Josef Lüth

Matex
Algebra Berechnungen am Computer

Inhaltsverzeichnis

Einleitung

Am Anfang stand der Wunsch, Gleichungen mit dem Computer berechnen zu können, um nicht auf das aufwendige und fehleranfällige Berechnen der Gleichungen per Hand angewiesen zu sein.

Eine derartige Funktion ist nur bei großen Mathematikanwendungen, wie Mathematica und Maple verfügbar. Diese Programme sind aber auf Studenten und Wissenschaftler zugeschnitten, sodass diese aufgrund ihrer Funktionsfülle schwer zu bedienen und teuer[1] sind. Außerdem stehen alle Funktionen in einer geschlossenen Anwendung, sodass man einzelne Module nicht kombinieren und in eigene Anwendungen einbauen kann. In Internetforen gab es immer wieder Anfragen von Entwicklern, die ein Modul zum Berechnen von Gleichungen suchten, dieses existierte aber noch nicht.

Da mich die Technik faszinierte, Terme[2] mit dem Computer zu berechnen, entschloss ich mich, ein einfaches und leistungsfähiges Tool –vor allem für Schüler- zum Berechnen von Termen bzw. Gleichungen zu entwickeln.

Dieses Programm –Matex- sollte ein erstes opensource[3] Modul zum Berechnen von Termen werden, sodass es jeder weiterentwickeln und in seine Anwendung einbauen kann, ohne die Technologie neu entwickeln zu müssen.

[1] Eine Lizenz kostet bei Mathematica $1495 und bei Maple $770
[2] Teil einer Gleichung, zum Beispiel: $2x^2/x$
[3] Quellcode wird veröffentlicht, darf weiterentwickelt und in eigene Anwendungen eingebaut werden

Vorgehensweise

Die Architektur

Zum Berechnen von Termen benötigt Matex einen Parser und einen Evaluator.

Der **Parser** (eng. grammatikalischer Analytiker) teilt den Term in seine Summanden, Faktoren, Potenzen, Funktionen, Variablen und Zahlen auf.

Anschließend ruft der Parser einen **Evaluator** (eng. Auswerteinrichtung) auf, der die Elemente addiert, subtrahiert, multipliziert oder potenziert.

Der Parser

Der Term muss, wie beim Compilieren einer Anwendung, analysiert und in seine Bestandteile zerlegt werden. Anschließend werden die Elemente nach den mathematischen Regeln angeordnet.

Der Übersetzungsprozess besteht aus zwei Teilen: Aus der lexikalischen- und aus der syntaktischen Analyse.

Die lexikalische Analyse

Der Parser übergibt den Term an den Scanner, der die lexikalische Analyse durchführt.

Bei der lexikalischen Analyse zerlegt der Scanner den Term in Operationszeichen (Plus, Minus…), Zahlen, Variablen und Funktionen (e, Pi).

Jedes Element wird als Token (eng. Zeichen) gespeichert und dem Parser übergeben.

Damit ist es möglich, Operationszeichen, Zahlen Variablen, Potenzen und Funktionen eindeutig zu identifizieren und zu unterscheiden, ohne auf die Elemente selbst zurückgreifen zu müssen.

Die Token werden dann dem Parser übergeben (siehe Abbildung 1).

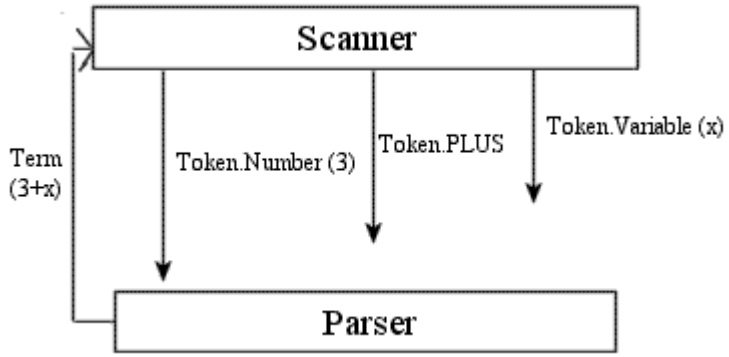

Abbildung 1: Beziehung zwischen Scanner und Parser (Beispiel: 3+x)

Die syntaktische Analyse

Bei der syntaktischen Analyse werden die Token vom Parser in einen Syntaxbaum (oder Ableitungsbaum) übertragen.

Dabei werden die Token der Reihe nach überprüft und es wird für jedes Token ein Knoten (oder eng. Node) erstellt.

Die Knoten von Operation (wie Plus, Multiplikation…) enthalten jeweils zwei weitere Knoten.
Zum Beispiel kann der Knoten Plus einen Knoten Multiplikation, für den ersten Summanden und einen Knoten Variable, für den zweiten Summanden, enthalten.

Beim Parsen werden zuerst die Plus- und Minusknoten, dann die Multiplikations- und Divisionsknoten, dann die Potenzialrechnungsknoten und zuletzt die Zahlen- und Variablen Knoten erstellt (siehe Abbildung 2).

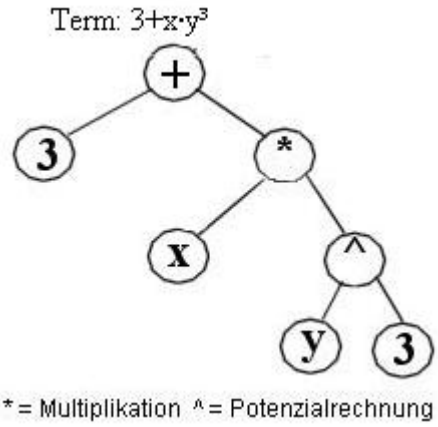

Abbildung 2: Ein Syntax Baum (Beispiel: 3+x·y³)

Jeder Knoten wird einzeln in der Klasse Node gespeichert.

Der Rekursive Abstieg

Nun wird der Rekursive Abstieg angewandt.

Ein praktisches Beispiel für den Rekursiven Abstieg ist der „Jugend Forscht“ Wettbewerb.

Dabei ist das Ziel, einen Bundessieger (vereinfacht dargestellt) zu ermitteln.
Damit dieser ermittelt werden kann, verlangt der Bundeswettbewerb, dass sich jeder Landeswettbewerb für einen Sieger entscheidet. Jeder der Landeswettbewerbe verlangt wiederum von seinen Regionalwettbewerben, dass sich diese jeweils für einen Sieger entscheiden, dann werden die Landessieger ermittelt.

Mit dieser Technik arbeit auch der Parser.

Hierbei sind die Knoten die Wettbewerbe, auch sie haben ein Ergebnis zu ermitteln.

Die Knoten enthalten dafür die Methode *evaluate*, die das Ergebnis der Operation, die Zahl oder die Variable zurückgibt. Für die Berechnung wird der Evaluator aufgerufen.

Dieser Ablauf ist wie eine Kettenreaktion: Wenn man den ersten Knoten berechnen lässt, lässt er das Ergebnis der zwei untergeordneten Knoten berechnen und diese lassen wiederum das Ergebnis der unteren Knoten berechnen… und so erhält man schließlich vom obersten Knoten das Ergebnis (siehe Abbildung 3).

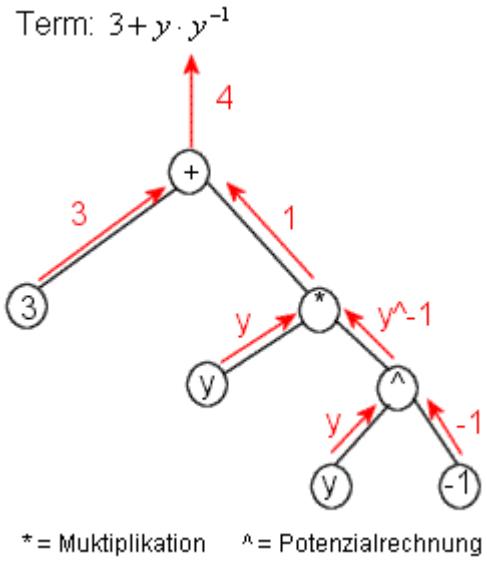

Abbildung 3: Die Ergebnisse werden an den nächst höheren Knoten übergeben (Beispiel: $3+y\cdot y^{-1}$)

Der Evaluator

Mit dem Evaluator lassen sich mathematische Operationen berechnen.
Dafür enthält der Evaluator Methoden, mit denen sich Klammern, Zahlen und Variablen addieren, subtrahieren, multiplizieren, dividieren und potenzieren lassen.

Das Additionsverfahren

Zum Addieren ruft der Parser die Methode *addiere* auf und übergibt ihr zwei Summanden.

Für das Additionsverfahren habe ich den Container-Vergleich-Algorithmus entworfen.

Dabei werden die Zahlen und Variablen jedes Summanden getrennt und als Container in einer Summandentabelle gespeichert (siehe Abbildung 4).

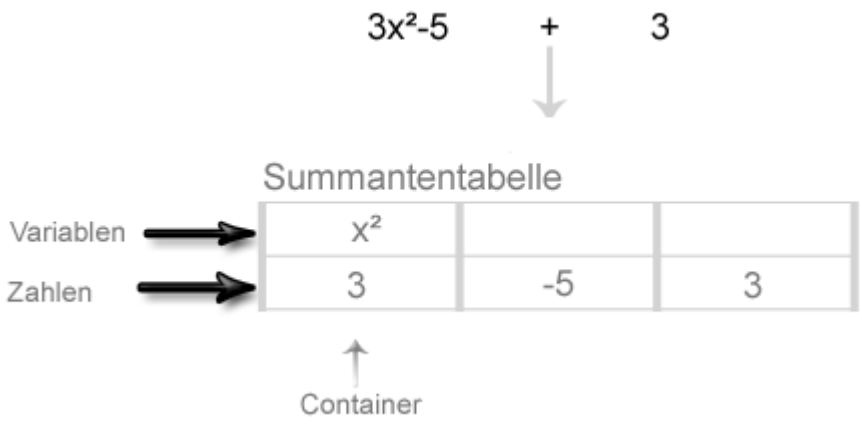

Abbildung 4: Erzeugen einer Summandentabelle

Nun werden die Container verglichen.

Wenn in Containern die Variablen gleich sind bzw. Container keine Variablen enthalten, werden die beiden Zahlen (bzw. Werte) der Container addiert (siehe Abbildung 5).
Zum Addieren der einzelnen Zahlen wird die Additionsmethode von Java verwendet.

Anschließend wird die Tabelle wieder in einen Term umgewandelt.

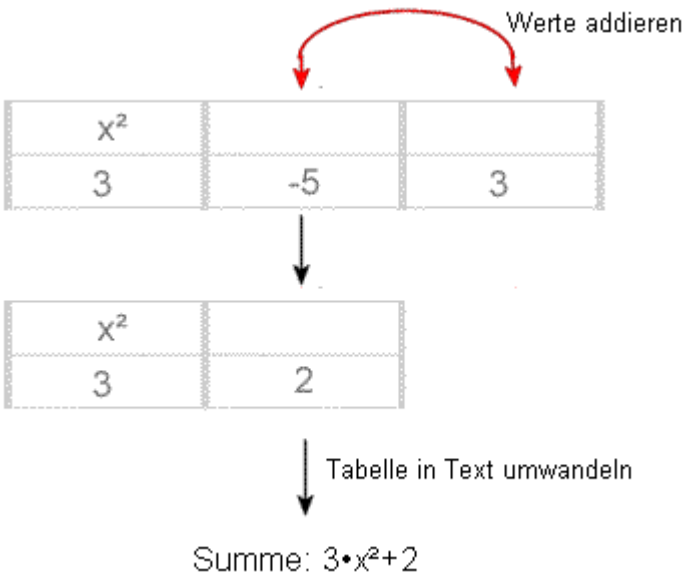

Abbildung 5: Die Werte von gleichen bzw. nicht vorhandenen Variablen werden addiert, dann wird die Tabelle wieder in einen Term umgewandelt

Schließlich übergibt die Additionsmethode das Ergebnis zurück.

Das Subtraktionsverfahren

Zum Subtrahieren wird die Methode *subtrahiere* aufgerufen, der Methode wird dabei ein Minuend und ein Subtrahend übergeben.

Beim Subtrahieren wird kein eigenes Berechnungsverfahren gebraucht.
Die Methode Subtraktion ruft die Methode *multipliziere* auf und lässt den Subtrahenden mit -1 multiplizieren, anschließend wird die Methode *addiere* aufgerufen und der negative Subtrahend wird zum Minuend addiert (siehe Abbildung 6).

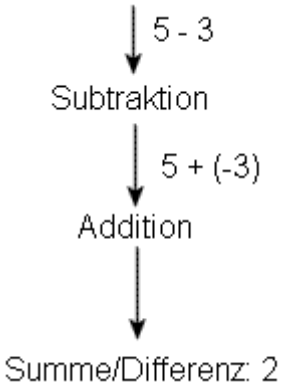

Abbildung 6: Minuend und negativer Subtrahend werden addiert

Anschließend wird das Ergebnis dem Parser übergeben.

Das Multiplikationsverfahren

Die Methode *multipliziere* wird vom Parser zum Multiplizieren aufgerufen, dabei werden der Methode zwei Faktoren übergeben.

Auch beim Multiplizieren ist der Container-Vergleich-Algorithmus anwendbar.

Bei diesem Algorithmus wird aus den Faktoren eine Faktorentabelle erstellt.

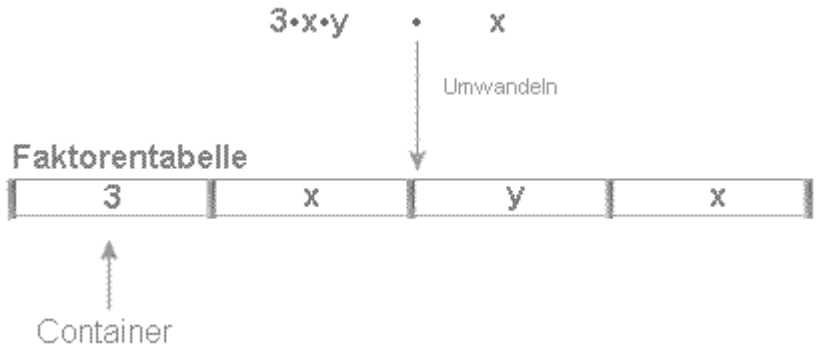

Abbildung 7: Aus den Faktoren wird eine Tabelle erstellt

Dann werden wieder die einzelnen Container (also Faktoren) miteinander verglichen.

Werden zwei Zahlen gefunden, dann werden diese multipliziert.
Wenn in zwei Containern die Variablen identisch sind, werden ihre Potenzen addiert (siehe Abbildung 8).

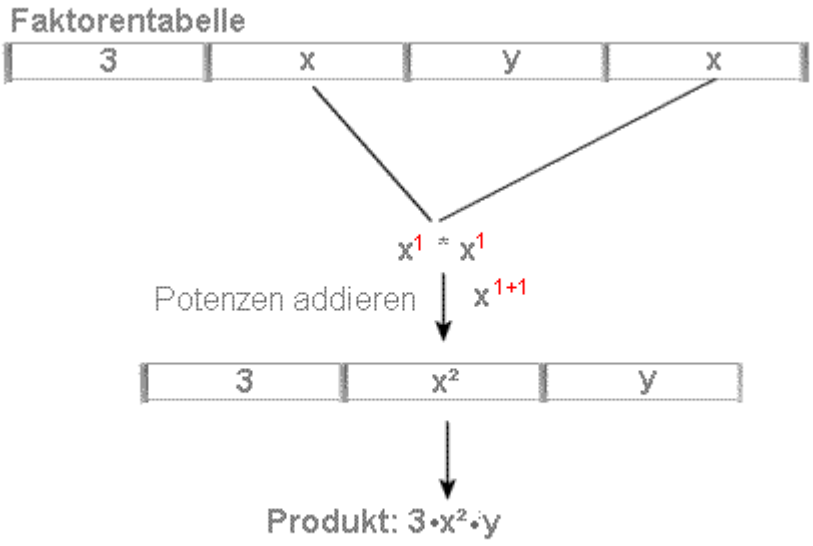

Abbildung 8: Multiplizieren von Variablen (Beispiel: 3 · x · y · x)

Nachdem Variablen und Zahlen multipliziert wurden, wird die Tabelle wieder in einen Term umgewandelt (dabei werden die Variablen alphabetisch angeordnet) und es wird das Ergebnis zurückgegeben.

Wenn mindestens einer der Faktoren eine Klammer enthält, muss ein anderes Berechnungsverfahren angewendet werden.
In diesem Fall werden die zwei Faktoren in ihre Summanden aufgeteilt, dann wird jeder Summand des ersten Faktors mit jeden Summanden des zweiten Faktors multipliziert, die Produkte werden schließlich addiert (siehe Abbildung 9).

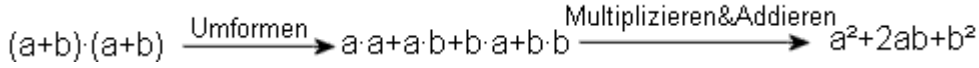

Abbildung 9: Multiplizieren mit Klammern

Das Divisionsverfahren

Die Methode *dividiere* berechnet den Quotienten, dafür muss ihr ein Dividend und ein Divisor übergeben werden.

Beim Dividieren wird, wie beim Subtrahieren, kein eigenes Verfahren zum Berechnen gebraucht.
Dividieren entspricht dem Multiplizieren des Dividenden mit dem Divisor, wobei die Potenz des Divisors mit -1 multipliziert werden muss.

$$x / y = x \cdot y^{-1}$$

Beispiel:

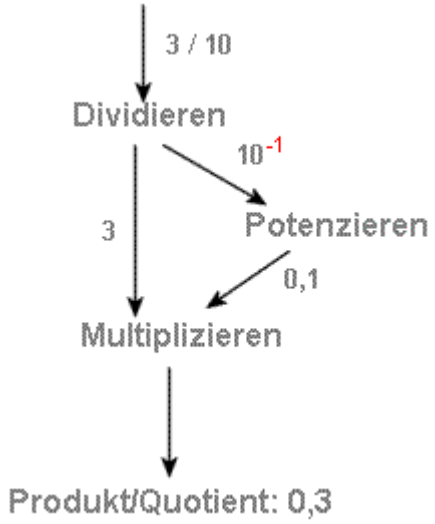

Abbildung 10: Divisionsverfahren (Beispiel: 3 / 10)

Die Methode gibt schließlich den Quotienten zurück.

Das Potenzierverfahren

Zum Potenzieren wird die Methode *potenzieren* aufgerufen, dabei müssen ihr eine Basis und eine Potenz übergeben werden.

Beim Berechnen muss zwischen dem Potenzieren von Summanden und dem Potenzieren von Zahlen, Variablen und Faktoren unterschieden werden.

Potenzieren von Summanden

Beim Potenzieren von Summanden wird die Basis entsprechend der Höhe der Potenz mit sich selbst multipliziert (siehe Abbildung 11). Zum Multiplizieren wird die *multiplizieren* Methode verwendet.

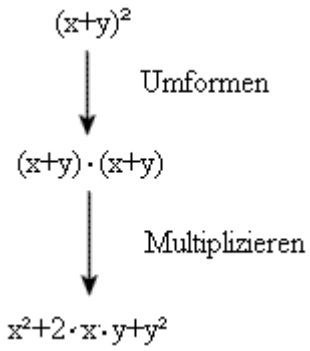

Abbildung 11: Potenzieren von Summanden

Wenn die Potenz negativ ist, wird dem Ergebnis zusätzlich die Potenz -1 hinzugefügt (siehe Abbildung 12).

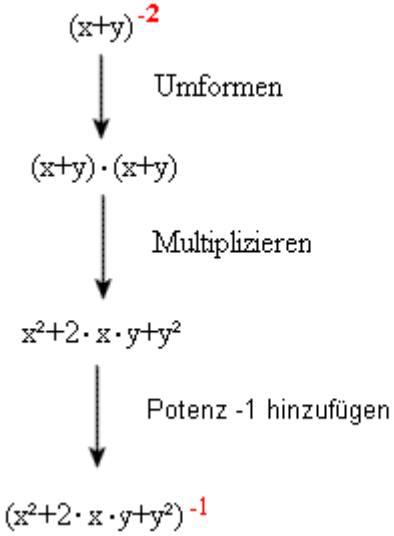

Abbildung 12: Potenzieren mit Summanden und einer negativen Potenz

Potenzieren von Zahlen, Variablen und Faktoren

Beim Potenzieren von Zahlen wird die Potenziermethode von Java verwendet, beim Potenzieren von Variablen wird die Potenz zu der Variable hinzugefügt.
Wenn eine Variable schon eine Potenz hat, wird die neue Potenz mit der schon vorhandenen multipliziert.

Falls die Basis mehrere Faktoren enthält, wird jeder Faktor einzeln potenziert.

Schließlich gibt die Methode *potenzieren* den Potenzwert zurück.

Weitere Berechnungsfunktionen

Anzumerken ist noch, dass Matex keine eigene Methode zur Wurzelberechnung besitzt. Darauf wurde verzichtet, da bei der Eingabe von Termen in den Computer kein eigenes Symbol für die Wurzel üblich ist.
Der Benutzer muss bei der Wurzelberechnung die Potenziermethode verwenden (z.B. $\sqrt{9} = 9^{1/2}$), das hat auch den Vorteil, dass er auch die nte Wurzel ziehen kann.

Matex kann mit der Kreiskonstante Pi und der natürlichen Zahl e rechnen, beide werden vom Programm in Zahlen umgewandelt. Beim Eingeben der beiden Funktionen muss ein Ausrufezeichen vorgestellt werden, damit Matex Pi und e von Variablen unterscheiden kann.

Der Inhalt von Klammern wird bei einer Operation als ganzes der Berechnungsmethode übergeben und mit dem anderen Operator addiert, subtrahiert, multipliziert, dividiert oder potenziert.

Ergebnis

Die Architektur und der Algorithmen wurde so entworfen, dass der Term möglichst effizient und komplett fehlerfrei berechnet werden kann.
So braucht Matex, obwohl es beliebig komplexe Terme fehlerfrei berechnet „nur“ 1500 Codezeilen und benötigt für Berechnungen auf modernen Rechnern meistens nur Millisekunden.

Term	Ergebnis	Berechnungszeit (Millisekunden)
$\frac{31.1035\text{g}}{19{,}3\frac{g}{cm^3}\cdot\Pi\cdot(1{,}5cm)^2}$	0.22799197570333987cm	110
$\frac{x+x}{x+y}$	$2+2\frac{x}{y}$	63
$(a+b)^3$	$a^3+3a^2b+3ab^2+b^3$	94
$\frac{5\cdot(5+\frac{y}{5})}{2}-4+5$	$13{,}5+\frac{1}{2}y$	94

Tabelle 1: Termberechnung mit einem 2 GHz PC

Anwendungen

Matex bietet aber nicht nur leistungsfähige Algorithmen, sondern es stellt auch einen neuen Ansatz bei dem Einsatz der Software vor, so wird Matex als Modul angeboten und kann nach dem Baukastenprinzip mit anderen Modulen kombiniert und erweitert werden.

Damit ist es Entwicklern erstmals möglich in ihrer Software algebraische Berechnungen, also Berechnungen mit Variablen durchzuführen.
Die meiste Software musste bisher wegen fehlender Module vor Berechnungen die Variablen durch Zahlen ersetzten und das schränkte ihre Fähigkeiten ein.

Um eine nahtlose Integration zu ermöglichen, entwickelte ich Matex mit der Programmiersprache Java, diese bietet den Vorteil, dass sie auf jeder Plattform laufen kann, auf PCs, Macintoshs, Handys, PDAs und auf vielen anderen Plattformen.
Zusätzlich ist das Modul opensource, sodass jeder Entwickler an der weiteren Optimierung von Matex mitwirken und es gegebenenfalls anpassen kann.

Neben einigen Anleitungen zur Integration veröffentlichte ich auch eine Software auf der Matex Homepage, die mit Matex arbeitet, den Matex Rechner. Er demonstriert die einfache Integration des Moduls und die Fähigkeiten von Matex.
Nach der Ankündigung von Matex in einem Forum ließen sich viele Entwickler, die bisher vergeblich ein einem solchen Modul gesucht hatten, die Vorabversion per E-Mail schicken.

Martin Lovel, ein amerikanischer Programmierer bei der Softwareschmiede Keris, entwickelte auf Basis des Matex Moduls ein Tool, das Gleichungen grafisch darstellen kann und verdeutlichte damit die vielfältigen Einsatzmöglichkeiten von Matex.

Ausblick

Die häufige Nachfrage von Entwickler nach einem Modul für algebraische Berechnungen und viele positive Feedbacks nach der Veröffentlichung bestätigen die Meinung, dass Matex ein großes Potenzial besitzt.
So gibt es mit Matex erstmals ein Modul, mit dem jede Anwendung algebraische Berechnungen durchführen kann. Matex kann, dank der Modulform, in fast allen Bereichen, von der Mathematik, Chemie und Physik bis in den Bereich der statistischen Auswertungen für Berechnungen eingesetzt und mit spezifischen Funktionen für die Anwendergruppe ausgestattet werden.

Es bestand Bedarf für eine einfach zu bedienende und günstige beziehungsweise kostenlose Software für Privatanwender, Lehre und Forschung.
So kann Software mit Matex mit nur wenig Programmieraufwand entwickelt und auf Servern (zum Beispiel als Webservice), auf PCs oder auf Handys eingesetzt werden.

Weitere Informationen

Für aktuelle Berichte und Dokumentationen zu Matex empfehle ich einen Besuch der Matex Homepage, unter www.matexSoft.com.
Sie bietet für Interessenten, Anwender und Entwickler neben zahlreichen Informationen auch eine Downloadmöglichkeit für Matex Software.

Danksagungen

Ich danke meiner Familie für ihre Unterstützung. Meinen Vater, der mich bei Problemen zum Weitermachen motivierte und mir Dinge wie Potenzrechnen erklärte, die noch nicht in der Schule durchgenommen wurden und meiner Mutter, die wegen meiner Schüler experimentieren Arbeit fast ein Jahr lang darauf Rücksicht nahm, dass ich im Haushalt weniger helfen konnte.
Nicht zuletzt danke ich auch meinen beiden Großvätern, die mich auf Fehler in der schriftlichen Arbeit aufmerksam machten und für manch einen Satz eine bessere Formulierung fanden. Ein besonderer Dank geht auch Thomas Bayer, Geschäftsführer von Orientation in Objects, der mir während eines Praktikums den wichtigen Tipp gab, einen Rekursiven Abstieg einzusetzen. Ohne seinen Tipp wären fehlerfreie und effiziente Berechnungen deutlich schwieriger zu realisieren gewesen.

Literaturverzeichnis

- Stanchfield, Scott: ANTLR Tutorial, http://javadude.com/articles/antlrtut/antlr3.html
- Mahmoud, Qusay: MIDP Inter-Communication with CGI and Servlets, http://developers.sun.com/techtopics/mobility/midp/articles/servlets/
- Sun Microsystems, Inc: Java SDK, http://java.sun.com/j2se/1.4.2/download.html