between the nominal value and the actual measurement;
- Uppertol – the upper tolerance limit;
- Lowertol – the lower tolerance limit;
- Actual – the actual measured value;
- Nominal – the nominal value of the parameter.

The results of these inspections clearly showed that the data were clean, consistent, and ready for statistical analysis. No further preprocessing was necessary.

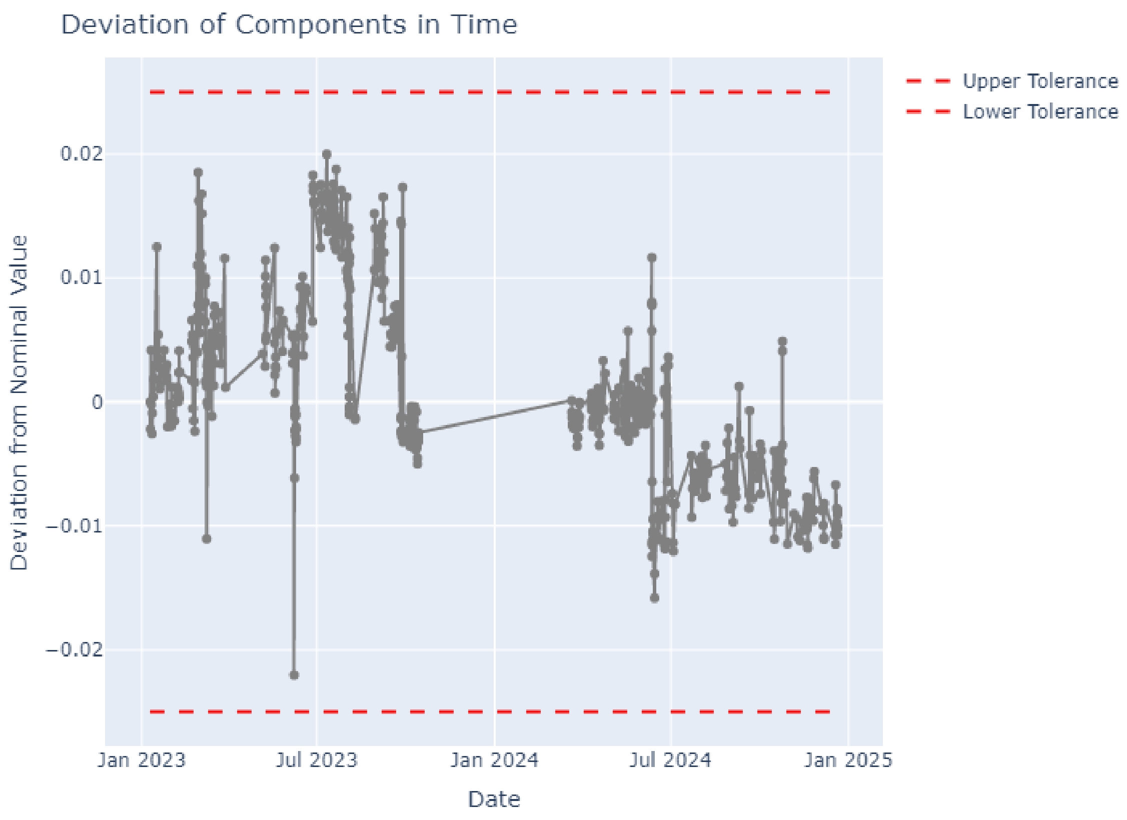
7.2 First Visual Inspection

After I checked the dataset and confirmed there were no duplicates, missing values, or obvious errors, I started with a first visual check of the data. This is not a complex step, but it helps to understand if the values look consistent or if something feels unusual before continuing with more detailed analysis.

What I did next was to plot the values from the "deviation" column. These values basically show the difference between the nominal reference and what was actually measured, and I decided to look at how they change over time. To make the graph easier to read and interpret, I also added the upper and lower tolerance levels, so the deviations can be seen more clearly in relation to the limits.

The code I used for the plot is shown below, and after that there is the figure that was generated.

```
# Create a line plot with the date on the x-axis and the deviation on the y-axis
fig = px.line(qc_rows, x='date_c', y='deviation', markers=True,
title='Deviation of Components in Time')
fig.update_traces(line=dict(color='gray'))

# Add the upper and lower tolerance line (constant value) to the chart
UT = qc_rows['uppertol'].iloc[0]
LT = qc_rows['lowertol'].iloc[0]

fig.add_trace(go.Scatter(
    x=[qc_rows['date_c'].min(), qc_rows['date_c'].max()],
    y=[UT, UT],
    mode='lines',
    line=dict(color='red', dash='dash'),
    name='Upper Tolerance'))

fig.add_trace(go.Scatter(
    x=[qc_rows['date_c'].min(), qc_rows['date_c'].max()],
    y=[LT, LT],
    mode='lines',
    line=dict(color='red', dash='dash'),
    name='Lower Tolerance'))

fig.update_layout(xaxis_title='Date', yaxis_title='Deviation from Nominal Value')

# Save as PNG for LaTeX/PDF output
pio.write_image(fig, "figures/deviation-plot.png", format="png")
```



From this visualization, we can already highlight some interesting points. First, the data appear reliable, and there is nothing that suggests an error on the dataframe. Thus, the ”data cleaning”, which has been just a check since there has not been the necessity to change anything, can be considered to be over.

Other than that, the plot does reveal some noteworthy aspects that influenced the design of the subsequent analysis. Most notably, there are no defective parts produced, which implies good capability of the working tools. Additionally, there is a big gap in the measurement timeline (from mid-October 2023 to mid March 2024), which could correspond to a production stop. To determine whether this was specific to the analyzed component, I repeated the visualization for other parameters and components, finding the same gap each time. This likely represents an issue in the company's data storage or acquisition process during that period, and nothing can be done to solve this problem. Finally, the data clearly exhibit changes in distribution and trends over time, which may correspond to a change in the machining tool used for production. For this reason, it would not be appropriate to compute Cpk indices on the full dataset without first splitting the data into separate intervals corresponding to different machine usage periods. The process utilized to identify these intervals marks the first substantial analytical step and will be described in detail in the next chapter, along with the other methods applied to reach the ultimate goal: tracking the evolution of Cpk over time.

8 Modeling and Analysis

8.1 Change Detection

As anticipated in the previous chapter, in order to correctly analyze the behavior of a parameter such as the Cpk, it is essential to identify the time points at which the machining tool was changed. Since I did not have access to any information about the actual changes of the production equipments, I had to implement a method capable of detecting these changes based on the available data.

The most indicative parameters of a potential machine change are the mean and variance of the measurements. Therefore, I computed moving averages and moving variances over a window of 10 consecutive data points. I thought that a big shift in either of these, or both of them, could indicate a tool change.

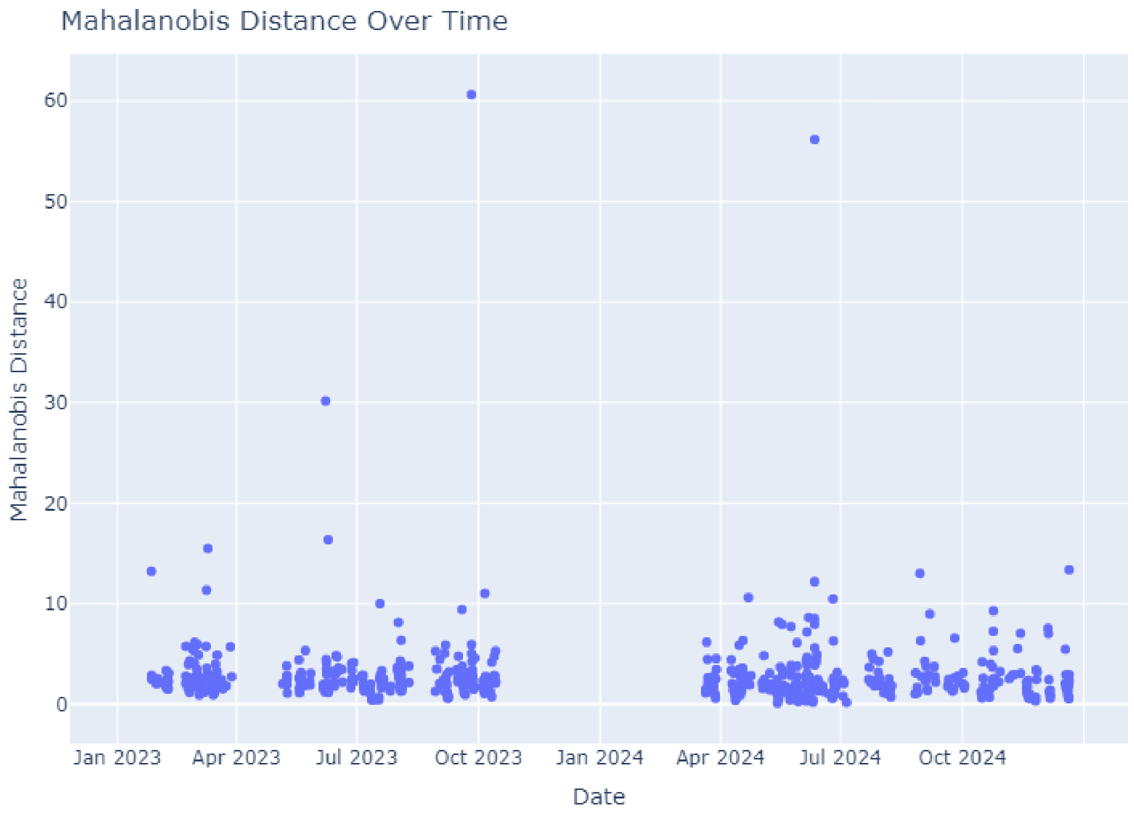
However, simply computing and comparing the differences in mean and variance at each new data point separately would not be optimal. That is because of the scale differences, which could bias the comparison if not normalized, and because the two variables could be correlated, and a change in one might influence the other. To address both issues, I used the Mahalanobis distance, which accounts for both the scale and the covariance between variables.

The Mahalanobis distance of a point from a multivariate distribution is defined as:

$$D_M(x) = \sqrt{(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

Generally speaking, x represents the vector containing the parameters of the new observation, μ represents the vector composed by the mean values of all the variables, and Σ represent the matrix of the covariances of all the variables. In this context, x represents the vector composed of the moving mean and moving variance at each new time window, basically removing the oldest data point and adding the new data point to compute the mean and the variance. μ represents the vector with the mean values of the previous moving averages both for the mean and the variance, and the covariance matrix represents how, in the last 10 windows, the mean and the variance are correlated between each other.

```
# Set the window size
window_size = 10

# Calculate the moving average and moving variance for the 'deviation' column
# Moving average
qc_rows['moving_avg'] = qc_rows['deviation'].rolling(window=window_size).mean()
# Moving variance
qc_rows['moving_var'] = qc_rows['deviation'].rolling(window=window_size).var()

# Initialize the column for Mahalanobis distance with NaN values
qc_rows['mahalanobis'] = np.nan

# Calculate the Mahalanobis distance for each point starting from the 20th index
for i in range((window_size*2)-1, len(qc_rows)):
    # Extract the previous window of 10 mean and variance values
    # Last 10 values of moving average
    prev_mean = qc_rows['moving_avg'].iloc[i-window_size:i]
    # Last 10 values of moving variance
    prev_var = qc_rows['moving_var'].iloc[i-window_size:i]

    # Compute the average mean and variance of the previous window
    mean_mean = np.mean(prev_mean)  # Mean of previous means
    mean_var = np.mean(prev_var)    # Mean of previous variances

    # Calculate the covariance matrix
    cov_matrix = np.cov(prev_mean, prev_var)

    # Invert the covariance matrix
    inv_cov_matrix = np.linalg.inv(cov_matrix)

    # Compute the Mahalanobis distance between the current mean/variance
    # and the previous window
    mahalanobis_score = mahalanobis([qc_rows['moving_avg'].iloc[i],
                                       qc_rows['moving_var'].iloc[i]],
                                       [mean_mean, mean_var], inv_cov_matrix)

    # Assign the calculated Mahalanobis distance to
    # the corresponding row in the DataFrame
    qc_rows.loc[qc_rows.index[i], 'mahalanobis'] = mahalanobis_score

# Create a scatter plot of Mahalanobis distance over time
fig = px.scatter(qc_rows, x='date_c', y='mahalanobis',
                 labels={'date_c': 'Date', 'mahalanobis': 'Mahalanobis Distance'},
                 title='Mahalanobis Distance Over Time')

# Save as PNG for LaTeX/PDF output
pio.write_image(fig, "figures/mahalanobis-plot.png", format="png")
```



As can be easily seen, by calculating the Mahalanobis distance between each moving window and the 10 previous ones, it was possible to highlight the points in time where the mean and the variance deviate significantly with respect to the general behaviour. The plot of the Mahalanobis distances reveals two distinct peaks, indicating potential change points in the machining tool: one at September 25, 2023, and another at June 11, 2024. To further support this hypothesis, I cross-checked these dates with the plot of the deviations over time. In both cases, there is a huge increase in the variability of the measurements shortly after the identified dates, which is a common indicator of a burn-in phase : a period during which the new machine has not yet stabilized its performance.

Consequently, the data can be split into three different intervals: the first from the beginning of the dataset to September 24, 2023. For the second, I chose to consider only the data after the big gap, since, from September 25 and the beginning of the gap, only few data were available, not in a sufficient quantity to compute the Cpk index. In fact,he second interval goes from the beginning of 2024 and June 10, 2024, while, the third goes from the end of June 2024, when the third machine has been stabilized, and the end of the dataset. These intervals will be used to calculate and analyze the Cpk index separately for each machine configuration.

8.2 Trends and Seasonality

Understanding if production data is influenced by recurring patterns (seasonality) can be extremely valuable in industrial environments. In a factory's routine, cycles related to work shifts, order schedules, maintenance activities and other repetitive events always exist. It is therefore important to understand if such cycles can affect the quality characteristics of the product. This would be undesirable, since the production should ideally meet predefined quality standards consistently, regardless of the external conditions.

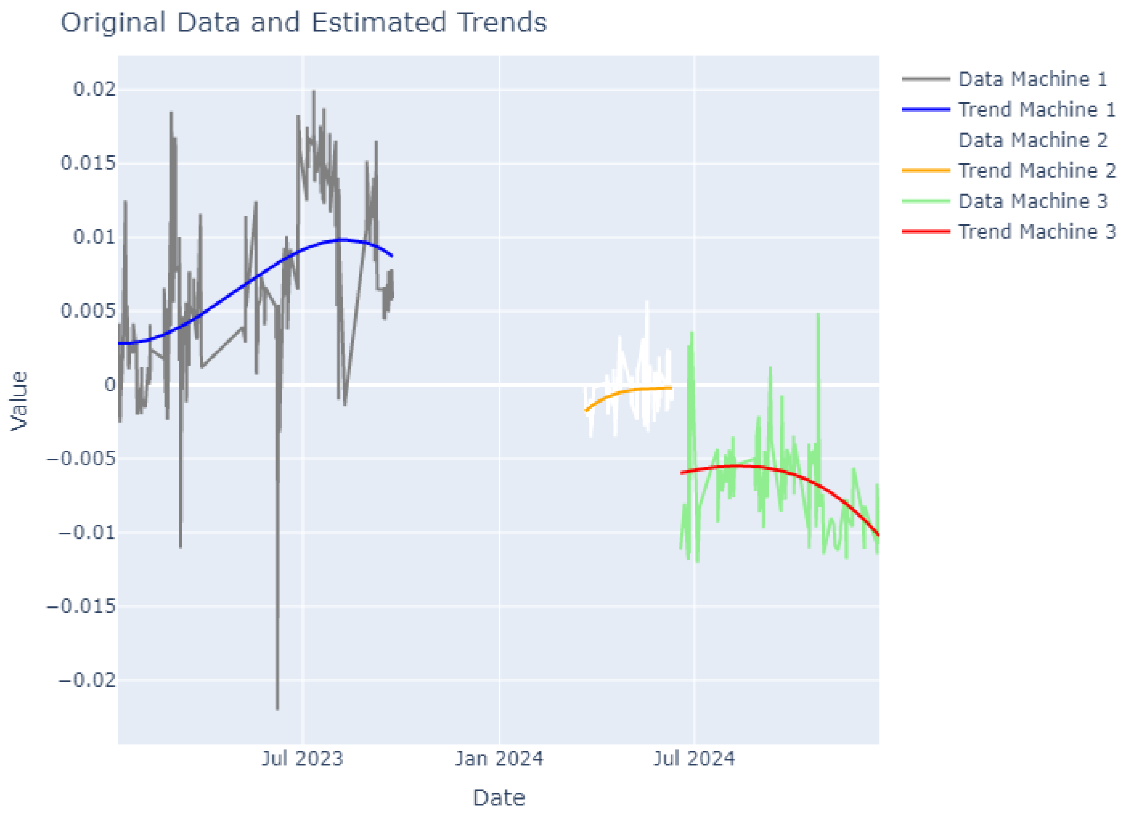
In fact, seasonality in the data may result in certain periods showing a higher frequency of defective items compared to others. If such patterns are discovered, identifying the root cause could help in defining corrective actions.

To investigate if any seasonality occurs in our data, it is first necessary to remove trends in the data, a process known as detrending. In this analysis, trends were extracted separately for each machining tool, using a third-degree polynomial model. This degree was chosen after comparing different polynomial fits (from linear up to cubic) and evaluating their root mean square error (RMSE). Cubic and linear models performed similarly, but the third-degree polynomial allowed for greater flexibility, still avoiding overfitting issues.

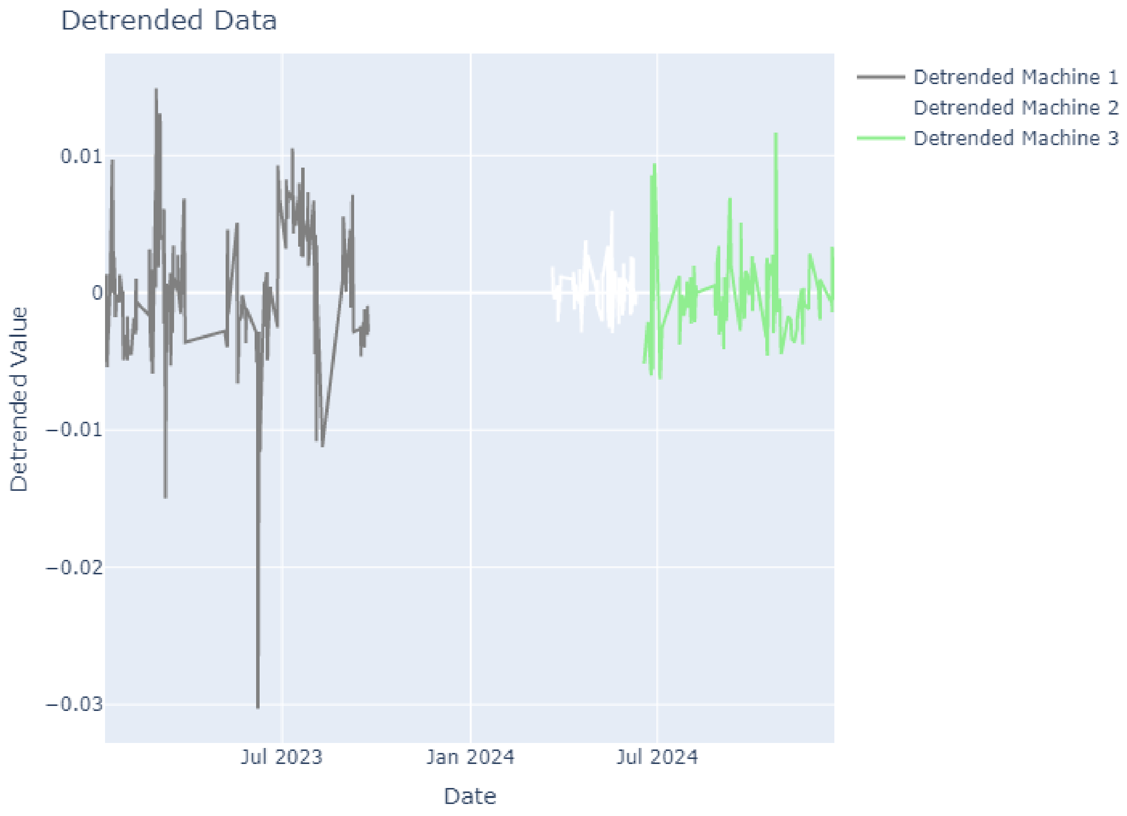
The following Python function was used to estimate the optimal trend for each machine and to compute the detrended datasets :

```python
# Function to find best trend
def estimate_trend(x, y, max_degree):
    # Convert datetime to numeric (in days)
    x_numeric = (x - x.iloc[0]).dt.total_seconds() / (60 * 60 * 24)
    best_rmse = float('inf')  # Initialize best RMSE with infinity
    best_trend = None  # Initialize best trend as None

    for deg in range(1, max_degree + 1):
        coeffs = np.polyfit(x_numeric, y, deg=deg)  # Fit polynomial of given degree
        trend = np.polyval(coeffs, x_numeric)  # Evaluate polynomial
        rmse = np.sqrt(mean_squared_error(y, trend))  # Calculate RMSE
        if rmse < best_rmse:
            best_rmse = rmse  # Update best RMSE
            best_trend = trend  # Update best trend

    return best_trend

# Define date boundaries
change1 = pd.to_datetime('2023-09-25')  # First change date
year_24 = pd.to_datetime('2024-01-01')  # Start of year 2024
change2 = pd.to_datetime('2024-06-11')  # Second change date
assessed_change2 = pd.to_datetime('2024-06-18')  # Assessed change date

# Split dataset into three parts based on date ranges

# Data before change1
df_machine1 = qc_rows[qc_rows['date_c'] < change1].copy()
# Data between year_24 and change2
df_machine2 = qc_rows[(qc_rows['date_c'] > year_24) &
              (qc_rows['date_c'] < change2)].copy()
```

```
# Data after assessed_change2
df_machine3 = qc_rows[qc_rows['date_c'] > assessed_change2].copy()

# Ensure date column is in datetime format
for df in [df_machine1, df_machine2, df_machine3]:
    df['date_c'] = pd.to_datetime(df['date_c'])  # Convert to datetime

# Estimate polynomial trends for each machine dataset
max_degree = 3  # Maximum polynomial degree
trend_m1 = estimate_trend(df_machine1['date_c'], df_machine1['deviation'], max_degree)
trend_m2 = estimate_trend(df_machine2['date_c'], df_machine2['deviation'], max_degree)
trend_m3 = estimate_trend(df_machine3['date_c'], df_machine3['deviation'], max_degree)

# Remove trend from the data to get detrended values
detrended_m1 = df_machine1['deviation'] - trend_m1  # Detrended machine 1 data
detrended_m2 = df_machine2['deviation'] - trend_m2  # Detrended machine 2 data
detrended_m3 = df_machine3['deviation'] - trend_m3  # Detrended machine 3 data

# Create plot for original data with trends
fig_trends = go.Figure()

fig_trends.add_trace(go.Scatter(x=df_machine1['date_c'], y=df_machine1['deviation'],
                                mode='lines', name='Data Machine 1',
                                line=dict(color='gray')))
fig_trends.add_trace(go.Scatter(x=df_machine1['date_c'], y=trend_m1,
                                mode='lines', name='Trend Machine 1',
                                line=dict(color='blue')))

fig_trends.add_trace(go.Scatter(x=df_machine2['date_c'], y=df_machine2['deviation'],
                                mode='lines', name='Data Machine 2',
                                line=dict(color='white')))
fig_trends.add_trace(go.Scatter(x=df_machine2['date_c'], y=trend_m2,
                                mode='lines', name='Trend Machine 2',
                                line=dict(color='orange')))

fig_trends.add_trace(go.Scatter(x=df_machine3['date_c'], y=df_machine3['deviation'],
                                 mode='lines', name='Data Machine 3',
                                 line=dict(color='lightgreen')))
fig_trends.add_trace(go.Scatter(x=df_machine3['date_c'], y=trend_m3,
                                 mode='lines', name='Trend Machine 3',
                                 line=dict(color='red')))

fig_trends.update_layout(
    title='Original Data and Estimated Trends',
    xaxis_title='Date',
    yaxis_title='Value')

# Save as PNG for LaTeX/PDF output
pio.write_image(fig_trends, "figures/trend-plot.png", format="png")

# Create plot for detrended data
fig_detrended = go.Figure()

fig_detrended.add_trace(go.Scatter(x=df_machine1['date_c'], y=detrended_m1,
                                    mode='lines', name='Detrended Machine 1',
                                    line=dict(color='gray')))
fig_detrended.add_trace(go.Scatter(x=df_machine2['date_c'], y=detrended_m2,
                                    mode='lines', name='Detrended Machine 2',
                                    line=dict(color='white')))
fig_detrended.add_trace(go.Scatter(x=df_machine3['date_c'], y=detrended_m3,
                                    mode='lines', name='Detrended Machine 3',
                                    line=dict(color='lightgreen')))

fig_detrended.update_layout(
    title='Detrended Data',
    xaxis_title='Date',
    yaxis_title='Detrended Value')

# Save as PNG for LaTeX/PDF output
pio.write_image(fig_detrended, "figures/detrend-plot.png", format="png")
```



Original Data and Estimated Trends
Value
Date
0.02
0.015
0.01
0.005
0
−0.005
−0.01
−0.015
−0.02
Jul 2023
Jan 2024
Jul 2024
Data Machine 1
Trend Machine 1
Data Machine 2
Trend Machine 2
Data Machine 3
Trend Machine 3



Detrended Data
Detrended Value
Date
0.01
0
−0.01
−0.02
−0.03
Jul 2023
Jan 2024
Jul 2024
Detrended Machine 1
Detrended Machine 2
Detrended Machine 3

After estimating the trend, I was able to see that the resulting curves gave some meaningful information. For example, looking at Machine 1, the process seemed to be well-centered at the beginning, but then gradually, the values started moving upward. This might indicate that the machine is drifting slightly from the target over time. Even though identifying this wasn't the main objective of the analysis, noticing such trends can actually be useful, especially for planning maintenance activities or for improving the stability of the process in the long term.

Once the trend was removed, I examined the remaining signal, the so-called residuals, to check whether there might be any kind of periodic pattern. However, since the data points were not collected at regular time intervals, the standard Discrete Fourier Transform couldn't really be used here. That's why I decided to use the Lomb-Scargle periodogram instead. This method is more suitable when time series are unevenly spaced.

Basically, the Lomb-Scargle approach is a kind of spectral analysis method that works using least squares, and it's designed for data with irregular time steps. It assumes the signal can be approximated as a combination of sine and cosine waves, but fitted to the actual time points rather than assuming equal spacing. This leads to a model where the signal is represented by a sum of sinusoidal terms, as in the equation below:

$$y(t) \approx \sum_k \left[A_k \cos(\omega_k t) + B_k \sin(\omega_k t)\right]$$

To test the presence of a specific frequency ω, we isolate a single term of the sum and fit the model:

$$y(t) = A\cos(\omega t) + B\sin(\omega t)$$

The coefficients A and B are obtained by minimizing the squared residuals between this model and the data (t_i, y_i), i.e., using a least-squares approach. However, for irregularly spaced observations, the sampled sine and cosine terms are not generally orthogonal, which can bias the estimation.

To resolve this, a time shift τ is introduced such that the basis functions become orthogonal over the discrete sampling. The model becomes:

$$y(t) = A\cos\left[\omega(t-\tau)\right] + B\sin\left[\omega(t-\tau)\right]$$

The value of τ is chosen so that the cross term between sine and cosine vanishes, which leads to the condition:

$$\tan(2\omega\tau) = \frac{\sum_i \sin(2\omega t_i)}{\sum_i \cos(2\omega t_i)}$$

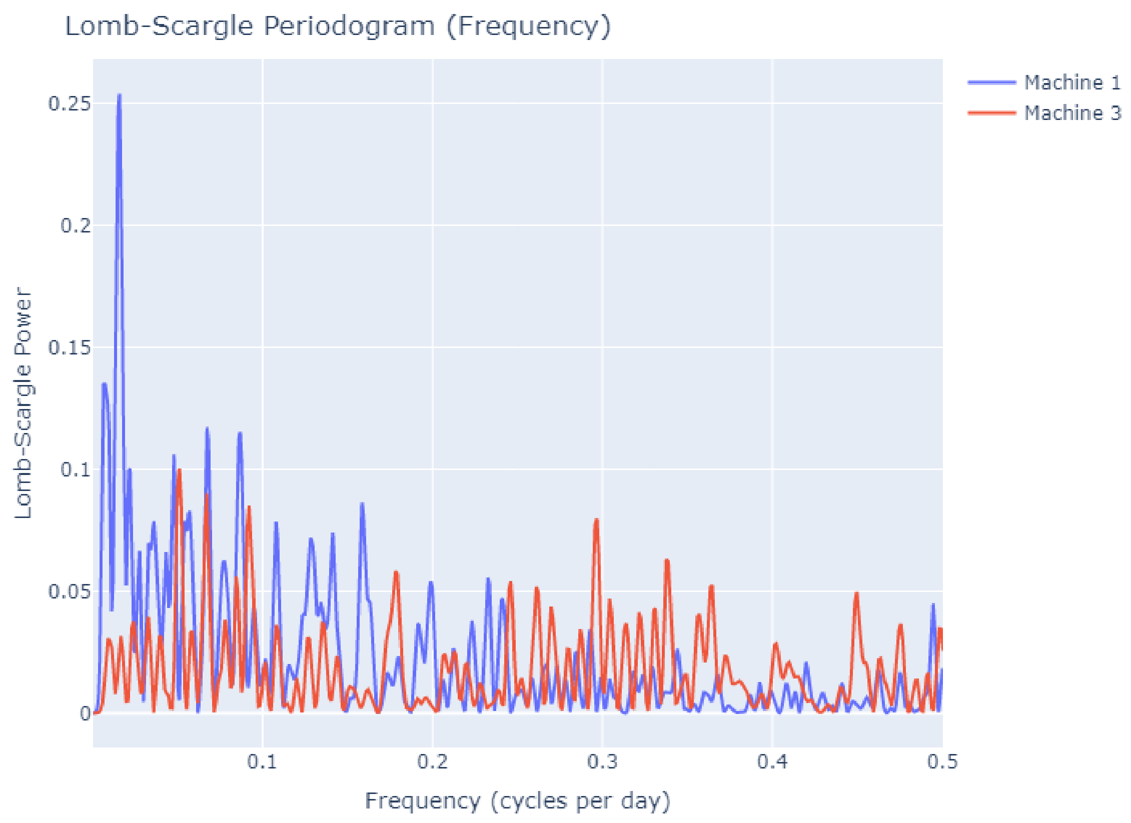
With this orthogonal basis, the Lomb-Scargle power at a given angular frequency ω is defined as:

$$P(\omega) = \frac{1}{2\sigma^2}\left[\frac{\left(\sum_i y_i \cos\left(\omega(t_i-\tau)\right)\right)^2}{\sum_i \cos^2\left(\omega(t_i-\tau)\right)} + \frac{\left(\sum_i y_i \sin\left(\omega(t_i-\tau)\right)\right)^2}{\sum_i \sin^2\left(\omega(t_i-\tau)\right)}\right]$$

Here, σ^2 is the variance of the observed data and serves to normalize the power spectrum. This formulation enables the detection of periodic components in unevenly sampled datasets by computing, for each frequency, the squared projection of the data onto an orthogonal sinusoidal basis centered around that frequency.

The Python implementation used in this project is shown below:

```python
# Function to perform the Lomb-Scargle periodogram
def lomb_scargle(df, dates, detrended_measurements):
    # Extract time values and convert to numerical format
    # (number of hours since first timestamp)
    time = df[dates].values
    time = (time - time[0]).astype('timedelta64[h]').astype(int)  # Time in hours

    # Compute Lomb-Scargle power spectrum
    # Maximum frequency set to 0.5 cycles per day due to Nyquist limit
    # because in some days, only one measurement is available,
    # so we cannot detect cycles shorter than 2 days
    frequency, power = LombScargle(time, detrended_measurements).autopower(
                       maximum_frequency=0.5/24)

    return frequency, power

# Compute Lomb-Scargle for the machine 1 dataset
freq_m1, power_m1 = lomb_scargle(df_machine1, 'date_c', detrended_m1)

# Compute Lomb-Scargle for the machine 3 dataset
freq_m3, power_m3 = lomb_scargle(df_machine3, 'date_c', detrended_m3)

# Create plot to show the power spectrum in the frequency domain
fig = go.Figure()

fig.add_trace(go.Scatter(
    x=24 * freq_m1,  # Convert frequency to cycles per day
    y=power_m1,
    mode='lines',
    name='Machine 1'))

fig.add_trace(go.Scatter(
    x=24 * freq_m3,
    y=power_m3,
    mode='lines',
    name='Machine 3'))

fig.update_layout(
    title='Lomb-Scargle Periodogram (Frequency)',
    xaxis_title='Frequency (cycles per day)',
    yaxis_title='Lomb-Scargle Power')


# Save as PNG for LaTeX/PDF output
pio.write_image(fig, "figures/lst-plot.png", format="png")
```



This function was applied to the first and third machining tool datasets. The second interval was excluded due to insufficient data points. The result revealed that for Machine 3, no clear peaks were observed, suggesting no dominant periodic component, while, for Machine 1, a visible peak appeared at a frequency of approximately $f = 0.016$ cycles/day, corresponding to a period $T = 1/f \approx 62$ days. This could potentially indicate a two-month cycle, although the presence of smaller side peaks suggests that this may not be a strongly consistent periodic signal.

The identification of such a frequency during real time production monitoring would warrant further investigation. Understanding the source of periodic variations could lead to improvements in process stability. However, given the lack of clearly noisy frequencies, especially in the third interval, no filtering was applied to the data before computing the Cpk index.

8.3 Cpk Representation in Time

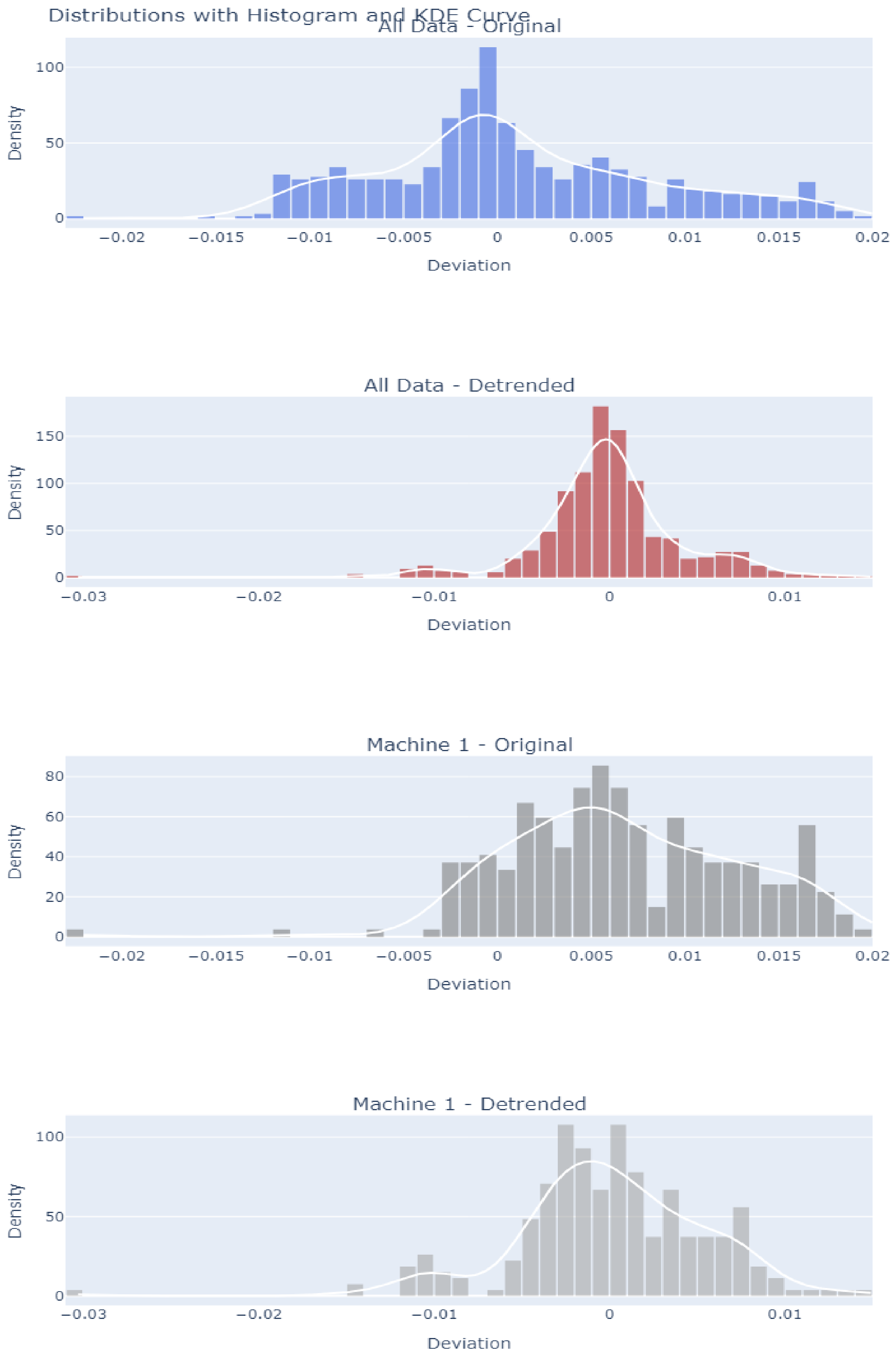
Since the calculation of the Cpk index relies on the assumption that the data follow a normal distribution, the first step in this phase of the analysis was to visually inspect the shape of the distributions. This was done with two main goals: to see if the data appeared approximately normal, and to evaluate whether removing the trend improved their adherence to a Gaussian distribution. Company already assumes normality of the production data, unless specific exceptions (previously discussed in subsection 5.3). Starting from this premise, if detrending brings the distribution closer to a normal shape, it would support the hypothesis that the trend observed in the raw data is caused by gradual changes in the behavior of the machining tool.

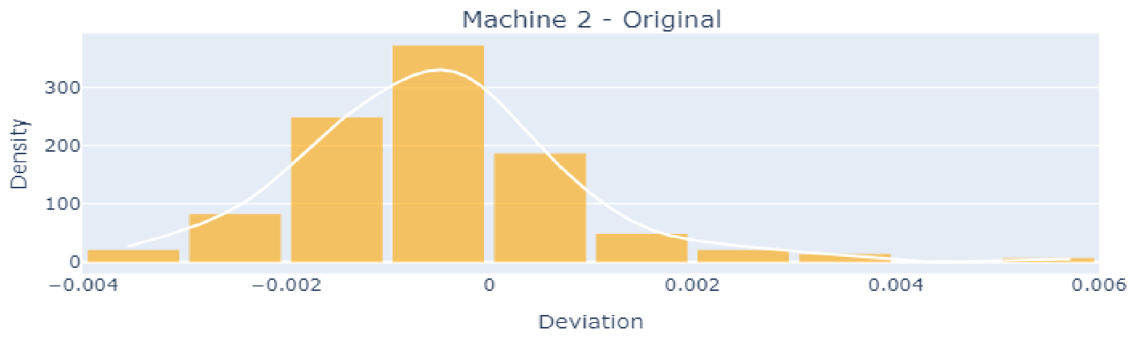
To conduct this analysis, the data were once again divided into the time intervals of the machines lifecyles, and also aggregated to capture general distribution. For each case, two histograms with corresponding kernel density estimates (KDE) were plotted : one before and one after detrending.

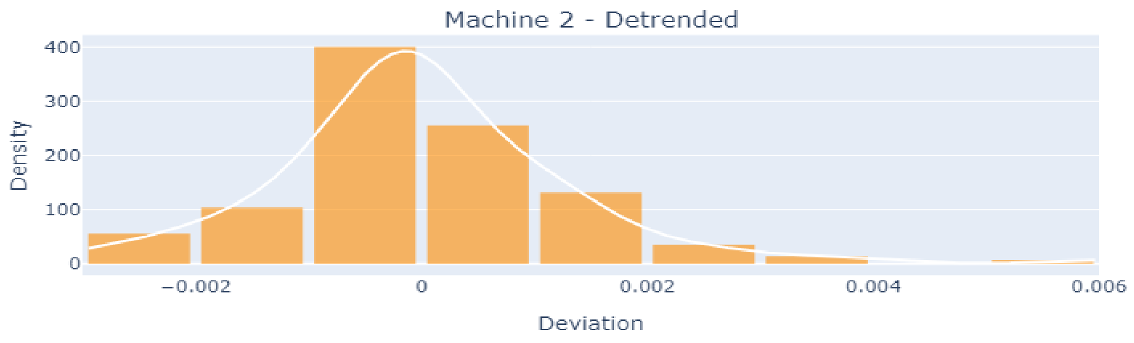
The following Python script was used to generate the plots:

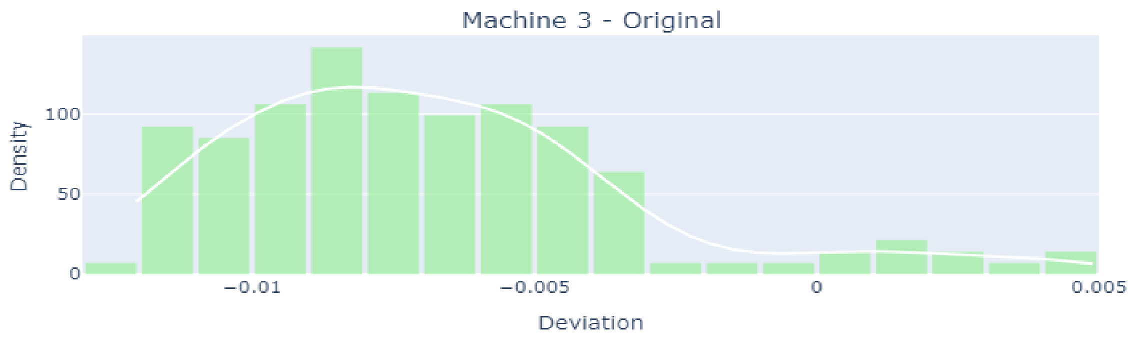
```python
# Function to calculate KDE (Kernel Density Estimation)
def get_kde(x_vals, data):
    kde = gaussian_kde(data)  # Fit a kernel density estimator to the data
    return kde(x_vals)  # Evaluate KDE at the points x_vals

all_original = qc_rows['deviation']
all_detrended = pd.concat([detrended_m1, detrended_m2, detrended_m3],
                ignore_index=True)

# List of datasets with names and colors for each plot
datasets = [(all_original, "All Data - Original", 'royalblue'),
    (all_detrended, "All Data - Detrended", 'firebrick'),
    (df_machine1['deviation'], "Machine 1 - Original", 'gray'),
    (detrended_m1, "Machine 1 - Detrended", 'darkgray'),
    (df_machine2['deviation'], "Machine 2 - Original", 'orange'),
    (detrended_m2, "Machine 2 - Detrended", 'darkorange'),
    (df_machine3['deviation'], "Machine 3 - Original", 'lightgreen'),
    (detrended_m3, "Machine 3 - Detrended", 'green')]

# Create subplots layout
fig = make_subplots(rows=8, cols=1,
    subplot_titles=[name for _, name, _ in datasets])

# Add histogram and KDE for each dataset
for i, (data, title, color) in enumerate(datasets):
    row = i + 1  # Determine which row to plot the data on

    # Calculate KDE
    # Create a range of x values for the KDE
    x_vals = np.linspace(data.min(), data.max(), 500)
    kde_vals = get_kde(x_vals, data)  # Get the KDE values for the given data

    # Add histogram to the plot
    fig.add_trace(go.Histogram(x=data,
        histnorm='probability density',
        name='Histogram',
        marker_color=color,
        opacity=0.6,
        xbins=dict(size=0.001),  # Bin size
        showlegend=False),
        row=row, col=1)

    # Add KDE curve to the plot
    fig.add_trace(go.Scatter(
        x=x_vals,
        y=kde_vals,
        mode='lines',
        line=dict(color='white', width=2),
        name='KDE',
        showlegend=False),
        row=row, col=1)

# Global layout settings
```

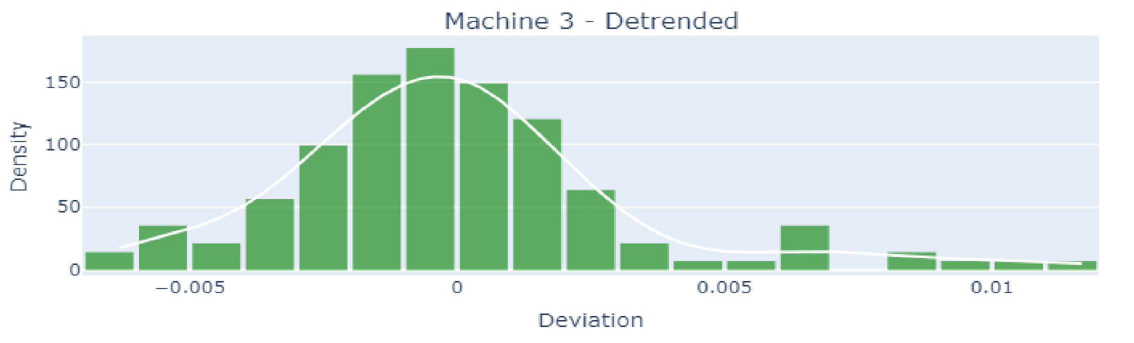
```
fig.update_layout(height=2600,
    title="Distributions with Histogram and KDE Curve",
    bargap=0.1)  # Gap between bars in the histogram



# Set axis labels for each subplot
for r in range(1, 9):
    fig.update_yaxes(title_text="Density", row=r, col=1)
    fig.update_xaxes(title_text="Deviation", row=r, col=1)

# Save as PNG for LaTeX/PDF output
pio.write_image(fig, "figures/norm-plot.png", format="png")
```


Distributions with Histogram and KDE Curve
All Data - Original
Density
Deviation
All Data - Detrended
Density
Deviation
Machine 1 - Original
Density
Deviation
Machine 1 - Detrended
Density
Deviation


Machine 2 - Original
Density
300
200
100
0
−0.004
−0.002
0
0.002
0.004
0.006
Deviation


Machine 2 - Detrended
Density
400
300
200
100
0
−0.002
0
0.002
0.004
0.006
Deviation


Machine 3 - Original
Density
100
50
0
−0.01
−0.005
0
0.005
Deviation


Machine 3 - Detrended
Density
150
100
50
0
−0.005
0
0.005
0.01
Deviation

It is important to note that the y-axis does not represent the absolute frequency of the measurements, but rather the probability density. This is because the histogram has been normalized using the probability density setting, which ensures that the area under the histogram sums to 1. This allows for a meaningful comparison with the KDE curve.

The results clearly show that, in all four cases (three intervals and the full dataset), the data appear more normally distributed after the trend is removed. This strongly confirm the presence of the trends in the production processes.

We can see that none of the distributions are perfectly Gaussian, particularly in the first time interval. In that case, a suspicious measurement strongly influences the trend estimation. However, we didn't consider it as an outlier since the Mahalanobis distance was not as great as the ones of the change dates. That is because the process at that time was simply operating with higher variance. Anyway, the presence of this data point, likely caused the estimated trend line to shift downward compared to its actual position. As a result, even after detrending, the distribution remains skewed toward the right.

Despite these imperfections, it is important to consider the whole context: the datasets are small, derived from different production lots, and span a long time period that includes many process changes and events. Under these conditions, it is reasonable to accept that the distributions are not perfectly normal, and the assumption of normality can still be considered valid for the purposes of Cpk calculation.

Once this assumption has been validated, the next challenge was to determine how to compute the Cpk index in a meaningful way, given the temporal distance of the data points. Standard Cpk estimation requires at least 30 data points to provide a stable estimate. In our computation, consider 30 data points would lead to a very high difference between the first and the last production dates, while computing a weekly Cpk, for example, would lead to have just few data points, not enough to compute the index. I needed to find a compromise and I consequently choose to compute the daily Cpk using every time the data of the last month, in which the production days are 20 or little more. This corresponds to have about 20/25 measurements each time, and I imposed that 20 should be the minimum number accepted with a filter.

However, applying a simple rolling calculation across the last 30 days measurements, would ignore the fact that these data points originate from batches spread across a wide time window. It seemed more reasonable to assign a weight to each measurement based on its recency, in order to obtain a more accurate estimate of the process capability at any given day.

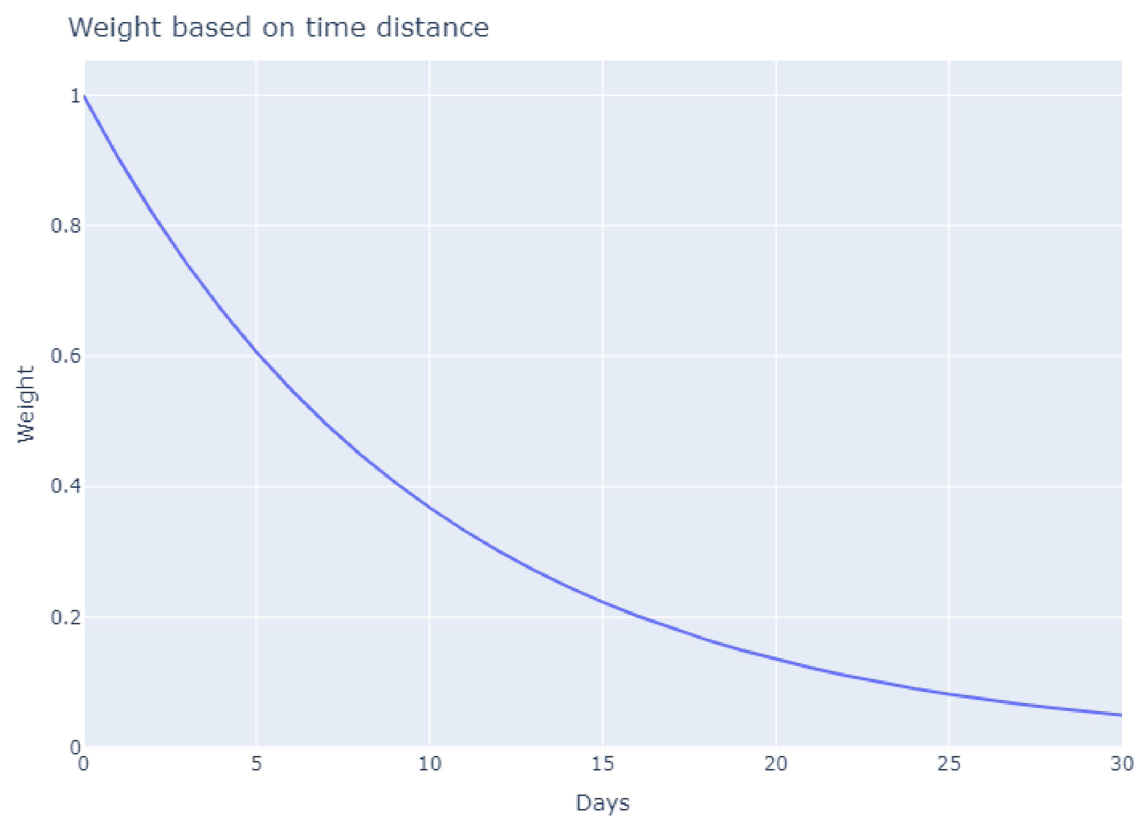
Among the possible weighting strategies, I choose an exponential decay function :

$$w(t) = e^{-At}$$

where t is the time elapsed in days from the measurement to the current day, and A is a decay parameter. The value of A was chosen arbitrarily in this analysis, but the optimal one can be find by analyzing how the Cpk truly varies from one production lot to another. For instance, I choose $A = 0.1$, which results in an exponential daily drop of around 9.52% in the weight assigned to past measurements.

The function and a plot of the weight curve are provided below to illustrate how this approach works:

```
# Function to assign weights
def weight(time_distance, A):
    # Calculate the weight using an exponential decay function
    # A represents the decay rate
    time_distance = np.array(time_distance)  # Convert input to NumPy array
    return np.exp(-A * time_distance)  # Exponential decay weighting

# Create a list representing time distances from 0 to 30
dists = [i for i in range(31)]

# Apply the weight function to generate the plot
weights = weight(dists, 0.1)

# Plot weights vs time distance
fig = go.Figure(data=go.Scatter(x=dists, y=weights))

fig.update_layout(title='Weight based on time distance',  # Plot title
        xaxis_title='Days',
        yaxis_title='Weight')

# Save as PNG for LaTeX/PDF output
pio.write_image(fig, "figures/w-plot.png", format="png")
```



At this point, the final tasks consist of calculating the Cpk index for each day of production using the weighting method previously described, and then analyzing how this index changes over time. While the computation of the mean is relatively easy to assign a weight to each data point, it is much more difficult to find a way to correctly compute the variance in the same scenario, particularly in some situations in which a bias can occur. For example, if we only have a data point corresponding to the date of the production day, and all the other data points which are considered to estimate the Cpk are far away (in days), the mean would result very similar to the value of the measurement taken on the production day, which is the best that we can do with our data. The variance instead, if singular variances are computed in the same way, would approach 0 because of the fact that the only relevant measurement is also the one which is very similar to the mean. For that reason, I decided to use the Bessel correction to improve the estimation of the variance:

$$\text{Var}_{\text{weighted}} = \frac{\sum_i w_i (x_i - \mu)^2}{\sum_i w_i - \frac{(\sum_i w_i^2)}{\sum_i w_i}}$$

This correction accounts for the loss of degrees of freedom due to weighting, leading to an unbiased estimate of the population variance.

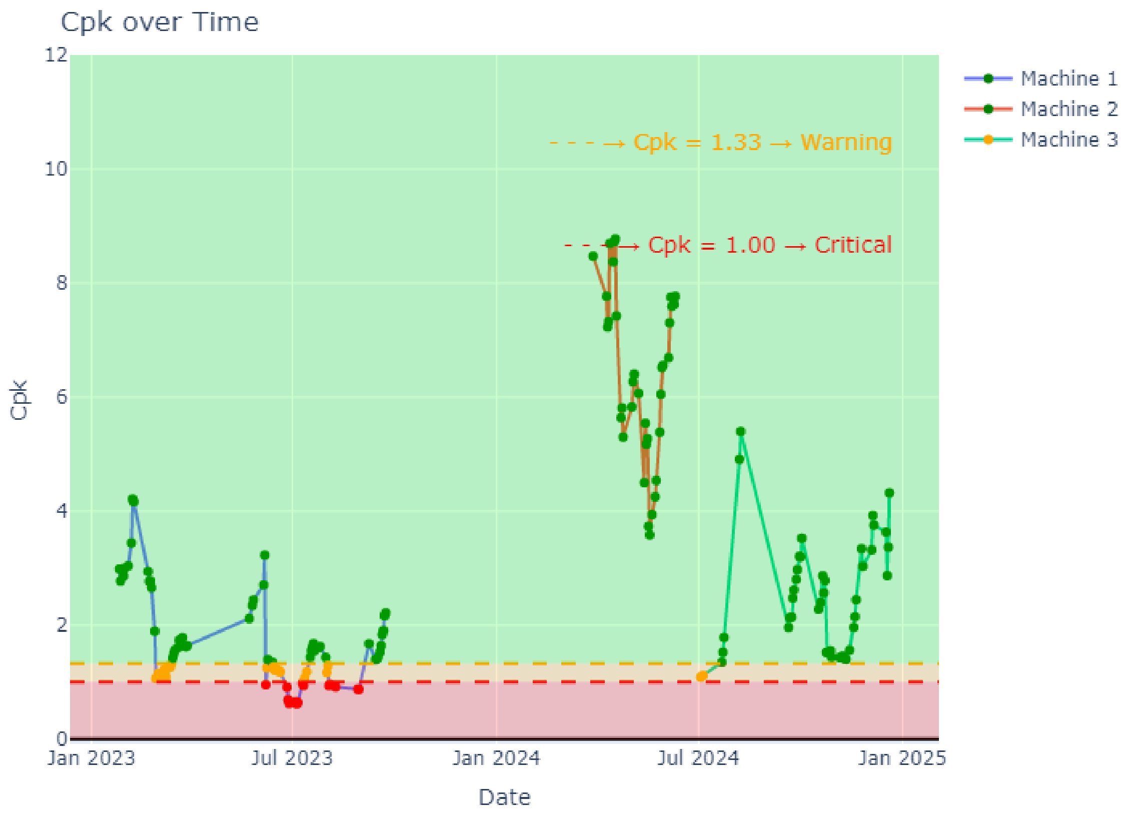
These two scripts were created, the first one to compute the index and the second one to visualize the results :

```python
window_size = 30  # Rolling time window size (in days)
min_measurements = 20  # Minimum number of measurements required to calculate Cpk
expected_value = qc_rows['nominal'].iloc[0]  # Extract the expected nominal value
max_value = expected_value + UT  # Compute the upper tolerance limit
min_value = expected_value + LT  # Compute the lower tolerance limit
A = 0.1  # Decay factor used in the weighting function

all_data = [df_machine1, df_machine2, df_machine3]
for df in all_data:
    # Add a column with only the date (no time) from datetime
    df['day'] = df['date_c'].dt.date  # Extract date component
    # Flag the last measurement of each production day
    df['last_measurement'] = df['day'] != df['day'].shift(-1)
    # Initialize the Cpk column with NaN values
    df['cpk'] = np.nan  # Prepare column for Cpk results
    # Count how many production days are present in the dataset
    production_days = df['last_measurement'].sum()
    # Filter only the last measurement entries for each day
    last_measurements = df.loc[df['last_measurement'] == True]  # Daily endpoints

    # Loop through each production day to compute Cpk
    for i in range(production_days):
        # Select the current day's last measurement
        production_day = last_measurements.iloc[i]
        date = production_day['day']  # Extract the date

        # Define the time window start (rolling back in time)
        window_start = date - pd.Timedelta(days=window_size - 1)
        # Filter the dataset to include only data within the rolling window
        window = df[(df['day'] >= window_start) & (df['day'] <= date)]
        total_measurements = len(window)  # Number of measurements in the window

        # Check if the window contains enough data to calculate Cpk
        if total_measurements < min_measurements:
            df.loc[last_measurements.index[i], 'cpk'] = np.nan
        else:
            # Initialize the list of weights
            alfa = []
            for j in range(len(window)):
                td = (date - window['day'].iloc[j]).days
                alfa.append(weight(td, A))

            # Normalize weights
            norm_factor = sum(alfa)
            normalized_alfa = [a / norm_factor for a in alfa]

            # Weighted mean
            mean = sum(window['actual'].iloc[k] * normalized_alfa[k]
                   for k in range(len(window)))

            # Weighted variance with Bessel correction
            squared_diffs = [(window['actual'].iloc[k] - mean) ** 2
                             for k in range(len(window))]
            num = sum(normalized_alfa[k] * squared_diffs[k]
                  for k in range(len(window)))
            den = 1 - sum(w ** 2 for w in normalized_alfa)
            var = num / den if den > 0 else 0  # Avoid division by 0
            sigma = np.sqrt(var)
            # Calculate Cpk for both upper and lower limits
            cpk_upper = (max_value - mean) / (3 * sigma)  # Upper process capability
            cpk_lower = (mean - min_value) / (3 * sigma)  # Lower process capability
            cpk = min(cpk_upper, cpk_lower)  # Take the worst-case (minimum) value

            # Assign the Cpk value to the corresponding row in the DataFrame
            df.loc[last_measurements.index[i], 'cpk'] = cpk
```

```
# Define Cpk thresholds for warning and critical levels
cpk_threshold_warning = 1.33  # Warning threshold for Cpk
cpk_threshold_critical = 1.00  # Critical threshold for Cpk

# Create a figure to visualize the Cpk values over time
fig = go.Figure()

# Iterate through the datasets for each machine and generate the plot
for idx, df in enumerate(all_data):
    df_valid = df.dropna(subset=['cpk'])  # Remove rows where Cpk is NaN

    # List to store colors for markers based on Cpk values
    colors = []
    for cpk in df_valid['cpk']:
        if cpk < cpk_threshold_critical:
            colors.append('red')
        elif cpk < cpk_threshold_warning:
            colors.append('orange')
        else:
            colors.append('green')

    # Add a trace to the figure using the date and Cpk values
    fig.add_trace(go.Scatter(x=df_valid['day'], y=df_valid['cpk'],
                             mode='lines+markers',
                             marker=dict(color=colors, size=6),
                             line=dict(width=2, dash='solid'),
                             name=f'Machine {idx+1}'))
```

```
# Add horizontal lines to indicate the critical and warning thresholds
fig.add_hline(y=cpk_threshold_critical, line_dash='dash', line_color='red')
fig.add_hline(y=cpk_threshold_warning, line_dash='dash', line_color='orange')

# Add rectangles to visually highlight the critical, warning, and normal regions
fig.add_hrect(y0=cpk_threshold_critical, y1=cpk_threshold_warning,
              fillcolor='rgba(255, 165, 0, 0.2)', line_width=0)
fig.add_hrect(y0=0, y1=cpk_threshold_critical, fillcolor='rgba(255, 0, 0, 0.2)',
              line_width=0)
fig.add_hrect(y0=cpk_threshold_warning, y1=12, fillcolor='rgba(0, 255, 0, 0.2)',
              line_width=0)

fig.update_layout(title='Cpk over Time',
                  xaxis_title='Date',
                  yaxis_title='Cpk',
                  xaxis=dict(showgrid=True),
                  yaxis=dict(showgrid=True, zeroline=True, zerolinecolor='black'))

# Add annotations to explain the threshold lines
fig.add_annotation(
    text='- - - → Cpk = 1.33 → Warning',
    xref='paper', yref='paper',
    x=0.95, y=0.9,
    showarrow=False,
    font=dict(size=14, color='orange'),
    borderwidth=1)

fig.add_annotation(
    text='- - - → Cpk = 1.00 → Critical',
    xref='paper', yref='paper',
    x=0.95, y=0.75,
    showarrow=False,
    font=dict(size=14, color='red'),
    borderwidth=1)

# Save as PNG for LaTeX/PDF output
```

```
pio.write_image(fig, "figures/cpk-plot.png", format="png")
```

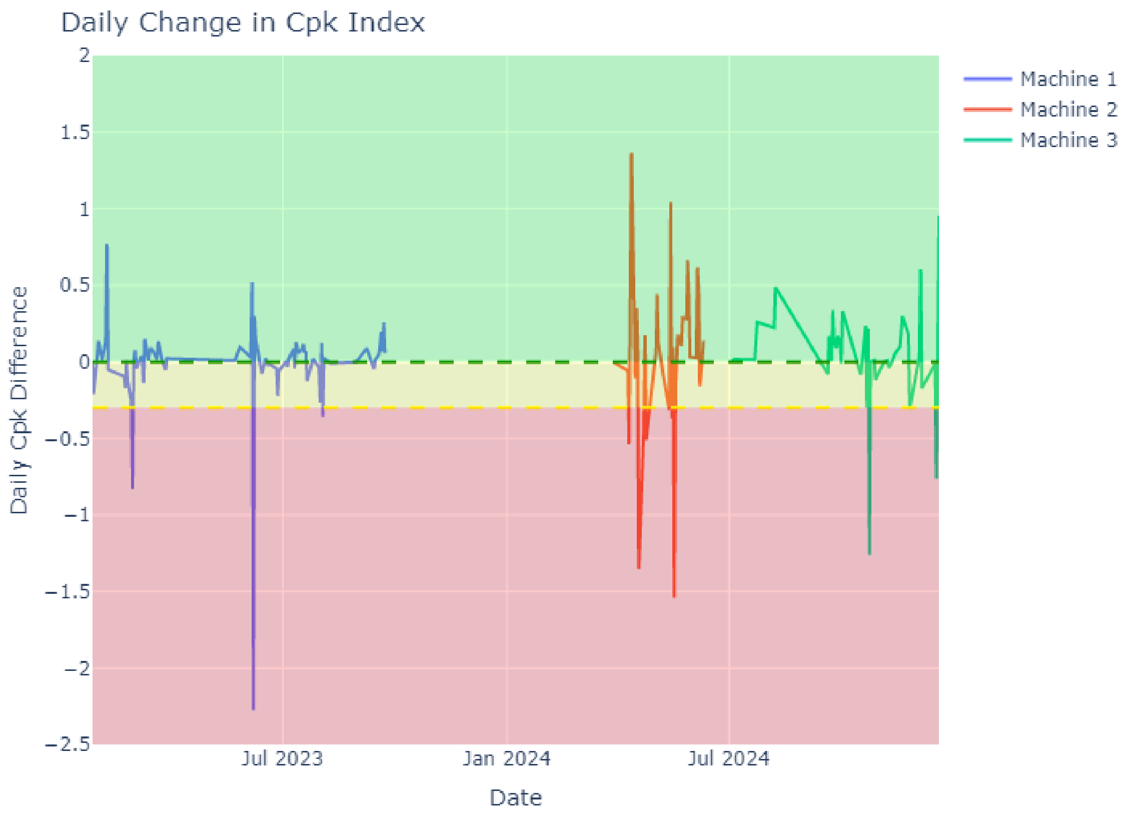


In the graph, we can see how the Cpk index changes over time for each of the three machines. This gives us a general idea of how the capability of each machine has evolved during the production period.

However, just looking at the index itself, is not easy to understand how quickly things are improving or getting worse. For this reason, I also calculated how much the Cpk changes from one day to the next, actually computing a sort of derivative. This allows us to better understand how a machine loses or gains capability over time.

```
# Define thresholds for good and warning levels
good = 0
warning = -0.3


# Create the figure for daily Cpk difference
fig_daily_diff = go.Figure()
```

```python
# Iterate through the dataset, calculating the daily Cpk change
for idx, df in enumerate(all_data):
    df_valid = df.dropna(subset=['cpk'])
    daily_slope = [0]  # Start with 0 for the first point

    for i in range(1, len(df_valid)):
        cpk_diff = df_valid['cpk'].iloc[i] - df_valid['cpk'].iloc[i-1]
        days_diff = (df_valid['day'].iloc[i] - df_valid['day'].iloc[i-1]).days
        daily_slope.append(cpk_diff / days_diff if days_diff != 0 else 0)

    # Add the trace for daily Cpk change
    fig_daily_diff.add_trace(go.Scatter(
        x=df_valid['day'],
        y=daily_slope,
        mode='lines',
        name=f'Machine {idx+1}'))

# Add threshold lines
fig_daily_diff.add_hline(y=good, line=dict(color='green', dash='dash'))
fig_daily_diff.add_hline(y=warning, line=dict(color='yellow', dash='dash'))

# Add highlighted regions
fig_daily_diff.add_hrect(y0=good, y1=2, fillcolor='rgba(0, 255, 0, 0.2)', line_width=0)
fig_daily_diff.add_hrect(y0=warning, y1=good, fillcolor='rgba(255, 255, 0, 0.2)',
                         line_width=0)
fig_daily_diff.add_hrect(y0=-2.5, y1=warning, fillcolor='rgba(255, 0, 0, 0.2)',
                         line_width=0)

fig_daily_diff.update_layout(
    title='Daily Change in Cpk Index',
    xaxis_title='Date',
    yaxis_title='Daily Cpk Difference')

# Save as PNG for LaTeX/PDF output
pio.write_image(fig_daily_diff, "figures/der-plot.png", format="png")
```



From both graphs, the Cpk evolution and the daily Cpk difference, it is possible to extract several meaningful insights. These findings represent exactly the kind of conclusions I aimed to reach through this analysis and all of these aspects will be discussed in detail in the next and final chapter.

9 Discussion and Conclusions

9.1 Discussion of Results

Although already introduced at the beginning of this thesis, it is important to briefly recall the main goals of the work. The objective was to explore the possible benefits of continuously monitoring the Cpk index of a machining tool, instead of calculating it only during standardization phases and then replacing the machine after a fixed number of cycles just because, on average, we know that the capacity tends to decrease in that time.

To start this discussion, I would like to focus first on the two previous plots: the evolution of the Cpk over time, and its "derivative", if we can call it that.

From the first graph, what we can see quite clearly is that there is not a precise or regular trend that describes the Cpk over time. It seems to follow more of a random pattern. This could already be a strong indication that trying to predict when the Cpk will reach a certain threshold is not really practical. In this sense, having a system that keeps the situation under control all the time, and that allows us to constantly monitor what is going on, could be much more useful than just making estimates based on expectations or historical data.

Unfortunately, we do not have complete datasets for any of the machines. The first dataset starts already in the middle of the machine's life cycle, the second one has a large gap with missing data, and the third one only covers the initial period. So, it is not really possible to observe a full picture of how the machine capacity evolves throughout its entire usage.

Even so, some interesting patterns can still be noticed. For example, the first machine seems to have been replaced during a time when the process had indeed lost capacity, and the Cpk dropped below 1 for at least three time periods. In that case, the decision to replace the machine appears justified.

The second machine, instead, is a clear example of where the typical approach to estimating machine life may fail. Assuming that the change from the second to the third machine actually happened in that moment, it would mean that the second machine was replaced after a period where the Cpk had never gone below 3.5 for at least three months. Even more surprising is that the last recorded Cpk before the change was 7.77, which is an extremely high value. A Cpk of 2 already corresponds to about 0.002 defects per million, so a Cpk of 7.77 basically means zero defects. And since the Cpk is exponentially related to the number of defects, it's easy to understand how such a high value shows excellent performance.

So, replacing a machine with such a high Cpk would likely not be a good idea. First, it is very unlikely that the new machine will perform better than the actual one, it could even perform worse, leading to more waste. Second, there are the costs of buying and installing a new machine, plus removing the old one. This could mean spending more without actually improving anything, maybe even making things worse. And this is in fact what seems to have happened in the shift from the second to the third machine.

Of course, it's possible that the assumption about the machine change is wrong, but even in that case, the usefulness of monitoring the Cpk doesn't lose its importance. If the drop in Cpk was caused by some other event, it would still be helpful to detect it early and try to understand the reason, like a shift in process centering, which can be seen in the graph on page 42. Then, corrective actions could be taken to solve the issue.

The other important point comes from the second graph, the one showing the 'derivative' of the Cpk. This part was also unexpected. One might have assumed that the machine capacity decreases gradually over time in a smooth way. But what we actually see in the graph is that the Cpk tends to change in "jumps". Especially for the first and third intervals, instead of seeing a negative and constant derivative, we find many moments where the derivative is close to zero, meaning the process stays stable for some time, and then suddenly there are big changes, positive or negative.

These positive jumps might be explained by maintenance operations, while the negative ones could come from unexpected events, maybe some material got stuck, or something like that. What's also interesting is that in some of these cases, there is no immediate recovery after the drop, which suggests that some events can affect the machine's capacity over the long term.

So, another important idea is that it could be very useful to track these kinds of events and link them to Cpk changes. This way, we can plan maintenance at the right moment and try to restore the previous performance level.

The case of the second machine is a bit different. The Cpk variation looks much more volatile, but that is actually because the values are on a much higher range. So, even a small change in the data can make the Cpk move a lot. For example, there's a much bigger difference between a Cpk going from 0.5 to 1.5 than from 7 to 8, and the second case could come from almost identical data. It's a matter of sensitivity.

Finally, for the third machine, the derivative is often positive, showing an improvement of capacity over time. Maybe this is due to a long period of the stabilization phase of the machine.

Compared to a traditional control plan that only looks at tolerance limits, this approach lets us see when things are going in the wrong direction, sometimes even when all the measurements are between the tolerance limits, while the control plan actually can not.

So, to summarize, what becomes clear from all of this is that continuously monitoring the Cpk can lead to a much better decision-making process. Basing decisions on real data, instead of assumptions or fixed schedules, allows for smarter choices and can help reduce waste and unnecessary costs, optimize maintenance operations and working machines replacements, and improve the overall quality.

9.2 Opportunities for Improving the Analysis

Most of the operations performed during this analysis were based on assumptions that are not completely accurate. The reason I continued the analysis in this way is mainly because the data, and in general the information available to me, was not really enough to carry out an "optimal" study. However, this level of information and data was still sufficient to reach the kind of conclusions that I presented in the previous chapter, that was the real goal of the thesis.

Clearly, if the objective were to actually monitor a real production process in an industrial context, it would be important to do it in a more optimized and precise way. For this reason, I would now like to list a few aspects that, although already mentioned in earlier chapters, were not explored deeply and could be developed better if more data or more information were available.

As already mentioned, it would be desirable to measure at least 30 pieces from the same production batch to compute the Cpk. For this reason, an optimal approach would consist of calculating the Cpk using 30 measurements for each lot and for each QC parameter of different components, and then aggregate the Cpks from different parameters and components that are produced using the same machining tool. This is because the Cpk index should not be strictly linked to a specific component's dimension or parameter, but rather to the performance of the machining tool itself. Since the same machine is often used to produce more than one type of component or parameter, it might be useful to group these different Cpks together. Doing so could even help reveal how the index reacts, for example, after a setup operation.

While this idea of grouping Cpks could be quite feasible with the right kind of informations, it is probably much less realistic to assume that 30 components will be measured for every batch. This is exactly why a method like the one used in this thesis becomes necessary. But even this method has space for improvement.

The first thing that should definitely be improved is the identification of machine change dates. In this work, I had to make assumptions because I didn't know the exact dates when the machines were replaced. But if the actual changeover moments were known, it would be much more accurate to "reset" the monitoring process each time a machine is replaced, and treat each new machine as the start of a new and independent process.

Another aspect that could be optimized is the weighting parameter used in the trend calculation. As already mentioned in the analysis, I chose this value arbitrarily. To find a better, more accurate value, it would be necessary to calculate the mean and variance (with 30 measurements per batch) for consecutive batches, observe how much they change from one batch to the next in terms of percentage, and repeat this over many examples to get an average. In this way, instead of assigning weights based on how many days ago the measurement was made, we could assign weights based on how many production batches ago it was recorded. This could lead to a more meaningful estimation of how each past measurement affects the current one.

Another factor that could improve the analysis is having more information about production events. For instance, if there was a production issue and all parts in one batch were measured (maybe to find the defective ones), and many were out of tolerance, that would create a cluster of data that could heavily skew the trend. This would make it much harder to detect any real seasonal effect or pattern. That's just one example, but in general, knowing when and why certain data points were recorded, and whether they should be included in the analysis and how, could help a lot to avoid mistakes and make the study more reliable.

There are also other cases that should be taken into account. For example, a component might be measured multiple times due to some unexpected event, or a new mold might require a full standardization even if the machining tool hasn't actually changed. This means that we could have 30 measurements that look like a "new start" in the Cpk monitoring, but in reality, they do not correspond to the beginning of the tool's life cycle.

All of these situations should ideally be considered to make the analysis more accurate and the results more trustworthy. Having more complete data and informations, along with consistent measurement routines, would definitely improve the reliability and usefulness of this kind of study.

References

Figure sources:

- Figures 1, 2 and 8 – Screenshots from internal Microsoft Excel documents used in the standardization process.
- Figure 3 – Screenshot from an internal Microsoft Word document outlining internal company rules.
- Figure 4 – Image taken from : https://www.dewalt.com/product/dch273b/en-us-20v-max-xr-sds-plus-brushless-1-l-shape-rotary-hammer-tool-only
- Figures 5, 6 and 7 – Screenshots from Windchill software (internal company use).
- Figure 9 – Image taken from: https://www.zeiss.com.sg/metrology/systems/cmms/bridge-type-cmms/contura.html
- Figure 10 – Image taken from: https://msiviking.com/zeiss-sensors-and-probes/
- Figures 11.1, 11.2, 13, 15 and 16 – Created using Minitab Statistical Software.

Online sources:

- 12 - https://sixsigmastudyguide.com/process-capability-cp-cpk/
- Table 14 – Based on content from Benchmark Six Sigma: https://www.benchmarksixsigma.com/forum/topic/34915-sigma-level/

Sources consulted (no direct references used in text):

- Introduction to Statistical Quality Control by Douglas C. Montgomery

- P. Vaníček (1 August 1969). ”Approximate Spectral Analysis by Least-squares Fit”
- Berrendero, Bueno-Larraz & Cuevas (2020). ”On Mahalanobis distance in functional settings”
- https://amnquality.com/en/what-is-ppap-in-the-quality-process/
- https://it.wikipedia.org/wiki/Distanza_di_Mahalanobis
- https://en.wikipedia.org/wiki/Least-squares_spectral_analysis
- https://calculatedcontent.com/2012/11/05/noisy-time-series/
- https://mathoverflow.net/questions/11803/unbiased-estimate-of-the-variance-of-a-weighted-mean
- Python Standard Library documentation: https://docs.python.org/3/library/
- SciPy documentation: https://docs.scipy.org/
- Pandas documentation: https://pandas.pydata.org/docs/
- NumPy documentation: https://numpy.org/doc/
- scikit-learn documentation: https://scikit-learn.org/stable/documentation.html
- Plotly documentation: https://plotly.com/python/
- Astropy Lomb-Scargle documentation: https://docs.astropy.org/en/stable/timeseries/lombscargle.html