Project B

Walmart Sales Forecasting

Author: Hemanth Jadiswami Prabhakaran
Matriculation No.: 7026000
Course of Studies: Mechanical Engineering

Author: Ayush
Matriculation No.: 7026XXX
Course of Studies: Mechanical Engineering

Author: Adil
Matriculation No.: 7026XXX
Course of Studies: Mechanical Engineering

First examiner: Prof. Dr. Elmar Wings

Submission date: May 26, 2025

University of Applied Sciences Emden/Leer · Faculty of Technology · Mechanical Engineering Department
Constantiaplatz 4 · 26723 Emden · http://www.hs-emden-leer.de

Contents

List of Figures

List of Tables

List of Listings

Acronyms

CNC Computerized Numerical Control

SPS Speicherprogrammierbare Steuerung

1. Introduction

Time series forecasting represents a fundamental domain in data science, with critical applications across numerous industries and sectors [FG19]. The ability to make accurate predictions based on historical patterns allows organizations to optimize operations, allocate resources efficiently, and maintain competitive advantage in increasingly challenging markets [PS17]. Within retail environments specifically, sales forecasting has emerged as an essential practice, enabling businesses to anticipate customer demand, manage inventory levels, and coordinate supply chain activities with greater precision [Zha21].

As global retail markets continue to evolve, companies face mounting pressure to refine their forecasting methodologies. Walmart, as the world's largest retailer with over 11,500 stores worldwide, exemplifies an organization where accurate sales prediction directly impacts operational success [Zha21]. With its extensive product range spanning groceries, electronics, apparel, and household goods across diverse geographic locations, Walmart confronts substantial challenges in forecasting sales across different departments and stores [Loy17]. The company's sales data exhibits complex patterns influenced by economic indicators, seasonal trends, promotional events, and holiday effects, necessitating sophisticated analytical approaches to generate reliable predictions.

Traditional time series forecasting has relied heavily on statistical methods such as Autoregressive Integrated Moving Average (ARIMA) models, which operate under assumptions of linearity and stationarity [PS17]. However, retail sales data frequently displays non-linear relationships and multiple forms of seasonality, particularly when examined at daily or weekly intervals [MMH18]. The weekly seasonality observed in retail data—corresponding to trading day effects when aggregated to monthly levels—represents a significant challenge for conventional analysis approaches [MMH18]. Additionally, special events and holidays such as Black Friday, Cyber Monday, Easter, and Labor Day introduce irregular patterns that further complicate the forecasting process.

Recent advances in machine learning have introduced alternative approaches for sales prediction, including regression trees, neural networks, and ensemble methods, which can potentially capture more complex patterns in retail data [PS17]. The relative effectiveness of these approaches compared to traditional statistical methods remains an active area of research, with empirical evidence suggesting that model performance varies considerably depending on data characteristics and forecasting horizons [FG19].

This study focuses on the Walmart sales dataset, which contains historical sales data from multiple departments across different Walmart stores. The dataset includes weekly sales figures along with additional variables such as store information, department details, holiday flags, temperature, fuel prices, unemployment rates, and consumer price indices [Loy17]. Through comprehensive analysis and modeling of this dataset, we aim to identify effective forecasting approaches that account for both regular seasonal patterns and special events that significantly impact retail sales. Our research contributes to the growing body of literature on retail sales forecasting by evaluating various methodologies and providing insights into the dynamics of department store sales across different temporal and spatial dimensions.

1.1. Introduction to the Project

This project focuses on analyzing and forecasting sales data from Walmart, one of the world's largest retail corporations. The dataset utilized in this study contains historical sales information from 45 Walmart stores located across different regions of the United States, with data spanning from February 2010 to November 2012 [Zha21]. Each store encompasses multiple departments, resulting in over 4,400 unique time series to analyze and forecast [Loy17]. The primary objective is to develop accurate predictive models that can effectively capture the underlying patterns in weekly sales across various departments and stores.

The Walmart dataset presents a rich resource for time series analysis due to its multifaceted nature. Each observation in the dataset includes weekly sales figures alongside several potential predictor variables, including store-specific information, department identifiers, holiday flags indicating special events, and economic indicators such as temperature, fuel prices, consumer price indices (CPI), and unemployment rates [Zha21]. The inclusion of these additional variables enables the exploration of both univariate and multivariate forecasting approaches, allowing for a comprehensive assessment of different methodological frameworks.

The project employs a structured analytical approach, beginning with exploratory data analysis to identify key patterns, trends, and seasonality components within the sales data. This initial investigation reveals significant variations in sales volumes across different stores and departments, as well as pronounced seasonal patterns and holiday effects that must be carefully considered in the modeling process. Following this exploratory phase, we implement and evaluate various forecasting methodologies, ranging from traditional time series techniques such as Seasonal-Trend decomposition using Loess (STL) and ARIMA models to more advanced machine learning approaches including regression trees and neural networks [PS17].

Through this systematic analysis, the project aims to contribute valuable insights into retail sales forecasting, particularly within large-scale multi-store environments where accurate predictions can significantly impact operational efficiency and financial performance. The findings have practical implications for inventory management, staff scheduling, and promotional planning within retail contexts.

1.2. Challenges

Forecasting retail sales at Walmart presents several significant challenges that must be addressed to achieve reliable predictions. First, the presence of multiple seasonal patterns in the data introduces complexity that conventional forecasting methods may struggle to capture adequately [MMH18]. Weekly sales data exhibits both annual seasonality (reflecting yearly consumption patterns) and weekly seasonality (corresponding to day-of-week effects), with these patterns potentially varying across different departments and store locations.

Second, the impact of special events and holidays represents a particularly challenging aspect of retail sales forecasting. The dataset identifies several major holidays—including Super Bowl, Labor Day, Thanksgiving, and Christmas—that significantly influence consumer purchasing behavior [Loy17]. These holiday effects are not uniform across all departments or stores, requiring careful modeling approaches to account for their differential impact. Furthermore, as noted by [MMH18], some holidays like Easter follow a lunar calendar and occur on different dates each year, complicating the identification of consistent patterns.

Third, the hierarchical structure of the data—encompassing multiple stores and departments—presents methodological challenges for forecasting. Decisions must be made regarding whether to develop individual models for each time series (bottom-up approach), aggregate the data and build more general models (top-down approach), or implement hierarchical forecasting methods that reconcile predictions across different levels of aggregation [FG19]. Each approach offers distinct advantages and limitations that must be carefully evaluated.

Fourth, the incorporation of external variables such as economic indicators introduces additional complexity. While these variables potentially enhance predictive accuracy by capturing broader economic conditions affecting consumer behavior, their integration requires addressing issues such as multicollinearity, appropriate lag structures, and potential nonlinear relationships with sales [Zha21]. The relative importance of these external factors may also vary across different store locations and departments, necessitating flexible modeling approaches.

Finally, the sheer volume of time series—comprising weekly sales for each department-store combination—presents computational challenges for model estimation and evaluation. This scale requires efficient algorithmic implementations and careful consideration of computational resources, particularly when implementing more complex machine learning approaches [PS17].

1.3. Applications

The applications of accurate sales forecasting for Walmart and similar retail organizations extend across numerous operational and strategic domains. First and foremost, precise sales predictions enable optimal inventory management—ensuring sufficient stock to meet customer demand while minimizing excess inventory that ties up capital and storage space [Zha21]. This balance is particularly critical for perishable goods where overstocking leads to waste and understocking results in lost sales opportunities.

Workforce planning represents another significant application area, where sales forecasts inform staffing decisions across different store departments and time periods. By anticipating fluctuations in customer traffic and sales volume, management can allocate human resources more efficiently, maintaining appropriate service levels during peak periods while controlling labor costs during slower periods [FG19]. This application becomes especially valuable during holiday seasons when both sales volumes and staffing requirements typically increase substantially.

Marketing and promotional planning also benefit considerably from accurate sales forecasts. By understanding the expected baseline sales and the potential impact of promotional activities, retailers can design more effective marketing campaigns and evaluate their return on investment more precisely [Zha21]. Furthermore, sales forecasts facilitate the evaluation of different markdown strategies preceding major holidays—a practice Walmart employs before events such as the Super Bowl, Labor Day, Thanksgiving, and Christmas [Loy17].

Supply chain optimization represents another critical application area. Accurate forecasts enable better coordination with suppliers, allowing for more precise ordering schedules and quantities. This coordination becomes particularly important for retailers like Walmart that operate extensive supply networks spanning multiple regions and countries. Improved forecasting can reduce supply chain disruptions, decrease lead times, and potentially lower transportation costs through more efficient logistics planning [FG19].

At a strategic level, sales forecasts inform financial planning and budgeting processes. Reliable projections of future sales provide the foundation for revenue forecasts, which subsequently influence decisions regarding capital expenditures, expansion plans, and shareholder communications [Zha21]. Additionally, accurate department-level forecasts can inform decisions about product assortment and space allocation within stores, potentially increasing overall sales per square foot—a key performance metric in retail operations.

1.4. Limitations

Despite the considerable value of sales forecasting, several limitations must be acknowledged when interpreting and applying the results of this study. First, the temporal scope of the available data (February 2010 to November 2012) represents a relatively short period that may not capture longer-term economic cycles or evolving consumer behaviors [Zha21]. This limited time frame particularly affects the model's ability to learn and predict the impact of infrequent events such as major economic downturns or structural changes in the retail landscape.

Second, while the dataset includes several economic indicators as potential predictors, it cannot account for all external factors that influence consumer purchasing decisions. Unobserved variables such as competitor actions, local events, changes in consumer preferences, or shifts in shopping channels (e.g., e-commerce versus physical retail) may significantly impact sales patterns in ways that the models cannot anticipate [FG19]. The growing influence of online shopping, which has accelerated in recent years, represents a particularly important factor that may not be fully captured in the historical data.

Third, the forecasting approaches implemented in this study necessarily involve simplifications and assumptions about the underlying data generating processes. As noted by [PS17], traditional time series models such as ARIMA assume linearity and stationarity, which may not hold for retail sales data exhibiting complex, non-linear patterns. While machine learning approaches offer greater flexibility, they too have limitations in terms of interpretability and potential overfitting to historical patterns that may not persist into the future.

Fourth, the aggregation of sales data at the weekly level obscures potentially valuable information about daily sales patterns. As demonstrated by [MMH18], daily retail data reveals more granular patterns, particularly regarding the impact of specific days of the week and holiday effects. The weekly aggregation in the Walmart dataset potentially masks these finer temporal dynamics, which could be relevant for operational decisions such as daily staffing or inventory replenishment.

Finally, while the dataset covers 45 Walmart stores, this represents only a small fraction of Walmart's total store network, which exceeds 11,500 locations worldwide [Zha21]. The generalizability of findings to other stores, particularly those in different countries or market environments, cannot be guaranteed. Cultural differences, varying economic conditions, and distinct shopping behaviors across regions may necessitate location-specific modeling approaches that cannot be fully explored with the available data.

Acknowledging these limitations provides important context for interpreting the results and suggests potential directions for future research, including the incorporation of more diverse data sources, exploration of higher-frequency sales data, and development of more flexible modeling approaches that can adapt to evolving retail environments.

1.5. Allgemeine mathematische Beschreibung Bézier-Kurve

Bézier curves are formed with the help of Bernstein polynomials. If at least two points and two tangents are known, control points can be determined with which a control polygon is formed. The course of the curve is oriented to this control polygon. The number of control points depends on the degree of the Bézier curve. A Bézier curve of degree n has n+1 control points. The Bézier curve is calculated using the De Casteljau algorithm.

Definition. In the interval [0; 1] the **Bernstein polynomial of nth degree** is defined by:

$$b_{i,n}(\lambda) = \binom{n}{i}(1-\lambda)^{n-i}\lambda^i, \quad \lambda \in [0,1], \quad i = 0, \dots, n \tag{1.1}$$

Definition. The binomial coefficient is defined by:

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}, \quad i = 0, \dots, n \tag{1.2}$$

Here the Bernstein polynomials $b_{i,n}$ form a basis of the vector space $\mathbb{P}^n(I)$ of polynomials of degree at most n over I. Thus every polynomial of degree at most n can be written uniquely as a linear combination. [Far02]

Beispiel. Determination of Bernstein polynomials for $n = 3$ holds:

$$\begin{aligned}
b_{3,0}(\lambda) &= \binom{3}{0}(1-\lambda)^{3-0}\lambda^0 &&= (1-\lambda)^3 \\
b_{3,1}(\lambda) &= \binom{3}{1}(1-\lambda)^{3-1}\lambda^1 &&= 3\lambda(1-\lambda)^2 \\
b_{3,2}(\lambda) &= \binom{3}{2}(1-\lambda)^{3-2}\lambda^2 &&= 3\lambda^2(1-\lambda) \\
b_{3,3}(\lambda) &= \binom{3}{3}(1-\lambda)^{3-3}\lambda^3 &&= \lambda^3
\end{aligned}$$

$b_{3,i}(\lambda)$ with $i = 0, 1, 2, 3$ are the cubic Bernstein polynomials of degree 3.

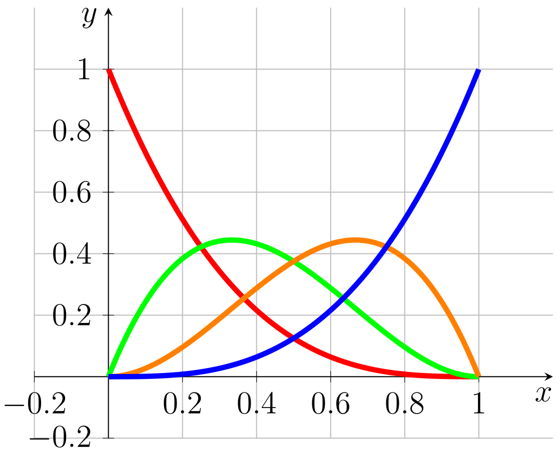


Figure 1.1.: Bernstein polynomial of degree 3

Definition. Given are the points

$$Q_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \quad \text{mit} \quad Q_i \in \mathbb{R}^2, \quad i = 0, 1, \ldots, n$$

A **Bézier curve** is then defined by

$$C(\lambda) = \sum_{i=0}^{n} Q_i \cdot b_{i,n}(\lambda) \tag{1.3}$$

The points $Q_i, i = 0, \ldots, n$ are called **control points**.

The control points of a Bézier curve form the so-called control polygon.

Bemerkung. Let the starting point P_0 and the end point P_1, as well as the tangents $\vec{t_0}$ and $\vec{t_1}$ be given. The tangents are not necessarily normalised.
The control points Q_0, Q_1, Q_2 and Q_3 of the associated Bézier curve are given by the following equations:

$$Q_0 = P_0, \qquad Q_1 = P_0 + \lambda_0 \vec{t_0}, \qquad Q_2 = P_1 - \lambda_1 \vec{t_n}, \qquad Q_3 = P_1 \tag{1.4}$$

[Jak+10]

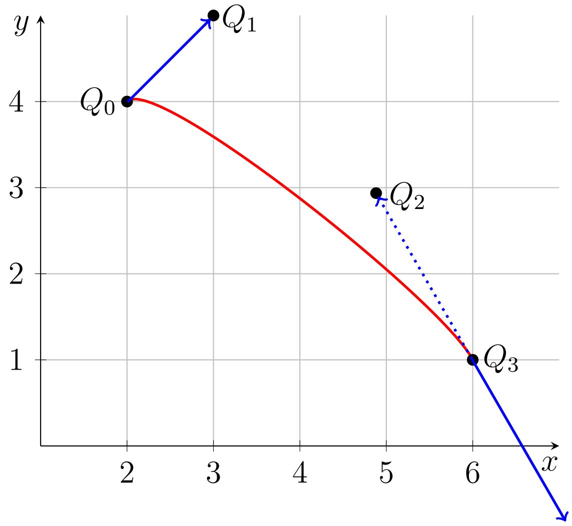


Figure 1.2.: Bézier curve for example 1.5

Beispiel. Given are

$$P_0 = \begin{pmatrix} 2 \\ 4 \end{pmatrix}, \quad \vec{t}_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \text{und} \quad P_1 = \begin{pmatrix} 6 \\ 1 \end{pmatrix}, \quad \vec{t}_1 = \begin{pmatrix} 1 \\ -2 \end{pmatrix}$$

Then, according to theorem **??** for control points of the associated Bézier curve results:

$$Q_0 = P_0 = \begin{pmatrix} 2 \\ 4 \end{pmatrix}, \qquad Q_1 = P_0 + \vec{t}_0 = \begin{pmatrix} 2 \\ 4 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 3 \\ 5 \end{pmatrix},$$

$$Q_2 = P_1 - \vec{t}_n = \begin{pmatrix} 6 \\ 1 \end{pmatrix} - \begin{pmatrix} 1 \\ -2 \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \end{pmatrix}, \qquad Q_3 = P_1 = \begin{pmatrix} 6 \\ 1 \end{pmatrix}$$

The Bézier curve and its control points are shown in the figure **??**.

2. Verrundung mit einem Kreisbogen

2.1. Gleichungen

Blending with an arc is a simple variant of smoothing corners. This variant enables the processing and the generation of files according to DIN 66025. For smoothing, three points P_0 and S and P_1 are always considered, thus a symmetric Hermite problem results. According to theorem **??** it is assumed to be in the standard form $HP(L, \alpha)$.

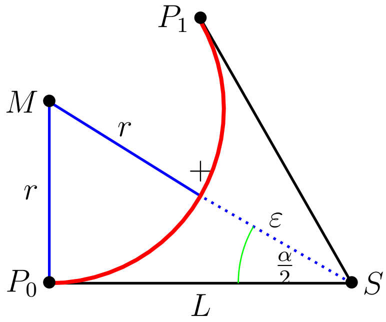


Figure 2.1.: Smoothing a corner with the help of an arc - triangle

The figure **??** represents the situation. The points P_0, S and P_1 are given. The included angle $\alpha = \angle(P_0; S; P_1)$ as well as the distance $L = \||S - P_0\|| = \||P_1 - S\||$ are shown in the graph. The red arc is the desired result. The distance of its centre M to S is then $r + \varepsilon$, where r is the radius of the arc and ε is the given tolerance. Thus we obtain a right triangle P_0SM whose edge lengths are L, $r + \varepsilon$ and r.

According to the definition of the sine and the tangent, it follows:

$$\sin\left(\frac{\alpha}{2}\right) = \frac{L}{r+\varepsilon} \qquad \text{und} \qquad \tan\left(\frac{\alpha}{2}\right) = \frac{L}{r}$$

$$\Leftrightarrow \quad \varepsilon = \frac{L}{\sin\left(\frac{\alpha}{2}\right)} - r \qquad \text{und} \qquad r = \cot\left(\frac{\alpha}{2}\right) L$$

The two equations can be combined to give the following conditions:

$$\varepsilon = L \cdot \frac{1-\cos\left(\frac{\alpha}{2}\right)}{\sin\left(\frac{\alpha}{2}\right)} \qquad \text{bzw.} \qquad L = \varepsilon \cdot \frac{\sin\left(\frac{\alpha}{2}\right)}{1-\cos\left(\frac{\alpha}{2}\right)}$$

This gives the equation for the radius:

$$r = L \cdot \cot\left(\frac{\alpha}{2}\right) = L \cdot \sqrt{\frac{1-\cos(\alpha)}{1+\cos(\alpha}} = L \cdot \frac{\sin(\alpha)}{1+\cos(\alpha)} = L \cdot \frac{P_{1,x}}{P_{1,y}}$$

The factor for converting L and ε can be simplified using trigonometric transformations.

$$\frac{1-\cos\left(\frac{\alpha}{2}\right)}{\sin\left(\frac{\alpha}{2}\right)} = \frac{1-\sqrt{\frac{1-\cos(\alpha)}{2}}}{\sqrt{\frac{1+\cos(\alpha)}{2}}} = \frac{\sqrt{2}-\sqrt{1-\cos(\alpha)}}{\sqrt{1+\cos(\alpha)}} = \frac{\sqrt{1+\cos(\alpha)}-\sin(\alpha)}{1+\cos(\alpha)}$$

By comparing this with the symmetric Hermite problem in standard form, the following notation is then obtained:

$$\frac{1-\cos\left(\frac{\alpha}{2}\right)}{\sin\left(\frac{\alpha}{2}\right)} = \frac{\sqrt{P_{1,x}}-P_{1,y}}{P_{1,x}}$$

The previous considerations are now summarised in the following sentence.

Satz. Let a symmetric Hermite problem $(P_0,\, \vec{t_0},\, P_1,\, \vec{t_1}, S, L)$ be given. Without restriction of generality, it is in the standard form

$$\left\{\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}L+L\cdot\cos(\alpha)\\L\cdot\sin(\alpha)\end{pmatrix}, \begin{pmatrix}\cos(\alpha)\\\sin(\alpha)\end{pmatrix}, \begin{pmatrix}L\\0\end{pmatrix}, L\right\}$$

with $L \in \mathbb{R}^{>0}$ uad $\alpha \in (-\pi; \pi]$.

Then an arc can be found that connects the points P_0 and P_1, has the same tangent directions at the two points.

For the circular arcs applies:

$$r = L \cdot \left|\frac{P_{1,x}}{P_{1,y}}\right|; \quad \phi_0 = -\operatorname{sign}(\alpha) \cdot \frac{\pi}{2}; \quad \phi_1 - \phi_0 = \operatorname{sign}(\alpha) \cdot \pi - \alpha = \beta;$$

$$M = P_0 + \operatorname{sign}(\alpha) \cdot r \cdot \vec{t_0^\perp} = \operatorname{sign}(\alpha) \cdot r \cdot \begin{pmatrix}0\\1\end{pmatrix}$$

From the specification of the distance L, the maximum error can be calculated, as shown in theorem **??**. According to the derivation, it is also possible to specify the maximum error ε and determine the maximum distance L from it.

Satz. Let a symmetric Hermite problem $(P_0, \vec{t_0}, P_1, \vec{t_1}, S, L)$ be given. Without restriction of generality, it is in the standard form

$$\left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} L + L \cdot \cos(\alpha) \\ L \cdot \sin(\alpha) \end{pmatrix}, \begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) \end{pmatrix}, \begin{pmatrix} L \\ 0 \end{pmatrix}, L \right\}$$

with $L \in \mathbb{R}^{>0}$ and $\alpha \in (-\pi; \pi]$.

a) Let the maximum error ε be given. The maximum distance L for which an arc exists according to the theorem **??** that takes the error into account is given by:

$$L(\varepsilon, \alpha) = \varepsilon \cdot \frac{P_{1,x}}{\sqrt{P_{1,x}} - P_{1,y}} = \varepsilon \cdot \frac{L + L \cdot \cos(\alpha)}{\sqrt{L + L \cdot \cos(\alpha)} - L + L \cdot \cos(\alpha)}$$

b) When the distance L is specified, the following maximum error results:

$$\varepsilon(L, \alpha) = L \cdot \frac{\sqrt{P_{1,x}} - P_{1,y}}{P_{1,x}} = L \cdot \frac{\sqrt{L + L \cdot \cos(\alpha)} - L + L \cdot \cos(\alpha)}{L + L \cdot \cos(\alpha)}$$

These considerations result in limitations for the use of this strategy, which are summarised in the following comment.

Bemerkung. The following conditions for applying the strategy of a circle must be fulfilled:

a)
$$P_0 \neq P_1$$

b)
$$\vec{t_0} = -\vec{t_1} \quad \Leftrightarrow \quad \alpha = 0$$

c)
$$\vec{t_0} = \vec{t_1} \quad \Leftrightarrow \quad \alpha = \pi$$

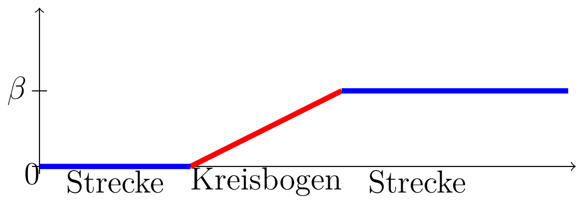


Figure 2.2.: Angular change with blending arc

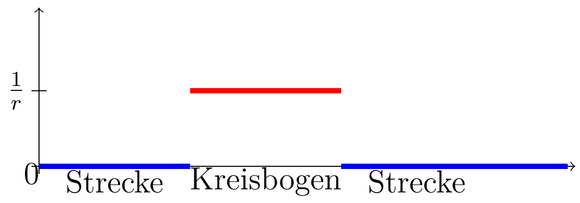


Figure 2.3.: Curvature progression for a blending arc

2.2. Bewertung

Rounding by means of a circular arc leads to a continuous course of the angle change. The figure **??** illustrates this.

Due to the rounding with a circular arc, the curvature of the curve is not continuous. The figure **??** shows that the curvature is constant in the range of distances 0 and on the circular arc with the value $\frac{1}{r}$, where r is the radius of the circular arc used. Due to the formula $a = \frac{v^2}{r}$ for a movement on a circular arc, the acceleration course exhibits continuity jumps at the transitions for a constant path velocity.

Depending on the angle change β at the corner S and the tolerance ε, the maximum length of the shortening of the lines can be calculated. The figure **??** shows the corresponding diagram, where the tolerance ε is set to the value 1.

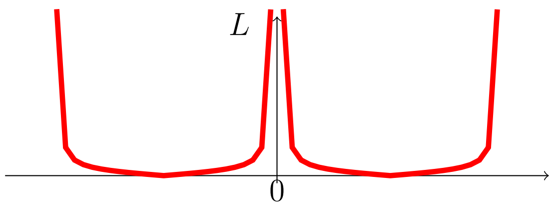


Figure 2.4.: Maximum range for a blending arc of a circle - $L(\varepsilon = const, \alpha)$

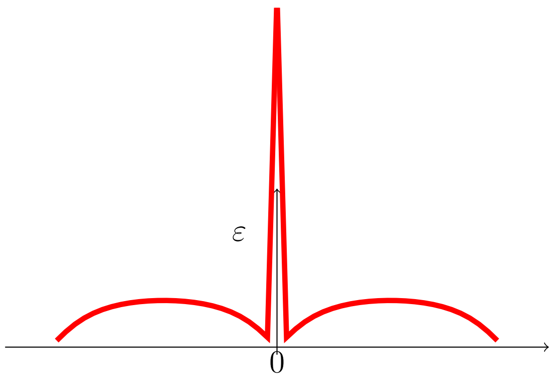


Figure 2.5.: Deviation when specifying the distance L when rounding with a circular arc

The area provided can also be specified. Depending on the distance L from the corner point S, the deviation ε can be calculated. The figure **??** represents the function as a function of L and the angle α.

The figures **??** and **??** show the curves of the length as a function of the angle change for a fixed tolerance and the error as a function of the angle change for a given length. If the angle change is minimal, then the error or length L is extreme. In this case, rounding by means of an arc is not possible. In order to develop a sensible strategy, a minimum angle change must be specified from which a blemnding arc is permitted. Likewise, a maximum angle change must also be defined. For an angular change of $\pm\pi$ is a bend. In this case, a blending is also not possible.

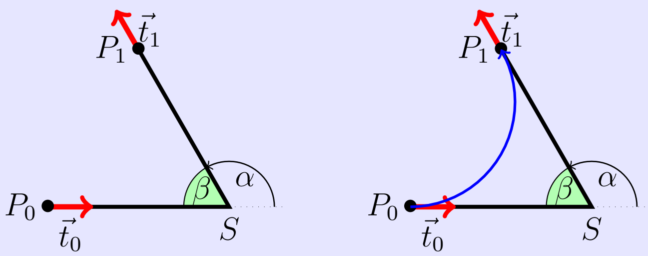
2.3. Examples

Beispiel. Let it be the symmetric Hermite problem

$$\left\{\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}6.0 + 6.0 \cdot \cos\left(\frac{2}{3}\pi\right)\\ 6.0 \cdot \sin\left(\frac{2}{3}\pi\right)\end{pmatrix}, \begin{pmatrix}\cos\left(\frac{2}{3}\pi\right)\\ \sin\left(\frac{2}{3}\pi\right)\end{pmatrix}, \begin{pmatrix}6.0\\0\end{pmatrix}, 6.0\right\}$$

with $L = 6$ and $\alpha = \frac{2}{3}\pi$.
For the blending arc follows:

$$M = \begin{pmatrix}0\\3,4641\end{pmatrix}; \quad r = 3,46412; \quad \phi_0 = -\frac{\pi}{2}; \quad \alpha = \frac{2}{3}\pi$$

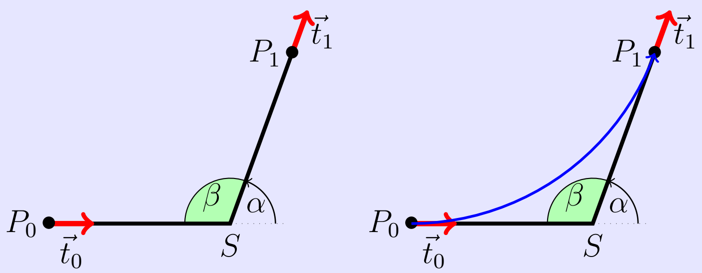


Beispiel. Let it be the symmetric Hermite problem

$$\left\{\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}6.0 + 6.0 \cdot \cos\left(\frac{7}{18}\pi\right)\\ 6.0 \cdot \sin\left(\frac{7}{18}\pi\right)\end{pmatrix}, \begin{pmatrix}\cos\left(\frac{7}{18}\pi\right)\\ \sin\left(\frac{7}{18}\pi\right)\end{pmatrix}, \begin{pmatrix}6.0\\0\end{pmatrix}, 6.0\right\}$$

with $L = 6$ and $\alpha = \frac{7}{18}\pi$.
For the blending arc follows:

$$M = \begin{pmatrix}0\\8,5689\end{pmatrix}; \quad r = 8,5689; \quad \phi_0 = -\frac{\pi}{2}; \quad \alpha = \frac{7}{18}\pi$$

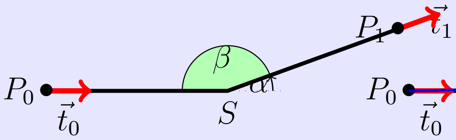


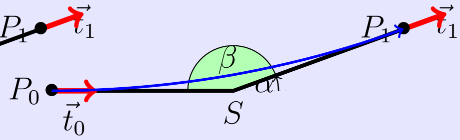
Beispiel. Let it be the symmetric Hermite problem

$$\left\{\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}6.0+6.0\cdot\cos\left(\frac{1}{9}\pi\right)\\6.0\cdot\sin\left(\frac{1}{9}\pi\right)\end{pmatrix},\begin{pmatrix}\cos\left(\frac{1}{9}\pi\right)\\\sin\left(\frac{1}{9}\pi\right)\end{pmatrix},\begin{pmatrix}6.0\\0\end{pmatrix},6.0\right\}$$

with $L=6$ and $\alpha=\frac{1}{9}\pi$.
For the blending arc follows:

$$M=\begin{pmatrix}0\\34,0277\end{pmatrix};\quad r=34,0277;\quad \phi_0=-\frac{\pi}{2};\quad \alpha=\frac{1}{9}\pi$$

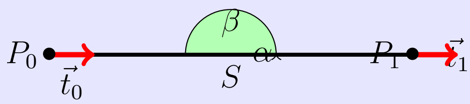




Beispiel. Let it be the symmetric Hermite problem

$$\left\{\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}12.0\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}6.0\\0\end{pmatrix},6.0\right\}$$

with $L=6$ and $\alpha=0$.
There is no blending arc here.

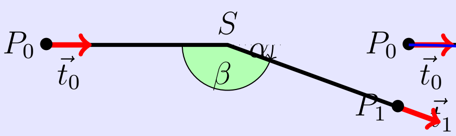


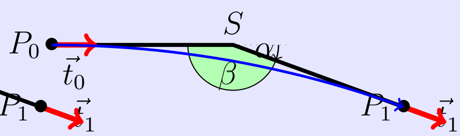
Beispiel. Let it be the symmetric Hermite problem

$$\left\{\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}1\\0\end{pmatrix},\begin{pmatrix}6.0+6.0\cdot\cos\left(-\frac{1}{9}\pi\right)\\6.0\cdot\sin\left(-\frac{1}{9}\pi\right)\end{pmatrix},\begin{pmatrix}\cos\left(-\frac{1}{9}\pi\right)\\\sin\left(-\frac{1}{9}\pi\right)\end{pmatrix},\begin{pmatrix}6.0\\0\end{pmatrix},6.0\right\}$$

with $L=6$ and $\alpha=-\frac{1}{9}\pi$.
For the blending arc follows:

$$M=\begin{pmatrix}0\\-34,0277\end{pmatrix};\quad r=34,0277;\quad \phi_0=\frac{\pi}{2};\quad \alpha=\frac{1}{9}\pi$$



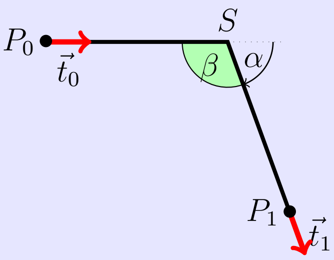


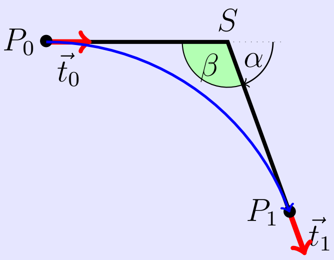
Beispiel. Let it be the symmetric Hermite problem

$$\left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 6.0 + 6.0 \cdot \cos\left(-\frac{7}{18}\pi\right) \\ 6.0 \cdot \sin\left(-\frac{7}{18}\pi\right) \end{pmatrix}, \begin{pmatrix} \cos\left(-\frac{7}{18}\pi\right) \\ \sin\left(-\frac{7}{18}\pi\right) \end{pmatrix}, \begin{pmatrix} 6.0 \\ 0 \end{pmatrix}, 6.0 \right\}$$

with $L = 6$ and $\alpha = -\frac{7}{18}\pi$.
For the blending arc follows:

$$M = \begin{pmatrix} 0 \\ -8,5689 \end{pmatrix}; \quad r = 8,5689; \quad \phi_0 = \frac{\pi}{2}; \quad \alpha = -\frac{7}{18}\pi$$



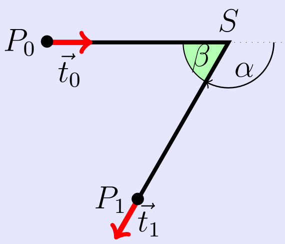


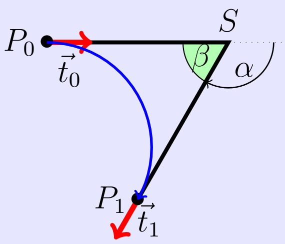
Beispiel. Let it be the symmetric Hermite problem

$$\left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 6.0 + 6.0 \cdot \cos\left(-\frac{2}{3}\pi\right) \\ 6.0 \cdot \sin\left(-\frac{2}{3}\pi\right) \end{pmatrix}, \begin{pmatrix} \cos\left(-\frac{2}{3}\pi\right) \\ \sin\left(-\frac{2}{3}\pi\right) \end{pmatrix}, \begin{pmatrix} 6.0 \\ 0 \end{pmatrix}, 6.0 \right\}$$

with $L = 6$ and $\alpha = -\frac{2}{3}\pi$.
For the blending arc follows:

$$M = \begin{pmatrix} 0 \\ -3,4641 \end{pmatrix}; \quad r = 3,46412; \quad \phi_0 = \frac{\pi}{2}; \quad \alpha = -\frac{2}{3}\pi$$





3. Maple files

[Wat17c; Wat17d; Wat17b; Wat17e; Wat17a; Wat17f]

3.1. Geometries with Maple

The algorithms presented can be well represented using geometries. Two aspects can be investigated. On the one hand, the effort for an implementation, the computational accuracy and the stability of an algorithm are interesting; on the other hand, the methods are to be evaluated and compared with regard to their usefulness. For this purpose, modules for the formula manipulation system Maple [Wat17c] were developed.

Maple offers the environment to implement algorithms easily and quickly. Furthermore, graphical representation is possible with simple means. However, a simple workbook of Maple is not designed for very large software projects. Here, one must resort to the possibility of creating and using libraries. On the one hand, Maple offers the possibility of creating one's own libraries, so-called modules. In this way, one remains within the environment and syntax of Maple. This path will be pursued further here. Another possibility is the use of libraries created by means of a higher programming language, e.g. C++, the so-called DLLs.

In the following, the creation and use of a test environment with the help of modules is described. First, the geometry elements that are stored in various modules and their use are described. The structure and use of a module in Maple is then explained so that own extensions and additions are possible.

3.2. Geometry elements used

This is a test environment for the algorithms presented, only geometries that have been described are also used.

- points (`MPoint`)
- lines (`MLine`)
- arcs (`MArc`)
- Bézier curves (`Bezier`)

- polygon courses (`MPolygon`)
- Geometry List (`MGeoList`)
- Hermite problems (`MHermiteProblem`)
- Symmetric Hermite Problems (`MHermiteProblemSym`)

For each geometry element a corresponding module has been created, the name of which is given in the list above. The list of which functions are available is presented in another section.

When implementing such a project, it quickly becomes clear that when using several geometry elements, a clear data structure and well-defined access to the data is essential. For example, when representing straight lines, the decision must be made whether the representation should be point-directional or by means of two points. The choice is generally made depending on the application. Here, the path is taken that, due to a well-defined access to the data, the use of both representations is possible.

3.3. Building the data structure

The idea is to use a data structure that is as uniform and simple as possible. Maple does offer the possibility to define objects. However, it is difficult to use within a procedural environment. Therefore, all data is basically represented as lists. The first list element always contains an identifier of the element ??. This is followed by the data, which in turn can be geometry elements. It should be noted that direct access to the data cannot be prevented; here the user is responsible.

Access to the data should be exclusively via procedures of a module. The following is an example of the data structure for points:

```
[MVPOINT,[x,y]]
```

The creation of a point then takes place via a procedure `New`:

```
NeuerPunkt := MPoint:-New(10,15);
```

When called up in the test file, the following is then output:

```
TestPO := ["Point",[10,15]]
```

These data structures are used for the calculation of geometries. They are used in functions or procedures. Unlike variables, the data structures and procedures are defined globally and not locally. Under the command `export` the procedures are declared at the beginning of the module and can thus also be used outside the module. If, for example, a data structure from `MPoint` is to be used in another module or outside the modules, it is called as follows:

```
P0 := MPoint:-New(x,y);
```

In addition to the modules for the geometries, there is a module (MConstant) for saving constants and names. There, for `MVPOINT`, for example. „Point “ or e.g. a constant for the comparison to zero for real numbers.

Finally there is the test file, which is not a module, in which the modules are loaded, called and tested in their function. More about this file later in „1.4 Maple test file “.

3.4. Structure of a module

The following section deals with the purpose of modules and their structure.

The project is about capturing the above geometries in a data structure and representing them in Maple. However, the calculations and formulas that have to be used are a hindrance to the final representation. Modules are very well suited for summarising and hiding the calculations that take place in functions. This way, the important functions can be accessed in Maple without seeing what is in them.

Example:

The function `Angle` from the module `MPoint`: Function to calculate an angle between location vector and x-axis:

```
Angle := proc(P)
    local alpha, x, y;
    x := GetX(P);
    y := GetY(P);
    alpha := 0;
    if abs(x) < 0.00001
        then
            alpha := 3*Pi/2;
        else
            alpha := Pi/2;
        end if;
    else
        alpha := arctan(y/x);
        if x < 0
        then
            alpha := alpha + Pi;
        else
            if y < 0
```

```
            then
                alpha := alpha + 2*Pi;
            end if;
        end if;
    end if;
    return factor(alpha);
end proc;
```

This function is long, but it only represents a small part of the programme for the representation in question. That is why it is in the module and can thus be called externally with a single command:

```
Winkel := MPoint:-Angle(P)
```

The following section describes the structure of a module. Since Maple offers a wide range of possibilities, a restriction is made. Only the structure of the modules used is described, here on the basis of the module`MPoint`. For more detailed possibilities, please refer to the Maple manual [Wat17c].

structure:

The module must first be started. This is done with the name (here always a capital M and the geometry) of the module and the following command:

```
MPoint := module()
```

Then, similar to procedures, the variables and functions must be defined. You can declare them either locally or globally. If the variables or procedures are only used and changed in the module, the declaration is made with the command `local` as follows:

```
local  Variable names, with, comma, separated;
```

If the variables or procedures are also to be usable outside the module, this is defined with `export`:

```
export Function names, with, comma, separated;
```

This is followed by the specification of the options:
`option package;`,
the description of the module:
`description “Self-selected module description, e.g.  module for points “`;
and the initialisation of the module:

```
ModuleLoad := proc
   MVPOINT:=MConstant:-GetPoint();
   print("Module MPoint is loaded");
end proc;
```

From here on, programming is done as usual in Maple. The previously defined procedure and variable names are used and programming is done with commands that are also used outside of a module in Maple. It is important to know that with modules, each function can be called constantly, so the order of the functions is irrelevant.

To finally end the module, use the following command:

```
end module;
```

3.4.1. General structure of a module

In summary, the modules have the following structure:

```
Modulname := module()

    local Names, with, comma, separated;

    export Names, with, comma, separated;

    global Names, with, comma, separated;[1]

    option package;

    description „ Self-selected module description“;


    ModuleLoad := proc()[2]
        MVPOINT:=MConstant:-GetPoint();
        print("Modul MPoint is loaded");
    end proc;

  Procedure1 := proc (Passing parameters)
        inhalt;
    end proc;

    Procedure2 := proc (Passing parameters)
```

[1]Possible, but not used in the modules
[2]Example MPoint

```
        content;
    end proc;


    ...

end module;
```

3.4.2. Saving a module

The management of modules is done automatically by Maple. However, since the modules are passed on and have to be edited individually, some settings have to be made. Therefore, the following prefix is used for each Maple file of the project:

```
1 restart;
2 with(LibraryTools);
3 lib := "C:/FH/Tools/Maple/MyLibs/Blending.mla";
4 march('open', lib);
5 ThisModule := 'MArc';
```

The first line initialises the system. The next 3 lines enable working with archives. In the second line, the tools are loaded so that the variable `lib` can be occupied. All modules are stored in an archive; in this example it is the file `Blending.mla` in the directory `C:/FH/Tools/Maple/MyLibs/`. The command `ThisModule := 'MArc';` contains the name of the current module.

A module is then saved using the command `savelib('ModuleName')`. A file `ModuleName.mla` is then created. Depending on the configuration of Maple, the set path where the file is automatically created may not be writable. Then you can configure the system so that the mla file is stored in the current directory or in a directory of your choice.

If the above prefix is used for a Maple file, all that is required to save the module is

```
savelib(ThisModule, lib);
```

3.4.3. Verwendung eines Moduls

A module that has been saved can now be used in other Maple worksheets. To do this, the command

```
with(ModuleName)
```

is used. If you have used a special path, Maple cannot find the file `ModuleName.mla`. Then the path must be made known, e.g.:

```
savelibname := "c:/Maple/MyLibs";", savelibname;
```

Now the procedures of the module can be accessed. An overview of all exported procedures is provided by the command

```
Describe(ModuleName)
```

is displayed. A list of the names including the names of the transfer parameters is displayed. If a procedure is equipped with the field `description`, this text is also displayed.

3.4.4. Creating a Maple module

Creating your own module is done quickly if you use the framework above. However, one should make some considerations beforehand and maintain a standard.

Before creating an own module, the data model and the corresponding procedures should be worked out. A programme flow chart is useful here. The central task of the module is usually quickly determined. In addition, however, the following rules should be observed.

Rule 1 Basically, both the module and each procedure receive a description. This is not limited to the optional argument `description`, but is placed in front of each procedure. The description basically contains the description of the task. The prerequisite is also mentioned or an error handling is described. Then follows the description of all input parameters, their function and their data structure. The return value is then described.

rule 2 Each module receives a procedure `Version()`, which returns the current version number.

rule 3 Data is not global. To access data, procedures `Set*` and `Get*` are provided. These procedures are also used within the procedures of a module.

rule 4 At least one test function is written for each procedure. The test function can be used to illustrate the use and any special features.

3.5. Functions of the modules

In the following, the individual modules are listed. The data structure of each module and all functions with associated tasks are listed.

3.5.1. Module `MPoint`

`MPoint` is a module for points and works with a data structure and with procedures/functions. The data structure `New` for a point is a list, which is structured as follows:

```
[MVPOINT, [x,y]]
```

Its first element is a name to identify the module. Your second element is again a list containing the elements of the geometry. In this case the name is "Point" and the elements are the x and y coordinates for a point. No distinction is made here between point and vector. A point can also be seen as a vector from the coordinate origin to the point.

Via the Get functions GetX and GetY one can capture the coordinates of the point and use them in other functions. The other functions can then be used to calculate with the points/vectors.

MPoint		
E	**New**	Data structure for a point
E	**GetX**	Reading the x-coordinate
E	**GetY**	reading the y-coordinate
E	**Angle**	calculating the angle between the x-axis and the point
E	**Add**	Calculates a linear combination of two points/vectors
E	**Sub**	Calculates the difference of two points/vectors
E	**Cos**	Calculates the cosine between two vectors
E	**Sin**	Calculates the sine between two vectors
E	**Scale**	Scales a vector with a factor
E	**Perp**	Calculates a vector that is orthogonal to the given vector
L	**IllustrateXY**	Plot function to illustrate a blue point
E	**Illustrate**	Plot of a blue point
E	**Plot**	Plot of a green point
E	**Plot2D**	Plot of a point with own options
E	**Length**	Calculates the distance of the point to the coordinate origin
E	**Uniform**	Normalises the vector to length 1
E	**LinetoVector**	Calculates vector from a distance
E	**Distance**	Calculates the distance between two points

3.5.2. Module MLine

MLine is a module for lines and works with a data structure and with procedures/functions. The data structure New for a line is a list which is structured as follows:

```
[MVLINE, [P0,P1]]
```

Its first element is a name to identify the module. Its second element is again a list containing the elements of the geometry. In this case the name is „Line“ and the elements are the start and end points for a line.

The data structure NewPointVerctor is also a data structure for a line, but here the line is defined by a start point and a direction vector.

The Get functions StartPoint and EndPoint can be used to capture the start and end points of the route and use them in other functions. The other functions can then be used to calculate with the routes.

	MLine	
E	New	Data structure for a line (two-point form)
E	NewPointVector	creation of a route (point-direction form)
E	StartPoint	reading the starting point
E	EndPoint	reading the end point
E	Position	Calculate a point that lies on the line
E	Plot2D	Plot a part of the route starting from the starting point
E	Plot2DTangent	Plot of a point with tangent
E	Plot2DTangentArrow	Plot of a point with tangent (as arrow)
E	LineLine	Calculation of the intersection of two straight lines.
E	AngleLine	Calculation of the angle between two straight lines

3.5.3. Module MArc

MArc is a module for circular arcs and works with a data structure and with procedures/functions. The data structure New for an arc is a list that is structured as follows:

```
[MVARC, [mx,my,r,phi,alpha]]
```

Its first element is a name to identify the module. Your second element is again a list containing the elements of the geometry. In this case the name is "Arc" and the elements are the x- and y-coordinate for the centre, the radius, the start angle and the angle change for an arc.

The Get functions GetM, GetMX, GetMY, GetR, GetPhi, and GetAlpha can be used to collect the data for an arc and use it in other functions. The other functions can then be used to calculate with the arcs.

	MArc	
E	New	Data structure for an arc
E	GetMX	Reading the x-coordinate of the centre point.
E	GetMY	read the y-coordinate of the centre point
E	GetR	read the radius
E	GetPhi	reading the start angle
E	GetAlpha	Reading the change of angle
E	GetM	reading the centre point
E	Position	Calculating a point on the arc
E	Plot2D	Plot a part of the arc starting from the start angle
E	Blend	calculating an arc from a symmetric Hermite problem

3.5.4. module MBezier

The New data structure for a polygon is a list constructed as follows:

`[MVBEZIER, [PointList]]`

Within the module `MBezier` exist the procedures listed in the following table.

MBezier

E	**New**	Manual input of control points for a Bézier curve
E	**Version**	Output of the verions
E	**BlendCurvature**	Determination of control points from symmetric Hermite problem
E	**BlendCurvatureEpsilon**	determination of control points from symmetric Hermite problem with given error
E	**Position**	position on the Bézier curve
E	**GetTheta**	Reading the angle
E	**GetEpsilon**	reading the tolerance
E	**GetControlPoint**	Read the control points
E	**Plot2D**	Read the Bézier curve
E	**PlotControlPoints**	representation of all control points

3.5.5. Module MPolygon

`MPolygon` is a module for polygons and works with a data structure and with procedures/functions. The data structure `New` for a polygon is a list that is structured as follows:

`[MVPOLYGON, [PointList]]`

Its first element is a name to identify the module. Your second element is again a list containing the elements of the geometry. In this case the name is "Polygon" and the elements are any number of points.

Using the Get functions `GetPoint` and `GetN` you can get the number of points and the points themselves and use them in other functions. The other functions can then be used to calculate with the points or the polygon course.

MPolygon

E	**New**	Data structure for a list of points
E	**GetPoint**	Reading the ith point from the point list
E	**GetN**	Determine the number of points in the point list
E	**Length**	Determination of the Euclidean length of the polygon
E	**Position**	Calculation of a point on the polygon course
E	**Tangents**	calculation of the tangent
L	**Plot2DAll**	Plot list of all points
E	**Plot2D**	Plot of all points as polygonal plots.
E	**Plot2DTangent**	representation of the polygon course with tangent
E	**PlotPoints**	representation of all points

3.5.6. module MGeoList

MeoList is a module for a geometry list and works with a data structure and with procedures/functions. The data structure New for a geometry list is a list that is structured as follows:

```
[MVGEOLIST, []]
```

Its first element is a name to identify the module. Its second element is again a list containing the elements of the geometry. In this case the name is „GeoList“ and the elements are any number of individual geometries.

Using the Get functions GeoGeo and GetN, the i-th element of the list and the number of elements in the list can be captured and used in other functions. The other functions can then be used to calculate with the geometries or the list. In the functions Length, Plot2DAll, Plot2D and Position the functions are called in themselves. This is possible because the functions work independently of each other.

	MGeoList	
E	**New**	Data structure for a geometry list
E	**Append**	Append a geometry element to the data structure
E	**Prepend**	Inserting a geometry element as the first element of the list
E	**Replace**	Replace a geometry element with another one.
E	**GetN**	Determine the number of geometry elements in the list
E	**GeoGeo**	Reading the i-th geometry element
E	**Length**	Calculate the Euclidean length of the geometry elements
E	**Position**	Calculates a point on the geometry
L	**Plot2DAll**	Plot function for plotting the geometry elements
E	**Plot2D**	Plot a part of the geometry list starting from the first element

3.5.7. Module MHermiteProblem

MHermiteProblem is a module for Hermite problems and works with a data structure and with procedures/functions. The data structure New for a route is a list, which is structured as follows:

```
[MVHERMITEPROBLEM, [P0,T0n,P1,T1n]]
```

Your first element is a name to identify the module. Its second element is again a list containing the elements of the geometry. In this case the name is „HermiteProb“ and the elements are two points with associated tangents (lines).

Using the Get functions StartPoint, EndPoint, StartTangent and EndTangent, the points and their tangents can be captured and used in other functions. The other functions can then be used to calculate with the data for the Hermite problem.

MHermiteProblem		
E	**New**	Data structure for a Hermite problem
E	**StartPoint**	reading the start point
E	**EndPoint**	Reading the end point command
E	**StartTangent**	reading the start tangent
E	**EndTangent**	Reading the end tangent
E	**Plot2D**	Plotting the Hermite problem

3.5.8. Module MHermiteProblemSym

MHermiteProblemSym is a module for symmetric Hermite problems and works with a data structure and with procedures/functions. The data structure New for a symmetric Hermite problem is a list built as follows:

```
[MVHERMITEPROBLEMSYM, [P0,T0n,P1,T1n,S,L]]
```

Your first element is a name to identify the module. Its second element is again a list containing the elements of the geometry. In this case the name is „SymHermiteProb“ and the elements are two points with associated tangents, their intersection, and the distance of the points to the intersection.

Via the Get functions StartPoint, EndPoint, StartTangent, EndTangent, CrossPoint and ParameterL one can acquire the data and use it in other functions. The other functions can then be used to calculate with the data for the symmetric Hermite problem.

MHermiteProblemSym		
E	**New**	Data structure for a symmetric Hermite problem
E	**StartPoint**	reading the start point
E	**EndPoint**	Reading the end point command
E	**StartTangent**	reading the start tangent
E	**EndTangent**	Reading the end tangent
E	**ParameterL**	reading of the distance
E	**CrossPoint**	reading of the intersection point
E	**Plot2D**	Plotting of the symmetrical Hermite problem
E	**Create**	creation of a Sym. Hermite problem by 3 points
E	**BlendArc**	rounding of the corner point by an arc

3.5.9. Module MConstant

MConstant is a module for storing fixed constants and names to identify data structures. In the locally declared functions, starting with CV, the constants for declaring the different geometries are defined. These are names for recognising the geometry elements in the test file. In the globally declared functions, starting with Get, the names for identifying the geometry elements are returned.

MConstant

L	**NULLEPS**	constant to compare to zero
L	**CVPOINT**	constant for geometry elements: Point
L	**CVLINE**	constant for geometry elements: Line
L	**CVARC**	constant for geometry elements: Arc
L	**CVPOLYGON**	constant for geometry elements: polygon
L	**CVGEOLIST**	constant for geometry elements: GeoList
L	**CVHERMITEPROBLEM**	constant for geometry elements: HermiteProb
L	**CVHERMITEPROBLEMSYMMETRIC**	Const. for geometry elements: SymHermiteProb
L	**CVBIARC**	Const. for geometry elements: Biarc
E	**GetNullEps**	return of the zero comparison command.
E	**GetPoint**	identifier for points
E	**GetLine**	Identifier for lines
E	**GetArc**	identifier for circular arcs
E	**GetPolygon**	identifier for polygons
E	**GetGeoList**	Identifier for geometry lists
E	**GetHermiteProblemSymmetric**	Identifier for symmetric Hermite problems
E	**GetHermiteProblem**	ID for Hermite problems
E	**GetBiarc**	identifier for Biarcs

3.5.10. Module MGeneralMath

MGeneralMath is a module for general mathematical functions and works with the data structures and procedures.

	MeneralMath	
E	**MPoint**	Data structure for a point
E	**MPointX**	Reading of the x-coordinate for a point
E	**MPointY**	reading the y-coordinate for one point
L	**MPointIllustrateXY**	plot structure for a blue point
E	**MPointIllustrate**	plot structure for a blue point
E	**MPointPlot**	Illustration of a green point
E	**MLine**	Data structure for a line
E	**MLineStartPoint**	Reading of the start point for a draw frame
E	**MLineEndPoint**	Reading the end point for a line
E	**MPointOnLine**	Calculation of a point on the line
E	**MLinePlot2D**	Plot the part of a line starting at the starting point
. E	**MLineLine**	calculating the intersection of two linesline

3.5.11. Module Biarc

Data Structure

Requirements and specifications:

The Biarc is to be used to solve a Hermite problem. A Hermite problem is defined as follows:

Given two points P_0 and P_1 with associated normalised tangents $\vec{t}_0$ and $\vec{t}_1$. The biarc must connect the points such that the tangents of the start and end points of the biarc coincide with the tangents of the Hermites problem.

Data structure:

The data structure New for a biarc must contain two arcs.

So the data structure for one arc is needed: MArc:-New.

This in turn contains the coordinates for the centre, the radius, the start angle and the angle change.

3.5.12. Module MBiarc

MBiarc is a module for Biarcs and works with a data structure and with procedures/functions. The data structure New for a Biarc is a list which is structured as follows:

```
[MVBIARC, [Arc0, Arc1]]
```

Its first element is a name to identify the module. Your second element is again a list containing the elements of the geometry. In this case the name is „Biarc“ and the elements are two arcs.

Using the Get functions GetArc0 and GetArc1 one can capture the two arcs for the biarc. The functions listed below can be used to calculate the data for the biarc.

MBiarc		
E	`New`	Data structure for a biarc
E	`GetArc0`	Reading of the first arc
E	`GetArc1`	Reading the second arc
E	`Circle`	calculating the circle K_J from the Hermite problem data.
E	`Plot2DCircle`	plot of the circle K_J
E	`angle`	Calculate the angle from the centre of the circle to points on the circle
E	`Plot2D`	Plot of the biarc
E	`ConnectionPoint`	Calculation of the connection point J (Equal Chord)
E	`TangentTj`	Calculation of the tangent to J
E	`Tangent Biarc`	rotation of Tj for the biarc.
E	`BiarcCenter`	Calculate the centres of the arcs of the biarc
E	`Biarc`	calculate the centre of the biarc.
E	`Position`	Determine a point on the biarc
E	`Blend`	Calculate the biarc only from the Hermite problem

3.6. Programme Flowchart

3.6.1. Overall Flow

3.6.2. Verrundung der Kurve

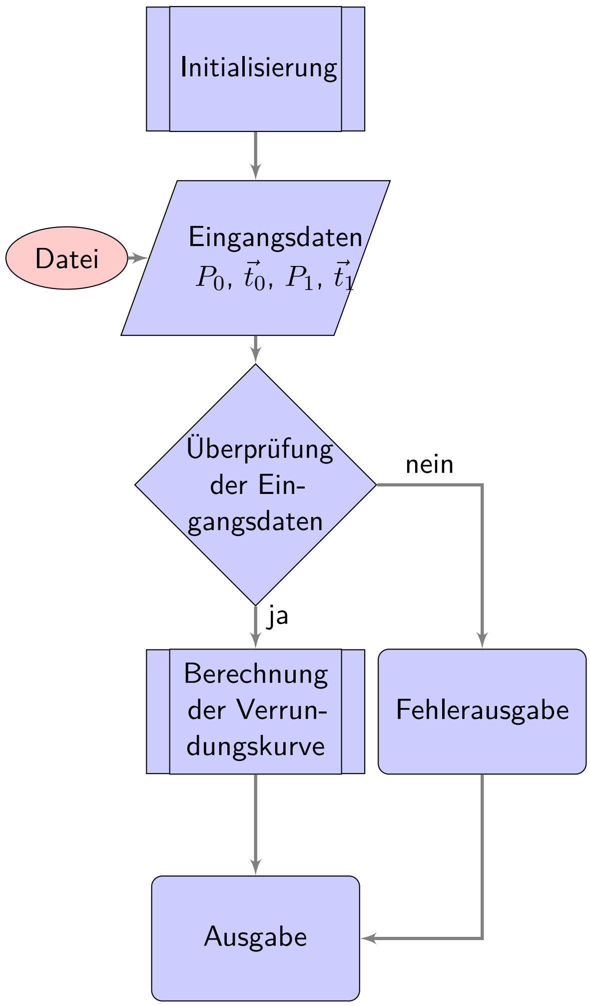


Figure 3.1.: Programme flow chart „Corner rounding“

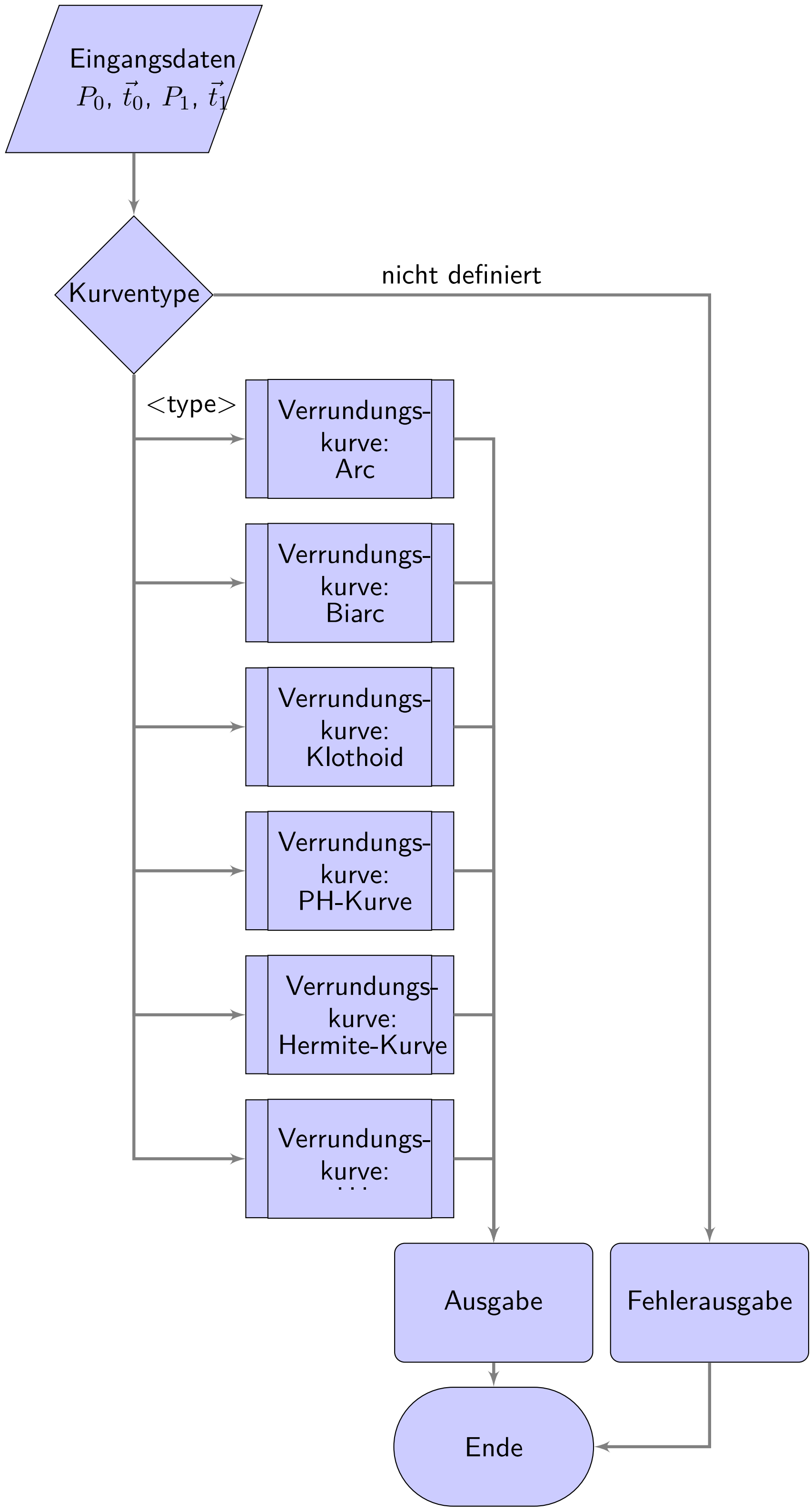


Figure 3.2.: Programme flowchart „Selection of the rounding strategies“

4. Representation of Python Programs

It is possible to integrate a part into the document, see 4.1. This is the most elegant method and is also preferable. Individual lines and line ranges can also be selected in the list of options. If a file is integrated, it is always as up-to-date as the document.

It may also be useful to integrate program lines, see 4.2.

The integration of programs with the help of images is pointless.

```
# Hello World for microcontroller boards
import pyb

redLED = pyb.LED(1) # built-in red LED
greenLED = pyb.LED(2) # built-in green LED
blueLED = pyb.LED(3) # built-in blue LED
while True:
    # Turns on the red LED
    redLED.on()
    # Makes the script wait for 1 second (1000 miliseconds)
    pyb.delay(1000)
    # Turns off the red LED
    redLED.off()
    pyb.delay(1000)
    greenLED.on()
    pyb.delay(1000)
    greenLED.off()
    pyb.delay(1000)
    blueLED.on()
    pyb.delay(1000)
    blueLED.off()
    pyb.delay(1000)
```

Listing 4.1.: The program “Hello World” in Python for microcontroller boards is inserted from the file `Blink.py`.

```python
# Hello World for microcontroller boards
import pyb

redLED = pyb.LED(1) # built-in red LED
greenLED = pyb.LED(2) # built-in green LED
blueLED = pyb.LED(3) # built-in blue LED
while True:
    # Turns on the red LED
    redLED.on()
    # Makes the script wait for 1 second (1000 miliseconds)
    pyb.delay(1000)
    # Turns off the red LED
    redLED.off()
    pyb.delay(1000)
    greenLED.on()
    pyb.delay(1000)
    greenLED.off()
    pyb.delay(1000)
    blueLED.on()
    pyb.delay(1000)
    blueLED.off()
    pyb.delay(1000)
```

Listing 4.2.: The program “Hello World” in Python for microcontroller boards has been inserted directly into the LaTeX file..

5. First Chapter

...

A Speicherprogrammierbare Steuerung (SPS) is ...

A Computerized Numerical Control (CNC) needs a SPS (PLC) to ...

6. CAGD

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

The book CAGD by Gerald Farin is a classic on splines. [Far02]

The standard 66025 for programming CNC machines is also a classic; however, it does not deal with splines. [DIN66025-2]

Mr. F. Farouki has dealt with both CNC machine programming and splines. His article[1] on a real-time interpolator also shows this. [FS17]

A new dimension of machine tools have emerged with the invention of 3D printers. [Rus+07]

Another aspect of automation technology is communication. Another milestone has been reached with 5G technology. another milestone has been reached.[2]

[1] co-author is J. Srinathu

[2] Zaf+20.

A. drawings with tikz

The package tikz is a powerful tool for creating graphics. Many introductions exist. Here only the first steps are shown, so that you can easily create flowcharts.

Drawing a line and arrows

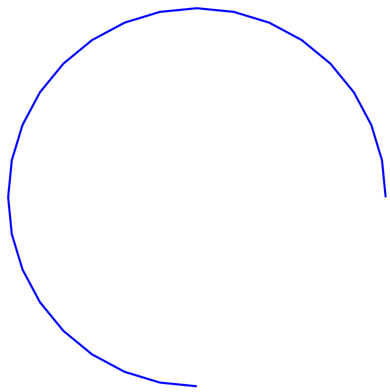
```
\begin{tikzpicture}
    \draw (0,0) -- (1,0);

    \draw[->] (2,0) -- (3,0);

    \draw[<->] (5,0) -- (6,0);
\end{tikzpicture}
```


Drawing a thick blue line

```
\begin{tikzpicture}
    \draw[line width=2pt, blue] (0,0) -- (1,0);

    \draw[line width=2pt, red, dotted] (2,0) -- (3,0);

    \draw[line width=2pt, dashed, green] (4,0) -- (5,0);

    \draw [thick,dash dot] (0,1) -- (5,1);
    \draw [thick,dash pattern={on 7pt off 2pt on 1pt off 3pt}] (0,2)
\end{tikzpicture}
```

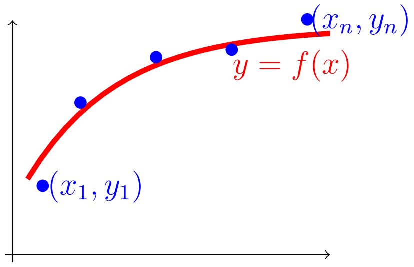
Drawing an arc

```
\begin{tikzpicture}
  \draw [blue,thick,domain=0:270] plot ({5+2.5*cos(\x)}, {1+2.5*sin(\
\end{tikzpicture}
```



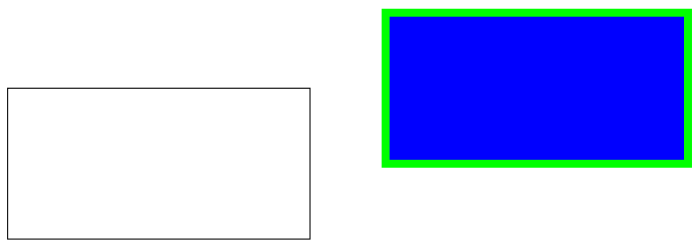
Draw a function

```
\begin{tikzpicture}[
    declare function={%
        F(\x)                    =3-2*pow(2.7979,-0.8*\x);
    }
 ]


    % Zeichnen der Funktion
    \draw[red,line width=2pt,domain=0:4] plot({\x},{F(\x)});

    % Bezeichnung
    \node[red] (O) at(3.5,2.5) {$y=f(x)$};

    % Punkte
    \node[blue] (P1n) at (0.9,0.9) {$(x_1,y_1)$};
    \node[blue] (P1) at (0.2,0.9) {$\bullet$};

   \node[blue] (P2) at (2.7,2.7) {$\bullet$};
   \node[blue] (P3) at (1.7,2.6) {$\bullet$};
   \node[blue] (P4) at (0.7,2) {$\bullet$};

   \node[blue] (P5n) at (4.4,3.1) {$(x_n,y_n)$};
   \node[blue] (P5) at (3.7,3.1) {$\bullet$};

   % Koordinatensystem
   \draw[color=black,->] (-0.2,-0.1) -- (-0.2,3.1);
   \draw[color=black,->] (-0.3,0) -- (4,0);
\end{tikzpicture}
```



Drawing rectangles and moving objects

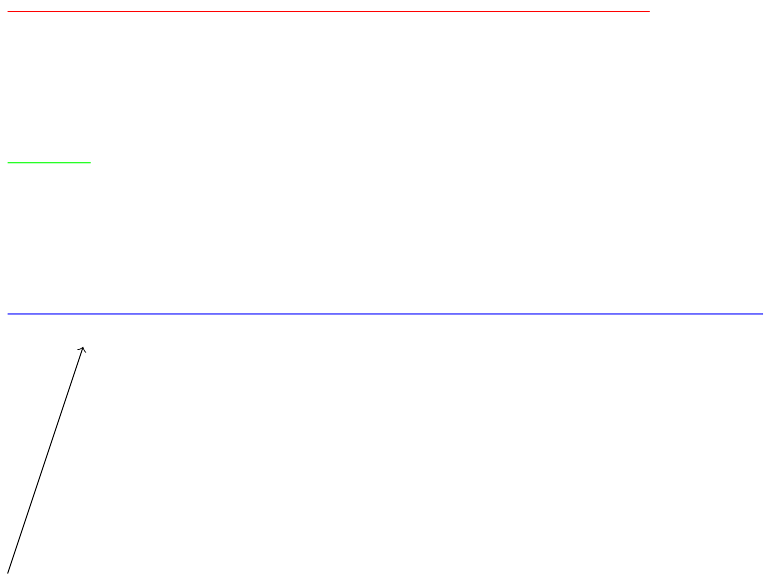
```
\begin{tikzpicture}
    \draw (0,0) -- (4,0)  -- (4,2) -- (0,2) -- cycle;

    \begin{scope}[shift={(5,1)}]
      \draw[green, fill=blue, line width=3pt] (0,0) -- (4,0)
-- (4,2) -- (0,2) -- cycle;
    \end{scope}
\end{tikzpicture}
```



Use of variables

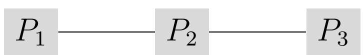
```
\begin{tikzpicture}
\pgfmathsetmacro{\PHI}{-15}
% Now use \PHI anywhere you want -15 to appear,
% can also be used in calculations like 2*\PHI
\def\x{10};

\draw[red] (0,4) -- (1-\PHI*0.5,4);
\draw[green] (0,2) -- (1+1/\x,2);
\draw[blue] (0,0) -- ({1+3*(\x/5+1)},0);
\end{tikzpicture}

\bigskip

%\usetikzlibrary{math} %needed tikz library

\begin{tikzpicture}
  \pgfmathsetmacro{\drehpunktx}{11.63815573}
```

```
  %Variables must be declared in a tikzmath environment but
  % can be used outside
  \tikzmath{\x1 = 1; \y1 =1;
  %computations are also possible
  \x2 = \x1 + 1; \y2 =\y1 +3; }
  \draw[->] (\x1, \y1)--(\x2, \y2);
\end{tikzpicture}
```



Use of points

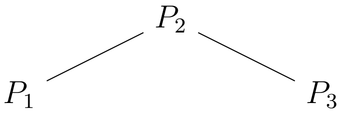
```
%\usetikzlibrary{backgrounds} % is needed

  \begin{tikzpicture}
    \node [fill=gray!30] (P1) at (0,0) { $P_1$ };
    \node [fill=gray!30] (P2) at (2,0) { $P_2$ };
    \node [fill=gray!30] (P3) at (4,0) { $P_3$ };

    \begin{scope}[on background layer]
        \draw (P1) -- (P3);
    \end{scope}
  \end{tikzpicture}
```



Use of nodes

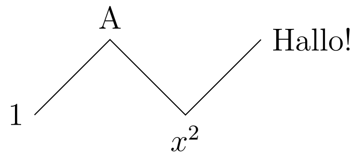
```
  \begin{tikzpicture}
    \node (P1) at (0,0){$P_1$};
    \node (P2) at (2,1){$P_2$};
    \node (P3) at (4,0){$P_3$};
```

```
    \begin{scope}
        \draw (P1) -- (P2) -- (P3);
    \end{scope}
\end{tikzpicture}
```



Use of nodes

```
\begin{tikzpicture}
  \draw (0, 0) node[left]{1}
        -- ++(1, 1) node[above] {A}
        -- ++(1,-1) node[below] {$x^2$}
        -- ++(1, 1) node[right] {Hallo!};
\end{tikzpicture}
```



B. Criteria for a good LaTeX project

A good report is not only characterised by good content, but also fulfils formal aspects. The following list should help to comply with basic rules. Before submitting, all points should be checked and ticked off.

- ☐ Is a suitable directory structure used?
- ☐ Is an appropriate division into files used?
- ☐ Are meaningful directory and file names used?
  - ☐ Are directory and file names such as report or term paper avoided?
  - ☐ Are meaningful names used for images and not „image1"?
  - ☐ Are directory and file names not too long?
  - ☐ Are special characters and spaces avoided?
- ☐ Are commands like \newline and \\ avoided?
- ☐ Are the LaTeXfiles clearly laid out?
  - ☐ Are indentations used in the LaTeXfiles?
  - ☐ Are free lines inserted?
  - ☐ Is the project mentioned in the header of the files?
  - ☐ Does the header of the files mention the main sources?
  - ☐ Is the author mentioned in the header of the files?
- ☐ Are all necessary files given?
- ☐ Are the temporary files deleted?
- ☐ Are graphics created by yourself?
- ☐ Are the sources of the images given?
- ☐ Are citations made with the command \cite?
- ☐ Is a bib file used?

- ☐ Are the entries of the bib-file clearly arranged?
- ☐ Are the entries of the bib-file edited to show them correctly?
- ☐ Are the correct types used for the bib entries?
- ☐ Are meaningful keys used in the bib files?

Unfortunately, the list cannot be complete, but it provides some clues.

C. Model Card

- Model Details - Basic information about the model
  - Person or organization developing model
  - Model date
  - Model version
  - Model type
  - Information about training algorithms, parameters, fairness constraints or other applied approaches, and features
  - Paper or other resource for more information
  - Citation details
  - License
  - Where to send questions or comments about the model
- Intended Use. Use cases that were envisioned during development.
  - Primary intended uses
  - Out-of-scope use cases
- Factors: Factors could include demographic or phenotypic groups, environmental conditions, technical attributes, or others
  - Relevant factors
  - Evaluation factors
- Metrics: Metrics should be chosen to reflect potential real world impacts of the model.
  - Model performance measures
  - Decision thresholds
  - Variation approaches
- Evaluation Data: Details on the dataset(s) used for the quantitative analyses in the card.
  - Datasets
  - Motivation
  - Preprocessing

- Training Data: May not be possible to provide in practice
  - When possible, this section should mirror Evaluation Data. If such detail is not possible, minimal allowable information should be provided here, such as details of the distribution over various factors in the training datasets.
- Quantitative Analyses
  - Unitary results
  - Intersectional results
- Ethical Considerations
- Caveats and Recommendation

D. List of Material

E. SBOM

requirements.txt

F. Methodology

Domain Knowledge:

- Part Domain, e.g. HW, application
- Part technologies, e.g. algorthms
- Part Tools, e.g. IDE

G. doxygen - Example

Bibliography

[DIN66025-2] DIN Deutsches Institut für Normung e.V. *DIN 66025-2:1988-09: Programmaufbau für numerisch gesteuerte Arbeitsmaschinen: Wegbedingungen und Zusatzfunktionen.* Norm. Berlin, Sept. 1988.

[FG19] R. Fildes and P. Goodwin. "Retail forecasting: Research and practice". In: *International Journal of Forecasting* 35.1 (2019), pp. 1–9. DOI: `10.1016/j.ijforecast.2018.09.008`. URL: `https://www.sciencedirect.com/science/article/abs/pii/S016920701930192X`.

[FS17] R. Farouki and J. Srinathu. "A real-time CNC interpolator algorithm for trimming and filling planar offset curves". In: *Computer-Aided Design* 86 (Jan. 2017). DOI: `10.1016/j.cad.2017.01.001`.

[Far02] G. Farin. *Curves and Surfaces for CAGD*. 5. [pdf]. San Diego, CA: Academic Press, 2002.

[Jak+10] G. Jaklic et al. "On Interpolation by Planar Cubic G^2 Pythagorean-Hodograph Spline Curves". In: *Mathematics of Computation* (2010). [pdf].

[Loy17] J. D. Loyal. "The Walmart Sales Project". In: *Unpublished Manuscript* (2017). URL: `https://joshloyal.github.io/assets/pdf/forecasting_intro.pdf`.

[MMH18] T. S. McElroy, B. C. Monsell, and R. J. Hutchinson. "Modeling of Holiday Effects and Seasonality in Daily Time Series". In: *U.S. Census Bureau Working Papers* (2018). URL: `https://www.census.gov/content/dam/Census/library/working-papers/2018/adrm/rrs2018-01.pdf`.

[PS17] J. J. Pao and D. S. Sullivan. "Time Series Sales Forecasting". In: *CS229: Machine Learning Final Projects* (2017). URL: `https://cs229.stanford.edu/proj2017/final-reports/5244336.pdf`.

[Rus+07] D. Russell et al. *Apparatus and Methods for 3D Printing*. United States Patent, Patent N0.: US 7,291,002 B2. 2007.

[Wat17a] Waterloo Maple Inc. 2017. *Description*. letzter Zugriff 14.09.2017. 2017. URL: `https://de.maplesoft.com/support/help/Maple/view.aspx?path=module/description`.

[Wat17b] Waterloo Maple Inc. 2017. *Export*. letzter Zugriff 14.09.2017. 2017. URL: `https://de.maplesoft.com/support/help/Maple/view.aspx?path=module/export`.

[Wat17c] Waterloo Maple Inc. 2017. *Module*. letzter Zugriff 14.09.2017. 2017. URL: `https://de.maplesoft.com/support/help/maple/view.aspx?path=module`.

[Wat17d] Waterloo Maple Inc. 2017. *ModuleLoad*. letzter Zugriff 14.09.2017. 2017. URL: `https://de.maplesoft.com/support/help/Maple/view.aspx?path=ModuleLoad`.

[Wat17e] Waterloo Maple Inc. 2017. *Option*. letzter Zugriff 14.09.2017. 2017. URL: `https://de.maplesoft.com/support/help/Maple/view.aspx?path=module/option`.

[Wat17f] Waterloo Maple Inc. 2017. *Procedures*. letzter Zugriff 14.09.2017. 2017. URL: `https://de.maplesoft.com/support/help/Maple/view.aspx?path=procedure`.

[Zaf+20] A. Zafeiropoulos et al. "Benchmarking and Profiling 5G Verticals' Applications: An Industrial IoT Use Case". In: *IEEE Conference on Network Softwarization, NetSoft 2020*. 2020. DOI: `10.1109/NetSoft48620.2020.9165393`.

[Zha21] J. Zhang. "Sales Prediction of Walmart Based on Regression Models". In: *Proceedings of the 2021 International Conference on Computers, Information Processing and Advanced Education*. 2021, pp. 294–298. DOI: `10.1145/3456887.3459308`. URL: `https://www.atlantis-press.com/article/125994715.pdf`.

Index