

# MIN

## Multimediale Module für Mathematik in Informatik und Naturwissenschaften

Michael Grossmann

Wolfgang Kuchlin

Universität Tübingen  
Wilhelm-Schickard-Institut für Informatik  
Arbeitsbereich Symbolisches Rechnen  
Sand 14, D-72076 Tübingen

<http://www-sr.informatik.uni-tuebingen.de>

<http://min.informatik.uni-tuebingen.de>

**Abstract:** Im Rahmen des Projekts *MIN* wurden Web-basierte komplexe Lehr-/Lernmodule für eine neue drei- bis viersemestrige Grundvorlesung *Mathematik für Informatiker und Bioinformatiker* entwickelt. MIN besteht neben den Vorlesungen aus einem begleitendem Lehrbuch, aus einer Verfilmung der Vorlesung, sowie aus Java Applets, die mathematische Sachverhalte visualisieren und Anwendungen demonstrieren. Dieser Beitrag behandelt vor allem die Applets und das Software-Framework von MIN. Insbesondere werden die Probleme beim Zeichnen von Funktionen mit hohen Frequenzen und Singularitäten sowie das Zeichnen impliziter Funktionen besprochen.

## 1 Einleitung

In der Praxis stellt die Mathematik im Grundstudium Informatik eine erhebliche Hürde für den Studienerfolg dar. Zum Teil ist dies unumgänglich, zum Teil aber ist dies in der Informatik historisch bedingt und aus heutiger Sicht unnötig. So dürfen Informatiker neben einer spezifischen Stoffauswahl und relevanten Anwendungsbeispielen auch erwarten, dass Mathematik nicht nur mit Kreide und Wandtafel bzw. Papier und Bleistift stattfindet sondern der Computer als sinnvolles Hilfsmittel genutzt wird.

Unser übergeordnetes Ziel ist es daher, die Mathematikausbildung zunächst für Informatik und Bioinformatik zu reformieren und dadurch unnötige Schwierigkeiten im Studium zu beseitigen. Das Projekt ist so ausgelegt, dass danach weitere Module für Mathematikvorlesungen für Naturwissenschaftler, z.B. für Biologen oder Physiker, erstellt werden können. Zu diesem Zweck wurde ein hypermediales, im Web verfügbares<sup>1</sup> Lehr-/Lernsystem entwickelt, das den Stoff einer neuen drei- bis viersemestrigen Grundvorlesung *Mathematik für Informatik und Bioinformatik* abdeckt. Das System besteht aus drei Grundbausteinen:

---

<sup>1</sup><http://min.informatik.uni-tuebingen.de/>

1. Ein vorlesungsbegleitendes und auch online verfügbares Lehrbuch [WHK04].
2. Eine digitale Verfilmung der Vorlesung mit Recherchemöglichkeit.
3. Interaktive Visualisierungen und Werkzeuge in Form von Java Applets.

Wir sprechen hier von einem Lehr-/Lernsystem, da es nicht nur im häuslichen Selbststudium zum Vertiefen der Vorlesungsinhalte sondern auch im Hörsaal von Dozenten zur Illustration der Vorlesungen eingesetzt wird. Umgekehrt ist es wesentlich, dass alle Computer gestützten Illustrationen der Dozenten auch den Lernenden in identischer Form zur Verfügung stehen. Wenn z. B. eine Vorlesungsverfilmung die Benutzung eines Programms zeigt, dann muss dem Lernenden bei der Nacharbeit zur Vorlesung ungeachtet seines Aufenthaltsorts und der lokalen PC Software exakt dasselbe Programm zur Verfügung stehen.

## 2 Vorarbeiten

Bereits 1996 wurde eine Konzeption von W. Küchlin, M. Wolff und B. Amrhein für eine durch den Computer illustrierte Mathematik vom Stifterverband für die deutsche Wissenschaft als *modellhafte Initiative zur Studienreform* ausgezeichnet. Mit den Fördermitteln wurden zunächst auf Basis von Maple Illustrationen geschaffen und in der Vorlesung erprobt, die dann im Lehrbuch *Analysis Alive!* [WGR98] dokumentiert wurden. In der Folge wurde mit der Umstellung auf Java Applets in einer Client-/Server Umgebung begonnen, wobei der Übergang zum Teil fließend war. So wurde zum exakten Zeichnen von Funktionen mit Singularitäten noch auf Funktionalität von Maple zurückgegriffen [BCK00].

### 2.1 Probleme bei der Verwendung von Computeralgebrasystemen

Durch den Einsatz eines Computeralgebrasystems (CAS) ergaben sich u. a. Versionsprobleme, Lizenzprobleme und Korrektheitsprobleme, die in der Praxis störend waren. Daher entschloss man sich für die weitere Arbeit, möglichst viele Visualisierungen direkt mit Java im Web zu realisieren und übernahm lediglich die Konzepte aus *Analysis Alive*. In diesem Zusammenhang kann MIN auch demonstrieren, wie weit man hier ohne CAS kommen kann und was die Grenzen und die Aufwände sind.

Ein gravierendes Problem stellten Versionsinkompatibilitäten dar. So ließen sich die auf Maple Version 4 und Version 5 basierten Illustrationen aus *Analysis Alive* mit der Version 6 nicht mehr ausführen. Selbst wenn man die Illustrationen beständig anpasst, werden in der Praxis nie alle Lernenden dieselbe Version des CAS installiert haben. Will man dies vermeiden, muss man zu einer Client-/Server-Architektur mit einem CAS-Server übergehen. Man hat dann aber dann ein recht anspruchsvolles Client-/Server-System aufzusetzen und (auch finanziell) zu unterhalten. Dieses muss eine Vielzahl von gleichzeitigen Benutzern effizient bedienen können und es muss unter Beibehaltung von Stabilität und Effizienz ständig auf die neuesten Versionen des CAS Kerns aufgerüstet werden.

Erst seit jüngerer Zeit ist es überhaupt möglich, kommerzielle Computeralgebrasysteme durch den Erwerb einer Serverlizenz in einer Client-/Server-Architektur zu nutzen. Oft ist aber auch eine lokale Installation des CAS nötig, u. a. wenn die vorhandene Bandbreite keine effiziente Übertragung von Bildern vom Server erlaubt. In diesem Fall hat man neben den Versionsproblemen auch wieder Lizenzprobleme zu lösen.

Ein recht überraschendes Problem ist, dass auch kommerzielle CAS oft falsche Ergebnisse liefern. Beispielsweise entstehen beim Zeichnen der Funktion  $f(x) = 1/\sin(\exp(x^2))$  mit Maple V9 Artefakte durch Singularitäten, auch wenn die Option `discont=true` gesetzt wird. Als weiteres Beispiel ergibt die Eingabe von `iscont(1/sin(x^2), x=-1..1)` den Wert `true`, obwohl  $\sin(0^2) = 0$  und  $1/0$  nicht definiert ist. Während man bei kommerziellen Systemen solche Probleme nicht selbst beheben kann, hat man bei einer eigenen, grundständigen Implementierung diese Möglichkeit. Im Kontext der Lehre lag unser Fokus insbesondere auch auf dem Vermeiden von Fehlern, die Lernende unter Umständen nicht als solche erkennen können (vgl. Abschnitt 6.2.1).

## 2.2 Weitere Projekte

Mittlerweile gibt es eine große Anzahl weiterer Projekte, die interaktive Java Applets für die Illustration mathematischer Themen einsetzen. Davon basieren einige auf kommerziellen Technologien wie etwa *Automatic Calculus and Algebra*<sup>2</sup> auf *WebMathematica*<sup>3</sup>. Das Projekt *math-kit* [UBB<sup>+</sup>04] greift auf das Computeralgebrasystem MuPad<sup>4</sup> und auf das Open Source Programm GEONExT zurück.

## 3 Reorganisierte Vorlesung und begleitendes Lehrbuch

Für MIN wurde zunächst von M. Wolff, P. Hauck und W. Küchlin eine neue, gestraffte und reorganisierte Mathematikvorlesung für Informatik und Bioinformatik konzipiert. Diese ist in einem Lehrbuch [WHK04] dokumentiert, welches im Text auf die interaktiven Visualisierungen (vgl. Abschnitt 5) verweist. Dadurch wird der Leser zur Vertiefung der Inhalte durch eigenes Ausprobieren und Verifizieren der bisher gebildeten mentalen Modelle [Ede96] motiviert. Außerdem bekommen algorithmische Verfahren ein größeres Gewicht, da sie direkt zu Programmen führen, mit denen auch größere und realistischere Beispiele gerechnet werden können, die von Hand schlicht nicht zu bewältigen wären.

Neben einer gedruckten Version wird das Lehrbuch auf den Projektseiten in Form von HTML-Seiten zur Verfügung gestellt, wobei die entsprechenden Querverweise auf die Java Applets als Hyperlinks vorhanden sind.

---

<sup>2</sup><http://www.calc101.com/>

<sup>3</sup><http://www.wolfram.com/products/webmathematica/index.html>

<sup>4</sup><http://www.sciface.de>

## 4 Verfilmte Vorlesung

Der *Tübinger Internet Multimedia Server* (timms)<sup>5</sup>, der am Zentrum für Datenverarbeitung (ZDV) in Tübingen entwickelt und implementiert wurde, stellt inhaltlich recherchierbare Videoaufzeichnungen von Lehrveranstaltungen der Universität Tübingen im Internet bereit. Mit dieser Technik wurde die Vorlesung *Mathematik für Informatik und Bioinformatik* vom ZDV verfilmt, über einen recherchierbaren Schlagwortindex modularisiert und auf dem timms-Server abrufbar bereitgestellt.

Wir sehen die Verfilmung nicht als Ersatz, sondern hauptsächlich als Ergänzung zur klassischen Vorlesung. Versäumte Vorlesungstermine können nun leichter nachgeholt und bei der Prüfungsvorbereitung können gezielt einzelne Lücken geschlossen werden. Ein Thema kann im Buch nachgelesen, über die Verfilmung „nachgehört“ und über das zugehörige Applet illustriert oder eingeübt werden.

Gerade bei Informatikern und BioInformatikern, deren primäres Interesse nicht die Mathematik ist, bestehen auch große Unterschiede in der mathematischen Vorbildung und im mathematischen Können. In Verbindung mit den hohen Teilnehmerzahlen ist es dann schwierig, ein Lerntempo zu finden, das allen Studenten gerecht wird. Durch die Steuerungsinteraktion [IH02] der Videosoftware des Browsers lässt sich die Vorlesung in kleine, wiederholbare Einheiten diskretisieren und das Lerntempo individuell anpassen.

## 5 Interaktive Applikationen im Web

Den dritten Grundbaustein von MIN stellen die interaktiven Applikationen und Werkzeuge dar, die in den Kontext der Web-Seite des Projekts integriert sind (vgl. Abbildung 1). Dabei handelt es sich um Java Applets, die mathematische Sachverhalte illustrieren. Für die Benutzung ist ausschließlich ein Java fähiger Browser notwendig. Für einige dreidimensionale Programme wird zusätzlich ein Java3D Plugin benötigt.

Zu jedem Applet gibt es eine Anleitung, die in Form eines Hypertextes die grundlegende Bedienung erläutert. Den Applets können beim Programmstart Parameter übergeben werden. Dadurch lassen sich mit ein und demselben Programm verschiedene vorgefertigte Beispiele in eine Webseite integrieren. Auf dieser Grundlage ist auch die Erstellung weiterer Lernmodule in anderen naturwissenschaftlichen Kontexten möglich.

Die Applikationen unterscheiden sich von anderen im Web verfügbaren Illustrationen, welche zum Beispiel mit Flash erstellt wurden dadurch, dass nicht ausschließlich vorgefertigte Beispiele dargestellt werden können. Es besteht insbesondere die Möglichkeit, eigene Eingaben zu machen, die an jeweilige Fragestellungen angepasst sind. Für Lernende kann es zum Bilden oder Festigen mentaler Modelle hilfreich sein, dass eigene Überlegungen und Schlussfolgerungen verifiziert (oder auch verworfen) werden können. Vorgefertigte Beispiele lassen hierzu oft zu wenig Spielraum.

---

<sup>5</sup><http://www.timms.uni-tuebingen.de>



Abbildung 1: Die Startseite des Applets „Funktionen einer Veränderlichen mit Singularitäten“. Das Programm lässt sich auch mit verschiedenen vorgegebenen Beispielen starten.

## 5.1 Die Java Applets im Überblick

Derzeit sind 42 Java Applets zu den Bereichen Analysis, lineare Algebra, diskrete Mathematik und Logik vorhanden. Davon sind alleine 29 der Analysis zugeordnet, was auch damit zusammen hängt, dass sich diese besonders *anschaulich* darstellen lässt.

Bei den Applets zur Analysis und linearen Algebra hat man häufig die Möglichkeit, Objekte wie Funktionen, Punkte, Vektoren, math. Folgen und Reihen etc. in ein Koordinatensystem in der Ebene oder im dreidimensionalen Raum einzuzichnen. Über ein Kontextmenü besteht die Möglichkeit, einzelne Objekte zu löschen, zu verstecken oder Einstellungen darauf zu ändern. Diese Möglichkeiten waren in der Anfangsphase der Entwicklung nicht vorhanden und wurden von den Dozenten als besonders hilfreiche Neuerung für den Einsatz in den Vorlesung angesehen. Damit ist es z.B. leicht möglich, nur die jeweils relevanten Teile passend zu den Erklärungen des Dozenten einzublenden. Die Bedienung der Metafunktionalität ist für Applikationen mit Koordinatensystem identisch und trägt dadurch weiter zu Akzeptanz durch Dozenten und Studenten bei.

Einige Applets beinhalten Animationen, die über eine einheitliche Animationskonsole bedient werden. Diese ermöglicht einfaches Abspielen, Anhalten sowie Einzelschritte vorwärts und rückwärts.

Für die GUI und die Darstellung der zweidimensionalen Objekte wird Java AWT eingesetzt. Objekte im Raum werden mit Hilfe von Java3D dargestellt und lassen sich mit der Maus drehen, um sie von allen Seiten betrachten zu können. Ursprünglich wurde für die

3D Applets das Computeralgebrasystem Maple in einer Client-/Server Architektur verwendet, wobei einzelne zweidimensionale Bilder dreidimensionaler Objekte an den Client übertragen und dargestellt wurden. Dies stellte aber zum einen Lizenzprobleme dar, zum anderen kann man sich einen dreidimensionalen Körper besser vorstellen, wenn man ihn dreht und die Änderung der resultierenden perspektivischen Projektion beobachtet. Dies war bei den von Maple erhaltenen 2D Plots nicht möglich. Aus diesem Grund wurde die ursprüngliche Thin Client Architektur wieder verworfen.

## 5.2 Vorstellung einzelner Applets

Im Folgenden wird eine kleine Auswahl von Applets kurz vorgestellt. Mit dem Applet *Funktionen einer Veränderlichen* ist es möglich, frei wählbare Funktionen einer Veränderlichen in ein Textfeld einzugeben und sich deren Graphen anzeigen zu lassen. Es können mehrere Funktionen in ein Koordinatensystem gezeichnet werden und es besteht die Möglichkeit zur Eingabe weiterer Objekttypen wie Folgen, Reihen, Punkte oder Vektoren. Das Applet *Funktionen einer Veränderlichen (mit Tangente)* stellt eine Erweiterung dar, bei dem für jeden x-Wert die Tangente an die Funktion gezeichnet werden kann. Das Applet *Animierte Differentiation* visualisiert den Übergang vom Differenzenquotient zum Differential in Form einer Animation. Das Applet *Integration* illustriert den Begriff des Riemann-Integrals anhand von Treppenfunktionen und dient gleichzeitig als Werkzeug zur symbolischen und numerischen Berechnung von Integralen und Flächeninhalten. Das Applet *Funktionengraph im Raum* (vgl. Abbildung 2) ermöglicht die Anzeige von Graphen einer Funktion zweier Veränderlicher, impliziten Flächen, Parameterkurven, Parameterflächen, Ebenen, Geraden, Vektoren und Punkten in einem Koordinatensystem. Das Werkzeug *Matrizenrechner* dient zur numerischen Berechnung von Matrizenoperationen, wobei auch Zerlegungen wie LR, OR oder das Lösen von LGS möglich ist. Weitere Applets gibt es etwa zu den Themen Folgen und Reihen, Fouriertransformation, Interpolation, Differentialgleichungen, Konvergenz von Funktionenfolgen, komplexe Zahlen, komplexe Folgen, iterierte Integration im Raum, Eulergraph, Boolesche Funktionen als Polynomfunktion, Modulorechnung, fehlererkennende Codes, Auswertung logischer Ausdrücke, Konfigurationssysteme, Resolutionskalkül der Aussagenlogik und Robotik.

Die Applets lassen sich grob in die Bereiche Visualisierung von mathematischen Objekten, Visualisierung von Algorithmen und das Rechnen größerer Beispiele klassifizieren, wobei die bisherigen Applets hauptsächlich die Visualisierung von Objekten abdecken.

## 6 Ein Softwareframework für die Analysis und die lineare Algebra

Es wurde ein umfangreiches Softwareframework entwickelt, das hauptsächlich auf die Analysis und die lineare Algebra ausgerichtet ist. Dieses derzeit aus etwa 400 Klassen bestehende Framework folgt dem Model-View-Controller Entwurfsmuster der Softwaretechnik. Es ist in die folgenden Pakete untergliedert:

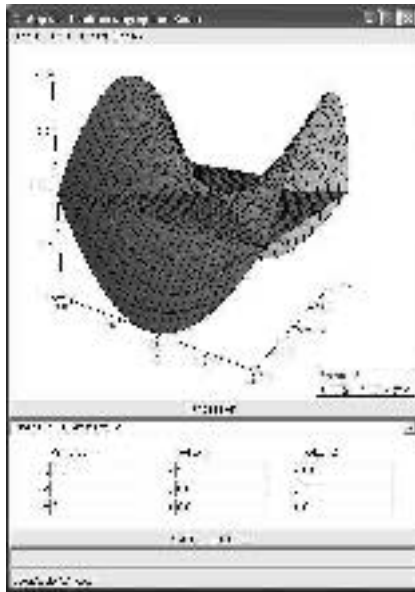


Abbildung 2: Das Applet: Funktionengraph im Raum.

- Das Paket **mathx** bildet mathematische Objekte wie Funktionen, Folgen, Reihen, Parameterkurven usw. ab und liefert die dazugehörigen Parser, um solche Objekte aus Zeichenketten, die in der Regel durch den Benutzer eingegeben werden, zu erzeugen. Das Paket umfasst ca. 100 Klassen und stellt die *Model* Komponente des Frameworks dar.
- Das Paket **cartSystem** dient zum Zeichnen von zwei-, bzw. dreidimensionalen mathematischen Objekten in ein Koordinatensystem und stellt die *View* Komponente des Frameworks dar. Es umfasst etwa 100 Klassen.
- Das Paket **caInWeb** stellt eine Schnittstelle dar, die über eine Client-/Server Architektur Zugriff auf ein Computeralgebrasystem ermöglicht. Es handelt sich um eine light weight Architektur, die derzeit 9 Klassen umfasst.
- Das Paket **applets** beinhaltet die eigentlichen Applets, wie in Abschnitt 5.2 beschrieben und umfasst derzeit etwa 120 Klassen.
- Das Paket **awtx** beinhaltet wiederverwendbare Komponenten für die grafische Benutzerschnittstelle (GUI) wie Layoutmanager oder generische Eingabepanel und Dialoge.

## 6.1 Das Paket mathx

Während das Zeichnen fest definierter Objekte wie etwa des Graphs der Funktion  $f(x) = x^2$  wenig Aufwand in der Modellbildung erfordert, ist dieser für beliebig definierte Objekte beträchtlich höher. Man benötigt zum einen eine abstrakte Darstellung für Funktionen und zum anderen Parser, die Zeichenketten in solche Objekte umwandeln. Es wurde daher eine Schnittstelle vom Typ `Function` definiert und Parser implementiert, die aus einer Zeichenkette ein Objekt dieses Typs erzeugen. Die Funktionen sind als Bäume implementiert, die sich beim Parsen direkt aus dem Syntaxbaum ergeben. Die Blätter eines solchen Baums stellen die identische Funktion oder die konstante Funktion dar, die inneren Knoten sind entweder unäre (z.B. für `cos`, `sin`, `exp`, `sqrt`) oder binäre Funktionen (z.B. für `+`, `-`, `*`, `/`). Alle Knoten eines solchen Baums implementieren die Schnittstelle `Function`. Dadurch lässt sich ein solcher Funktionenbaum einfach rekursiv für beliebige Argumente auswerten und mit anderen Bäumen zu neuen Funktionen verknüpfen.

Neben der Schnittstelle `Function` gibt es auch noch ein Konzept für Folgen und Reihen, welche Funktionen mit eingeschränktem Definitionsbereich darstellen, sowie ein Konzept für Mengen, um Definitions- und Wertebereich einer Funktion festlegen zu können. Um Teilfolgen definieren zu können, sind Folgen mit eingeschränktem Wertebereich notwendig. Zudem gibt es die Möglichkeit, Folgen und Reihen rekursiv zu definieren.

## 6.2 Das Paket cartSystem

Während im Paket `mathx` die Modellbildung stattfindet, stellt dieses Paket die View-Komponente dar. Abbildung 3 zeigt ein Klassendiagramm der Architektur.

Die Klasse `CartSystem` stellt ein Koordinatensystem dar. Dieses besitzt eine Liste von Objekten vom Typ `Drawable`. Ein `Drawable` hat die abstrakte Methode `render`, die dafür sorgt, dass sich das `Drawable` selbst in das Koordinatensystem einzeichnet. Weiter-

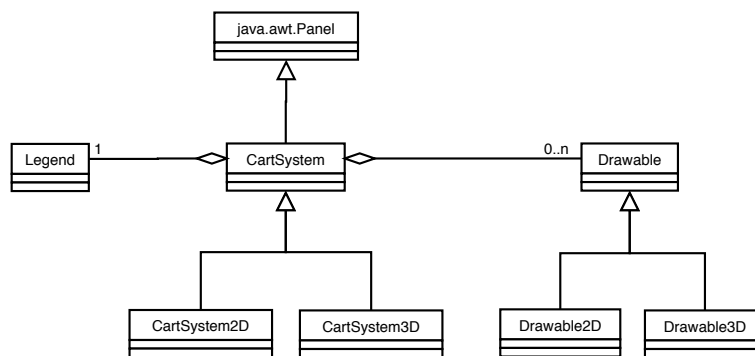


Abbildung 3: Architektur des Pakets `cartSystem`.

hin wird in einem `Drawable` die Farbe und ein `Visibilitätsstatus` festgelegt. Von `Drawable` wird die Klasse `Drawable2D` für zweidimensionale Objekte abgeleitet. Konkrete Implementierungen stellen etwa die Klassen `DrawablePoint`, `DrawableVector` oder `DrawableFunction` dar. Das Zeichnen findet über einen Java AWT `Graphics` Kontext statt. Um das Framework um neue Objekte zu erweitern, wird von `Drawable2D` abgeleitet, wobei die Methoden `render` und `getBoundingRect` implementiert werden müssen. Für dreidimensionale Objekte wird von `Drawable3D` abgeleitet. Hier findet das Rendering nicht direkt statt, sondern es wird ein Java3D `BranchGroup` Objekt zurück gegeben, wenn `render` aufgerufen wird.

### 6.2.1 Das exakte Zeichnen von Funktionen

Eine elementare und zugleich zentrale Hürde stellte das exakte Zeichnen von Funktionen einer Veränderlichen dar. Korrektes Zeichnen ist gerade im Kontext der Lehre wichtig, da Artefakte aufgrund von Aliasing oder Singularitäten von Studenten in den Grundvorlesungen nicht erkannt und falsch gedeutet werden.

Beim einfachen Samplingalgorithmus können Artefakte aufgrund von Aliasing und Singularitäten entstehen [Gro04]. Um Aliasing zu vermeiden, wurde das Verfahren von Fateman [Fat92] zum Zeichnen mit Intervallarithmetik erweitert. Dabei ist ein adaptiver Algorithmus entstanden, der mittels einer Heuristik beweist, dass Aliasing vorkommt<sup>6</sup> und in diesem Fall mit Intervallarithmetik zeichnet. Artefakte, die beim Zeichnen mit Intervallarithmetik eventuell auftreten und in [ABK95] beschrieben werden, können dadurch nur an den Stellen vorkommen, an denen auch Sampling zu Artefakten führen würde.

Das Problem beim Zeichnen von Funktionen mit Singularitäten wird in [BCK00] beschrieben und dort mit Hilfe einer Client-/Server-Architektur gelöst, welche Maple auf dem Server verwendet, um Singularitäten zu berechnen, die vom Java-Client beim Zeichnen berücksichtigt werden. Dieses Verfahren basiert hauptsächlich auf Methoden des symbolischen Rechnens und ist daher auf eine Teilmenge möglicher Funktionen beschränkt. Da man beim Zeichnen von Funktionen auf dem Bildschirm jedoch nur eine (numerisch) beschränkte Auflösung zur Verfügung hat und deshalb nur so genau rechnen muss, wie man auch zeichnen kann, wurde zur Lösung des Problems nun ein symbolisch-numerischer Algorithmus entwickelt [Gro04]. Der Algorithmus verwendet das von Roach [Roa94] beschriebene Verfahren zur Nullstellenbestimmung. Dieses setzt voraus, dass die zu untersuchende Funktion stetig ist und verwendet Intervallarithmetik und die symbolische Ableitung, die im Paket `mathx` implementiert sind. Mit diesem Verfahren wird beispielsweise der Nenner einer Division auf einem Intervall auf die Existenz von Nullstellen untersucht. Falls solche existieren, ist die Division auf diesem Intervall nicht stetig, was im Graph entsprechend angezeigt wird. Im anderen Fall kann der untersuchte Term mit dem Verfahren von Roach auf Nullstellen untersucht werden, falls dieser wiederum Teil eines weiteren Nenners ist. Abbildung 4 zeigt den Graph der Funktion  $f(x) = 1/\sin(1/x^2)$ , der mit diesem Verfahren erstellt wurde und vergleicht das Ergebnis mit Maple.

---

<sup>6</sup>Dieses Verfahren beweist nicht, dass kein Aliasing vorkommt. Falls Aliasing nicht erkannt wird, verwendet der Algorithmus jedoch Standard-Sampling und ist somit nicht schlechter als die Algorithmen kommerzieller Computeralgebrasysteme.

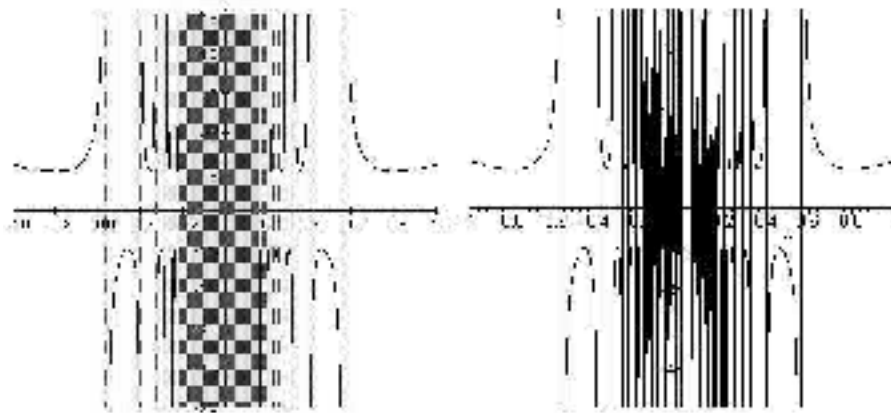


Abbildung 4: Der Graph der Funktion  $f(x) = 1/\sin(1/x^2)$ . Links werden mit dem symbolisch-numerischen Verfahren Singularitäten durch eine rote, gepunktete, senkrechte Linie dargestellt. Ein rot gepunkteter Balken zeigt einen Bereich an, in dem Singularitäten so gehäuft vorkommen, dass ein vernünftiges Zeichnen unmöglich ist. Der rechte Graph wurde mit Maple erstellt; durch das Zeichnen über Singularitäten entstehen Punkte, die nicht zum Graph von  $f$  gehören.

### 6.2.2 Das Zeichnen von impliziten Kurven und Flächen

Implizite Kurven wurden mit Hilfe des in [Sny92] beschriebenen Verfahrens gezeichnet, welches Intervallarithmetik verwendet, um die quadratische Laufzeit beim Sampling zu vermeiden. Dabei wurde bei Erreichen einer vorgegebenen Auflösung auf Sampling umgeschaltet, womit keine Artefakte durch die in [MSV<sup>+</sup>02] beschriebene Überabschätzung entstehen können. Für implizite Flächen wurde das in der Computergrafik übliche Marching Cubes [LC87] Verfahren eingesetzt. Das Grundverfahren ist bei der Klassifizierung von Teilflächen nicht eindeutig. Zur Auflösung der Mehrdeutigkeiten wurde das von [NH91] vorgeschlagene Asymptotic Decider Verfahren eingesetzt, welches eine Multi Entry Cubical Table verwendet. Zudem wurde das Gitter für die Darstellung separat so berechnet, dass nur Kanten auf den Würfelseiten angezeigt werden [Gro04]. Dadurch erhält man Höhenlinien anstatt des durch den Marching Cubes Algorithmus bedingten unregelmäßigen Gitters.

### 6.3 Das Paket caInWeb

Obwohl auf Computeralgebrasysteme möglichst verzichtet werden sollte, ist dennoch an einigen Stellen eine Anbindung nützlich. So berechnet zum Beispiel das Applet *Integration* den Flächeninhalt zusätzlich symbolisch mit Hilfe der Stammfunktion, falls dies möglich ist. Ursprünglich basierte die Anbindung an das Computeralgebrasystem auf Java RMI. Dies war aber durch die an der Fakultät für Informatik sich zunehmend verschärfte Firewallpolitik problematisch, da zusätzlich freigeschaltete Ports nicht ohne weiteres mög-

lich waren. Daher wurde nach einer einfachen Lösung gesucht, die sich auf den Port 80 (HTTP) beschränkt, da dieser in der Regel immer zur Verfügung steht. Dabei ist eine Architektur entstanden, bei der sowohl das Netzwerkprotokoll als auch das Computeralgebrasystem beliebig austauschbar sind [Gro04], um auch eventuellen zukünftigen Anforderungen gewachsen zu sein. Als Netzwerkprotokoll kommt dabei HTTP mit Hilfe von HTTP Servlets zum Einsatz.

#### **6.4 Verwendbarkeit des Frameworks**

Gegenüber anderen Systemen besteht der Vorteil, dass das Framework vollständig in Java implementiert ist und der Quellcode je nach Anforderung erweitert werden kann. Das Framework lässt sich auf drei Ebenen nutzen:

1. Auf Hypertext-Ebene zur Erstellung neuer Module bzw. Lehr-/Lerneinheiten unter Verwendung vorhandener Applets. Diese können über das Applet Tag von HTML parametrisiert werden, um eigene Beispiele umzusetzen.
2. Auf High Level Java Ebene zur Erstellung eigener Applets auf Grundlage des bisherigen Frameworks, etwa durch Verwendung eines Koordinatensystems in Verbindung mit vorhandenen Drawables.
3. Auf tieferer Java Ebene für die Erweiterung des Frameworks, etwa durch Erweiterung des Parsers oder das Erstellen neuer Drawable Objekte.

### **7 Weitere Arbeit**

Während die bisherigen Applets vornehmlich die Visualisierung von Objekten abdecken, wurden die Visualisierung von Algorithmen und das Rechnen größerer Beispiele nur in Ansätzen bearbeitet. Zur Visualisierung von Algorithmen gibt es beispielsweise ein Applet zu Eulergraphen und zum Davis-Putnam Algorithmus zur Erfüllbarkeitsprüfung in booleschen Algebren. Das Rechnen größerer Beispiele wird in den Applets zur Modulorechnung und zu fehlererkennenden Codes ermöglicht. Der Fokus für die weitere Arbeit wird sich nun verstärkt auch auf diese Bereiche konzentrieren.

### **Danksagung**

MIN wurde 2003–2005 vom Ministerium für Wissenschaft, Forschung und Kunst des Landes Baden-Württemberg aus dem Fonds zur Stärkung der Lehre in der Programmlinie *Modularisierung* gefördert. Wesentliche Beiträge stammen von den Projektpartnern Prof. M. Wolff und Prof. D. Kaletta sowie von Prof. P. Hauck und Dr. H. Abele. Weiterer Dank gilt allen Mitarbeitern und Studierenden, deren Arbeiten in MIN eingeflossen sind.

## Literatur

- [ABK95] R. Avitzur, O. Bachmann und N. Kajler. From honest to intelligent plotting. In *ISSAC'95: Proc. 1995 Intl. Symp. on Symbolic and Algebraic Computation*, Seiten 32–41. ACM Press, 1995.
- [BCK00] D. Bühler, C. Chauvin und W. Kuchlin. Plotting Functions and Singularities with Maple and Java on a Component-based Web Architecture. In V. G. Ganzha, E. W. Mayr und E. V. Vorozhtsov, Hrsg., *Computer Algebra in Scientific Computing – CASC 2000*, Samarkand, Uzbekistan, Oktober 2000. Springer.
- [Ede96] W. Edelmann. *Lernpsychologie*. Beltz Psychologie-Verlags-Union, 1996.
- [Fat92] R. Fateman. Honest plotting, global extrema, and interval arithmetic. In *ISSAC'92: Intl. Symp. on Symbolic and Algebraic Computation*, Seiten 216–223. ACM Press, 1992.
- [Gro04] M. Grossmann. Ein Framework zur Erstellung von Applikationen für die computer-gestützte Mathematikausbildung. Diplomarbeit, Universität Tübingen, 2004.
- [IH02] J. Issing und P. Klimsa (Hrsg.). *Information und Lernen mit Multimedia und Internet*. Beltz, 2002.
- [LC87] W. E. Lorensen und H. E. Cline. Marching Cubes: A High-Resolution 3D Surface Construction Algorithm. In *SIGGRAPH'87 Conf. Proc.*, Jgg. 21 of *Computer Graphics*, Seiten 163–169, July 1987.
- [MSV<sup>+</sup>02] R. Martin, H. Shou, I. Voiculescu, A. Bowyer und G. Wang. Comparison of interval methods for plotting algebraic curves. *Comp. Aided Geom. Des.*, 19(7):553–587, 2002.
- [NH91] G. M. Nielson und B. Hamann. The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes. In *Proc. of IEEE Visualization '91*, Seiten 83–91, October 1991.
- [Roa94] K. Roach. Symbolic-numeric nonlinear equation solving. In *ISSAC'94: Proc. 1994 Intl. Symp. on Symbolic and Algebraic Computation*, Seiten 278–284. ACM Press, 1994.
- [Sny92] John M. Snyder. Interval analysis for computer graphics. *Computer Graphics*, 26(2):121–130, 1992.
- [UBB<sup>+</sup>04] L. Unger, M. J. Bauch, A. Baudry, M. Bungenstock, B. Mertsching, G. Oevel, K. Padberg und B. Thiere. math-kit - Ein multimedialer Baukasten für die Mathematikausbildung im Grundstudium. *SoftwareTechnikTrends*, 24(1), February 2004.
- [WGR98] M. Wolff, O. Gloor und C. Richard. *Analysis Alive*. Birkhäuser Verlag, 1998.
- [WHK04] M. Wolff, P. Hauck und W. Kuchlin. *Mathematik für Informatik und BioInformatik*. Springer, 2004.