

Inhaltsverzeichnis

1	Vorwort	2
2	Allgemeines	2
2.1	Definition des Funktionsplotters.....	3
2.2	Allgemeiner Programmablauf	2
3	Eingabe und Ausgabe	4
3.1	Die Eingabe (Klasse Plotter).....	4
3.1.1	Verwendete Objekte.....	4
3.1.2	Aufbau der Klasse Plotter.....	5
3.1.3	Mögliche Eingaben.....	5
3.2	Die Ausgabe (Klasse KoSys).....	6
3.2.1	Funktionsprinzip der Graphenausgabe	7
3.2.2	Skalierung des Graphen	7
3.2.3	Verwendete Objekte.....	8
3.2.4	Aufbau der Klasse KoSys.....	8
4	Berechnung des f(x) Wertes (Klasse engine)	9
4.1	Grundlegendes Vorgehen	9
4.1.2	Schematische Darstellung einer Funktion.....	10
4.2	Aufbau der Klasse engine	10
4.2.1	Aufteilung der Funktion in ein Array	10
4.2.2	Untersuchen des Arrays auf Klammern	10
4.2.3	Berechnung der Teilfunktionen	12
4.2.4	Berechnung des Endwertes	14
5	Literaturverzeichnis	15
6	Erklärung	16
7	Anhang	
7.1	Quellcodes	
7.1.1	Klasse Plotter	
7.1.2	Klasse KoSys	
7.1.3	Klasse engine	

1 Vorwort

Diese Arbeit beschäftigt sich mit der Entwicklung eines Java-Programms für einen Funktionsplotter, der Funktionsgraphen in einem Koordinatensystem darstellt.

Die Funktionen sollen dabei durch Benutzerinnen und Benutzer im Prinzip frei - nur durch einige mögliche Rechenoperationen eingeschränkt - eingegeben werden können. Auch der Bereich des Koordinatensystems, innerhalb dessen die Ausgabe des Graphen erfolgen soll, soll von Benutzern/Benutzerinnen eingegeben werden können.

Der Funktionsplotter soll allen Schülerinnen und Schülern zur Nutzung im Unterricht (z.B. in Mathematik oder Physik) oder auch außerhalb zur Verfügung gestellt werden.

PLOT PLEASURE for all

Claudine Zillmann, Gregor Kulikowski, Sabrina Langele und Tuba Hezer

2 Allgemeines

In den folgenden zwei Abschnitten wird der Begriff „Funktionsplotter“ definiert und ein grober Programmablauf beschrieben.

2.1 *Definition: Funktionsplotter*

„Funktionsplotter dient meist als Bezeichnung für ein Computerprogramm, welches zu einzugebenden mathematischen Funktionen den Funktionsgraphen zeichnet.“¹

2.2 *Allgemeiner Programmablauf*

Der theoretische Programmablauf eines Funktionsplotters erscheint vorerst relativ einfach zu sein:

¹ Akademie.de – Netlexikon, Stichwort Funktionsplotter.
<http://www.net-lexikon.de/Funktionsplotter.html>, Rev. 2004-02-27

Der Benutzer gibt eine Funktion $f(x)$ in Abhängigkeit von x und den Bereich in dem diese gezeichnet werden soll an.

Das x wird zuerst durch den ersten Wert aus dem angegebenen Bereich, x_0 ersetzt. Danach wird der sich daraus ergebende Term $f(x_0)$ berechnet und in dem Punkt $P_1(x_0 | f(x_0))$ wird im Koordinatensystem ein Punkt eingezeichnet und das x wird um einen minimalen Wert erhöht.

Dieser Vorgang wiederholt sich solange, bis das x den durch den Bereich festgelegten maximalen Wert erreicht und somit den Graphen durch viele kleine Punkte darstellt.

Die praktische Umsetzung der Theorie ist jedoch weitaus komplizierter und kleinschrittiger. Grundlegend unterteilt man Programme in drei Aufgabengruppen:

- Eingabe
- Verarbeitung
- Ausgabe

Diese drei Gruppen werden durch drei Klassen realisiert, die diese Aufgabengruppen abdecken:

- Plotter
- Engine
- KoSys

Die Klasse Plotter, die Hauptklasse bzw. das Hauptprogramm, übernimmt die Eingaben und die Kommunikation zwischen den anderen beiden Klassen.

Die Klasse Engine stellt alle Dienste zur Verfügung, die für komplexe Rechenoperationen nötig sind.

Die Klasse KoSys stellt das Koordinatensystem und den „Drucker“ des Programms dar, d.h. sie ist für die graphischen Ausgaben zuständig.

Zusätzlich gibt es noch die Klasse „start“, die lediglich die Klasse Plotter aufruft, somit für den Programmablauf irrelevant ist und deshalb nicht weiter bearbeitet wird.

3 Eingabe und Ausgabe

In diesem Abschnitt werden die Programmabläufe der Klassen „Plotter“ und „KoSys“, welche für die Ein- und Ausgabe zuständig sind, näher dokumentiert. Die kompletten Quelltexte mit zusätzlichen Dokumentationen befinden sich im Anhang.

3.1 Die Eingabe (Klasse Plotter)

Die Eingabe besteht aus drei Eingabefeldern; eines für die Funktion, eines für den Bereich bzw. die Skalierung und ein drittes für die Zeichengenauigkeit. Die Eingabefelder sind Objekte der Klasse „TextFeld“, welche zu der Klassensammlung „KnopfundCo“ gehört. (Siehe Abs. 5 Literaturverzeichnis)

3.1.1 Verwendete Objekte

Die Klasse Plotter verwendet folgende Objekte:

-Klassen aus der Sammlung „StiftundCo“:

- **display**, ein Objekt der Klasse „Bildschirm“
Das Objekt *display* stellt die Grundlage für eine graphische Oberfläche dar, bildlich dargestellt: „Die Tafel zum Zeichnen“
- **tast**, ein Objekt der Klasse „Tastatur“
Das Objekt *tast* stellt die Tastatur des Computers zur Verfügung, die für die Eingaben benötigt wird.

-Klassen aus der Sammlung „KnopfundCo“

- **tfeld1, tfeld2, tfeld3** sind Objekte der Klasse „TextFeld“ und stellen Texteingabefelder auf dem *display* dar.
- **bfeld1, bfeld2, bfeld3, ..., bfeld7** sind Objekte der Klasse „BeschriftungsFeld“ und ermöglichen es, Schriftzüge auf dem *display* darzustellen.

-Klassen, die direkt zum Funktionsplotter gehören:

- ***kosys1*** ist ein Objekt der Klasse „KoSys“
Das Objekt ist für die graphische Ausgabe des Koordinatensystems und des Graphen zuständig. (Abs. 3.2)
- ***engine1*** ist ein Objekt der Klasse „engine“
Das Objekt realisiert die Algorithmen, die zur Berechnung eines Funktionswertes an der Stelle x benötigt werden. (Abs. 4)

3.1.2 Aufbau der Klasse Plotter

Die Klasse Plotter ist vergleichsweise kompakt aufgebaut. Es werden zuerst alle benötigten Objekte und Variablen deklariert und initialisiert (*Quelltext: Klasse Plotter, Zeile 1-56*). Die folgende Schleife bewirkt, dass das Programm auf Eingaben wartet, bis *ENTER* gedrückt wird (*Quelltext: Klasse Plotter, Zeile 59-65*).

Sobald *ENTER* gedrückt wird, werden zuerst die Achsen mit dem gewählten Bereich beschriftet (*Quelltext: Klasse Plotter, Zeile 67-71*). Anschließend werden die Eingaben Variablen zugewiesen, Textvariablen und Zahlvariablen konvertiert (String to Double) und der minimale und maximale x -Wert, der aus dem Bereich errechnet wird, bestimmt (*Quelltext: Klasse Plotter, Zeile 73-95*).

Danach folgt eine Schleife, die den aktuellen x -Wert generiert und die Dienste zum Berechnen des Funktionswertes aufruft. Die erhaltenen Werte werden dann skaliert (Abs. 3.2.2) und der Klasse KoSys (Abs. 3.2) zur Ausgabe übergeben.

3.1.3 Mögliche Eingaben

Eingabefeld 1:

Funktionen können folgende Operatoren beinhalten:

+	Addition	$x + y$
-	Subtraktion	$x - y$
*	Multiplikation	$x \cdot y$
/	Division	$x : y$

^	Exponent	x^y
sqt()	Quadratwurzel	\sqrt{x}
sin()	Sinus	$\sin x$
cos()	Kosinus	$\cos x$
tan()	Tangens	$\tan x$
(Klammer auf	
)	Klammer zu	

Bitte beachten: Bitte Punkt als Komma benutzen; 0.01 nicht 0,01!!!

Bei (-x...) oder (-4...) bitte (0-x...) oder (0-4...) schreiben

Eingabefeld 2:

Der Bereich muss eine positive, reelle Zahl sein. Es wird vom –Bereich bis zum +Bereich geplottet.

Eingabefeld 3:

Die Genauigkeit muss eine positive reelle Zahl kleiner 1 sein. Je kleiner der Wert ist, desto genauer wird die Funktion gezeichnet (relevant bei großen Steigungen). Jedoch sinkt die Zeichengeschwindigkeit bei kleineren Werten.

Gute Ergebnisse werden mit Werten zwischen 0.0001 und 0.001 erreicht.

Bitte beachten: Bitte Punkt als Komma benutzen; 0.01 nicht 0,01!!!

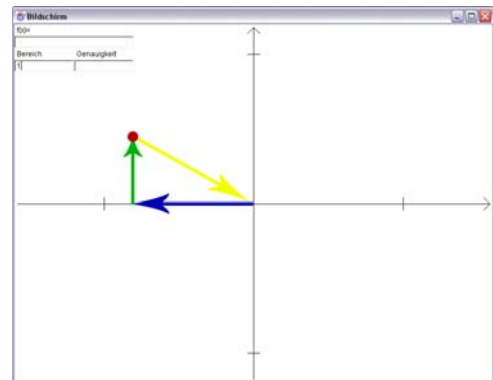
3.2 Die Ausgabe (Klasse KoSys)

Die Ausgabe besteht aus zwei Komponenten, dem statischen Koordinatensystem und dem variablen Graphen. Das Koordinatensystem wird im Gegensatz zum Graphen pauschal beim Aufruf des Programms gezeichnet. Dies ist möglich, da es unabhängig von Funktion und Variablen ist und somit immer identisch ist. Der Graph kann jedoch erst nach Eingabe der Funktion gezeichnet werden, da dieser erst durch die Eingabe bestimmt wird.

3.2.1 Funktionsprinzip der Graphenausgabe

Prinzipiell zeichnet man Funktionen mit der Hand, indem man eine Wertetabelle erstellt, diese Punkte in ein Koordinatensystem einträgt und sie miteinander verbindet. Nach einem ähnlichen Prinzip arbeitet die Ausgabe des Plotters: Zu einem x-Wert wird der y-Wert errechnet und diese werden dann wie folgt rekursiv in das Koordinatensystem eingetragen:

1. Ein Ausgabeobjekt, das sich im Ursprung befindet bewegt sich um x – Stellen in horizontaler Richtung; nach links wenn $x < 0$; nach rechts wenn $x > 0$ (Quelltext: Klasse KoSys, Z 66f).
2. Das Ausgabeobjekt bewegt sich um y – Stellen in vertikaler Richtung nach oben falls $y > 0$; nach unten falls $y < 0$ (Quelltext: Klasse KoSys, Z 68f).
3. Das Ausgabeobjekt zeichnet einen Punkt (Quelltext: Klasse KoSys, Z 72.)
4. Das Ausgabeobjekt bewegt sich zurück zum Ursprung. (Quelltext: Klasse KoSys, Z 74 & Z 77-82)



Dies wird solange wiederholt, bis das x seinen maximalen Wert erreicht hat.

Vorteile dieses Systems sind die einfache Umsetzbarkeit und Skalierbarkeit. Ein Nachteil ist jedoch, dass bei großen Steigungen der Graph nicht mehr als durchgezogene, sondern als gepunktete Linie dargestellt wird. Dies ist der Grund, weshalb das Eingabefeld „Genauigkeit“ implementiert wurde, denn je kleiner dieser Wert gewählt wird, umso steilere Graphen können korrekt dargestellt werden.

3.2.2 Skalierung des Graphen

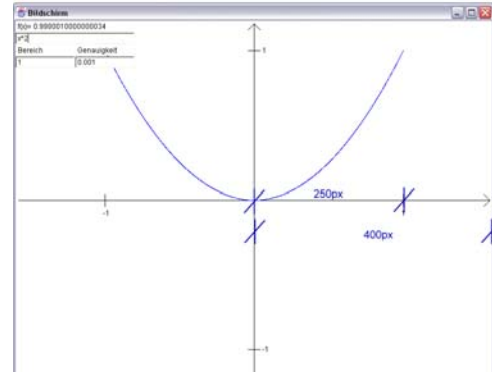
Um das Prinzip der Skalierung des Graphen zu erläutern, muss das Koordinatensystem zuerst genauer betrachtet werden:

Insgesamt ist das Fenster 800 x 600 Pixel groß, folglich ist jeder Quadrant 400 x 300 Pixel groß. Die Einheitsmarke auf den Achsen ist jeweils 250 Pixel vom Ursprung entfernt. Bei einem Plottbereich von 1 stellt diese Marke eine Längeneinheit dar, die somit aus 250 Pixeln besteht. Da die x - und y -Werte jedoch

in Pixeln vorliegen, müssen sie mit einem entsprechenden Streckungsfaktor multipliziert werden, damit sie an der richtigen Stelle gezeichnet werden. Dieser Faktor ergibt sich aus der Entfernung der Einheitsmarke vom Ursprung (250 Pixel) und dem Plottbereich (variabel). Die Länge von 250 Pixeln muss auf die Anzahl der Einheiten (dem Bereich) aufgeteilt werden. Daraus ergibt sich der Streckungsfaktor

$$s = \frac{250}{\text{Bereich}}$$

(Quelltext: Klasse Plotter, Z 87-88).



3.2.3 Verwendete Objekte

-Klassen aus der Sammlung „StiftundCo“:

- **st1**, ein Objekt der Klasse „BuntStift“ der Oberklasse „Stift“
Das Objekt *st1* ist das Pendant zum Objekt *display*, *st1* kann Linien, Formen und Schriften auf dem *display* darstellen.

3.2.4 Aufbau der Klasse KoSys

Die Klasse KoSys beinhaltet drei Dienste, den Konstruktor und die Aufträge *plott* und *reset*. Der Konstruktor wird automatisch bei der Initialisierung der Klasse KoSys (Quelltext: Klasse Plotter, Z 43) aufgerufen. Dieser initialisiert wiederum einen Buntstift, welcher das Koordinatensystem zeichnet (Quelltext: Klasse KoSys, Z 9-60).

Der Auftrag *plott* wird ebenfalls von der Klasse Plotter aufgerufen (Quelltext: Klasse Plotter, Z 91), welche die aktuelle x-y-Koordinate als Parameter an den Auftrag *plott* übergibt, der diese dann nach dem in Abs. 3.2.1 beschriebenen Prinzip zeichnet (Quelltext: Klasse KoSys, Z 63-82).

4 Berechnung des $f(x)$ Wertes (Klasse engine)

Gegeben sei die Funktion $f(x) = \frac{3 \cdot \sin(x^2 + 13)}{x - 1}$. Diese Funktion beinhaltet alle

Rechenoperationen, die der Funktionsplotter beherrscht (Multiplikation, Division, Addition, Subtraktion, Exponenten, den Sinus als erweiterte Funktion und Klammern) und ist somit optimal um das Vorgehen des Funktionsplotters bei der Berechnung des $f(x)$ -Wertes zu erläutern.

Es folgen zuerst allgemeine Überlegungen, die dann auf das Programm übertragen werden.

4.1 Grundlegendes Vorgehen

Um einen Term zu berechnen wendet man normalerweise die Vorrangregeln an, d.h. man wertet die Operatoren (Rechenzeichen) nach folgendem Schema der Reihe nach aus:

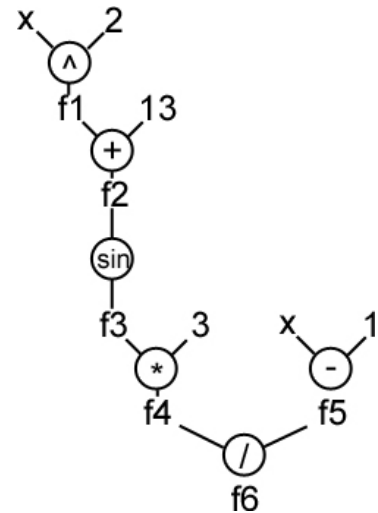
Innere Klammer vor äußerer Klammer vor Exponent vor Multiplikation (Division) vor Addition (Subtraktion). Brüche werden dabei als Divisionen betrachtet, dessen Dividend der eingeklammerte Zähler und Divisor der eingeklammerte Nenner ist.

$$\text{Bsp.: } \frac{(25 + 12) * 3}{13 - 34} = ((25 + 12) * 3) : (13 - 34)$$

Jedoch ist dieses Schema nicht direkt auf eine Funktion anwendbar, da diese von einer Variable abhängt. Um die schrittweise Berechnung einer Funktion trotzdem visuell verdeutlichen zu können, stellt man diese schematisch in Form eines Baumes dar.

4.1.2 Schematische Darstellung einer Funktion

Die schematische Darstellung einer Funktion verdeutlicht die einzelnen Rechenoperationen, die nötig sind um einen Wert $f(x_1)$ der Funktion f an der Stelle x_1 zu bestimmen. Um eine Funktion in ein Baumschema zu übertragen, ist es, diese in einzeiliger Form zu schreiben: $(3 * \sin(x^2 + 13)) / (x - 1)$. Diese Form ergibt den rechts dargestellten Baum:



Die Auswertung erfolgt rekursiv von oben nach unten indem man jeweils zwei Operanden mit dem Operator in ihrem Knotenpunkt verknüpft und das Ergebnis in die nächst - tiefere Ebene schreibt. Dies wird wiederholt, bis man in der untersten Ebene das Ergebnis erhält.

4.2 Aufbau der Klasse engine

Die Klasse engine ist in zwei Aufgabenbereiche unterteilt. Der erste Aufgabenbereich, der durch den Dienst „parseFunktion“ (*Quelltext: Klasse engine, Z 27-227*) realisiert wird, umfasst den analytischen Aufgabenteil. Dieser Dienst „schneidet“ die Funktion für den Dienst „getfx“ zu, der den zweiten Aufgabenbereich darstellt. Der Dienst „getfx“ (*Quelltext: Klasse engine, Z245-375*) ist für alle Rechenoperationen zuständig.

Auf das Baumschema bezogen unterteilt der Dienst „parseFunktion“ die Funktion in Ebenen und Zweige und der Dienst „getfx“ verknüpft die Zweige mithilfe der Knotenpunkte.¹

4.2.1 Aufteilung der Funktion in ein Array

Die erste Aufgabe des Dienstes „parseFunktion“ ist die Funktion, die als Zeichenkette vorliegt, in ein Array (*im Quelltext fArr[]*) zu unterteilen. Der Sinn

¹Idee nach: Uni-Magdeburg – Belegaufgaben Sommersemester 2000 Aufgabe7
http://www.witi.cs.uni-magdeburg.de/iti_db/lehre/algds/beleg2/beleg2.html, Rev. 2004-03-01

dieser Unterteilung ist, die Abfrage der Operatoren und Operanden möglichst einfach zu gestalten.

In einer Zeichenkette wird jeder einzelnen Ziffer, jedem Zeichen und jedem Buchstaben ein eigener Index zugeordnet. Dies lässt sich am besten mit Hilfe einer Tabelle verdeutlichen:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
(3	*	s	i	N	(x	^	2	+	1	3))	/	(x	-	1)

Wie man sieht, wird jedes Zeichen der Funktion $(3 * \sin(x^2 + 13)) / (x - 1)$ in eine eigene Spalte geschrieben und kann über seinen Index (1. Zeile der Tabelle) abgerufen werden. Der Nachteil bei diesem System ist, dass das Programm bei jeder Operation überprüfen müsste, ob eine ein- oder mehrstellige Zahl vorliegt. Um dieses Problem zu umgehen wird die Zeichenkette in ein Array geschrieben, das man durch folgende Tabelle beschreiben kann:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
(3	*	sin	(x	^	2	+	13))	/	(x	-	1)

Die Vorteile des Arrays liegen einerseits in der direkten Abrufbarkeit der Operanden und Operatoren, sowie auch im späteren Berechnen der Funktion.

Das Umschreiben der Zeichenkette ins Array erfolgt spaltenweise von links nach rechts (*Quelltext: Klasse engine, Z54-136*). Jede Spalte wird auf Ziffern, Operatoren und erweiterte Funktionen überprüft. Wenn eine Ziffer vorliegt, wird überprüft, ob die Zahl mehrstellig ist, indem die folgenden Spalten ebenfalls auf Ziffern überprüft werden. Zusammengehörige Zifferketten werden als Zahl ins Array geschrieben. (*Quelltext: Klasse engine, Z60-72*).

Analog wird die Zeichenkette auf erweiterte Funktionen untersucht. (*Quelltext: Klasse engine, Z80-125*)

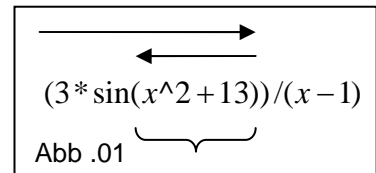
Um das spätere Berechnen der Funktion zu erleichtern, werden die x nicht als x ins Array übertragen, sondern direkt durch den aktuellen x-Wert ersetzt, damit der Term im Array direkt berechnet werden kann. (*Quelltext: Klasse engine Z127ff*) Zusammengefasst liegt jetzt ein Array `fArr[]` mit der Länge *gesamt* vor, in der die Funktion in Operatoren und Operanden unterteilt ist.

4.2.2 Untersuchen des Arrays auf Klammern

Bei der Berechnung eines Terms wird, wie in Abs. 4.1 beschrieben, die innerste Klammer zuerst berechnet. Dies gilt auch für den Funktionsplotter, jedoch dass dieser die innerste Klammer erst „finden“ muss. Dazu noch mal die Tabelle des Arrays:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
(3	*	sin	(x	^	2	+	13))	/	(x	-	1)

Die innere Klammer dieser Funktion wird in Spalte 4 geöffnet und in Spalte 10 wieder geschlossen. Damit das Programm die Zusammengehörigkeit dieser Stellen findet, untersucht es schrittweise jede Spalte, ob sich eine „Klammer auf“ oder eine „Klammer zu“ in dieser Spalte befindet und speichert dessen Stellen in *klammerA* und *klammerZ*



ab. Bei der ersten gefundenen „Klammer zu“ wird die Stelle der zuletzt gefundenen „Klammer Auf“ abgefragt und somit der Bereich der Klammer ermittelt. (Abb. 01)

Bevor die Klammer vom Dienst getfx berechnet wird, wird überprüft, ob vor der Klammer eine erweiterte Funktion steht, um diese beim Berechnen direkt zu beachten. Die Klammer wird nun isoliert vom Rest der Funktion berechnet und durch den erhaltenen Wert ersetzt, analog zum Auswerten von Bäumen. (Quellcode: Klasse engine Z162-213)

4.2.3 Berechnung der Teilfunktion

Die Berechnung der isolierten Klammer erfolgt ebenfalls analog zur Auswertung eines Baumes. Zwei Zahlen werden mittels der zwischen ihnen stehenden Operanden verknüpft:

0	1	2	3	4
x	^	2	+	13

Es werden zuerst die Spalten der Operanden gesucht, die in diesem Fall 1 und 3 sind und die Knotenpunkte im Baumschema darstellen. Nun wird die Spalte vor dem Knotenpunkt mit der Spalte nach dem Knotenpunkt mittels des Opera-

tors im Knotenpunkt verknüpft. Das Ergebnis L_1 wird dann in die Spalte vor dem Operator geschrieben und der Rest des Terms rückt um zwei Spalten auf:

0	1	2
L_1	+	13

Somit verkürzt sich das Array der Terms um zwei Stellen. Dieses Vorgehen wird rekursiv ausgeführt, bis das Array nur noch eine Spalte mit dem Ergebnis L_2 enthält:

0
L_{2a}

(Quellcode: Klasse engine Z.255-352)

Falls vom Dienst noch eine erweiterte Operation der Klammer übergeben wird, wird diese noch auf L_2 angewandt:

$$L_2 = \sin(L_{2a})$$

(Quellcode: Klasse engine Z. 354-372)

Das Array, das die gesamte Funktion enthält und aus dem die Teilfunktion isoliert wurde, wird nun im Bereich der Klammer durch L_2 ersetzt:

Aus

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
(3	*	sin	(x	^	2	+	13))	/	(x	-	1)

wird

0	1	2	3	4	5	6	7	8	9	10
(3	*	L_2)	/	(x	-	1)

(Quelltext: Klasse engine Z. 202 & 229-243)

4.2.4 Berechnung des Endwertes

Den Endwert erhält man durch rekursives Anwenden (*Quelltext: Klasse engine, Schleife Z. 165-211*) der unter 4.2.2 und 4.2.3 beschriebenen Vorgänge, denn es werden alle Klammern nacheinander berechnet, woraus im Array fArr[] ein klammerloser Term entsteht, der nach dem in 4.2.3 beschriebenen Prinzip berechnet wird. (*Quelltext: Klasse engine Z. 216-221*)

Die Funktion wird also schrittweise verkürzt, bis nur noch eine Spalte mit dem Ergebnis vorliegt, welches an die Klasse Plotter, die es zur Ausgabe weitergibt, zurückgegeben wird.

5 Literaturverzeichnis

- AVG-Wesel, Klassenbibliotheken,
<http://www.avg-wesel.de/fb/inf/lfb/sumklassenbibliotheken/default.htm>
Rev. 2004-03-01
 - AVG-Wesel, Klasse Bildschirm (StiftundCo)
<http://www.avg-wesel.de/fb/inf/lfb/sumklassenbibliotheken/bildschirm.htm>
Rev. 2004-03-01
 - AVG-Wesel, Klasse Stift (StiftundCo)
<http://www.avg-wesel.de/fb/inf/lfb/sumklassenbibliotheken/stift.htm>
Rev. 2004-03-01
 - AVG-Wesel, Klasse Hilfe (StiftundCo)
<http://www.avg-wesel.de/fb/inf/lfb/sumklassenbibliotheken/hilfsbibliothek.htm>
Rev. 2004-03-01
 - AVG-Wesel, Klasse TextFeld (KnopfundCo)
<http://www.avg-wesel.de/fb/inf/lfb/sumklassenbibliotheken/textfeld.htm>
Rev. 2004-03-01
 - AVG-Wesel, Klasse BeschriftungFeld (KnopfundCo)
<http://www.avg-wesel.de/fb/inf/lfb/sumklassenbibliotheken/beschriftungsfeld.htm>
Rev. 2004-03-01
 - Learn-Line NRW (Landesinstitut für Schule), Klasse Textwerkzeug
www.learn-line.nrw.de/angebote/oop/medio/sum/sum-werkzeuge.pdf
Rev. 2004-03-01
 - Uni-Magdeburg – Belegaufgaben Sommersemester 2000
http://www.witi.cs.uni-magdeburg.de/iti_db/lehre/algds/beleg2/beleg2.html
Rev. 2000-04-04
 - Alle Grafiken sind selbstständig mit dem Programm „Adobe Photoshop“ erstellt worden.
 - Als Java Editor wurde der Java Editor von Gerhard Römer benutzt
<http://www.bildung.hessen.de/abereich/inform/skii/material/java/editor.htm>
 - Alle Dokumente liegen im Anhang auf CD-ROM vor
-

7 Anhang

Der Anhang beinhaltet die Quellcodes der Klassen Plotter, KoSys und engine, sowie die Ausdrücke der Internetquellen und eine CD-ROM mit dem Funktionsplotter, der erforderlichen Software und einer „liesmich.txt“, die Hilfestellungen zur Installation beinhaltet.
