Grafischer Darstellungsrechner

Authors: Priller Patrick, Mairhofer David, Pernthaler Daniel

stpripat@bx.fallmerayer.it, stmaidav@bx.fallmerayer.it, stperdan@bx.fallmerayer.it

Dokumantation

Informatik
Klasse 4AT

Professor: Rainer Ulrich

Brixen, den 03. Mai 2023

Inhaltsverzeichnis

1 Themenbeschreibung

1.1 Projektziele

In diesem Projekt verfolgen wir das Ziel, eine benutzerfreundliche und leistungsfähige grafische Funktionen-Plotting-Anwendung zu entwickeln, die es Benutzern ermöglicht, mathematische Funktionen und ihre Ableitungen zu visualisieren und zu analysieren. Um dieses Ziel zu erreichen, haben wir uns folgende spezifische Ziele gesetzt:

- Entwicklung einer benutzerfreundlichen Oberfläche, die es Benutzern ermöglicht, Funktionen einfach einzugeben und zu bearbeiten.
- Unterstützung einer breiten Palette von mathematischen Funktionen, einschließlich linearer, quadratischer, exponentieller und trigonometrischer Funktionen.
- Integration von APIs wie mxParser, um die Funktionalität und Benutzerfreundlichkeit der Anwendung zu erweitern.
- Implementierung von Funktionen zur Berechnung und Anzeige von Ableitungen und Kurvendiskussionen.
- Bereitstellung von Werkzeugen zur Anpassung der Darstellung von Funktionen und des Plotbereichs.
- Ermöglichung von Zoomen und Verschieben des Plotbereichs, um die Anzeige von Funktionen flexibel zu gestalten.
- Entwicklung eines effektiven Testverfahrens, um sicherzustellen, dass die Anwendung korrekt funktioniert und benutzerfreundlich ist.
- Erstellung einer umfassenden Dokumentation, die den Entwicklungsprozess, die Systemarchitektur und die Benutzung der Anwendung beschreibt.

Durch das Erreichen dieser Ziele wollen wir eine leistungsstarke und benutzerfreundliche Plotter-Anwendung schaffen, die sowohl für den Bildungsbereich als auch für professionelle Anwender wertvoll ist und die Visualisierung und Analyse von mathematischen Funktionen vereinfacht.

1.2 Potenziell nutzbare APIs

In unserem Plotter-Projekt haben wir verschiedene APIs in Betracht gezogen, um die Funktionalität und Benutzerfreundlichkeit der Anwendung zu erweitern. Zwei der vielversprechendsten APIs, die wir untersucht haben, sind mxParser.

1.2.1 mxParser

mxParser ist eine vielseitige und leistungsfähige Bibliothek zur Analyse und Berechnung mathematischer Ausdrücke. Die Bibliothek ist in Java geschrieben und unterstützt eine breite Palette von Funktionen, einschließlich algebraischer, trigonometrischer, exponentieller und logarithmischer Funktionen. Wir haben mxParser in unserem Plotter-Projekt verwendet, um mathematische Ausdrücke zu analysieren und ihre Werte zu berechnen. Die offizielle Website von mxParser bietet umfassende Dokumentation und Beispiele zur Verwendung der Bibliothek: `http://mathparser.org/`

1.3 Technologien und Tools

Bei der Entwicklung des Plotters haben wir eine Reihe von Technologien und Tools verwendet, um eine solide und benutzerfreundliche Anwendung zu erstellen. In diesem Abschnitt werden die wichtigsten Technologien und Tools vorgestellt, die wir im Projekt eingesetzt haben:

1.4 Programmiersprache und Plattform

- **Java**: Wir haben uns für die Programmiersprache Java entschieden, da sie plattformübergreifend, objektorientiert und gut geeignet für die Entwicklung von grafischen Anwendungen ist. Java bietet eine breite Palette von Bibliotheken und Frameworks, die die Implementierung der benötigten Funktionen erleichtern.
- **JavaFX**: Für die Erstellung der grafischen Benutzeroberfläche haben wir JavaFX verwendet, ein modernes Framework für die Entwicklung von Desktop-Anwendungen in Java. JavaFX ermöglicht die Erstellung von ansprechenden und interaktiven Benutzeroberflächen und bietet viele nützliche UI-Komponenten, die für unser Projekt relevant sind.

1.5 APIs und Bibliotheken

- **mxParser**: Diese leistungsfähige Bibliothek wird verwendet, um mathematische Ausdrücke zu analysieren und ihre Werte zu berechnen. mxParser unterstützt eine Vielzahl von Funktionen, was es uns ermöglicht, eine breite Palette von mathematischen Funktionen in unserer Anwendung zu unterstützen.

1.6 Entwicklungsumgebung und Tools

- **IntelliJ IDEA**: Wir haben die IntelliJ IDEA-Entwicklungsumgebung verwendet, um unseren Code zu schreiben, zu testen und zu debuggen. IntelliJ IDEA bietet eine Vielzahl von Funktionen, die die Entwicklung von Java- und JavaFX-Anwendungen erleichtern, einschließlich Code-Vervollständigung, Refactoring-Tools und Integration mit Versionskontrollsystemen wie Git.
- **Maven**: Maven ist ein Build-Management-Tool, das wir verwendet haben, um den Build-Prozess zu automatisieren und die Abhängigkeiten unseres Projekts zu verwalten. Maven erleichtert die Integration von externen Bibliotheken wie mxParser und stellt sicher, dass unser Projekt auf verschiedenen Plattformen und Systemen konsistent gebaut und ausgeführt werden kann. In unserem Fall war Maven zusätzlich vorteilhaft, da es die Verwendung von JavaFX in IntelliJ IDEA stark vereinfacht hat.

- **Git**: Für die Versionskontrolle und das Teamwork haben wir Git und GitHub verwendet. Git ermöglicht es uns, Änderungen am Code nachzuvollziehen, verschiedene Versionen der Anwendung zu verwalten und die Zusammenarbeit zwischen Teammitgliedern zu erleichtern. Generell hatten wir bei GitHub kaum Probleme, bis auf kleine Konflikte jeweils beim Mergen. Wir haben den Fehler gemacht, oft Dateien wie beispielsweise "workspace" oder ähnliche Dateien von IntelliJ zu commiten, die auf allen Geräten natürlich unterschiedlich sind und so kleine Konflikte entstanden, die recht schnell behoben wurden. Die Mergekonflikte wurden manuell behoben. Was sehr hilfreich war, waren die Commitnachrichten, durch die wir am Ende genau wussten, was gemacht wurde (wichtig für die Dokumentation). Für die Zukunft wäre es besser, nicht zu oft zu commiten und nur fertige, funktionierende Codeteile zu commiten, da sonst alles unübersichtlich wird. Die "rollback"-Funktion mussten wir glücklicherweise nie verwenden. Ganz zu Beginn wurde einmal aus Versehen ein ganzer Branch gelöscht, was aber schnell kompensiert werden konnte.

Durch die Kombination dieser Technologien und Tools konnten wir eine leistungsfähige und benutzerfreundliche Anwendung entwickeln, die es Benutzern ermöglicht, mathematische Funktionen effizient zu visualisieren und zu analysieren.

2 Vorgehensmodell: Kanban

2.1 Grundlagen von Kanban

Kanban ist ein visuelles Projektmanagement-System, das ursprünglich in der Automobilindustrie entwickelt wurde und auf der Kanban-Methode basiert. Für unser Plotter-Projekt wurde Kandan gewählt, da es eine hohe Flexibilität bietet und es uns ermöglicht, einen guten Überblick über den Fortschritt des Projekts zu behalten.

Kanban besteht aus einer Reihe von Spalten, die verschiedene Phasen des Projekts repräsentieren, z. B. “Zu erledigen“, “In Arbeit“ und “Fertig“. Innerhalb dieser Spalten werden Aufgaben als Karten dargestellt, die Informationen zu Priorität, Verantwortlichen und Fälligkeitsdatum enthalten. Karten werden von einer Spalte zur nächsten verschoben, wenn sie von einer Phase zur nächsten fortschreiten.

Dieses System erleichtert die Verfolgung des Fortschritts und hilft dabei, Engpässe oder Verzögerungen schnell zu identifizieren. Es fördert außerdem eine kontinuierliche Verbesserung, indem es Teams dazu anregt, regelmäßig den Workflow zu überprüfen und anzupassen.

Die folgende Abbildung zeigt ein Beispiel für ein Kandan-Board:

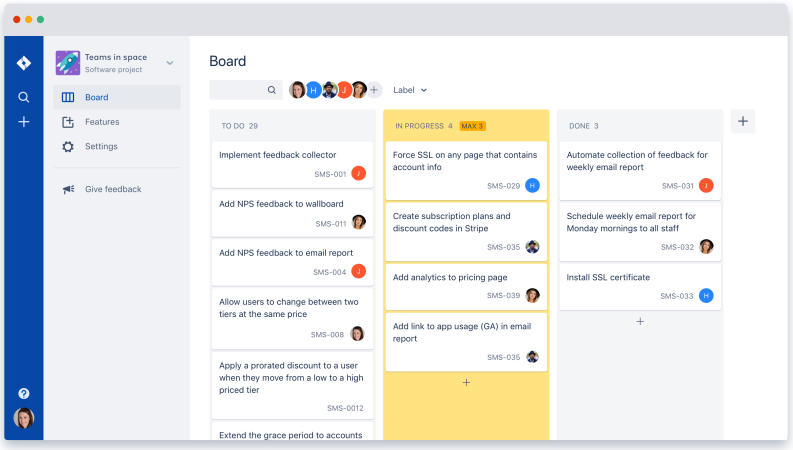

Abbildung 1: Beispiel eines Kandan-Boards

Insgesamt bietet Kandan eine einfache und effektive Möglichkeit, den Fortschritt unseres Plotter-Projekts zu verwalten und sicherzustellen, dass wir kontinuierlich Verbesserungen vornehmen und auf unsere Ziele hinarbeiten.

2.2 Anwendung von Kanban

Im Verlauf unseres Projekts haben wir uns für das Vorgehensmodell Kanban entschieden, um den Entwicklungsprozess zu organisieren und zu steuern. Kanban ist eine agile Methode, die auf kontinuierlicher Verbesserung und Flexibilität basiert. Es hilft, den Arbeitsfluss zu visualisieren, Engpässe zu identifizieren und die Produktivität des Teams zu steigern.

Wir haben ein Kanban-Board verwendet, um unsere Aufgaben und den Fortschritt im Projekt darzustellen. Unser Kanban-Board war in mehrere Spalten unterteilt, die den verschiedenen Phasen des Arbeitsablaufs entsprechen, z.B. 'To Do', 'In Progress', 'Review' und 'Done'. Jede Aufgabe wurde als Karte auf dem Board dargestellt, und wir haben die Karten zwischen den Spalten verschoben, um den Fortschritt der Aufgaben zu visualisieren.

Um den Entwicklungsprozess effektiv zu gestalten, haben wir unser Programm in kleinere Teile aufgeteilt, die wir Vorgänge (in Jira, dem von uns verwendeten Projektmanagement-Tool, genannt) nennen. Dies hat dazu beigetragen, dass wir uns auf bestimmte Aspekte der Anwendung konzentrieren und diese unabhängig voneinander entwickeln konnten.

In regelmäßigen Stand-up-Meetings haben wir den Status unserer Aufgaben überprüft, Engpässe identifiziert und die Prioritäten im Projekt angepasst. Dadurch konnten wir sicherstellen, dass die wichtigsten Aufgaben stets im Fokus standen und effektiv bearbeitet wurden.

2.3 Vorteile und Erfahrungen

Durch die Anwendung von Kanban in unserem Projekt haben wir verschiedene Vorteile erfahren:

- **Flexibilität**: Kanban hat es uns ermöglicht, schnell auf Änderungen im Projektumfeld oder in den Anforderungen zu reagieren. Dadurch konnten wir unsere Prioritäten anpassen und sicherstellen, dass wir uns auf die wichtigsten Aspekte des Projekts konzentrierten.
- **Transparenz**: Durch die Visualisierung des Arbeitsablaufs auf dem Kanban-Board konnten alle Teammitglieder den aktuellen Status der Aufgaben und den Fortschritt des Projekts einsehen. Dies hat zu einer besseren Kommunikation und Zusammenarbeit im Team geführt.
- **Effizienz**: Die Anwendung von Kanban und die Aufteilung des Programms in kleinere Vorgänge haben dazu beigetragen, Engpässe im Arbeitsablauf zu identifizieren und zu beseitigen. Dies hat die Produktivität des Teams gesteigert und dazu geführt, dass Aufgaben schneller und effizienter abgeschlossen wurden.
- **Kontinuierliche Verbesserung**: Die regelmäßigen Stand-up-Meetings und die Möglichkeit, den Arbeitsablauf anzupassen, haben dazu beigetragen, kontinuierliche Verbesserungen im Projektprozess zu fördern. Dadurch konnten wir mögliche Probleme frühzeitig erkennen und Lösungen entwickeln, um diese zu beheben.

Insgesamt hat die Anwendung von Kanban in unserem Projekt dazu beigetragen, einen effizienten und flexiblen Entwicklungsprozess zu gestalten, der es uns ermöglichte, eine erfolgreiche und benutzerfreundliche Anwendung zu entwickeln.

3 Anforderungsanalyse

3.1 Funktionale Anforderungen

Funktionale Anforderungen beschreiben die Funktionalitäten, die das System bieten soll, um die gewünschten Ziele zu erreichen. Für das Plotter-Projekt umfassen die funktionalen Anforderungen:

- Eingabe einer mathematischen Funktion durch den Benutzer.
- Grafische Darstellung der eingegebenen Funktion auf einer zweidimensionalen Zeichenfläche.
- Verschieben und Zoomen der Zeichenfläche zur besseren Analyse der Funktion.
- Hinzufügen und Entfernen von Funktionen.
- Möglichkeit zur Bearbeitung der Funktionen, einschließlich Ableitung und Kurvendiskussion.
- Verwaltung von Variablen und Konstanten in der Funktion.
- Anpassung der Darstellung der Funktion, wie z.B. Farbe und Sichtbarkeit.

3.2 Nicht funktionale Anforderungen

Nicht funktionale Anforderungen beziehen sich auf die Qualitätsaspekte eines Systems und legen fest, wie gut die funktionalen Anforderungen erfüllt werden. Für das Plotter-Projekt umfassen die nicht funktionalen Anforderungen:

- Benutzerfreundlichkeit: Die Anwendung sollte einfach und intuitiv zu bedienen sein.
- Leistung: Die Anwendung sollte effizient arbeiten und in angemessener Zeit eine grafische Darstellung der Funktionen erstellen.
- Skalierbarkeit: Die Anwendung sollte in der Lage sein, mit einer wachsenden Anzahl von Funktionen und Variablen umzugehen.
- Portabilität: Die Anwendung sollte auf verschiedenen Betriebssystemen lauffähig sein.
- Zuverlässigkeit: Die Anwendung sollte stabil und fehlerfrei arbeiten.

3.3 Aktivitätsdiagramm

Ein Aktivitätsdiagramm ist eine grafische Darstellung der Arbeitsabläufe und Abläufe innerhalb eines Systems. Es zeigt, wie verschiedene Aktivitäten und Entscheidungen innerhalb des Systems aufeinander folgen und miteinander verknüpft sind.

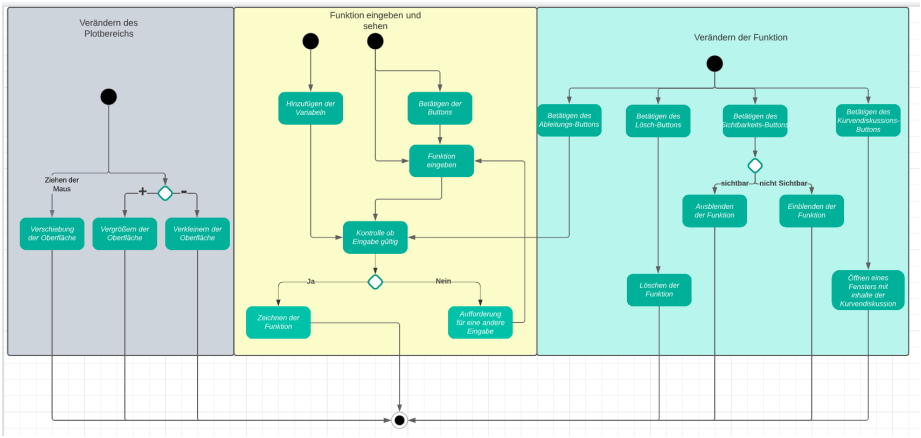

Abbildung 2: Aktivitätsdiagramm des Plotter-Projekts

Das Aktivitätsdiagramm beschreibt den Ablauf eines Plotters. Ein Prozess beginnt damit, dass der Nutzer in ein Textfeld schreiben kann, dies kann er entweder mithilfe von Buttons für Sonderzeichen oder auch direkt. Nachher wird die Eingabe auf ihre Gültigkeit überprüft. Parallel kann der User auch Variablen hinzufügen, welche auf ihre Gültigkeit anschließend überprüft werden, sollte die Variable falsch eingegeben werden, nimmt es die letzte gültige Eingabe und setzt die Variable auf deren Wert, und gibt gleichzeitig eine Aufforderung aus, dass die Variable richtig eingegeben werden soll. Sollte die Funktion einen falschen Syntax haben, wird wiederum eine Aufforderung angezeigt, welche fordert die Funktion richtig einzugeben. Sollten die Eingaben gültig sein, wird die Funktion gezeichnet.

Wenn der Nutzer die Ansicht des Plottbereichs verändern will, kann er mithilfe des Mausrads zoomen. Zudem kann der User mithilfe von Ziehen der Maus die Oberfläche in x und y Richtung verschieben. Wenn der Benutzer mit einer vorhandenen Funktion interagieren möchte, kann er die Schaltflächen verwenden. Diese Schaltflächen ermöglichen es ihm, die Funktion auszublenden, wenn sie sichtbar ist, oder sie wieder anzuzeigen, wenn sie ausgeblendet wurde. Außerdem kann er die Funktion löschen, ihre Ableitung berechnen oder sich die vollständige Kurvendiskussion anzeigen lassen.

3.4 Zeiteinteilung

Feature	**Zeit est.**	**Priorität**	**Zeit act.**
Eingegebene mathematische Funktion auf dem Plottbereich anzeigen (Grafisches + Berechnung der Funktion)	10	10	12
Mathematische Variablen	6	8	3
Zoom im Plottbereich	3	7	3
Plottbereich bewegen	2	7	1+
Beschriftung der Achsen	1	8	2
Ableitungsbutton der Funktion	5	4	2
Sichtbarkeitsbutton der Funktion	2	5	1
Löschbutton der Funktion	1	6	1
Kurvendiskussionsbutton der Funktion	8	3	N/A
Styling der Oberfläche	4	4	5
Javadocs	5	8	5
JUnit	6	$8 > 1$	6
Dokumentation des Programms	8	10	20

Tabelle 1: Zeiteinteilung

Die Tabelle 1 zeigt eine Übersicht über die Anforderungen an das Projekt. Für jede Anforderung wird die geschätzte Zeit für die Implementierung, die Priorität der Anforderung und die tatsächlich aufgewendete Zeit aufgeführt. Es wird deutlich, dass einige Anforderungen, wie die Darstellung der eingegebenen mathematischen Funktion auf dem Plotbereich und die Dokumentation des Programms, eine hohe Priorität hatten und entsprechend mehr Zeit in Anspruch nahmen. Andere Anforderungen, wie der Ableitungsbutton und der Kurvendiskussionsbutton, hatten eine niedrigere Priorität und benötigten weniger Zeit. Der Kurvendiskussionsbutton wurde aus diesem Grund beispielsweise nicht mehr umgesetzt, da das Verhältnis von Zeitaufwand und Priorität zu hoch war. Insgesamt zeigt die Tabelle, dass die effektive Zeit oft von der geschätzten Zeit abweicht, was auf die Komplexität und die unvorhergesehenen Herausforderungen bei der Implementierung der verschiedenen Anforderungen hinweist.

4 Use-Case-Diagramm

4.1 Erklärung

Das Use-Case-Diagramm beschreibt die Funktionalität des Plotters und zeigt, wie Benutzer mit der Anwendung interagieren können. Der Benutzer kann in ein Textfeld seine gewünschte Funktion eingeben und mithilfe von Buttons Sonderzeichen in das Textfeld schreiben. Der Benutzer hat auch die Möglichkeit, verschiedene Buttons zu verwenden, um die Funktion zu bearbeiten, wie zum Beispiel die Sichtbarkeit, Ableitung, Kurvendiskussion oder die Funktion zu löschen. Zusätzlich kann der Benutzer den Anzeigebereich der Funktion mithilfe von Zoomen oder Verschieben verändern.

4.2 Diagramm

Das folgende Diagramm zeigt das Use-Case-Diagramm für unseren Plotter:

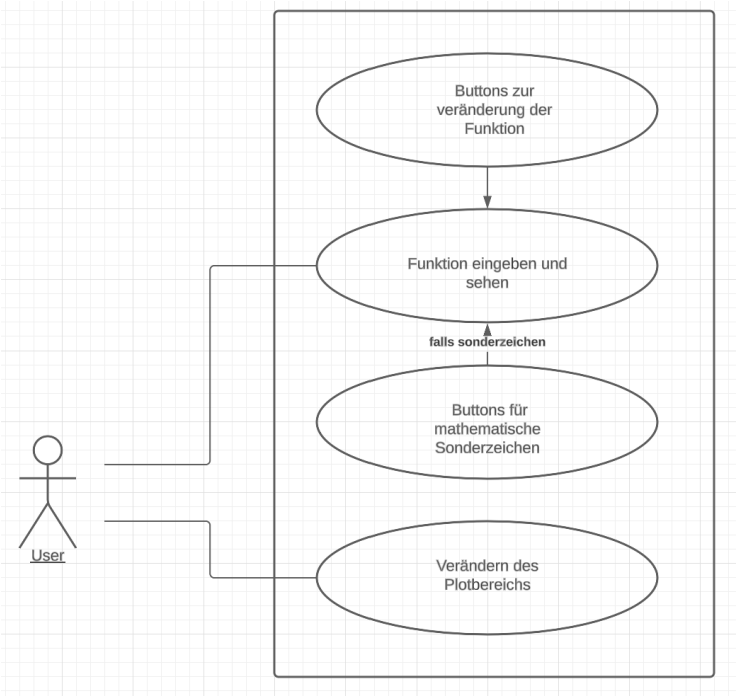

Abbildung 3: Use-Case-Diagramm des Plotters

Das Diagramm veranschaulicht die verschiedenen Aktionen, die ein Benutzer ausführen kann, um mit der Anwendung zu interagieren und mathematische Funktionen zu visualisieren und zu analysieren.

5 Systemarchitektur

5.1 Übersicht

Die Systemarchitektur des Projekts ist in zwei Hauptkomponenten unterteilt: das Frontend und das Backend. Das Frontend ist für die Benutzeroberfläche und die Interaktionen mit dem Benutzer verantwortlich, während das Backend die Berechnungen und die Verarbeitung der eingegebenen mathematischen Funktionen übernimmt. Zusammen bieten sie eine effektive und benutzerfreundliche Anwendung zum Plotten von mathematischen Funktionen und ihren Ableitungen.

5.2 Frontend

Das Frontend der Anwendung ist mit JavaFX implementiert, einer leistungsstarken und flexiblen Bibliothek für die Erstellung grafischer Benutzeroberflächen. Es bietet eine Vielzahl von UI-Komponenten und Steuerelementen, die für die Implementierung der Anwendung erforderlich sind, einschließlich Szenen, Gruppen, Schaltflächen, Eingabefelder und mehr. Das Frontend ist für die Organisation und Darstellung der Benutzeroberfläche verantwortlich, einschließlich der Eingabefelder für Funktionen, Schaltflächen für mathematische Operationen und Anzeige der Funktionsplots im Plotbereich.

Abbildung 4: Screenshot der Benutzeroberfläche der Anwendung

5.3 Backend

Das Backend der Anwendung ist für die Verarbeitung der eingegebenen mathematischen Funktionen, die Berechnung der Ableitungen und die Aktualisierung des Plotbereichs verantwortlich. Es verwendet die mxparser-Bibliothek, um mathematische Ausdrücke zu parsen und auszuwerten. Die Hauptklassen der Anwendung sind verantwortlich für die grafischen Darstellungen der Funktionen und ihrer Ableitungen. Die Klassen DerivativeCalculator, PlotFunction und PlotArea waren ursprünglich die Hauptkomponenten des Backends und arbeiteten zusammen, um die gewünschte Funktionalität bereitzustellen. Es ist jedoch zu beachten, dass die DerivativeCalculator-Klasse später nicht mehr benötigt wurde, da mxparser eine ähnliche Funktionalität bietet. Die Ableitungsrechenfunktion von mxparser ist jedoch nicht so effizient. Daher besteht die Möglichkeit, dass wir in Zukunft die DerivativeCalculator-Klasse erneut implementieren könnten, um eine effizientere Berechnung der Ableitungen zu ermöglichen.

5.4 Klassendiagramm

Das Klassendiagramm zeigt die Struktur und die Beziehungen zwischen den Hauptklassen der Anwendung. Es hilft dabei, die Verantwortlichkeiten jeder Klasse und die Interaktionen zwischen den Klassen besser zu verstehen.

- **DerivativeCalculator**: Diese Klasse ist verantwortlich für die Berechnung der Ableitungen der eingegebenen mathematischen Funktionen. Sie verwendet die mxparser-Bibliothek und interagiert mit der PlotFunction-Klasse, um die Ableitungen zu berechnen und bereitzustellen.
- **Layout**: Diese Klasse ist ein benutzerdefiniertes JavaFX Group-Objekt, das die Benutzeroberfläche der Anwendung organisiert. Es enthält die Komponenten wie Eingabefelder, Schaltflächen und den Plotbereich und stellt sicher, dass diese Elemente auf dem Bildschirm richtig angeordnet sind.
- **PlotArea**: Diese Klasse ist ein benutzerdefiniertes JavaFX Group-Objekt, das den Plotbereich der Anwendung darstellt. Es verwaltet das Hinzufügen, Entfernen und Aktualisieren von Funktionen, Variablen und Gitterlinien im Plotbereich.
- **PlotFunction**: Die PlotFunction-Klasse verwaltet die grafische Darstellung einer mathematischen Funktion. Sie berechnet die Positionen der Liniensegmente, die die Kurve der Funktion approximieren, und steuert die Sichtbarkeit der Funktion im Plotbereich.
- **Plotter**: Die Plotter-Klasse ist die Hauptklasse der JavaFX-Anwendung und steuert den gesamten Programmablauf. Sie verwendet die mxparser-Bibliothek, um die eingegebenen mathematischen Funktionen zu analysieren und auszuwerten.
- **ScrollPaneFunctionsElement**: Diese Klasse stellt ein einzelnes Funktionselement in der ScrollPane dar, einschließlich eines Textfelds zur Eingabe der Funktion und Schaltflächen zum Berechnen der Ableitung, Löschen der Funktion, Steuern der Sichtbarkeit und Anzeigen der Funktionsanalyse.

- **ScrollPaneVariablesElement**: Diese Klasse stellt ein einzelnes Variablenelement in der ScrollPane dar, einschließlich eines Textfelds zur Eingabe des Variablenwerts und einer Schaltfläche zum Löschen der Variable.

Die Klassen interagieren miteinander, um die Hauptfunktionalitäten der Anwendung bereitzustellen. Die Plotter-Klasse agiert als zentraler Koordinator, der die anderen Klassen steuert und verwaltet, um das Plotten von Funktionen, Berechnung von Ableitungen und Aktualisierung der Benutzeroberfläche zu ermöglichen.
Im Folgenden finden Sie das Klassendiagramm für das Projekt:

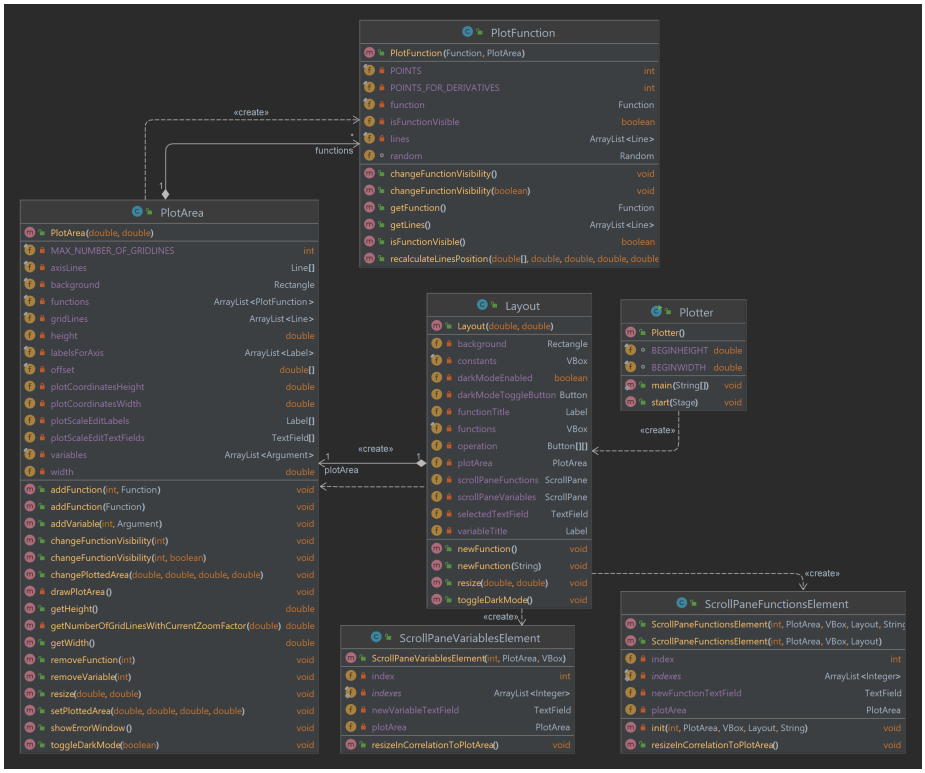

Abbildung 5: Klassendiagramm der Hauptklassen

6 Umsetzung (Programmierung)

6.1 UI-Prototyp

Der UI-Prototyp wurde erstellt, um eine Vorstellung davon zu bekommen, wie die endgültige Benutzeroberfläche aussehen sollte. Dabei orientierte sich das Design optisch am Taschenrechner von Windows. Der Prototyp wurde mithilfe des Figma-Tools erstellt, das eine unkomplizierte und einfache Gestaltung ermöglichte. Die Abbildung 6 zeigt den UI-Prototypen. Das Design diente als Basis für die Frontend-Entwicklung und half bei der Entscheidungsfindung für verschiedene Designelemente.

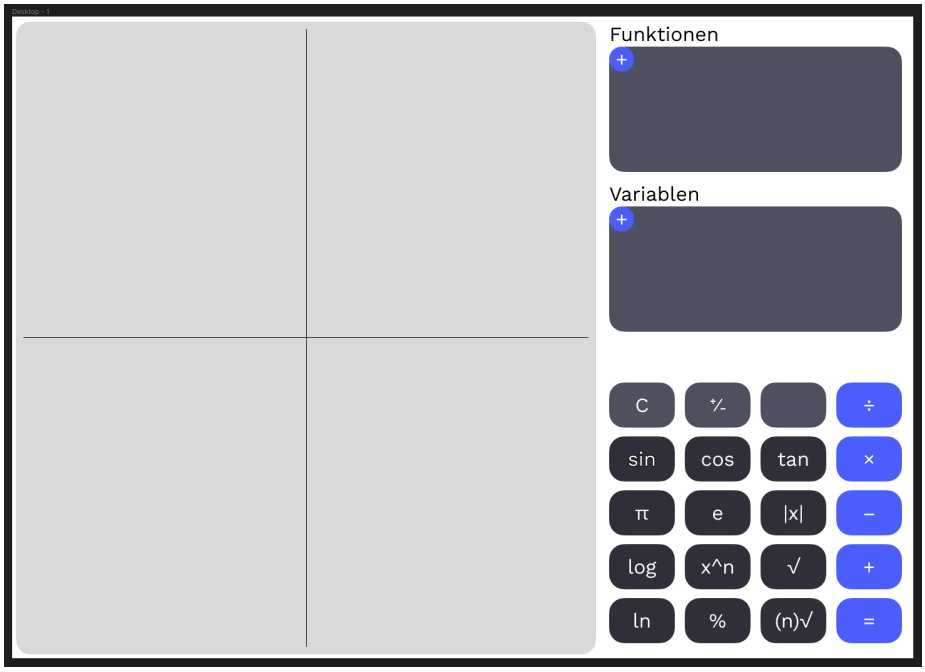

Abbildung 6: UI-Prototyp des Plotters

6.2 Backend-Entwicklung

Die Backend-Entwicklung fokussierte sich auf die Berechnung von Funktionswerten und Ableitungen sowie die Verarbeitung von Variablen und Konstanten. Eine der größten Herausforderungen bestand darin, die Neuberechnung der einzelnen Punkte des Graphen bei Verschiebung zu optimieren. Dies war besonders wichtig, um die Rechenaufwändigkeit und die Ausführungszeit der Anwendung zu reduzieren. Die Programmierung des Gitternetzes erwies sich ebenfalls als anspruchsvoll. Besonders das Verschieben des Gitters und das korrekte Darstellen der Funktion bei veränderter Skalierung und Positionierung erforderten eine präzise Koordination der Berechnungen.

6.3 Frontend-Entwicklung

Die Frontend-Entwicklung konzentrierte sich auf die Gestaltung der Benutzeroberfläche und die Implementierung der verschiedenen UI-Komponenten. Insbesondere die Erstellung und Formatierung der Knöpfe für die Eingabe von Funktionen, Variablen und mathematischen Operationen stellte eine Herausforderung dar. Die Verwendung von HTML, CSS und JavaFX für das Styling und Layout der Benutzeroberfläche erleichterte den Prozess. Dennoch war es schwierig, alle Elemente so zu gestalten, dass sie korrekt miteinander interagierten, insbesondere beim Verschieben des Netzwerks oder beim Anpassen der Größe des gesamten Fensters. Das Erstellen von Rändern und optisch ansprechenden Knöpfen erforderte ebenfalls besondere Aufmerksamkeit.

6.4 Teststrategie

Um die Qualität und Korrektheit der implementierten Funktionen zu gewährleisten, wurde eine Teststrategie entwickelt und JUnit-Tests für verschiedene Aspekte der Anwendung erstellt. Es wurden verschiedene Arten von Funktionen getestet, wie beispielsweise lineare, quadratische und höhergradige Funktionen. Die Tests verglichen die berechneten Werte der Anwendung mit den tatsächlichen Werten und stellten sicher, dass die Abweichungen innerhalb eines akzeptablen Bereichs lagen (in der Regel auf drei Nachkommastellen genau).

Gelegentlich kam es zu geringfügigen Abweichungen in den Ergebnissen, die jedoch auf die Unvermeidlichkeit von Rundungsfehlern und die Begrenzungen von Java bei der Verarbeitung von Gleitkommazahlen zurückzuführen sind. Beispielsweise kann es vorkommen, dass statt dem Wert 3,0 der Wert 3,000000000003 errechnet wird. Diese Abweichungen sind jedoch für die Zwecke der Anwendung vernachlässigbar und beeinträchtigen nicht die Funktionalität des Plotters.

Um sicherzustellen, dass die grafische Darstellung der Funktionen korrekt war, haben wir festgestellt, dass das Testen mit JUnit allein ungeschickt war. Daher haben wir eine alternative Methode angewendet. Wir haben das Programm gestartet und die grafische Oberfläche manuell überprüft, um sicherzustellen, dass die dargestellten Funktionen den erwarteten Ergebnissen entsprachen. Dieser Ansatz ermöglichte es uns, visuell zu überprüfen, ob die Funktionen korrekt gezeichnet wurden und ob die grafische Darstellung den Anforderungen entsprach. Durch diese kombinierte Herangehensweise konnten wir sowohl die Genauigkeit der berechneten Werte als auch die Richtigkeit der grafischen Darstellung der Funktionen gewährleisten.

7 Benutzung (User Guide)

7.1 Installation und Einrichtung

Um die Plotter-Anwendung zu installieren und einzurichten, befolgen Sie die folgenden Schritte:

1. Stellen Sie sicher, dass JDK 11 oder höher und JRE 18 installiert sind, da das Programm sonst nicht ausgeführt werden kann.
2. Laden Sie die ausführbare Datei aus dem Release-Teil des GitHub-Repositories herunter: `https://github.com/TheD4rkCoder/Plotter/releases`
3. Führen Sie die heruntergeladene Datei aus, um die Anwendung zu starten.
4. Beachten Sie, dass JavaFX 18 oder höher ebenfalls auf Ihrem Computer installiert sein muss, um die grafische Benutzeroberfläche korrekt anzeigen zu können.
5. Bei Problemen können genauere Informationen und Anweisungen in der Datei README.md im GitHub-Repository gefunden werden.

7.2 Bedienung der Anwendung

1. Geben Sie eine Funktion in das Textfeld ein und drücken Sie Enter. Die Funktion wird dem Graphen hinzugefügt.

Abbildung 7: Eingabefeld Funktion

2. Klicken Sie auf die Schaltflächen neben der Funktion, um verschiedene Aktionen auszuführen:
    - f': Berechnen und Anzeigen der Ableitung der Funktion
    - **X**: Entfernen der Funktion aus dem Graphen
    - **O**: Umschalten der Sichtbarkeit der Funktion im Graphen

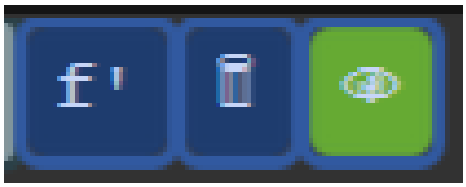

Abbildung 8: Funktionsschaltflächen

7.3 Beispiele

Folgende Beispiele zeigen verschiedene Funktionen, die mit dem Plotter dargestellt werden können:

1. Lineare Funktion: $y = 2x + 1$

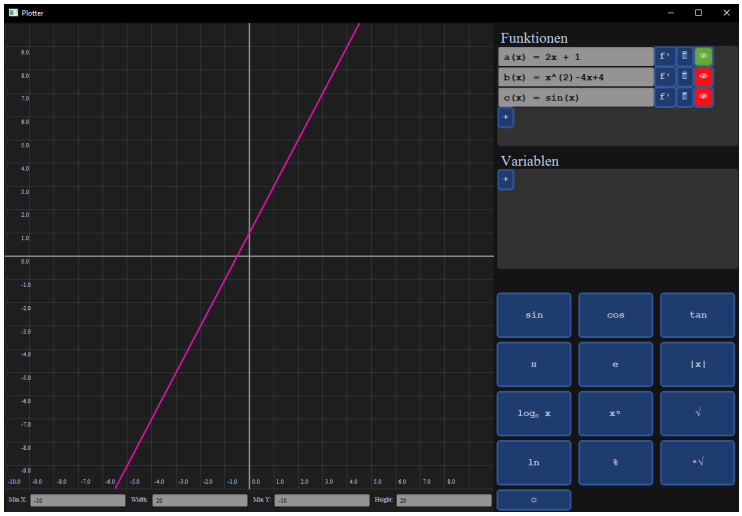

Abbildung 9: Lineare Funktion

2. Quadratische Funktion: $y = x^2 - 4x + 4$

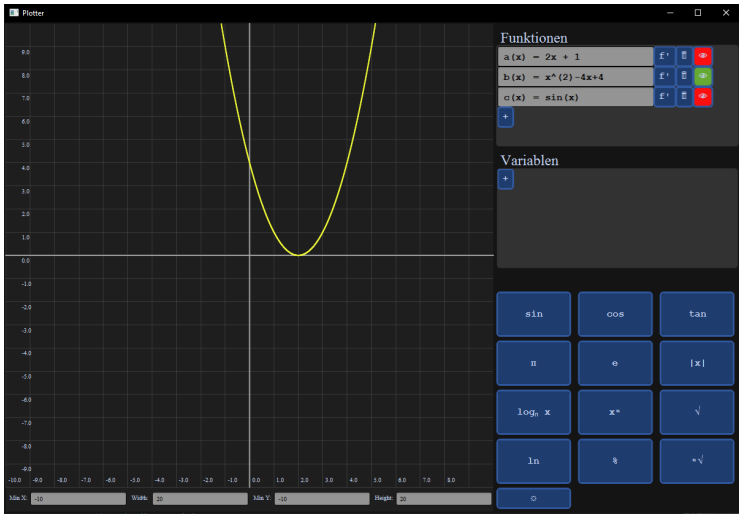

Abbildung 10: Quadratische Funktion

3. Trigonometrische Funktion: $y = \sin(x)$

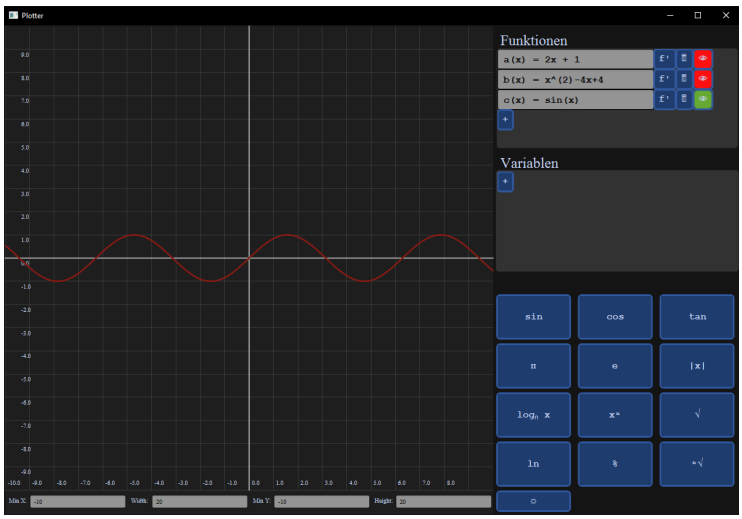

Abbildung 11: Trigonometrische Funktion

8 Fazit

8.1 Zusammenfassung

In diesem Projekt haben wir erfolgreich eine Java-basierte Plotter-Anwendung entwickelt, die es Benutzern ermöglicht, mathematische Funktionen und ihre Ableitungen grafisch darzustellen. Wir haben das flexible und leicht verständliche Kanban-System zur Verwaltung unserer Arbeitsabläufe verwendet. Durch den Einsatz von APIs wie mxParser und JavaFX haben wir eine effiziente und benutzerfreundliche Anwendung erstellt, die eine breite Palette von Funktionen unterstützt. Es war uns jedoch nicht möglich, alle Anforderungen zu erfüllen, da uns die Zeit für die Implementierung einer umfassenden Funktionsanalyse fehlte. Eine solche Analyse hätte die Berechnung von Nullstellen, Extremwerten und Wendepunkten umfassen können. Aufgrund der Komplexität dieser Funktionen und des begrenzten Zeitrahmens konnten wir diese Erweiterungen nicht vollständig umsetzen. Dennoch haben wir eine solide Grundlage geschaffen, auf der zukünftige Entwicklungen aufbauen können, um die Plotter-Anwendung weiter zu verbessern und zu erweitern.

8.2 Erreichte Ziele

Die wichtigsten erreichten Ziele sind:

- Erstellung einer benutzerfreundlichen grafischen Benutzeroberfläche.
- Implementierung von Funktionen zum Hinzufügen, Entfernen und Anzeigen von Funktionen und deren Ableitungen.
- Bereitstellung von Funktionen zur Anpassung der Darstellung des Plottbereichs und zum Zoomen.
- Erfolgreiche Integration externer APIs und Bibliotheken, wie mxParser und JavaFX, zur Verbesserung der Anwendung.

8.3 Zukünftige Erweiterungen

Mögliche zukünftige Erweiterungen der Plotter-Anwendung könnten sein:

- Hinzufügen weiterer Funktionstypen, wie parametrische und Polarkurven.
- Implementierung von Funktionen zur Analyse von Funktionen, wie Nullstellen, Extremwerte und Wendepunkte.
- Verbesserung der Leistung und Effizienz der Anwendung, insbesondere bei der Darstellung komplexer Funktionen und bei der Interaktion mit dem Plottbereich.
- Erweiterung der Anwendung um 3D-Plotting-Funktionen, um dreidimensionale Funktionen und Oberflächen darzustellen.

9 Anhang

9.1 Arbeitstagebuch (Planung)

Person	Datum	Zeit	Tätigkeiten
Daniel	16.01.	1h	Auf github registriert, Beginn Use-Case
David	16.01.	1h	Erstellung des Git-Repositorys, Besprechung des Projekts
Patrick	16.01.	1h	Erstellung Jira Projekt, Beginn Erstellung UseCaseDiagram(Lucid)
Daniel	30.01	1h	Fertigstellung Use-Case Diagramm
David	30.01	1h	helfen beim Use-Case Diagramm
David	06.02.	1h	Schauen, wie ein Aktivitätsdiagramm und Ablaufdiagramm aussieht
David	12.02.	1h	Anforderungsanalyse

Tabelle 2: Planung

9.2 Arbeitstagebuch (Umsetzung)

Person	Datum	Zeit	Tätigkeiten
Alle	17.04.	3h	Tests schreiben
David	19.04.	4h	Zeichnen der PlotFunction in PlotArea
Daniel	26.04.	1h	ButtonArea und Layout
David	26.04	3h	PlotArea changePlottedArea und drawPlotArea Methoden
Patrick	30.04.	2h	Erstellung verschiedener Lizenzdateien
Patrick	30.04.	1h	Aktualisierung JavaDOCS
David	30.04.	2h	CSS styling, kleine Fixes
David	30.04	4h	Menü Funktionen, Konstanten und Knöpfe updated
David	02.05.	3h	Ableitungen umgesetzt
Patrick	02.05.	1h	Grid Labels
Patrick	02.05.	1h	Erstellung Doku, Aktualisierung JavaDOCS
David Patrick	02.05.	1h	Darkmode hinzugefügt
David	05.05.	2h	Resizing umgesetzt
Patrick	05.05.	2h	Umstrukturierung der Klassen, lambdas hinzugefügt
Patrick	06.05.	1h	Styling
David	06.05.	2h	Styling
Patrick	07.05.	6h	Weiterarbeitung Doku, Mergekonflikte beheben
David	15.05.	2h	Label, scale und test Fixes
Patrick	17.05.	1h	SECURITY.md, CONTRIBUTING.md, PULL_REQUEST_TEMPLATE.md ect
David	18.05.	2h	Bugfixes
Patrick	19.05.	3h	JavaDOCS aktualisiert
David	20.05.	3h	Kommentieren/bugfixing
Patrick	20.05.	1h	Updated README.md
David	20.05.	1h	Kommentieren
Patrick	21.05.	5h	Weiterarbeitung Doku
Patrick David	21.05.	3h	Verbesserung Dokumente und abschließendes Gespräch

Tabelle 3: Umsetzung

9.3 Quellcode

Der Quellcode für das Plotter-Projekt ist auf GitHub verfügbar:
`https://github.com/TheD4rkCoder/Plotter/tree/master/src/main/java/com/example/plotter`

9.4 Anwendungsdiagramm

Das Anwendungsdiagramm kann mit diesem Link abgerufen werden:
`https://lucid.app/lucidchart/3770db6f-dc16-404d-9219-95c7e45f323b/edit?viewport_loc=-131%2C-12%2C1805%2C823%2Cfwoxo931mB.p&invitationId=inv_0618e768-45cb-485d-9af9-8d05938abe2a`

9.5 Use-Case-Diagramm

Das Use-Case-Diagramm finden Sie unter diesem Link:
`https://lucid.app/lucidchart/64b10723-f63c-4e1a-9ac8-d6f3276d1257/edit?view_items=bLvq2pE~~b30&invitationId=inv_95cc8be5-d6fb-4751-bdc9-31c3e56f0414`

9.6 Screenshots

Folgende Screenshots können in den Anhang aufgenommen werden:

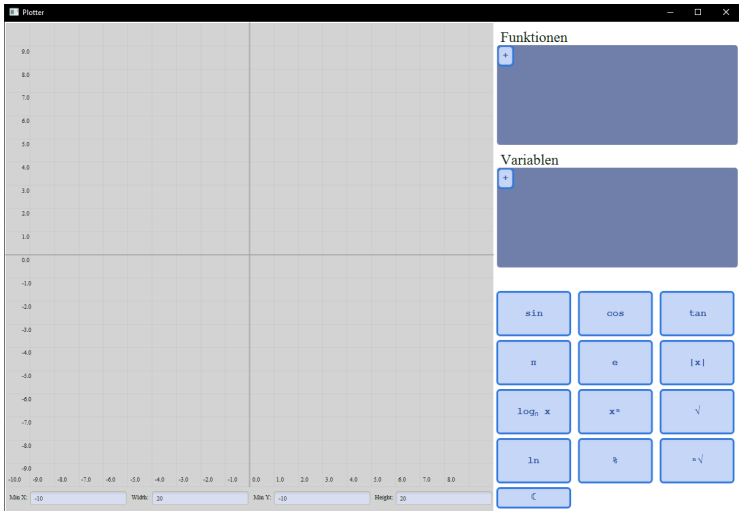

Abbildung 12: Startbildschirm: Leer, ohne Funktionen, im weißen Modus

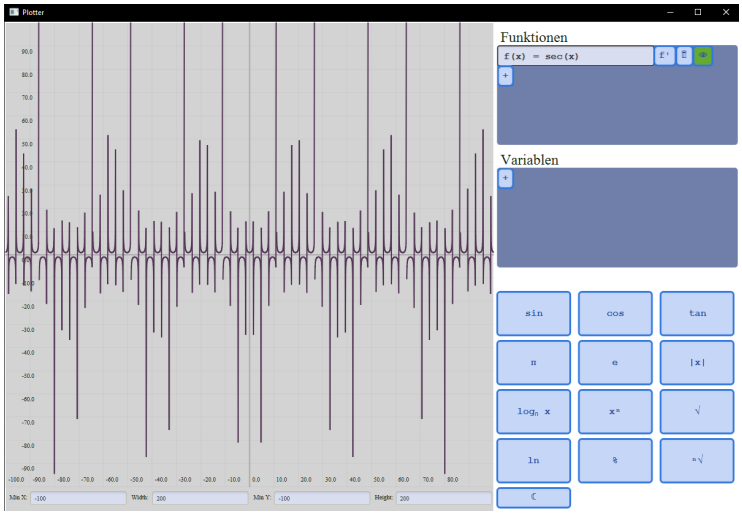

Abbildung 13: Eingabe und Darstellung der Funktion $sec(x)$

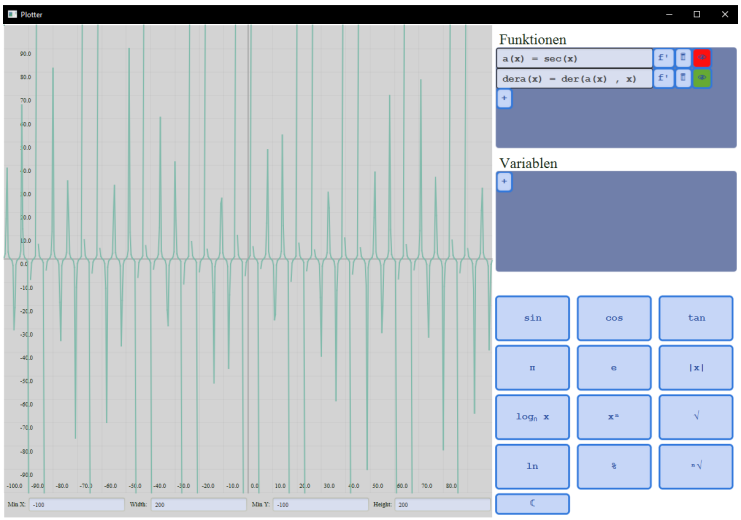

Abbildung 14: Ableitung der $sec(x)$ Funktion und Ausblenden

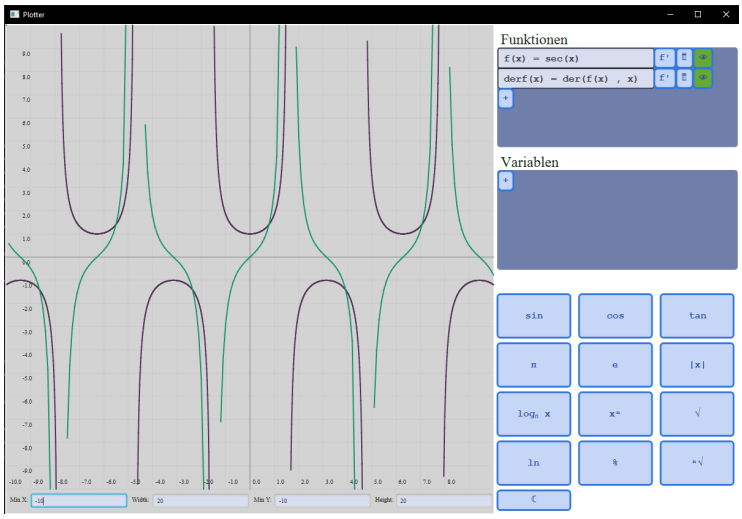

Abbildung 15: Zoom: Gezoomte Ansicht der $sec(x)$ Funktion

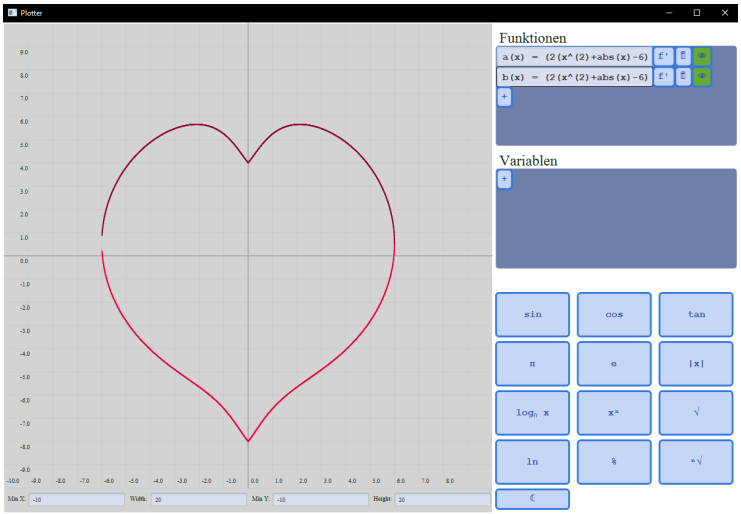

Abbildung 16: Funktion in der Form eines Herzens

9.7 Lizenzinformationen

Dieses Projekt steht unter der MIT-Lizenz, die eine permissive Open-Source-Lizenz ist. Die MIT-Lizenz ermöglicht die uneingeschränkte Nutzung, Kopie, Modifikation und Weiterverteilung des Projekts, solange die Lizenzbestimmungen eingehalten werden.

Die MIT-Lizenz hat folgende Bedingungen:

1. Die Urheberrechts- und Lizenzinformationen müssen in allen Kopien oder wesentlichen Teilen des Projekts enthalten sein.

2. Die Software wird "wie sie ist" bereitgestellt, ohne jegliche ausdrückliche oder stillschweigende Garantie, einschließlich, aber nicht beschränkt auf, die Garantie der Marktgängigkeit, der Eignung für einen bestimmten Zweck und der Nichtverletzung von Schutzrechten.

3. In keinem Fall haften die Autoren oder Inhaber von Urheberrechten für Schäden oder sonstige Haftungsansprüche, die sich aus oder im Zusammenhang mit der Software oder deren Verwendung oder anderem Handeln mit der Software ergeben.

Für weitere Informationen zur MIT-Lizenz können Sie auf folgenden Link verweisen: `https://opensource.org/licenses/MIT`

Durch die Verwendung der MIT-Lizenz wird sichergestellt, dass das Projekt frei zugänglich ist und von anderen Entwicklern zur Zusammenarbeit, Verbesserung und Erweiterung des Projekts verwendet werden kann.

Abbildungsverzeichnis