



Data Science and Analytics

ORG Data Thread and Weakness Analysis

Student 1: Vijay Singh (7025700)
Student 2: Manoj S (7025700)
Student 3: Vatsal Mahajan (7025700)
Course of Studies: Industrial Informatics

First examiner: Prof. Dr. Elmar Wings
Second examiner: Prof. Dr. Walter Colombo

Submission date: January 5, 2025

Contents

Acronyms	ix
1. Introduction	1
1.1. Case:	1
1.2. Challenges and Results:	1
2. Positioning Your Analytics	3
2.1. Existing Solutions:	3
2.2. State-of-the-Art:	3
2.2.1. Overview of Current Solutions:	3
2.2.2. Capabilities and Limitations:	3
2.2.3. Relevance to Analytics:	3
2.2.4. Emerging Trends:	4
3. Application Sector	5
3.1. Application Sector:	5
4. Purpose of Your Analytics	7
4.1. What It Does:	7
5. Positioning Within Standard Reference Architectures	9
6. Connected Infrastructure Components	11
6.1. Infrastructure Integration:	11
7. Major Requirements	13
7.1. Requirements:	13
8. Knowledge Discovery in Databases (KDD) Process	15
8.1. Topic Description	15
8.1.1. Special Challenges	15
8.2. Database	16
8.3. Data Selection	16
8.3.1. Origin	16
8.3.2. Data Format	16
8.3.3. Features	17
8.3.4. Size	17

Contents

8.4. Data Preparation	17
8.4.1. Merging and Conversion	17
8.4.2. Initial Cleaning	17
8.4.3. Key Attributes	18
8.5. Data Transformation	18
8.6. Data Mining	19
8.6.1. Feature Extraction	19
8.6.2. Clustering	19
8.6.3. Anormalities	20
8.7. Model	20
8.7.1. DBSCAN	20
8.7.2. Isolation Forest	20
8.8. Validation/Verification	21
8.9. Data Visualization	21
8.10. Conclusion	21
8.11. Allgemeine mathematische Beschreibung Bézier-Kurve	22
9. Verrundung mit einem Kreisbogen	25
9.1. Gleichungen	25
9.2. Bewertung	27
9.3. Examples	29
10. Maple files	33
10.1. Geometries with Maple	33
10.2. Geometry elements used	33
10.3. Building the data structure	34
10.4. Structure of a module	35
10.4.1. General structure of a module	37
10.4.2. Saving a module	38
10.4.3. Verwendung eines Moduls	38
10.4.4. Creating a Maple module	38
10.5. Functions of the modules	39
10.5.1. Module MPoint	39
10.5.2. Module MLine	40
10.5.3. Module MArc	41
10.5.4. module MBezier	41
10.5.5. Module MPolygon	42
10.5.6. module MGeoList	43
10.5.7. Module MHermiteProblem	43
10.5.8. Module MHermiteProblemSym	44
10.5.9. Module MConstant	44
10.5.10. Module MGeneralMath	45
10.5.11. Module Biarc	46
10.5.12. Module MBiarc	46

10.6. Programme Flowchart	47
10.6.1. Overall Flow	47
10.6.2. Verrundung der Kurve	47
11.First Chapter	49
12.CAGD	51
A. drawings with tikz	53
B. Criteria for a good \LaTeX project	59

List of Figures

1.1. Original Data format	1
5.1. Mapping Functionalities in RAMI 4.0 and IIRA	10
8.1. KDD Process	16
8.2. Converted CSV file	18
8.3. Error log	18
8.4. Result after applying data transformation	19
8.5. Bernstein polynomial of degree 3	23
8.6. Bézier curve for example 8.11	24
9.1. Smoothing a corner with the help of an arc - triangle	25
9.2. Angular change with blending arc	28
9.3. Curvature progression for a blending arc	28
9.4. Maximum range for a blending arc of a circle - $L(\varepsilon = \text{const}, \alpha)$	28
9.5. Deviation when specifying the distance L when rounding with a circular arc	29
10.1. Programme flow chart „Corner rounding“	47
10.2. Programme flowchart „Selection of the rounding strategies“	48

List of Tables

Acronyms

CNC Computerized Numerical Control

SPS Speicherprogrammierbare Steuerung

1. Introduction

1.1. Case:

The focus of this analysis is to identify and mitigate potential threats and weaknesses within Orgadata's SimplyTag system. SimplyTag facilitates quick access to construction-related data through a web app, making it essential to safeguard against malicious actors attempting to exploit the system. This involves analyzing logs to detect suspicious activity and ensure data integrity.

1.2. Challenges and Results:

Challenges:

- Handling large-scale logs to pinpoint anomalies. 1.1
- Differentiating legitimate user activity from malicious attempts.
- Establishing efficient protocols to respond to detected threats.

Results Achieved:

- Implementation of enhanced monitoring protocols using trace IDs and HTTP status codes.
- Reduced data breaches by identifying and blocking unauthorized requests.

[illegible]

Figure 1.1.: Original Data format

2. Positioning Your Analytics

2.1. Existing Solutions:

The analytics developed for Orgadata's SimplyTag system are positioned uniquely when compared to conventional tools. Traditional solutions like **Intrusion Detection Systems** (IDS) and log monitoring platforms (e.g., Splunk, ELK Stack) offer general frameworks for detecting anomalies and breaches. However, they often lack customization tailored to specific applications. Orgadata's approach is distinguished by its precise utilization of trace IDs, HTTP status codes, user agent patterns, and paths to monitor requests in real-time.

2.2. State-of-the-Art:

2.2.1. Overview of Current Solutions:

- Existing tools such as Splunk, ELK Stack, and Intrusion Detection Systems (IDS) provide robust frameworks for monitoring and analyzing security events. These solutions excel at processing vast amounts of log data and identifying potential threats in general scenarios.
- Security platforms often use rule-based or heuristic approaches for anomaly detection but may struggle with domain-specific customizations.

2.2.2. Capabilities and Limitations:

- **Capabilities:** Tools like Splunk and ELK Stack provide scalability, integration options, and comprehensive dashboards for security analytics. IDS focuses on real-time threat detection.
- **Limitations:** Lack of precise tailoring for SimplyTag's requirements, such as analyzing specific HTTP status codes and user agent patterns. High false-positive rates and inability to integrate seamlessly with Orgadata's infrastructure are additional challenges.

2.2.3. Relevance to Analytics:

The SimplyTag analytics focus on filling these gaps by leveraging domain-specific insights, such as monitoring trace IDs and HTTP response patterns to identify

2. Positioning Your Analytics

anomalies and unauthorized activity.

1. The use of trace IDs enables seamless tracking of individual requests across logs, allowing for detailed insights into potential vulnerabilities.
2. By monitoring HTTP status codes, the system identifies and flags suspicious patterns, such as unexpected 200 codes for unauthorized paths.
3. Integration of user agent analysis ensures that illegitimate devices or configurations can be quickly identified and addressed.

2.2.4. Emerging Trends:

- AI-driven anomaly detection is gaining traction, enabling systems to learn and adapt to evolving threats.
- Zero-trust architecture and real-time monitoring advancements align with SimplyTag's goals of enhancing data security and operational reliability.

This focused methodology bridges gaps left by traditional systems, providing a solution that is both targeted and scalable for SimplyTag's operational needs.

3. Application Sector

3.1. Application Sector:

This project belongs to the Industry domain, specifically the Construction Software sector, with a focus on::

- **Threat Detection:** Monitoring system logs to identify malicious activity and unauthorized data access.
- **Data Integrity Assurance:** Safeguarding construction-related data from breaches to ensure accurate and reliable information.
- **Operational Security Optimization:** Enabling proactive measures to address potential weaknesses and improve system resilience.

By contributing to the broader domain of secure digital construction tools, the project aligns with emerging trends like data-driven operational efficiency and advanced cybersecurity measures tailored for industry-specific applications.

4. Purpose of Your Analytics

4.1. What It Does:

This project belongs to the Industry domain, specifically the Construction Software sector, with a focus on:

- **Condition Monitoring:** Tracks HTTP requests in real-time to identify anomalies. This includes analyzing user agents and status codes to pinpoint irregularities that could indicate potential breaches or vulnerabilities.
- **Prevention:** Identifies and blocks malicious requests before they lead to breaches. Proactive security measures ensure that sensitive data and system integrity remain uncompromised.
- **Diagnosis:** Conducts post-incident analysis to refine future detection capabilities. By learning from past incidents, the analytics evolve to handle emerging threats more effectively.
- **Optimization:** Enhances system performance by ensuring secure operations without disrupting user experience. This involves balancing security measures with system usability, especially in high-demand environments like construction software solutions.

5. Positioning Within Standard Reference Architectures

This solution aligns with several modern frameworks and architectures:

- **Smart-City:** The analytics align with secure data handling standards in urban development contexts, ensuring efficient and secure information exchange in large-scale urban projects..
- **HoT/IIRA:** The approach supports reliability and security within industrial systems, adhering to frameworks like the Industrial Internet Reference Architecture (IIRA) for seamless integration and operational consistency. 5.1
- **RAMI 4.0:** Within the Reference Architecture Model for Industry 4.0 (RAMI 4.0), the analytics are positioned in the "Information Layer," where data processing, analysis, and anomaly detection occur. It also contributes to the "Functional Layer" by enabling actionable insights and decision-making based on real-time monitoring. 5.1
- **ISO Standards Compliance:** The analytics are developed in alignment with international standards such as ISO 27001, ensuring robust security practices and data protection.
- **Cybersecurity Frameworks:** Incorporating principles from the NIST Cybersecurity Framework, the analytics bolster detection, prevention, and response mechanisms for threats.

These alignments ensure that **SimplyTag analytics** not only address Orgadata's specific needs but also conform to global benchmarks, making them scalable and applicable across various industrial and urban contexts.

5. Positioning Within Standard Reference Architectures

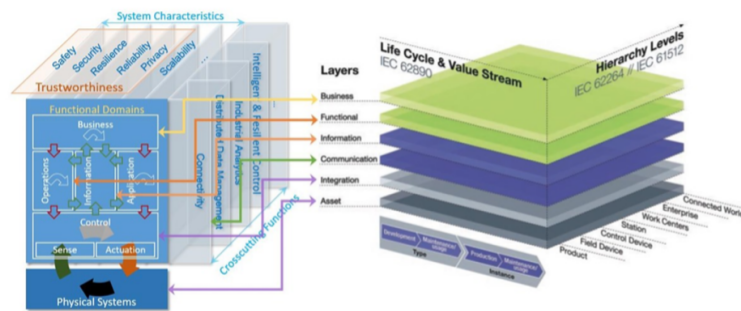


Figure 5.1.: Mapping Functionalities in RAMI 4.0 and IIRA

6. Connected Infrastructure Components

6.1. Infrastructure Integration:

- **SimplyTag API Backend:** Serves as the primary source for log data. It collects and provides detailed information about HTTP requests, including trace IDs and status codes, for analysis.
- **Orgadata's Logikal System:** Acts as the operational backbone for data-driven decision-making. This system contextualizes log data by linking it with construction-related project details, enabling precise threat detection.
- **External Threat Detection Tools:** To enhance the analytics, integration with external tools such as SIEM platforms or advanced machine learning models for threat prediction can be implemented. These tools offer additional layers of analysis and improve overall system resilience.
- **Cloud Infrastructure:** SimplyTag leverages cloud infrastructure to ensure scalability and reliability. The cloud platform supports large-scale log storage and computational power required for real-time analytics.
- **Mobile and Web Interfaces:** The analytics extend to user-facing interfaces like the SimplyTag mobile and web apps, ensuring seamless user interaction while maintaining robust security measures.

By connecting these components, the analytics create a comprehensive security ecosystem that is robust, scalable, and aligned with Orgadata's operational goals.

7. Major Requirements

7.1. Requirements:

- **Structural:** the analytics must seamlessly integrate with existing SimplyTag and Orgadata infrastructure.
- **Behavioral:** Real-time anomaly detection with high accuracy and minimal false positives.
- **Functional:** Efficient parsing of HTTP status codes and trace IDs to identify suspicious activity.
- **Technological:** Implementation of advanced algorithms for log analysis and anomaly detection.

This structured approach ensures that the Threat and Weakness Analysis for Orgadata's SimplyTag system is robust, effective, and well-positioned within industry standards.

8. Knowledge Discovery in Databases (KDD) Process

KDD plays a pivotal role in helping organizations navigate the over whelming volumes of data generated in today's information-driven world. KDD is a structural approach to data analysis to discover and make explicit knowledge available in extensive data sets [Wings:2024]. This methodology involves a series of well-defined steps, including data selection, preprocessing, transformation and analysis, leading to the generation of actionable knowledge as shown in the figure 8.1. [KDD:2000]

KDD is particularly relevant for analyzing log file data in the context of threat and weakness analysis. The structured approach ensures that patterns and anomalies are identified efficiently, enabling the identification of targeted solutions for improving system security and operational performance.

In this project, the KDD process will serve as the backbone for organizing and implementing analytics tasks, ensuring a systematic exploration of the data to uncover valuable insights.

8.1. Topic Description

This section of the report outlines the objective to focuses on identifying anomalies, vulnerabilities, and inefficiencies within the dataset. The primary goal is to uncover potential security risks, operational weaknesses, and performance bottlenecks through a comprehensive analysis of system logs. The Knowledge Discovery in Databases (KDD) process was utilized to manage the complexity of the dataset effectively. This structured methodology enabled efficient data handling, cleaning and preparation. By leveraging the KDD process, the analysis distinguished patterns indicative of normal and abnormal behavior, identified clusters of high-risk events based on suspicious trace of useragents. By combining statistical techniques, machine learning algorithms, and domain expertise, the analysis contributed significantly to the overall enhancement of system monitoring and management.

8.1.1. Special Challenges

- The challenges in this dataset includes efficiently handling the high volume of log entries.
- Additionally, the dataset used in this project was not labeled, which posed difficulties for supervised analysis.

8. Knowledge Discovery in Databases (KDD) Process

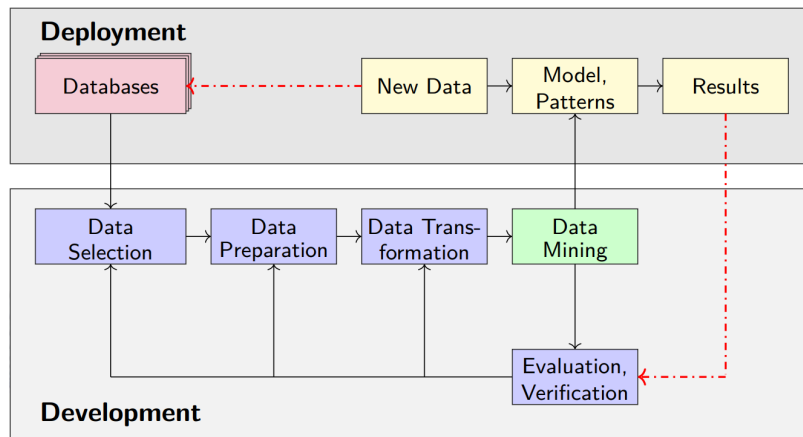


Figure 8.1.: KDD Process
[Wings:2024]

- Another challenge was that log entries were not sequentially arranged by **TraceId**, as logs are ordered by timestamp due to concurrent requests. Grouping and organizing requests using **TraceId** was necessary for accurate analysis, adding complexity to preprocessing.

8.2. Database

The database used in this project comprised of log files collected from the system. The analysis was performed collectively on nine log files dated 20th November, 21st November, 23rd November, 24th November, 26th November, 27th November, 29th November, 30th November, and 1st December, 2024.

8.3. Data Selection

Data selection is a critical step that directly influences the project's outcomes. It focuses on identifying relevant portions of the log files for analysis while filtering out irrelevant data to ensure only attributes contributing to the project objectives are prioritized.

8.3.1. Origin

Logs generated by monitoring tools and event management systems, specifically collected from the Graylog server.

8.3.2. Data Format

The file format is `.log`.

8.3.3. Features

The dataset includes log entries with the following attributes

- Timestamps
- PID
- Logger
- Message
- Scope (e.g., TraceId, RequestID)
- Application
- State
- EventID

8.3.4. Size

The dataset is approximately 649 MB size.

8.4. Data Preparation

Data preparation in the Knowledge Discovery in Databases (KDD) process, ensures the dataset is clean, consistent and ready for analysis.

8.4.1. Merging and Conversion

- The initial step involved merging all nine individual log files into a single consolidated file.
- Then the merged file, originally in format `.log`, were converted to CSV file to facilitate easier manipulation as shown in the figure 8.2.

8.4.2. Initial Cleaning

- Excluding entries without a valid TraceId that contain generic error or warning messages as shown in figure 8.3.

8. Knowledge Discovery in Databases (KDD) Process

A	B	C	D	E	F	G	H	I
Level	Timestamp	PID	Logger	Message	Scope	Application	State	EventId
INFO	2024-11-28T10:09:12.7685	3E+05	Microsoft.AspNet	Request starting HTTP/1.1 PATCH http://api.owds.org/ [SpanId: '9cf398563257a011', RequestId: '0HNBEI (Name: 'Ofcas.Datasafe.WebApi (Protocol: 'HTTP/1.1', Method: 'PATCH', 'C (Id: 1, Name: None)				
				Request: Protocol: HTTP/1.1 Method: PATCH Scheme: http PathBase: Path: /api/v1/identities/08DCFF11-2EBD-45C3-80E8-97552ACD7C62 Connection: close Host: api.owds.org User-Agent: Embarcadero URI Client/1.0 Authorization: [Redacted] Content-Type: application/json-patch+json Content-Length: 17 x-forwarded-proto: [Redacted]				
INFO	2024-11-28T10:09:12.7692	3E+05	Microsoft.AspNet	x-forwarded-port: [Redacted]	[SpanId: '9cf398563257a011', RequestId: '0HNBEI (Name: 'Ofcas.Datasafe.WebApi (Protocol: 'HTTP/1.1', Method: 'PATCH', 'S (Id: 1, Name: 'RequestLo			
INFO	2024-11-28T10:09:12.7701	3E+05	System.Net.Http	Start processing HTTP request GET https://id.orgada [SpanId: '1fa4aab449c01925', RequestId: '0HNBEI (Name: 'Ofcas.Datasafe.WebApi (HttpMethod: 'GET', 'Uri: 'https://id.orgada (Id: 100, Name: 'Request				
INFO	2024-11-28T10:09:12.7704	3E+05	System.Net.Http	Sending HTTP request GET https://id.orgada.com/ [SpanId: '1fa4aab449c01925', RequestId: '0HNBEI (Name: 'Ofcas.Datasafe.WebApi (HttpMethod: 'GET', 'Uri: 'https://id.orgada (Id: 100, Name: 'Request				
INFO	2024-11-28T10:09:12.7930	3E+05	System.Net.Http	Received HTTP response headers after 22.3768ms - [SpanId: '1fa4aab449c01925', RequestId: '0HNBEI (Name: 'Ofcas.Datasafe.WebApi (ElapsedMilliseconds: 22.3768, StatusCode (Id: 101, Name: 'Request				
INFO	2024-11-28T10:09:12.7935	3E+05	System.Net.Http	End processing HTTP request after 23.4104ms - 200 [SpanId: '1fa4aab449c01925', RequestId: '0HNBEI (Name: 'Ofcas.Datasafe.WebApi (ElapsedMilliseconds: 23.4104, StatusCode (Id: 101, Name: 'Request				
INFO	2024-11-28T10:09:12.8144	3E+05	Microsoft.AspNet	Executing endpoint 'Ofcas.Datasafe.WebApi.Controllers [SpanId: '9cf398563257a011', ParentId: '00000000 (Name: 'Ofcas.Datasafe.WebApi (EndpointName: 'Ofcas.Datasafe.WebApi.C (Id: 0, Name: 'ExecutingE				
INFO	2024-11-28T10:09:12.8140	3E+05	Microsoft.AspNet	Route matched with action = "UpdateIdentityV1", o [TraceId: '17d200ee2dcccace5dd7221c1b3568d7', (Name: 'Ofcas.Datasafe.WebApi (RouteData: 'action = "UpdateIdentityV1", (Id: 102, Name: 'Control				
INFO	2024-11-28T10:09:12.8155	3E+05	System.Net.Http	Start processing HTTP request GET https://id.orgada [TraceId: '17d200ee2dcccace5dd7221c1b3568d7', (Name: 'Ofcas.Datasafe.WebApi (HttpMethod: 'GET', 'Uri: 'https://id.orgada (Id: 100, Name: 'Request				
INFO	2024-11-28T10:09:12.8158	3E+05	System.Net.Http	Sending HTTP request GET https://id.orgada.com/ [TraceId: '17d200ee2dcccace5dd7221c1b3568d7', (Name: 'Ofcas.Datasafe.WebApi (HttpMethod: 'GET', 'Uri: 'https://id.orgada (Id: 100, Name: 'Request				

Figure 8.2.: Converted CSV file

Level	Timestamp	PID	Logger	Message	Scope
ERROR	2024-11-28T18:16:5	668	Microsoft.EntityFrameworkCore	An error occurred using the connection to database 'datasafe' on server '192.168.244.135'.	[ParentId: '0000000000000000', C

Figure 8.3.: Error log

8.4.3. Key Attributes

The following were the key parameters that were identified and prioritized for analysis,

- **TraceId**: Used to track individual requests across log entries, enabling the grouping and analysis of entire request flows.
- **HTTP Status Code**: Identifies the result of a request (e.g., success, client errors, or server errors). Codes such as 200, 404 and 500 provide critical insights into system health.
- **Path**: Represents the API endpoint or resource accessed during a request, useful for pinpointing affected components or services.
- **User-Agent**: Provides details about the client or system making the request, helping to identify trends in usage or unusual activity from specific sources.

8.5. Data Transformation

This section focuses on manipulating and reshaping the cleaned data into a structured format ready for analysis. The key transformation includes,

1. Grouping log entries by **TraceId** to reconstruct complete request flows.
2. Parsing relevant fields (**TraceId**, **HTTP Status Code**, **Path**, **User-Agent**) from JSON structures.
3. Transforming each row to represent a unique log entry with all associated fields (e.g., aligning **TraceId** with its corresponding attributes) as shown in figure 8.4.

	A	B	C	D
	Trace-id	HTTP Status Code	Path	User Agent
1	00000e83229182560bc00dc08d8d0895	200	/api/v1/nodes/08dd0fba-4f8b-4103-8a95-aed373124a23/ele	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G
2	0000ca9c0504724ec0352ca2d927e26	204	/api/v1/references/HTTPS:%2F%2FOWDS.ORG%2FAG.KZF4I	Mozilla/5.0 (Linux; Android 14; SM-P620 Build/UP1A.231005.007; wv) AppleWe
3	0000d58f79c6040f625be46853e6143f	200	/api/v1/references/d5731268-72f8-4350-8b42-d44b4fa4efe1/codes	
4	0001693d167addef160ab0ddfb31d8	200	/api/v1/services/8054f549-6c59-4191-afb3-d31efc46f17e	Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrc
5	0001c4d4a65e06934a9e38f0f71b6496	204	/api/v2/nodes/ebf9e23b-3b78-4c30-9dab-cf963fb01f1a/pro	Mozilla/5.0 (Linux; Android 14; SM-S911B Build/UP1A.231005.007; wv) AppleWe
6	0001d8ca5b76e77d80a9a3ec83903f7d	204	/api/v1/nodes/08dac0aa-8ed5-4ee8-866f-1c22990e0ef0/ele	Mozilla/5.0 (iPhone; CPU iPhone OS 17_4 like Mac OS X) AppleWebKit/605.1.15
7	0001fbbee4d73c2f6a9b28dde44c8e77	200	/api/healthz	curl/8.5.0
8	000273de5951b79ad885cc2de52675be	200	/api/healthz	curl/8.5.0
9	0002b1825f778a4cb8e0f56e765cc	204	/api/v2/assets/650a0fb5-2e26-44c4-8055-2e7a691e1f9b/no	Mozilla/5.0 (iPhone; CPU iPhone OS 18_1_1 like Mac OS X) AppleWebKit/605.1.1
10	0002f7cc5075af12e46bd55ee4636d81	200	/api/v1/references/43bdf33e-7709-4e25-bbbb-848e309714bb/codes	
11	000302bc46d6f6b65289bc235560acc8f	200	/api/v2/versions	check_http/2.4.0 (monitoring-plugins 2.4.0)
12	00030a9eb3eaa3e54a28873d94e81ea	200	/api/v2/versions	check_http/2.4.0 (monitoring-plugins 2.4.0)
13	000360463cc19ebcad352f42441055b3	200	/api/v2/versions	check_http/2.4.0 (monitoring-plugins 2.4.0)
14	000379bd694343567e14425b1985acfe	200	/api/v1/elements/d5fd8aa5-89e1-4f7e-9dbf-0f180ef64ceb/	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like G
15	000439937e3f6ec27d4ee47d56d33d57	200	/api/v1/references/HTTPS:%2F%2FOWDS.ORG%2FAG.M9FB	Mozilla/5.0 (Linux; Android 14; moto g14 Build/UTLB34.102-54-1; wv) AppleWe
16	000472fc41c4d21ae9516f91ec850212	200	/api/v1/elements/81dbd634-7fff-4bc3-b04c-c7dcdad5383d/	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:132.0) Gecko/20100101 Firefox/

Figure 8.4.: Result after applying data transformation

8.6. Data Mining

Data mining involves extracting hidden patterns and anomalies from datasets. In this project, clustering and anomaly detection were utilized to uncover system irregularities and potential security threats efficiently.

8.6.1. Feature Extraction

Feature extraction involves converting raw data into a format that can be processed by machine learning algorithms.

- In this project, TF-IDF (Term Frequency-Inverse Document Frequency) Vectorization technique is used, to convert textual data (like the Path attribute in the log data) into numerical features. This allows the model to understand which paths are accessed frequently and which are less common, making it easier to identify unusual patterns.
- TF-IDF vectors were combined with additional numeric attributes such as HTTP status codes, unique paths and user-agents.

8.6.2. Clustering

Clustering was performed using DBSCAN (Density-Based Spatial Clustering of Applications with Noise):

- DBSCAN groups requests that shared similar patterns, such as accessing the same paths with similar HTTP Status Codes and User-Agents were grouped into clusters.
- Requests that did not belong to any cluster were flagged as anomalies (DBSCAN_Cluster = -1).

8.6.3. Anormalities

Along with clustering, Anomaly detection focuses on identifying unusual behaviors that deviate significantly from the norm using Isolation forest.

- The model analyzed the combined features (TF-IDF vectors and numeric attributes) to detect anomalous requests.
- By integrating DBSCAN and Isolation Forest approach, robust anomaly detection was ensured through both density-based and statistical methods.

8.7. Model

The model selection was driven by the need to perform both clustering and anomaly detection effectively. A combination of unsupervised and ensemble-based learning techniques was employed to analyze request patterns and identify anomalies.

8.7.1. DBSCAN

DBSCAN is a density-based clustering algorithm, groups data points that are closely packed together based on a density threshold. Data points far from any dense cluster are treated as "noise" or anomalies.

Key Parameters

- **eps**: Set to 0.5, defining the maximum distance between points to form a cluster.
- **min_samples**: Set to 5, specifying the minimum number of points required to form a cluster.

Outcome

Requests not assigned to any cluster (cluster = -1) were flagged as anomalies.

8.7.2. Isolation Forest

Isolation Forest is an ensemble learning technique designed to detect outliers by isolating data points. It works on the principle that anomalies are easier to isolate because they have unique features.

Key Parameters

- **n_estimators**: Set to 100, specifying the number of decision trees.
- **contamination**: Set to 0.01, indicating the proportion of expected anomalies in the dataset.
- **random_state**: Ensured reproducibility of results with a fixed seed value (42).

Outcome

Requests identified as anomalies were flagged with a value of -1.

8.8. Validation/Verification

8.9. Data Visualization

8.10. Conclusion

8.11. 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 n^{th} degree** is defined by:

$$b_{i,n}(\lambda) = \binom{n}{i} (1 - \lambda)^{n-i} \lambda^i, \quad \lambda \in [0, 1], \quad i = 0, \dots, n \quad (8.1)$$

Definition. The binomial coefficient is defined by:

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}, \quad i = 0, \dots, n \quad (8.2)$$

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. [Farin:2002]

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

$$b_{3,0}(\lambda) = \binom{3}{0} (1 - \lambda)^{3-0} \lambda^0 = (1 - \lambda)^3$$

$$b_{3,1}(\lambda) = \binom{3}{1} (1 - \lambda)^{3-1} \lambda^1 = 3\lambda(1 - \lambda)^2$$

$$b_{3,2}(\lambda) = \binom{3}{2} (1 - \lambda)^{3-2} \lambda^2 = 3\lambda^2(1 - \lambda)$$

$$b_{3,3}(\lambda) = \binom{3}{3} (1 - \lambda)^{3-3} \lambda^3 = \lambda^3$$

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

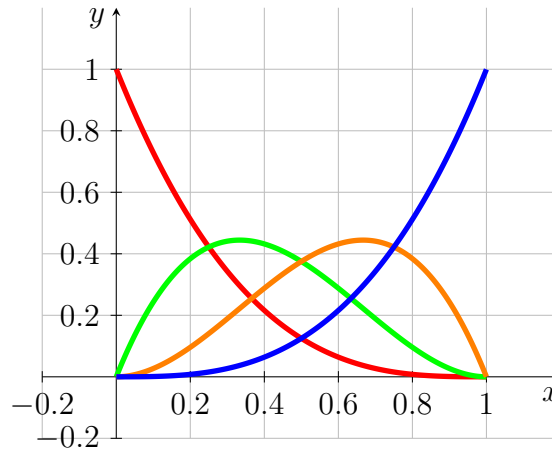


Figure 8.5.: Bernstein polynomial of degree 3

Definition. Given are the points

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

A **Bézier curve** is then defined by

$$C(\lambda) = \sum_{i=0}^n Q_i \cdot b_{i,n}(\lambda) \quad (8.3)$$

The points $Q_i, i = 0, \dots, n$ 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:

$$Q_0 = P_0, \quad Q_1 = P_0 + \lambda_0 \vec{t}_0, \quad Q_2 = P_1 - \lambda_1 \vec{t}_1, \quad Q_3 = P_1 \quad (8.4)$$

[Jaklic:2010]

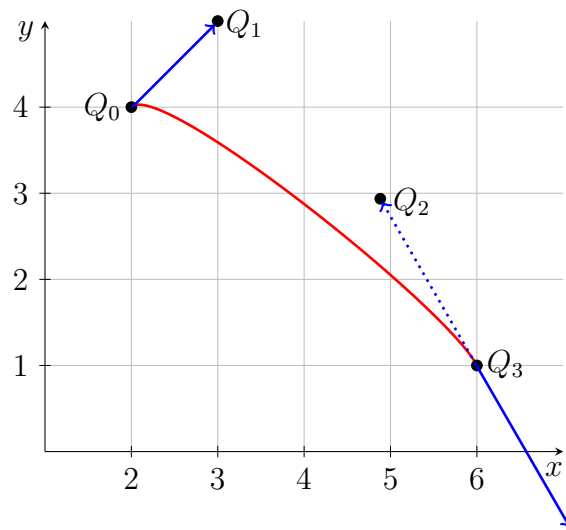


Figure 8.6.: Bézier curve for example 8.11

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:

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

$$Q_2 = P_1 - \vec{t}_1 = \begin{pmatrix} 6 \\ 1 \end{pmatrix} - \begin{pmatrix} 1 \\ -2 \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \end{pmatrix}, \quad Q_3 = P_1 = \begin{pmatrix} 6 \\ 1 \end{pmatrix}$$

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

9. Verrundung mit einem Kreisbogen

9.1. 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 $HP(L, \alpha)$.

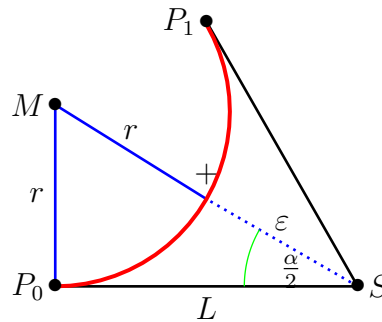


Figure 9.1.: Smoothing a corner with the help of an arc - triangle

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:

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

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

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

9. Verrundung mit einem Kreisbogen

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

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

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:

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

$$M = P_0 + \text{sign}(\alpha) \cdot r \cdot \vec{t}_0^\perp = \text{sign}(\alpha) \cdot r \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

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)

$$\vec{t}_0 = -\vec{t}_1 \Leftrightarrow \alpha = 0$$

- c)

$$\vec{t}_0 = \vec{t}_1 \Leftrightarrow \alpha = \pi$$

9.2. 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.

9. Verrundung mit einem Kreisbogen

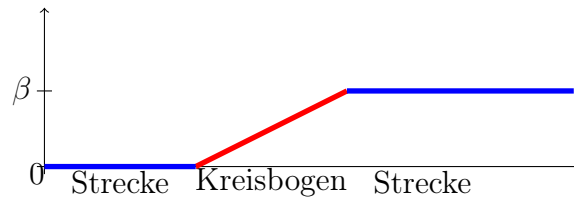


Figure 9.2.: Angular change with blending arc

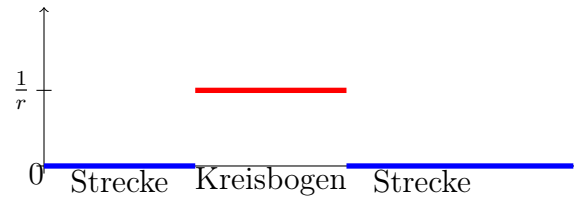


Figure 9.3.: Curvature progression for a blending arc

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.

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 blending 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.

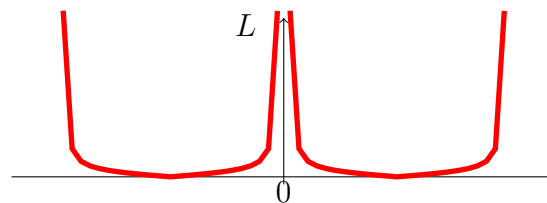


Figure 9.4.: Maximum range for a blending arc of a circle - $L(\varepsilon = \text{const}, \alpha)$

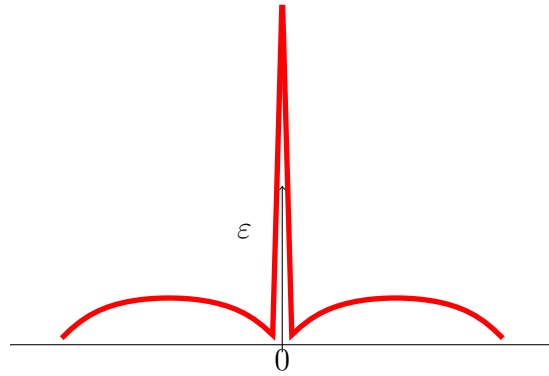


Figure 9.5.: Deviation when specifying the distance L when rounding with a circular arc

9.3. Examples

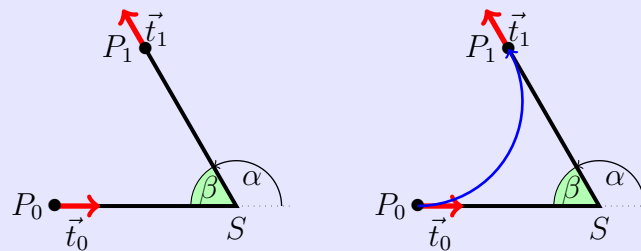
Beispiel. Let it be the symmetric Hermite problem

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

with $L = 6$ and $\alpha = \frac{2}{3}\pi$.

For the blending arc follows:

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



9. Verrundung mit einem Kreisbogen

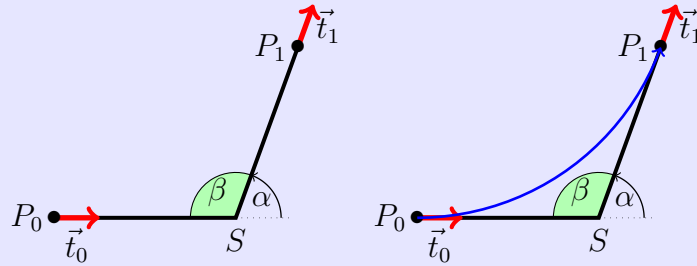
Beispiel. Let it be the symmetric Hermite problem

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

with $L = 6$ and $\alpha = \frac{7}{18}\pi$.

For the blending arc follows:

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



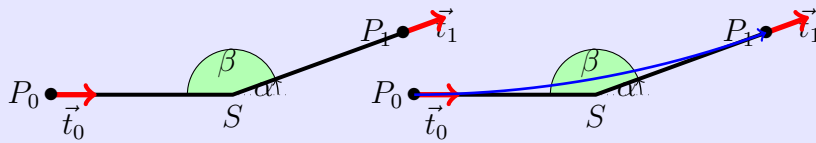
Beispiel. Let it be the symmetric Hermite problem

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

with $L = 6$ and $\alpha = \frac{1}{9}\pi$.

For the blending arc follows:

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

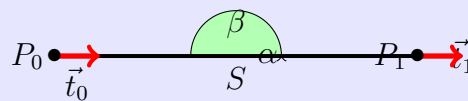


Beispiel. Let it be the symmetric Hermite problem

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

with $L = 6$ and $\alpha = 0$.

There is no blending arc here.



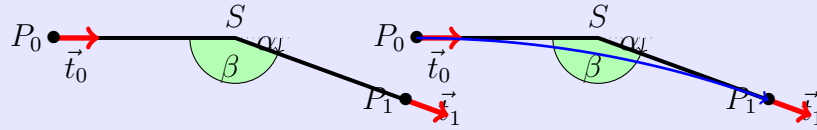
Beispiel. Let it be the symmetric Hermite problem

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

with $L = 6$ and $\alpha = -\frac{1}{9}\pi$.

For the blending arc follows:

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



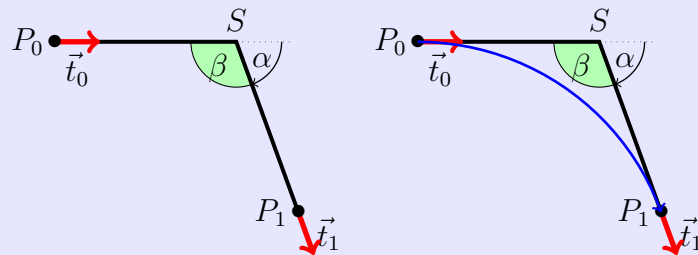
Beispiel. Let it be the symmetric Hermite problem

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

with $L = 6$ and $\alpha = -\frac{7}{18}\pi$.

For the blending arc follows:

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



9. Verrundung mit einem Kreisbogen

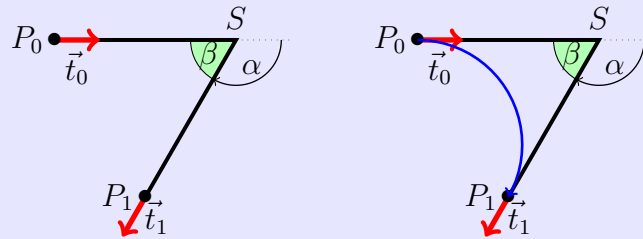
Beispiel. Let it be the symmetric Hermite problem

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

with $L = 6$ and $\alpha = -\frac{2}{3}\pi$.

For the blending arc follows:

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



10. Maple files

[Wat:2017a; Wat:2017b; Wat:2017c; Wat:2017d; Wat:2017e; Wat:2017f]

10.1. 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 [Wat:2017a] 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.

10.2. 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**)

10. Maple files

- 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.

10.3. 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:

```
TestP0 := ["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“.

10.4. 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;
```

10. Maple files

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 [Wat:2017a].

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;
```

10.4.1. 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)
        content;
    end proc;

    ...

end module;

```

¹Possible, but not used in the modules

²Example **MPoint**

10.4.2. 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);`

10.4.3. 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.

10.4.4. 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.

10.5. 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.

10.5.1. 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.

10. Maple files

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

10.5.2. 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

10.5.3. 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

10.5.4. module MBezier

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

10. Maple files

`[MVBEZIER, [PointList]]`

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

`MBezier`

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

10.5.5. 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	<code>New</code>	Data structure for a list of points
E	<code>GetPoint</code>	Reading the ith point from the point list
E	<code>GetN</code>	Determine the number of points in the point list
E	<code>Length</code>	Determination of the Euclidean length of the polygon
E	<code>Position</code>	Calculation of a point on the polygon course
E	<code>Tangents</code>	calculation of the tangent
L	<code>Plot2DA11</code>	Plot list of all points
E	<code>Plot2D</code>	Plot of all points as polygonal plots.
E	<code>Plot2DTangent</code>	representation of the polygon course with tangent
E	<code>PlotPoints</code>	representation of all points

10.5.6. 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

10.5.7. 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).

10. Maple files

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

10.5.8. 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

10.5.9. 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

10.5.10. 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

10.5.11. 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 Hermite 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.

10.5.12. 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

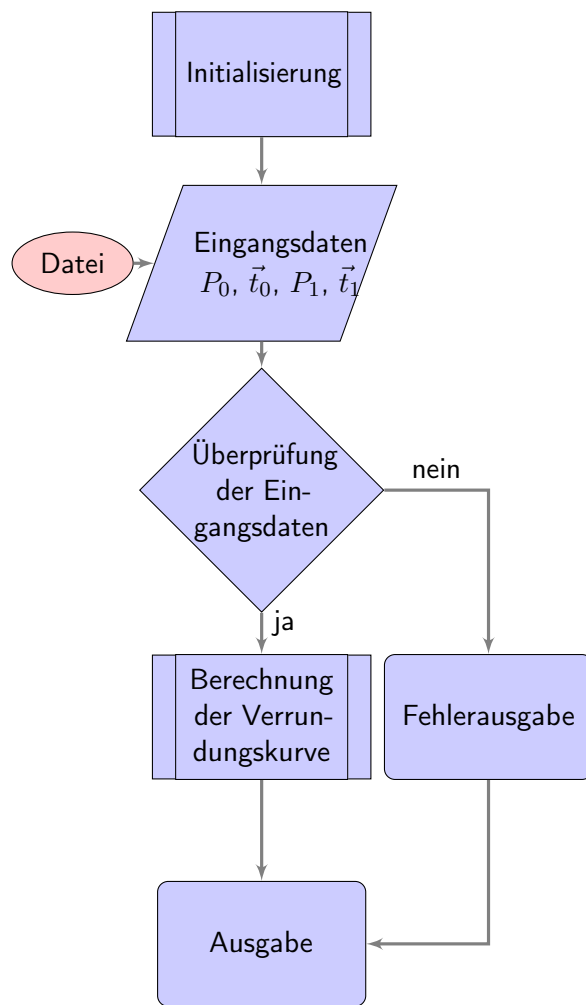


Figure 10.1.: Programme flow chart „Corner rounding“

10.6. Programme Flowchart

10.6.1. Overall Flow

10.6.2. Verrundung der Kurve

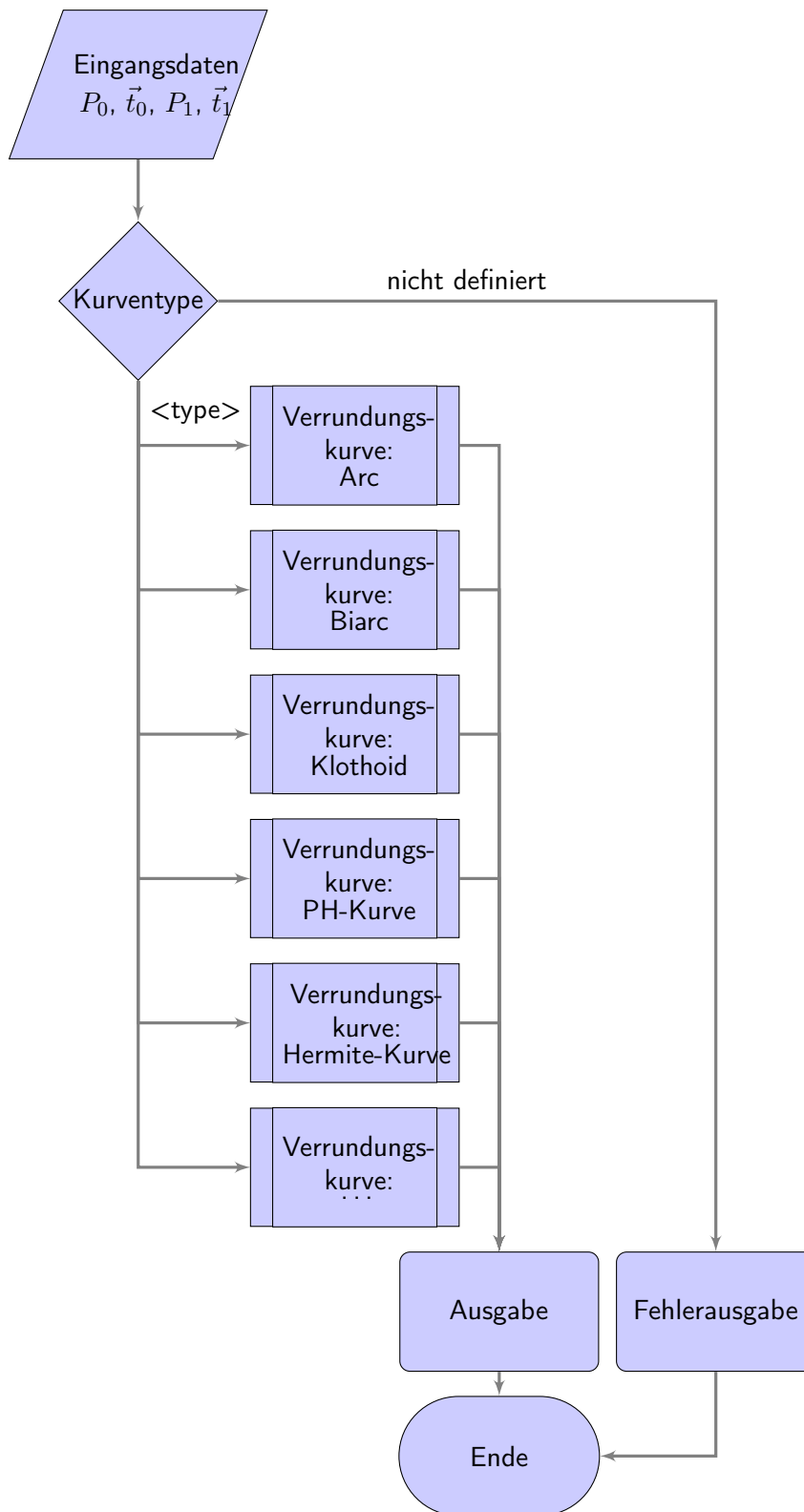


Figure 10.2.: Programme flowchart „Selection of the rounding strategies“

11. First Chapter

...

A Speicherprogrammierbare Steuerung (SPS) is ...

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

12. 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. [**Farin:2002**]

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

Mr. F. Farouki has dealt with both CNC machine programming and splines. His article¹ on a real-time interpolator also shows this. [**Farouki:2017**]

A new dimension of machine tools have emerged with the invention of 3D printers. [**Patent3D**]

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

¹co-author is J. Srinathu

²**Zafeiropoulos:2020.**

A. drawings with tikz

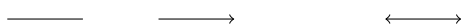
The package tikz is a powerful tool for creating graphics. Many introductions exist. Here only the first steps are shown, so that you can easily create flowcharts.

Drawing a line and arrows

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

  \draw[->] (2,0) — (3,0);

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



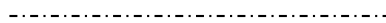
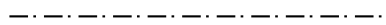
Drawing a thick blue line

```
\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);

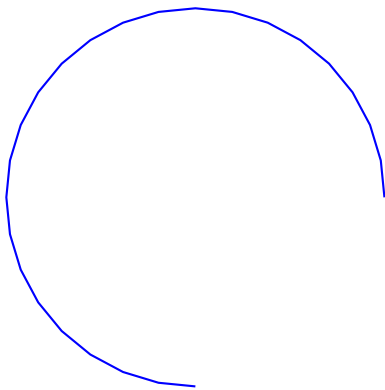
  \draw [thick, dash dot] (0,1) — (5,1);
  \draw [thick, dash pattern={on 7pt off 2pt on 1pt off 3pt}] (0,2) —
\end{tikzpicture}
```



Drawing an arc

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


A. drawings with tikz



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)});

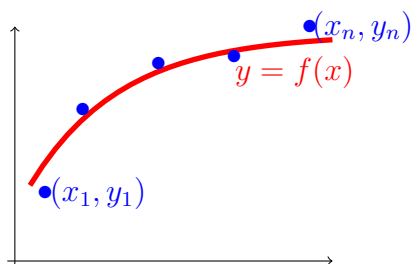
% Bezeichnung
\node[red] (O) at (3.5,2.5) {$y=f(x)$};

% 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) {$\bullet$};

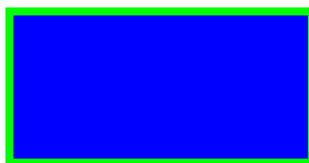
% Koordinatensystem
\draw[color=black,->] (-0.2,-0.1) — (-0.2,3.1);
\draw[color=black,->] (-0.3,0) — (4,0);
\end{tikzpicture}
```



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) —
  \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}

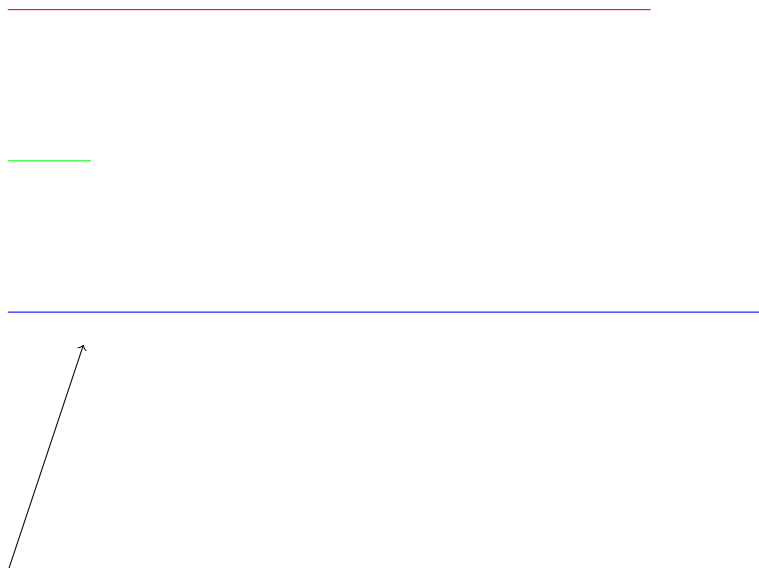
\bigskip

%\usetikzlibrary{math} %needed tikz library

\begin{tikzpicture}
\pgfmathsetmacro{\drehpunkt}{11.63815573}
%Variables must be declared in a tikzmath environment but
```

A. drawings with tikz

```
% 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}
```

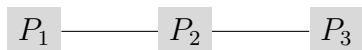


Use of points

```
%\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}
```



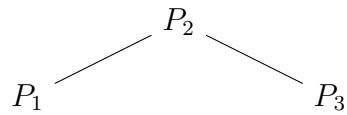
Use of nodes

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

```

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

```

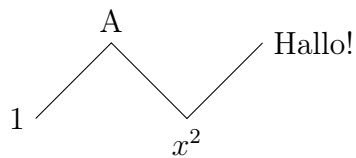


Use of nodes

```

\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 L^AT_EX 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 L^AT_EXfiles clearly laid out?
 - ☐ Are indentations used in the L^AT_EXfiles?
 - ☐ 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?

B. Criteria for a good \LaTeX project

- ☐ 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.