Konzeptionelle Überlegungen zur Einbeziehung informatischer Inhalte und Methoden beim Computereinsatz im Mathematikunterricht der Sekundarstufe 2

D i s s e r t a t i o n

zur Erlangung des akademischen Grades
doctor rerum naturalium (Dr. rer. nat.)
im Fach Mathematik

eingereicht an der

Mathematisch-Naturwissenschaftlichen Fakultät II
der Humboldt-Universität zu Berlin

von Eberhard Lehmann
geb. 3.3.1936 in Berlin

Präsident / Präsidentin der Humboldt-Universität zu Berlin

Prof. Dr. J. Mlynek .....

Dekan / Dekanin der Mathematisch-Naturwissenschaftlichen Fakultät II

Prof. Dr. E. Kulke

Gutachter / Gutachterin

1. PD Dr. Ingmar Lehmann	Humboldt-Universität Berlin, Institut für Mathematik
2. Prof. Dr. Wilfried Herget	Martin-Luther-Universität Halle-Wittenberg, Fachbereich Mathematik und Informatik
3. Prof. Dr. Wolfram Koepf	Universität Gesamthochschule Kassel, Fachbereich Mathematik und Informatik

Tag der mündlichen Prüfung: 6. Juni 2003

Vorwort

Die Schulmathematik steht zur Zeit vor zahlreichen interessanten Herausforderungen. Diese sind u. a. erwachsen

- aus den Ergebnissen der weltweiten Studien TIMSS und PISA, die dem Mathematikunterricht an den deutschen Schulen schlechte Noten beschert haben und
- aus den vielfältigen Möglichkeiten, die die neuen Medien für den Unterricht, speziell auch für den Mathematikunterricht bringen.

Zu den Herausforderungen gehören u. a. die Vermittlung

- einer neuen Unterrichtskultur mit offenen Arbeitsformen,
- einer veränderten Aufgabenkultur mit offenen Aufgabenstellungen,
- neuer Kompetenzen für Lehrer und Schüler.

In fast allen Bereichen der Praxis hat die Datenverarbeitung – und damit auch die Informatik – ihre überragende Bedeutung längst nachgewiesen. Für die Schule hängt manch einer trotz vieler Misserfolge weiter dem Gedanken nach, Inhalte der Informatik in die Schulfächer zu integrieren und den Informatikunterricht damit möglicherweise zu verwässern oder gar wieder abzuschaffen. Dabei wird jedoch häufig der Unterschied zwischen dem fachbezogenen Computereinsatz und dem, was Schulinformatik ausmacht, übersehen. Andererseits kann Schulinformatik nicht ohne Mathematik auskommen, denn viele Inhalte der Informatik beruhen auf mathematischen Grundlagen.

Mathematik lebt u. a. von den zu rechnerischen oder grafischen Problemlösungen notwendigen Algorithmen, die mittels entsprechender Programmiersprachen (Bezug zur Informatik!) als mächtige mathematische Werkzeuge eingesetzt werden können – im Gegenzug benötigt Informatik immer wieder Erkenntnisse der Mathematik zur Entwicklung von Algorithmen für die Bearbeitung ihrer Probleme. So sollte man meinen, dass sich auch der Schulunterricht in Mathematik und in Informatik gewisser Inhalte und Methoden des jeweils anderen Faches bedient. Die Schulpraxis zeigt, dass das leider nur in Ausnahmefällen geschieht. Selbst Lehrer, die beide Fächer vertreten, sind häufig nicht in der Lage die Vernetzungen zwischen beiden Fächern auszunutzen. So setzen z. B. etliche Informatiklehrer den Computer eigenartigerweise nicht in ihrem Mathematikunterricht ein.

Diese Arbeit beschäftigt sich insbesondere mit der Frage,

- wie weit Mathematikunterricht Aspekte der Informatik (gewisse Inhalte und Methoden) aufnehmen kann. Sie setzt sich zum Ziel, unterrichtliche Vernetzungsmöglichkeiten zwischen den beiden Fächern aufzuzeigen und für den Mathematikunterricht nutzbar zu machen.

Diese Zielsetzung ist

- eng verknüpft mit den vielen Möglichkeiten des Computereinsatzes im Mathematikunterricht.

Zur detaillierteren Untersuchung werden solche Aspekte näher betrachtet, die für beide Fächer von Bedeutung sind.

- Dabei wird es besonders darum gehen, die Folgerungen für einen Mathematikunterricht unter Einbeziehung informatischer Methoden und Inhalte auszuloten.
- Die theoretischen bzw. allgemeingültigen Überlegungen münden in Vorschläge für konkreten Mathematikunterricht.

Eine Abiturklausur aus dem Jahr 1954 dient zum Einstieg in die Thematik. Hier werden erste Hinweise gegeben, was der Computer mit verschiedenen Softwaresystemen für die Lösung von Mathematikaufgaben leisten kann. Kapitel 1 erläutert dann unter Berücksichtigung älterer Erfahrungen Grundlagen, die für die Entwicklung von Vernetzungskonzepten wichtig sind. Die Voraussetzungen für einen modernen Mathematikunterricht, der auch auf offene Unterrichtsformen, eine neue Aufgabenkultur und auf Computereinsatz setzt, werden auf Grund der besseren Verfügbarkeit von Rechnern und durch Lehrerfortbildungsmaßnahmen zunehmend günstiger. Kapitel 1.5 zeigt diverse Szenarien für die Unterrichtsgestaltung, wobei die für die Arbeit besonders relevanten Bereiche noch einmal deutlich markiert werden.
Kapitel 2 bringt eine ausführliche und begründende Darstellung informatischer Methoden und Inhalte, die für eine Einbeziehung in den Mathematikunterricht als besonders geeignet erscheinen. Dabei werden die theoretischen Ausführungen immer wieder mit Beispielen aus der Unterrichtspraxis angereichert.
Die in Kapitel 3 angebotenen Beispiele sind in der Regel aus meinen langjährigen Tätigkeiten als Mathematik- und Informatiklehrer und Seminarleiter in diesen Fächern entstanden und in der Regel in ihren Teilen unterrichtserprobt. In einigen Fällen neu ist die hier besonders betonte direkte Vernetzung von Mathematik- und Informatikinhalten, z. B. bei der Unterrichtseinheit „Zustandsgraphen" (Kapitel 3.3). Die Erfahrungen aus beiden Fächern zeigen, dass sich diese Unterrichtsangebote auch tatsächlich realisieren lassen. Sie sind auch nicht auf ein Bundesland (Berlin) ausgerichtet, sondern bei entsprechenden Voraussetzungen überall verwendbar. Über die Ziele einer Dissertation hinaus habe ich den Anspruch, Lehrern in einer nachfolgenden Veröffentlichung der Arbeit konkrete Hinweise für ihren Unterricht gegeben werden. Aus dieser Zielsetzung erwächst auch der Wunsch, gewisse Kernaussagen besonders hervorzuheben, etwa durch Texte in Quadern oder Textsymbolen, wie sie das Textverarbeitungsprogramm zur Verfügung stellt. Diese Kennzeichnungen dienen damit auch einer zusätzlichen Strukturierung der Zusammenhänge.
Kapitel 3 soll also die vorhergehenden mehr theoretischen Überlegungen durch vielseitige Unterrichtsbeispiele, häufig in Form konkreter Unterrichtseinheiten, verdeutlichen. Für diese Zielsetzung gibt es verschiedenartige Angebote. Es handelt sich um

- kurze Unterrichtseinheiten, die sich an verschiedenen Stellen des Mathematikunterrichts einschieben lassen (Kapitel 3.1, 3.2, 3.4),
- umfangreichere Themen, die einen größeren Zeitbedarf haben (Kapitel 3.3) und
- die Skizzierung der Inhalte eines Lineare Algebra/Analytische Geometrie - Kurses unter Einbeziehung von Informatikinhalten von der Dauer eines Halbjahres (Kapitel 3.5).

Für die Vorschläge werden auch mögliche Einordnungen in geltende Unterrichtsinhalte angegeben.
Kapitel 4 rundet die Betrachtung durch zusammenfassende Darstellungen und Bemerkungen ab.
Zu den in der Arbeit angesprochenen Themen gibt es umfangreiche Literatur. Diese bezieht sich allerdings meistens auf jeweils eines der Fächer Mathematik und Informatik. Es wird auffallen, dass häufiger eigene Werke von mir aufgeführt werden. Dieses Vorgehen wird verständlich, wenn man bedenkt, dass ich seit circa dreißig Jahren über Themen zum Mathematik- bzw. Informatikunterricht in Buchform oder in Fachzeitschriften veröffentliche. In der Dissertation werden jedoch nur Werke genannt, die für das Arbeitsthema auch wirklich relevant sind.

Berlin, November 2002 Eberhard Lehmann

Konzeptionelle Überlegungen zur Einbeziehung informatischer Inhalte und Methoden beim Computereinsatz im Mathematikunterricht der Sekundarstufe 2

Inhaltsverzeichnis

1. Grundlagen für die Entwicklung von Konzepten zur Einbeziehung informatischer Aspekte und zum Computereinsatz im Mathematikunterricht

1.1 Eine Abiturklausur aus dem Jahr 1954

1954 konnte mein Mathematiklehrer seinen Vortrag zur analytischen Geometrie noch unbehelligt von den heutigen Sorgen des Mathematikunterrichts darbieten (Abb.1.1-a). TIMSS und PISA (siehe S. 180) mit ihren schlechten Ergebnissen für den deutschen Mathematikunterricht und ihren möglichen Folgerungen waren noch nicht im Gespräch. Neue Aufgabenkultur, offene Unterrichtsformen (in heutigem Sinne) und Einflüsse von Computeralgebrasystemen oder Internet lagen noch in weiter Ferne.

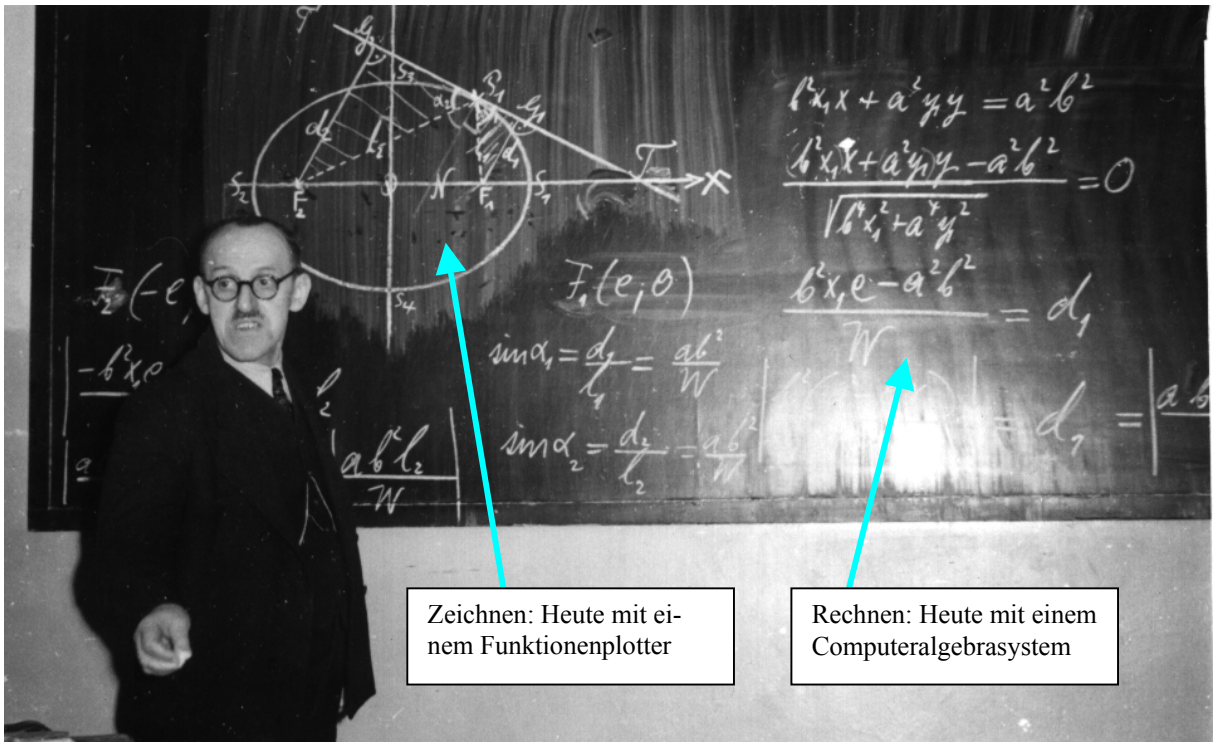


Abbildung 1.1-a: Mathematik-Unterricht 1954

Die seinerzeit gestellten Abituraufgaben vom 6.12.1954 (Abb. 1.1-b) würden in dieser Form zur Zeit von keiner Genehmigungsstelle ihre Zustimmung erhalten.

- Die zu erbringenden Teilleistungen werden dem Schüler nicht deutlich,
- die Anforderungen sind also zu wenig detailliert,
- zu wenig Analysis,

wären vermutlich einige der Einwände, vielleicht würde es auch heißen: „Zu schwer"!

Heute stellt sich zusätzlich die Frage, wie weit hier Rechnereinsatz möglich ist, denn schließlich sind hier diverse komplexe Rechnungen und Zeichnungen durchzuführen, für die ein

1.2 Grundlegende konzeptionelle Überlegungen

1.2.1 Zusammenhänge zwischen Mathematik- und Informatikunterricht

Die Darstellung in diesem Teilkapitel dient einer ersten Einführung in die bei der Verknüpfung mathematischer und informatischer Inhalte und Methoden entstehenden Probleme. Sie ist gleichzeitig Grundlage und Begründung für die ausführlichen Darstellungen der einzelnen Aspekte in späteren Kapiteln.

Die Einbeziehung informatischer Aspekte in den Mathematikunterricht ist auf verschiedenen Ebenen und mit unterschiedlichen Strategien möglich. Wir betrachten hierzu
Abbildung 1.2.1-a.

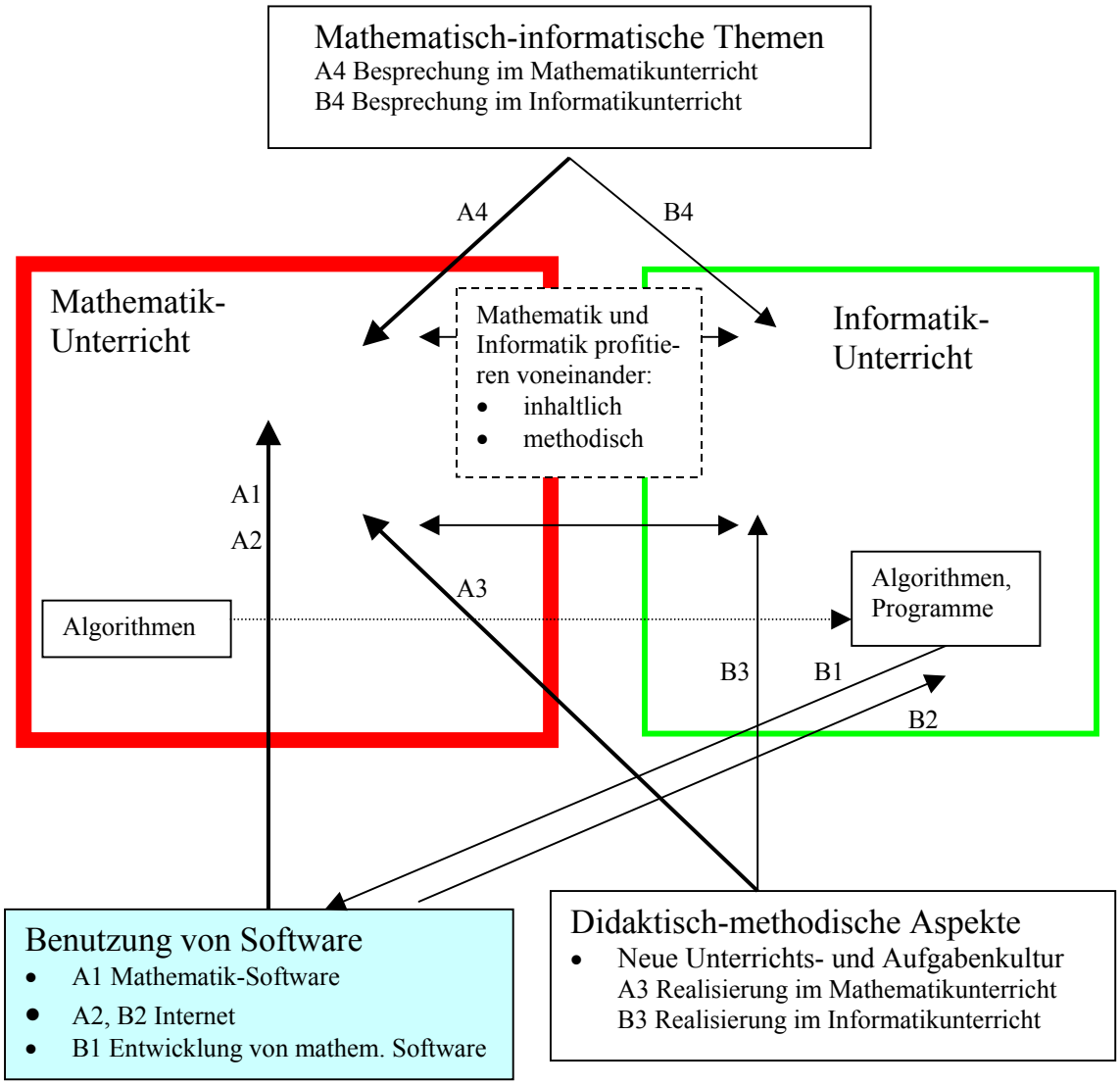


Abb. 1.2.1-a: Mathematik und Informatik – einige Zusammenhänge

Hinweis: Diese Abbildung wird später verfeinert, siehe Kapitel 4.

Der Mathematikunterricht kann auf mehreren Wegen von informatischen Ansätzen profitieren:

A1
Mathematikunterricht benutzt mathematische Software. Die Softwareprodukte sind Konstruktionen der Informatik, die durch Programmierung der (mathematischen) Algorithmen und geeigneter Oberflächenstrukturen entstehen. Hierzu gehören auch die im Internet angebotenen Demonstrationsprogramme und Lernumgebungen.

A2
Mathematikunterricht benutzt das Internet zur Recherche über mathematische Inhalte. Das Angebot hierfür ist inzwischen fast unübersehbar.

A3
Mathematik- und Informatikunterricht bedienen sich neben ihrer eigenen Didaktik/Methodik auch **geeigneter didaktisch-methodischer Ansätze des jeweils anderen Faches und weiterer neuerer methodischer Ansätze.**

A4
Mathematikunterricht behandelt Themen, in denen neben der Mathematik **informatische Inhalte von wesentlicher Bedeutung sind.** Allerdings können solche Themen auch von der Informatik für sich okkupiert werden – ein Aspekt, der bei Lehrplangestaltungen kaum beachtet wird. Ein Beispiel liefert die Kryptologie, die z. B. in Berlin an verschiedenen Stellen von Mathematik- und Informatiklehrplänen zu finden ist.

In den folgenden Ausführungen wird es darum gehen, die Aspekte A1 bis A4 näher zu untersuchen, um zu Vorschlägen für eine Berücksichtigung informatischer Inhalte und Methoden im Mathematikunterricht zu gelangen und exemplarisch einige Unterrichtssequenzen anbieten zu können.
Umgekehrt profitiert natürlich auch der Informatikunterricht von den mathematischen Inhalten und den Methoden (Pfeile in Richtung Informatik, Abb. 1.2.1-a). Die Auswirkungen eines modernen Mathematikunterrichts auf den Informatikunterricht sind inzwischen erheblich. Sie werden zum Beispiel bei dem Modulkonzept deutlich, siehe Kapitel 2.1.5. Der Aspekt Mathematik → Informatik wird jedoch in dieser Arbeit nicht näher untersucht.

1.2.2 Überblick über eine mögliche Platzierung mathematisch-informatischer Themen im Mathematikunterricht

Grundsätzlich können mathematisch-informatische Aspekte in beiden Sekundarstufen (und auch schon früher) berücksichtigt werden. So gibt es schon jetzt einige diesbezügliche Lehrplanangebote in der Sekundarstufe 1. Für die Berliner Schule seien genannt:

Mathematik mit Computern in der Sekundarstufe 1
Unterrichtseinheiten im Wahlpflichtfach Mathematik (Klasse 9, 10)

- Modellbildung
- Kryptologie
- andere Unterrichtseinheiten, in denen der Einsatz des Computers sinnvoll ist, beispielsweise in den Einheiten Matrizenrechnung oder Statistik.

Im Mathematik-Pflichtunterricht der Klassen 7 bis 10 fehlen konkrete Hinweise auf den Computereinsatz. Man findet lediglich allgemeine Absichtserklärungen. Dennoch findet inzwischen die Arbeit mit Computern auch bei Lehrern der Sekundarstufe 1 Akzeptanz**.** Hierzu gehören zum Beispiel die dynamischen Geometriesysteme EUKLID und GEONEXT. Richtungsweisende Arbeit wird in dem anschließend beschriebenen CAS-Projekt geleistet.

Das Berliner CAS-Projekt Sekundarstufe 1

Hier arbeiten alle neunten Klassen von vier Schulen und alle achten Klassen einer weiteren Schule mit dem Computeralgebrasystem des Taschencomputers TI-92-Plus, der allen Schülern ständig zur Verfügung steht. Näheres hierüber findet man im Endbericht des Projekts:

Lehmann, E.: „Berliner CAS-Projekt - Sekundarstufe 1 im Rahmen des BLK-Sinus-Projekts" (Senatsverwaltung für Bildung, Jugend und Sport, Berlin September 2002).

Dieses Projekt wird in Klasse 10 fortgesetzt und an zwei der Schulen in den neuen Klassen 9 durchgeführt.

Mathematik mit Computern in der Sekundarstufe 2 (Klasse 11–13)

In dem zur Zeit gültigen Lehrplan gibt es zwar Hinweise für den Computereinsatz, konkrete Beschreibungen für einzelne Unterrichtseinheiten fehlen allerdings. Gedacht ist in erster Linie an den Einsatz mathematischer Unterrichtssoftware. Der Bezug zur Informatik wird völlig vernachlässigt. An einigen Schulen wird der Computer bis hin zu Mathematik-Abituraufgaben verwendet. Die hierbei benutzte Software ist im Wesentlichen das Computeralgebrasystem DERIVE. Vereinzelt wird in der Sekundarstufe 2 auch mit Funktionenplottern, Tabellenkalkulation, dynamischen Geometriesystemen oder speziellen Programmen wie ANALYGEO oder POVRAY (für Analytische Geometrie und Computergrafik) gearbeitet. Seit Mitte 2002 beschäftigt sich der „Berliner CAS-Arbeitskreis" mit besonderen Problemen des Computereinsatzes. Außerdem werden sich ab April 2003 sechs Schulen an einem neuen CAS-Projekt (Sekundarstufe 2) mit dem Taschencomputer VOYAGE 200 (Texas Instruments) für Grundkurse beteiligen.

Mathematische Zusatzkurse

Eine weitere Möglichkeit der Verknüpfung von Mathematik und Informatik im Rahmen des Berliner Mathematik-Lehrplans besteht in den im Lehrplan formulierten Erweiterungsgrundkursen oder in weiteren Zusatzkursen, die sich Lehrer vom Landesschulamt genehmigen lassen können. Diese Kurse sind für gute Grundkursschüler und für Leistungskursschüler gedacht und sollen auf Leistungskursniveau stattfinden. Im Lehrplan (*) werden u. a. folgende Themen genannt:

- Inzidenzgeometrie
- Nichteuklidische Geometrie
- Logik
- Zahlentheorie
- Numerische Mathematik
- Differentialgleichungen
- Unendliche Reihen
- Markowketten

(*) Der Senator für Schulwesen, Berufsausbildung und Sport: Rahmenpläne Berlin, Fach Mathematik (Gymnasium), Juni 1990

Diese Kurse sind deshalb besonders geeignet, weil sie dem Lehrer wegen der Freiräume gute Gestaltungsmöglichkeiten geben. Gewisse Freiräume finden sich auch im Profilkurs Mathematik, Klasse 11. Diese lassen sich von einem kundigen Lehrer auch für informatische Ansätze nutzen. Bei allen genannten Unterrichtsangeboten (aber auch im normalen Mathematikunterricht) könnten Vernetzungsmöglichkeiten zur Informatik genutzt werden.

Insgesamt kann man zur Zeit von folgender Situation ausgehen:

> Mit zunehmender mathematischer Erfahrung der Schüler und durch Teilnahme etlicher Schüler am Informatikunterricht, aber auch durch die nun schon häufige private Benutzung von Computern vergrößern sich die Möglichkeiten einer Verknüpfung von Mathematik und Informatik. Für Untersuchungen und Vorschläge bzgl. der Einbeziehung informatischer Inhalte in den Mathematikunterricht liegen daher besonders für die Sekundarstufe 2 günstige Voraussetzungen vor.

Unterricht mit informatischen Aspekten in den Standardgebieten der Sekundarstufe 2

Der für den Mathematikunterricht zur Zeit wohl wirkungsvollste und weitestgehende Ansatz für eine Verknüpfung mathematischer Inhalte und Methoden mit informatischen kann für die drei Standardgebiete Analysis, Lineare Algebra / Analytische Geometrie und Stochastik formuliert werden. Hier lassen sich diverse Themen in den Lehrplänen finden, die dafür geeignet erscheinen, wobei auch gebietsübergreifende Themen gefunden werden können. Die folgende Tabelle nennt Beispiele, die sich sicher durch weitere interessante Themen ergänzen lassen.

Ausgewählte informatische Inhalte und ihre Verwendung in der Mathematik

Mathematik-Lehrplanthema	Beispiele für geeignete Software	einige passende informatische Inhalte
Analysis • Hüllkurven • Ortskurven • Flächen-berechnungen	CAS, Funktionenplotter, *ANIMATO*	Elemente der Computergrafik Algorithmen, programmiersprachliche Aspekte
Lineare Algebra, Analytische Geometrie • Magische Quadrate • Abbildungsgeometrie	CAS, *ANALYGEO, POVRAY*	Matrizen als Datenspeicher, Algorithmen auf zweidimensionalen Feldern, komplexe Datenstrukturen Elemente der Computergrafik
Stochastik • Binomialverteilung • Geometrische Verteilung • Markow-Ketten	CAS spezielle Programme	Baumstrukturen Zustandsgraphen, Automaten

Besondere Möglichkeiten finden sich bei Beachtung gebietsübergreifender Aspekte, u. a. aus Analysis, Lineare Algebra/Analytische Geometrie, Stochastik und endlicher Mathematik. Diese sind allerdings noch keine Lehrplanthemen und erfordern vom Lehrer in besonderem Maße eigene Gestaltungsfähigkeiten.

Gebietsübergreifende Themen • Zustandsgraphen	CAS, Funktionenplotter, kleine spezielle Programme	Graphen als Beschreibungsmittel für Algorithmen, Graphen als Datenspeicher, Speichern von Daten in Matrizen
• Rekursion	Funktionenplotter, CAS	Funktionale Sprachen, funktionales Programmieren
Weitere Themen • Kryptologie mit Zahlentheorie	CrypTool (Lernsoftware)	Algorithmen, Datensicherheit

Die folgende Abbildung zeigt beispielhaft Möglichkeiten für die Berücksichtigung gebietsübergreifender Aspekte. Man findet in ihr Elemente der Analysis (Parameterdarstellung von Kurven – hier Kardioiden), der funktionalen Programmierung und der Computergrafik.

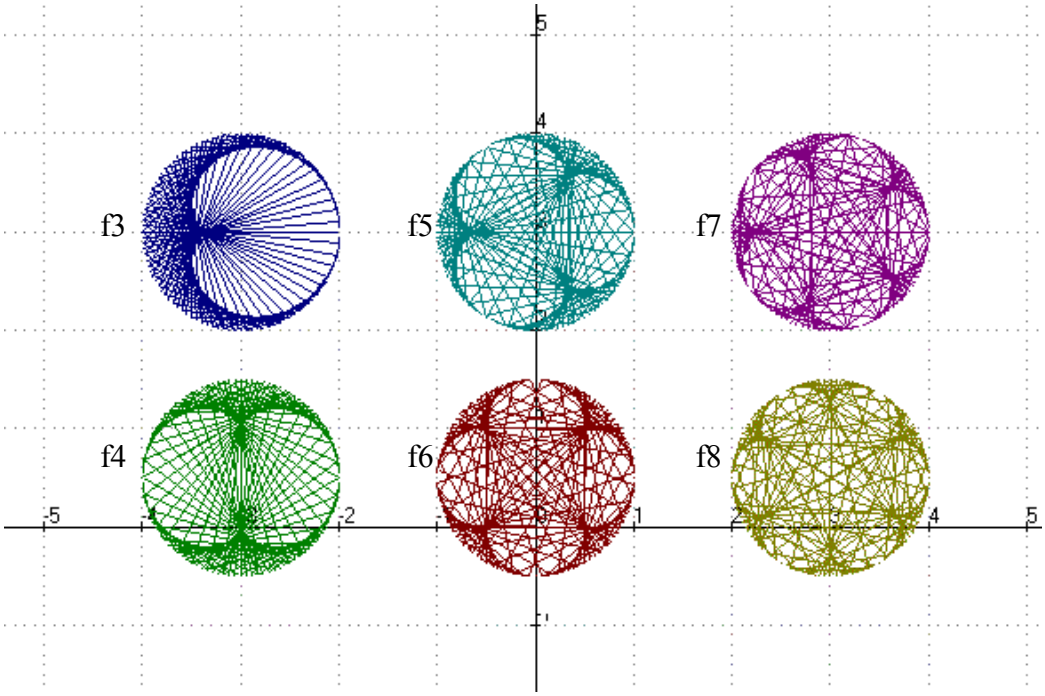


Abb. 1.2.3-a: Kardioiden – interessante Objekte der Computergrafik – Programmieren mit Funktionen

```
f1=a+cos(b*t)  //das ist x(t)          f2=a+sin(b*t)    // das ist y(t)
Mit Aufrufen dieses Bausteins werden nun Strecken zwischen Kurvenpunkten gezeich-
net.
f3=f1(-3,1),f2(3,1),f1(-3,2),f2(3,2)    f4=f1(-3,1),f2(0.5,1),f1(-3,3),f2(0.5,3)
f5=f1(0,1),f2(3,1),f1(0,4),f2(3,4)      f6=f1(0,1),f2(0.5,1),f1(0,5),f2(0.5,5)
f7=f1(3,1),f2(3,1),f1(3,6),f2(3,6)      f8=f1(3,1),f2(0.5,1),f1(3,7),f2(0.5,7)
```

In späteren Kapiteln werden einige der oben genannten Themen aufgegriffen, weitere treten dann noch hinzu (siehe insbesondere Kapitel 3).

1.2.3 Ausgewählte Methoden des Informatikunterrichts und ihre Anwendung im Mathematikunterricht

Einige informatische Methoden eignen sich für die Mathematik in besonderem Maße und können bei vielen mathematischen Themenstellungen eingesetzt werden. Ausführlichere Untersuchungen zu einigen hier genannten Methoden folgen in Kapitel 2.

Methode	**Verwendung in der Informatik**	**Verwendung in der Mathematik**
Modellierung, Zerlegung in Teilprobleme	Bearbeitung komplexer Probleme	Bearbeitung komplexer Probleme (siehe Kapitel 2.1.2, 2.1.3)
Projektmethode	Bearbeitung komplexer Probleme, Software-Life-Cycle (SLC)	Bearbeitung umfangreicherer Problemstellungen, angepasster SLC (Ausarbeitung in Kapitel 2.1.4)
Analyse-Techniken	Analysieren von Programmen, Entdecken von Algorithmen	Analyse von Problemlösungen, Entdecken von Lösungsverfahren (siehe Kapitel 2.1.5.5, 2.1.6)
Modulkonzept, Prozedurkonzept, Wiederverwendung von Modulen und Prozeduren	Wiederverwenden von Modulen und Algorithmen bei anderen Problemstellungen, Wiederverwenden vorhandener (Teil-) Problemlösungen	Wiederverwenden von Modulen und Algorithmen bei anderen Problemstellungen, Wiederverwenden vorhandener (Teil-) Problemlösungen, Bausteindefinitionen (siehe Kapitel 2.1.5)
Programmieren	Softwareerstellung	Kleine Programme zu ausgewählten mathematischen Algorithmen (Ausarbeitung in den Kapiteln 1.3.1 und 2.1.6)

Abb. 1.2.3-a: Ausgewählte Methoden der Informatik

CAS (Computeralgebrasystem) in der Zeit abnehmender Hand-Rechenkompetenzen gute Hilfe leisten könnte.

Abituraufgaben (mathematischer Zug)
gestellt am 6.12.1954 an der Schadow-Oberschule in Berlin-Zehlendorf

Abitur 1954

1) Wie weit sind zwei Punkte voneinander entfernt, deren Polaren für die Parabel $y^2 = 5x$ die Gleichungen I: 10x - 3y = 6 und II: 2x – 3y = 6 besitzen? Es ist ferner zu zeigen, daß der Schnittpunkt der beiden Polaren der Pol der Verbindungslinie der beiden Punkte P1 und P2 ist.

2) Ein zylindrischer, oben offener Behälter vom (lichten) Inhalt V0 und der konstanten Wandstärke a ist mit möglichst wenig Material M herzustellen.
Wie groß müssen der innere (lichte) Radius und die Höhe dieses zylindrischen Hohlgefäßes sein?

3) Der durch Drehung der Ellipse $b^2x^2 + a^2y^2 = a^2b^2$ um die x-Achse entstehende Drehkörper soll in der Richtung der x-Achse zentrisch so durchbohrt werden, daß der Rauminhalt des ringförmigen Restkörpers gleich der Hälfte des Rauminhaltes des Rotationsellipsoids ist.
Der Halbmesser ρ der Bohrung ist zu bestimmen.

Abbildung 1.1-b: Abituraufgaben aus dem Jahr 1954

Benutzen wir Aufgabe 3 dieser alten Abiturklausur, um die heutigen Möglichkeiten des Softwareeinsatzes zu zeigen!

Der Einsatz moderner Hilfsmittel beim Lösen einer alten Abituraufgabe

a) Skizze erstellen
Das sollte hier von Hand erfolgen. Natürlich könnte man auch passende Software benutzen (Textverarbeitung, Zeichenprogramm, ...)

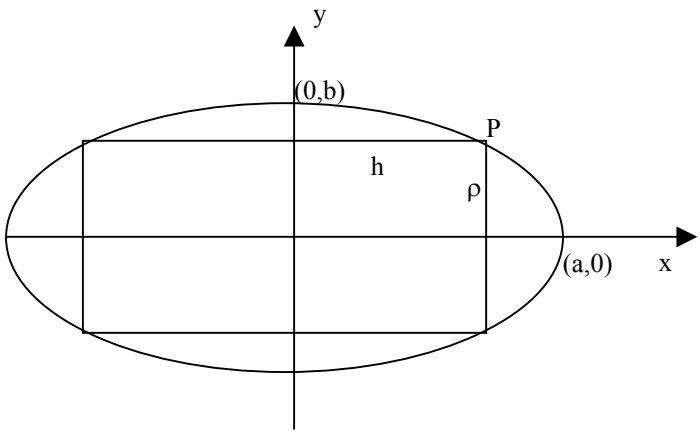


Abbildung 1.1-c: Planfigur zur Abituraufgabe 3

b) Rechnerische Lösung:

Hinweis: Für diese wird möglichst der Text des Lösungsoriginals aus dem Jahr 1954 verwendet.

Man bestimmt zunächst das Volumen des Rotationsellipsoids nach der Formel $V = 2\pi \int_0^a y^2 dx$. Die weitere Dokumentation der Arbeit befindet sich neben dem TI-92-Bild.

Die Ellipsengleichung ist gegeben: $b^2x^2 + a^2y^2 = a^2b^2$.

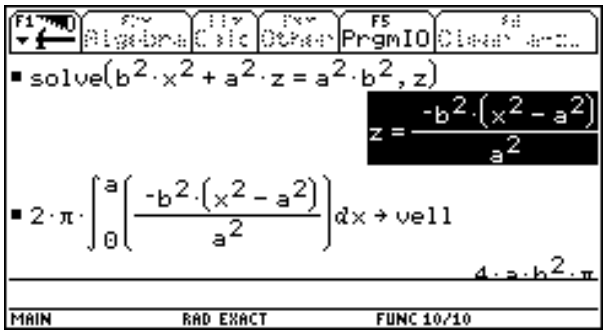


Abb.1.1-d

Da in den entsprechenden Formeln stets y^2 vorkommt, wird für die CAS-Rechnung y^2 →z substituiert.

Damit ist das Volumen des Rotationsellipsoids bestimmt.

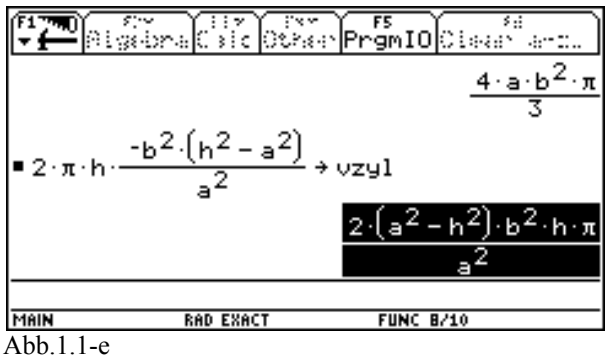


Abb.1.1-e

Das Volumen des Zylinders berechnet sich aus $V = 2\pi\rho^2 h$. Dabei ist hier ρ gleich dem y-Wert und h gleich dem x-Wert von P.

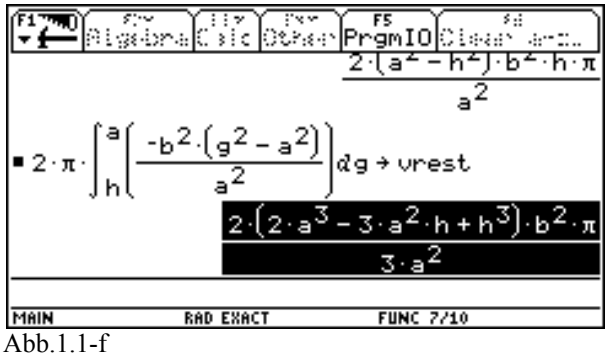


Abb.1.1-f

Das Volumen vrest der Kappe des Ellipsoids (von x = h bis x = a) berechnet sich mit dem angegebenen Term im Integral. Dabei wurde g als Integrationsvariable eingeführt.

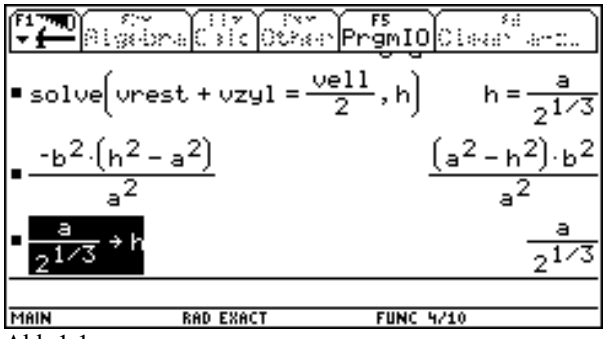


Abb.1.1-g

Laut Aufgabenstellung ist vrest + vzyl gerade die Hälfte des Volumens des Rotationsellipsoids.

Eine Termumformung (hier bedeutungslos).

Berechnung der Höhe des Zylinders unter den angegebenen Bedingungen.

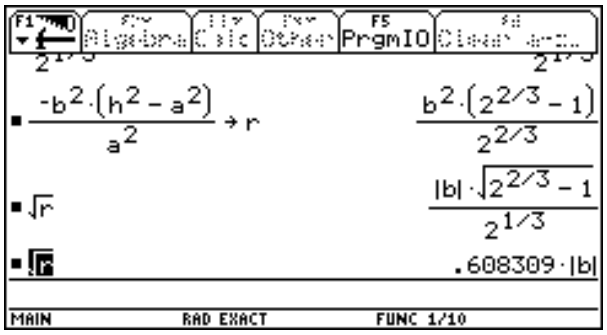


Abb.1.1-h

Der Term r beschreibt das Quadrat des Radius ρ.

Der Halbmesser ρ ist also etwa gleich dem 0.6-fachen von b.

Diese Aufgabenlösung möge zur Einstimmung in die vielfältige Problematik der Verbindung von Mathematik und Computer dienen. Gleichzeitig zeigt die Lösung die Leistungsfähigkeit von Computeralgebra. In diesem Fall arbeitet das CAS fast ausschließlich mit Variablen und führt dabei recht komplizierte Rechnungen durch, die man in einem modernen Mathematikunterricht wohl kaum noch von Hand ausführen würde. Weiterhin wird deutlich, dass die Denkarbeit zum Finden von Ansätzen, die Kompetenz des Anwendens der richtigen Formeln und die des Auswertens der erzeugten Terme die gleiche wie früher bleibt.

Für die heutigen Abituraufgaben (2002) werden detaillierte Formulierungen zur Aufgabenbeschreibung (für den Schüler) und zu den von den Schülern erwarteten Anforderungen (für den Gutachter) erwartet, zumindest für das Land Berlin, das noch kein Zentralabitur hat (2002). Allerdings findet zur Zeit eine rege Diskussion darüber statt, wie man die heute gewünschten offenen Aufgaben und den gewünschten offeneren Unterricht und Computereinsatz auch in Klausur- und speziell Abituraufgaben einbringen kann. Bemerkenswert für die heutige Situation ist auch die bei der obigen Musterlösung verwendete Form der Dokumentation des Lösungswegs.

Bezug zur Informatik
Ein Bezug zur Informatik, der ja in dieser Dissertation besonders interessiert, lässt sich bei den hier angebotenen drei Aufgaben durch die Verwendung von Software zum Rechnen und Zeichnen sowie durch das in der Software mögliche Arbeiten mit Modulen (Bausteinen) herstellen:

- Durchgehende Benutzung eines Computeralgebrasystem (TI-92), siehe Musterlösung zu Aufgabe 3,
- Anwendung eines Funktionenplotters zur Visualisierung bei Aufgabe 1,
- Anwendung eines Ray-Tracing-Programms (z. B. POVRAY) zur Erstellung fotorealistischer Szenen zu den Aufgaben 2 und 3.

Direktere Bezüge zur Schulinformatik werden in den folgenden Kapiteln an anderen Beispielen erarbeitet.

1.3 Entwicklungslinien in der Schulmathematik unter dem Einfluss mathematischer Software und der Informatik

Lehren aus der Vergangenheit – Hinweise für die Zukunft

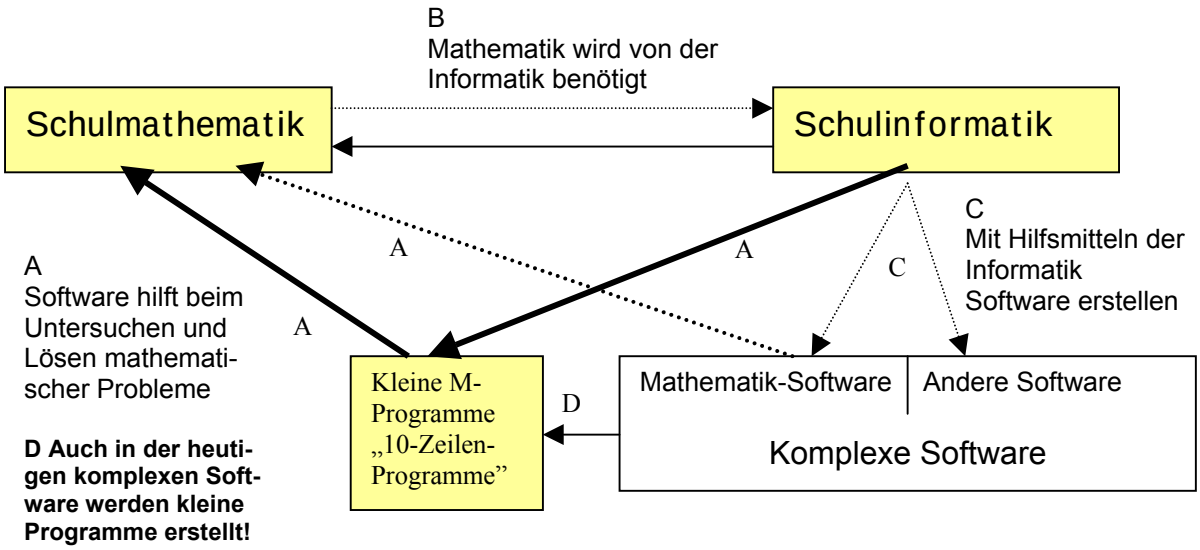


Abbildung 1.3-a: Einige Abhängigkeiten zwischen Mathematik und Informatik in der Schule

Das Beispiel von Kapitel 1.1 führte in die Thematik des Computereinsatzes im Mathematikunterricht (Wege A in Abbildung 1.3-a) ein. Hierzu werden nun in Kapitel 1.3 die wichtigsten Entwicklungslinien von den ersten Anwendungen des Computers in der Schulmathematik bis zur heutigen Situation dargestellt. Die Entwicklung der Schulinformatik wird dabei allerdings nur gestreift. Wir verfolgen zunächst den fett ausgezogenen Weg A in Abb. 1.3-a.

Die folgenden Ausführungen orientieren sich besonders

- an den Tagungen und Veröffentlichungen des „Arbeitskreises Mathematikunterricht und Informatik in der Gesellschaft für Didaktik der Mathematik" und
- an langjährigen eigenen Erfahrungen aus meinen Tätigkeiten als Fachseminarleiter für Mathematik, später für Informatik und umfangreichen Unterrichtserfahrungen in beiden Fächern an der Rückert-Oberschule (Gymnasium) in Berlin-Schöneberg (1964 bis 2001).

Betrachten wir das Inhaltsverzeichnis von „Computer in der Schule" *[Gra85],* aus dem Jahr 1985, siehe Abbildung 1.3-b. Hier wird deutlich, dass der genannte Arbeitskreis sich anfangs besonders mit dem **C**omputereinsatz **i**m **M**athematikunterricht (CiM) und nur wenig mit informatischen Fragen befasst hat. Das änderte sich später – zumindest auf einigen Tagungen, siehe Kapitel 1.3.3 (Tagung 1992).

Computereinsatz im Mathematikunterricht bedeutete damals:

- Den Computer zum numerischen Rechnen, manchmal auch schon zum Zeichnen zu benutzen und
- geeignete Mathematikprogramme dafür zu erstellen (meistens in der Programmiersprache BASIC, später oft in PASCAL).

Inhaltsverzeichnis

Abb. 1.3-b: Inhaltsverzeichnis aus *[Gra85]*

1.3.1 Die „10-Zeilen-Programme”

Der Bezug zur Informatik spielte sich hier auf der programmiertechnischen Ebene ab. Die Programme wurden in der Schule in der Regel von einigen engagierten Lehrern erstellt, die meistens auch die in der Schule sich entwickelnde Informatik unterrichteten. Dazu mussten die entsprechenden Algorithmen analysiert werden. Dieses Vorgehen war allerdings an den Schulen kaum verbreitet. So schreibt z. B. *L. Klingen* in seinem Beitrag (siehe Abb. 1.3-b):

„Vielfach herrschen gänzlich falsche Vorstellungen über den Aufwand, den der Einstieg in Betriebssystem und Programmiersprache fordert. Einige Schulen verfügen heute schon über eine technische Umgebung, welche diesen Aufwand so minimiert, daß man das vorgeschriebene Mathematikcurriculum vollständig ausfahren kann.”

Die folgenden Jahre zeigten, dass *Klingen* die Problematik falsch einschätzte. In der Tat waren die technischen und organisatorischen Hemmnisse so groß, dass CiM noch lange eine Ausnahme bildete. Auch das Schreiben von Programmen blieb für die meisten Lehrer unerreichbar. Diejenigen Lehrer, die das schafften, mussten sich mit einem weiteren Problem auseinandersetzen, wenn sie die Algorithmen im Unterricht besprechen oder die Schüler sogar selbst programmieren lassen wollten: Die Zeit reichte einfach nicht, um den umfangreichen Lehrplan zu erfüllen. So gelang nur wenigen Lehrern eine sinnvolle Integration des Computerein-

satzes in ihren Mathematikunterricht. Auch Demonstrationsprogramme wurden von den Lehrern nicht angenommen.

Sinnvolle Integration bedeutet hier eine

- Verbesserung des Mathematikunterrichts durch CiM an dafür geeigneten Stellen des Lehrplans.

Dennoch entstand eine Fülle von Programmen für Algorithmen aus beiden Sekundarstufen – aber eben ohne große Verbreitung zu finden. Selbst das schöne Buch von *A. Engel [Eng77]* konnte hier kaum etwas ändern:
Arthur Engel: Elementarmathematik vom algorithmischen Standpunkt, Klett-Verlag, Stuttgart 1977.

Die Idee der „10-Zeilen-Programme" war für die meisten Lehrer immer noch nicht umsetzbar. Auf die Rolle des Programmierens im Mathematikunterricht, wie man sie heute sehen kann, wird später noch ausführlicher eingegangen, siehe Kapitel 2.1.7 .

Hier werden noch einige der seinerzeit betrachteten Kurzprogramme angegeben. Dabei wird auch ihre heutige Relevanz eingeschätzt.

Beispiel 1: Iteration
(BASIC-Programm aus: *I. O. Kerner: Numerische Mathematik mit Kleinstrechnern, VEB Deutscher Verlag der Wissenschaften, Berlin 1985*).

```
10      INPUT a, x0
20      LET x1 = (x0 +a/x0)/2
30      PRINT x1
40      LET x0=x1
50      STOP
60      GOTO 20
```

Nach der Eingabe von z. B. a = 2 und x0 = 1 hält das Programm mit der Anzeige des Wertes x1 = 1.5 an. Es wird nach einem Tastendruck in Zeile 20 fortgesetzt und liefert immer bessere Näherungswerte für $\sqrt{2}$. Ein ähnliches Programm würde heute z. B. mit dem CAS-Taschencomputer TI-92 programmiert werden. Mit einem CAS geht es aber auch anders – und für Schüler und Lehrer einfacher. Hierzu später mehr in Kapitel 2.1.7.

Beispiel 2: Der (3a+1)-Algorithmus, *[Eng77, S. 11]*
Dieser Algorithmus, beginnend mit a, erzeugt Zahlenfolgen, die nach einer endlichen Anzahl von Schritten enden (oder auch nicht?). Er könnte im Mathematikunterricht z. B. beim Thema „Folgen" benutzt werden und dort zur experimentellen Arbeit dienen.

(1) Starte mit einer beliebigen natürlichen Zahl a.

(2) Falls a = 1, beende das Programm.

(3) Falls a gerade ist, ersetze a durch a/2 und fahre bei (2) fort.

(4) Falls a ungerade ist, ersetze a durch 3a+1 und fahre bei (2) fort.

Engel gibt das folgende BASIC-Programm an

```
10      INPUT A
20      PRINT A;
30      IF A=1 THEN 90
40      IF  A/2 < > INT(A/2) THEN 70
50      A = A/2
60      GOTO 20
70      A = 3A+1
80      GOTO 20
90      END
```

Hinweis: Eine Bearbeitung des Problems mit einem CAS und dem Programm ANIMATO folgt in Kapitel 3.4.2.

Startet man beispielsweise mit a = 3, so ergibt sich die Zahlenfolge 3, 10, 5, 16, 8, 4, 2, 1. Näheres zu diesem Problem einschließlich einer instruktiven grafischen Darstellung und der Verbindung zur Informatik wird in Kapitel 3.1 dargestellt. Beide Beispiele zeigen:

Mit kleinen Programmen kann viel erreicht werden. Insbesondere ergibt sich die Möglichkeit experimentellen Arbeitens:

- Experimentieren mit verschiedenen Eingaben,
- Ausgaben auswerten,
- Vermutungen aufstellen und ggf. verifizieren.

Diese Arbeitsweise ist heute hoch aktuell und mit den heutigen Mitteln leichter als damals zu realisieren, siehe Kapitel 1.4. Kleine Programme können heute auch mit der vorliegenden komplexen Software (z. B. im CAS) geschrieben werden, siehe Weg D in Abb. 1.3-a.

Beispiel 3: Matrizenprodukt und Modulprogrammierung (PASCAL-Prozedur aus dem *Softwaresystem MATRIX*, *E. Lehmann, Comet-Verlag, Duisburg1990*)

In diesem Beispiel wird ein Aspekt vorgestellt, der später erhebliche Bedeutung gewinnen sollte. Mit Hilfe von fest definierten Funktionen und ihren Parametern – hier die Matrizenmultiplikation – können nämlich auch komplexe Probleme mit kurzen Programmen bewältigt werden.

Programmtext	**Erläuterungen**

procedure PROD (var mat3:matrix; mat1,mat2:matrix);
(* berechnet das Produkt zweier Matrizen, falls
mat1.s=mat2.z *)
(* [produkt:boolean] ist global definiert *)
var i,j,k: integer;
s : real;
 	
begin
if mat1.s <> mat2.z then
begin
writeln;
writeln('Das Matrizenprodukt ist nicht definiert!');
mat3.z:=0; mat3.s:=0;
rk:=readkey;
produkt:=false;
exit;
end
else
	Der Datentyp *matrix* wurde folgendermaßen definiert:
	
	TYPE matrix = RECORD
	wert: ARRAY[1..maxzeilen, 1..maxspalten]
	OF REAL; {Matrixelemente}
	m: INTEGER; {Zeilenanzahl}
	n : INTEGER; {Spaltenanzahl}
	END;
	
	Die Spaltenanzahl der 1. Matrix stimmt nicht mit der Zeilenanzahl der 2. Matrix überein.
	
	Bei Übereinstimmung:

begin
produkt:= true;
mat3.z := mat1.z; mat3.s:=mat2.s;
 	
for i:=1 to mat1.z do
for k:=1 to mat2.s do
begin
s:=0;
for j:=1 to mat1.s do
s:=s + mat1.wert[i,j] * mat2.wert[j,k];
mat3.wert[i,k]:=s;
end;
end
end (* of PROD *);
	Zeilen- und Spaltenanzahl der Ergebnismatrix
	
	Über alle Zeilen von mat1,
	über alle Spalten von mat2,
	
	über alle Spalten von mat1,
	Bildung der Skalarprodukte.
	Das errechnete Element an der Stelle (i,k)
	der Produktmatrix
	Das Produkt steht in der Matrix mat3.

Hinweis: Mit der hier benötigten Programmlänge kommt ein „10-Zeilen-Programm" an seine Grenzen.

Entsprechende Prozeduren gibt es für zahlreiche weitere ein- und zweistellige Matrizenverknüpfungen. Damit kann das Programmieren und Lösen komplexer Matrizenaufgaben im Wesentlichen auf ein Aufrufen geeigneter Matrizenprozeduren mit passenden Parametern zurückgeführt werden.
Dieser Ansatz ist verallgemeinerungswürdig und kann heute beim CAS-Einsatz eine wesentliche Rolle spielen (mehr dazu u. a. in Kapitel 2.1.5)

1.3.2 Weitere Mathematikprogramme

Demonstrationsprogramme
Ein weiterer Ansatz zur Verbreitung von Computern in der Schulmathematik versuchte es mit Demonstrationsprogrammen, die mathematische Sachverhalte veranschaulichten. Auch hier gab es keinen Durchbruch, da es keine genügende Anzahl von Computern an den Schulen gab. Diese wurden dann auch noch von den Informatiklehrern quasi als ihr Eigentum betrachtet. Damit scheiterte auch dieser Ansatz an organisatorischen Gegebenheiten, aber auch an mangelnder Kompetenz von Lehrern.

Heute gibt es jedoch auch für Demonstrationsprogramme gute Einsatzmöglichkeiten mit eigenen methodischen Wegen, zumal die Bedienung durch die modernen Oberflächen wesentlich vereinfacht werden kann. Beispielsweise finden sich im Internet zu zahlreichen Problemen leicht bedienbare Java-Applets.

Trainingsprogramme
Gelegentlich werden auch Trainingsprogramme eingesetzt, die gewisse Grundfertigkeiten im Rechnen üben helfen (Drill and Practice). Beispiele sind Programme für die schrittweise Lösung linearer Gleichungssysteme oder zum Umformen von Termen. Trainingsprogramme sind auch heute Randerscheinungen, zumal sich heute das langwierige Üben von Algorithmen durch die Verfügbarkeit von CAS anders als früher darstellt. Man vergleiche hierzu z. B. *[Herget, Heugl, Kutzler, Lehmann, MNU, Heft 8, 2001]*. Dort wird die Frage gestellt und untersucht: *„Welche handwerklichen Rechenkompetenzen sind im CAS-Zeitalter unverzichtbar?"*

Die Mathematik-Lehrpläne der 80-er Jahre und auch noch einige Jahre danach drückten sich bezüglich des Computereinsatzes (wie auch vorher bzgl. des Taschenrechnergebrauchs) sehr vorsichtig aus und gaben bestenfalls einige Hinweise im allgemeinen Teil – meistens ohne Bezug auf konkrete inhaltliche Fragen. In dem Berliner Rahmenplan von 1973 werden die breiten Einsatzmöglichkeiten des Computers in den Themenbereichen des Mathematikunterrichts noch nicht erwähnt. Es gab lediglich einen speziellen, von den übrigen Angeboten isolierten Kurs B5: „Behandlung mathematischer Probleme mit Computereinsatz", in dem einige ausgewählte mathematische Fragestellungen zur Bearbeitung mit Computern empfohlen wurden.

Bemerkenswert sind dann jedoch die *„Empfehlungen zum Computereinsatz im Mathematikunterricht der Sekundarstufe II"* [*MNU, 1986, Heft 2*] . Hier wird schon mit großer Klarheit gesehen, welche Chancen sich durch Computereinsatz bieten.

„Für den Einsatz des Computers im Mathematikunterricht sehen wir folgende Schwerpunkte:

- *Einsatz von Rechnern als Werkzeug für die Bearbeitung von Problemen, d. h. Computer als Arbeitsmittel und nicht primär als simples Demonstrationsmedium*
- *verstärkte Nutzung der Computer-Grafik als Arbeitsmittel, als Verständnishilfe und zur Motivation,*
- *stärkere Einbeziehung experimentellen Arbeitens in den Mathematikunterricht, z. B. zum Auffinden mathematischer Gesetzmäßigkeiten und mathematischer Modelle,*
- *Herausstellung und Förderung des Anwendungsbezuges,*
- *Förderung des selbständigen und kooperativen Arbeitens,*
- *kritische Reflexion und Bewertung des Rechner-Einsatzes.*

...

Schulischer Mathematikunterricht muß sich mit der Existenz neuartiger Software, z. B. Computer-Algebra-Systemen auf Mikrocomputern, auseinandersetzen. Er sollte die Verbreitung solcher Systeme zum Anlaß nehmen, seine Bildungs- und Zielvorstellung zu reflektieren und den Umfang der zu vermittelnden algorithmischen Fertigkeiten immer wieder neu zu bestimmen unter Berücksichtigung des Zusammenhangs zwischen Übung und Verständnis. Es erhebt sich insbesondere die Frage, welches Gewicht algorithmische Teile bei schriftlichen Leistungsmessungen haben sollen."

Auszug aus [MNU86]

Auch wird bereits 1986 auf die Existenz der neuartigen Computeralgebrasysteme hingewiesen. Ein Beispiel ist das System MUMATH. So wird deutlich, wie lange die Bemühungen um einen effizienten Computereinsatz im Mathematikunterricht schon gehen und wie relativ wirkungslos derartige Empfehlungen gewesen sind. Bis zum Jahr 2002 sind seit diesen Empfehlungen bereits 17 Jahre vergangen und bestenfalls in den letzten fünf Jahren wird die Akzeptanz deutlich größer.

Als Ergebnis der 80-er Jahre muss man aber festhalten:

> Das Programmieren und das dafür notwendige Analysieren von Algorithmen (z. B. in Form von Programmablaufplänen oder Struktogrammen) konnte sich im Mathematikunterricht nicht durchsetzen! Komplexe Softwaresysteme für den Mathematikunterricht waren noch kaum bekannt. Damit war der Computereinsatz im Mathematikunterricht auf wenige Ausnahmen beschränkt.

Viele der seinerzeit schon reichlich vorhandenen einschlägigen Bücher – einige wurden oben genannt – erfüllten allerdings mit ihren zahlreichen ausgeklügelten Algorithmen für die unterschiedlichsten Probleme wichtige Aufgaben für die weitere – nun professionelle – Entwicklung von Mathematik-Software. Sie waren die Vorstufe für die heutigen komplexen Softwaresysteme, z. B. zur Computeralgebra (CAS).

Die weiteren Entwicklungen bis heute lassen sich in den Heften des *„Arbeitskreises Mathematik und Informatik" der GDM* (siehe Literaturverzeichnis, z. B. *[Her93]*) , aber auch in einer Fülle von Beiträgen in Fachzeitschriften und Büchern gut verfolgen. Aber trotz des vorliegenden umfangreichen Materials blieb die Verbreitung des Computereinsatzes im Mathematikunterricht gering. Ein grundlegender Wandel konnte erst dann erwartet werden, wenn die Lehrer (und die Schüler) nicht mehr genötigt waren, eigene Programme für den Mathematikunterricht zu erstellen.

> Es wurden Systeme benötigt, die die Standardaufgaben des Mathematikunterrichts auf einfache Weise erledigen und gleichzeitig Möglichkeiten selbstständigen, entdeckenden Lernens bereitstellen können.

Inzwischen gibt es zahlreiche derartige Systeme. Diese wurden zwar in der Regel nicht direkt für den Schulunterricht entwickelt, jedoch erwiesen sich Teilbereiche davon als sehr nützlich für den Unterricht. Im Wesentlichen sind es drei Arten von Softwarepaketen, die heute für den Mathematikunterricht zur Verfügung stehen:

CAS	Computeralgebrasysteme,
DGS	Dynamische Geometriesysteme,
TK	Tabellenkalkulationsprogramme.

1.3.3 Mathematik und Informatik – erste Integrationsansätze

Ansätze zur Integration informatischer Inhalte in den Mathematikunterricht lassen sich u. a. in den schon oben erwähnten Berichten über die Tagungen des Arbeitskreises „Mathematik und Informatik" in der GDM verfolgen. Hier ist insbesondere der *Bericht von 1992* zu nennen:

„Wieviel Termumformung braucht der Mensch" – Fragen zu Zielen und Inhalten eines künftigen Mathematikunterrichts angesichts der Verfügbarkeit informatischer Methoden (Horst Hischer, Hrsg., Franzbecker-Verlag, Hildesheim 1993).

In diesem Bericht schreiben *H. Löthe und C. Wagenknecht* über die *„Integration informatischer Begriffe in die Mathematik – erste Erfahrungen" (S. 24 f.)*. Die Erfahrungen beruhen auf dem Projekt CIMS (Computerintegration in das Mathematiklehrerstudium) 1991 an der PH Ludwigsburg. Die Autoren beschäftigen sich dabei insbesondere mit „Formen und Stufen der begrifflichen Integration". Sie unterscheiden:

- Additive Hinzunahme von Begriffen aus der Informatik
- Ausweitung von Begriffen der traditionellen Mathematik (z. B. des Funktionsbegriffs)
- Konkretisierung mathematischer Begriffe (durch Implementation am Computer und Anwendung)
- Notation vs. „Ding an sich" (unterschiedliche Problemsituationen können unterschiedliche Notationen erfordern)

- Konstruktive Fassung mathematischer Begriffe (durch Erarbeitung aus Anwendungsbereichen)

.

Konkretisiert werden die Überlegungen insbesondere an Beispielen für das Arbeiten mit Funktionen in der Sprache SCHEME. Die von den Autoren durchgeführten feinsinnigen Überlegungen zum Funktionsbegriff sind allerdings für die Schulpraxis wenig hilfreich. Was für die Praxis bleibt, ist die immer wieder belegte Unterrichtserfahrung:

> Es ist in der Regel sinnvoll für den Unterrichtserfolg, einen Gegenstand, einen Sachverhalt, eine Aufgabenstellung usw. in verschiedenen Sichtweisen zu bearbeiten.

In dem oben genannten Tagungsbericht findet sich auch ein Beitrag *„Informations- und kommunikationstechnologische Bildung – allgemeine und fachbezogene Ziele" (letztere für den Mathematikunterricht), S.146–147.* Dieser ist ein Auszug aus

Neue Technologien und Allgemeinbildung, Band 11: Mathematik – Anregungen für den Unterricht, Niedersächsiches Kulturministerium (Hrsg.), Hannover 1990.

Als fachbezogene Ziele werden dort für den Mathematikunterricht genannt:

- Modellbildung und Simulation,
- Algorithmus,
- Codierung,
- Möglichkeiten und Grenzen des Rechners.

Das sind Aspekte, die auch im Rahmen der vorliegenden Arbeit eine wichtige Rolle spielen, siehe Kapitel 2.

In dem Beitrag von *Horst Hischer* geht es *um „Neue Technologien als Anlaß einer erneuten Standortbestimmung für den Mathematikunterricht", S. 148*, ein Thema, das heute in der Diskussion über den Einsatz neuer Medien in der Schule wieder hoch aktuell ist. Passend für das Anliegen meiner Arbeit ist besonders These 5:

„Auch die Wissenschaft Informatik spielt über die Informations- und Kommunikationstechnologien eine bedeutsame Rolle im Rahmen der technologischen Synergie. Eine ihrer wesentlichen fundamentalen Ideen und Methoden ist die Algorithmik. Hierin erweist sich die Informatik als Sprößling der Mathematik, gehört doch der Algorithmus zu den mathematischen Urbeständen. Die während der Zeit des Bourbakismus in der Mathematik in den Hintergrund getretene algorithmische Methode bekommt jedoch im Rahmen neuerer mathematischer Theorien und Anwendungen – Fraktale, Chaostheorie, Katastrophentheorie, Dynamische Systeme – eine bedeutsamere Rolle als je zuvor."

Selbstverständlich muss dieser Aspekt in der vorliegenden Arbeit berücksichtigt werden! Man vergleiche hierzu die Ausführungen in den Kapitel 2.1.6 und 2.1.7 sowie in den vielen Konkretisierungen des Kapitels 3.

Es wird sich aber zeigen, dass eine Einbeziehung informatischer Aspekte nur über Algorithmik zu wenig ist, zumal deren Bedeutung angesichts der CAS-Systeme heute neu zu bewerten ist.

Auch in der *„Stellungnahme zur Einbeziehung von Inhalten und Methoden der Informatik in den Mathematikunterricht der Sekundarstufe 1 und in die Hochschulausbildung von Mathematiklehrern" (GDM, Juli 1981),* veröffentlicht u. a. in dem oben genannten Tagungsbericht 1992, wird die Aufnahme informatischer Methoden in die Hochschulausbildung von Mathematiklehrern gefordert, wenn auch einseitig auf Algorithmik ausgerichtet. 1986 weist dann die GDM in ihren *„Überlegungen und Vorschläge zur Problematik Computer und Unterricht" (in MNU, 1986, Heft 6),* auf diverse diesbezügliche Schwierigkeiten der Akzeptanz hin:

„Wir müssen heute aber auf verwickelte und tiefliegende Probleme hinweisen, vor die sich allgemein der Unterricht, insbesondere der Mathematikunterricht, in Konzeption und Praxis durch die verschiedenen möglichen Weisen des Umgangs mit dem Computer gestellt sieht, und auf spezielle Probleme, die sich mit der angestrebten informationstechnischen Bildung für alle Schüler und Jugendlichen gegeben sind."

Explizit werden dann genannt und ausgeführt: Probleme der Rechtfertigung, inhaltliche Veränderungen des Mathematikunterrichts, Veränderungen des Lernens, Probleme fächerübergreifender Ansätze und Probleme der Lehrerfortbildung.

Zusammenfassend lässt sich feststellen, dass es bisher kaum gelungen ist, bewusst informatische Aspekte im Mathematikunterricht zu berücksichtigen oder gar in den Mathematik-Lehrplänen zu verankern.

1.4. Mathematikunterricht heute

1.4.1 Die heutigen Unterrichtsvoraussetzungen

Die Bedingungen für eine zeitgemäße Gestaltung von Mathematikunterricht haben sich in den letzten Jahren sehr verbessert und sind teilweise durchaus günstig.

1.4.1.1 Hardware und Software

Hardware – oft sind die notwendigen Voraussetzungen gegeben

- Die Ausstattungen vieler Schulen mit Personalcomputern, mit schulinternen Netzen und mit Verbindungen zum Internet sind inzwischen meistens gut.
- Als weitere Möglichkeit speziell für den Mathematikunterricht sind an etlichen Schulen grafische Taschenrechner oder als weitergehende Variante Taschencomputer mit CAS (z. B. TI-92-Plus) vorhanden, teilweise in höherer Stückzahl. Dabei besteht an einigen Schulen auch die Möglichkeit, derartige Rechner an Schüler auch für längere Zeit zu verleihen und damit auch für Hausarbeiten verfügbar zu machen. In einigen Bundesländern sind graphische Taschenrechner (ohne CAS) verbindlich eingeführt.
- Als Projektionsmöglichkeiten im Klassenraum stehen Beamer (für PCs) oder View Screens für Taschenrechner und Taschencomputer zur Verfügung.
- Zahlreiche Schüler haben auch zu Hause selbst oder über ihre Eltern Personalcomputer und Internet-Anschluss.

Hardware – es gibt aber auch noch Behinderungen

- Computerräume stehen angesichts der vielen Klassen und Kurse mit drei oder mehr Mathematikstunden häufig nicht – zum Stundenplan passend – zur Verfügung. In der Regel hat der Informatikunterricht Vorrang. Unterricht mit dem Computer auf Anmeldung ist weit weniger günstig als spontane Entscheidungsmöglichkeit für einen Computereinsatz. Bemerkung: Bei Taschencomputern oder graphischen Taschenrechnern in der Hand der Schüler entsteht dieses Problem nicht!
- Zu den Behinderungen gehört auch die mangelnde Vertrautheit vieler Mathematiklehrer mit den an der Schule vorhandenen Computernetzen.

Software – viele Angebote

- Als besonders weitreichend für fast alle Bereiche der Schulmathematik sind Computeralgebrasysteme (CAS) anzusehen. Viele Schulen besitzen das Programm DERIVE, für Taschencomputer gibt es sehr ähnliche, fest installierte Software, etwa beim TI-92-PLUS. Andere Computeralgebrasysteme sind MAPLE oder MATHCAD.
- Dynamische Geometriesysteme (DGS) sind besonders für den Unterricht in der Sekundarstufe 1 geeignet. Beispiele: EUKLID, CABRI, SKETCHPAD, GEONEXT, CINDERELLA. Bei diesen Systemen werden zunehmend auch Schnittstellen zur Darstellung von Funktionstermen oder gar zu CAS geschaffen.
- Auch Tabellenkalkulationsprogramme decken einen größeren mathematischen Bereich ab. Beispiel: EXCEL
- Funktionenplotter haben teilweise spezielle Eigenschaften, die über die Grafik von CAS hinausgehen. Beispiel: ANIMATO – mit besonderen Stärken in der Gestaltung der Bilder und bei mathematischen Animationen.

- Es gibt diverse Programme für einen engeren Anwendungsbereich, z. B.
  – ANALYGEO für Belange der Analytischen Geometrie
  – POVRAY für fotorealistische Gestaltung von „Szenen", ebenfalls verwendbar für Analytische Geometrie und andere Anwendungen in der Computergrafik.
- Für die Software gibt es zumindest bei einigen Programmen günstige Schullizenzen oder sogar Landeslizenzen. Häufig können die Schüler die Software auch zu Hause auf ihren Rechnern benutzen, was sehr wünschenswert ist.
- Die Software kann in der Regel in diversen Arbeitsweisen eingesetzt werden: Rechnen, Zeichnen, Programmieren, Experimentieren und Entdecken im Rahmen zahlreicher Unterrichtsformen.

Software – auch hier gibt es noch Behinderungen

- An erster Stelle steht die mangelnde Vertrautheit vieler Lehrer mit den Programmen, besonders mit ihren sinnvollen Einsatzmöglichkeiten.
- Computereinsatz fordert zwingend andere Unterrichtsmethoden als den bisherigen lehrerzentrierten Unterricht. Es fehlt gerade hierin an geeigneter Fortbildung der Lehrer.
- Die meisten Systeme sind nicht für die Belange der Schule konstruiert. Der Lehrer muss also häufig nach methodischen Wegen suchen, um das Überangebot an Optionen einzuschränken.

1.4.1.2 Neue Unterrichtskultur, neue Aufgabenkultur

Computeralgebrasysteme und andere Software leisten, wie oben angedeutet, in manchen Schulen bereits einen wesentlichen Beitrag zu einem modernen Mathematikunterricht. Dieser wird zur Zeit bestimmt durch Bemühungen, die zu einer Steigerung der Effizienz des mathematisch-naturwissenschaftlichen Unterrichts führen sollen. Hierzu gehören u. a. die Aspekte

- Veränderte Unterrichtskultur mit offenen Unterrichtsformen,
- mehr Schülerzentrierung, insbesondere Bemühungen zu mehr selbständiger Arbeit der Schüler,
- neue Aufgabenkultur, d. h. unter anderem: Offene Aufgabenstellungen, mehr Anwendungsbezug,
- experimentelles, entdeckendes Lernen,
- Einsatz neuer Medien.

Weiteres hierzu kann auch aus Abbildung 1.4-a abgelesen werden. Außerdem wird hingewiesen auf die Ausführungen zur TIMSS-Studie von *Henn [Hen99]* und *Blum, W. / Neubrand, M. [Blu98]*.

Abbildung 1.4-a zeigt die Module des 1998 aufgelegten bundesweiten Modellversuchs SINUS zur Steigerung der Effizienz des mathematisch-naturwissenschaftlichen Unterrichts.
Hinweis: Das Computermodul wurde von mir eingefügt und ist in dem Modellversuch nicht enthalten.

Steigerung der Effizienz des Mathematikunterrichts

Hauptziel aller Bemühungen zur Steigerung der Effizienz des Mathematikunterrichts ist eine Weiterentwicklung der gesamten Unterrichtskultur, insbesondere hin zu offenen Unterrichtsformen, in denen der Lehrer nicht mehr die dominierende Rolle spielt und bei denen sich die Leistungsfähigkeit der Schüler besser entwickeln kann.

Modul 1 Weiterentwicklung der Aufgabenkultur im mathematisch-naturwissenschaftl. Unterricht	Modul 2 Naturwissenschaftliches Arbeiten (experimentelles Arbeiten)	Modul 3 Aus Fehlern lernen
Modul 4 Sicherung von Basiswissen – verständnisvolles Lernen auf unterschiedlichen Niveaus	Modul 5 Zuwachs von Kompetenz erfahrbar machen: Kumulatives Lernen	Modul 6 Fächergrenzen erfahrbar machen: Fachübergreifendes und fächerverbindendes Lernen
Modul 7 Förderung von Mädchen und Jungen	**Computer-Modul** (ein im Projektansatz nicht erwähntes Modul) Einsatz neuer Medien: Computeralgebrasysteme (CAS), Computergrafik (CGK), dynamische Geometriesysteme (DGS), weitere Unterrichtssoftware, Internet (WWW)	Modul 8 Entwicklung von Aufgaben für die Kooperation von Schülern
Modul 9 Verantwortung für das eigene Lernen stärken	Modul 10 Prüfen: Erfassen und Rückmelden von Kompetenzzuwachs	Modul 11 Qualitätssicherung innerhalb der Schule und Entwicklung schulübergreifender Standards

Abb. 1.4-a: SINUS-Modellversuch → Steigerung der Unterrichtseffizienz

Die folgende Abbildung 1.4-b zeigt einige wichtige Elemente für eine neue Unterrichtskultur und ihre Vernetzung.

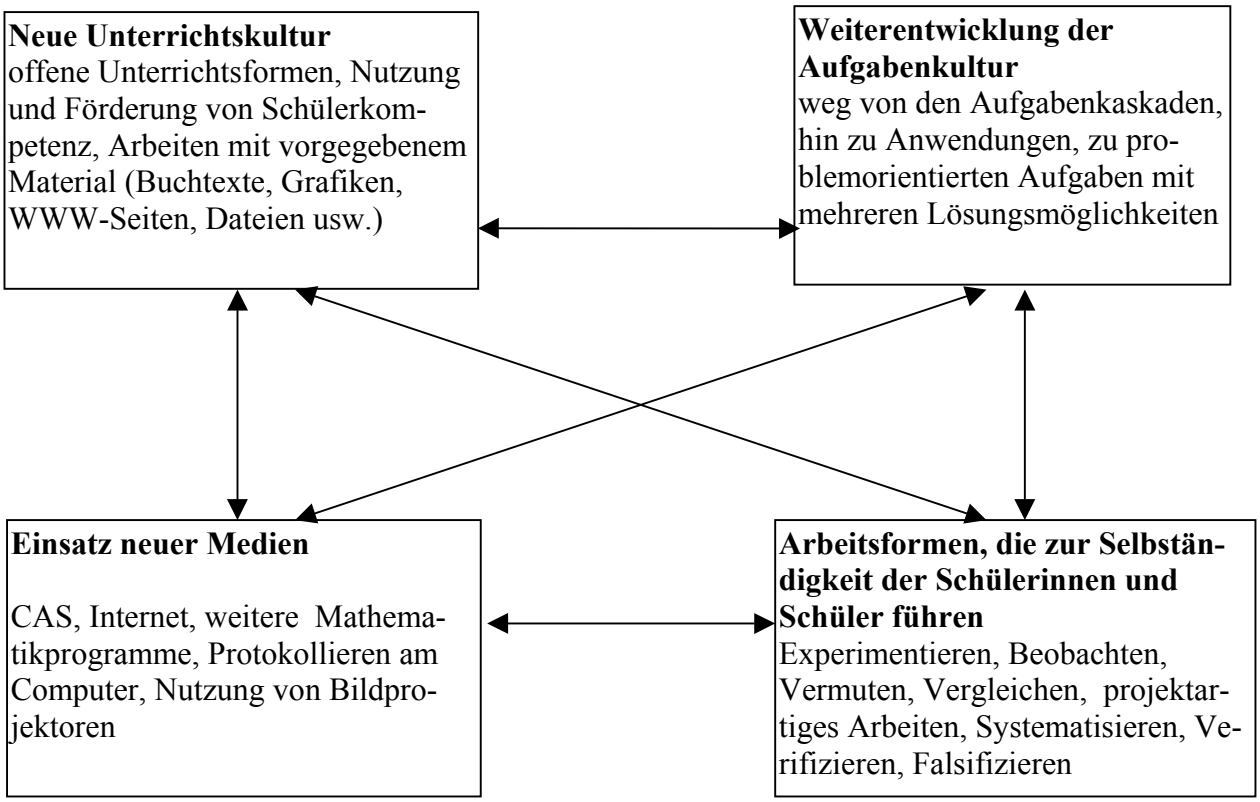


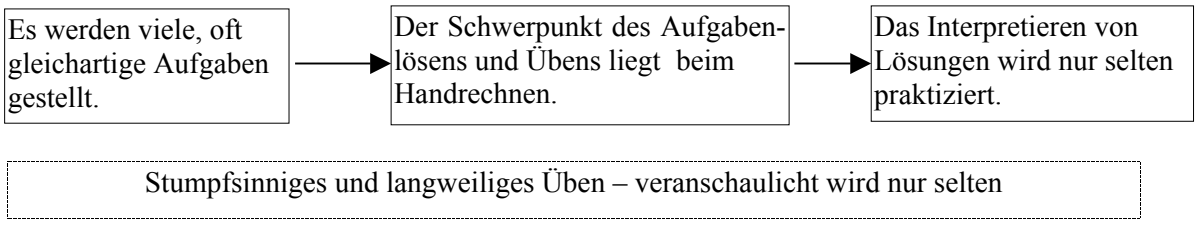
Abb 1.4-b: Einige Aspekte für einen modernen Mathematikunterricht

Aus diesen Gegebenheiten ergibt sich auch für die Lehrer eine stark veränderte Rolle. Sie werden mehr und mehr zu „Unterrichtsmanagern".

Die neue Rolle der Lehrer als „Unterrichtsmanager" :
Sie sind Organisatoren der Unterrichtsabläufe, sie haben den Überblick, kennen die (mathematischen) Hintergründe, sie lassen Arbeitswege dokumentieren, lassen analysieren und systematisieren, ordnen lokal und global, sie bewerten.

Bei der Analyse der Unterrichtspraxis erweist sich die Weiterentwicklung der Aufgabenkultur als besonders dringlich. Aber noch ist die in Abbildung 1.4-c dargestellte Auffassung zum Aufgabenlösen weit verbreitet:

Aufgabenbearbeiten im Mathematikunterricht alter Prägung



Stumpfsinniges und langweiliges Üben – veranschaulicht wird nur selten

Abb. 1.4-c: Herkömmliche Aufgabenkultur

Das Wunschbild dagegen sieht so aus:

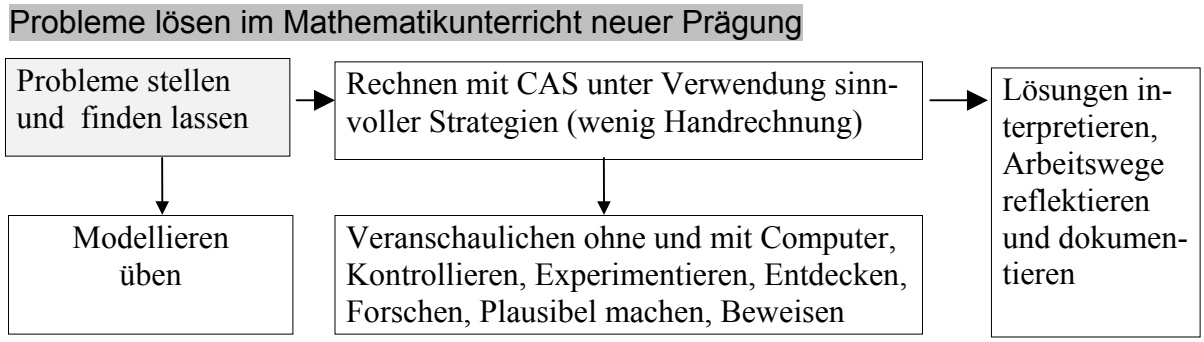


Abb. 1.4-d: Probleme lösen im Mathematikunterricht und die neue Rolle des Übens

Mathematikunterricht neuer Prägung stellt mehr offene Probleme,

- übt deren Modellierung,
- sucht nach unterschiedlichen Lösungswegen,
- benutzt in sinnvoller Weise den Computer zum Rechnen und häufigen Veranschaulichen, zum Kontrollieren, zum Experimentieren, Forschen, Entdecken, Vermuten,
- erlaubt den Schülern, Fehler zu machen und diese zu reflektieren,
- übt den sinnvollen Umgang mit anderen neuen Medien,
- übt die Darstellung des Arbeitsweges,
- übt auch die Interpretation.

Angesichts der neuen Möglichkeiten erhält auch das Beweisen einen neuen Stellenwert:
Sind die Computerergebnisse zuverlässig, siehe z. B. Kapitel 3.4.6? – Muss das überhaupt noch bewiesen werden?

Abbildung 1.4-e zeigt Erweiterungen der heutigen Aufgabenpraxis, die zu offeneren Aufgabenstellungen führen können. Man wird daran erinnert, dass in der Unterrichtspraxis zur Zeit vielfach mit engen Aufgabenstellungen gearbeitet wird. Vorhandene Möglichkeiten, wie sie in der Abbildung angedeutet sind, werden noch zu wenig ausgenutzt.
Bislang kaum praktizierte Möglichkeiten sind:

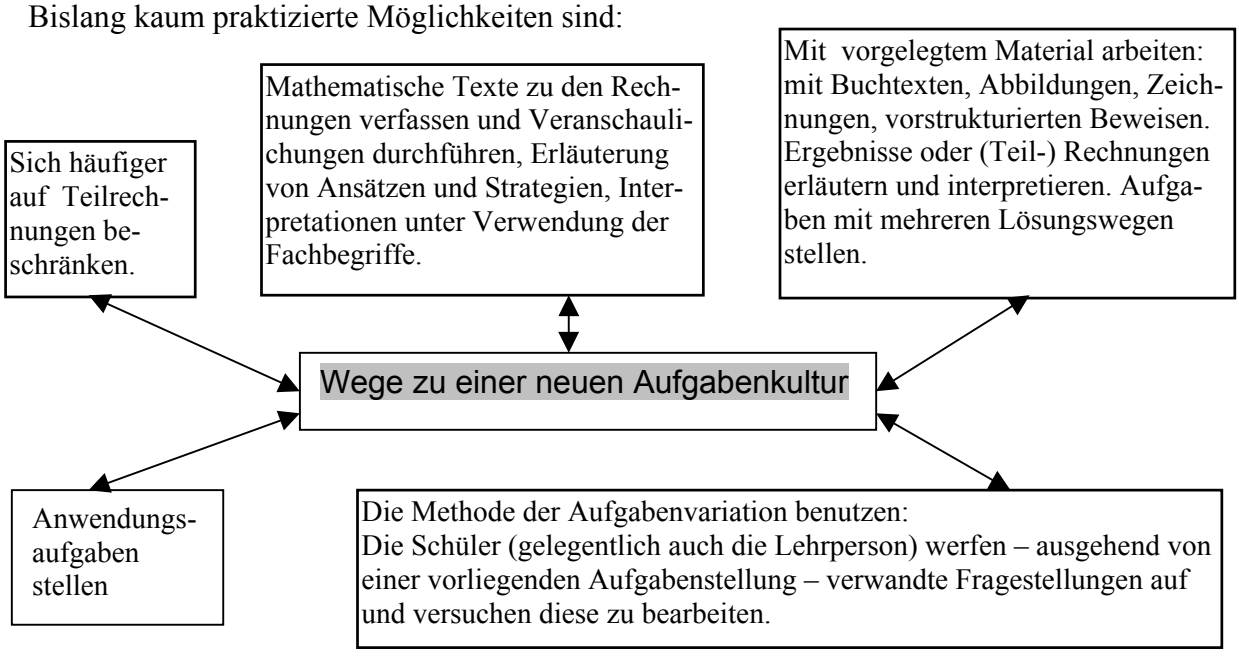


Abb. 1.4-e: Neue Aufgabenkultur – die Aufgabenstellungen sind viel variabler und offener

Abbildung 1.4-f zeigt ein Beispiel für eine sehr offene Problemstellung.

Aufgabe:

Erfindet mathematische Aufgabenstellungen zu dem abgebildeten Türgitter!

Derartige offene Aufträge führen oft zu phantasievollen Aufgaben, von denen der Lehrer in Abhängigkeit von der Lerngruppe und vom Lehrplan entscheiden muss, ob er sie bearbeiten lassen kann. In jedem Fall haben die Schüler eine hohe Motivation zum Lösen der selbstgestellten Aufgaben. So kann z. B. projektartiger Unterricht entstehen.

Abb. 1.4-f: Bilder als Anlässe, neue Aufgabenstellungen zu entwerfen

Konsequente Ausnutzung von Schülerkompetenz

Offene Unterrichtseinstiege, etwa über komplexe Problemstellungen, die auch außermathematische Aspekte (Anwendungen) in Betracht ziehen, erreichen weit mehr Schüler, als das bei engen, streng an die Mathematik angelehnten Problemen der Fall ist! Entsprechendes lässt sich sogar für innermathematische Aufgabenstellungen feststellen, sofern diese offen und weit genug gefasst sind. Damit ist für den Unterricht zumindest in der Phase des Ideensammelns (brainstorming) mehr Schülerkompetenz vorhanden. Durch geschickte Steuerung des Lehrers geht es nun darum, dieses Anfangsinteresse für die längerfristige Motivation und damit auch für die mathematischen Lernziele zu nutzen. Das kann nicht erreicht werden, indem man sich sofort auf die mathematischen Aspekte zurückzieht. Gelegentlich sollten auch Ideen verfolgt werden, die fachübergreifend sind. Dadurch darf man sich erhoffen, mehr Schüler als gewöhnlich auch für den mathematischen Anteil am Gesamtproblem zu interessieren. Bei einem derartigen Ansatz zeigt sich erfahrungsgemäß immer wieder, dass eine breite Schülerkompetenz für die verschiedensten Bereiche vorliegt. Diese gilt es auszunutzen, indem man sie im Unterricht allen Beteiligten zur Verfügung stellt.

Der Schülerspezialist bringt sein Wissen ein

- im Unterrichtsgespräch,
- in Form eines Vortrags,
- bei der Partnerarbeit,
- durch Hilfestellung für andere Schüler(gruppen) usw.

Schüler unterrichten Schüler!

Schüler werden zu „Hilfslehrern" – nach dem Motto: „Schüler unterrichten Schüler" – in Bereichen, in denen dem Lehrer möglicherweise Kompetenz fehlt, aber auch in Bereichen, in denen er diese Kompetenz durchaus hat! Aber der Lehrer muss ja aus pädagogischen Gründen nicht alles an die Schüler weitergeben, was er zu dem Thema weiß! Denn:

> Steuerung des Unterrichts und der Unterrichtsinhalte durch die Schülerinnen und Schüler bedeutet gleichzeitig mehr Motivation für diese!

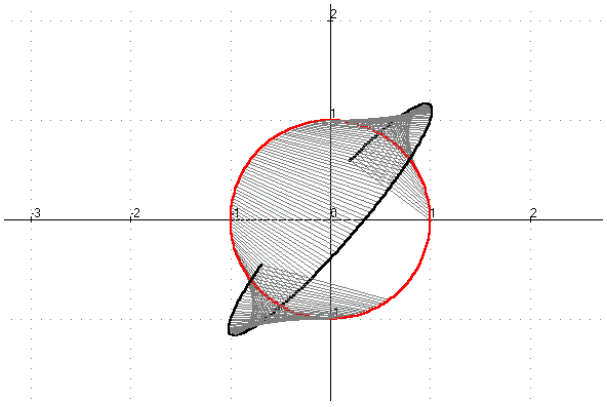
Weitere Ausführungen zum Thema „Selbstständigkeit" bei Schülern findet man z. B. in *[Lehmann, E. (2000)]: Neues Lernen, neue Medien – selbständige Schüler / innen? ZKL-Texte, Münster 2000, Hrsg.: Udo Amelung].*

Projektartige Arbeitsformen

Für die Förderung selbstständiger Arbeit bei Schülern eignen sich in besonderem Maße projektartige Arbeitsformen. Nach einem Projekt mit dem Thema „Besondere (2,2)-Matrizen" (Klasse 11) schreibt ein Schüler (siehe *[Leh94a, S.67]*):

„Die Schüler konnten als Individuen arbeiten und ihre Persönlichkeit entwickeln. Die Schüler konnten frei arbeiten und hatten nicht diesen Druck des Lernens. Der Lehrer gab den einzelnen Gruppen nur Hinweise... Die Teamarbeit spielt bei der Projektarbeit eine große Rolle. Die Schüler mußten aufeinander eingehen, haben gelernt, Formeln, Beweise und Behauptungen zu konstruieren, was nicht immer leicht war. Die Projektarbeit hat viele Vorteile: Das Lernen in kleinen Gruppen fällt leichter, der Unterricht ist lockerer und Vieles mehr. Aber auch die Nachteile sind nicht zu übersehen. Die einzelnen Teams fixieren nur bestimmte Themen und dadurch muss das, was von den anderen Gruppen zusammengestellt wurde, nachgearbeitet und gelernt werden (z. B. für die Klausur)."

„Projektunterricht" wird angesichts seiner Bedeutung u. a. für die Entwicklung der Schülerpersönlichkeit (Selbständigkeit) und der Erarbeitung von Problemlösungen im Team Kapitel 2.1.4 ausführlicher dargestellt. Vorab ein Beispiel: „Vom Einheitskreis zur Ellipse".



$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} 0.5 & 1 \\ 0.6 & 1 \end{pmatrix} \bullet \begin{pmatrix} \cos(t) \\ \sin(t) \end{pmatrix}$$

Abb. 1.4-g: Abbildung des Einheitskreises mit einer Matrix $\begin{pmatrix} 0.5 & 1 \\ 0.6 & 1 \end{pmatrix}$. Das Ergebnis ist eine Ellipse.

Variationen der Matrixelemente oder auch des Ausgangsobjektes kann leicht zu einem Projekt, z. B. im Kurs „Lineare Algebra und Analytische Geometrie" führen!

Selbstverständlich darf im Unterricht nicht nur mit offenen Problemstellungen gearbeitet werden!

> Einseitigkeit in Aufgabenkultur und Unterrichtsform sollte stets vermieden werden. Selbst eine anfangs attraktiv erscheinende Form bleibt für die Schüler nicht auf Dauer attraktiv.

Mögliche Aufgabenstellungen ergeben sich aus Abbildung 1.4-h. Je nach Kenntnisstand der Lerngruppe und auch der Kompetenzen des Lehrers können die genannten Möglichkeiten ausgeschöpft werden.

.

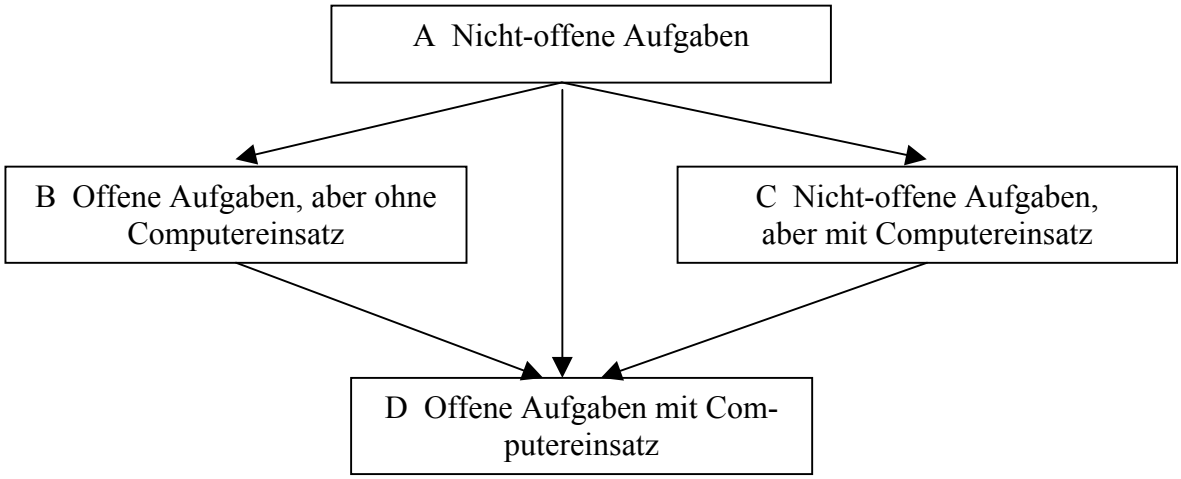


Abb. 1.4-h: Offene Aufgabenstellungen ohne und mit Computereinsatz

Die höchsten Anforderungen an die Lehrerkompetenzen stellen die offenen Aufgaben mit Computereinsatz.

Beispiele:

Zu A: Berechne die Ableitung der Funktion $y = x^3$ durch ausführliche Herleitung mit Hilfe des Grenzwerts des Differenzenquotienten.

Zu B: Ermittle die Ableitung der Funktion $y = x^3$.

Hier sind die Wege frei gestellt (z. B. wie bei A oder grafisch oder mit Hilfe der Produktregel für $y = x \cdot x^2$ oder ...).

Zu C: Ermittle die Ableitung der Funktion $y = x^3$ mit Hilfe deines CAS.

Zu D: Erstelle eine Computer-Animation, die den Vorgang bei der Bildung des Grenzwerts der Differenzenquotientenfunktion zur Funktion $y = x^3$ verdeutlicht.

1.4.2 Aufgabenbeispiele – eine Klausur

Die folgenden Klausuraufgaben aus dem Jahr 1999 zeigen einige der neuen Möglichkeiten. Diese werden im Aufgabentext kurz kommentiert (*siehe kursiven Text in den Rechtecken*).

2. Klausur im Leistungskurs Mathematik
Leh-MA1, 6.12.99, 3 Schulstunden

Neue Aufgabenkultur!

1) etwa 35´
Führen Sie die folgenden Flächeninhalts-Berechnungen durch.
Hinweis: Es sind jeweils passende Skizzen anzufertigen. Die rechnerischen Ansätze sind zu begründen. Die TI-92-Eingaben und -Ausgaben sind zu notieren und ggf. zu kommentieren.

a) Für die Flächen zwischen der cos-Kurve und der sin-Kurve im Bereich [0, π/2].

b) f(x) = 0.2*(x^3 - 3x^2 - 6x + 8) , g(x) = 0

Ansätze finden, Rechnungen mit dem Taschencomputer TI-92.

2) etwa 60´
Das folgende TI-92-Bild zeigt eine Anwendung der so genannten Trapezregel zur angenäherten Flächenberechnung.

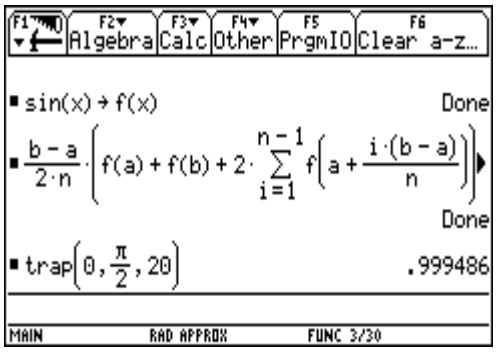
a) Erläutern Sie die in dem definierten Baustein trap(a,b,n) vorkommenden Variablen und seine Anwendung.

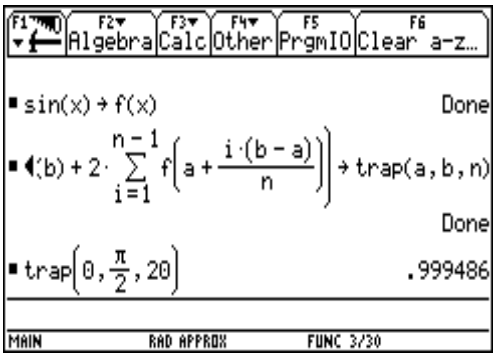
b) Ermitteln Sie weitere Werte für n = 5, 10, 15, 25, 30. Auswertung?

c) Vergleichen Sie mit dem Wert, den das dazugehörige Integral liefert.

d) Begründen Sie die Formel durch eine passende Zeichnung.

Einen Baustein erläutern und den TI-92 benutzen. Eine Formel verstehen.



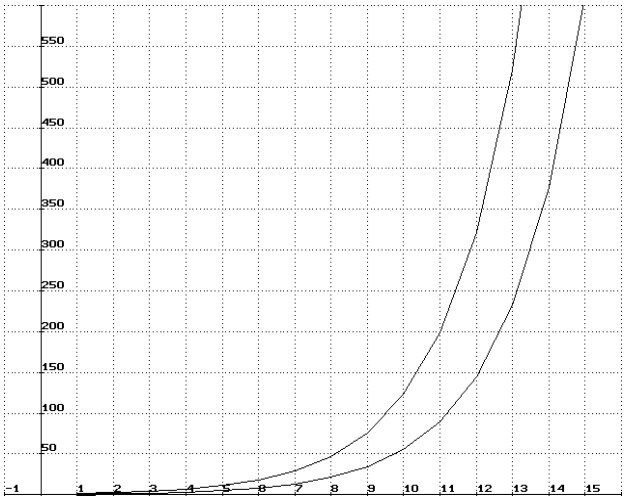


3) etwa 40´ (Hinweis: PLOT11 ist ein Funktionenplotter.)

Nach dem Starten der PLOT11-Datei FOLGE1.HPL sehen Sie in der F1-Maske mehrere Einträge. Drücken Sie F3 zur Ausgabe einer Wertetafel bzw. F4 für die Grafik (die Ausgaben sehen Sie auch unten).

Die F1-Maske:

f 1: {n=1:1:{n=2:1:f1(n-1)+f1(n-2)}}
f 2: (0.5*(1+sqrt(5)))^n
f 3: (0.5*(1-sqrt(5)))^n
f 4: 1/sqrt(5)*(f2(n)-f3(n))



x,t	f1	f2	f3	f4
1.0000	1.0000	1.6180	-0.6180	1.0000
2.0000	1.0000	2.6180	0.3820	1.0000
3.0000	2.0000	4.2361	-0.2361	2.0000
4.0000	3.0000	6.8541	0.1459	3.0000
5.0000	5.0000	11.0902	-0.0902	5.0000
6.0000	8.0000	17.9443	0.0557	8.0000
7.0000	13.0000	29.0344	-0.0344	13.0000
8.0000	21.0000	46.9787	0.0213	21.0000
9.0000	34.0000	76.0132	-0.0132	34.0000
10.0000	55.0000	122.9919	0.0081	55.0000
11.0000	89.0000	199.0050	-0.0050	89.0000
12.0000	144.0000	321.9969	0.0031	144.0000
13.0000	233.0000	521.0019	-0.0019	233.0000
14.0000	377.0000	842.9988	0.0012	377.0000
15.0000	610.0000	1364.0007	-0.0007	610.0000

a) Was fällt Ihnen so alles auf? (1., 2., ...)

b) Äußern Sie sich zur Art der Darstellungen bei f1 und f4 in der F1-Maske.

c) Betrachten Sie die Funktion $f(x) = (0.5*(1+sqrt(5)))^x$, fertigen Sie eine Skizze des Graphen an, x aus [-10, 10].

d) In welchem Punkt P hat der Graph die Steigung 1? Wie heißt die Gleichung der Tangente durch P?
Hinweis: Zu c) und d) : Dokumentation der TI-92-Arbeit

Erläuterung einer Zeichnung in Verbindung mit der zugehörigen Wertetafel und von Folgen- und Funktionstermen (Software: ANIMATO). Verschiedene Darstellungen der Fibonacci-Folge. Handskizze des Grafen von f. Tangentengleichung berechnen – Benutzung des Taschencomputers.

1.5 Szenarien und Ziele für einen modernen Mathematikunterricht

Wie jeder Unterricht entwickelt sich auch der Mathematikunterricht aus einer Fülle von Bedingungen. Die Komplexität konzeptioneller Überlegungen zu diesem Bedingungsfeld wird an den nun folgenden Überblicksdarstellungen deutlich.

Abbildung 1.5-a zeigt zahlreiche zu beachtende Aspekte:
- Fachinhalte und Fachmethoden,
- Aufgabenkultur,
- Unterrichtskultur,
- Kompetenzen von Lehrern und Schülern,
- Medien und ihre Einsatzformen (z. B. als Demonstrationsmedium),
- sonstige Bedingungen (z. B. organisatorische).

Unter Berücksichtigung dieser Aspekte entstehen diverse Unterrichtsszenarien.

Ein anspruchsvolles Szenarium ergibt sich beispielsweise aus der Vernetzung von mathematischen Inhalten und Methoden
- mit informatischen Inhalten und Methoden,
- in Form eines Projektunterrichts,
- mit komplexen und offenen Problemstellungen und
- unter Verwendung neuer Medien, insbesondere im Rahmen eines experimentellen Computereinsatzes.

Die Vielfalt der Szenarien und der sich ergebenden Konzepte kann in der vorliegenden Arbeit nur für einige Vernetzungen verdeutlicht werden.

> Dabei wird der Schwerpunkt auf der Konzeption eines Mathematikunterrichts liegen, der informatische Methoden und Inhalte einbezieht, dabei selbstverständlich den Computer einsetzt und in der Regel von offenen Aufgabenstellungen und offenen Unterrichtsformen ausgeht.

Für derartige Konstellationen werden viele Beispiele aus der Unterrichtspraxis und für die Unterrichtspraxis vorgestellt. Hierfür ergeben sich zahlreiche Hinweise aus Abbildung 1.5-b. Diese Abbildung zeigt den großen Vorrat an Verknüpfungsmöglichkeiten für konzeptionelle Überlegungen zur Berücksichtigung informatischer Inhalte und Methoden im Mathematikunterricht. Viele der Aspekte werden sich dann in den folgenden Kapiteln wiederfinden.

Beide Abbildungen bieten somit einen Überblick über zu berücksichtigende Themen und mögliche inhaltliche Verknüpfungen. Sie bilden damit auch die Grundlage für die nun folgenden konkreten und detaillierten Ausführungen.

.

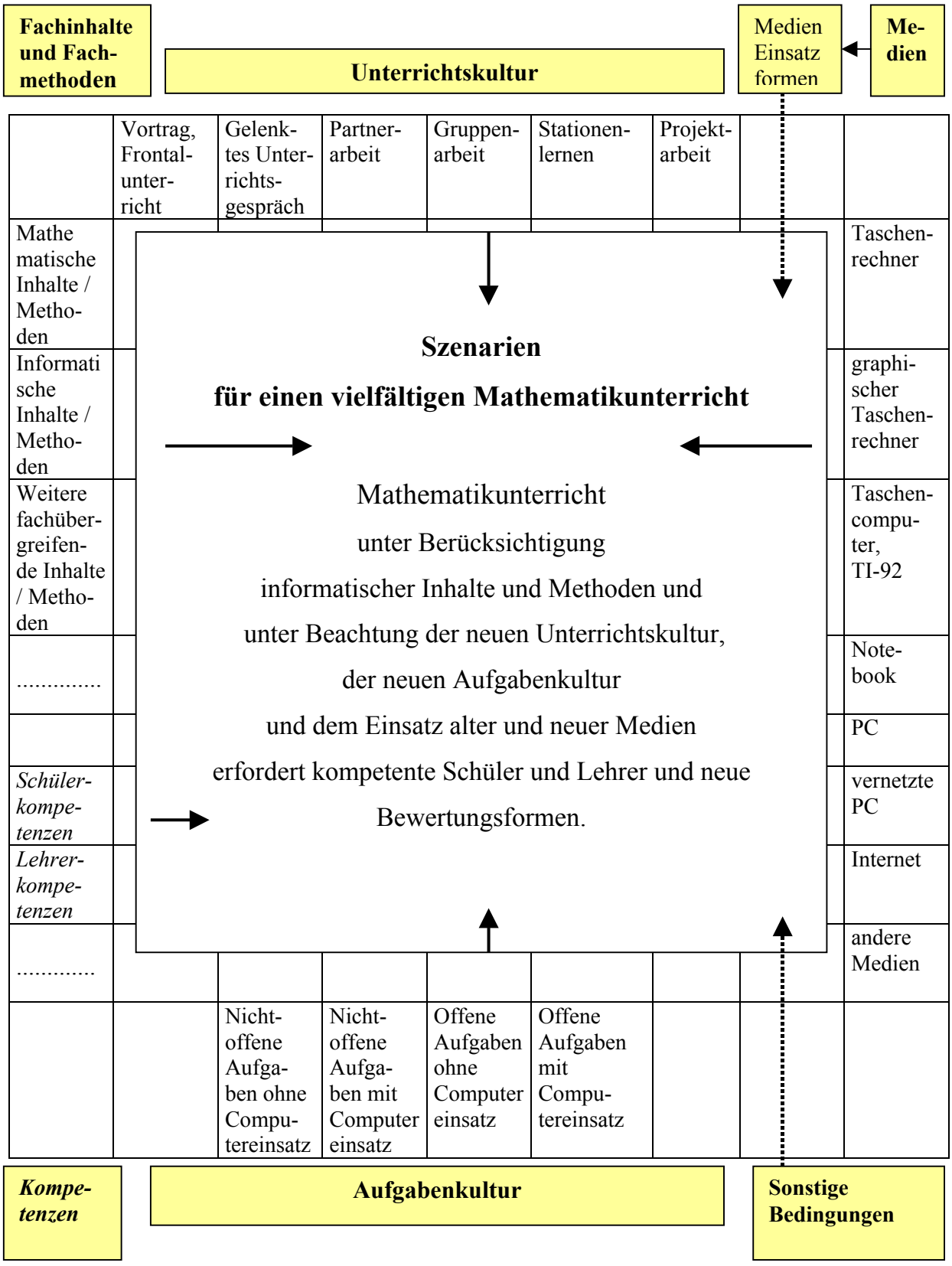


Abb. 1.5-a : Mathematikunterricht – wer die Wahl hat, hat die Qual

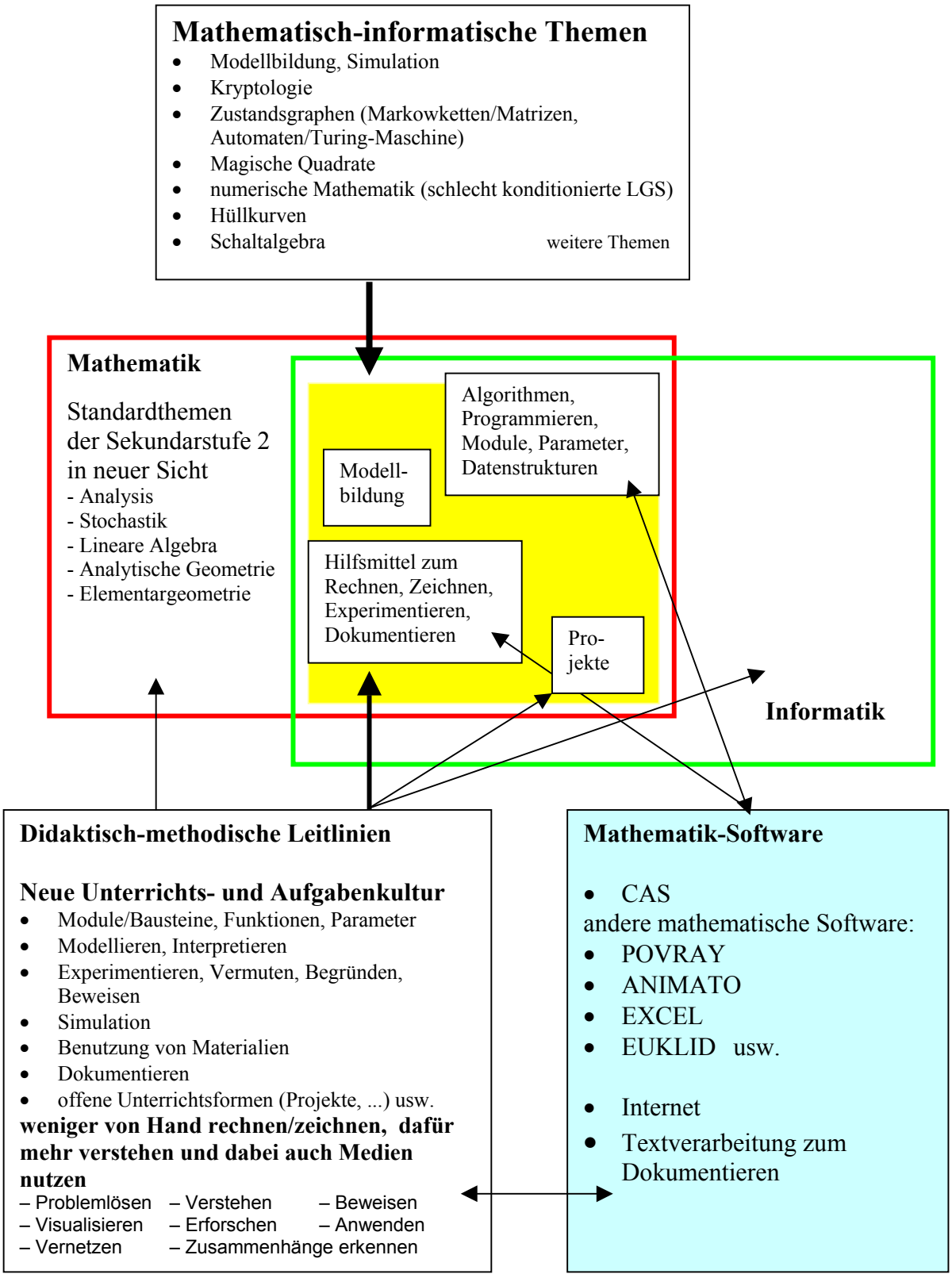
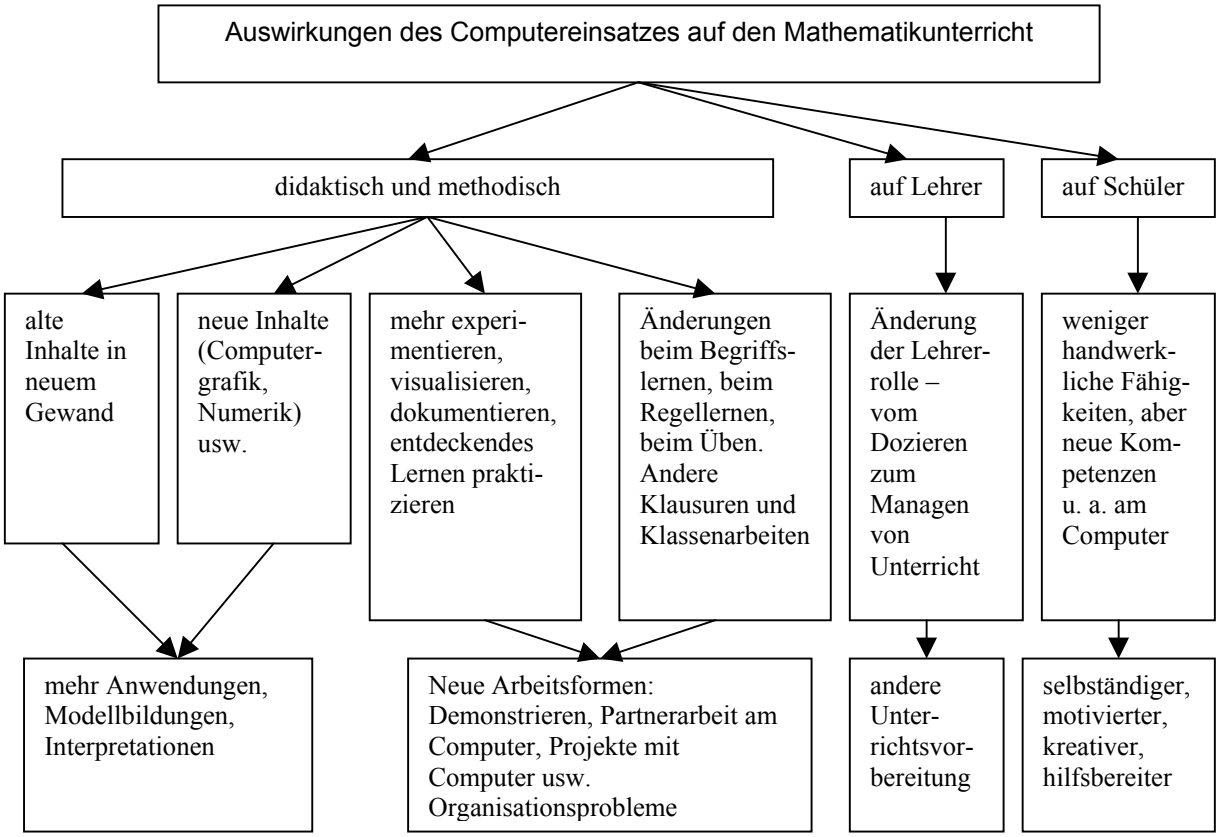


Abb. 1.5-b: Überblicksdarstellung zur Konzeptentwicklung

Die bisherigen Erfahrungen,

- dokumentiert in vielen einschlägigen Büchern und Fachzeitschriften,
- erprobt im Unterricht,
- diskutiert in Vorträgen und Fortbildungsveranstaltungen,

haben gezeigt, dass die Auswirkungen des Computereinsatzes auf den Mathematikunterricht erheblich sind. Abbildung 1.5-c vermittelt, in welchem Umfang das geschieht.



Weniger routinemäßige Handrechnungen und Handzeichnungen, dafür mehr verstehen!

Computereinsatz im Mathematikunterricht bringt erhebliche didaktisch-methodische Bereicherungen und führt zu einer neuen Lehrer- und Schülerrolle.

Computereinsatz muss aber immer daran gemessen werden, ob er in der aktuellen Situation zu einer Qualitätssteigerung des Unterrichts führt.

Abb. 1.5-c: Auswirkungen des Computereinsatzes im Mathematikunterricht

Der Verlust einiger bisheriger Unterrichtsgewohnheiten

Angesichts der Fülle der sich aus obigen Abbildungen ergebenden Möglichkeiten für die Gestaltung eines modernen Mathematikunterrichts sollten Hindernisse und Hemmnisse nicht vergessen werden! Sie betreffen in starkem Maße den unterrichtenden Lehrer, aber auch die Schüler.

- Der Lehrer kann nicht mehr wie bisher unterrichten!
- Bislang gewohnte und geübte grundlegende Fertigkeiten gehen teilweise verloren.
- Es besteht die Gefahr, die mathematische Struktur der bisherigen Lehrgänge zu verlieren.
- Auch die bisher gewohnten Unterrichtsziele werden sich zugunsten anderer/weiterer Ziele ändern.
- Die Komplexität des Unterrichts wird sich erheblich vergrößern.
- Form und Inhalt von Lernzielüberprüfungen ändern sich zugunsten einer größeren Vielfalt.
- Der Lehrer muss sich und seinen Schülern neue Arbeitsweisen und ein neues Rollenverständnis vermitteln.
- Gewohnte Formen der Unterrichtsvorbereitung können nicht mehr angewandt werden.

Damit erlebt der Mathematiklehrer seit einiger Zeit dramatische Veränderungen seines Arbeitsfeldes, Veränderungen, die sich angesichts der verbreiteten Verfügbarkeit von Computern wohl nicht aufhalten lassen.

So bleibt den Lehrern kaum etwas übrig, als sich auf die neuen Gegebenheiten einzustellen durch

- aktive Teilnahme an Fortbildungsveranstaltungen,
- Kooperation im Fachbereich Mathematik/Informatik,
- Nutzen der vielfältigen konkreten Unterrichtsbeschreibungen in Büchern und Fachzeitschriften,
- Nutzen der Angebote des Internet,
- Umstellen ihrer Lehrerrolle,
- Anerkennung und Nutzung der Schülerkompetenzen, insbesondere über neue Medien,
- aber auch Vermeiden von Gefahren der neuen Entwicklungen, wie beispielsweise die einseitige Betonung nur neuerer Unterrichts- oder Aufgabenformen und damit etwa Vernachlässigung des Übens, Faktenwissens und mathematischer Hintergründe.

Viele der genannten Aspekte wirken auch auf die Schüler und führen bei diesen zu veränderten Vorgehens- und Verhaltensweisen.

Abbildung 1.5-d bringt ein Beispiel für ein Unterrichtsszenarium aus der Erfahrung in einem Leistungskurs Mathematik beim Thema „Lineare Algebra" (siehe auch Kapitel 3.5).

.

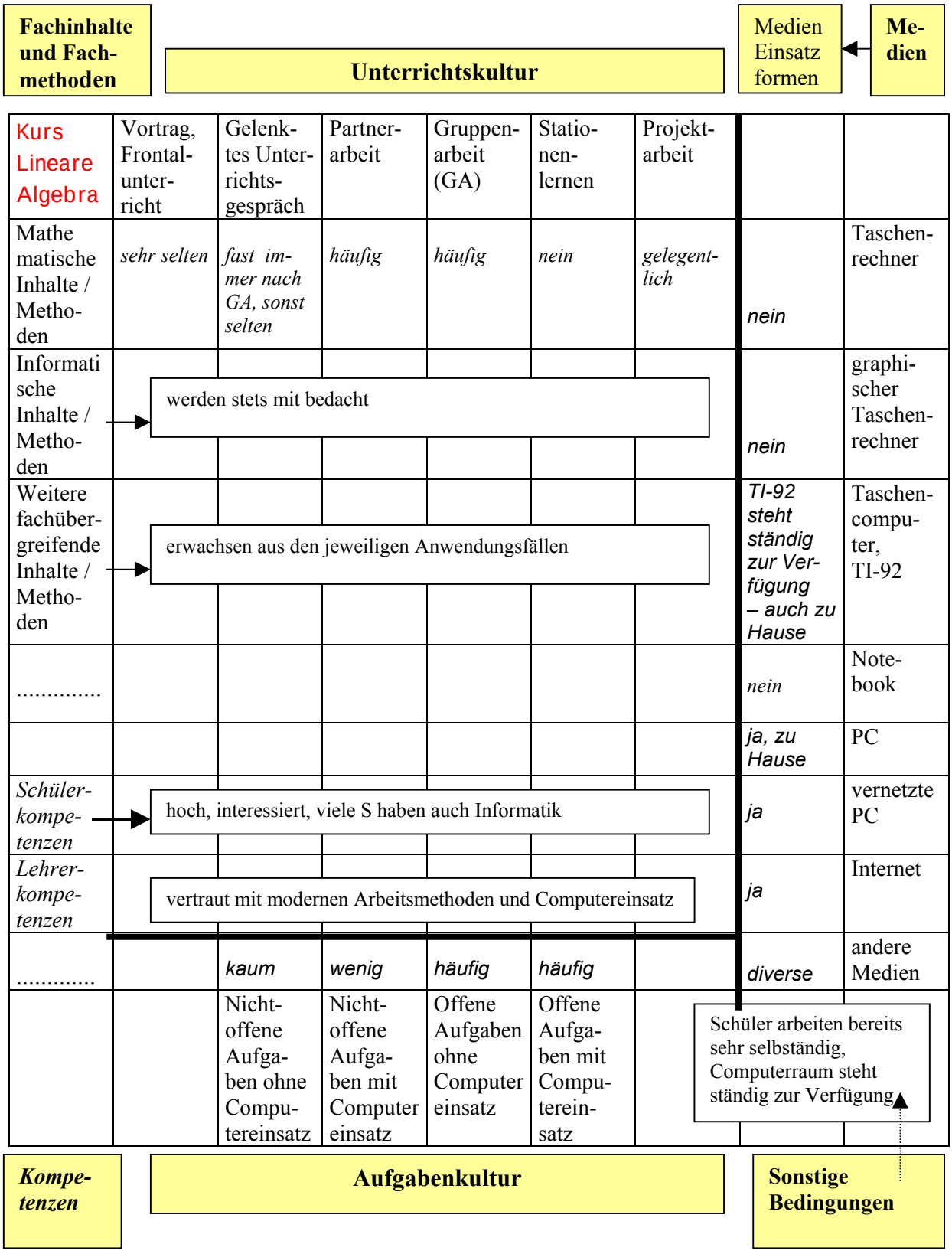


Abb. 1.5-d : Mathematikunterricht – Unterrichtsszenarium in einem Leistungskurs Lineare Algebra

2. Informatische Methoden und Inhalte und ihre Anwendungsmöglichkeiten im Mathematikunterricht

„Der Mathematikunterricht muss sich dieser Herausforderung durch die Informatik stellen, und er könnte davon durchaus profitieren – durch schrittweise Öffnung gegenüber eines prozeßorientierten Sichtweise, insbesondere im Bereich Modellbildung und Simulation, durch behutsame Übernahme von projektorientierten Arbeitsweisen, durch wohlbegründete Einbeziehung von geeigneten Themen der Informatik" *[Her93]*.

In Kapitel 1 wurden verschiedene Aspekte der „Begegnungen zwischen Mathematik und Informatik" an der allgemeinbildenden Schule (Gymnasium) dargestellt, die zu ersten konzeptionellen Überlegungen zur Einbeziehung informatischer Methoden und Inhalte im Mathematikunterricht führten.

In diesem Kapitel folgt nun eine systematischere Untersuchung der Anwendungsmöglichkeiten informatischer Methoden und Inhalte im Mathematikunterricht. Dabei wird insbesondere der Frage nachgegangen, wie das Schulfach Mathematik, das ja in dieser Arbeit im Mittelpunkt steht, von der Informatik profitieren kann.

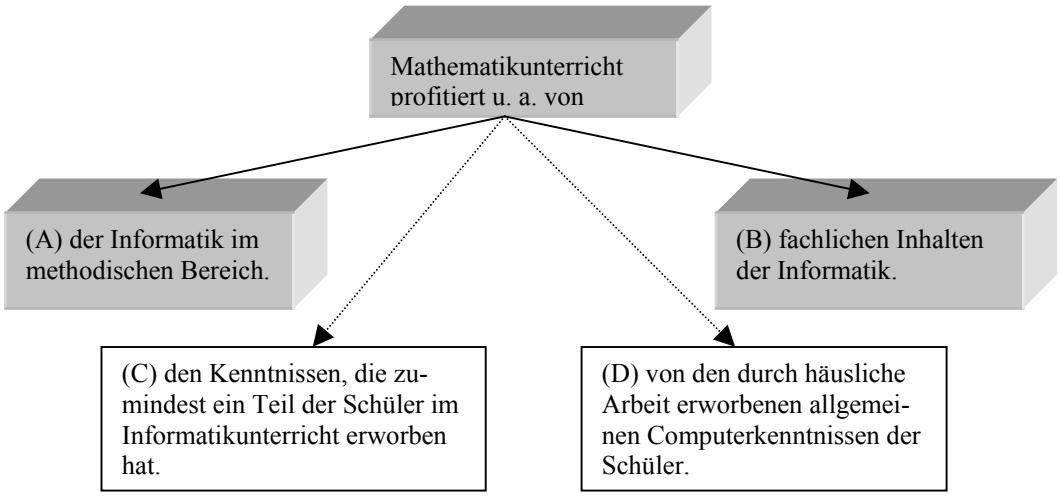


Abb. 2-a

Bei der Planung und Durchführung von Mathematikunterricht sollten alle vier in Abb. 2-a genannten Aspekte berücksichtigt werden. (C) und (D) sind abhängig von der jeweils vorliegenden Lerngruppensituation und werden leider im konkreten Unterricht bislang in der Regel übersehen, obgleich sich bei Berücksichtigung dieser Aspekte wesentliche Bereicherungen des Unterrichts ergeben können. Für die vorliegende Arbeit interessieren diese Aspekte allerdings nur beiläufig. Wichtig sind hier die Aspekte (A) und (B).

2.1 Überlegungen zur Nutzung von Methoden der Informatik im Mathematikunterricht

2.1.1 Komplexe Systeme – Zerlegung in Teilsysteme

Fundamentale Ideen
Die Probleme in Zusammenhang mit der Stofffülle haben im Mathematikunterricht schon vor Jahren unter anderem dazu geführt, sich näher mit den fundamentalen Ideen des Faches zu beschäftigen. So wurde in vielen Lehrplänen bewusst das Spiralprinzip berücksichtigt, leicht ablesbar beispielsweise bei der Behandlung linearer Gleichungssysteme in verschiedenen Klassenstufen oder beim Funktionsbegriff und den diversen Funktionsklassen, die im Verlauf des Mathematikunterrichts behandelt werden.

Die Informatik muss sich mit ähnlichen Problemen auseinandersetzen, allerdings in noch stärkerem Maße als die Mathematik, denn in kurzer Zeit haben sich aufgrund des breiten Anwendungsbereichs der Informatik viele neue Gebiete entwickelt. Die Innovationszyklen sind außerordentlich kurz.

> Fundamentale Ideen sind grundlegende Prinzipien, Denkweisen und Methoden.

[A. Schwill, Fundamentale Ideen der Informatik, ZDM 1993, Heft 1, S. 20 f.].

Nach *Schwill* sind fundamentale Ideen

- in verschiedenen Bereichen des Fachgebietes erkennbar (Horizontalkriterium),
- in verschiedenen Altersstufen aufzeigbar und vermittelbar (Vertikalkriterium),
- längerfristig relevant (Zeitkriterium) und
- stehen in Beziehung zu Sprache und zum Alltag (Praxiskriterium).

> Eine sehr weitreichende fundamentale Idee der Informatik ist die der komplexen Systeme mit den Aspekten
> - Benutzung eines vorhandenen Systems,
> - Analyse des Systems,
> - Wartung des Systems,
> - Konstruktion eines (neuen) Systems.

Diese Idee wurde Grundlage des Berliner Informatik Rahmenplans von 1993.
Senatsverwaltung für Schule, Berufsbildung und Sport: Vorläufiger Rahmenplan für Unterricht und Erziehung in der Berliner Schule, gymnasiale Oberstufe, Fach Informatik, gültig ab Schuljahr 1993/94. – Anmerkung: Es handelt sich um den 4. Informatiklehrplan in Berlin.

2.1.2 Modellbildung bei komplexen Systemen

In den Naturwissenschaften und der Technik dienen häufig Experimente dazu, Informationen zur Bestätigung oder Widerlegung von Hypothesen zu gewinnen. Modelle werden dagegen oft zur Lösung von Aufgaben eingesetzt, deren Durchführung am Original selbst nicht möglich oder zu aufwendig ist. Auch der Erstellung des auf Seite 45 dargestellten Systems LOTTO ging eine Modellbildung voraus.

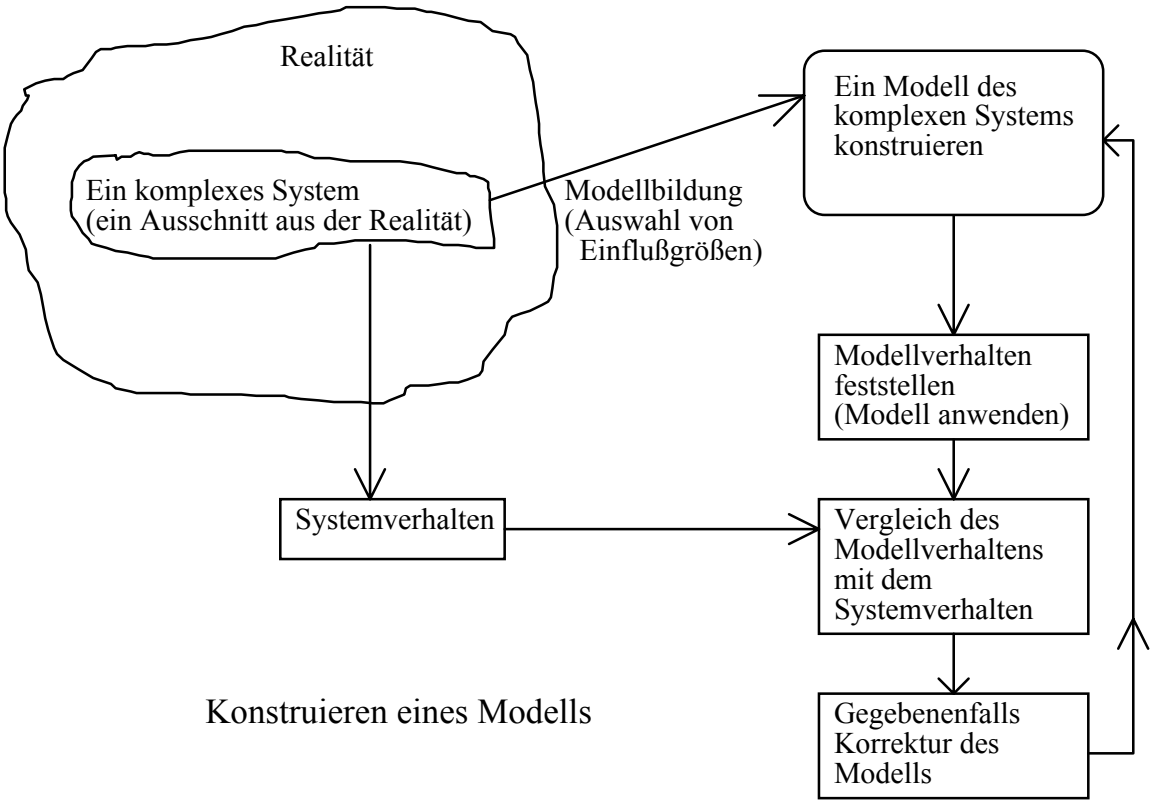


Abb. 2.1.2-a: Modelle konstruieren

Das folgende Beispiel (Abb. 2.1.2-b) zeigt, wie man im Informatikunterricht oder im Unterricht in informationstechnischer Grundbildung komplexe Systeme einführen und in Ausschnitten bearbeiten kann. Es wird deutlich, wie das Gesamtsystem in Teilsysteme und damit in Teilprobleme zerlegt werden kann, die dann möglicherweise getrennt einer Lösung zugeführt werden können. Das seinerzeit von einem Informatikkurs programmierte Teilsystem LOTTO (siehe Abb. 2.1.2-b) bearbeitete einen Teilbereich des umfassenden komplexen Systems. Es stellt hier einen Ausschnitt aus der Realität dar.

Die intensive Beschäftigung mit Softwaresystemen auf verschiedenen Betrachtungsebenen und Handlungsebenen führt zu wesentlichen informatischen Erkenntnissen! Man wird aber auch sehen, dass die Mathematik von den genannten Gedankengängen profitieren kann.

In diesem Fall sind die informatischen Erkenntnisse u. a. die folgenden:

(a) Beschäftigung mit dem Anwendungsbereich (ggf. auch mit gesellschaftlichen Aspekten der Datenverarbeitung),
(b) Kennenlernen praxisrelevanter spezifischer Arbeitsmethoden,

(c) das Erkennen von System-Bausteinen und das Verstehen von Benutzeroberflächen,
(d) das Erkennen von Schnittstellen,
(e) Verwenden von Hilfsprogrammen zur Erstellung von Oberflächen (Schnittstellen)
(f) Wiederverwenden vorgefertigter Bausteine, Konstruktion eigener Bausteine

Lebenslauf eines Lottoscheins

(aus einer ITG-Unterrichtseinheit "Wettleidenschaft" in Klasse 9, ITG: Informationstechnische Grundbildung, 1994)

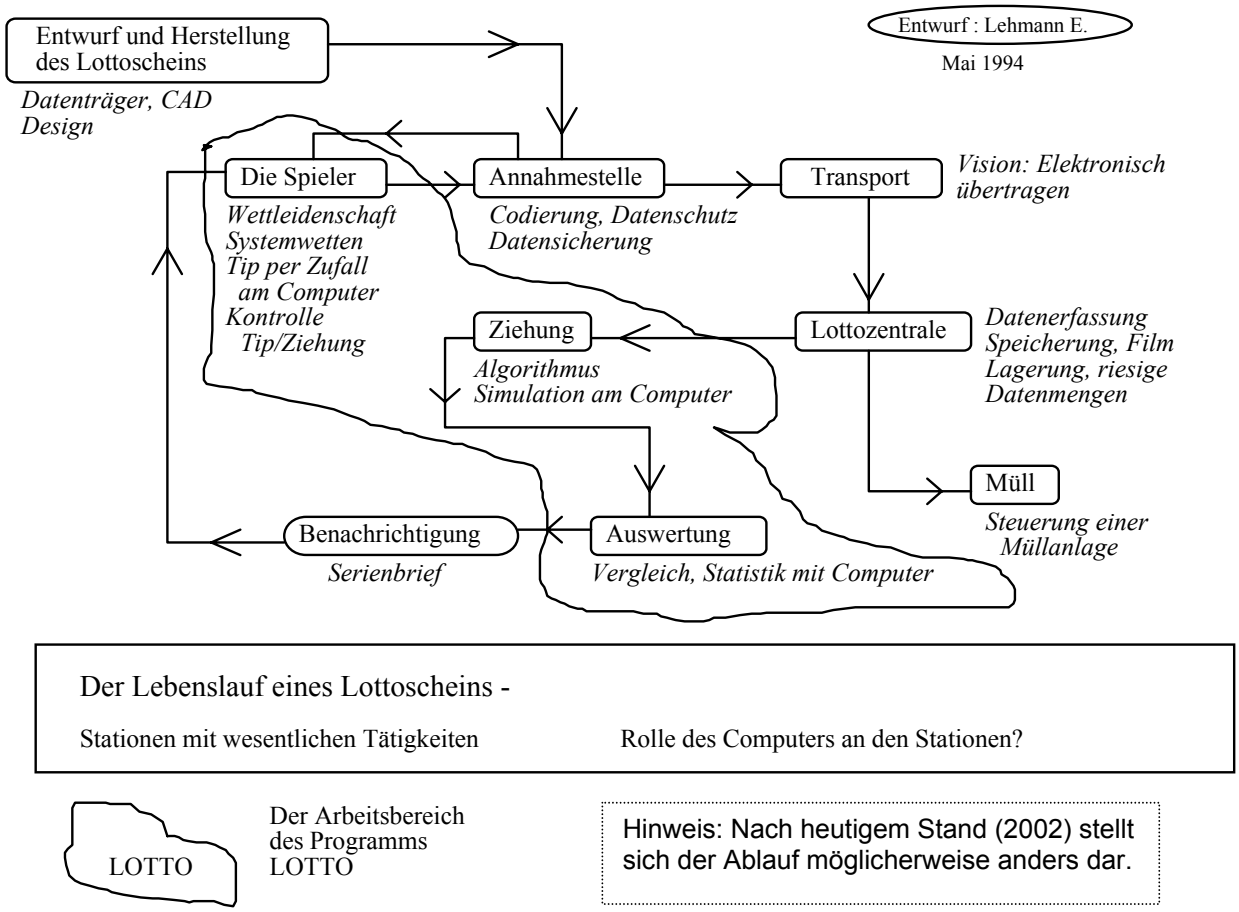


Abb.2.1.2-b: Das Zahlenlotto als Beispiel eines komplexen Systems

Ausschnitte aus der Realität beschreiben

Abbildung 2.1.2-b zeigt eine Gedankensammlung, die das komplexe System „ZAHLEN-LOTTO" beschreibt. Mit der Aufgabe *Erstelle eine Abbildung, die den „Lebenslauf eines Lottoscheins" beschreibt*, wurde es von Schülern erarbeitet, die außerdem den Auftrag hatten, nach Einsatzmöglichkeiten des Computers innerhalb dieses Systems zu suchen. So erklären sich die kursiv gedruckten Stichwörter. Für einen Ausschnitt aus dem System steht ein in Objekt-Pascal (Delphi) programmiertes (und auch schon recht komplexes) Teilsystem LOTTO zur Verfügung, das die Bereiche *Tippen, Ziehung, Auswertung* realisiert. Es fasst mehrere Komponenten zu einem Ganzen zusammen. Die Komponenten bearbeiten wohlbestimmte Aufgaben, die u.a. in der Oberfläche des Programmsystems deutlich werden.

Der Zweck eines Softwareprodukts besteht in seiner Anwendung für von ihm verlangte Aufgaben. Schon die Anwendung und die Dokumentation verraten uns viel von dem **Leistungsumfang** des Systems. Damit wird bereits der Anfang einer Systemanalyse vollzogen. Durch

das **Hineinsehen in das System** (aus welchen Bausteinen besteht es, wie sind die Datenstrukturen, wie wurden die Teilprobleme programmiert ...?) kann die Analyse fortgesetzt werden. Diese Kenntnisse benötigt man, um z. B. das System an möglicherweise veränderte Gegebenheiten anzupassen, es also zu warten.

Das Lotto-System aus mathematischer Sicht

In dem Überblick von Abb. 2.1.2-b und in der folgenden Abbildung 2.1.2-c sind auch diverse mathematische Probleme enthalten. Es sind u. a.:

- Algorithmen zur Simulation der Lottoziehung, u. a. Erzeugung von Lotto-Zufallszahlen
- Codierungsprobleme
- Statistische Fragestellungen usw.

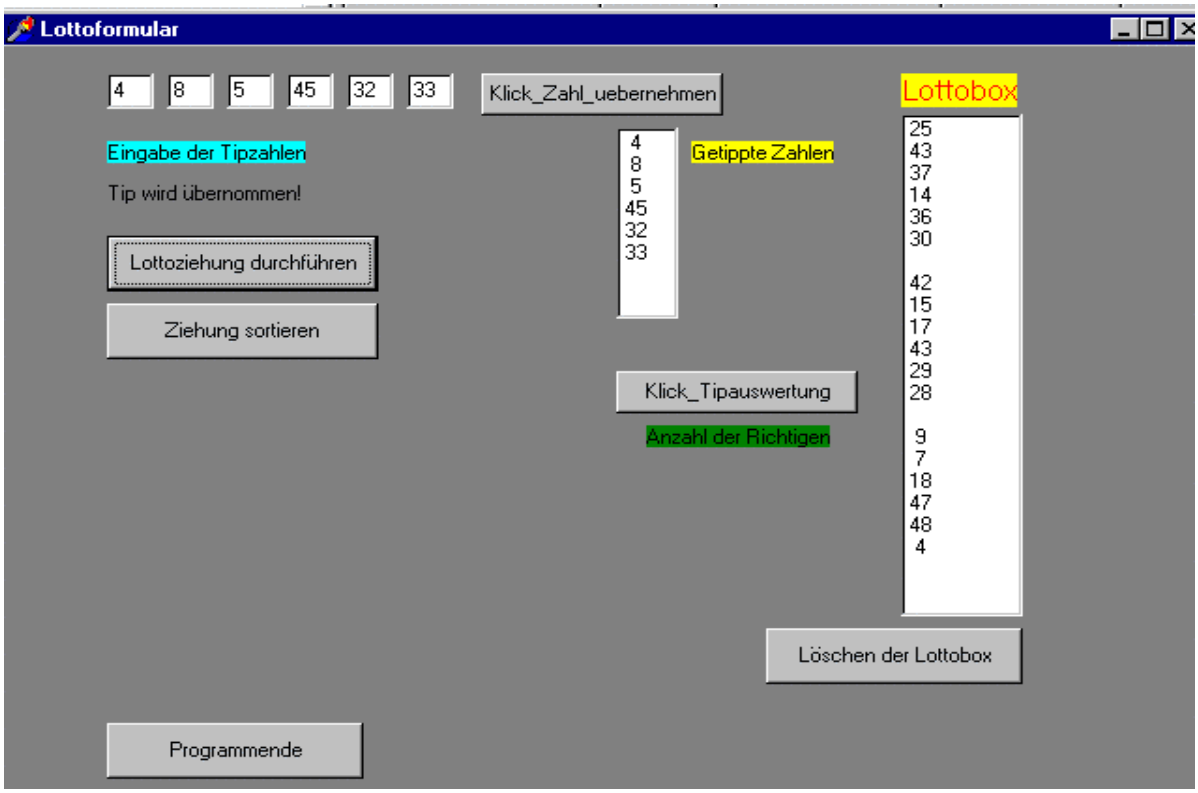


Abb. 2.1.2-c Die Oberfläche eines DELPHI-Programms zur Simulation von Lottoziehungen und Auswertungen

Die Oberfläche des in der Programmierumgebung DELPHI mit der objektorientierten Programmiersprache „Objekt-Pascal" erstellten Simulationsprogramms lässt erkennen, wie die Zerlegung des (für den Schulunterricht) komplexen Lottosystems für die Entwurfsarbeiten ausgesehen haben könnte.

Eingabe der Tippzahlen: Der Kunde gibt seinen Tipp ab; das könnte z. B. ein 10-Wochen-Tippschein sein. Die getippten Zahlen erscheinen in der Spalte „getippte Zahlen".

Lottoziehung durchführen: Zufällige Erzeugung von Ziehungen. Diese werden in der Lottobox gespeichert.

Klick_Tippauswertung: Der Kundentipp wird mit den durchgeführten Ziehungen verglichen. Die Anzahl der Richtigen wird bekanntgegeben.

Ziehung sortieren: Das erzeugte Ziehungstupel wird aufsteigend sortiert.

Löschen der Lottobox: Das ist wichtig für einen Neustart der Simulation.

Programmende.

Abb. 2.1.2-d

Hinter diesen Modulen stehen mehrere Algorithmen, die sich teilweise auch mit einem Computeralgebrasystem realisieren lassen So werden in Abbildung 2.1.2-e Sechs-Tupel von Zufallszahlen erzeugt, so wie es auch im Lottosystem benötigt wird. Dabei muss aber noch bedacht werden, dass im Lottosystem innerhalb eines 6-Tupels keine Zahlen doppelt vorkommen.

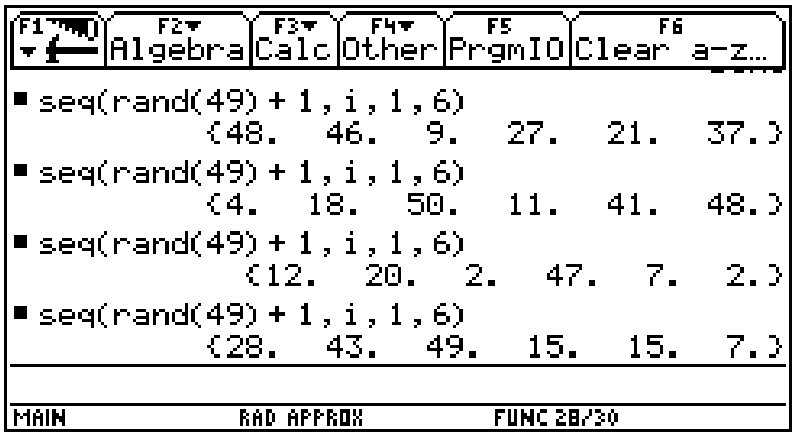


Abb. 2.1.2-e: Erzeugung von 6 Lottozahlen mit dem CAS des TI-92

Wir können festhalten:

> Eine mathematisch orientierte Sicht auf dafür geeignete komplexe informatische Systeme kann auch für den Mathematikunterricht auf interessante Projekte führen!

Außerdem erkennt man:

Softwareprodukte können als komplexe Systeme aufgefasst werden. Dieser Ansatz wird in Kapitel 2.1.3 näher untersucht und für Unterrichtszwecke aufbereitet.

Lernen von der Informatik

Das Verhältnis zwischen Mathematik und Informatik ist für den Schulunterricht in beide Richtungen von erheblicher Bedeutung (siehe z. B. *[Her93], [Leh94], Schw94], [Wei97]*). Das immer noch um breite Anerkennung und Verbindlichkeit ringende Schulfach Informatik enthält einerseits viele mathematische Grundlagen, obwohl diese im Unterricht oft zurückgedrängt werden, kann aber andererseits insbesondere im methodischen Bereich wertvolle Anregungen für den Mathematikunterricht liefern. Zu dem letztgenannten Aspekt folgen nun einige Ausführungen, die auch in Zusammenhang mit der Verwendung von Computeralgebrasystemen stehen.

Zunächst werden drei Leitlinien der Informatik genannt, die für das in Kapitel 2.1.5 näher dargestellte Bausteinkonzept und damit für den Mathematikunterricht von großer Bedeutung sind:

- Zerlegen eines komplexen Problems in Teilprobleme (Modularisierung, Modulkonzept)
- das Prozedurkonzept
- Bearbeitung der Teilprobleme und Zusammenfügen zur Gesamtlösung

Wichtige Aspekte für das Bausteinprinzip, siehe Kapitel 2.1.5!

Lösungsvielfalt

Die Informatik macht uns (und den Schülern) vor, dass sich Probleme häufig auf vielfältige Weise lösen lassen.

Das ist ein Aspekt, der gerade heute von einem zeitgemäßen Mathematikunterricht gefordert wird. Hierzu noch einmal das Lottoproblem mit einer zweiten Lösung:

```
                    H A U P T M E N U E


          1 Tippen, von Hand oder Computer
          2 Ziehung u. Vergleich mit Tip·
          3 Statistik·····················
          4 Anleitung, Hilfen·····
          5 Abbruch (Quit)···············

          Bitte mit dem Cursor auswählen!
```

An dem Hauptmenü des Lottosystems erkennen wir die hauptsächlichen Funktionen „Tippen, Ziehung, Auswertung (Statistik)". Nach Aufruf von Option 1 können wir u. a. Tipps von Hand eingeben:

```
Eine Option mit der Cursor (auf/ab)-Taste wählen
und die RETURN-Taste drücken!

HILFE    COMPUTERTIPS    HANDEINGABE     TIPS ANSEHEN    ZUM HAUPTMENÜ
```

L O T T O
6 aus 49

L O T T O S C H E I N

```
 1  2  3  4  5  6  7  8  9 10

11 12 13 14 15 16 17 18 19 20

21 22 23 24 25 26 27 28 29 30

31 32 33 34 35 36 37 38 39 40

41 42 43 44 45 46 47 48 49
```

```
Beenden mit "*" !
Name :
```

Ausgehend vom Hauptmenü können wir auch das Hilfemenü aktivieren.

```
                H I L F E - M E N U E

          1   Allgemeine Lottoregeln
          2   Tippen
          3   Ziehung
          4   Statistik / Auswertung
          5   Hauptmenue

        Geben Sie Ihre Wahl ein :    ?
```

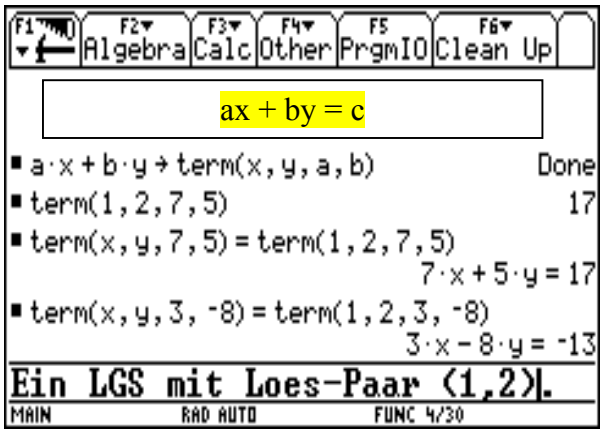
Offenbar liegt hier ein anderer Ansatz vor als bei der zuerst dargestellten Lösung.

Unterrichtserfahrungen zeigen, dass die Schüler im Informatikunterricht sehr viel leichter verschiedene Lösungen eines Problems erarbeiten können, als das im Mathematikunterricht der Fall ist. Das gilt insbesondere für Programmierprobleme. Aber auch im Mathematikunterricht lässt sich der Aspekt der Lösungsvielfalt verfolgen. Im Rahmen des bundesweiten SINUS-Projekts liegen dafür zahlreiche Beispiele vor. Hier folgt ein Beispiel aus einer meiner Lehrerfortbildungsveranstaltungen (Bremen 2002).

Arbeiten mit Parametern in der Sekundarstufe 1

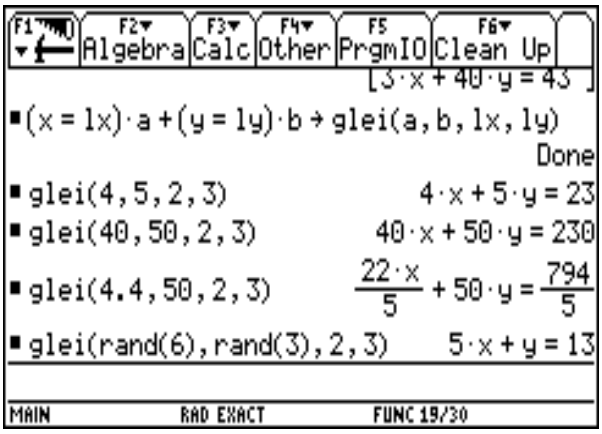
Wie baut der Lehrer lineare Gleichungssysteme (LGS) mit so schönen „glatten" Lösungen?

Lösung 1



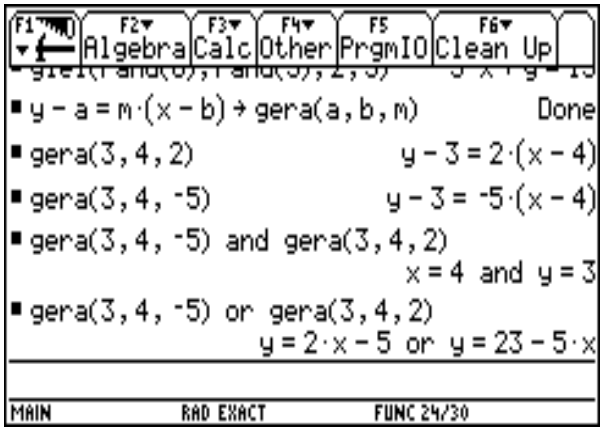
Lösungsidee:
Linke und rechte Seite als Terme

Lösung 2



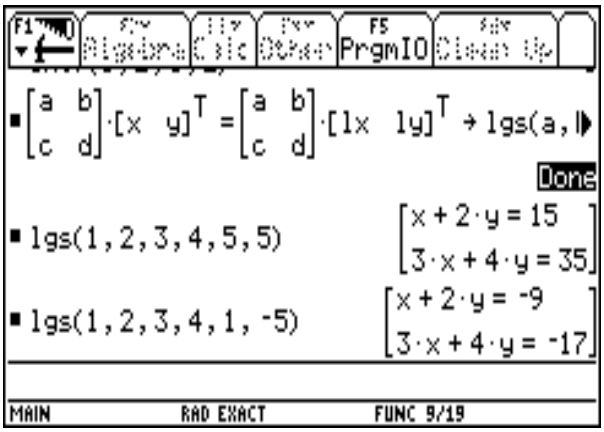
Lösungsidee:
Operationen auf den Gleichungen x=lx und y=ly

Lösung 3



Lösungsidee:
Über Punkt-Steigungsform/geom. Deutung, das entspricht der Fragestellung:
Zeichne möglichst viele Geraden durch den Punkt P(a, b).

Lösung 4



Lösungsidee:
Linke Seite des LGS = Rechte Seite des LGS
..... → lgs(a,b,c,d,lx,ly)

Mit den obigen Lösungsideen lassen sich leicht LGS mit 2 Variablen und auch mit 3 oder 4 oder mehr Gleichungen erzeugen. Für die obige Aufgabenstellung wurde hier jeweils das CAS des TI-92-Plus benutzt.

2.1.3 Sichtweisen auf Softwareprodukte

Die informatische Sicht

Abbildung 2.1.3-a beschreibt ein gegebenes Softwareprodukt aus der Sicht der Informatik, aufgefasst als komplexes System:

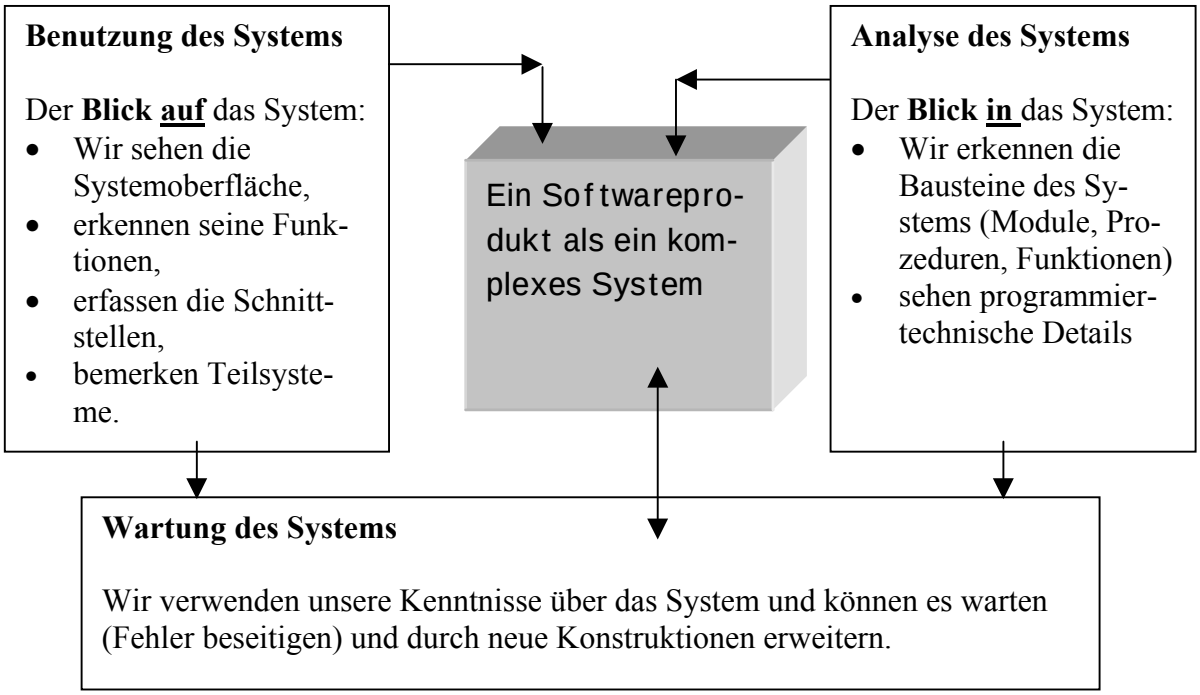


Abb. 2.1.3-a: Sichtweise der Informatik auf ein Softwareprodukt

Die mathematische Sicht

Aus der Sicht der Mathematik stellen sich Softwareprodukte nicht so differenziert dar. Hier überwiegt eindeutig der Nutzungsaspekt. Der Blick in das System findet auf andere Weise statt, siehe Abbildung 2.1.3-b.

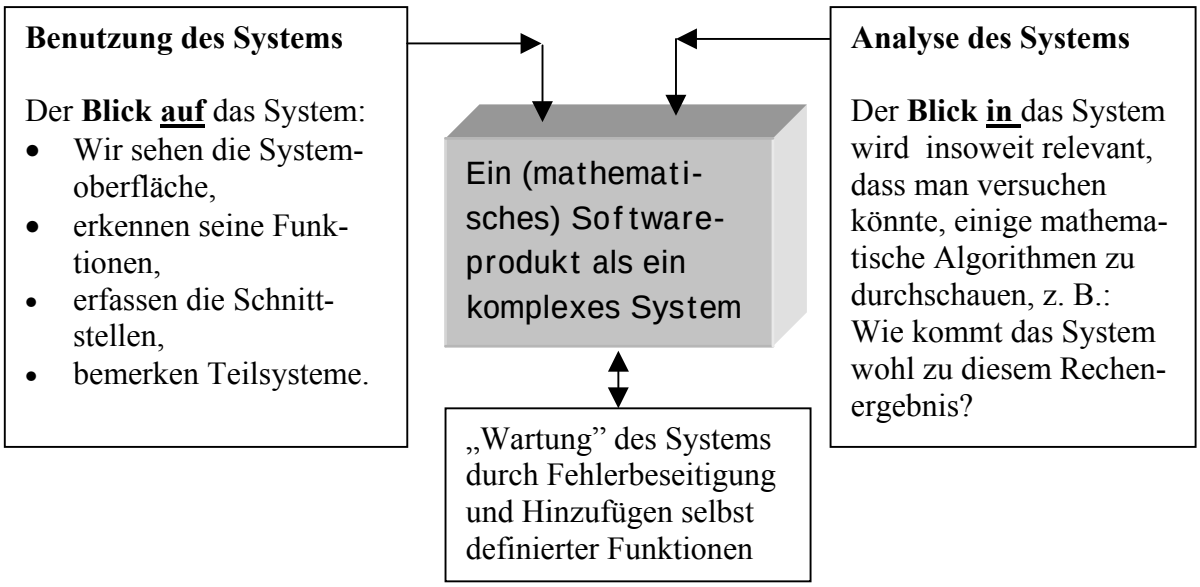


Abb. 2.1.3-b: Sichtweise der Mathematik auf ein Softwareprodukt

Von besonderer Bedeutung ist hier die Möglichkeit, das System durch eigene, selbstdefinierte Funktionen (Bausteine), ggf. auch durch programmierte Funktionen erweitern zu können. Auf diesen Aspekt wird in Kapitel 2.1.5 ausführlich eingegangen.

2.1.4 Projektmethode

Wie oben bereits bemerkt, werden in der Schulinformatik im Rahmen von Projektarbeit komplexe Informatik-Probleme behandelt. Der Modellierungsvorgang beginnt mit einer Brainstorming-Phase, die zunächst zu einer Präzisierung der Problemstellung führt und danach in die eigentliche Modellbildung übergeht.

Ein Phasenmodell für Informatik-Projekte

Projekte im Informatik-Unterricht werden in der Regel in Anlehnung an Methoden des Software-Engineering nach einem Software-Life-Cycle durchgeführt, wie er in Abbildung 2.1.4-a dargestellt wird. Die einzelnen Phasen stellen an den Bearbeiter unterschiedliche Anforderungen und die verwendeten Arbeitsformen sind unterschiedlicher Art.

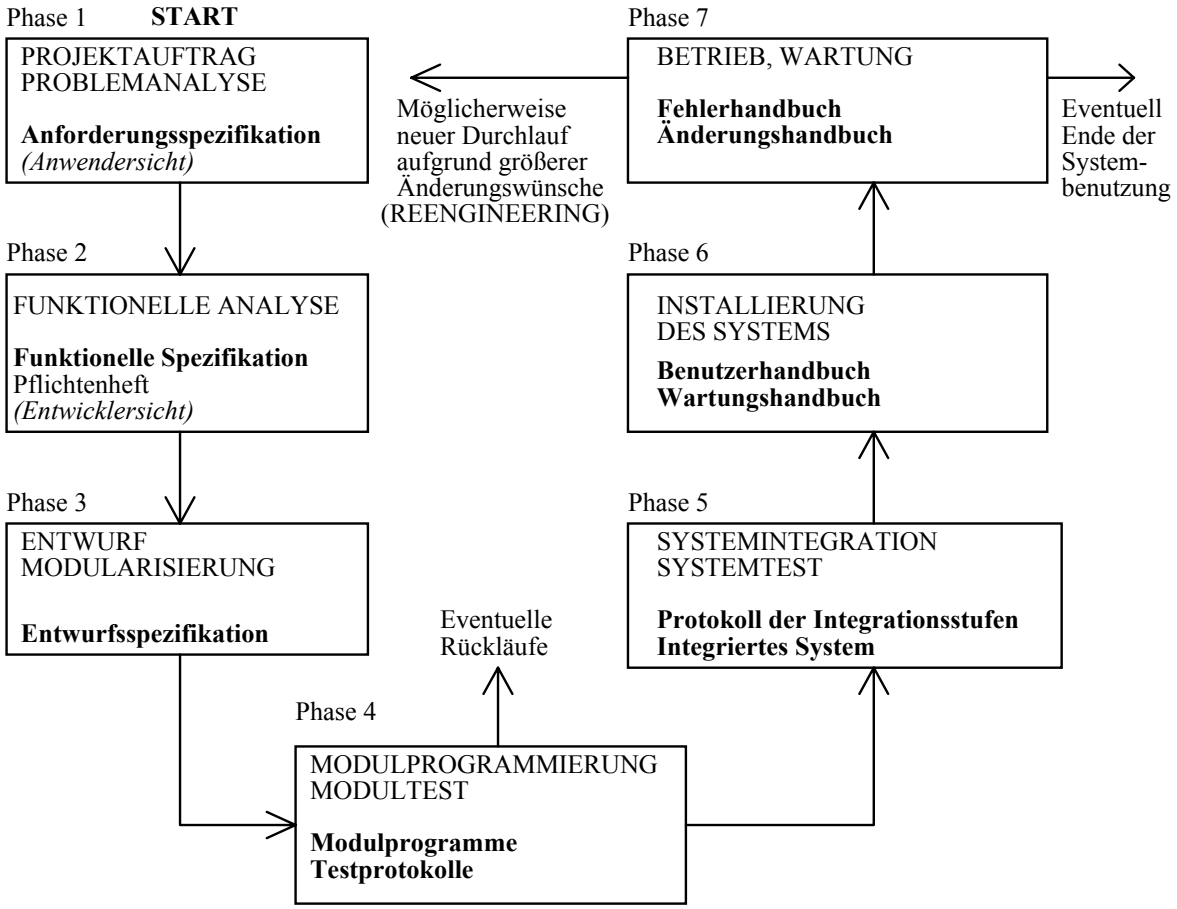


Abb. 2.1.4-a: Life-Cycle für Softwareprodukte, nach
Lehmann, E.: Projekte im Informatikunterricht – Software-Engineering, Dümmler-Verlag, Bonn 1995

Zur Relevanz solcher Phasenmodelle

Zunächst muss bemerkt werden, dass der Software-Life-Cycle (wir schreiben kurz SLC) eine idealisierte Darstellung einer Softwareentwicklung ist. Es hat sich gezeigt, dass die hier genannten Phasen tatsächlich auftreten, dass aber oft Vermischungen zwischen den Phasen und insbesondere Rückgriffe auf Vorhergehendes nötig sind. Dennoch ist der SLC eine wesentliche Hilfe bei der Durchführung von Softwareprojekten

Ein Phasenmodell für Mathematik-Projekte

Das Phasenmodell für Informatik-Softwareprojekte kann unter Beachtung der spezifischen mathematischen Gegebenheiten auf Mathematik-Projekte übertragen werden. Hierfür wird hingewiesen auf das *[Leh99a]: Projekte im Mathematikunterricht*. Dort wird der folgende Ablaufplan vorgestellt (S. 7):

Phase 0: Projektvorbereitung
(Vorkenntnisse, organisatorischer Rahmen, vorhandene Software, ...)

Phase 1:
Offene (komplexe) Problemstellung (meistens durch den Lehrer, möglichst gebietsübergreifend oder fachübergreifend)
Brainstorming
Ordnen: Zerlegung des Problems in Teilprobleme
Auswahl von Teilproblemen zwecks Bearbeitung, Präzisierungen
Festlegung der Teamstruktur, Gruppeneinteilung

„Überwachung“ durch den Lehrer als Projektmanager

Phase 2: Arbeit in den Gruppen
(ggf. Lehrerhilfe)
- Materialbeschaffung
- Benutzung von Hilfsmitteln (Computer, ...)
- Kommunikation mit anderen Gruppen
- Dokumentationsarbeiten

Zwischenzusammenfassungen
- Berichte, Kritik
- neue Direktiven vom Team bzw. vom Lehrer

Phase 3: Integration der Arbeitsergebnisse
- Endberichte
- Vorlegen der Dokumentationen, Ergänzungen, Zusammenstellung
- Beurteilung und Wertung, Kritik
- Ordnen

Phase 4: Der mathematische Ertrag (stärkere Lehrerhilfe)
- Ordnen der mathematischen Ergebnisse, Lehrplanbezug
- Einordnen in größere Zusammenhänge

Abb. 2.1.4-b: Ablauf eines mathematischen Projekts

Im Einzelnen kann auch hier der Projektablauf andere Formen annehmen. Dennoch nennt die Abbildung wichtige Aspekte für viele Projekte.
Angesichts der großen Bedeutung von Projektunterricht für Ziele dieser Arbeit werden nun einige vertiefende Betrachtungen zur Projektarbeit im Mathematikunterricht angeschlossen.

Der Projektbegriff

Projekt (lat.) bedeutet *Plan, Vorhaben* oder auch *Entwurf*.

Der Projektbegriff ist damit so allgemein, dass er in den verschiedensten Zusammenhängen außerhalb und innerhalb der Schule verwendet werden kann, wovon dann auch in Büchern, Zeitschriften, Zeitungen usw. reichlich Gebrauch gemacht wird. In dieser Arbeit geht es um Projektarbeit an Schulen. Bekanntlich findet auch dort Projektarbeit in sehr unterschiedlichen Ausprägungen statt. Genannt seien z. B. Projekttage, Projektwochen, Projekte über ein Kurssemester, Projekte über mehrere Unterrichtsstunden hinweg – fachbezogen oder fachübergreifend.

In der Regel geht es bei Projekten um für den jeweiligen Bereich komplexe Aufgabenstellungen, die dann im Gegensatz zu Routineaufgaben auch mit besonderen Organisationsformen und Methoden bearbeitet werden müssen.

Die obigen Bemerkungen zeigen bereits, dass es wenig fruchtbar ist, den Projektbegriff genauer zu definieren. Wir werden uns für unsere schulischen Projekte vielmehr darauf beschränken, besondere Intentionen von Projektarbeit zu benennen, um so eine Abgrenzung gegenüber dem sonstigen Unterricht vorzunehmen zu können.

Warum Projekte im Mathematikunterricht? Projektziele

Im Mathematikunterricht überwiegt in der Regel die relativ eng an den Inhalten des Lehrplans ausgerichtete Arbeit. Die behandelten Themen haben meistens eine geringe Weite, problemorientierte, offene Ansätze sind selten. Die vorherrschende Unterrichtsform ist das Unterrichtsgespräch. Ein mathematisches Projekt, in dem ein Team an einem komplexen Problem (häufig realitätsnah, aber auch innermathematisch) arbeitet, setzt ein entsprechendes Engagement des Lehrers voraus und erfordert einigen Mut desselben.

Mut
- zur Abweichung von engen Lehrplanvorgaben
- zu einem großzügigen Zeitrahmen
- zu einer anderen, aufwendigeren Unterrichtsform
- sich neuen Anforderungen didaktisch-methodischer Art zu stellen
- einen möglicherweise ungewissen Ausgang zu erleben
- zur Bewältigung überraschender Situationen und Probleme

Projektunterricht kann für die Beteiligten besonders interessant sein. Er bringt aber auch etliche Schwierigkeiten mit sich – auf diese wird unten näher eingegangen. Zunächst werden einige Intentionen von Projektunterricht formuliert, die nicht nur auf den Mathematikunterricht zutreffen; sie werden diesen dabei aber besonders berücksichtigen.

Projektunterricht ist eine besondere Arbeitsform, die ihre eigenen Ziele hat. Mit der Projektarbeit werden neben den mathematischen Zielen allgemeine Ziele angestrebt, u. a.:

- Teamfähigkeit entwickeln
- einzeln und im Team Entscheidungen treffen können
- Kritikfähigkeit zu eigener und fremder Arbeit entwickeln
- Artikulationsfähigkeit entwickeln
- Notwendigkeit und Sinn von Arbeitsteilung einsehen
- selbstständig arbeiten können
- Erlangen von Planungskompetenz
- diverse Arbeitsmittel benutzen und zur Verfügung stehende Ressourcen richtig einschätzen können
- das Gewinnen und Auswerten von Informationen üben
- gemeinsam gewonnene Arbeitsergebnisse integrieren können
- Bewusstmachen des Lern- und Arbeitsprozesses in einer sozialen Gruppe
- Bewusstmachen der benutzten Arbeitsmethoden
- Erzeugung von Produkten zur eigenen Verwendung oder zur Benutzung durch andere Personen bzw. Lerngruppen
- Überwindung des Auseinanderfallens von Theorie und Praxis sowie festgelegter schulischer Fächergrenzen durch Berücksichtigung fächerübergreifender Aspekte.

Projektziele

Spezielle, auf den Anwendungsbereich oder die Realität bezogene Ziele sind:
- Komplexität realer Problemstellungen erkennen können
- die Auswirkungen unterschiedlicher Designentscheidungen einschätzen können.

Die obigen Zielsetzungen für Projekte im Mathematikunterricht werden gerade zurzeit besonders unterstützt durch die Forderungen nach einer offenen Unterrichts- und Aufgaben-kultur im Mathematikunterricht, wie sie in den Modulen des BLK-Projekts „Steigerung der Effizienz des mathematisch-naturwissenschaftlichen Unterrichts“ (SINUS-Modellversuch) sichtbar werden, siehe Kapitel 1.4.1.2. Viele der dort genannten Module gewinnen ihre Relevanz auch durch projektartigen Unterricht.

Projektunterricht unterstützt die Intentionen des BLK-Modellversuchs in besonderem Maße!

Produktorientierung

Projekte sollten produktorientiert sein, in der Regel ist nur so der Antrieb für die Schüler vorhanden, um ein Projekt durchzustehen. Dabei geht es u. a. um die Dokumentation der Projektarbeit und das Aufbereiten der Ergebnisse, z. B. mit Textverarbeitung und Grafik.

Für Mathematikprojekte (Entsprechendes gilt für alle Fächer) sind z. B. folgende Formen der „Veröffentlichung“ möglich:

(1) Aushang von Projektergebnissen auf Tafeln im Klassenraum oder an anderen geeigneten Stellen in der Schule oder auch bei auswärtigen Veranstaltungen.

(2) Zusammenfassung der Ergebnisse in einem kleinen „Projektbuch“, das jeder Projektteilnehmer erhält. Das Projektbuch kann aber auch noch zur Präsentation andernorts dienen oder gar gegen ein kleines Entgelt (Auffrischung der Klassenkasse) abgegeben werden.

(3) Benutzung des Projektbuches in anderen Lerngruppen.

(4) Veröffentlichung einer Projektbeschreibung in der Schülerzeitschrift, vielleicht sogar in einer Fachzeitschrift.

(5) Die Erfahrungen zeigen, dass sich die Schüler zur Dokumentation des Projektes in zunehmendem Maße der neuen Medien bedienen:
- Benutzung eines Textverarbeitungsprogramms,
- Scannen von Bildern,
- Herstellen von Grafiken mit Grafikprogrammen usw.

Diese Ansätze führen dann auch dazu, dass beispielsweise
- Ergebnisse in das Internet gestellt werden,
- Dokumentationen (und eventuell Programme) auf eine Diskette oder eine CD gebracht werden.

Die Rolle des Lehrers bei Projektarbeit

Aus größeren Projekten, etwa auch aus dem Informatikunterricht, ist der Begriff des Projektmanagements bekannt. Er erweist sich als nützlich, wenn es darum geht, die hier gegenüber verbreiteteren Unterrichtsformen andersartigen Aufgaben des Lehrers zu verstehen. So ergibt sich eine erweiterte Sichtweise.

Ein **Projektmanager** hat die Aufgabe, das Projekt zu führen und zu verwalten. In der Schule wird der Projektmanager in der Regel der unterrichtende Lehrer sein. Unter günstigen Bedingungen können gelegentlich auch verständige Schüler (mit pädagogischem Geschick) Projektleiter sein.
Der Projektmanager ist verantwortlich für die Organisation und Vorgehensweise der Projektgruppe und muss in der Lage sein, Probleme zu erkennen, sich um sie zu kümmern und sich mit seinen Mitarbeitern um systematische, konstruktive Lösungen zu bemühen.

Bei mathematischen Projekten hat der Lehrer insbesondere folgende Aufgaben zu übernehmen:

Der Lehrer als Projektmanager

- Auswahl des Projektthemas, jedenfalls in den meisten Fällen,
- Aufteilung in Gruppen, ggf. in Zusammenarbeit mit den Schülern,
- Leitung bei der Präzisierung von Aufgabenstellungen,
- Leitung gemeinsamer Diskussionen,
- Organisation von Zwischenberichten,
- Bereitstellung von den Schülern nicht bekannten mathematischen und anderen Hilfsmitteln,
- Bereitstellung von Medien,
- Hilfestellung
  - durch Hinweise auf geeignete Computerprogramme und Bedienungshinweise,
  - beim Finden schwieriger Ansätze,
  - in mathematischen Detailfragen,
  - bei der Dokumentation.

Die Rolle der Schüler

Auch die Schülerrolle verändert sich gegenüber der von ihnen im normalen Unterricht erwarteten Rolle. Man darf deshalb nicht erwarten, dass die in einer Lerngruppe erstmalige Anwendung der Projektmethode von den Schülern sofort in der gewünschten Weise praktiziert werden kann. Umso wichtiger ist es, den Schülern bei ihrem Projekt das Besondere dieser Arbeitsform zu verdeutlichen.

In einer Befragung nach einem mathematischen Projekt in Klasse 11 über „Abbildungsgeometrie mit Matrizen" schreibt ein Schüler zu den Aspekten „Rolle des Schülers" und „mathematische Erkenntnisse":

„Rolle des Schülers: Die Schüler konnten als Individuen arbeiten und ihre Persönlichkeit entwickeln. Die Schüler konnten frei arbeiten und hatten nicht diesen Druck des Lernens. Der Lehrer gab den einzelnen Gruppen nur Hinweise, wie die Aufgabe besser oder überhaupt zu lösen sei. Die Teamarbeit spielt bei der Projektarbeit eine große Rolle. Die Schüler mussten aufeinander eingehen, haben gelernt, Formeln, Beweise und Behauptungen zu konstruieren, was nicht immer leicht war.
Die Projektarbeit hat viele Vorteile: Das Lernen in kleinen Gruppen fällt leichter, der Unterricht ist lockerer und vieles mehr. Aber auch die Nachteile sind nicht zu übersehen. Die einzelnen Teams fixieren nur bestimmte Themen und dadurch muß das, was von den anderen Gruppen zusammengestellt wurde, nachgearbeitet und erlernt werden (z. B. für die Klausur).
Meiner Meinung nach ist Gruppenarbeit eine gute Sache, denn durch sie fällt die Schule nicht mehr als Last auf, sondern man geht gerne zu diesem Unterricht (sogar in der 6. Stunde). Die Gruppenarbeit war zwar toll, aber immer geht das nicht, denn man muss auch lernen in einer großen Gruppe (Klasse) auszukommen, insbesondere für das Studium, da wird auch nicht in Gruppenarbeit erlernt, sondern der Professor übernimmt diese Aufgabe.
Da ich nur wenig Kenntnisse am Computer habe (vom Spielcomputer einige Kenntnisse), fand ich die Arbeit mit den Computerprogrammen recht gut, und sie war nicht so trocken, wie gewöhnlicher Unterricht. Durch sie habe ich neue Kenntnisse in Sachen Programmen bekommen. Die interessanteste Arbeit war die Arbeit am Computer. Die Graphen und Abbildungen, die man anhand des Programms PLOT11 erstellen konnte – das war für mich faszinierend.
Mathematische Erkenntnisse: Beweisführungen, Erstellen von Formeln."

In der Tat schult projektartiger Unterricht in besonderem Maße die Selbstständigkeit der Schüler und fördert ihr Selbstbewusstsein. Der Lehrer muss sich ohnehin mehr auf selbstständigere und kenntnisreiche Schüler einstellen, insbesondere wenn es um die Nutzung der neuen Medien geht.

Teamfähigkeit – eine Schlüsselqualifikation

Teamfähigkeit gehört heute zu den Schlüsselqualifikationen vieler Berufe. Darauf muss in der Ausbildung reagiert werden. Teamfähigkeit bedeutet

- kooperatives Arbeiten und Konfliktbewältigung in einer Gruppe, insbesondere
  - Artikulationsfähigkeit zum Vertreten eigener Meinungen und
  - Aufnahmefähigkeit für andere Meinungen sowie
  - Kritikfähigkeit eigener und fremder Arbeit gegenüber entwickeln,

- Notwendigkeit und Sinn von Arbeitsteilung einsehen.

2.1.5 Module – CAS-Bausteine

2.1.5.1 Informatische Grundlagen

Hinweis: Die Ausführungen in Kapitel 2.1.5 übernehmen teilweise Überlegungen aus meinem kürzlich erschienenen Buch über Computeralgebrasystem-Bausteine *[Leh02]*. Da diese Überlegungen auch hier für das Verständnis des Bausteinkonzepts (Vernetzung Informatik/Mathematik) unerlässlich sind, erscheint es mir angemessen, diese Überlegungen ohne Umweg über andere Literatur in Teilen zur Verfügung zu stellen. In dem Werk findet man ausführlichere Untersuchungen zum Bausteinprinzip.
Angesichts des in dieser Arbeit häufiger verwendeten Begriffs „Baustein" sind zunächst einige Ausführungen zu den Begriffen „Modul", „Prozedur" und „Baustein" nötig.

Modul
In der Literatur wird im Zusammenhang mit CAS immer wieder der Begriff „Modul" genannt, häufig ohne dass er definiert wird. So wird er beispielsweise im *TI-92-Handbuch (1995)* oder im Handbuch für *DERIVE für WINDOWS 1997* nicht erwähnt. In der Informatik spielt der Begriff eine wesentliche Rolle. Der Informatik-Duden (1993) sagt hierzu (S.433):

„Modul (engl. module): Der Begriff wird in mehreren Bedeutungen verwendet:
1. Software-Modul: Bausteine, aus denen sich ein Software-System zusammensetzt, bezeichnet man als Module. Die Beziehungen zwischen Modulen werden durch ↑Schnittstellen festgelegt. Viele Programmiersprachen unterstützen ein solches Modulkonzept...
   Ein Modul wird in der Praxis als ein in sich zusammenhängender Baustein aufgefasst, der stets folgende Eigenschaften besitzen sollte (vgl. auch ↑Software-Engineering):
   - Er ist logisch oder funktional in sich abgeschlossen.
   - Wie er arbeitet oder implementiert ist, braucht außen nicht bekannt zu sein (*information hiding*).
   - Er besitzt klar definierte Schnittstellen nach außen.
   - Er ist überschaubar und somit leicht testbar.
   - Er sollte nur möglichst wenige andere Module verwenden.

   ...
2. Realisierung eines abstrakten ↑Datentyps. Orientiert man sich an diesen theoretischen Grundlagen, so lassen sich die unter 1. genannten fünf wichtigen Eigenschaften recht gut erfüllen.
   (Anmerkung (kein Zitat): Datentyp – Zusammenfassung von Wertebereichen und Operationen auf ihnen zu einer Einheit. Beim abstrakten Datentyp sind seine Eigenschaften von besonderem Interesse, unabhängig von einer Programmiersprache).
3. Hardware-Baustein mit wohldefinierten Ein- und Ausgängen.
4. In der Mathematik: Zahl, bezüglich derer der Rest bei einer Division gebildet wird ..."
(Zitatende)

Der auch im Informatik-Duden mehrfach verwendete Begriff „Baustein" wird dort nicht weiter definiert. Für die mathematischen Betrachtungen in dieser Arbeit sind die oben genannten Punkte 1 und 2 von Interesse. Aus Sicht der Informatik sind Module in der Regel Einheiten innerhalb größerer Systeme. Diese Auffassung spiegelt sich jedoch in den konkreten Aufgabenstellungen der Schulmathematik nur bedingt wider. Der in dieser Arbeit verwendete Bausteinbegriff knüpft daher eher an den Prozedurbegriff und den Funktionsbegriff an.

Im *Informatik-Duden (1993), Seite 433 f.,* heißt es dann auch:
„Das Modulkonzept geht in seiner Leistungsfähigkeit über das Prozedurkonzept ... hinaus. Während sich beim Prozedurkonzept die Zerlegung auf die operationale Ebene beschränkt, d.h. nur eine Operation kann in mehrere andere zerlegt werden, verallgemeinert das Modulkonzept diese Idee auf die Daten und Operationsebene. Daten (Datentypen, Variablen, Konstanten) und Operationen darauf (Prozeduren, Funktionen) werden zu einer Einheit (einem Modul) zusammengefaßt."

2.1.5.2 Das Prozedurkonzept

Bei der Beschäftigung mit informatischen Ansätzen für den Mathematikunterricht muss auch auf die Rolle des Programmierens im Mathematikunterricht eingegangen werden. Nähere Ausführungen hierzu folgen in Kapitel 2.1.6. Dabei sind auch die hier folgenden Ausführungen über das Prozedurkonzept von Bedeutung.

Zu den wichtigsten Strategien des Programmierens in imperativen Programmiersprachen wie z. B. PASCAL oder auch in Computeralgebrasystemen gehört das Prozedurkonzept. Was ist darunter zu verstehen?

(1) Prozeduren bei der Zerlegung eines Problems in Teilprobleme

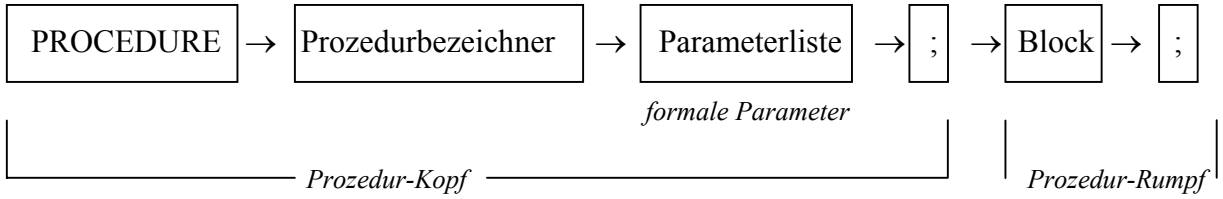
Viele Probleme sind so umfangreich und komplex, dass eine Zerlegung in Teilprobleme sinnvoll ist. Damit wird die Bearbeitung durchsichtiger und kann ggf. auch an verschiedene Bearbeiter delegiert werden. Für diese Teilprobleme kann man nun passende Prozeduren konstruieren, so dass sich schließlich die gesamte Problemlösung als eine Sammlung geeignet angeordneter Prozeduren darstellt.

(2) Wiederverwendbarkeit von Prozeduren

Man wird dabei auch bemerken, dass immer wieder Prozeduren benötigt werden, die die gleiche Aufgabe bearbeiten, etwa die Eingabe von Zeichenketten. Damit gewinnen manche Prozeduren einen universellen Charakter, so dass sie für eine Problemlösung mehrfach oder auch bei der Bearbeitung anderer Problemstellungen eingesetzt werden können. Das setzt allerdings eine geeignete, universell brauchbare Formulierung voraus. Derartige Prozeduren sollte man in Bibliotheken zusammenfassen, um sie immer auf einfache Weise zur Verfügung zu haben.

(3) Definition von Prozeduren

Prozeduren müssen zum Beispiel in TURBO-PASCAL im Vereinbarungsteil des übergeordneten Programmteils definiert werden. Wie das Syntaxdiagramm zur Prozedurvereinbarung zeigt, haben Prozeduren einen ähnlichen Aufbau wie ein Programm. Die Prozeduraufrufe können an jeder Stelle des Anweisungsteils des Programms erfolgen.



Beispiel (in der Programmiersprache TURBO-PASCAL):

Prozedurvereinbarung

```
PROCEDURE Gehe_zu_Position(x,y : INTEGER);
BEGIN ...
...                          Block
END ;
```

Prozeduraufruf innerhalb eines längeren Programms:

```
Gehe_zu_Position (12, 14).
```

Für die Unterrichtsarbeit ist die Unterscheidung zwischen Prozedurvereinbarung und Prozeduraufruf besonders wichtig!

Die im Prozedur-Kopf möglicherweise genannten Parameter sind so genannte "formale Parameter". Beim Prozeduraufruf werden sie ersetzt durch die "aktuellen Parameterwerte". Dabei muss auf gleiche Reihenfolge der Parameter geachtet werden.

(4) Prozedurarten

Es gibt verschiedene Arten von Prozeduren:

	Aufrufbeispiele
Prozeduren ohne Parameter	Clrscr, lösche Bildschirm
Prozeduren mit Parametern	Gehe_zu_Position(12,14)
Funktionen ohne Parameter	y : = x*x
Funktionen mit Parameter	y : = potenz(basis, hochzahl)

Funktionen sind besondere Prozeduren, die bei einem Aufruf stets einen Wert zurückliefern und damit Teil einer Anweisung sind, während ein Prozeduraufruf eine eigene Anweisung ist.

Bausteine
Prozeduren (Funktionen) können in dem oben beschriebenen Sinne als Bausteine für umfangreichere informatische Problemlösungen bezeichnet werden. – In entsprechender Weise kann nun auch von Bausteinen für mathematische Problemlösungen gesprochen werden.

2.1.5.3 Bausteine und ihre Parameter

Parameter bei Kurvenscharen und bei Parameterdarstellungen

Bekanntlich sind Kurvenscharen ein Standardthema in der Analysis, das viel interessante Mathematik beinhaltet und nicht ohne Grund beliebt im Unterricht und bei Klausuraufgaben ist. Hierbei werden die Schüler – möglicherweise erstmals – auch mit dem Umgang mit Parametern vertraut gemacht. Weitere Gebiete, in denen schon lange mit Parametern gearbeitet wird, sind die vektorielle analytische Geometrie (z. B. bei Geradengleichungen) oder Kurven in Parameterform. Dabei werden vielfach knappe Termabkürzungen wie f, r, y verwendet, mit denen man viel an Verständnis und an unterrichtlichen Möglichkeiten verschenkt. Schreib-

weisen, die die Parameter explizit nennen, erweisen sich bei den heute vorliegenden Möglichkeiten als günstiger. So kann man z. B. schreiben:

Kurzschreibweise	ausführliche Notation / Aufrufbeispiel
Kurvenscharen $f(x) = ax^3 + bx^2 + cx + d$ mit $x, a, b, c, d \in R$	$f(x, a, b, c, d) = ax^3 + bx^2 + cx + d$ Aufrufbeispiel: f(x, 1, 2, 3, 4)
Vektorielle Geradengleichung $r = r_1 + tu$	$r(t) = r_1 + tu$ Aufrufbeispiel: r(5)
Kurve in Parameterform $x = \cos(t), y = \sin(t)$	$x(t) = \cos(t)$ und $y(t) = \sin(t)$ Aufrufbeispiel: $x(\pi/3), y(\pi/3)$
Trapez-Flächeninhalt $A = (a + b)*h / 2$	$A(a, b, h) = (a + b)*h / 2$

Es lohnt sich, derartige Schreibweisen schon frühzeitig – schon in der Sekundarstufe 1 (ab Klasse 7!) – zu verwenden. Hierzu ein Beispiel aus dem neuen Schulbuch
„Mathematik - Neue Wege", Klasse 7, Hrsg. A. Lergenmüller, G. Schmidt, Schroedel-Verlag, Hannover 2001.

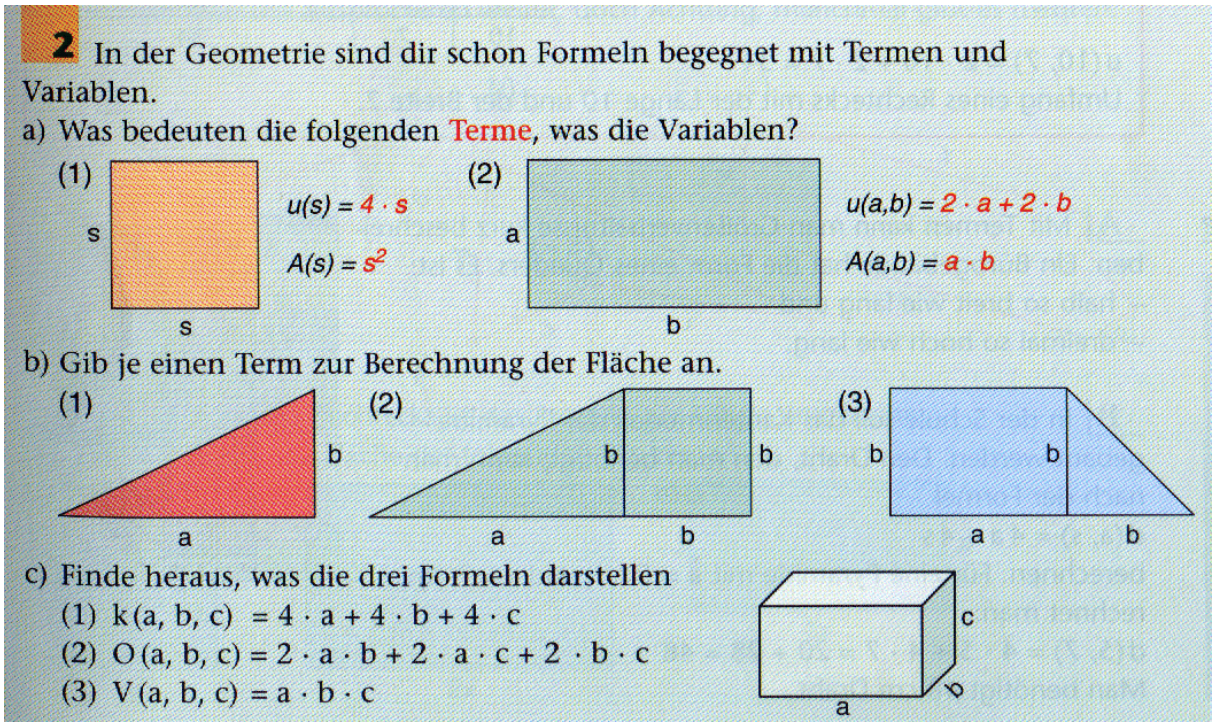


Abb. 2.1.5.3-a: Verwendung von Parametern im Schulbuch „Neue Wege" (s. o.), S. 39

Welche Gründe sprechen dafür, was ist zu beachten?

- Die Terme erscheinen unter einem gemeinsamen Aspekt, indem die Schreibweise auf das Wesentliche reduziert wird, nämlich auf die gewählte Einsetzung, Unterschiede werden deutlicher.

- An der kompakten Schreibweise (z.B. f(x, a, b, c, d)) wird deutlicher, dass man diverse Einsetzungen für die Parameter vornehmen kann. Dieser Aspekt verstärkt die Motivation, Einsetzungen zu erproben, und gibt damit den Anreiz zu experimenteller Arbeit.

- Statt der Einzelexemplare eines Objekttyps rücken nun Teilmengen von Objekten ins Blickfeld. Zum Beispiel ist nun nicht mehr nur eine einzelne Kurve von Interesse, sondern es geht gleich um Kurvenscharen – und erst aus dem Vergleich verschiedener Exemplare erschließen sich in der Regel die entscheidenden Eigenschaften. Damit wird auch der Wunsch nach Fallunterscheidungen unterstützt.

- Die Überlegungen bezüglich der einsetzbaren Werte verbreitern das Anwendungsfeld des Terms auf (manchmal unerwartete) unterschiedliche Bereiche. So ist es z.B. möglich in gewisse Terme statt reeller Zahlen Funktionsterme oder auch quadratische Matrizen einzusetzen. Beispiel: Einsetzungen in den Baustein (a+b)^2→ Binom(a,b). Hinweis: → ersetzt die „STOre-Taste" auf dem TI-92.

- Die Schreibweise mit Parametern ist „computeralgebra-freundlich" und wird von den CAS-Systemen entsprechend unterstützt – sowohl in algebraischer als auch in grafischer Hinsicht. Beispiel: define parabel(x, p, q) = x^2 + p*x +q bzw. x^2 + p*x +q→parabel(x, p, q) .

- Die schon für die Sekundarstufe 1 sehr nützliche Konzeption von Unterricht unter Parameter-Aspekten erweist sich als ausgezeichnete Vorbereitung für die Probleme der Sekundarstufe 2.

Und schließlich ein besonders wesentlicher Aspekt, der bedeutungsvoll für die Überlegungen zum Programmieren im Mathematikunterricht ist:

- Bausteine ersetzen häufig kleine Programme und machen somit einen Teil des Programmierens überflüssig. Gerade hierin liegt u. a. die Bedeutung von Computeralgebrasystemen!

Man kann somit festhalten:

> Im „Denken in Parametern und Bausteinen" liegen erhebliche unterrichtliche Möglichkeiten, die sich durch die Verwendung von Computeralgebrasystemen (CAS) und Funktionenplottern noch verstärken bzw. erst realisierbar werden.

Diese Aussage gilt sowohl für die Sekundarstufe 2 als auch für die Sekundarstufe 1. Betrachtet man z. B. den folgenden Auszug aus mathematischen Formelsammlungen für die Sekundarstufe 1, so ahnt man die Fülle der Möglichkeiten.

$$(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3 \qquad y = m \cdot x + n$$

$$W = \frac{G}{100} \cdot p \qquad V = \frac{1}{3}\pi r^2 h \qquad A = \frac{1}{2}(a+c) \cdot h$$

$$x_{1/2} = -\frac{p}{2} \pm \sqrt{(\frac{p^2}{4} - q)} \qquad e = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$$

$$x = a\sin(t) + c$$
$$y = b\cos(t) + d$$

$$x^2 + y^2 = r^2$$

$$y = a \cdot x^2 + b \cdot x + c, \qquad y = (x-a) \cdot (x-b)$$

Überall in der Sekundarstufe 1 begegnet der Schüler Termen mit Parametern. Erst recht in der Sekundarstufe 2! Also sollte man die daraus erwachsenen Möglichkeiten auch verwenden.

Allerdings:
Beim Unterricht mit dem Prozedurkonzept mit den häufig erfolgenden Prozeduraufrufen muss dafür gesorgt werden, dass der zugrunde liegende, ausführlich geschriebene Term stets im Blickfeld bleibt.

Wie werden CAS-Bausteine definiert?

In Computeralgebrasystemen kann man z. B. eingeben und damit einen Baustein definieren:

x^2 + p*x +q → parabel(x,p,q) (Notation des Taschencomputers TI-92) oder

define parabel(x,p,q) = x^2 + p*x +q. (Notation TI-92 und DERIVE).
Man bewirkt damit die Speicherung des Terms $x^2 + p \cdot x + q$ in eine Variable „parabel“ mit den Parametern x, p und q. Dieser Baustein steht nun für spätere Aufrufe zur Verfügung. So kann man zum Beispiel parabel(x,3,4) = 0 eingeben und erhält die Gleichung $x^2 + 3x + 4 = 0$.

Informatiker würden z. B. programmieren:
PROCEDURE parabel(x,p,q : REAL) oder auch als Funktion
FUNCTION parabel(x,p,q : REAL) : REAL,
jeweils gefolgt von einer passenden Anweisungsfolge.

Im *Duden Informatik (2.Auflage, Dudenverlag, 1993, S. 507)* wird definiert:

„**Parameter**: Platzhalter in einer Programmeinheit, der erst bei der konkreten Verwendung (↑Aufruf) der Programmeinheit festgelegt wird. Die in der Programmeinheit stehenden Platzhalter bezeichnet man als **formale Parameter**, die im Aufruf der Programmeinheit stehenden Werte als **aktuelle Parameter**. Programmeinheiten sind ↑ Prozeduren, ↑ Funktionen, ↑ Module oder andere spezielle Konstrukte. ... Parameter können ↑ Konstanten, ↑ Variablen, ↑ Marken, ↑ Prozeduren, ↑ Funktionen usw., d.h. alle in einem Programm definierbaren Objekte sein.”
„**Parameterübergabe**: Bei einem ... ↑ Aufruf müssen die formalen Parameter auf festgelegte Art durch die aktuellen Parameter ersetzt werden.”

In der Informatikdidaktik hat man früher (noch dem „Programmieren im Kleinen“ verhaftet) derartige Definitionen von Prozeduren mit Parametern und die Arbeit mit diesen als schwierig eingestuft und im Anfangsunterricht zunächst nur Prozeduren ohne Parameter betrachtet. Das hat sich mit wachsenden Unterrichtserfahrungen als nicht nötig erwiesen. Mit den heutigen Hilfsmitteln können von Anfang an komplexere Problemstellungen angegangen werden. Die Komplexität jedoch ist nur zu bewältigen, wenn man auf Konzepte zurückgreift, wie auf die der wiederverwendbaren Bausteine und die des Einsatzes von Tools (Hilfsprogrammen). Solche wiederverwendbaren Bausteine sind u. a. Prozeduren aus Baustein-Sammlungen (Modulen).

So ist z. B. die Prozedur

```
PROCEDURE Input_string (spalte,zeile:      INTEGER;
                        infotext:          STRING;
                        VAR input:         STRING;
                        stringlaenge:      INTEGER;
                        erlaubt:           zeichenmenge);
```

ein mächtiger Baustein, mit dem man eine Zeichenkette (input) maximal vorgegebener Länge (stringlaenge), beginnend an einer bestimmten Bildschirmposition (spalte,zeile) eingeben kann. Dazu wird noch ein Aufforderungstext geschrieben (infotext), und es werden nur Zeichen aus einer vorher festgelegten Menge (zeichenmenge) angenommen. Die Verwendung dieser Prozedur erfolgt durch einen Aufruf wie z. B.

```
Input_string(2,4,´Eingabe des Namens: ´, nachname,20, [´A´..´Z´,´-´]).
```

Damit hat man einen sehr universell verwendbaren Baustein zur Verfügung, der in der Programmierpraxis im Unterricht vielfach zur Anwendung gekommen ist. Ein kleiner überschaubarer Programmausschnitt belegt diese Aussage:

```
REPEAT
    FOR i:=0 TO 5 DO
        FOR j:=0 TO 9 DO
        Input_string(4+i*11,3+j,' ',planstelle[i,j],10,  ['A'..'Z','a'..'z','0'..'9','.','-']);
    Input_zeichen(1,23,'Alles Ok ? (j,n) = ',antwort, ['J','N','j','n']);
UNTIL antwort IN ['j','J'];
```

In diesem Ausschnitt aus einem TURBO-PASCAL-Programm wird der Baustein innerhalb zweier Schleifen verwendet, um z. B. den Stundenplan eines Schülers an sechs Schultagen einzugeben. Ein zweiter ähnlicher Baustein ist Input_zeichen(...). An einer anderen Stelle des Programms heißt es:

```
Input_string(1,1,'Ausgabe des Plans - Dateiname = ´, dateiname, 12, ['A'..'Z','a'..'z','0'..'9','.','-']);
```

An Bildschirmposition (1,1) wird zunächst der Text ausgegeben. Dann wird die Eingabe eines Dateinamens der maximalen Länge 12 erwartet. Dieser darf nur aus den angegebenen Zeichen bestehen. Wieder zeigt Input_string seine Fähigkeiten. Beide oben genannten Bausteine und weitere Eingabe-Bausteine sind in einer Prozedursammlung gespeichert.
Für die Mathematik kann eine solche Sammlung eine **"Bausteinkiste"** werden, aus der wir uns je nach Bedarf Bausteine heraussuchen und miteinander kombinieren können.

Die Erfahrungen haben gezeigt, dass den Schülern derartige Prozedur-Aufrufe mit Parametern schon im Informatik-Anfangsunterricht nicht schwerfallen, sofern die Bedeutung der Parameter klar dokumentiert ist. Mit Hilfe von Bausteinen wird in der Informatik auch die Programmierung komplexer Systeme möglich.

Das Programmieren solcher Systeme wird damit zu einem erheblichen Teil ein Zerlegen in Teilsysteme (Module) und ein Konfigurieren passender Bausteine. Prozeduren (Funktionen) können also in dem oben beschriebenen Sinne als Bausteine für umfangreichere informatische Problemlösungen bezeichnet werden. – An diese Erfahrungen kann der Mathematikunterricht anknüpfen, denn in entsprechender Weise kann nun auch von der Anwendung von Bausteinen für mathematische Problemlösungen gesprochen werden.

2.1.5.4 Das Bausteindreieck

Die folgenden Ausführungen liefern grundlegende Informationen über die didaktisch-methodischen Möglichkeiten des Bausteineinsatzes.

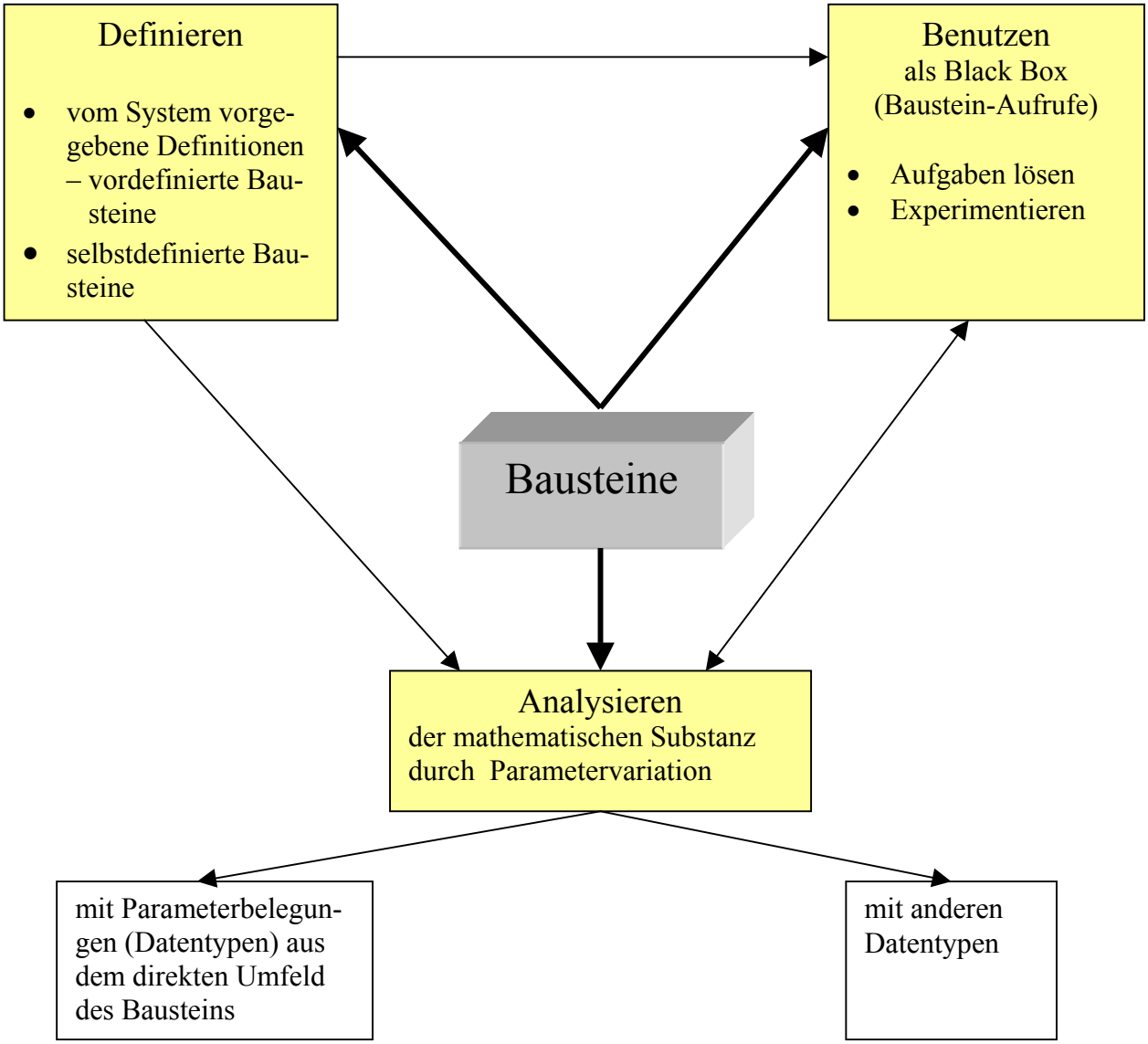


Abb. 2.1.5.4-a: **Das Bausteindreieck**: Definieren, Benutzen, Analysieren

Das Bausteindreieck zeigt Möglichkeiten im Umgang mit einem Baustein. Nach seiner Definition kann er in Form von Bausteinaufrufen auf Probleme angewendet werden. Der Baustein kann aber auch analysiert werden, indem man gezielte Aufrufe durchführt. Außerdem können häufig auch die Datentypen variiert werden, um den Baustein in anderen Wertebereichen zu erforschen.

Die Überlegungen werden nun an einem konkreten Beispiel in Abbildung 2.1.5.4-b verdeutlicht.

Das Bausteindreieck an einem Beispiel

Beispiel:
Bausteindefinition $(a+b)^2$ → bino2(a,b)

a und b sind formale Parameter

Aufrufe: Zum Beispiel bino2(3x, -2y) oder bino2(5,6).
3x, –2y bzw. 5, 6 sind aktuelle Parameter

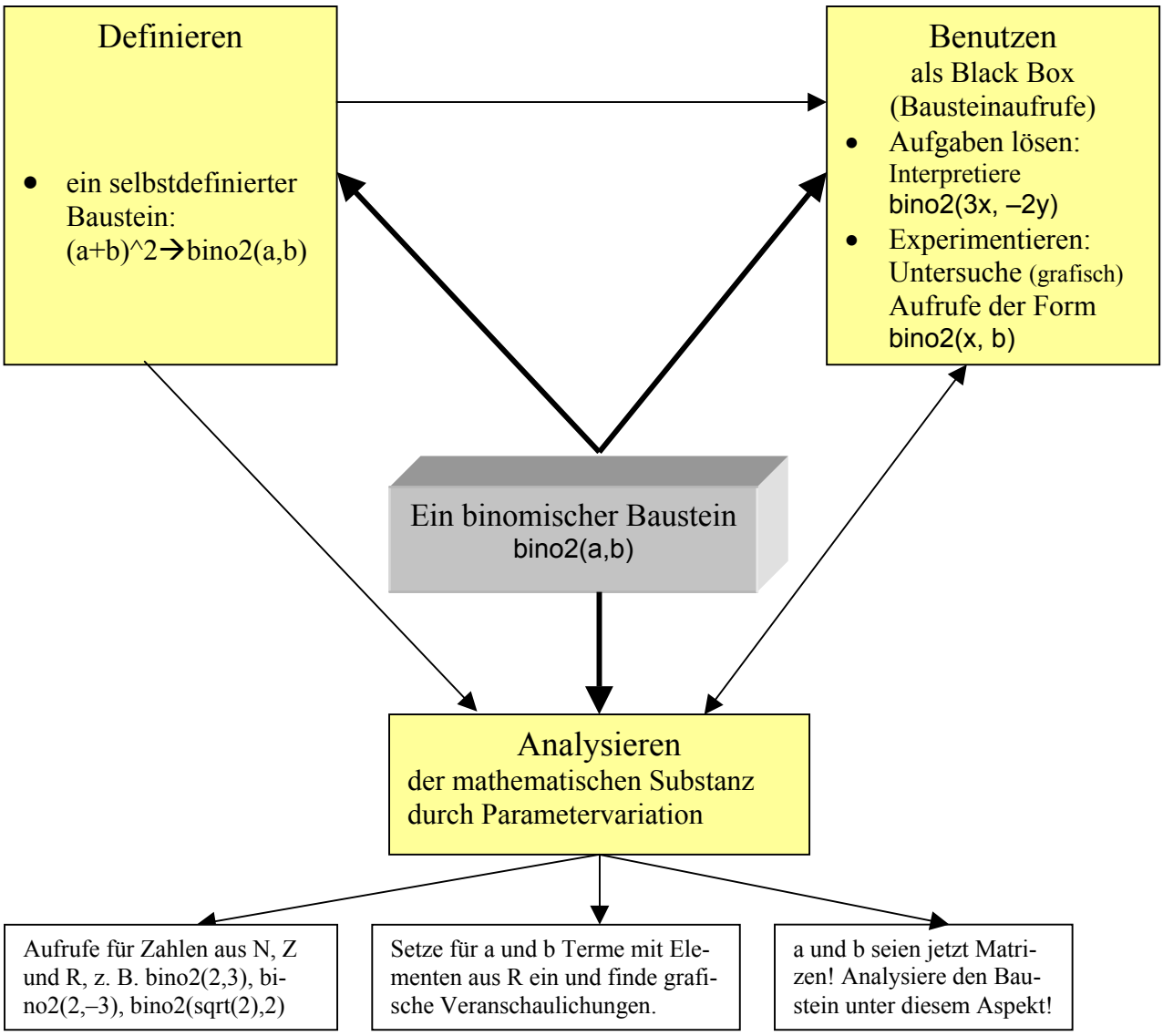


Abb. 2.1.5.4-b: Das *Bausteindreieck* am Beispiel eines binomischen Bausteins

Unterricht mit Bausteinverwendung könnte an jeder der drei Ecken des Bausteindreiecks beginnen.

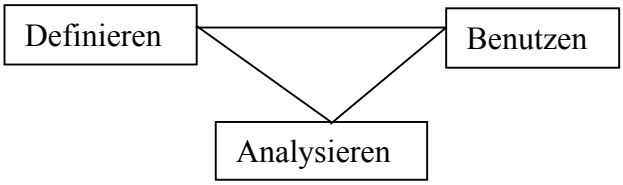


Abb. 2.1.5.4-c: Startpunkte für die Arbeit mit Bausteinen

2.1.5.5 Bausteine definieren, benutzen, analysieren

Alle folgenden Ausführungen sind davon abhängig, wie weit dem Benutzer bereits Bausteine bekannt sind oder ihm erstmals begegnen. Je nach Situation sind dann unterschiedliche Wege möglich. Wir beginnen hier mit der Definition eines Bausteins.

A Definieren eines Bausteins

> Es gehört zu den bevorzugten Arbeitsweisen mit einem CAS, Objekte (Terme, Gleichungen usw.) mit einem passenden Namen und geeigneten Parametern zu versehen, jedenfalls wenn das Objekt mehrmals verwendet werden soll.

Beispiele:

Anlass, Erläuterung	Definition des Bezeichners	Benutzen des Bezeichners
Der Funktionsterm f(x) = a*sin(x+b)+c soll erforscht werden.	sinus(x,a,b,c) : = a*sin(x*b+)+c	• sinus(x,2,0,4) Das ist der Funktionsterm 2sin(x)+4.
Wir wollen Anwendungen binomischer Formeln bearbeiten.	binomi(a,b,n) : = (a+b)^n	• binomi(a,b,3), also $(a+b)^3$ • binomi(x,1,2)
Für den folgenden Unterricht werden mehrfach Zeichnungen der Differenzenquotientenfunktion benötigt.	diffq(x,h) : = (f(x+h)–f(x)) / h	• f(x) : = sin(x) diffq(x, 0.1) • f(x) : = e^x diffq(x, 0.1)

B Benutzen eines Bausteins

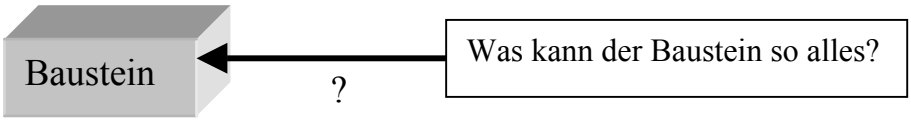
Diese Anwendung wird am Beispiel des Bausteins solve(...) gezeigt. Dieser liegt hier also schon definiert vor. Er ist eine Black Box, die vielleicht auch schon früher verwendet wurde oder die für die Schüler neu ist. Er dient jetzt beispielsweise zur Lösung der folgenden Aufgabe.

Aufgabe:	**Lösung:**
Bestimme den Schnittpunkt der Geraden g1 und g2. g1: y = 2x +1, g2: –3x – 4y = +2	Im Schnittpunkt S der beiden Geraden (falls sie einen besitzen) sind x- und y-Wert bei beiden Geraden gleich. Somit kann man S durch Lösung eines linearen Gleichungssystems bestimmen. Hierfür können wir den Baustein solve benutzen: solve(y = 2x +1 and –3x – 4y = +2, {x,y}). Als Schnittpunkt ergibt sich S(–6/11, –1/11).

Bei dieser Aufgabe wurde der Baustein solve(...) zur Lösung eines linearen Gleichungssystems verwendet – also als Black Box., denn für den Lösungsalgorithmus haben wir uns hier nicht interessiert, siehe D.

C Analyse eines Bausteins

Hat man einen Baustein mehrfach benutzt und einige seiner Anwendungsmöglichkeiten erkannt, besteht möglicherweise der Wunsch, in den Baustein hineinzublicken:



Ein Beispiel:

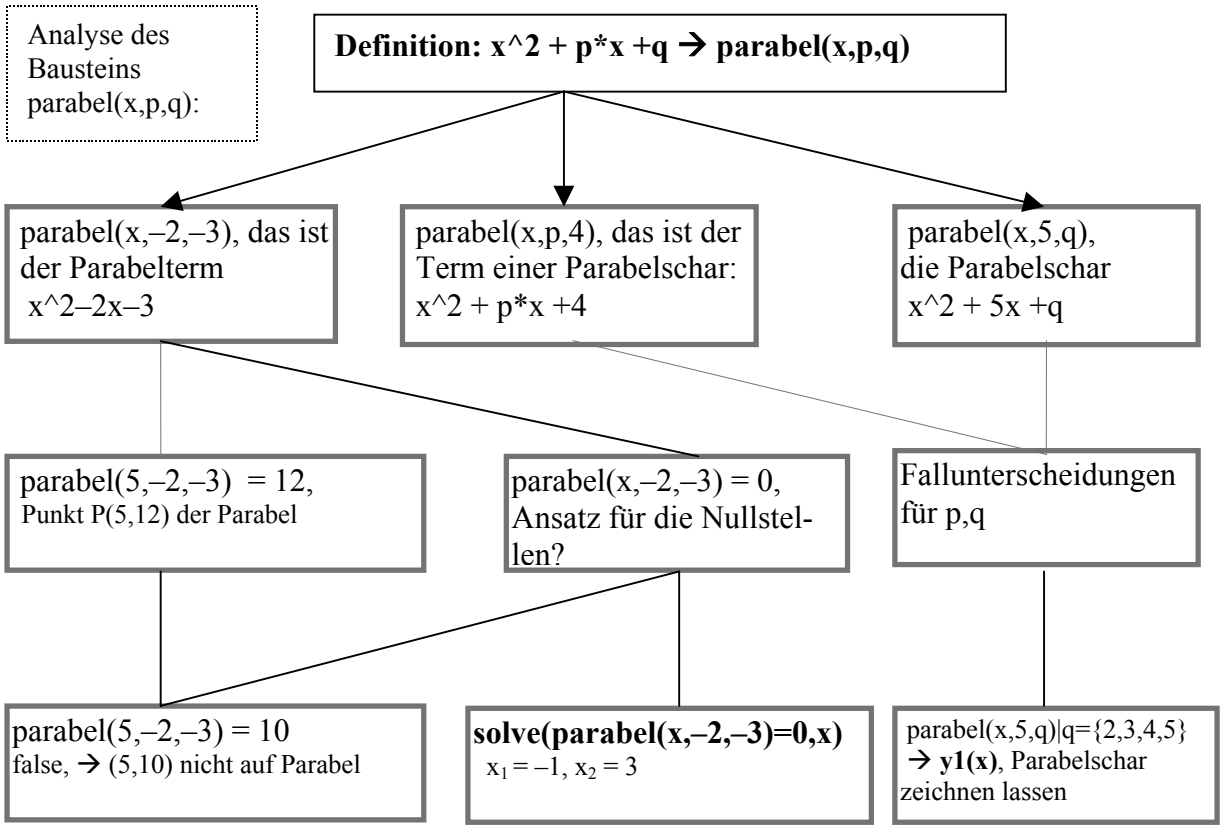


Abb. 2.1.5.5-a: Beispiele für Bausteinaufrufe

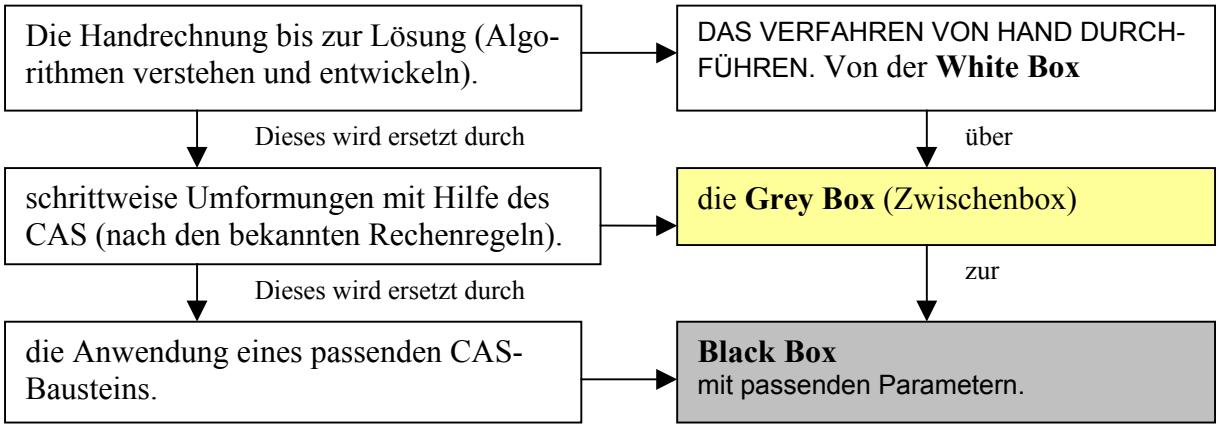
D Kennenlernen von Bausteinen

Bausteinen kann man sich von verschiedenen Seiten nähern, unter anderem so:

***Bottom up*–Vorgehensweise**
Ein Baustein entsteht aus einer Aufgabenserie mit einander ähnlichen Termen oder Figuren. Weitere Bausteinaufrufe erweitern die Kenntnisse.

Baustein als *Black Box*
Ein Baustein ist schon bekannt oder wird vorgegeben und für Anwendungen benutzt

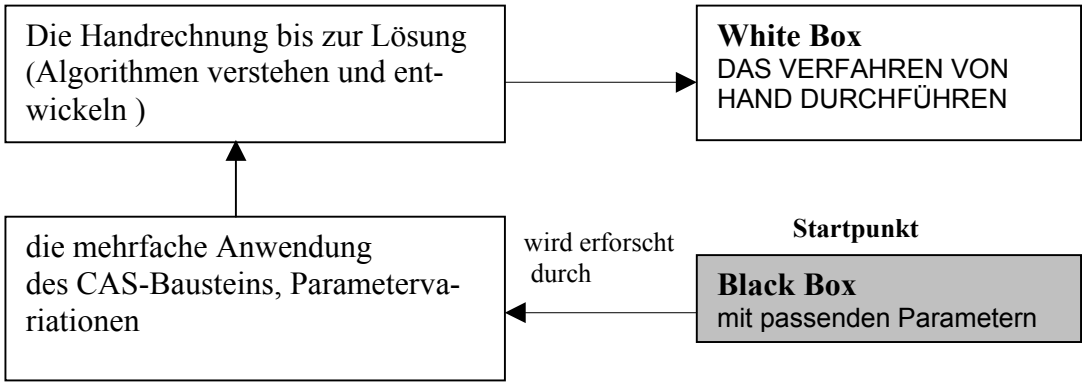
***Top down*-Vorgehensweise**
Ein vorgegebener Baustein wird analysiert durch eine Vielzahl von Aufrufen und Beobachtung der Wirkung.



Die Abbildung zeigt:
(1) Ein Baustein wird schrittweise entwickelt – bis er als Black Box „abgelegt" werden kann.

Aber auch der folgende Weg ist möglich:

(2) Ein Baustein, der in Form einer Black Box vorliegt, wird analysiert – eine White Box entsteht. Hinweis: Man beachte den Startpunkt und die Pfeilrichtungen.



2.1.5.6 Warum Bausteine mit Parametern im Unterricht? Unterrichtserfahrungen

Neben den schon in 2.1.5.3 genannten Vorzügen der Bausteinverwendung im Mathematikunterricht haben meine Unterrichtserfahrungen gezeigt:

Erweiterung des Vorrats an Strategien zur Problembearbeitung
Die Suche nach einem Lösungsansatz ist häufig auch die Suche nach einem geeigneten, schon vorhandenen oder noch zu definierenden Baustein.

Allgemeine Lösungen von Problemen rücken wieder mehr in der Vordergrund
Da Bausteindefinitionen in der Regel mit Parametern erfolgen, werden auch die mit Zahlenwerten gestellten Aufgaben schon beim Lösungsansatz häufig verallgemeinert.

Mehr Selbstständigkeit bei der Arbeit mit mathematischen Inhalten
Ein vorhandener oder gerade definierter Baustein regt wegen seiner Parameter von sich aus zum Forschen und Entdecken an. Offene Unterrichtsformen unterstützen diesen Ansatz.

Bausteine vernetzen
Jeder Baustein enthält eine eigene kleine „Mathematikwelt“, die auch den Schülern deutlich werden kann. An die Stelle vieler Einzelaufgaben tritt nun eine durch den Baustein miteinander vernetzte, zusammengehörige „Mathematikwelt“. **Die bisher übliche sequentielle Abarbeitung des Lehrplans wird aufgebrochen** zugunsten einer mehr gleichzeitigen Bearbeitung von Inhalten aus der jeweiligen „Mathematikwelt”. Dafür bieten sich dann auch projektartige Organisationsformen an.

Der Unterricht und die Unterrichtsinhalte werden insgesamt transparenter
Das Bausteinprinzip führt unabhängig vom gerade behandelten mathematischen Gebiet zu einer gemeinsamen Sichtweise und erweist sich damit als eine wesentliche Leitidee.

Die Konstruktion von Bausteinen bereitet funktionales Programmieren vor.
Insofern wird ein Beitrag zum Informatikunterricht geleistet.

2.1.5.7 Beispiele für Bausteine

Beispiel 1 – Ein Baustein wird veranschaulicht

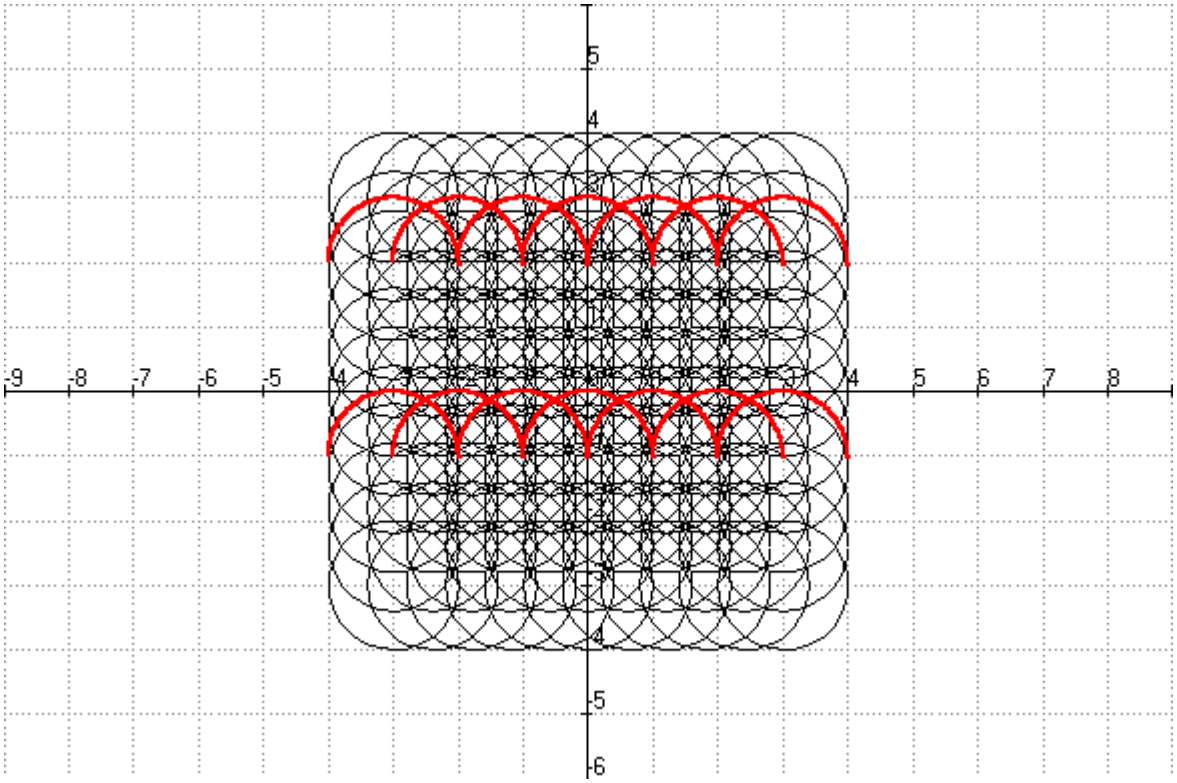


Abb. 2.1.5.7-a: Ein Bild, das auch dem Künstler *Francois Morellet* gefallen würde – erzeugt durch Bausteinaufrufe.

Programmierung im Programm ANIMATO:

f1=	a*cos(t)+b	Bausteindefinition in Parameterdarstellung x(t, a, b)
f2=	a*sin(t)+b	y(t, a, b)
f3=	f1(1,u),f2(1,v)	Bausteinaufruf für viele Kreise, u und v sind Laufvariablen
f4=	f1(1,u),f2(1,-1)	Oben wurde t aus [0, 6.28], gewählt, hier nur aus [0, 3.14],
f5=	f1(1,u),f2(1,2)	daher ergeben sich bei f4 und f5 Halbkreise – sie wurden hier mit einer größeren Strichbreite und grau gezeichnet.

Hinweis: Das Ausblenden des Koordinatensystems würde zu einem Bild führen, wie sie *Francois Morellet* in ähnlicher Weise (ohne Computer) erstellt hat, siehe z. B. *Katalog zur Ausstellung <Morellet>, 22.1.-20.5.2002, Museum Wirth, Künzelsau.*

Beispiel 2 – Visualisierung eines Parabelbausteins

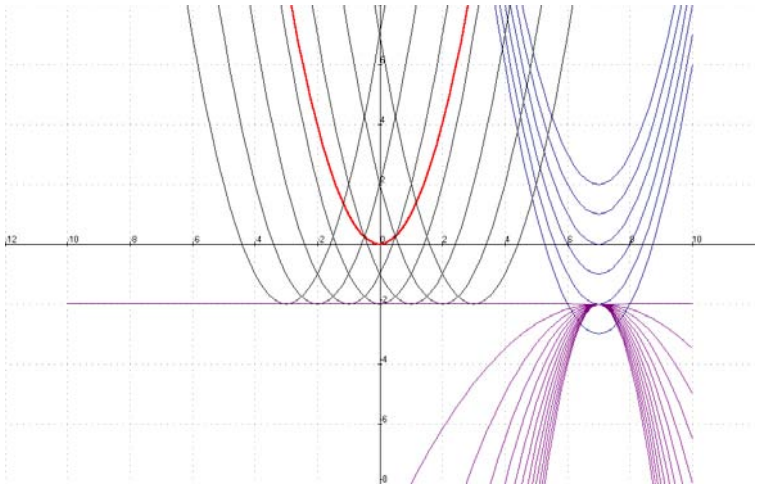


Abb. 2.1.5.7-b: Parametervariation bei einem Parabelbaustein, Strategien zur Erforschung

f1=	a*(x-b)^2+c	der Parabelbaustein
f2=	f1(1,0,0)	Normalparabel
f3=	f1(1,7,u)	a = 1, b = 7, Variation von c, u ∈ {–3, –2, –1, 0, 1, 2}
f4=	f1(1,v,-2)	a = 1, Variation von b, c = –2, v ∈ {–3, –2, –1, 0, 1, 2, 3}
f5=	f1(v,7,-2)	Variation von a, b = 7, c = –2, v ∈ {–2, –1.833, ... 0.333}

Beispiel 3 – Eine Anwendung des Differenzenquotienten-Bausteins

$$\frac{\big(f(x+h)-f(x)\big)}{h} \rightarrow diffq(x,h)$$

Durch Wahl verschiedener h-Werte mit $h \rightarrow 0$ werden die Graphen der Differenzenquotientenfunktionen immer mehr zum Graphen der Ableitungsfunktion ($h \neq 0$).

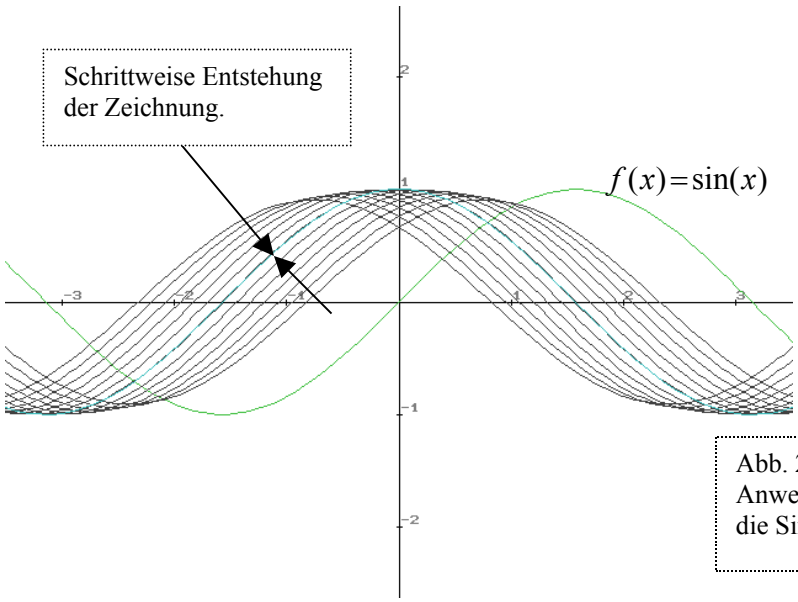


Abb. 2.1.5.7-c:
Anwendung des Bausteins auf die Sinusfunktion

Man vergleiche hierzu auch Seite 161!

2.1.6 Algorithmen

Die engste Verbindung zwischen Mathematik und Informatik ist sicherlich über die in beiden Fächern benötigten Algorithmen gegeben. Der Informatikunterricht realisiert diese in passenden Programmiersprachen, der Mathematikunterricht tut das nur gelegentlich – siehe u. a. Kapitel 2.1.7. Algorithmen durchziehen den gesamten Mathematikunterricht und standen lange im Mittelpunkt des Unterrichts, was dann u. a. zu den langweiligen Übungen zum Handrechnen mit Aufgaben aus den umfangreichen (und in der Regel unmotivierten) Aufgabenplantagen der Schulbücher führte. Angesichts der heute verfügbaren Computeralgebrasysteme ändert sich zur Zeit die Sachlage. Auf umfangreiche Handrechnungen soll in der Regel kein Wert mehr gelegt werden, sie werden dem Rechner überlassen, siehe
W. Herget, H. Heugl, B. Kutzler, E. Lehmann: Welche handwerklichen Rechenkompetenzen sind im CAS-Zeitalter unverzichtbar? In MNU 2001, Heft 8.
Über diesbezügliche Aspekte wurde bereits u.a. in den Kapiteln 1.4 und 2.1.5 nachgedacht. Heute heißt ein Motto: **Weniger Handrechnen – mehr verstehen!**
Dieses Verstehen betrifft auch die Algorithmen im Mathematikunterricht. Es gibt jedoch andere, bessere Methoden als zu versuchen, das Verständnis durch vielfaches Rechnen zu erreichen.

Wege zum Analysieren von Algorithmen sind vor allem
- passende Visualisierungen (Struktogramme, Ablaufpläne, andere grafische Darstellungen),
- die Verwendung von Programmläufen mit geeigneten Eingabedaten, Zwischenergebnissen und Ausgabedaten,
- die Untersuchung des Programm-Quelltextes (ggf. auch nur von Teilen desselben), aber auch das
- Handrechnen einfacher, charakteristischer Beispiele.

Man beachte hierzu auch die Ausführungen in Kapitel 2.1.5 zum Analysieren von Bausteinen, in denen ja Programme bzw. Algorithmen versteckt sind. Abbildung 2.1.6-a nennt weitere Aspekte, die beim Analysieren von Algorithmen von Bedeutung sind.

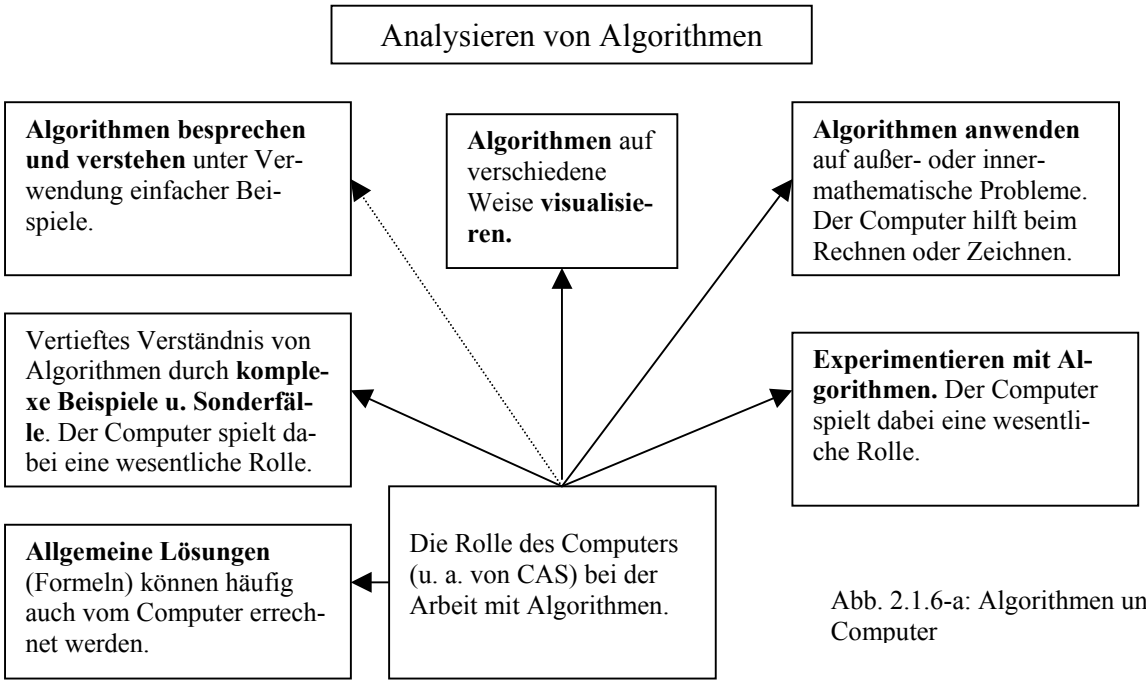


Abb. 2.1.6-a: Algorithmen und Computer

Algorithmen visualisieren

Für kürzere Algorithmen sind u. a. geeignet
- Programmablaufpläne,
- Baumdiagramme,
- Struktogramme,
- Übergangsgraphen (z. B. bei Markow-Ketten und Automaten – siehe Kapitel 3.3),
- Animationen zur Darstellung der algorithmischen Abläufe.

Für umfangreichere Algorithmen eignen sich gut
- Struktogramme (durch Unterscheidung verschiedener Bearbeitungstiefen),
- reduzierte Baumdiagramme (hierbei werden Pfade eines Baumdiagramms zusammengefasst, siehe Abb. 2.1.6-f),
- Animationen zur Darstellung der algorithmischen Abläufe.

Struktogramme
Ein bewährtes Hilfsmittel zur Veranschaulichung von Algorithmen sind Struktogramme.
- Struktogramme sind häufig eine Vorstufe der Programmierung. Sie unterstützen das strukturierte Entwerfen und Programmieren sowie das Analysieren von Algorithmen.
- Sie zeigen den Ablauf eines Algorithmus in übersichtlicher Form und beschreiben ihn in Kurzform
- Sie dienen zur Dokumentation von Algorithmen.

Struktogrammsymbole
Die Aktionen eines Algorithmus werden in geeigneter Form in Strukturblöcke eingetragen. Diese Strukturblöcke können je nach Problemstellung beliebig ineinander geschachtelt sein.
Die Darstellung kann in unterschiedlicher Beschreibungstiefe erfolgen, indem ein grobes Struktogramm zunehmend verfeinert wird, bis eine Stufe erreicht ist, die sich leicht in einer Programmiersprache realisieren lässt. Das müssen dann nicht unbedingt die elementaren Anweisungen der Sprache sein, gelegentlich ist die Übersetzungsstufe schon erreicht, wenn sich für den Teilalgorithmus ein passender Baustein findet.
Ein Beispiel für die Anwendung dieser Technik auf ein mathematisches Problem erfolgt etwas später in diesem Kapitel.

Beispiel 1: Visualisierung eines Algorithmus zum Gleichung lösen

Die Beispiel stammt aus dem Unterricht von Klasse 9.

Alte Aufgabenstellung:
Bestimme die Lösungsmenge: $(x + 3)^2 = (x - 5)^2$.
Hinweis: In einer derartigen Aufgabenformulierung kommt es in der Regel lediglich auf das Rechnen an.

Die Umformulierung der Aufgabe zielt auf die „neue Aufgabenkultur" ab. Sie berücksichtigt dabei die Aspekte
- wenig Handrechnen,
- Benutzung eines CAS,
- Fehler finden,
- Rechnung kontrollieren,
- Lösungsweg bewerten,
- Verstehen der zugrunde liegenden Mathematik.

Neue Aufgabenstellung:
Mathias und seine Mitschüler erhalten die folgende Aufgabe:

Bestimme die Lösungsmenge: $(x + 3)^2 = (x – 5)^2$. Die Bearbeitung von Mathias sieht so aus:

$(x + 3)^2 = (x – 5)^2$ // √
$(x + 3) = (x – 5)$ // –x
$3 = – 5$, also gilt L = { }

Was sagt das CAS? Und was sagst du?

CAS-Lösung (mit dem TI-92)

Eingaben	Ausgaben, Kommentare
$(x + 3)^2 = (x – 5)^2$ → gl(x) Solve(gl(x) , x)	done x = 1 Die Lösungsmenge ist L = {1}. Es gibt nur eine Lösung.

Eine mögliche Schülerlösung:
Wie man durch Einsetzen erkennen kann, ist x = 1 eine Lösung, denn g(1) ist $(1 + 3)^2 = (1 – 5)^2$; also 16 = 16, das ist eine wahre Aussage. – Mathias darf aus der Gleichung nicht auf beiden Seiten die Wurzel ziehen, denn ...

Abbildung 2.1.6-a zeigt (siehe folgende Seite), welche Situation hier vorliegt:

- Es geht um den Schnittpunkt der beiden Parabeln, der offenbar bei S(1, 16) liegt. x=1 ist die Lösung der Gleichung.

- Bei Äquivalenzumformungen müsste die Lösung stets auf der Geraden x = 1 bleiben.

- Durch das (falsche) Wurzelziehen ergeben sich nur die Geraden mit den Gleichungen y = x–5 und y = x+3, die parallel sind, also keinen Schnittpunkt haben (Lösungsmenge leer).

- Betrachtet man aber alle hier möglichen Geradengleichungen, also y = x–5 und y = –(x–5) sowie y = x+3 und y = –(x+3), so gibt es Schnittpunkte, nämlich bei x = 1.

Sicher werden die Schüler auch die Lösung durch Ausmultiplizieren wählen:
$(x + 3)^2 = (x – 5)^2$
$x^2+6x+9 = x^2–10x+25$
$16x = 16$
$x = 1$.

Visualisierung A – graphische Darstellung der Terme im Koordinatensystem in Form einer Animation

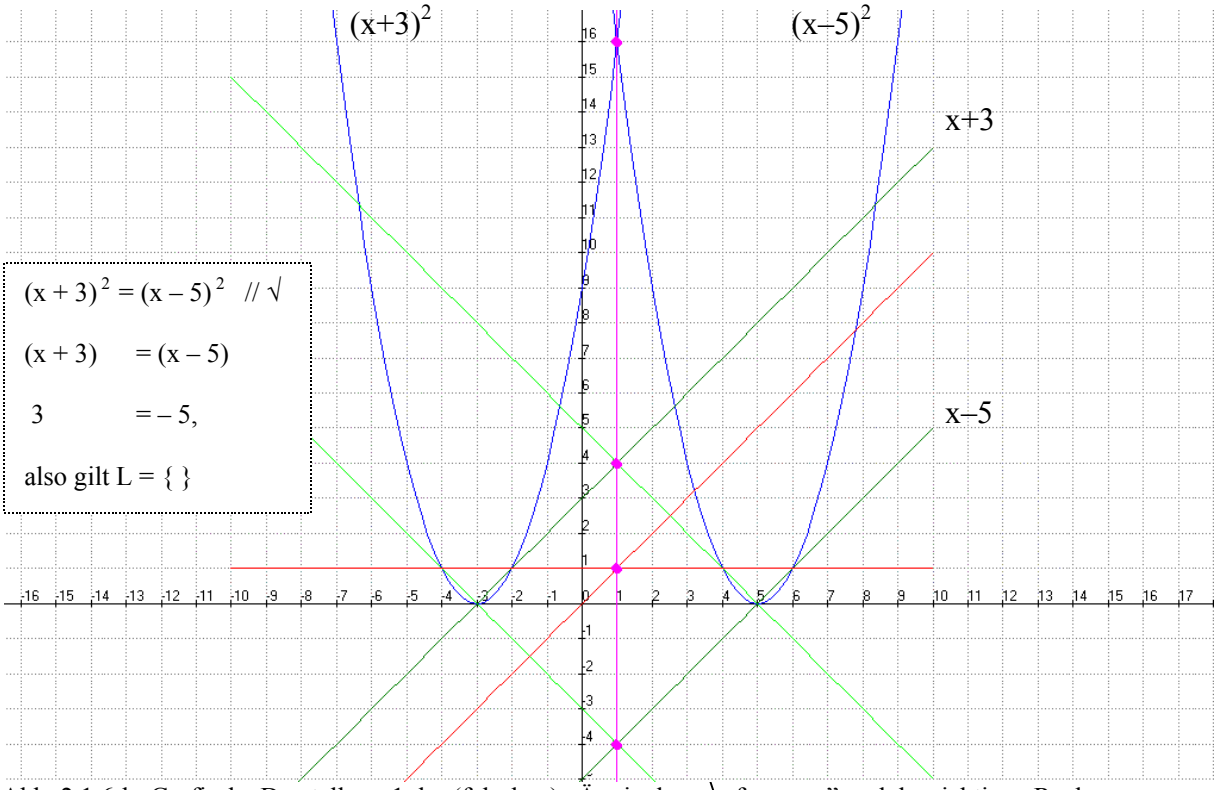


Abb. 2.1.6-b: Grafische Darstellung 1 der (falschen) „Äquivalenzumformung" und der richtigen Rechnung

An dieser Visualisierung lassen sich etliche Sachverhalte ablesen, die das mathematische Verständnis der Situation wesentlich bereichern.

1) Der Schnittpunkt der Parabeln liegt bei S(1, 16). Nach den Äquivalenzumformungen muss sich also x = 1 ergeben.

2) Die Umformung von Mathias führt zu zwei Geradengleichungen y = x+3 und y = x–5. Die Geraden schneiden sich jedoch nicht mehr. Demnach müsste die Lösungsmenge leer sein. Wo ist die Lösung x = 1 geblieben?

3) Nimmt man die beiden Geraden mit den Gleichungen y = –(x+3) und y = –(x–5) hinzu (die Terme –(x+3) und –(x–5) würden sich ja beim Wurzelziehen auch ergeben können) und kombiniert die vier Geraden anders, so tauchen die Schnittpunkte mit x = 1 wieder auf.

4) Weitere Umformung ergibt schließlich den Punkt P(1,1). Die Lösungsmenge ist also L = { 1}.

5) Wichtige Erkenntnis: Bei der grafischen Darstellung aller hier durchgeführten Äquivalenzumformungen liegen die Schnittpunkte aller Objekte auf der senkrechten Geraden mit der Gleichung x = 1. – Ist das bei allen Gleichungen und ihren Umformungen so?

Mit dem Programm ANIMATO lässt sich Abbildung 2.1.6-b schrittweise in Form einer Animation erzeugen, so dass die Vorgänge für den Schüler noch transparenter werden.

Visualisierung B – Struktogramm

.

$(x + 3)^2 = (x – 5)^2$ // √			
Fallunterscheidung			
Fall 1 $(x + 3) = (x – 5)$	Fall 2 $(x + 3) = –(x – 5)$	Fall 3 $–(x + 3) = –(x – 5)$	Fall 4 $–(x + 3) = (x – 5)$
$L_1 = \{\ \}$ Lösung falsch	$L_2 = \{1\}$ Lösung richtig	$L_3 = \{\ \}$ Lösung falsch	$L_4 = \{1\}$ Lösung richtig

Abb. 2.1.6-c: Darstellung der „Äquivalenzumformung" in einem Struktogramm

Auch diese Visualisierung trägt zu einem tieferen Verständnis des Sachverhalts bei.

Visualisierung C – Baumdiagramm

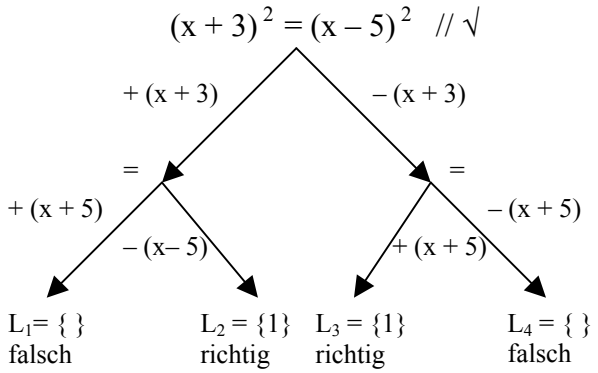


Abb. 2.1.6-d: Darstellung der Äquivalenzumformung in einem Baumdiagramm

Beispiel 2 – Visualisierung eines Algorithmus als Übergangsgraph

Wir betrachten das Kaufverhalten von 1000 Käufern über einen längeren Zeitraum hin und wollen Voraussagen über den langfristigen Anteil der Käufer von Tageszeitung A gewinnen.

Die festgestellen Daten:

Tägliche Übergänge zwischen den Zuständen:

Kauf von A	→ danach wieder Kauf von A	80 % der Leser
Kauf von A	→ danach Kauf von nA	20 % der Leser
Kauf von nA	→ danach Kauf von A	50 % der Leser
Kauf von nA	→ danach wieder Kauf von nA	50 % der Leser

Hinweis: nA bedeutet, dass A nicht gekauft wird.

Visualisierung A

Die Übergänge zwischen den zwei Zuständen werden durch eine Matrix S beschrieben:

	A	nA	
A	0.8	0.2	= S
nA	0.5	0.5	

Visualisierung B

Die Übergänge zwischen den zwei Zuständen werden durch einen Übergangsgraphen beschrieben:

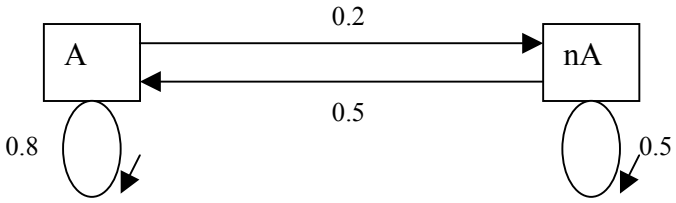


Abb. 2.1.6-e: Algorithmus in Form eines Übergangsgraphen

Der Algorithmus besteht in dem mehrmaligen Durchlaufen des Graphen (da der Vorgang über einen längeren Zeitraum betrachtet wird) und der Rechnung für die langfristige prozentuale Verteilung der Käufer.

Visualisierung C

Die Übergänge zwischen den zwei Zuständen werden durch ein reduziertes Baumdiagramm beschrieben.

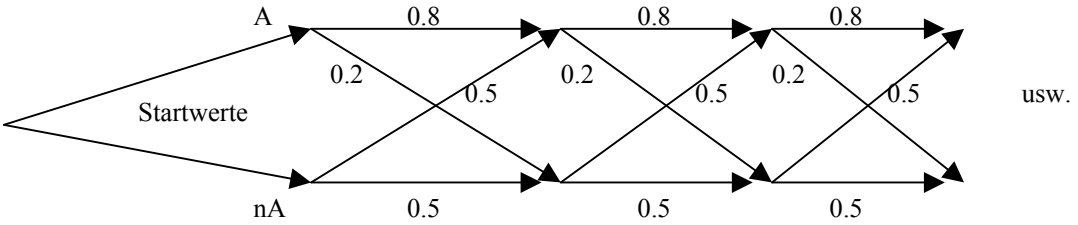


Abb. 2.1.6-f: Algorithmus als reduziertes Baumdiagramm

Der reduzierte Baum kürzt den ausführlichen Baum mit seinen vielen Verzweigungen ab. Anders als bei Visualisierung B lassen sich die einzelnen Stufen des Prozesses gut erkennen und rechnerisch leichter umsetzen.

2.1.7 Programmieren im Mathematikunterricht

2.1.7.1 Was ist „Programmieren”?

> *„Unter Programmieren versteht man zum einen den Vorgang der Programmerstellung und zum anderen das Teilgebiet der Informatik, das die Methoden und Denkweisen beim Entwickeln von Programmen umfasst.” (Informatik-Duden, Duden-Verlag 1993, S.549)*

Computeralgebrasysteme, z. B. DERIVE 5, stellen in der Regel Elemente des imperativen und des funktionalen Programmierens zur Verfügung.

Prozedurale oder imperative Programmiersprachen sind nach einem ablauforientierten Sprachkonzept konstruiert. Ein Programm stellt eine Folge von Anweisungen dar. Der Programmierer muss also in zeitlichen Abläufen denken und seine **Problemlösung aus nacheinander auszuführenden Schritten** aufbauen. Charakteristisch sind das Variablenkonzept und Begriffe wie Unterprogramme (Prozeduren), Wertzuweisungen, Schleifen, bedingte Verzweigungen usw.

„Programme berechnen Funktionen, die Eingabedaten in Ausgabedaten abbilden. In der funktionalen Programmierung beschreibt man daher die Beziehungen zwischen Ein- und Ausgabedaten mit Hilfe mathematischer Ausdrücke, indem man elementare Ausdrücke für einfache Funktionen zugrundelegt und hieraus mit Operationen, die auf Funktionen definiert sind, komplexere Funktionen darstellt. ... Das wichtigste Konstruktionsprinzip ist hierbei die Rekursion. **Ein Programm besteht aus einer Menge von Ausdrücken, die Funktionen definieren.** Eine Berechnung ist die Anwendung einer Funktion auf eine Liste von Werten oder Ausdrücken.” (*Informatik-Duden, Duden-Verlag 1993, S. 545*.)

Damit ist das funktionale Programmieren näher an den Begrifflichkeiten der Mathematik als das imperative Programmieren.

An den Erläuterungen erkennt man, dass der Entwurf von Programmen mit bestimmten Methoden und Denkweisen (Problemzerlegung, strukturiertes Programmieren, objektorientiertes Programmieren, funktionales Programmieren usw.) erfolgt, von denen der Mathematikunterricht profitieren kann, die aber dort teilweise nur bedingt eigenes Thema sein können. Der Vorgang der Programmerstellung soll jedoch im Mathematikunterricht nicht in den Vordergrund rücken.

Kennzeichen des Programmierens mathematischer Fragestellungen sind u.a.:

- Analysieren von Algorithmen,
- Finden geeigneter Parameter,
- Konstruieren von Prozeduren und Funktionen (Bausteinen),
- Konstruieren von Wiederholungsschleifen,
- Berücksichtigung spezieller Bedingungen (Fallunterscheidungen).

Wozu im Mathematikunterricht programmieren?
Die Programmierung erfolgt u. a.

- zur Lösung spezieller Probleme, die sich nicht mit dem CAS elementar (z. B. durch Verwendung vordefinierter oder selbstdefinierter Bausteine) bearbeiten lassen,
- Herstellung von Black Boxes zum Zweck der
  – Mehrfachverwendung und zur
  – Förderung experimentellen Arbeitens,
- zur Analyse von Algorithmen.

Bedenken gegen das Programmieren im Mathematikunterricht

- Viele Lehrer scheuen sich vor dem Programmieren! Die Schülerspezialisten können es besser!
- Vielen Schüler fällt das Programmieren schwer!
- Programmieren führt wegen mancher Misserfolge zur Demotivation von Schülern!
- Programmieren verstärkt die Unterschiede in der Leistungsfähigkeit von Schüler!
- Programmieren ist zeitaufwendig und kann sich leicht verselbstständigen!
- Es gibt im CAS genügend andere Möglichkeiten, um Problemlösungen zu finden oder „Programmierlernziele" zu erreichen!

2.1.7.2 Einführende Beispiele – Programmieren früher und heute

Die folgenden Ausführungen beschäftigen sich mit einem speziellen Aspekt von CAS bzw. anderer Software, dem Programmieren in diesen Systemen im Mathematikunterricht.

Hier gibt es Anzeichen, dass sich bei einigen Lehrenden mit dem übertriebenen Schreibenlassen von Programmen eine gefährliche Praxis einschleichen könnte, die dem Mathematikunterricht eher schadet als nutzt. Die Problematik des Programmierens muss sehr diffenziert untersucht werden – sie ist u. a. abhängig von der jeweiligen Lerngruppe, von der Lehrerkompetenz und von den Unterrichtszielen. Bezüglich des Programmierens im Mathematikunterricht sind auch frühere Erfahrungen wichtig. Diese wurden in Kapitel 1.3 bereits aufgezeigt. Ergänzend werden zwei weitere Beispiele gebracht, die nun aber auch mit den heutigen Möglichkeiten angegangen werden und damit den Wandel auf diesem Gebiet kennzeichnen.

Beispiel 1: Quadratische Gleichung

1988: PASCAL-Programm	2001: Heute könnten die Schüler mit dem vordefinierten Baustein SOLVE arbeiten. Baustein im CAS des TI-92
PROGRAM quadratische_gleichung; *USES Crt;* *VAR p,q,diskriminante,x1,x2: REAL;* *BEGIN* *Clrscr;* *WRITE('p= '); READLN(p);* *WRITE('q= '); READLN(q);* *diskriminante:=p*p/4-q;* *IF diskriminante>=0 THEN* *BEGIN* *x1:=-p/2+sqrt(diskriminante);*	*solve(x^2 + p*x + q = 0, x) → pqformel(p,q)* *Aufrufe des Bausteins wie z.B.* *pqformel(3,-4), pqformel(-3,1)* *lösen dann sofort die entsprechende quadratische Gleichung oder geben die Meldung „false" (keine Lösung in R) aus.*

`x2:=-p/2-sqrt(diskriminante);`
`WRITELN('Lösungen:');`
`WRITELN('x1=',x1:10:4);`
`WRITELN('x2=',x2:10:4);`
`END`
`ELSE WRITE('Keine Lösungen!');`
`READLN;`
`END.`	

Abb. 2.1.7.1-a: Ein „10-Zeilen-Programm" aus den 80-er Jahren und eine heutige Lösung

Beispiel 2: Länge von Breitenkreisen
Das folgende kleine BASIC-Programm stammt aus *[Leh88].*

Es berechnet die Länge von Breitenkreisen auf der Erdkugel.

1988, BASIC-Programm	2001: Heute könnten die Schüler einfach eine Folge definieren (Syntax des TI-92).

`10 m = 2*3.1416*6370`
`20 a = 0`
`30 IF a > 90 THEN 90`
`40 w = 3.1416*a / 180`
`50 u = m*cos(w)`
`60 PRINT a, u`
`70 a = a+5`
`80 GOTO 30`
`90 END`
	`seq(2*π*6370*cos(w),w,0,90,5)`
	
	*Kommentar: Hier wird eine Zahlenfolge aus dem Term 2*π*6370*cos(w) berechnet, die Werte für w laufen von 0° bis 90° mit der Schrittweite 5°.*

Abb. 2.1.7.1-b: Ein „10-Zeilen-Programm" 1988 und die heutige Lösung

Weitere Beispiele für solche kleinen Programme finden sich an vielen Stellen, beispielsweise in den Büchern von Arthur Engel, z. B. in *[Eng77].*

Im Vorwort heißt es: *„Durch die weite Verbreitung der Computer und Taschenrechner ist die Zeit reif geworden für die nächste Reform unter dem Schlagwort „algorithmisches Denken". Der Begriff des Algorithmus sollte als Leitbegriff für die Schulmathematik dienen. Wir müssen den gesamten Schulstoff vom algorithmischen Standpunkt neu durchdenken."*

Vierzehn Jahre später liest man auf dem Rückdeckel des Buches

Arthur Engel: Mathematisches Experimentieren mit dem PC, Klett 1991.

„Der PC bietet neue heuristische Methoden – die Mathematik liefert effiziente Algorithmen".

Engel rückt damit zwei Aspekte in den Vordergrund, die für die Berücksichtigung informatischer Aspekte im Mathematikunterricht wichtig sind:
- Algorithmisches Denken, effiziente Algorithmen,
- heuristische Methoden.

Für die beiden oben genannten Bücher war für den Lehrer die Kenntnis einer Programmiersprache unumgänglich. Das waren in diesem Fall BASIC (1977) bzw. TURBO-PASCAL (1991). Die Entwicklung zeigte dann aber:

Die kleinen "10-Zeilen-Programme", dennoch oft mit relativ großer Wirkung und **häufig gedacht zum Experimentieren mit mathematischen Ansätzen**, konnten sich jedoch in der Schule wegen fehlender Hardware und vor allem wegen fehlender Akzeptanz durch die Lehrer nicht in breitem Maße durchsetzen. – Siehe Kapitel 1.3.1.

Die heutige Hardwaresituation ist jedoch viel günstiger und auch die Akzeptanz ist durch zahlreiche Fortbildungsveranstaltungen und die Entwicklungen um TIMSS und PISA erheblich größer geworden!

2.1.7.3 Programmieren im CAS

Aus Sicht der Schulmathematik brachten die Computeralgebrasysteme eine Antwort auf die oben genannten Probleme bezüglich der Akzeptanz des Programmierens.

- Sie enthalten nämlich **fertige Bausteine,** in denen die Algorithmen versteckt sind, und lassen die **Definition neuer Bausteine** zu. Damit lässt sich der Umfang des Programmierens erheblich zurückdrängen! Mittels der Bausteine kann es nun auf einer höheren, aber doch einfacheren Ebene stattfinden.
- Viele Problemlösungen lassen sich durch kleine funktionale Programme finden. **Elemente funktionalen Programmierens** finden sich in CAS-Systemen, aber auch in anderen Softwaresystemen, wie z. B. dem in dieser Arbeit mehrfach benutzten Animationsprogramm ANIMATO oder dem Raytracing-Programm POVRAY. Diese Systeme vereinfachen für den Schüler das Programmieren auch dadurch, dass sich Eingaben sofort grafisch visualisieren lassen. So ist eine sofortige Kontrolle der Programmierarbeit möglich.

Einige Beispiele für Bausteine hierzu wurden oben bereits notiert. Sie werden hier aus CAS-Sicht wiederholt.

Beispiel	**Bausteindefinition**	**Bausteinaufrufe**
1. Lösung einer quadratischen Gleichung mit Hilfe eines Bausteins und unter Benutzung des vordefinierten „solve"-Befehls	define formel(p,q) = solve(x^2 + p*x + q = 0, x)	formel(3, –4), formel(1, 1)
2. Berechnung einer Tabelle mit der Länge von Breitenkreisen unter Benutzung des vordefinierten „seq"-Befehls (Sequenz)	Verwendung von „seq", w ist aktuelles Argument, das von 0 bis 90 läuft.	seq(2*π*6370*cos(w), w,0,90,5)
3. Berechnung der Länge von Breitenkreisen mit Hilfe eines Bausteins und unter Benutzung des „seq"-Befehls	define folge(k,s) = seq(2*π*k*cos(w),w,0,90,s).	folge(6370, 5), weitere Aufrufe.

- seq(2*π*6370*cos(w),w,0,90,5) entspricht einem kleinem Programm, das man früher ausführlicher aufschreiben musste, siehe oben. Ersetzt man beispielsweise den Erdradius durch einen beliebigen Kugelradius k und die Schrittweite für die Breitenkreise durch einen Parameter s, so wird dieses Programm sofort flexibler und ist gut zum Experimentieren geeignet.

Mit der Verfügbarkeit von CAS ist also die Notwendigkeit des Schreibens von Programmen drastisch gesunken. Das ist für den Mathematikunterricht eine sehr positive Entwicklung.

Vernachlässigung algorithmischer Aspekte?
Mit der Möglichkeit, Black Boxes zu verwenden, verliert allerdings der durch das Programmieren in den Vordergrund gerückte algorithmische Aspekt an Bedeutung. Es ist jedoch für das Verständnis mathematischer Sachverhalte durch die Schüler unumgänglich, ausgewählte Algorithmen zu kennen. Hier ist in erster Linie an die mathematischen Standardalgorithmen im Schulunterricht zu denken. Will man Algorithmen studieren, so ist man jedoch nicht unbedingt auf die Realisierung in einer Programmiersprache angewiesen. Vielmehr geht das auch durch grafische Darstellungsformen wie z. B. Flussdiagramme und Struktogramme oder durch die Analyse von Bausteinen. Hierüber wurden bereits oben Aussagen gemacht (siehe Kapitel 2.1.5 und Kapitel 2.1.6).
Unterrichtserfahrungen und viele Beiträge zum Computereinsatz im Mathematikunterricht weisen nach, dass sich mit dem Computer die Möglichkeiten experimentellen Arbeitens erheblich vergrößert haben, siehe z. B. *[Eng91], [Leh94a], [Leh99b], [Tie77], [Böh02], [Schum01], [Her90]*. Damit verbessert sich auch die Situation bezüglich der Verwendung heuristischer Methoden.

CAS-Programmierfans
Angesichts dieser Entwicklung verwundert es, dass einige Lehrer sich offensichtlich dem Programmieren in CAS verschrieben haben. Hierfür ist eine Programmiersprache nötig, die z. B. vom CAS des TI-92 in einer Form angeboten wird, die bei längeren Programmen leicht zu den längst von der Informatik abgelehnten „Spaghetti-Codes" führt.

Die folgenden Programmbeispiele dienen dazu, einige Abgrenzungen zwischen Problemlösungen ohne bzw. mit Programmen zu ermöglichen.

Das Schreiben von Programmen im Programmeditor des TI-92
Schon bei dem folgenden kleinen Programm sind einige Programmierkenntnisse nötig.

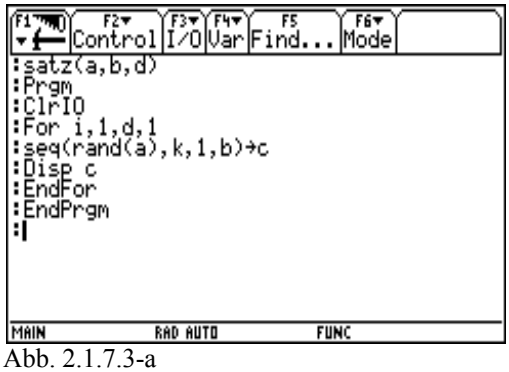


Abb. 2.1.7.3-a

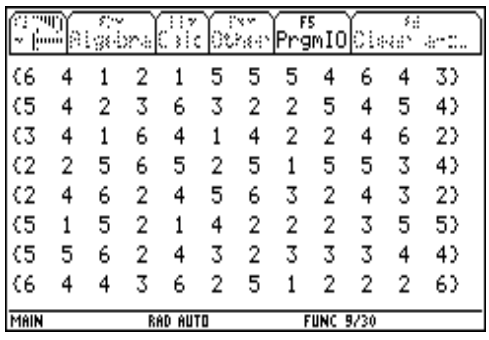


Abb. 2.1.7.3-b

Beispiel 1: satz(a,b,d), **Erzeugung von Würfelzahlen**
Mit diesem kleinen Programm können u.a. Würfelzahlen erzeugt werden, aus denen man ablesen kann, wann ein vollständiger Satz erreicht ist (jede Würfelzahl muss mindestens einmal vorkommen). Der Aufruf: satz(6, 12, 8) liefert die obige Tabelle.

Beispiel 2: trapez(a,b,n), **Flächeninhaltsberechnung**
Hier wird eine Trapezformel zur näherungsweisen Flächeninhaltsberechnung zwischen Graph und x-Achse definiert. Der Anspruch ist hier schon erhöht.

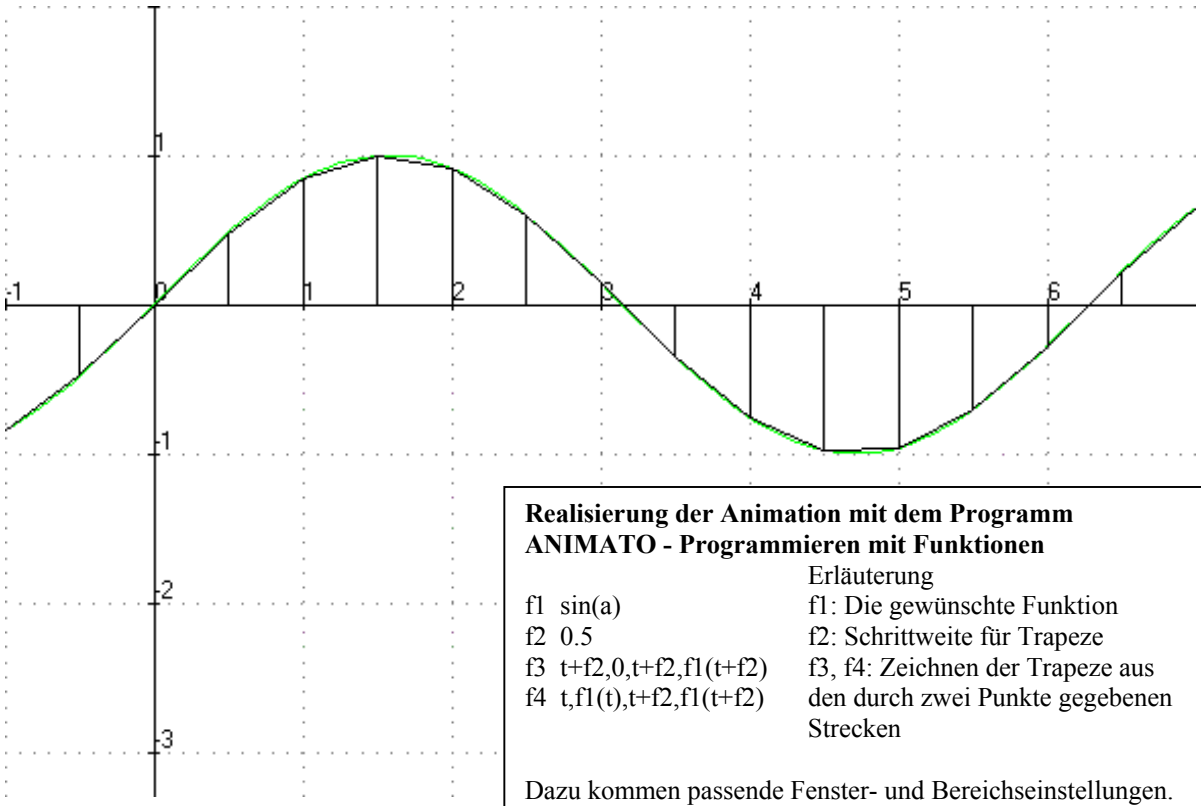
```
Func
    Local h, term1, term2, i
    (b–a) / n → h
    f(a) + f(b) → term1
    Σ(f(a+i*h), i, 1, n–1) → term2
    h/2*(term1 + 2*term2)
EndFunc
```

Beispiel 3: drawtrap(a,b,n), **Trapeze auf den TI-Bildschirm zeichnen**
Hier wird die Grenze des Programmierens für einen normalen Mathematikkurs deutlich überschritten. Ein derartiges Programm sollte bestenfalls vom Lehrer oder von kompetenten Schülern (z. B. aus Informatikkursen) hergestellt werden. Es kann dann aber von allen benutzt oder auch als Demonstrationsprogramm eingesetzt werden.

```
drawtrap(a,b,n),
Prgm
Local h,i
@Festlegung des x-Achsen-Bereiches
a-(b-a)/10 → xmin
a+(b-a)/10 → xmax
@Zeichnen der Funktion
CrlDraw
Graph f(x)
(b-a)/n → h
@Zeichnen der Trapeze
For i, 0, n-1
Line a+i*h, 0, a+i*h, f(a+i*h)
Line a+(i+1)*h, 0, a+(i+1)*h, f(a+(i+1)*h)
Line a+i*h, f(a+i*h), a+(i+1)*h, f(a+(i+1)*h)
EndFor
EndPrgm
```

*(*Aus: *Günter Schmidt u. a., Numerische Verfahren mit dem TI-92, Funktionen, Programme, Graphen, Texas Instruments, 1999)*

Eine für die Schüler mit wenigen Befehlen zu erstellende Visualisierung kann auf einfache Weise dem Programm ANIMATO erfolgen.



Realisierung der Animation mit dem Programm ANIMATO - Programmieren mit Funktionen

	Erläuterung
f1 sin(a)	f1: Die gewünschte Funktion
f2 0.5	f2: Schrittweite für Trapeze
f3 t+f2,0,t+f2,f1(t+f2)	f3, f4: Zeichnen der Trapeze aus
f4 t,f1(t),t+f2,f1(t+f2)	den durch zwei Punkte gegebenen Strecken

Dazu kommen passende Fenster- und Bereichseinstellungen.

Abb.2.1.7.3-c: Trapeze unter der Sinuskurve

Die Beispiele 4 und 5 zeigen einige Elemente funktionalen Programmierens, die für den Unterricht geeignet sind.

Beispiel 4 – Programmieren mit Funktionen 1

Problemstellung: Enthält eine vorgegebene Bit-Folge eine gerade oder ungerade Anvon zahl Einsen? – Beispiel: (0 0 1 1 1 0 1) enthält eine gerade Anzahl von Einsen.

Das Problem kann mit einem endlichen Automaten (Akzeptor, erkennender Automat) bearbeitet werden. Dieser lässt sich folgendermaßen darstellen (vergl. Kap. 3.3.1/3.3.2):

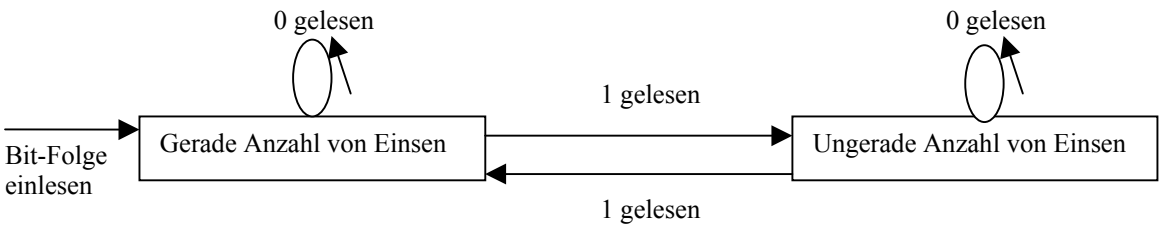


Abb. 2.1.7.3-d: Endlicher Akzeptor zur Überprüfung von Bit-Folgen

Programmierung in DERIVE 5

Derive 5 Eingaben und Ausgaben	Erläuterungen
#1 f2(n) := If n = 0 0 f2(n - 1) + FLOOR(2·RANDOM(1)) #2 f2(7) #3 5 #4 VECTOR(f2(10), i, 1, 10) #5 [5, 4, 6, 6, 3, 5, 5, 5, 5, 6] #6 VECTOR(FLOOR(2·RANDOM(1)), i, 1, 10) #7 [0, 0, 0, 1, 0, 1, 1, 0, 1, 0]	Definition der Funktion f2(n): Anfangswert f2(0)=0, ansonsten den Vorgänger nehmen und jeweils eine Zufallszahl 0 oder 1 addieren. n ist die Wortlänge der Bit-Folge Test: Eine Bit-Folge der Länge 7 enthält hier 5 Einsen. 10 Bitfolgen (i=1..10) der Länge 10 enthielten 5, 4, 6, ... Einsen. Beispiel für eine Bit-Folge der Länge 10. Hier sind es 6 Nullen und 4 Einsen.

Veranschaulichung des Akzeptors mit ANIMATO (Einlesen einer zufälligen Bit-Folge).

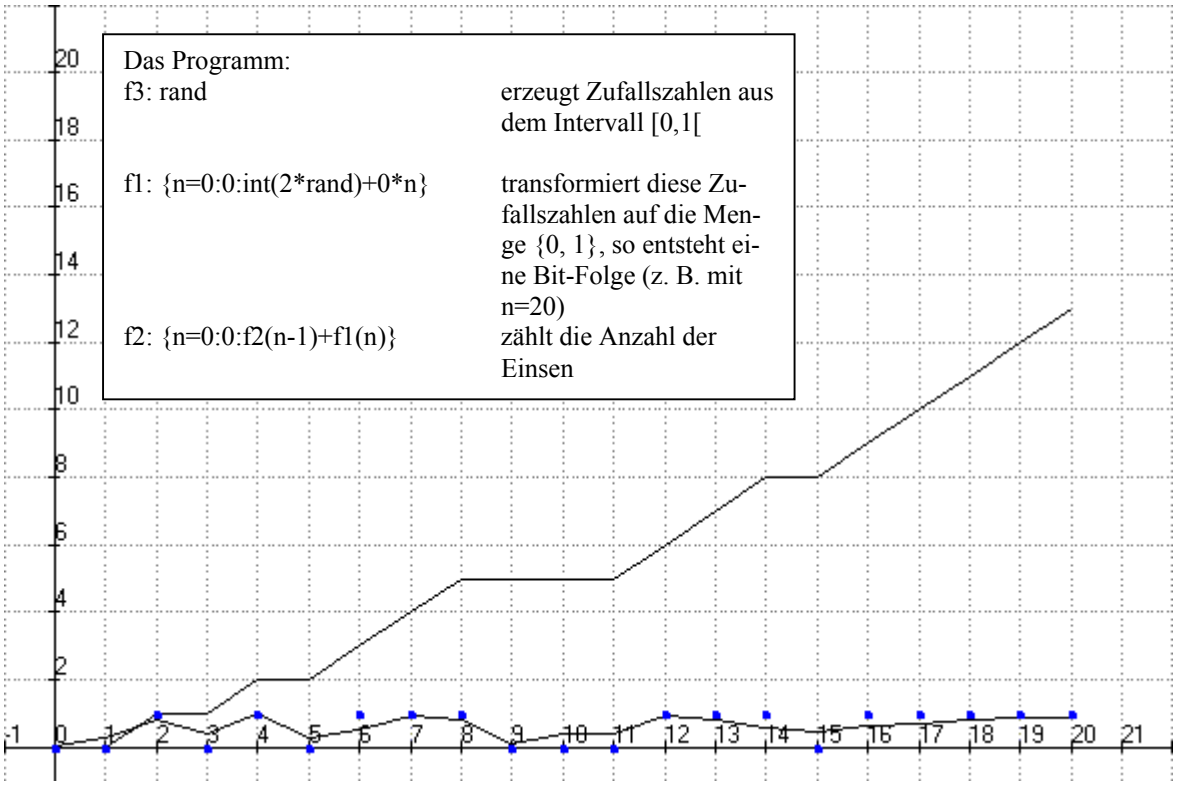


Abb. 2.1.7.3-e: Graphische Darstellung einer Bit-Folge (Punkte) und einer rekursiv definierten Funktion zum Zählen der Einsen (hier sind es 13 Einsen in der Bit-Folge). Der untere Graph zeigt die Zufallswerte aus [0,1[.

Das Programmieren besteht in diesem Fall aus einer logischen Aneinanderreihung von Funktionstermen (oder auch Relationstermen) und der Wahl sinnvoller Darstellungsbereiche. An der folgenden Wertetabelle lässt sich der Vorgang nachvollziehen.

x	f1	f2	f3	x	f1	f2	f3
0	0	0	0.93917661	11	0	4	0.11960204
1	1	1	0.69960631	12	0	4	0.22486038
2	1	2	0.75786615	13	0	4	0.20084231
3	0	2	0.064851833	14	0	4	0.44569231
4	0	2	0.28547014	15	0	4	0.48619037
5	1	3	0.89120151	16	0	4	0.25379192
6	0	3	0.24637593	17	0	4	0.32703635
7	0	3	0.17682424	18	0	4	0.39082003
8	0	3	0.02026429	19	1	5	0.62837611
9	1	4	0.8219245	20	1	6	0.55226295
10	0	4	0.24634541				

Hinweis: Das in Beispiel 4 bearbeitete Problem wird in
Puhlmann, Hermann: Funktionales Programmieren, eine organische Verbindung von Informatikunterricht und Mathematik, in LOGIN 1998, Heft 2, S. 46 f.
angesprochen und dort mit einem kleinen Programm in der funktionalen Sprache Standard-ML bearbeitet.

Beispiel 5 – Programmieren mit Funktionen 2 – Rekursion
Typisch für funktionales Programmieren ist die Verwendung rekursiv definierter Funktionen. Beispiel 5 berücksichtigt diesen Aspekt, verwendet außerdem das Bausteinprinzip und bereitet auf die Unterrichtseinheit über Markow-Ketten in Kapitel 3.3 vor. Abbildung 2.1.7.3-f stellt den Verlauf einer Markow-Kette dar.

Wir betrachten die Markow-Kette mit der Übergangsmatrix S = $\begin{matrix} p(11) & p(12) \\ p(21) & p(22) \end{matrix}$

und der (n–1)-ten Verteilung V(n-1) = $\begin{matrix} x(n-1) & y(n-1) \end{matrix}$.

Für die n-te Verteilung gilt V(n) = V(n–1) * S, also

x(n) = p(11)*x(n–1) + p(21)*y(n–1)
x(n) = p(11)*x(n–1) + p(21)*(1–x(n–1))

Daraus entsteht die Rekursionsformel **x(n) = (p(11) – p(21))*x(n–1) + p(21)**.

In ANIMATO kann man dazu den Baustein f1 definieren:

f1: {a=0:u:(b-c)*f1(a-1)+c}	Wenn a = 0, dann Anfangswert u (u kann man mehrere Anfangswerte durchlaufen lassen!), sonst (b–c)*f1(a–1)+c} rechnen (gemäß der obigen Rekursionsformel).
f2: f1(a, 0.1, 0.85)	Bei f2 wird der Baustein aufgerufen.
f5: (b-c)^a*u + + c*(1-(b–c)^a)/(1-(b-c))	f5 und f6 dienen zur Kontrolle von f1 und f2. – Hier steht ein entsprechender Baustein in expliziter Darstellung.
f6: f5(a, 0.1, 0.85)	f6: ruft diesen Baustein mit den gleichen Werten wie oben auf. Dabei gelten folgende, an anderer Stelle vorgenommene Einstellungen: a: Von 0 bis 20, Schrittweite 1 u: 0, 1/3, 2/3, 1, Anfangswerte

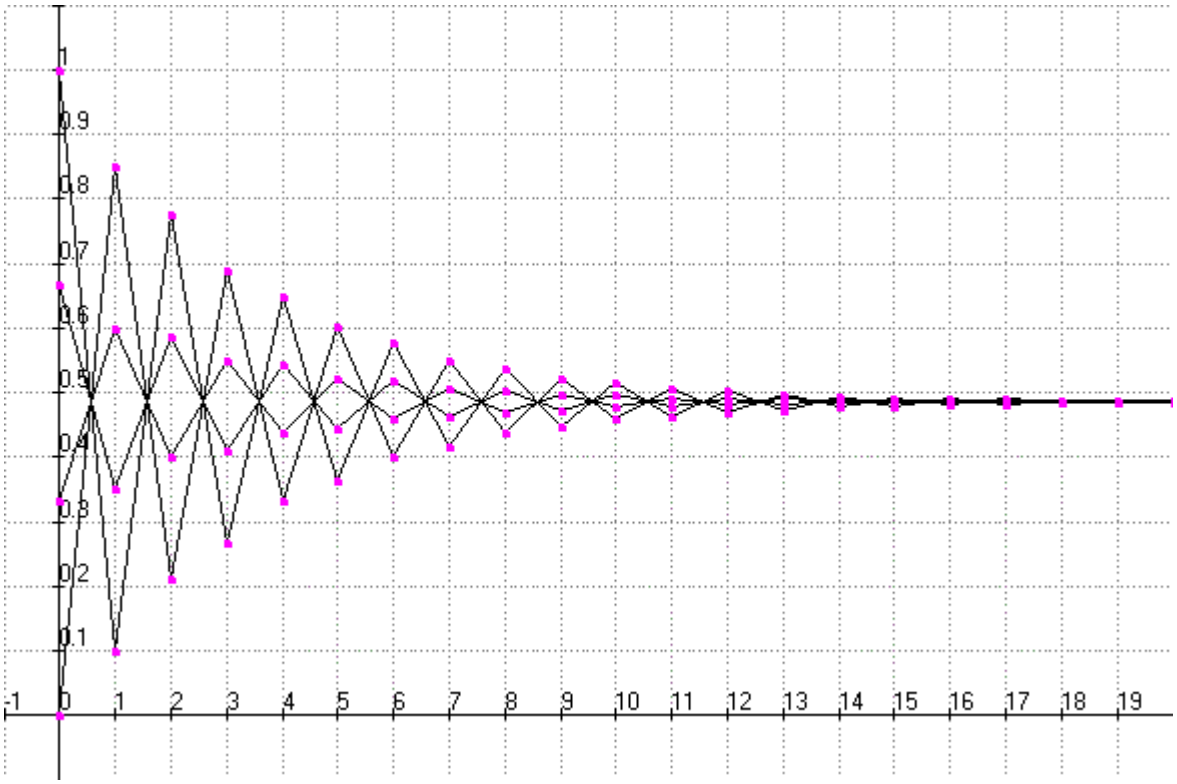


Abb. 2.1.7.3-f: Konvergenz der Markow-Kette – unabhängig vom Startwert

Beispiel 6 – Programmieren mit Funktionen 3 – Iteration in DERIVE 5

Wie kompakt und leistungsfähig Bausteinaufrufe sein können, zeigt auch die Benutzung von „iterates", einem vordefinierte Baustein in dem CAS-System DERIVE 5.

iterates(x^2, x, t, 3) liefert die Funktionsterme t, t^2, t^4, t^8. In der üblichen mathematischen Schreibweise bedeutet das:

$x_1 = t$

$x_{n+1} = (x_n)^2$

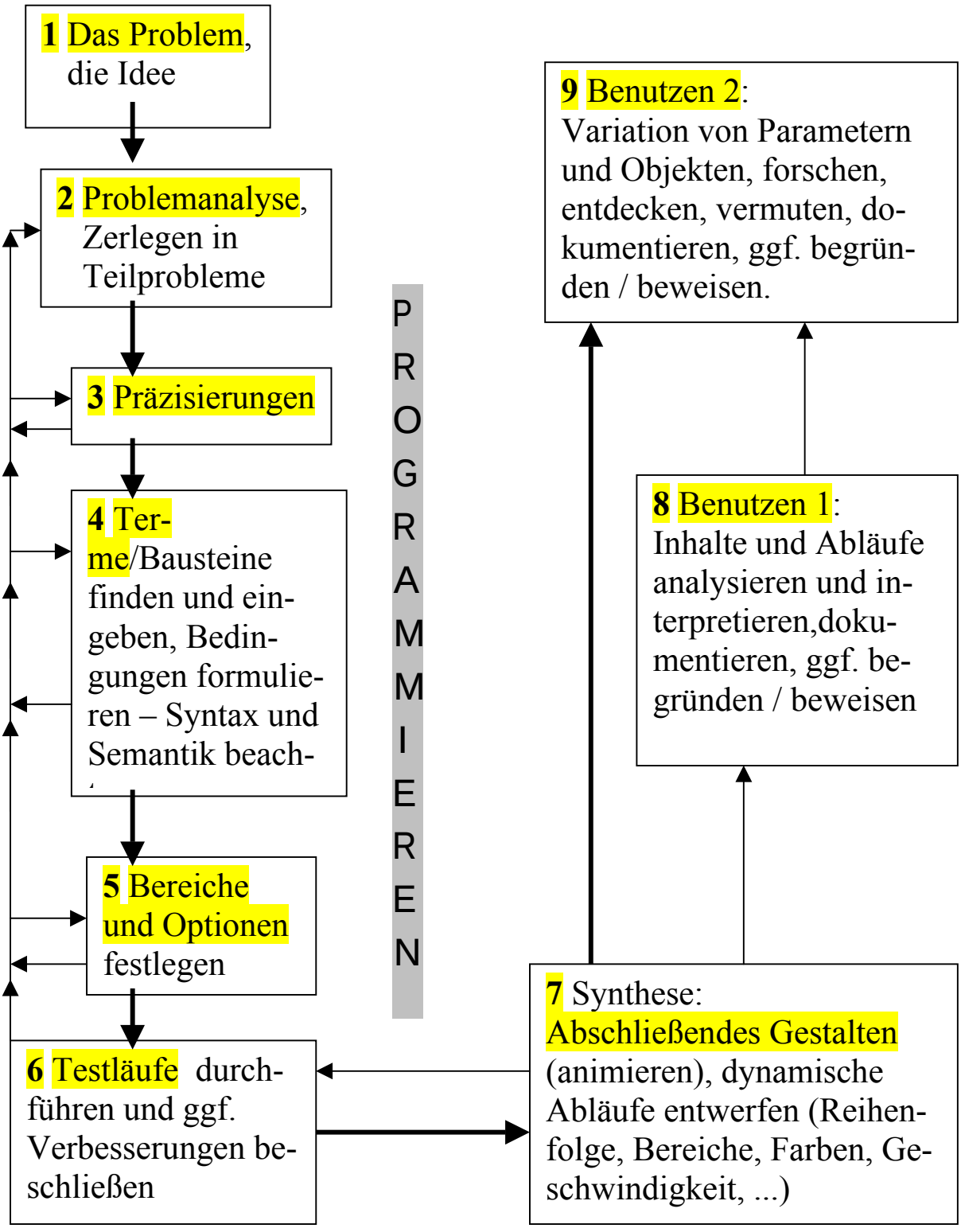
für n = 1 bis 3.

Erstellen einer Animation als Programmierprojekt

Abbildung 2.1.7.3-f kann man auch schrittweise als Animation entstehen lassen, indem man den im Programm notierten Termen passende Laufbereiche, Laufzeiten und andere Eigenschaften, z. B. Farben zuordnet.

Auf der folgenden Seite (Abb. 2.1.7.3-g) wird gezeigt, dass man das Erstellen einer derartigen Animation durch funktionales Programmieren auch als Programmierprojekt auffassen kann.

A Programmieren mit Funktionen/Relationen
im Animationsprogrammsystem ANIMATO (Progamm PLOT2.EXE)



B Bemerkung: Der Informatikunterricht würde sich auch für die Analyse des Programmsystems ANIMATO (Parser, syntaktische Analyse, die Mathematikmaschine, ...) interessieren.

Abb. 2.1.7.3-g: Animationen als Programmierprojekt

Die Bedeutung des eigenen Handelns bei der Benutzung oder eigenen Gestaltung einer Animation wird durch Abbildung 2.1.7.3-h deutlich.

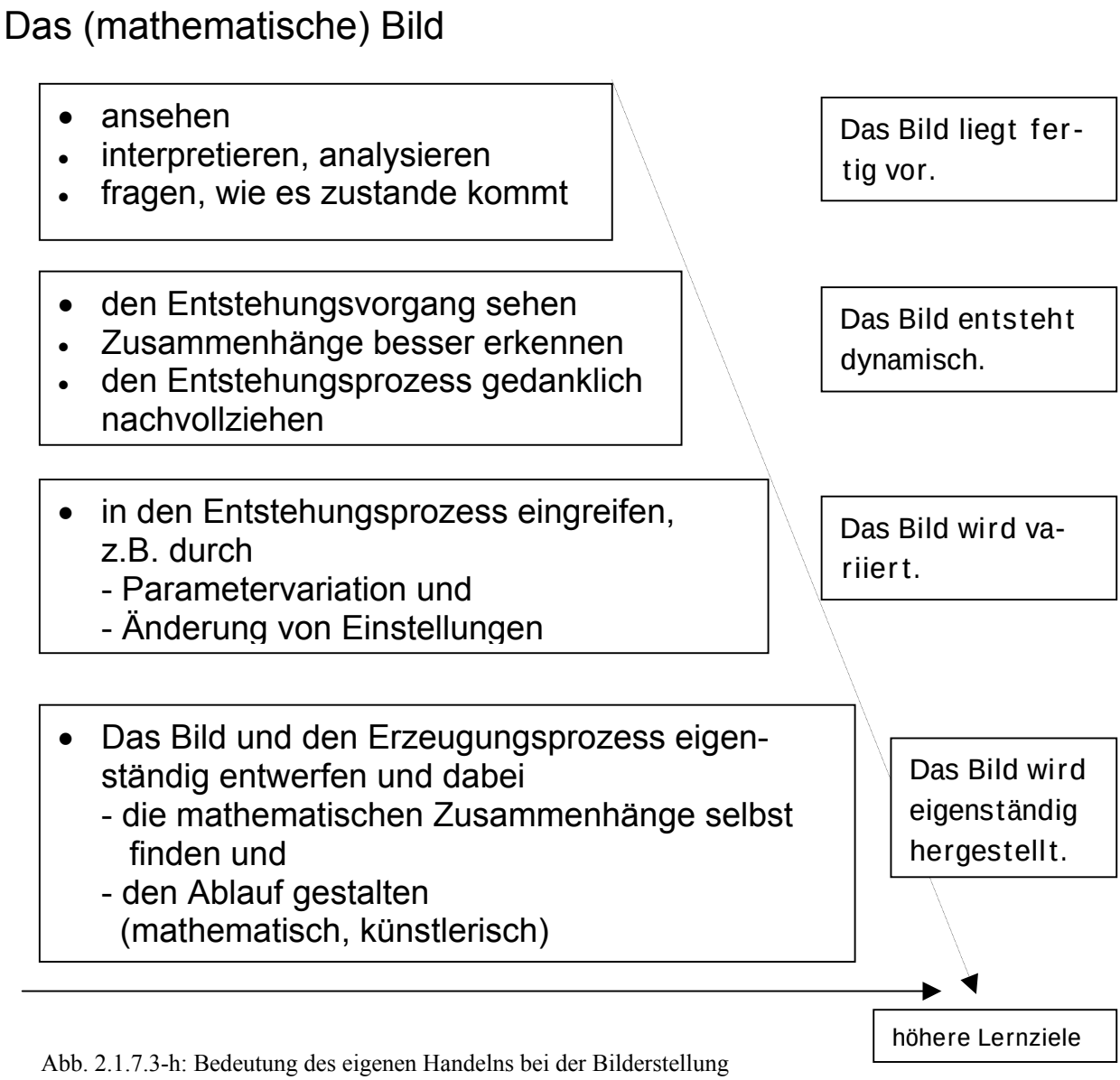


Abb. 2.1.7.3-h: Bedeutung des eigenen Handelns bei der Bilderstellung

Der Anspruch an den Schüler steigt, je mehr er selbst tätig wird. Gleichzeitig werden damit höhere Lernziele erreicht. Die Motivation für eine derartige Arbeitsweise ist hoch.

Abschließend wird eine Empfehlung zur Rolle des Programmierens im Unterricht vorgelegt.

Baustein- und Programmieraktivitäten im M-Unterricht (in CAS)	Im Leistungskurs	In Klasse 11, im Grundkurs	Im Klassenunterricht 7–10 (Aktivitäten von 7 bis 10 ansteigend)
Benutzen (Aufrufen) fertiger Bausteine; z. B. *seq(i^2,i,1,10)*	ja	ja	ja
Selbstdefinieren und Benutzen von Bausteinen (Funktionen) mit Parametern; z. B. *m*x+n* → *gerade(x,m,n)*	ja	ja	ja
Experimentieren mit Bausteinhilfe	ja	ja	ja
Analyse von Bausteinen durch Parametervariationen	ja	gelegentlich bis häufig	gelegentlich bis häufig
Analyse von Baustein-Algorithmen	häufig	gelegentlich	gelegentlich
„10-Zeilen-Programme" benutzen (Demonstration, experimentieren)	ja	ja	ja
„10-Zeilen-Programme" analysieren: Algorithmen, Parametervariation	häufig	selten	selten
„10-Zeilen-Programme" programmieren (Programmeditor benutzen)	gelegentlich	nein	nein
„Viele-Zeilen-Programme" benutzen	ja	selten	nein
„Viele-Zeilen-Programme" analysieren	gelegentlich, auch Teilalgorithmen betrachten	nein	nein
„Viele-Zeilen-Programme" programmieren (Programmeditor benutzen)	nein, bestenfalls Schülerspezialisten	nein	nein

Abb.2.1.7.3-i: Übersicht zum Benutzen und Konstruieren von Programmen im Mathematik-Unterricht

Zusammenfassung

Auf das Schreiben von Programmen im CAS-Programmeditor (und erst recht in einer anderen Programmiersprache) sollte in normalen Mathematikkursen aus folgenden Gründen weitgehend verzichtet werden:

- Die Phasen der Entwicklung des Computereinsatzes im Mathematikunterricht der Schule zeigen, dass das Programmieren für eine weite Verbreitung des Computers im Unterricht hinderlich war. Ein großer Teil der Lehrenden wollte sich nicht damit befassen. Programmieren ist auch für die Lernenden schwer und zudem zeitaufwändig. Diese Einschätzung gilt auch heute noch.
- Das Erstellen längerer Programme in CAS-Systemen leidet unter den gleichen Mängeln wie seinerzeit am PC, etwa in BASIC. Es führt schnell zu unübersichtlichen Programmen mit den bekannten „Spaghetti-Codes", die der Informatiker mit Recht ablehnt. Längere Programmierphasen in Computeralgebrasystemen sind daher für den normalen Mathematikunterricht ein eher belastender Rückschritt. Diese Aussage schließt jedoch nicht aus, dass gelegentlich geeignete Schüler auch Programme als Black Boxes zur Verwendung für alle bereitstellen und zuweilen über deren Konstruktion berichten.
- Die oben genannten Gründe waren es ja (neben vielen anderen), die zur Entwicklung leichter zu bedienender Mathematik-Software und insbesondere von Computeralgebrasystemen geführt haben.
- Programmieren im Mathematikunterricht bedarf des kompetenten Lehrers, der die Chancen des Programmierens, aber auch die Gefahren übertriebenen Programmierens richtig einschätzen kann.
- Als Alternativen für umfangreiches Programmieren im CAS-Programmeditor bieten sich an:
  a) Benutzung vor- und selbstdefinierter Bausteine (siehe u. a. Beispiel 6). Diese können vielseitig verwendet werden, etwa zur Problemlösung, zum Experimentieren und zum Analysieren der in ihnen verborgenen mathematischen Inhalte.
  b) Funktionales Programmieren in kleinem Umfang. Hierbei können insbesondere rekursive Programme verwendet werden.

Abbildung 2.7.1.3-i fasst wesentliche Aspekte zusammen und gibt Unterrichtsempfehlungen für den Umgang mit CAS-Bausteinen und der Programmerstellung im CAS.

3. Mathematisch-informatische Unterrichtssequenzen und Projekte

Kapitel 3 verdeutlicht die vorhergehenden mehr theoretischen Überlegungen und Einzelbeispiele durch vielseitige konkrete Unterrichtssequenzen. Angesichts dieser Zielsetzung gibt es verschiedenartige Angebote.

A1	A2	A3
Kurze Unterrichtseinheiten, die sich an verschiedenen Stellen des Mathematikunterrichts einschieben lassen: Kapitel 3.1, 3.2, 3.4.	Umfangreichere Themen, die einen größeren Zeitbedarf haben: Kapitel 3.3.	Skizzierung der Inhalte eines Kurses Lineare Algebra/Analytische Geometrie unter Einbeziehung von Informatikinhalten von der Dauer eines Halbjahres: Kapitel 3.5.

Abb. 3-a: Planung für die Darstellung von Unterrichtssequenzen

Für alle Vorschläge werden auch mögliche Einordnungen in gängige Lehrpläne angegeben.

3.1 Magische Quadrate

3.1.1 Magische Quadrate zwischen Mathematik und Informatik

Die Attraktivität dieser Thematik für den Mathematikunterricht beruht auf mehreren Faktoren.

- Magische Quadrate faszinieren die Schüler erfahrungsgemäß – und das in verschiedenen Klassenstufen. Magische Quadrate können von Klasse 7 (oder auch schon vorher) bis zum Abitur in unterschiedlicher Bearbeitungstiefe eingesetzt werden.
- Definiert man magische Quadrate über Bausteine mit Parametern (siehe unten), ergeben sich hervorragende Möglichkeiten zu experimentellem Arbeiten. Gleichzeitig wird durch ein derartiges modulares Arbeiten die Verbindung zur Informatik besonders deutlich.
- Da magische Quadrate spezielle Matrizen bzw. Tabellen sind, können wichtige informatische Themen angesprochen werden, z. B. das Thema der Definition einer geeigneten Datenstruktur und von Operationen auf diesen Daten (abstrakte Datenstruktur). In der Praxis werden oft sehr umfangreiche Matrizen verwendet, die häufig speziell aufgebaut sind, z. B. dünnbesetzte Matrizen (viele Nullen). Das wieder bedingt besondere Formen der Datenspeicherung.
- Magische Quadrate können in vielfältiger Form gut in Kursen zur Linearen Algebra verwendet werden (siehe Kapitel 3.5), insbesondere bei Gleichungssystemen, beim Thema „Vektorräume" und als besonders geformte Matrizen.
- Zu magischen Quadraten gibt es eine Fülle von Material im Internet, auf das im Unterricht gut aufgebaut werden kann. Hieraus ergeben sich auch zahlreiche Varianten von Aufgabenstellungen.

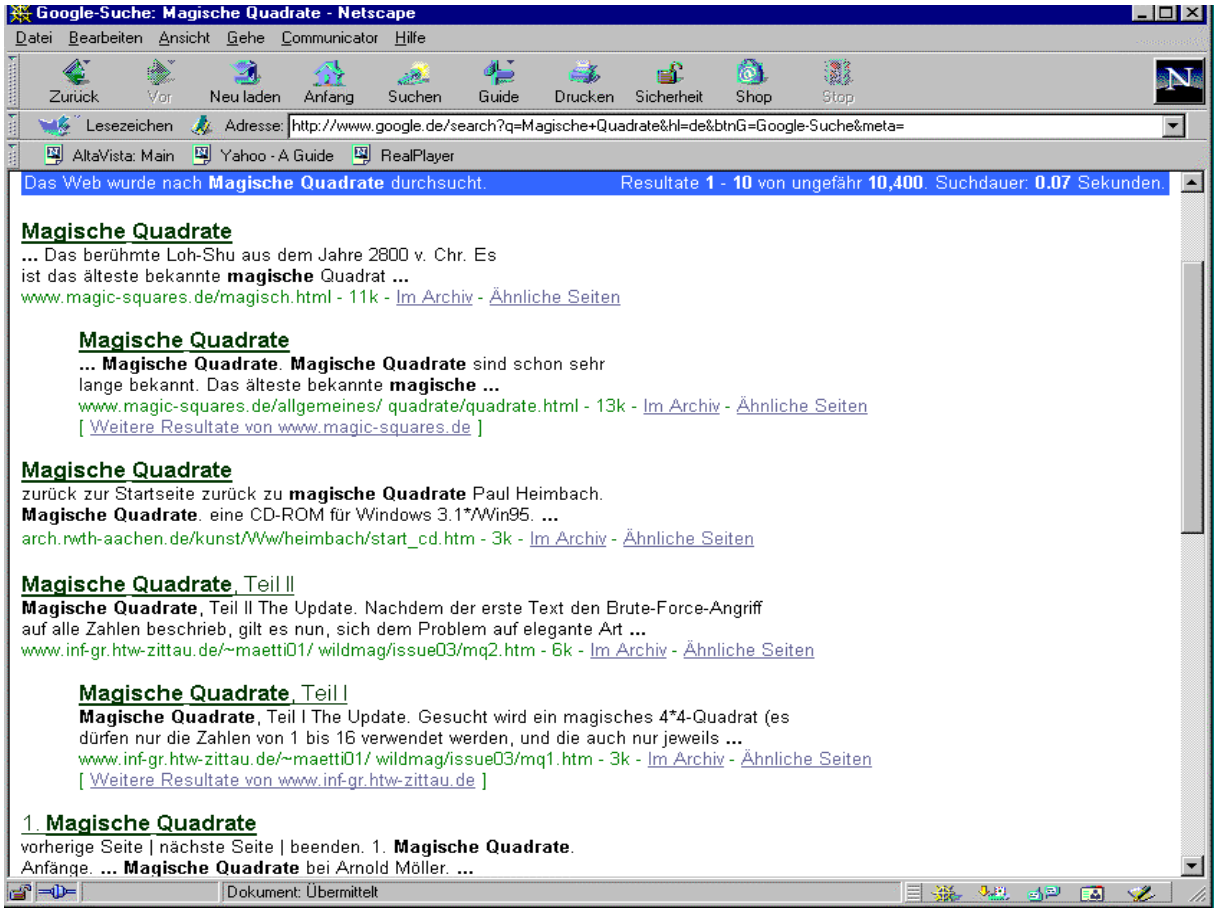
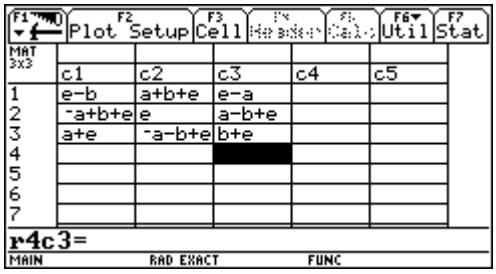


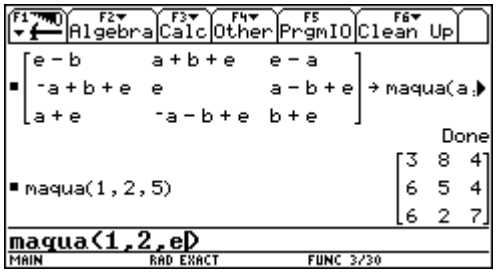
Abb. 3.1-a: Einige Angebote aus dem Internet zum Thema „magische Quadrate" (siehe im Internet auch unter „magic squares")

3.1.2. Einige Unterrichtsideen zu magischen Quadraten

Hinweise zum unterrichtlichen Einsatz finden sich jeweils bei den genannten Ideen.

(1) Definition eines Bausteins mit dem CAS des TI-92. Eingabe der Daten als (3,3)-Matrix.





(2) Arbeit mit dem Baustein: Diverse Aufgabenstellungen, z. B. ein magisches (3,3)-Quadrat erzeugen, in dem jede der Zahlen 1, 2, ..., 9 vorkommt.
Die Fragestellung ist für Lerngruppen beider Sekundarstufen geeignet.

(3) Recherche im Internet, in Gruppen werden Fragestellungen herausgesucht, bearbeitet, vorgetragen, diskutiert.

- Wie kommt es zu den angegebenen Formeln für magische Quadrate verschiedener Ordnung? (für Kurse zur Linearen Algebra, Thema lineare Gleichungssysteme)
- Magische Quadrate mit anderen Symbolen als Zahlen.

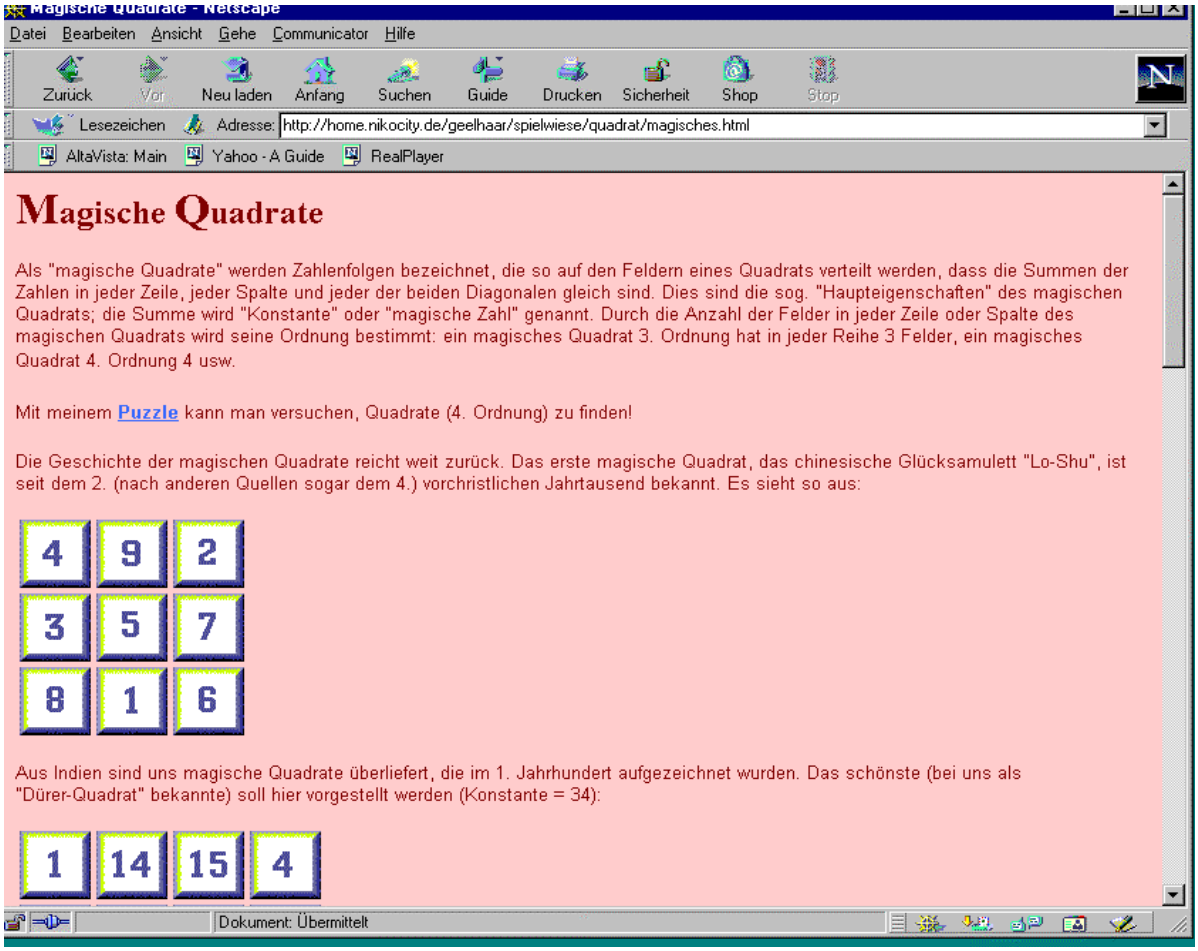


Abb. 3.1.1-b: Eine der vielen Internetseiten zum Thema „magische Quadrate"

(4) Suche andere magische Figuren (geeignet für verschiedene Lerngruppen)!

(5) Was haben magische Quadrate mit Vektorräumen zu tun (für den Leistungskurs Lineare Algebra, Thema: Linearkombination von Matrizen)?

(6) Programmierung magischer Quadrate (für den Leistungskurs Lineare Algebra oder für Informatikkurse).

Hinweis: Der Datentyp *matrix* kann in der Programmiersprache PASCAL z. B. folgendermaßen definiert werden:

```
TYPE matrix = RECORD
        wert: ARRAY[1..maxzeilen, 1..maxspalten] OF REAL; {Matrixelemente}
          m: INTEGER;  {Zeilenanzahl}
          n: INTEGER;  {Spaltenanzahl}
              END;
```

(7) Entwurf einer Wandzeitung über magische Quadrate zum Aushang in der Schule (mehr für die Sekundarstufe 1).

3.1.3 Eine Abituraufgabe zu magischen Quadraten

Die Aufgabe setzt auf der obigen Bausteinidee aus der Informatik auf, verwendet ein Ergebnis aus dem Internet und führt mit linearen Gleichungssystemen zu einem Kernthema der linearen Algebra in der Schule. Hierzu gehört auch eine Teilaufgabe zur Abgeschlossenheit bei Vektorräumen. Damit werden hier mathematische und informatische Inhalte verknüpft.
Auch für den Unterricht im Kurs Lineare Algebra ergeben sich durch die verschiedenen Teilaspekte verschiedene Einsatzmöglichkeiten.

Eine Abituraufgabe (Leistungskurs-Abitur 2001 an der Rückert-Oberschule)

1. Ausgehend von der Matrix M =

a	b	c
d	e	f
g	h	i

findet sich im WWW

die **(a, b, e)-Formel**

e–b	e+a+b	e–a
e–a+b	e	e+a–b
e+a	e–a–b	e+b

für die Konstruktion magischer (3,3)-Quadrate.

1.1) Wie groß ist die magische Summe s?
- Definieren Sie einen Baustein magic(a,b,e) für den TI-92 und
- erzeugen Sie mit diesem zwei magische Quadrate mit natürlichen Zahlen.
- Welche Bedingungen müssen dazu a, b, e erfüllen?
- Welche Rolle spielt der Parameter e?

1.2) Welche Belegung der Parameter a, b, e muss man wählen, um die Formel von den Parametern h, i und e abhängig zu machen?
Ergebnis: Bei dem passenden Aufruf ergibt sich die neue **(h, i, e)-Formel**

2e–i	2e–h	–e+h+i
–2e+h+2i	e	4e–h–2i
3e–h–i	h	i

1.3) Leiten Sie die (h, i, e)-Formel aus der oben gegebenen Matrix M durch Lösung eines geeigneten linearen Gleichungssystems her. Die magische Summe sei s.
Notieren Sie hierzu das LGS und das ermittelte Endschema in übersichtlicher Weise.

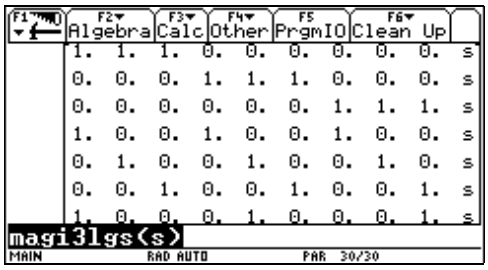
1.4) A und B seien zwei magische (3,3)-Quadrate mit den magischen Summen s1 und s2.
Zeigen Sie, dass jede Linearkombination x*A + y*B wieder ein magisches Quadrat ist.
Magische Summe? – Verwenden Sie beim Beweis die Matrizen $A = (a_{ik\,(3,3)})$ und $B = (b_{ik\,(3,3)})$. – Hinweis: Es reicht, den Beweis für eine Zeile zu führen.

Lösungen und Bewertungen

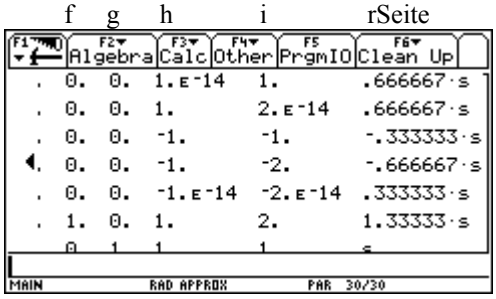
Aufgabenteil, Lösungsskizzen, Erwartungen	Bewertungseinheiten BE, Anforderungsbereiche AB, Erläuterungen
Aufgabe 1.1 • Matrix M eingeben im Matrix-Editor des TI-92, M→ magic(a,b,e) • Aufrufe z.B. magic(1,2,10), magic(2,4,15) • a < e und b < e und a+b <e • Es gilt e = s/3 bzw. s = 3e. Aufgabe 1.2 e + b = i, also b = i – e e – a – b = h, also a = e – h – b = e – h – (i – e) = 2e – h – i, also muss man aufrufen magic(2e–h–i, i–e, e). Dann ergibt sich mit dem TI-92: 2e–i 2e–h –e+h+i –2e+h+2i e 4e–h–2i 3e–h–i h i	4 BE, AB 1 3 BE, AB 1 Baustein definieren 3 BE, AB 2 Die Transformation einer Matrix in eine andere mittels eines Bausteins ist neu. Ansatz und Umformungen sind nicht selbstverständlich. Daher AB 3. 3 BE, AB 3 Arbeit mit dem Baustein

Aufgabe 1.3
LGS aufstellen, notieren und Matrix eingeben:

a b c d e f g h i rSeite



f g h i rSeite



(evtl. in Bruchform 1.3333 = 4/3 usw.)
Aus der letzten Zeile liest man z. B. ab:
g + h + i = s. Man kann also h und i als freie Variable wählen. Dann heisst die letzte Zeile der Formel [s – h – i h i] bzw. [3e – h – i h i]. Entsprechend liest man die anderen Zeilen ab. Insgesamt entsteht die oben angegebene (h, i, e)-Formel mit e = s/3.

Dünn besetzte Matrix

6 BE, AB 2
(Arbeiten mit umfangreichen Datenmengen)
Hinweis: Hier ergibt sich eine Querverbindung zur Informatik zu den Themen „Dateiorganisation, Hash-Verfahren"

LGS mit 10 Variablen und 8 Gleichungen

4 BE, AB 2 (Befehl rref anwenden, notieren)

4 BE, AB 2 (Endschema auswerten)

Aufgabe 1.4 Beweis für erste Zeile führen Neue Zeilensumme: x*a11 + x*a12 + x*a13 + y*b11 + y*b12 + y*b13 =	6 BE, AB 1 Einen Beweis führen

x*(a11 + a12 + a13) + y(b11 + b12 + b13) = x*s + y*s = (x+y)*s neue Zeilensumme. Andere Zeilen und Spalten und Diagonalen entsprechend. Beweisführung kommentieren	

Summe 33 BE 100 %	13 BE, AB 1 39 %	17 BE, AB 2 52 %	3 BE, AB 3 9 %

3.1.4 Datenspeicherung bei Matrizen

Oben wurde bereits eine mögliche Datenstruktur für die Speicherung von Matrizen angegeben:

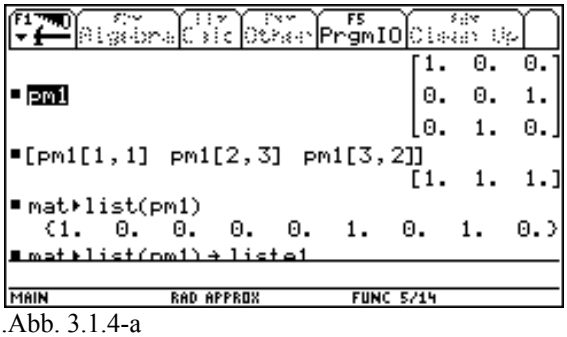
```
TYPE matrix = RECORD
                wert:   ARRAY[1..maxzeilen, 1..maxspalten]  OF REAL;
                        {Matrixelemente}
                m:      INTEGER;    {Zeilenanzahl}
                n :     INTEGER;    {Spaltenanzahl}
              END;
```

Wie schon bemerkt, kommen in Anwendungen häufig Matrizen vor, die besondere Gestalt aufweisen; so ist die obige Datenstruktur nicht immer die günstigste. Hierzu ein Zitat aus

Kurbel, K.: Datenabstraktion und Modularisierung – eine Fallstudie aus der linearen Programmierung, in Informatik Spektrum (Organ der Gesellschaft für Informatik), August 1984, Heft 3, S. 127–137

„Gemeinsames Merkmal großer LP-Probleme (Anmerkung: lineare Programmierung / Optimierung) ist die extrem dünne Besetzung der Matrizen. Der Anteil der von Null verschiedenen Elemente liegt oft nur zwischen 0.5–1%, häufig sogar darunter. Zur Speicherung dünn besetzter Matrizen wurden verschiedene Speicherformate entwickelt, von denen das spaltenweise gepackte Schema („row index/columns pointer scheme") im LP-Bereich am weitesten verbreitet ist."

Auch das in der Abituraufgabe in Kapitel 3.1.3 angegebene lineare Gleichungssystem mit 10 Variablen und 8 Gleichungen weist viele Nullen und wenige Einsen auf. Der Sachverhalt kann schon an wenig umfangreichen Matrizen demonstriert werden, wozu auch ein CAS gut geeignet ist. Ein erster Hinweis auf die Problematik findet sich schon bei der normalen Eingabe einer Matrix in ein CAS. In der Regel wird dort zunächst eine Matrix mit lauter 0-Elementen angegeben, so dass eine Eingabe bei wenigen Elementen ungleich Null schnell erledigt ist.



.Abb. 3.1.4-a

Datenspeicherung einer Permutationsmatrix
1) Speicherung als (3,3)-Matrix PM1.
2) Aufruf der Einsen durch Angabe der Position. – Wenn man weiß, dass die Elemente nur 0 oder 1 sind, reicht auch die Speicherung der Positionen, an denen eine 1 steht. Das spart deutlich an Speicherplatz.
3) Der Befehl matlist(pm1) führt zu einer Umwandlung der Matrix PM1 in eine Liste LISTE1.

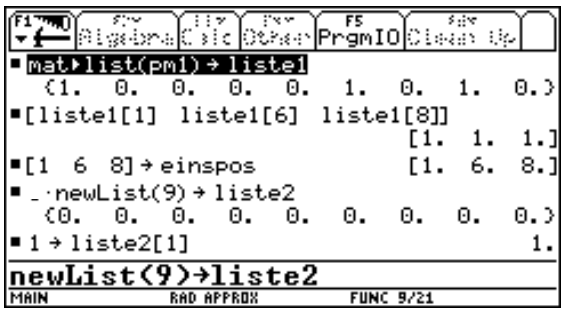


Abb. 3.1.4-b

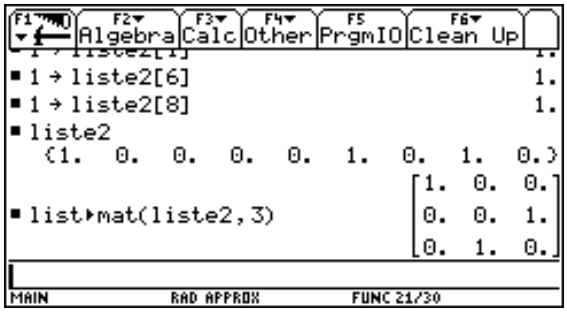


Abb. 3.1.4-c

Wie erwartet, führt der Aufruf der Listenelemente an den Positionen 1, 6 und 8 zu den dort stehenden Einsen. Nach diesen einleitenden Bemerkungen, wird nun der Aufbau einer (3,3)-Matrix mit Einsen an den Positionen (1,1), (2,3) und (3,2) simuliert. Das entspricht einer Liste mit 9 Elementen und Einsen an den Positionen 1, 6, 8.
Diese Positionen seien also bekannt.
1) Zunächst wird eine LISTE2 mit 9 Elementen erzeugt (Befehl *newList(9)*).

2) Danach weisen wir den Listenpositionen 1, 6, 8 Einsen zu.

LISTE2 sieht nun so aus:
[1 0 0 0 0 1 0 1 0].

3) Der Befehl listmat(liste2,3) erzeugt aus der Liste LISTE2 eine Matrix mit 3 Zeilen (und hier auch 3 Spalten). Diese ist in der gewünschten Form.

Zusammenfassung:
Zur Erzeugung von Matrizen mit vielen Nullen und wenigen Einsen genügt es ihren Typ und die Positionen der Einsen zu kennen. – Man erzeugt zunächst eine Nullmatrix des gewünschten Typs und bringt dann die Einsen an die bekannten Positionen. Bezieht man diese Überlegungen auf umfangreiche dünn besetzte Matrizen, so wird deutlich, dass man erheblich an Speicherplatz sparen kann.

Die angesprochene Problematik ist in ein wichtiges Problem der Informatik eingebettet – das der Dateiorganisation (Speichern und Suchen in Dateien). Zugriffe auf Dateielemente können erfolgen u. a. durch

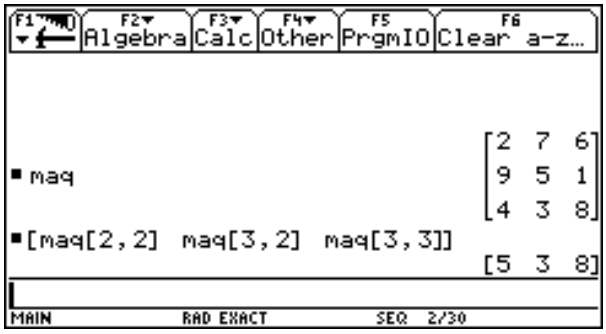
- einen Zugriff über den Dateninhalt oder
- über die Speicherposition der Daten.

Hash-Funktion
Es gibt jedoch auch eine Mischform, die diese Ansätze verbindet. Die sich daraus ergebende Datenstruktur wird Hash-Tabelle genannt. In einer Hash-Tabelle wird dem jeweiligen Dateninhalt mittels einer Speicherfunktion (Hash-Funktion h: X (Datenraum) →A (Adressraum)) eine eindeutige Speicheradresse zugeordnet. Eine mögliche Form einer Hash-Funktion ist
$h(x) = f(x) \bmod p$, z. B. $h(x) = x \bmod 17$. Hier zeigt sich erneut die Verwendungsmöglichkeit der Modulo-Funktion, siehe auch Kapitel 3.4.1 und 3.4.3. Nähere Informationen findet man z. B. unter der Internetadresse *ivs.cs.uni-magdeburg.de-dumke/EAD/Skript38.html* oder im Informatik-Duden.

Im Folgenden wird ein Weg gezeigt, wie man auch über magische Quadrate auf die Problematik des Speicherns von speziellen Matrizenformen gelangen kann.

Problem:
Wie viel Elemente muss man bei einem magischen (3,3)-Quadrat mindestens kennen, wenn man die magische Summe s = 15 kennt?

Zur Bearbeitung des Problems könnte man mit einem Beispiel beginnen. Es sei
maq = $\begin{pmatrix} 2 & 7 & 6 \\ 9 & 5 & 1 \\ 4 & 3 & 8 \end{pmatrix}$. Offenbar reicht z. B. die Kenntnis der Elemente 5, 3, 8 (in der in maq angegebenen Position und mit s = 15). Benutzt man auch noch, dass für das mittlere Element e gilt 3e = s, so reichen sogar die Kenntnis von 5, 3 und 8. Damit kann man für die Speicherung der Daten so vorgehen:



Das magische Quadrat hat augenscheinlich die magische Summe 15. Außerdem gilt für das mittlere Element 5*3 = 15.

Aus dieser knappen Datenspeicherung lässt sich das magische Quadrat durch entsprechende Rechnungen rekonstruieren.

Abb. 3.1.4-d

$\begin{pmatrix} a & b & c \\ d & 5 & f \\ g & 3 & 8 \end{pmatrix}$, mit dieser Matrix lässt sich das vollständige Quadrat herstellen.

Man erkennt hier einen unmittelbaren Zusammenhang zwischen Formen der Datenspeicherung und mathematischen Überlegungen. Einen engen Bezug gibt es zu dem zum magischen (3,3)-Quadrat gehörenden Gleichungssystem, siehe obige Abituraufgabe.

Zusammenfassung: Einige Ideen für die Arbeit mit magischen Quadraten im Mathematikunterricht

.

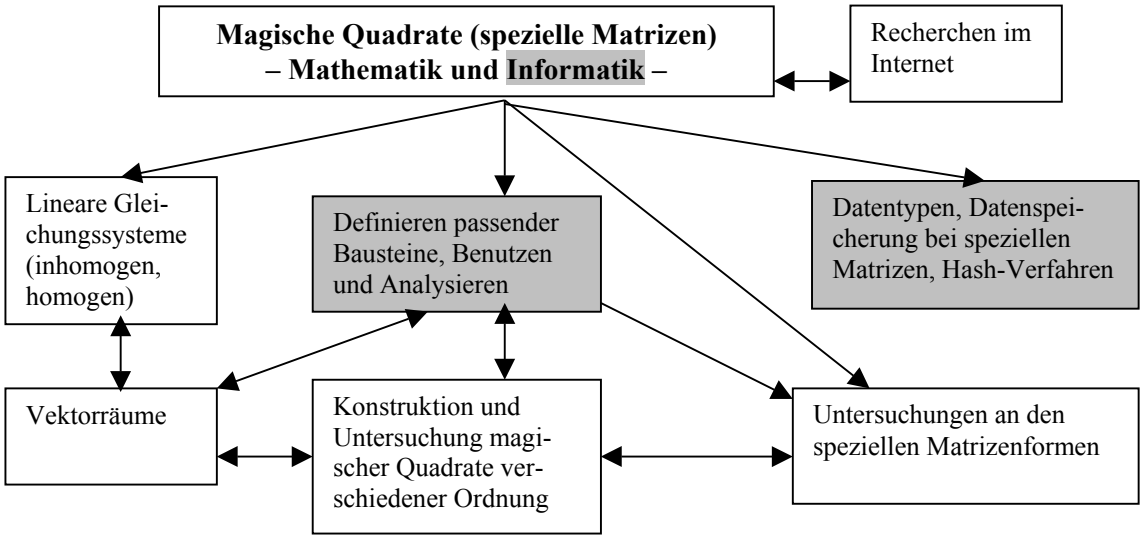


Abb. 3.1.4-e: Magische Quadrate im Unterricht – vielseitig einsetzbar – auch im Kurs „Lineare Algebra"

3.2 Eine mathematisch-informatische Entdeckungsreise

Teilverhältnisse auf Dreiecksseiten – ein weiteres Projekt für wenige Stunden

Hinweise zur unterrichtlichen Verwendung des Themas

(A) Etwa in Klasse 8 kann man die Mittelpunkte der Dreiecksseiten konstruieren und über die Mittelpunktsformeln berechnen lassen. Diese können dabei ggf. hergeleitet werden.	(B) In Klasse 10 oder 11 ist das Thema „Folgen" relevant. Die Entdekkungsreise ist eine schöne Anwendung rekursiv definierter Folgen.	(C) Im Analysiskurs kann man zunächst wie in (B) vorgehen. Anschließend können explizite Darstellungen gesucht und Grenzwerte gebildet werden.	(D) Im Kurs „Analytische Geometrie" kann es neben den Teilpunktberechnungen um die Schwerpunktberechnung gehen. Außerdem kann mit anderen Teilverhältnissen experimentiert werden.	(E) Grafische Darstellungen sind z. B. im CAS oder mit ANIMATO schon in (A)-(D) erwünscht. Zusätzlich können diese unter dem Aspekt der Computergrafik gesehen und möglicherweise programmiert werden.

Die Entdeckungsreise zielt insbesondere auf folgende Aspekte ab:

- Mathematik betreiben mit rekursiv definierten Funktionen,
- Programmieren lernen/üben mit rekursiv definierten Funktionen/Relationen,
- entdeckendes Lernen praktizieren mit Hilfe einer dynamischen Animationssoftware, die die eigene Konstruktion von Animationen ermöglicht,
- das Wechselspiel zwischen Visualisierung, Numerik (Wertetafeln) und exakten Beweisen erfahren.
- Mathematik verstehen durch Experimentieren, Entdecken und Beweisen!

Mittendreiecke – Ausgangspunkt für eine mathematische Entdeckungsreise

Gegeben ist ein Dreieck. Die Mittelpunkte der drei Seiten werden verbunden, so dass ein neues Dreieck entsteht. Wird dieser Vorgang für die folgenden Dreiecke wiederholt, so erhält man eine Folge von Mittendreiecken.
Der geschilderte Ansatz und Variationen der Aufgabenstellung können eine interessante mathematische Entdeckungsreise einleiten!

Abbildung 3.2-a zeigt die Ausgangskonfiguration (im Unterricht sollte man sie zunächst durch Handzeichnungen erzeugen lassen).

Problembearbeitung - Rekursion

Das Problem lässt sich elegant mit rekursiv definierten Folgen angehen. Derartige Folgen sind ein überzeugendes Beispiel für eine Schnittstelle zwischen Mathematik und Informatik. Einerseits lassen sich viele mathematische Probleme mit rekursiven Funktionen bearbeiten, andererseits gehören rekursiv definierte Funktionen zu den Grundelementen funktionaler Programmiersprachen. Die Schnittstelle spiegelt sich u. a. bei der Arbeit mit einem Funktionenplotter wider. Hier wird die schon mehrmals erwähnte Animationssoftware ANIMATO benutzt.

Bei der Interpretation der entstehenden Bilder kann man u. a. folgende Besonderheiten feststellen:

- Entstehen einer „Spirale" beim Verbinden aller Punkte A_i, B_i oder C_i,
- gleiche Proportionen der Seitenlängen für alle Dreiecke,
- alle entstehenden Dreiecke einer Teilungsstufe sind kongruent,
- Ähnlichkeit der aufeinanderfolgenden Dreiecke.

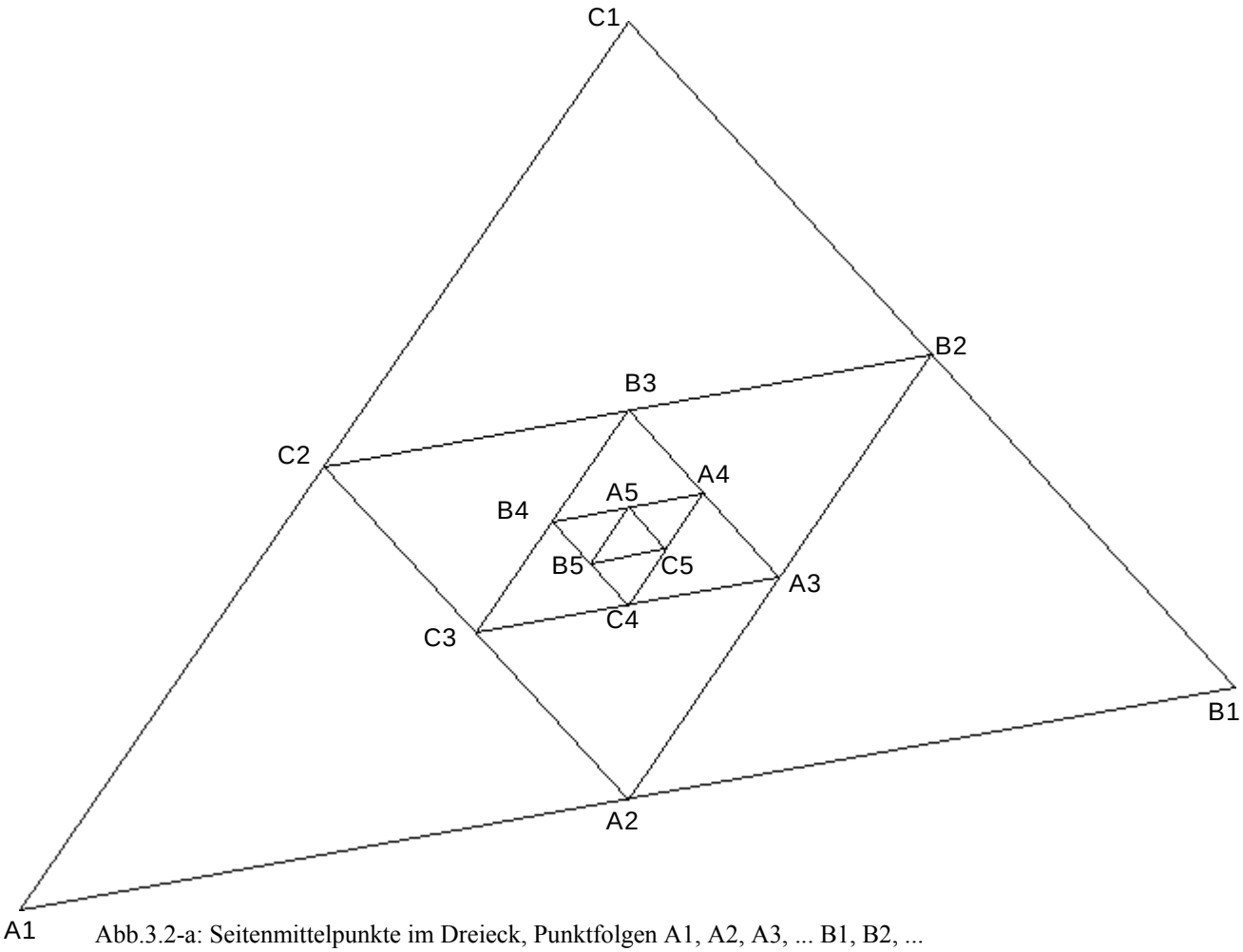


Abb.3.2-a: Seitenmittelpunkte im Dreieck, Punktfolgen A1, A2, A3, ... B1, B2, ...

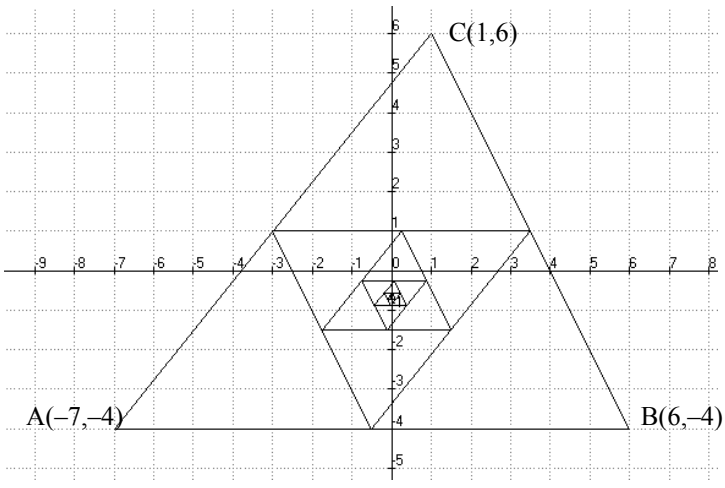


Abb. 3.2.-b: Zeichnung im Koordinatensystem

Die Programmierung von ANIMATO zeigt die verwendeten Rekursionsformeln.

```
f1      {n=1:-7:(f1(n-1)+u*f2(n-1))/(1+u)}
f2      {n=1:6:(f2(n-1)+u*f3(n-1))/(1+u)}
f3      {n=1:1:(f3(n-1)+u*f1(n-1))/(1+u)}
f4      {n=1:-4:(f4(n-1)+u*f5(n-1))/(1+u)}
f5      {n=1:-4:(f5(n-1)+u*f6(n-1))/(1+u)}
f6      {n=1:6:(f6(n-1)+u*f4(n-1))/(1+u)}
f7      f1(n),f4(n),f2(n),f5(n)
f8      f2(n),f5(n),f3(n),f6(n)
f9      f3(n),f6(n),f1(n),f4(n)
f10     f1(n),f4(n)
```

Ein Programm für die Animationssoftware ANIMATO (oder für den MS-DOS-Funktionenplotter HL-PLOT11).

Programmieren mit rekursiv definierten Funktionen

Programmerläuterungen

Terme	**Erläuterungen**
f1 {n=1:-7:(f1(n-1)+u*f2(n-1))/(1+u)}	f1 berechnet die x-Werte der A-Punkte aus den vorhergehenden A- und B-Punkten; für den Mittelpunkt ist u=1.
f2 {n=1:6:(f2(n-1)+u*f3(n-1))/(1+u)}	f2 berechnet die x-Werte der B-Punkte aus den vorhergehenden B- und C-Punkten
f3 {n=1:1:(f3(n-1)+u*f1(n-1))/(1+u)}	f3 berechnet die x-Werte der C-Punkte aus den vorhergehenden C- und A-Punkten
f4 {n=1:-4:(f4(n-1)+u*f5(n-1))/(1+u)}	f4 berechnet die y-Werte der A-Punkte aus den vorhergehenden A- und B-Punkten; für den Mittelpunkt ist u=1.
f5 {n=1:-4:(f5(n-1)+u*f6(n-1))/(1+u)}	f5 berechnet die y-Werte der B-Punkte aus den vorhergehenden B- und C-Punkten
f6 {n=1:6:(f6(n-1)+u*f4(n-1))/(1+u)}	f6 berechnet die y-Werte der C-Punkte aus den vorhergehenden C- und A-Punkten
f7 f1(n),f4(n),f2(n),f5(n)	f7 zeichnet die Strecken AB
f8 f2(n),f5(n),f3(n),f6(n)	f8 zeichnet die Strecken BC
f9 f3(n),f6(n),f1(n),f4(n)	f9 zeichnet die Strecken CA
f10 f1(n),f4(n)	f10 zeichnet die A-Punkte in einer anderen Farbe, um die Ortskurve darzustellen

Die Ausgangspunkte der Berechnung sind A(-7,-4), B(6,-4) und C(1,6).

Vermutung 1

Die Punktfolgen A1, A2, A3, ..., B1, B2, B3, ..., C1, C2, C3, ... konvergieren auf einen gemeinsamen Punkt zu.

Ein Blick in die Wertetafel für die Funktionen f4, f5, f6 (diese stellen die y-Folgen der drei Punktfolgen dar) bestätigt das numerisch:

x,t	u	f4	f5	f6	
1	1	-4	-4	6	
2	1	-4	1	1	
3	1	-1.5	1	-1.5	
4	1	-0.25	-0.25	-1.5	
5	1	-0.25	-0.875	-0.875	
6	1	-0.5625	-0.875	-0.5625	
7	1	-0.71875		-0.71875	-0.5625
8	1	-0.71875		-0.640625	-0.640625
9	1	-0.6796875		-0.640625	-0.6796875
10	1	-0.66015625		-0.66015625	-0.6796875
11	1	-0.66015625		-0.66992188	-0.66992188
12	1	-0.66503906		-0.66992188	-0.66503906
23	1	-0.66666508		-0.66666746	-0.66666746
24	1	-0.66666627		-0.66666746	-0.66666627
25	1	-0.66666687		-0.66666687	-0.66666627
26	1	-0.66666687		-0.66666657	-0.66666657
27	1	-0.66666672		-0.66666657	-0.66666672
28	1	-0.66666664		-0.66666664	-0.66666672
29	1	-0.66666664		-0.66666668	-0.66666668
30	1	-0.66666666		-0.66666668	-0.66666666
31	1	-0.66666667		-0.66666667	-0.66666666
32	1	-0.66666667		-0.66666667	-0.66666667
33	1	-0.66666667		-0.66666667	-0.66666667

Für u = 1 ergeben sich die jeweiligen Mittelpunkte der Dreiecksseiten.

Der Grenzwert der y-Folgen scheint Gy = – 2/3 zu sein.

Entsprechend sieht man:
Der Grenzwert der x-Folgen scheint Gx = 0 zu sein.

Eine entsprechende grafische Darstellung der y-Folgen veranschaulicht das noch zusätzlich:

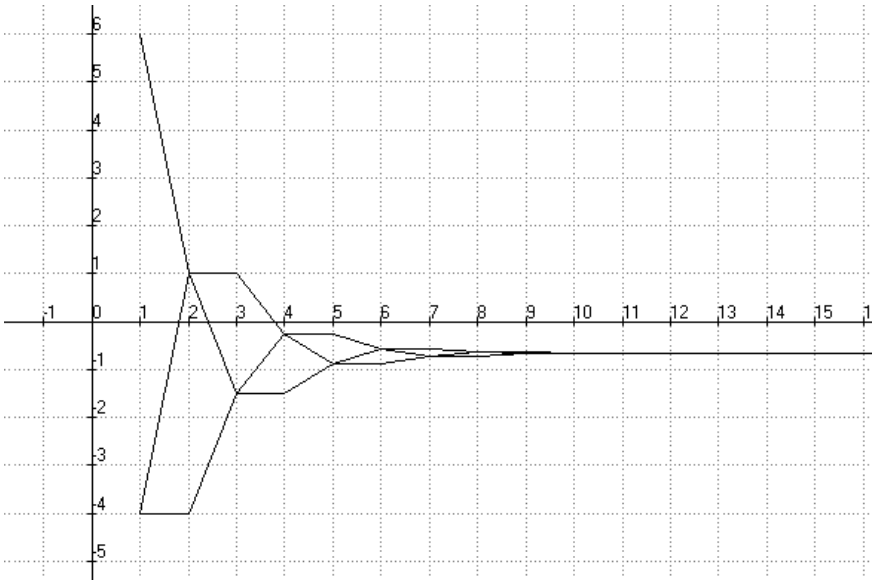


Abb. 3.2-c: Die drei y-Folgen der Punktfolgen $< A_i >$, $< B_i >$ und $< C_i >$.

Vermutung 2
Der Punkt G ist Schwerpunkt des Ausgangsdreiecks (sogar aller Dreiecke)!

Beweis: Der Schwerpunkt berechnet sich mit den angegebenen Formeln für S_x und S_y.

$$S_x = \frac{A_x + B_x + C_x}{3} = \frac{-7+6+1}{3} = 0 \text{ und } S_y = \frac{A_y + B_y + C_y}{3} = \frac{-4+(-4)+6}{3} = -\frac{2}{3}.$$

Andere Teilverhältnisse

Abbildung 3.2.4 wandelt die Idee der Mittelpunkte ab und benutzt sehr kleine Teilstrecken, z. B. das Teilverhältnis u = 0.02 (für die Mittelpunkte war oben u = 1).

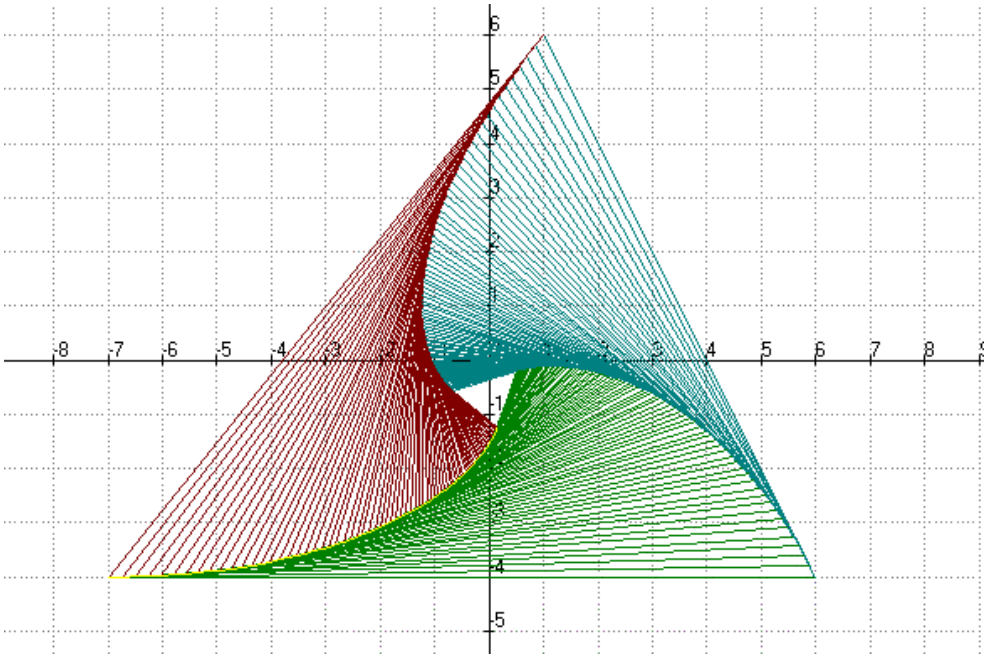


Abb. 3.2-d: Die Situation für u = 0.02

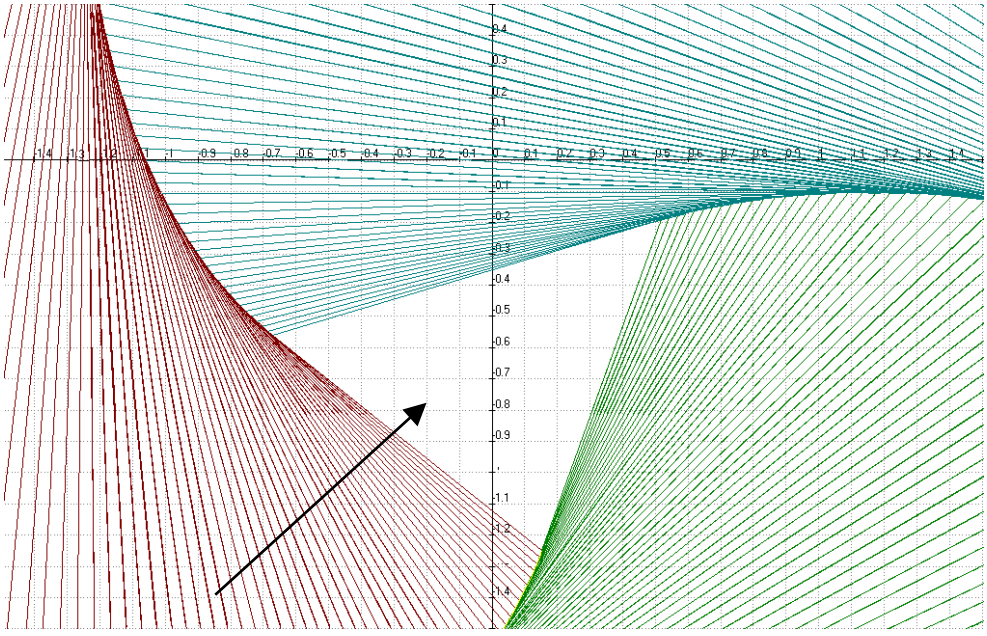


Abb. 3.2-e: Ein Blick ins „Innere”

Je nach Wahl von u können also verschiedene Figuren erzeugt werden. Die Anzahl der Schritte kann unterschiedlich gewählt werden. Die Zeichengeschwindigkeit und andere Optionen wie z. B. Farbe, Anzahl der Berechnungen können je nach gewünschter Animation gesteuert werden.

Wir vergrößern nun die Anzahl der Iterationsschritte auf n=500 und zoomen die Zeichnung um den vermutlichen Grenzpunkt herum.

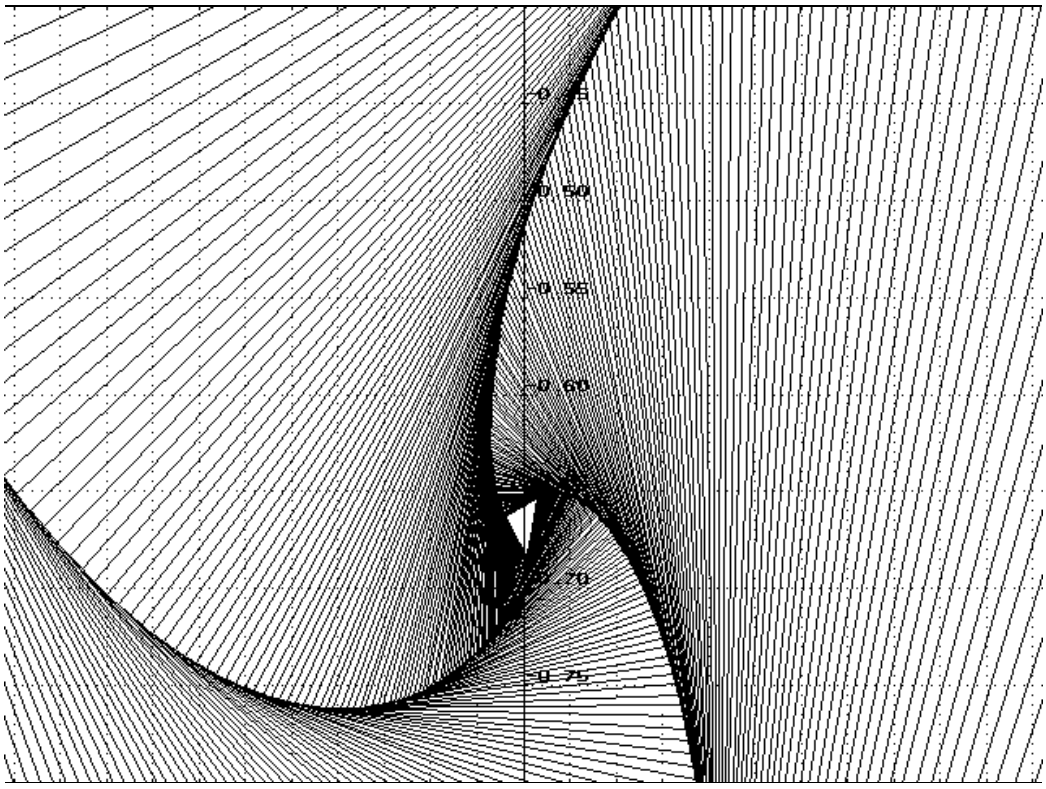


Abb.3.2-f: Zoom um den Konvergenzpunkt herum

Abbildung 3.2-g ergänzt die bisherigen Darstellungen und zeigt die Ortskurven der A-B-C-Punkte, so wie sie auch in Abbildung 3.2-f erkennbar sind.

In
Eberhard Lehmann: Von den Mittendreiecken zu Teilpunktpolygonen, Zeitschrift ML Mathematiklehren 1988, Heft 27, Seite 13-19,
werden zu den oben benutzten rekursiv definierten Formeln für die x-Werte

f1 {n=1:-7:(f1(n-1)+u*f2(n-1))/(1+u)},

f2 {n=1:6:(f2(n-1)+u*f3(n-1))/(1+u)},

f3 {n=1:1:(f3(n-1)+u*f1(n-1))/(1+u)},

explizite Formeln hergeleitet, mit denen dann der Grenzwert der Folgen exakt berechnet werden und als Schwerpunkt des Dreiecks identifiziert werden kann.

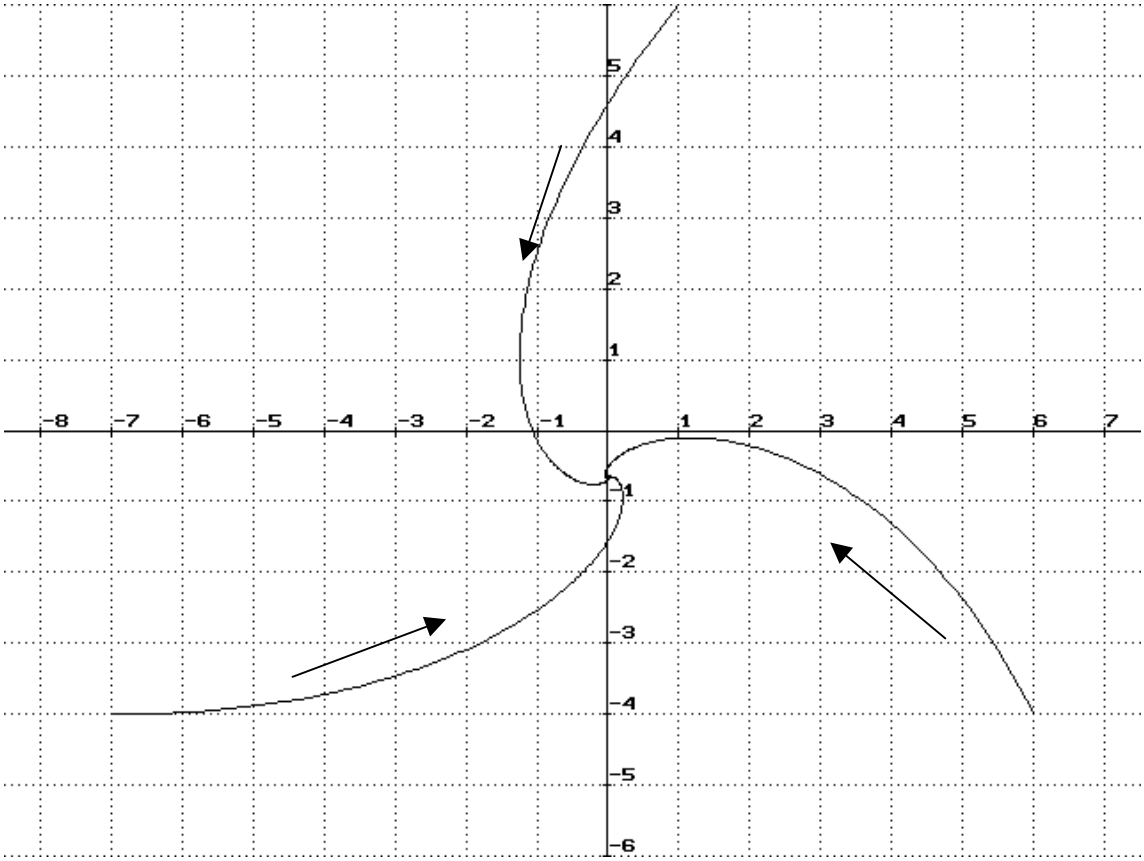


Abb.3.2-g: Die drei Punkt-Folgen im Koordinatensystem

Zusammenfassung und Ausblick

Der informatische Anteil dieses Kapitels bestand in der Bereitstellung geeigneter Animationssoftware und deren funktionaler Programmierung. Da die Schüler mit dieser Software durch die Wahl verschiedener Optionen gestalterisch tätig werden können, kommt die hier auf Papier nicht darstellbare Dynamik der Figuren bei der Arbeit am Rechner voll zum Tragen, so dass eine motivierende Unterrichtsreihe entstehen kann. Die sich aus den Bildschirmdarstellungen ergebenden Vermutungen können durch mathematische Überlegungen exakt erklärt werden.

3.3 Zustandsgraphen in Informatik und Mathematik – ein längeres Projekt

Recherchen im Internet zu den Begriffen „Zustandsgraphen" oder „Übergangsgraphen" bringen eine reiche Ausbeute. Hier ein kleiner Auszug, der neben weiteren, in 3.3.1 genannten Aspekten, von der Bedeutung des Themas zeugt.

aus dem Internet

Ohne Titel
... Aufgabe 3 - **Übergangsgraph**, Erkennen von Sprachen- (3+3 Punkte). Geben Sie einen
endlichen Automaten an , der die Sprache L = { a 3 b 3 b m c n a 2 | n>=0 , m ...
www.mathe2.uni-bayreuth.de/axel/ informatik1_ws0001_blatt1.html - 5k - Im Archiv - Ähnliche Seiten

'Grundlagen Digitaler Systeme' - Formelsammlung
... Sequentielle Schaltsysteme II. statisch gesteuerte, ungetaktete Schaltsysteme.
Trajektorie/Raumkurve: Zustandsübergänge in **Übergangsgraph** bei konstanter ...
www-user.tu-chemnitz.de/~rola/ hauptstud/ds/frml_ds.html - 42k - Im Archiv - Ähnliche Seiten

Musteraufgaben des Oberschulamts Karlsruhe
... der Übergänge: Verbale Beschreibung der Übergänge zwischen verschiedenen Stadien;
Pfeildiagramm (andere Bezeichnungen: Graph, **Übergangsgraph**, Flußgraph ...
www.lehrer.uni-karlsruhe.de/~za242/ osa/mstproz/Muster98.html - 31k - Im Archiv - Ähnliche Seiten

[PDF]Einführung in die Informatik II Automaten
Dateiformat: PDF/Adobe Acrobat - HTML-Version
... für die Übergangsfunktion O Die Übergangsfunktion eines endlichen Automaten lässt
sich auf verschiedene Art und Weise darstellen: als Graph **Übergangsgraph** ...
wwwbruegge.in.tum.de/teaching/ss01/Info2/ vorlesung/folien/09_Automaten_4.pdf - Ähnliche Seiten

Abb. 3.3-a: Internetrecherche zum Begriff „Übergangsgraph"

3.3.1 Endliche Automaten und Markow-Ketten

Kapitel 3.3 ist dem Thema „Zustandsgraphen" (häufig wird auch von „Übergangsgraphen" gesprochen) und einigen wichtigen, damit zusammenhängenden Fragestellungen und Algorithmen gewidmet. Es wird gezeigt, dass diese Thematik besonders geeignet ist, Vernetzungen zwischen Mathematik und Informatik aufzuzeigen. In der Schulinformatik wird, besonders in Leistungskursen, das Thema „Endliche Automaten" oder auch das Thema „Turingmaschinen" behandelt. In einigen Mathematiklehrplänen wird das Gebiet „Markow-Ketten" genannt. Dabei werden auch Zustandsmengen und Zustandsgraphen (Übergangsgraphen) verwendet. Insgesamt findet das Thema jedoch noch zu wenig Beachtung, obwohl es in Mathematik, Informatik und anderen Wissenschaften eine wesentliche Bedeutung hat. Abb. 3.3-a gab hierüber bereits einige Informationen. Weitere Hinweise ergeben sich aus
[Wer95], S. 33 f.:

„Viele Beispiele für Systeme mit endlichen Zustandsmengen findet man in der Informatik. Häufig benutzte Programme – wie Texteditoren oder lexikalische Analyser – werden oft als Systeme mit endlichen Zustandsmengen entworfen. Die Theorie der endlichen Automaten ist damit ein nützliches Werkzeug zum Design solcher Systeme. Das vorliegende Konzept tritt in verschiedensten Gebieten auf, was der vielleicht wichtigste Grund für die intensive Untersuchung von Systemen mit endlichen Zustandsmengen ist."

Zustandsgraphen sind in der Regel immer dann geeignet, wenn es um die Darstellung von Prozessabläufen geht.

- In der Informatik ist das z. B. bei der Behandlung von **endlichen Automaten** der Fall.
- In der Mathematik können u. a. **Markow-Ketten** als Beispiel genannt werden, in dem Zustandsgraphen fundamental sind.

Um mit den Graphen zu arbeiten, werden besondere Formen der **Datenspeicherung** (z.B. der Datentyp Matrizen) und auf den Graphendaten arbeitende **Algorithmen** benötigt.

Die obigen Argumente sind wichtige Gründe, um hier ein ausführlicheres Angebot für den Mathematikunterricht zu unterbreiten.

Die folgende Zusammenstellung von Begriffen und die Erläuterungen dienen u.a. dazu, die Vernetzung zwischen den genannten Themen schon bei den Grundlagen zu verdeutlichen.
Hinweis: Damit ist nicht etwa gemeint, dass man eine Unterrichtsreihe mit diesen Begriffsfestlegungen beginnen sollte.

Definition des endlichen Automaten

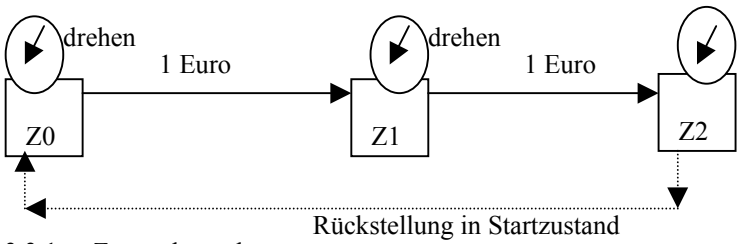
Ein endlicher Automat ist *ein 6-Tupel A=(X, Y, Z, δ, λ, Z0).*

X: *Eingabealphabet*	X,Y und Z sind nichtleere, endliche Mengen.
Y*: Ausgabealphabet*	
Z: *Zustandsmenge*	Menge der Zustände, die der Automat annehmen kann, ggf. mit Endzuständen.

$Z0 \in Z$ ist der *Anfangszustand***.**

δ: X x Z → Z ist die *Übergangsfunktion*
λ: X x Z → Y ist die *Ausgabefunktion*

Zustandsgraph eines Blumen-Automaten



(Einzahlung gesperrt)
Drehen: Fach öffnet sich,
Blumen entnehmen, danach
automatische Rückstellung
auf Anfangszustand Z0.

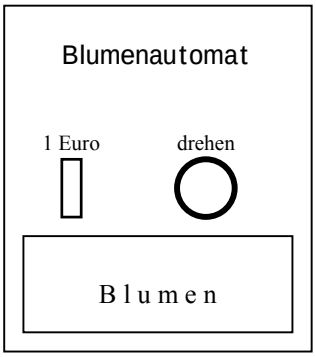
Abb. 3.3.1-a: Zustandsgraph

In diesem Fall sind die charakteristischen Automaten-Daten:

Eingabealphabet X = {1 Euro, drehen}
Ausgabealphabet Y = {nichts, Blume}
Zustandsmenge Z = {Z0, Z1, Z2}
Anfangszustand Z0

Übergangsfunktion δ: X x Z → Z
Die Paare aus X x Z haben die Form (Eingabeelement, Zustand).
Die einzelnen Eingaben bewirken bei den einzelnen Zuständen
den Übergang zu den (Folge-)Zuständen.

(drehen, Z0)	→ Z0	(1 Euro, Z0)	→ Z1
(drehen, Z1)	→ Z1	(1 Euro, Z1)	→ Z2
(drehen, Z2)	→ Z0	(1 Euro, Z2)	→ Z2, weil Einzahlung gesperrt



Jedem möglichen Paar (Eingabe,Zustand) ist ein Folgezustand zugeordnet.

Ausgabefunktion λ**:** X x Z → Y

(drehen, Z0)	→ nichts	(1 Euro, Z0)	→ nichts
(drehen, Z1)	→ nichts	(1 Euro, Z1)	→ nichts
(drehen, Z2)	→ Blume	(1 Euro, Z2)	→ nichts

Jedem möglichen Paar (Eingabe, Zustand) ist eine Ausgabe zugeordnet.

Die Angaben kann man auch zu einer *Automatentabelle* zusammenführen:

Eingaben	Zustand Z0	Zustand Z1	Zustand Z2
1 Euro	nichts, Z1	nichts, Z2	nichts, Z2
drehen	nichts, Z0	nichts, Z1 *	Blume, Z0

Eine weitere (später verwendete) Möglichkeit besteht im Notieren von 4-Tupeln, etwa für die Stelle
* (Aktueller Zustand, Eingabe, Ausgabe, Folgezustand) = (Z1, drehen, nichts, Z1)

Beispiel eines Eingabe-Protokolls

Über die Eingaben in den Automaten kann man Protokoll führen:

Lfd. Nummer	Zustand	Eingabe	Ausgabe	Folgezustand
1	Z0	1 Euro	nichts	Z1
2	Z1	drehen	nichts	Z1
3	Z1	1 Euro.	nichts	Z2
4	Z2	drehen	Blume	Z0

Die Eingaben bilden ein *Eingabewort* W = (1 Euro, drehen, 1 Euro, drehen). Die Zustandswechsel können in der *Zustandskette* K = (Z0, Z1,Z1,Z2, Z0) zusammengefasst werden. Als Ausgabe wird das *Ausgabewort* V = (nichts, nichts, nichts, Blume) erzeugt.

Markow-Kette – Grundbegriffe

Gegeben sei eine *Folge von Zufallsexperimenten* mit dem *endlichen Zustandsraum* $Z = \{Z_1, Z_2, ...Z_n\}$. Es möge jeweils nur einer der n Zustände eintreten.

Diese Folge bildet eine *endliche Markow-Kette*, wenn die *Übergangswahrscheinlichkeiten* $p_{ji}(t)$, (i, j = 1..., n), dass der Zustand Z_i im t-ten Experiment (Übergang, Schritt) eintritt, nur davon abhängt, welcher Zustand Z_j im (t–1)-ten Experiment vorlag. Die Übergangswahrscheinlichkeiten werden in einer *Übergangsmatrix S* zusammengefaßt. Sind diese Übergangswahrscheinlichkeiten auch noch unabhängig von der Nummer des Experiments, so spricht man von einer *endlichen homogenen Markow-Kette*.

Für homogene Markow-Ketten sind also die Übergangsmatrizen von Experiment zu Experiment gleich. Für den Start des Systems liegt in der Regel eine *Anfangsverteilung* v_0 mit den augenblicklichen Wahrscheinlichkeiten für die einzelnen Zustände vor.

Eine endliche homogene Markow-Kette kann also durch das Tripel (Z, S, v_0) beschrieben werden.
Z Zustandsraum, S Übergangsmatrix,
v_0 Anfangsverteilung

Kernstück der Theorie über Markow-Ketten ist der folgende Satz:

Grenzwertsatz: Gegeben sei eine endliche homogene Markow-Kette (Z, S, v_0) mit der Anfangsverteilung v_0 , der Übergangsmatrix S und dem Zustandsraum $Z = \{Z_1, Z_2, ..., Z_n\}$. Wenn es ein t gibt, so dass die Übergangsmatrix S^t (Übergangsmatrix nach t Schritten) mindestens eine Spalte k_0 hat, in der alle Elemente positiv sind, dann existieren

a) alle Grenzwerte $lim\ p_{jk}^{(t)},\ t \to \infty$ (Grenzwahrscheinlichkeiten), der t-stufigen Übergangswahrscheinlichkeiten, und es gilt $\lim p_{jk}^{(t)} = g_k$ (unabhängig von den Zeilennummern j) für alle Zustände Z_j und Z_k des Zustandsraums. Dabei ist die Summe aller g_k gleich 1. Die Matrix $S^\infty = G$ aus den Grenzwahrscheinlichkeiten hat also lauter gleiche Zeilen
$g = [g_1, g_2, ..., g_n]$.

b) $g = [g_1, g_2, ..., g_n]$ ist unabhängig von der Anfangsverteilung v_0 .

c) $g = [g_1, g_2, ..., g_n]$ ist die einzige Lösung des linearen Gleichungssystems
w = wS mit $w_1 + w_2 + ... + w_n = 1$. Die stationäre Verteilung w ist also unter den angegebenen Voraussetzungen gleichzeitig auch Grenzverteilung: w = g.

Dieser Satz lässt sich im Unterricht
- an Beispielen (für zwei oder mehr Zustände) plausibel machen, da man ja einfach nachrechnen kann oder auch
- für n = 2 Zustände beweisen.
- Ein Beweis für n>2 kann nur mit leistungsstarken Schülern besprochen werden. Er wird in *[Leh86b], S. 101 f.* allgemein, daneben aber auch mit einem begleitenden Zahlenbeispiel durchgeführt.

Markow-Ketten mit endlich vielen Zuständen sind auch Beispiele für endliche Automaten. Häufig lassen sich Systeme durch absorbierende Markow-Ketten beschreiben, siehe Kapitel 3.3.4 (Crap-Spiel).

Als ein Weg (von vielen möglichen Wegen) durch eine Unterrichtseinheit „Zustandsgraphen" – gedacht für Leistungskurse – wird der in Abb. 3.3.1-b skizzierte Weg vorgeschlagen.

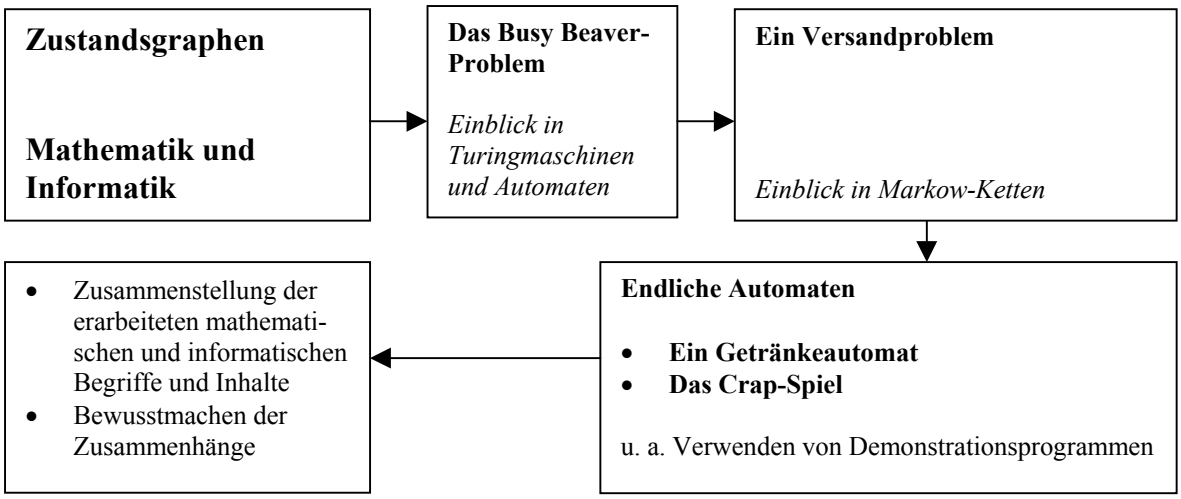


Abb. 3.3.1-b

Die bei diesem Lehrgang auftretenden Vernetzungen (und weitere, hier nicht verfolgte Zusammenhänge) sind vielfältig; Abbildung 3.3.1-d (Seite 112) sagt Einiges darüber aus.

Hinweis:

Ein weniger anspruchsvoller Weg (für Grundkurse oder für Wahlpflichtunterricht in Klasse 10, Berlin) würde sich bei Behandlung von **Markow-Ketten mit nur 2 Zuständen** ergeben, zumal hier mehrere **Visualisierungsmöglichkeiten** bestehen. Hierzu wird u. a. auf das auf dieser Seite genannte Buch *[Leh86b]* und auf das Softwarepaket
Lehmann, E.: Markow-Ketten, ein Programmsystem unter MS-DOS, Leh-Soft, Berlin 1986,
verwiesen.

Abbildung 3.3.1-c gibt einen Überblick über die diversen inhaltlichen Ansatzpunkte bei der Einbeziehung von Markow-Ketten mit zwei Zuständen in den Unterricht.

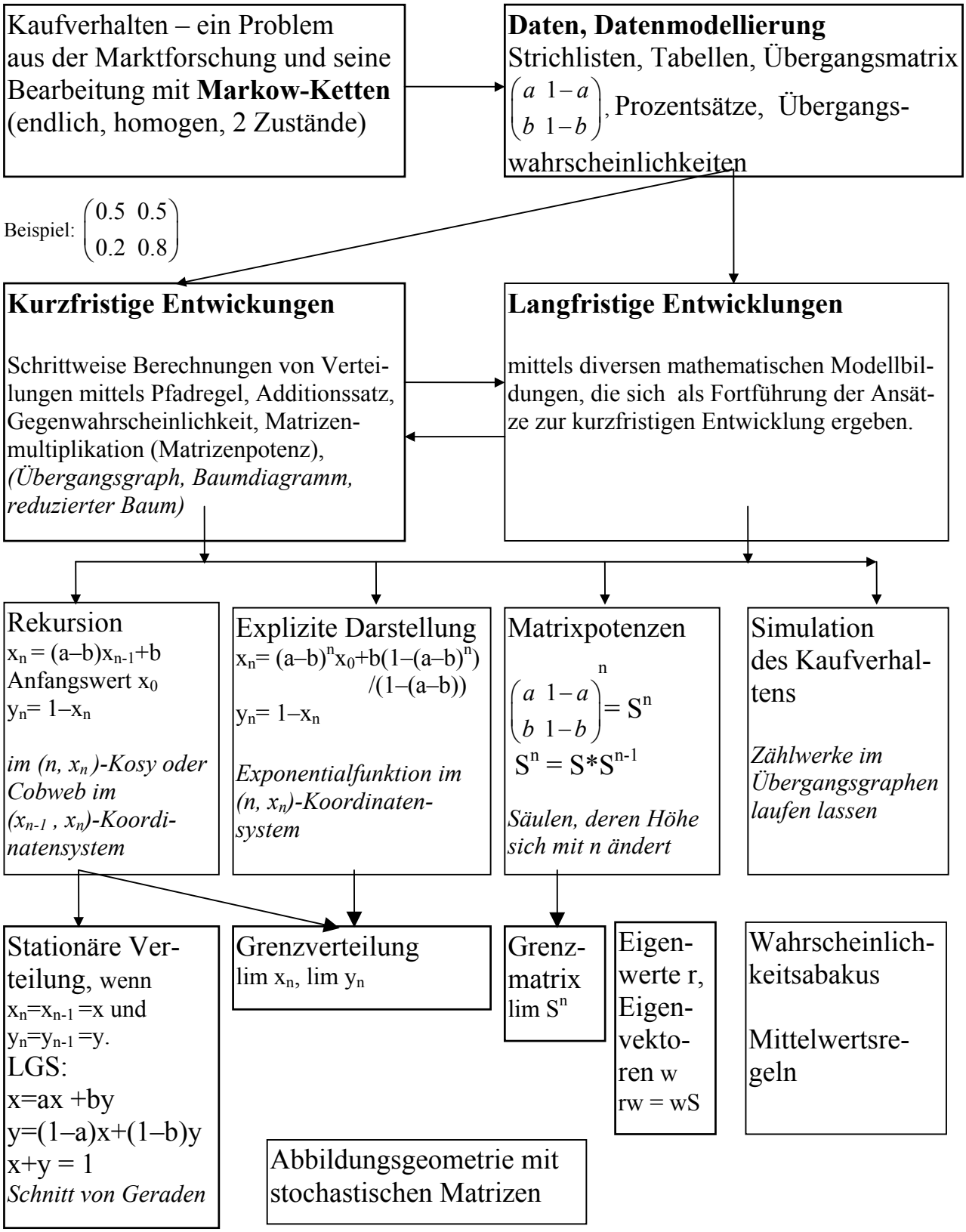


Abb.3.3.1-c: Markow-Kettten mit 2 Zuständen, Überblick zu gebietsübergreifenden Ansätzen im Mathematik und Informatikunterricht; *kursiv: grafische Darstellungsmöglichkeiten*

Unterricht nach Abbildung 3.3.1-d geht von je einer Problemstellung aus Informatik (Busy-Beaver-Problem) und Mathematik (Versandproblem) aus. Modellbildungsprozesse führen zu verschiedenen mathematischen und informatischen Inhalten. Je nach Unterrichtsvoraussetzung (u. a. Lehrplan) und den zeitlichen und organisatorischen Möglichkeiten können nun verschiedene Wege eingeschlagen werden, wobei für die Mathematik besonders die Vernetzungsmöglichkeiten innerhalb der drei Standardgebiete Analysis, Lineare Algebra und Stochastik interessieren. Zu diesen Gebieten gesellen sich nun noch die informatischen Ansätze.
In den Kapiteln 3.3.2 und 3.3.3 werden die oben genannten Kernprobleme (Busy-Beaver-Problem, Versandproblem) näher ausgeführt.

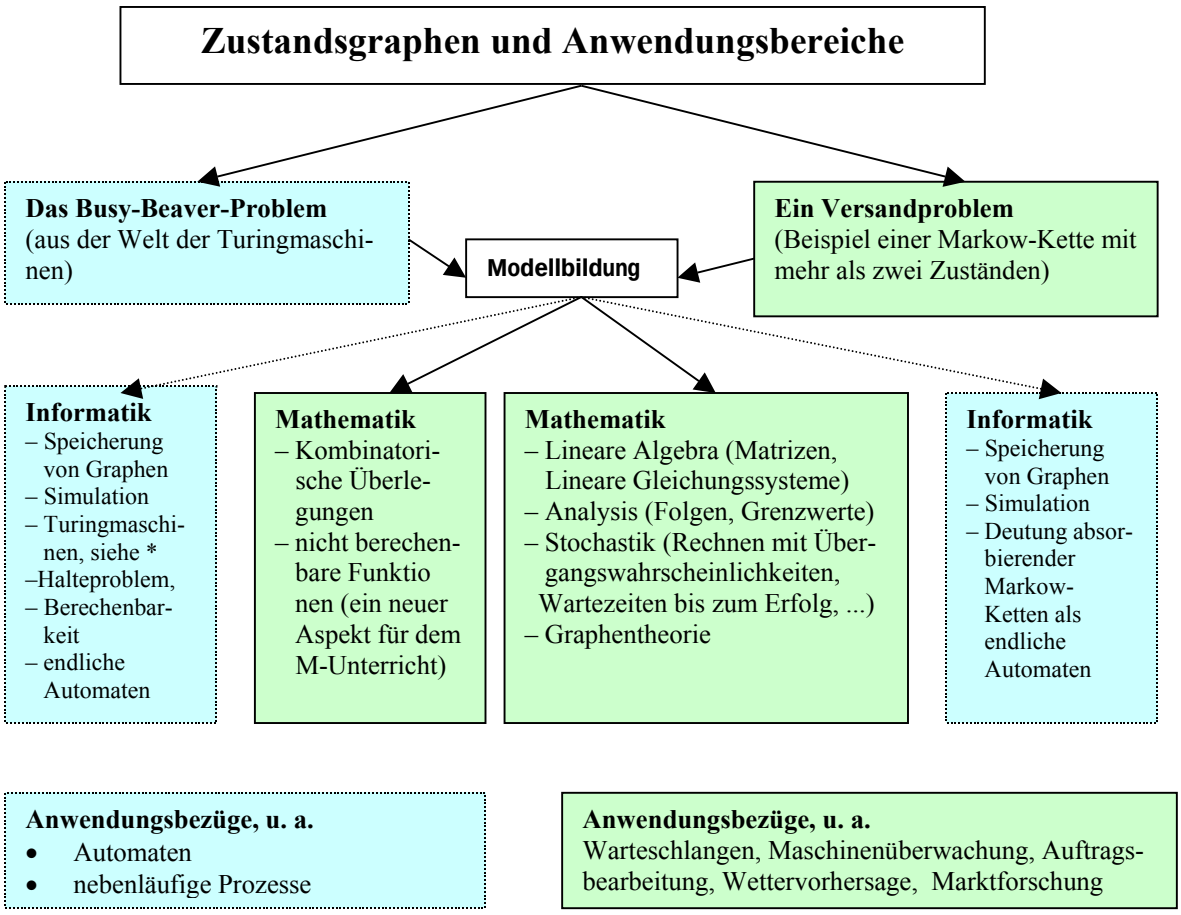


Abb. 3.3.1-d: Algorithmen und Anwendungen von Zustandsgraphen und verwandten Problemen im Mathematik- und Informatikunterricht

* 1936 hat der englische Mathematiker **A.M.Turing** (1912-1954) ein universelles Automatenmodell vorgeschlagen. **„Turingmaschinen"** sind spezielle Automaten, siehe Seite 108. Man kann sich eine Turingmaschine als ein **Band** vorstellen, das **in Felder unterteilt** ist, auf denen genau ein **Zeichen** steht. Ein **Lese- bzw. Schreibkopf** kann den Inhalt des gerade darunter befindlichen Feldes lesen und beschreiben. **Das Band merkt sich** also **Zwischenergebnisse**. Das **Band kann sich** jeweils um ein Feld nach links oder rechts **bewegen**. Dabei werden jeweils gewisse **Zustände aus einer endlichen Zustandsmenge** angenommen. Diese Aktionen werden **durch ein Programm gesteuert**, das zu den Zustandsänderungen führt.
Das Turingmaschinenmodell ist ein wichtiges Modell zur Untersuchung von Grundlagenproblemen der theoretischen Informatik, in Zusammenhang mit dem Begriff „Berechenbarkeit" und dem „Halteproblem".

3.3.2 Das Busy-Beaver-Problem – Turingmaschinen

Man nehme ein nach beiden Seiten **unendliches Band**, das **Bandalphabet** { #, / } und eine **Turingmaschine** mit n Zuständen Z0, Z1, Z2, Z(n-1) und einem zusätzlichen Haltezustand ZE.
In jedem Schritt soll die Turingmaschine eines der Symbole # oder / schreiben und eine Bewegung nach links oder rechts machen oder aber anhalten. Die Turingmaschine mit n Zuständen, die auf dem anfangs leeren Band (anfangs lauter #-Zeichen) die meisten Striche schreibt, erhält den Titel **fleißiger Biber** (die Strichfolge darf Lücken enthalten).

Gründe für die Besprechung des Busy-Beaver-Problems im Mathematikunterricht

(1) Das Problem ist für kleine n (n aus {1,2,3,4}) leicht lösbar – siehe folgende Seiten –, aber für größere n ist ein regelrechter Wettbewerb entstanden, bei dem zu einer vorgegebenen Anzahl n von Zuständen immer fleißigere Biber konstruiert wurden. Insofern findet das Problem auch **bei Schülern großes Interesse**. Nach meinen Erfahrungen ist es ein „Selbstläufer".

(2) Es gibt **zahlreiche Möglichkeiten, das Problem in der Schule anzugehen**:
Materialsuche im Internet oder in der Literatur, Vorgabe und Analysieren von Algorithmen, eigenes Entwickeln von Algorithmen (als Tabellen, Zustandsgraphen, Programm), experimentelles Arbeiten mit Programmen.

(3) Für die Informatik (und für die Mathematik) ist das Problem wichtig als Beitrag für die Frage nach der **Berechenbarkeit von Funktionen** und als Beitrag zum **Halteproblem**.

(4) Im Informatikunterricht kann das Problem ein ganzes Feld von Folgethemen erzeugen (siehe Abb. 3.3.1-d), u. a. die Themen: **Endliche Automaten, Turingmaschine**.

(5) Das Thema ist auch für den Mathematikunterricht interessant. Dabei geht es um einige Überlegungen kombinatorischer Art und wie oben schon formuliert um Zusammenhänge mit Themen der Mathematik wie die Anwendungen von Markow-Ketten. Zudem stellt das Thema **„nichtberechenbare Funktionen"** (bisher völlig vernachlässigt) eine wichtige Ergänzung zu den vielen berechenbaren Funktionen im Mathematikunterricht dar.

Fleißige Biber – das Busy-Beaver-Problem

Den Unterricht zum Thema „Zustandsgraphen" könnte man z. B. folgendermaßen mit einem Arbeitsblatt beginnen.

(1) Der Übergangsgraph von Abbildung 3.3.2-a stellt einen Algorithmus dar, der das „Busy-Beaver-Spiel" für den Fall von 3 Zuständen Z0 (Anfangszustand), Z1, Z2 (Zwischenzustände) und einem Endzustand ZE beschreibt.
(2) Dieser Algorithmus arbeitet auf einem unendlich langen Band mit lauter Kreuzen (#), die beim Abarbeiten des Algorithmus im Verlauf der Arbeit teilweise durch Striche (/) ersetzt werden.
(3) Das gegebene Band sei ... # ...
(4) Das Programm startet an der Stelle ↑ im Zustand Z0 ↑
und arbeitet nun nach dem Algorithmus von Abb. 3.3.2-a.

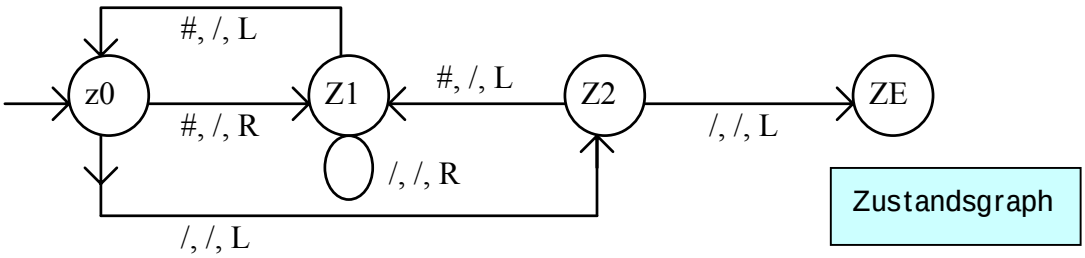


Abb. 3.3.2-a: Algorithmus als Zustandsgraph

Fragestellung:
Wie viel Striche auf dem Band erzeugt der Algorithmus (Lücken sind erlaubt)?
Wie viel Schritte benötigt er dafür?
Oder
Wie viel Baumstämme legt der fleißige Biber ab?

Erläuterung des Zustandsgraphen
Beim Start in Z0 sind laut Zustandsgraph (auch Übergangsgraph genannt) zwei Möglichkeiten:

a1) Zeigt der Bandzeiger auf #, so wird an diese Stelle ein / geschrieben und der Zeiger bewegt sich um eine Stelle nach rechts.
a2) Zeigt der Bandzeiger auf /, so wird das Zeichen / beibehalten und der Zeiger bewegt sich um eine Stelle nach links.

Entsprechendes sagt der Übergangsgraph für die Zustände Z1 und Z2 aus. Bei Zustand ZE endet das Programm.

Weitere Darstellungsformen des Algorithmus
Für den Algorithmus gibt es in der Literatur auch noch andere Darstellungsformen, die insbesondere für längere Algorithmen nützlicher sind. Genannt seien noch die Tabellenform und die 5-Tupelform.

Algorithmus in Tabellenform (Turingtafel)

	/	#
Z0	(/, L, Z2)	(/, R, Z1)
Z1	(/, R, Z1)	(/, L, Z0)
Z2	(/, L, ZE)	(/, L, Z0)
ZE	---	---

Algorithmus in Form eines Turingprogramms, 5-Tupel

(Z0, /, /, L, Z2)
(Z0, #, /, R, Z1)
(Z1, /, /, R, Z1)
(Z1, #, /, L, Z0)
(Z2, /, /, L, ZE)
(Z2, #, /, L, Z1)

Abb. 3.3.2-b: Weitere Darstellungsformen von Algorithmen für Turingmaschinen

Die 5-Tupel sind so aufgebaut:
(aktueller Zustand, gelesenes Zeichen, zu schreibendes Zeichen, Bewegung des Bandkopfes, Folgezustand)

Lösung
Beim Analysieren des Übergangsgraphen in Abb.3.3.2-a oder des Programms in Abb. 3.3.2-b zeigt sich, dass der Algorithmus auf dem Band 6 Striche (/) erzeugt. Er benötigt dazu 13 Schritte.

Simulationsprogramm

Ein anderer Unterrichtseinstieg könnte mit Hilfe eines Simulationsprogrammes erfolgen. Das Programm BEAVER3 *[Lehmann, Eberhard: Die Turingmaschine im Anfangsunterricht – ein Bericht von den ersten Stunden eines Informatikkurses in Klasse 11, in LOGIN 1999, Heft 6, S. 44–52]* zeigt eine Simulation eines Turing-Programms für den Fall n = 3 Zustände. Beginnend mit einem leeren Band (überall #-Zeichen) werden die Schritte des Kopfes einzeln auf Tastendruck abgearbeitet, Anzeigen auf der Oberfläche dienen der Veranschaulichung der Vorgänge. Man kann sich auf diese Weise gut in die Problematik einarbeiten. Die Oberfläche des Programms ist in Abbildung 3.3.2-c erkennbar, abgedruckt ist ein Zwischenstand bei der Programmabarbeitung. Man sieht, dass zur Zeit 5 Striche erzeugt sind. Das Programm erzeugt insgesamt sechs Striche und bricht dann ab.

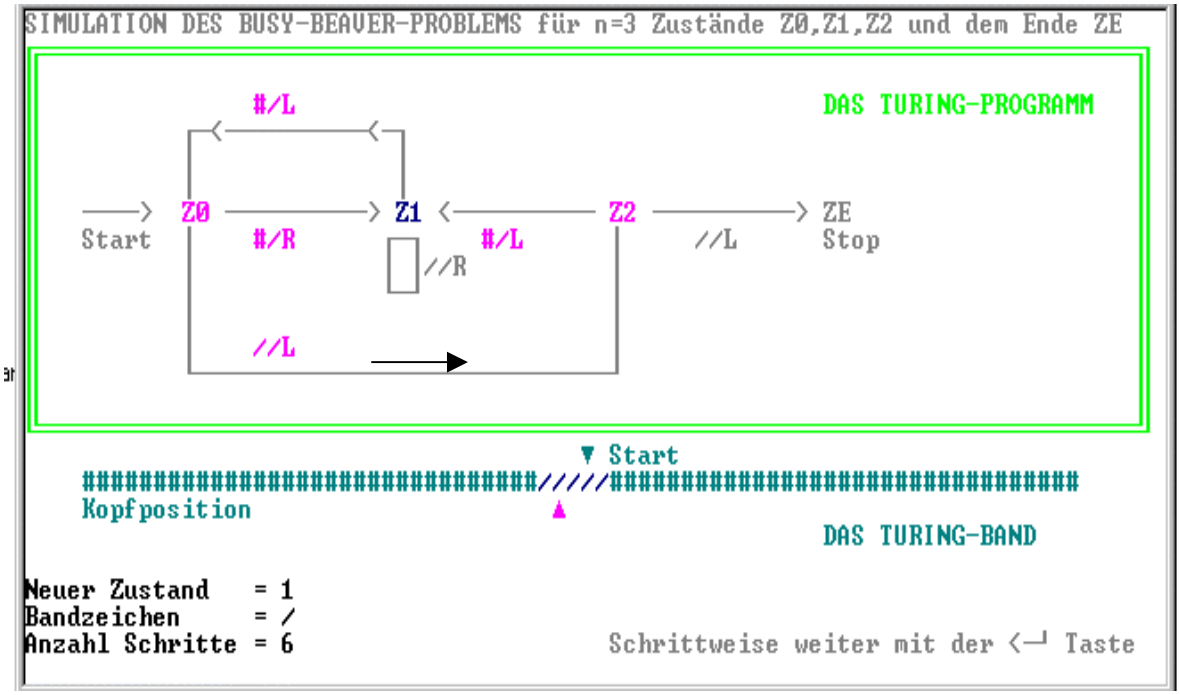
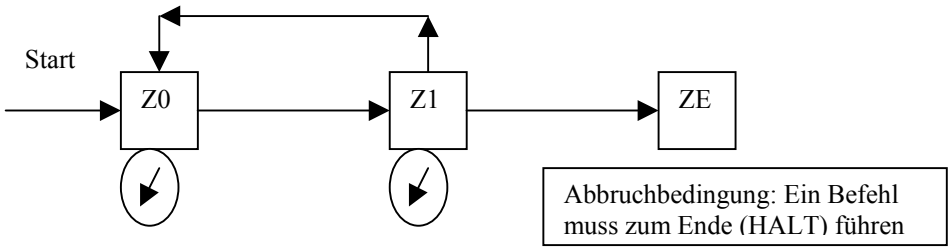
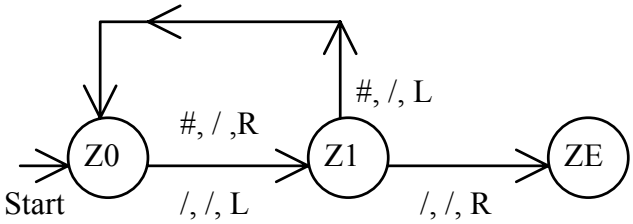


Abb.3.3.2-c: Simulationsprogramm für das Busy Beaver-Problem mit n = 3 Zuständen und Endzustand

Nach dieser Einführung in die Problematik können die Schüler zunächst den Fall n = 2 Zustände untersuchen. Der Übergangsgraph muss dazu passend beschriftet werden.



In jedem Zustand kann der Biber auf 2 Fälle stoßen: Er findet das Zeichen # oder /. Also können wir in jedem Zustand 2 Turing-Befehle vergeben. Aber welche sind die, die möglichst viele Striche erzeugen? Probieren! Als Busy Beaver stellt sich das Programm von Abbildung 3.3.2.-e heraus.



Busy Beaver für n = 2

Abb. 3.3.2-e: Busy-Beaver für n = 2 Zustände (Z0 und Z1) und den Endzustand ZE

Unser Biber schafft mit der Zustandsfolge {Z0, Z1, Z0, Z1, Z0, Z1, ZE} 4 Baumstämme heran. – Wie wäre es mit dem folgenden Programm?

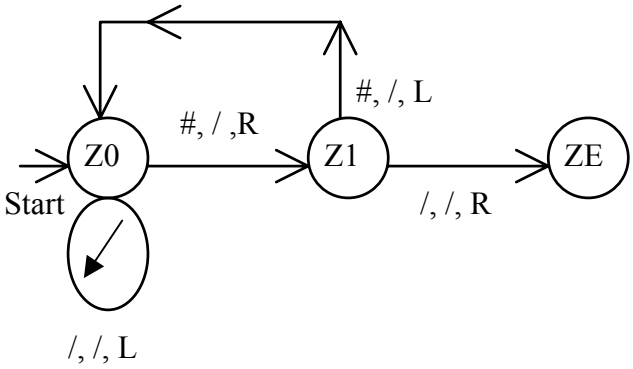


Abb. 3.3.2-f: Dieser Biber ist etwas weniger fleißig, er schafft nur 3 Baustämme heran. – Kein Busy-Beaver!

Selbstverständlich könnte man z. B. mit (#, /, R) beliebig viele Striche erzeugen, aber wie bricht man ab? Man muss dazu /, /, ... benutzen, also vorher irgendwann einmal nach links gehen, um auf einen Strich zu treffen! **Man erkennt hieran, dass es ein besonderes Problem ist, die richtige HALT-Bedingung an der richtigen Stelle zu finden.**

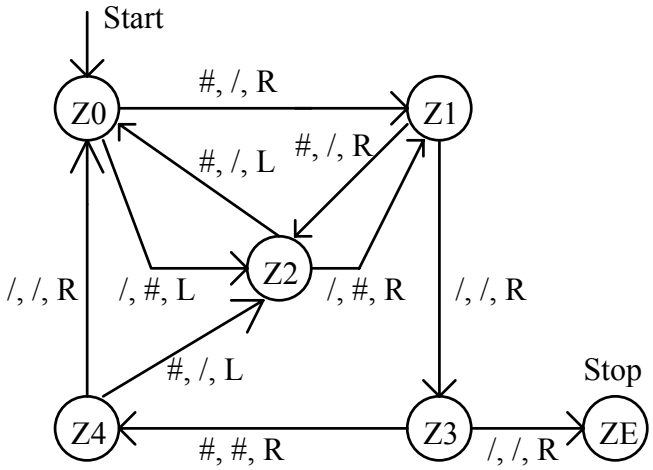
Hinweis:
Der Ablauf eines Turing-Programms lässt sich gut durch Aufschreiben des jeweils aktuellen Zustands merken. Dabei entsteht eine **Zustandsfolge**. So kann man beispielsweise gut mit einer anderen Lösung vergleichen oder seine eigene Lösung rekonstruieren.

Zustandsfolge für Abb. 3.3.2-e: {Z0, Z1, Z0, Z1, Z0, Z1, ZE}
Zustandsfolge für Abb. 3.3.2-f: {Z0, Z1, Z0, Z0, Z1, ZE}

Mit dem Begriff der Zustandsfolge bereitet man auch gleichzeitig spätere mathematische Überlegungen an Markow-Ketten vor – siehe Kapitel 3.3.4.

Die Fälle n = 4 (13 Striche) und n = 5 wurden dann im Verlauf des Unterrichts der Turingmaschine übergeben und zum Erstaunen der Schüler war die Anzahl der Striche bei n = 5 (Graph siehe unten) bereits auf 501 Striche bei 134467 Schritten angewachsen. Die Problematik wurde immer deutlicher; offenbar kommt es bei den entworfenen Turingprogrammen darauf an, die Maschine irgendwann zum Halten zu bringen. Für den Fall n = 5, Strichzahl 501, wird

hier noch der Übergangsgraph angegeben. Es gibt für n = 5 aber noch bessere Algorithmen, so dass es sich hier nicht um einen *Busy*-Beaver handelt.



n = 5 Zustände

Abb.3.3.2-g: Beaver für 5 Zustände (und Endzustand), 501 Striche, aber kein Busy-Beaver

nach *[Gas92]*.

Die Betrachtungen – schon für das noch recht kleine n = 5 – zeigen, dass die Arbeitsweise der Turing-Maschine nur noch schwer zu durchschauen ist. Insbesondere ergibt sich die Frage, wann die Maschine anhalten soll. Wenn man z. B. mit seinem Algorithmus bei 501 Strichen (und damit bei einem möglichen Maximum) in Zustand Z3 angekommen ist, ergibt sich die Frage, ob man durch Fortsetzung, z. B. mit (Z3, /, /, R, Z1), vielleicht noch zu einer höheren Strichzahl kommt oder ob man anhalten soll (Z3, /, /, R, ZE).

Nachdem der Umgang mit dem Busy Beaver-Problem durch verschiedene Aktivitäten gefestigt ist, kann man sich Fragen zuwenden, die zu allgemeineren Aussagen führen.

Satz 1: Zu jeder Anzahl n von Zuständen muss ein Busy Beaver existieren!

Die Aussage ist plausibel, da zu dem Busy Beaver eine spezielle Turingtafel gehört und es ersichtlich nur endlich viele Turingtafeln zu jedem n und dem Bandalphabet B_2 = { /, # } gibt. Satz 2 sagt aber noch Präziseres aus!

Satz 2: Die Anzahl der Turingmaschinen mit n Zuständen beträgt $T(n)=(4(n+1))^{2n}$ **.**

Beweis
Zum besseren Verständnis werden die allgemeinen Überlegungen mit dem Spezialfall n = 3 unterstützt. Hierzu ist die obige Darstellung des Algorithmus in Tabellenform nützlich:

Algorithmus in Tabellenform als Turingtafel

Aktueller Zustand	Aktueller Bandinhalt ist /		Aktueller Bandinhalt ist #	
Z0	(/, L, Z2)	*(1,1)*	(/, R, Z1)	*(1,2)*
Z1	(/, R, Z1)	*(2,1)*	(/, L, Z0)	*(2,2)*
Z2	(/, L, ZE)	*(3,1)*	(/, L, Z1)	*(3,2)*
ZE	---		---	

Abb. 3.3.2-h

(1,1), (1,2), ... sind die Tabellenpositionen

n Zustände (dazu kommt ein Endzustand)	3 Zustände (und ein Endzustand), siehe Turingtafel Abb. 3.3.2-h
Jede zugehörige Turingtafel hat 2*n Einträge (für die Nicht-Endzustände).	In diesem Fall sind es 2*3 = 6 Einträge
Man könnte nun alle diese Turingtafeln auflisten und diejenige heraussuchen, die die meisten Striche (/) erzeugt.	für n = 3 ist das gerade die Tafel von Abb.3.3.2-h. Man kann aber für n=3 auch noch andere Tafeln notieren. Wie viel andere?
Jeder Eintrag in die Tafel besteht aus einem Tripel.	z. B. (/, L, Z2)
Insgesamt gibt es an jeder Tabellenposition (n+1)*2*2 Tripel: (n+1) Zielzustände {Z0, Z1, .. Z(n–1), ZE}, 2 zu schreibende Zeichen {/, #}, 2 Richtungen auf dem Band {L, R}.	In diesem Fall sind es 4*2*2 = 16 mögliche Tripel. 4 Zielzustände, 2 Zeichen, 2 Richtungen. Für die Position *(1,1)* der Tabelle sind das: (/, L, Z0), (/, L, Z1), (/, L, Z2), (/, L, ZE), (/, R, Z0), (/, R, Z1), (/, R, Z2), (/, L, ZE), (#, L, Z0), (#, L, Z1), (#, L, Z2), (#, L, ZE), (#, R, Z0), (#, R, Z1), (#, R, Z2), (#, L, ZE),
Insgesamt gibt es damit $T(n)=(4(n+1))^{2n}$ Turingmaschinen. 2*n ist die Anzahl der Positionen in der Turingtafel.	Es gibt in der Turingtafel 6 Positionen, jede Position ist mit einem von 16 Tripeln besetzt. Damit gibt es insgesamt $16*16*16*16*16*16 = 16^6$ Tripel. Nach der Formel ist $T(3) = (4*(3+1))^{2*3}$, also ebenfalls gleich 16^6.

Abb. 3.3.2-i: Überlegungen zum Beweis von Satz 2

Die Anzahl der möglichen Turingtafeln, berechnet mit einem CAS, kann man Abbildung 3.3.2-j entnehmen.

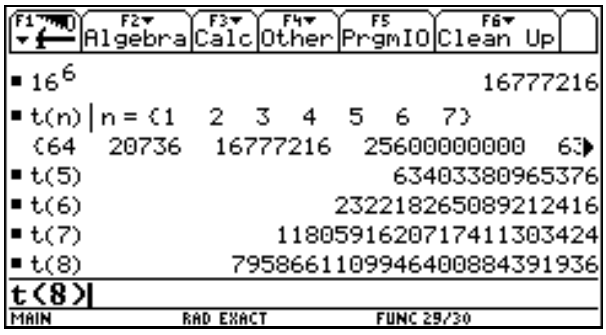


Abb. 3.3.2-j: Berechnung von Werten der Funktion T(n)

Es ist also nur mit gewaltigem Aufwand verbunden, alle möglichen Turingtafeln ausrechnen zu wollen, um den jeweiligen Busy Beaver zu finden, zumal man außerdem zu jeder Tafel den Algorithmus ablaufen lassen muss, um damit die Anzahl der Striche und Schritte zu finden. Den Versuch eines passenden Algorithmusses für das Problem findet man auf Seite 121.

Die Zusammenfassung bisheriger Ergebnisse und einiger Ergänzungen ergibt die Tabelle in Abbildung 3.3.2-k

Anzahl der Zustände (ohne Endzustand) n	Anzahl der möglichen Turingtafeln $T(n)=(4(n+1))^{2n}$	Busy Beaver Anzahl der Striche $\sum(n)$	Busy Beaver Anzahl der Schritte S(n)
1	64	1	
2	20 736	4	
3	16 777 216	6	13
4	25 600 000 000	13	
5	63 403 380 965 376	$\geq$ 4098	
6	28^{12}		
7	32^{14}		
8	36^{16}		
k	berechenbare Funktion	nicht berechenbare Funktion	nicht berechenbare Funktion

Abb. 3.3.2-k

Zum Fall n = 5 schreibt *A. K. Dewdney*

„Der Sprung von $\sum = 13$ bei n=4 auf $\sum \geq 4098$ bei n=5 ist symptomatisch für die nichtberechenbare Natur von $\sum$. Die Zahl 4098 hat eine interessante Geschichte hinter sich.

In der Augustausgabe 1984 des *Scientific American* erschien ein Artikel über den damals fleißigsten 5-Zustands-Biber, der 1984 von Uwe Schult, einem deutschen Informatiker, gefunden wurde. Schults fleißiger Biber erzeugt 501 Einsen, bevor er anhielt. Als Antwort auf den Artikel führte George Uhing, ein amerikanischer Programmierer, eine Computersuche nach fleißigen Bibern mit fünf Zuständen durch und fand einen, der 1915 Einsen [*Anm.: Striche*] ausgab, bevor er anhielt. Später, 1989, wurde Uhings Rekordbiber durch einen neuen, fleißigeren Biber verdrängt, der von Jürgen Buntrock und Heiner Marxen in Deutschland entdeckt wurde. Der Buntrock-Marxen-Biber, der bei einer dreitägigen Suche auf einem Hochgeschwindigkeitsrechner gefunden wurde, schreibt 4098 Einsen, bevor er anhält!" *[A. K. Dewdney: Der Turing-Omnibus – eine Reise durch die Informatik mit 66 Stationen, Springer-Verlag, Berlin-Heidelberg, 1995, S. 287]*

Zur Vorbereitung der folgenden Betrachtungen ist es zweckmäßig, die oben bereits verwendete Funktion $\sum(n)$ genauer zu definieren, siehe auch *[Gas92], Seite 205 f.]*.

Definition: Die Funktion $\sum: \mathbb{N} \to \mathbb{N}$ sei definiert durch $\sum(n)$:= Maximale Anzahl von Strichen, die eine haltende (Erreichen eines Endzustands) Turingmaschine mit n Zuständen und dem Bandalphabet {/, #} auf das Turingband – bestehend aus lauter #-Zeichen – schreibt.

1) Zu jedem $n \in \mathbb{N}$ gibt es nur endlich viele Turingmaschinen, ihre Anzahl sei gleich T(n), siehe oben,
2) von diesen Turingmaschinen halten bei Start auf einem leeren Band (nur #-Zeichen) endlich viele, ihre Anzahl sei gleich TH(n),
3) diese TH(n) Turingmaschinen erzeugen dabei auch nur eine endliche Menge von Strichanzahlen (diese Menge besitzt ein Maximum).

- Damit gehört zu jedem $n \in \mathbb{N}$ eine natürliche Zahl $\sum(n)$ und $\sum$ erweist sich als eine in üblichem Sinne mathematisch definierte Funktion.

Ein Algorithmus zur Berechnung von $\sum(n)$ für jedes n könnte folgendermaßen arbeiten:

(1) Eingabe von n
(2) Ermitteln aller Turingmaschinen mit n Zuständen (ihre Anzahl ist T(n))
(3) Ermitteln aller nicht haltenden Maschinen und diese aussondern.
(4) Alle haltenden Turingmaschinen (ihre Anzahl ist TH(n)) auf einem leeren Band (lauter #-Zeichen) starten
(5) Jeweils die entstehenden /-Anzahlen notieren (in einer Menge M sammeln)
(6) Das Maximum in M nehmen. Das ist dann der Busy-Beaver.

Dieser Algorithmus arbeitet jedoch nicht wie gewünscht, weil es in (3) nicht gelingt, alle nicht haltenden Maschinen zu finden. Damit sind auch (4)-(6) irrelevant.

Mit den vorstehenden Überlegungen ist nun auch der Begriff der berechenbaren Funktion vorbereitet. *Duden Informatik, Duden-Verlag, 1993, S.80 f.:*

> „Eine Funktion f: M→ N heisst berechenbar, wenn es einen Algorithmus gibt, der für jeden Eingabewert m aus M, für den f(m) definiert ist, nach endlich vielen Schritten anhält und als Ergebnis f(m) liefert; in allen Fällen, in denen f(m) nicht definiert ist, bricht der Algorithmus nicht ab. – Anschaulich gesprochen (Anm.: und für die Schüler sehr handlich) ist eine Funktion berechenbar, wenn man sie programmieren kann."

Im normalen Mathematikunterricht wird angesichts des dort verwendeten Funktionenmaterials auf den Begriff der berechenbaren Funktion nicht eingegangen, ein Mangel, der sich bei Berücksichtigung informatischer Aspekte, wie oben gezeigt, beseitigen lässt! Man beachte dazu auch Kapitel 3.4.2.

Satz 3: $\sum(n)$ **(Anzahl der Striche bei n Zuständen bis der Busy Beaver ermittelt ist) und**
S(n) (Anzahl der Schritte bei n Zuständen bis der Busy Beaver ermittelt ist)
sind nicht berechenbare Funktionen.

Der Beweis dieses Satzes ist für den Schulunterricht wohl zu schwierig. Man findet ihn u. a. im Internet bei *[Reinhard Völler (http://www.informatik.fh-hambung.de/~voeller/th/thinf/node16/html)]*. Außerdem wird erneut auf das oben genannte Buch *[Gas92]* verwiesen.
An der gleichen Stelle findet man auch einen Kandidaten für einen Busy Beaver mit 6 Zuständen. Nach den dortigen Angaben soll das folgende Programm (für diese Arbeit etwas umgeschrieben) 95 524 079 Striche erzeugen und nach 8 690 333 381 690 951 Schritten anhalten. Hier das Programm:

(Z0, /, /, R, Z0)	(Z0, #, /, R, Z1)
(Z1, /, /, L, Z1)	(Z1, #, /, L, Z2)
(Z2, /, /, L, Z3)	(Z2, #, #, R, Z5)
(Z3, /, #, L, Z4)	(Z3, #, /, R, Z0)
(Z4, /, /, L, Z5)	(Z4, #, /, L, ZE)
(Z5, /, #, L, Z2)	(Z5, #, #, L, Z0)

Abb. 3.3.2-l: Auf der Suche nach einem Busy Beaver für n = 6 Zustände, zzgl. einem Endzustand ZE.

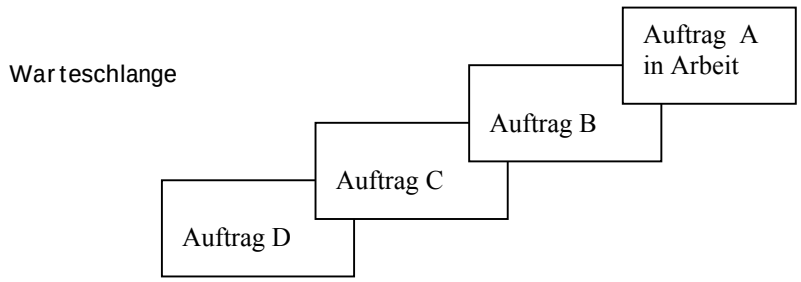
Zusammenfassung:
Die Darstellung dürfte gezeigt haben, dass die obigen Inhalte auch für einen Mathematik-Leistungskurs geeignet sind. Besondere Informatikkenntnisse sind nicht notwendig, man kann mit dem Thema quasi „bei Null" anfangen. Der Wert der Betrachtungen für einen Mathematikkurs liegt u.a. in dem Kennenlernen, Benutzen und Entwerfen ungewohnter Algorithmen und nicht berechenbarer Funktionen. Sollte der Mut fehlen, derartige Überlegungen in den normalen Unterricht zu integrieren (etwa im Rahmen eines Stochastikkurses in Zusammenhang mit Markow-Ketten, siehe Kapitel 3.3.3 und 3.3.4), bleibt die Möglichkeit, einen der Projekttage der Schule für das Vorhaben zu nutzen.

3.3.3 Ein Versandproblem – Markow-Ketten Vernetzung zwischen Mathematik und Informatik

In Kapitel 3.3.1 wurden bereits einige Gedanken über Markow-Ketten formuliert und auf Bearbeitungsmöglichkeiten bei Vorliegen von zwei Zuständen hingewiesen. Gegenüber früheren Unterrichtsvorschlägen und Veröffentlichungen über Markow-Ketten können heutzutage Computeralgebrasysteme (CAS) eingesetzt werden, was zu neuen unterrichtlichen Möglichkeiten führt. Dazu gehört auch die Bearbeitung von Markow-Ketten mit mehr als zwei Zuständen – im folgenden Beispiel werden vier Zustände verwendet. Dabei können verschiedentlich auch informatische Inhalte berücksichtigt werden. Somit erweisen sich Markow-Ketten als besonders geeignet, die in dieser Arbeit angestrebten Ziele zu verdeutlichen. Hierfür kommen u. a. die folgenden Vernetzungen zwischen Mathematik und Informatik in Betracht:

- Die den Fächern gemeinsamen Methoden der **Modellbildung**,
- **die auftretenden Algorithmen**
  - – Speicherung von Tabellen / stochastische Matrizen,
  - – Matrizenmultiplikation, Matrizenpotenzen, weitere Matrizenoperationen,
  - – Bearbeitung linearer Gleichungssysteme (Gauss-Verfahren),
  - – Folgen (rekursiv definiert),
  - – Simulation,
- Veranschaulichung durch **Zustandsgraphen (Übergangsgraphen),**
- das in beiden Fächern nützliche Verfahren der **Simulation,**
- **Anwendungsbezug,**
- Einsatz von Computeralgebrasystemen und ggf. anderer Software.

Diese Ansätze werden in den Kapiteln 3.3.3 (Versandproblem) und 3.3.4 (Crap-Spiel) verfolgt. Als Ausgangsaufgabe in Kapitel 3.3.3 dient eine von mir im Jahr 2001 gestellte Leistungskurs-Abituraufgabe. Die Aufgabenstellungen entfalten sich an einer Warteschlange, die hier in Form eines Versandproblems auftritt. Über die Abituraufgabe hinaus werden Variationen von Problemstellung und Lösungen angeboten, um die Vernetzung zwischen Mathematik und Informatik (die in der Abituraufgabe nicht explizit benannt wurde) noch mehr zu verdeutlichen.



Markow-Ketten und die vielen Möglichkeiten ihrer Behandlung sind ein Musterbeispiel für die heute geforderten mathematischen Aktivitäten (offene Unterrichtsformen, neue Aufgabenkultur, Medieneinsatz). Auch der für den neuzeitlichen Unterricht viel propagierte Anwendungsbezug findet hier eine starke Berücksichtigung, da Markow-Ketten in vielen Anwendungssituationen wichtig sind. Das sind u. a.: Warteschlangenprobleme, Bevölkerungsbewegungen, Marktforschung, Buchstabenhäufigkeiten, Labyrinthe, Spiele.

Eine Abituraufgabe

Leistungskurs Mathematik (Lehmann) – Abitur 2001 – Lineare Algebra, Markow-Kette Kurs MA-3

Versandabteilung – Warteschlange

Eine Versandabteilung erhält an jedem Morgen zu Beginn der Arbeitszeit Aufträge. Der Auftragseingang an verschiedenen Tagen erfolge unabhängig voneinander. An einem Tag sollen höchstens zwei Bestellungen eingehen. Die Wahrscheinlichkeiten für den Eingang keiner, einer oder zweier Bestellungen seien 0.3, 0.5, 0.2. Die Erledigung eines Auftrags möge genau einen Tag in Anspruch nehmen. An einem Tag wird also stets nur ein Auftrag bearbeitet und erledigt. Falls schon vor den Neuzugängen am Morgen drei Aufträge vorliegen, wird höchstens noch ein Auftrag angenommen. Die Höchstzahl unbearbeiteter Aufträge ist also gleich 3.

1.1 Begründen Sie, dass eine (4,4)-Matrix als Übergangsmatrix geeignet ist. Bestätigen Sie die in der Übergangsmatrix S(4,4) fett markierten Werte durch geeignete Überlegungen.

S(4,4) =

	Z0	*Z1*	*Z2*	*Z3*
Z0	**0.8**	0.2	0	0
Z1	**0.3**	**0.5**	**0.2**	0
Z2	0	0.3	0.5	0.2
Z3	0	0	0.3	0.7

Übergangsmatrix
mit 4 Zuständen

1.2 In wie viel Prozent der Fälle liegen am Morgen des vierten Tages zwei Aufträge vor (ohne eventuelle Neuzugänge des vierten Tages), wenn am Morgen des ersten Tages ein Auftrag vorlag?

1.3 Wie entwickelt sich das System langfristig?

a) Begründen Sie, dass man diese Frage mit Hilfe eines LGS beantworten kann, und zeigen Sie, dass sich das folgende LGS ergibt:

t1	t2	t3	t4	Rs
–0.2	0.3	0	0	0
0.2	–0.5	0.3	0	0
0	0.2	–0.5	0.3	0
0	0	0.2	–0.3	0
1	1	1	1	1

mit der stationären Verteilung
T = [t1, t2, t3, t4]

und mit S(4,4) (siehe oben)

b) Lösen Sie das LGS mit dem CAS. Ergebnis auf 4 Nachkommastellen und in Bruchform!

c) Kommentieren Sie das Lösungstupel unter dem Aspekt, dass die Abteilungsleitung eine Verringerung des Personalbestands in Erwägung zieht.

1.4 In dem obigen Eingangstext zur Aufgabe heißt es u. a.: „Die Wahrscheinlichkeiten für den Eingang keiner, einer oder zweier Bestellungen seien 0.3, 0.5, 0.2." Dieser Satz wird nun abgewandelt zu „Die Wahrscheinlichkeiten für den Eingang keiner, einer oder zweier Bestellungen seien a, b, c mit a+b+c = 1." Definieren Sie einen Matrizenbaustein M(a, b, n) für die Potenzen der allgemeinen Übergangsmatrix zu obigem Versandproblem. *Hinweis: Es gilt: c = 1 – (a+b)*. Kontrollieren Sie den Baustein, indem Sie mit ihm die in 1.1. gegebene Übergangsmatrix erzeugen. Nennen Sie zwei über die obigen Fragen hinausgehende Problemstellungen, die sich mit dem Baustein bearbeiten lassen.

Ende der Abituraufgabe

Lösungsskizzen und Ergänzungen

Ergänzung 1: Zustandsgraph

Die in der Übergangsmatrix gegebenen Daten können in einem Zustandsgraphen dargestellt werden.

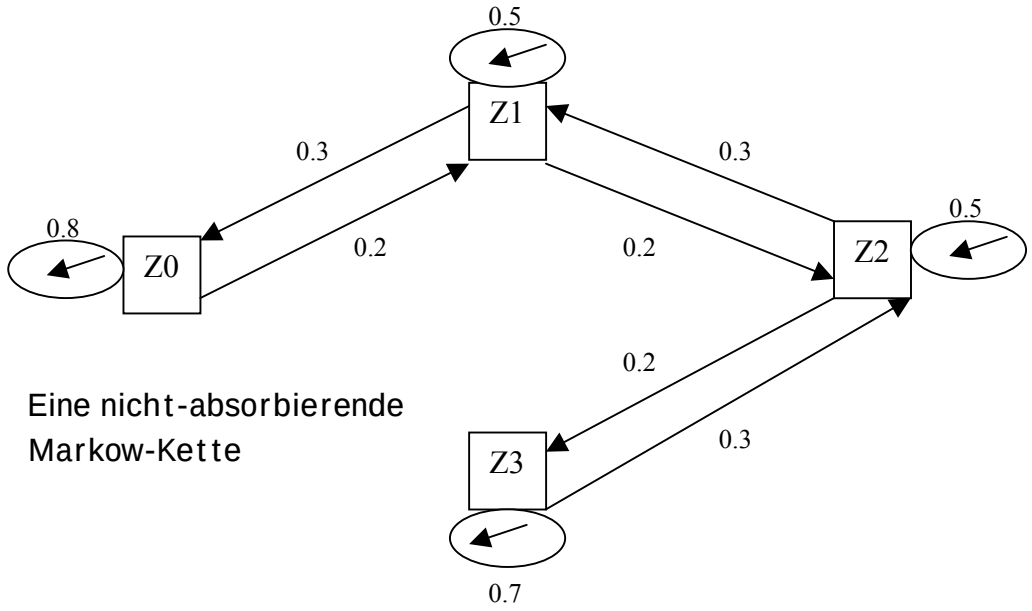


Abb.3.3.3-a : Zustandsgraph für das Versandproblem

Auffälligerweise gibt es bei diesem Zustandsgraphen keinen Endzustand. Es ist eine nicht-absorbierende Markow-Kette. In Kapitel 3.3.4 wird ein Beispiel für eine absorbierende Markow-Kette angegeben (Crap-Spiel-Automaten). Während die Kanten des Graphen bei dem Busy Beaver-Problem mit bestimmten Handlungsanweisungen versehen waren, werden hier die Kanten zwischen den Zuständen mit Wahrscheinlichkeiten bewertet.

Bearbeitung von Aufgabe 1.1

Als jeweiliger Zustand der Kette wird die Anzahl der unbearbeiteten Aufträge genommen. Das können 0, 1, 2, 3 noch vorliegende Aufträge sein, da die Höchstzahl unbearbeiteter Aufträge 3 sein soll.

Übergänge:

Z0 nach Z0: Es lag kein Auftrag vor. Im nächsten Zeitpunkt liegt wieder keiner vor. Eventuell eingehende Aufträge müssen also bearbeitet worden sein. D. h., es ist kein Auftrag (Wahrscheinlichkeit 0.3) oder ein Auftrag (Wahrscheinlichkeit 0.5) eingegangen, der dann am gleichen Tag bearbeitet wurde; zusammen also 0.3 + 0.5 = 0.8. Entsprechend kann für die anderen Übergänge argumentiert werden. Insgesamt entsteht schließlich die in Aufgabe 1.1 angegebene Matrix.

Bearbeitung von Aufgabe 1.2

- Eingabe der Übergangsmatrix S am CAS,
- Anfangsvektor definieren v = [0, 1, 0, 0],
- Ansatz: v * S^3 am CAS bilden,
- v * S^3 = [0.423, 0.323, **0.186,** 0.068],
- Auswertung der Ergebnismatrix, Ergebnis angeben: 0.186,
- Eingaben und Ergebnisse werden dokumentiert.

Bearbeitung von Aufgabe 1.3

a) Bei Existenz einer stationären Verteilung muss T*S = T sein (keine Änderung des Systems mehr), mit der Zusatzbedingung t0 + t1 + t2 + t3 = 1 (T ist sie eventuell existierende stationäre Verteilung).
T*S = T ist ein (4,4)-LGS, das umgeformt werden muss auf T*(S–E) = O und an das noch die Zeile t0 + t1 + t2 + t3 = 1 angehängt werden muss, so dass ein (5,4)-LGS mit 5 Gleichungen und 4 Variablen entsteht.

Das LGS lautet (Eingabe der Matrix MK am CAS des TI-92)

MK =

t1	t2	t3	t4	Rs
–0.2	0.3	0	0	0
0.2	–0.5	0.3	0	0
0	0.2	–0.5	0.3	0
0	0	0.2	–0.3	0
1	1	1	1	1

Die stationäre Verteilung wird mit Hilfe des LGS bestimmt.

b) Anwendung des CAS-Befehls rref(MK), Lösungsvektor notieren:
rref(MK) =

t1	t2	t3	t4	Rs	
1	*0*	*0*	*0*	*0.4153*	*27/65*
0	*1*	*0*	*0*	*0.2769*	*18/65*
0	*0*	*1*	*0*	*0.1846*	*12/65*
0	*0*	*0*	*1*	*0.1231*	*8/65*
0	*0*	*0*	*0*	*0*	

T = [27/65, 18/65, 12/65, 8/65]

c) Im Mittel sind an einem Tag in 42 % der Fälle alle Aufträge bearbeitet, in 12 % der Fälle liegen 3 Aufträge vor. Immerhin lassen sich ca. 30 % der Aufträge nicht innerhalb eines Tages ausführen, weil 2 oder 3 Aufträge vorliegen. Die Entlassung von Personen würde den Wert noch vergrößern.

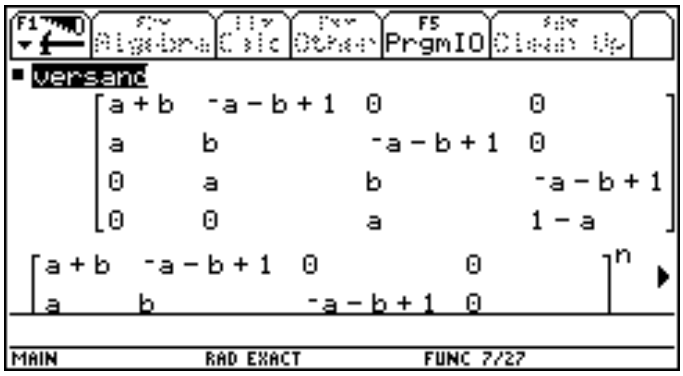
Bearbeitung von Aufgabe 1.4

S =

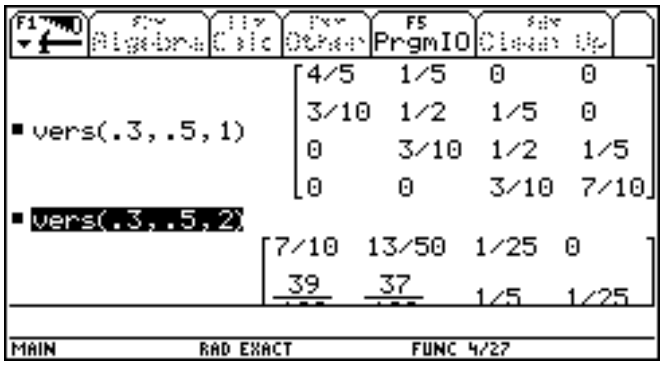
a+b	c	0	0
a	b	c	0
0	a	b	c
0	0	a	b+c

=

a+b	1–(a+b)	0	0
a	b	1–(a+b)	0
0	a	b	1–(a+b)
0	0	a	1–a

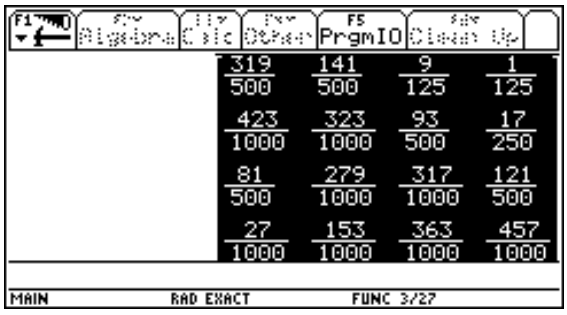


Die Matrix „versand" wird eingegeben. Danach wird der **Baustein [versand]^n → vers(a,b,n)** definiert (nur in Teilen sichtbar).

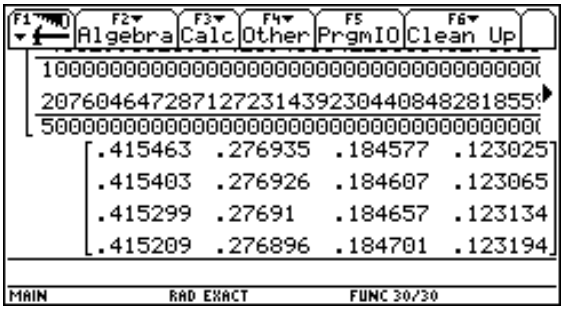


Untersuchungen mit Matrizenpotenzen

Der Kontrollaufruf vers(0.3, 0.5, 1) liefert gerade die oben in Zahlen gegebene Übergangsmatrix. – Mit dem Baustein ergeben sich nun diverse Varianten der obigen Aufgabenstellung, z. B. beliebige Belegungen der Parameter und schnelle Berechnungen der Auswirkungen mit Hilfe der Potenzen oder das Erkennen von Gesetzmäßigkeiten durch Aufrufe wie vers(a, b, 2), vers(a, b, 3) usw. Hierzu noch einige TI-Bilder:



vers(0.3, 0.5, 3)

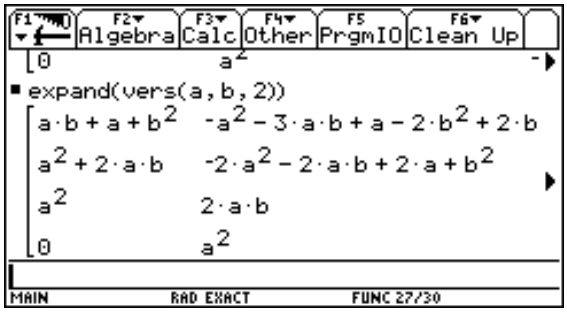


vers(0.3, 0.5, 50), 50-te Matrizenpotenz

Das langfristige Verhalten des Systems kann also auch mit den Potenzen der Übergangsmatrix „versand" untersucht werden. Die 50. Potenz zeigt deutlich eine typische Eigenschaft von ergodischen Markow-Ketten:

> Mit wachsendem Exponenten der Potenzen der Übergangsmatrix stimmen
> - die Matrixzeilen und
> - die Werte in jeder Spalte immer mehr überein.

Der Beweis dieses Satzes kann mit guten Schülern bewältigt werden, siehe *[Leh86b], S. 101-110]*.



Die ersten beiden Spalten der Matrix „versand^2".

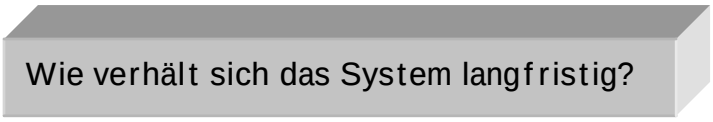
Entdecken von Gesetzmäßigkeiten?

Das Entdecken von Gesetzmäßigkeiten für die Folge einiger Elemente auf den gleichen Positionen der Matrizenpotenzen ist hier kaum möglich, in anderen Fällen geht das. Beweise für die Folgenterme können dann mit vollständiger Induktion erfolgen. Man kann in solchen Fällen die **Grenzmatrix** exakt bestimmen.

Phasen der Modellbildung beim Versandproblem

Hinter der Bearbeitung der obigen Abituraufgabe stehen einige Modellbildungsphasen in engerem Sinn, die im Folgenden verdeutlicht werden sollen. **Gleichzeitig wird angedeutet, wie die Einführung mathematischer Begriffe** *(kursiv geschrieben)* **in den Prozess eingebettet werden kann**. **Die im vorliegenden Fall vorgeführten Modellierungsvorgänge führen jeweils zu anderen mathematischen Modellen. Bei Markow-Ketten ergänzen sich diese in überzeugender Weise, weil die Modellbildungen zu gut vergleichbaren Ergebnissen führen.**

Die folgende Darstellung ist so aufbereitet, dass eine mögliche Abfolge von Modellbildungsphasen deutlich wird. Antrieb für die Modellierung ist – wie häufig auch bei anderen Anwendungen von Markow-Ketten – die Fragestellung:



Schritte der Modellbildung

Schritt 1:	Aufbau der Übergangsmatrix, Festlegung der Zustände, Veranschaulichung
Schritt 2:	Elemente der Übergangsmatrix finden, Veranschaulichung
Schritt 3:	Langfristige Entwicklung des Systems untersuchen, mehrstufige Übergangswahrscheinlichkeiten berechnen
Schritt 4:	Einsatz eines CAS, der Baustein „vers(a, b, n)"
Schritt 5a1:	Bildung von Potenzen mit Hilfe des Bausteins
Schritt 5a2:	Interpretation der Ergebnisse aus Schritt 5a1
Schritt 5b1:	Stationäre Verteilung suchen, Fixvektor ermitteln, Lösung eines LGS
Schritt 5b2:	Interpretation der Ergebnisse aus Schritt 5b1
Schritt 6:	Ggf. weitere Modellrechnungen und deren Interpretation
Schritt 7:	Simulation des Versandproblems

Modellbildung, Schritt 1
Festlegung der Zustände, Aufbau der Übergangsmatrix, Veranschaulichung

Für das System lassen sich ***Zustände*** festlegen, zwischen denen das System hin- und her pendelt. Im vorliegenden Fall sind die Anzahlen der noch unbearbeiteten Aufträge geeignet. Die Daten für den Wechsel zwischen den einzelnen Zuständen lassen sich in sogenannten ***Übergangsmatrizen*** zusammenfassen.

1) Wir gehen davon aus, dass 4 Zustände vorhanden sind:

Z0	es liegt kein Auftrag vor	Z1	es liegt ein Auftrag vor
Z2	es liegen zwei Aufträge vor	Z3	es liegen drei Aufträge vor

Damit hat die Übergangsmatrix die Form
(# Übergang von Z3 nach Z2)

von\nach	Z0	Z1	Z2	Z3
Z0				
Z1				
Z2				
Z3			#	

Wenn anfangs ein Auftrag vorliegt, kann man die ***Anfangsverteilung*** als ***Wahrscheinlichkeitsvektor*** in der Form (0 1 0 0) schreiben.
Z0 Z1 Z2 Z3

In der Regel ist es nützlich, sich die Vorgänge zu veranschaulichen, um so Modellvorstellungen leichter entwickeln zu können. In Anlehnung an die Idee der Baumdiagramme kann das hier z. B. folgendermaßen geschehen:

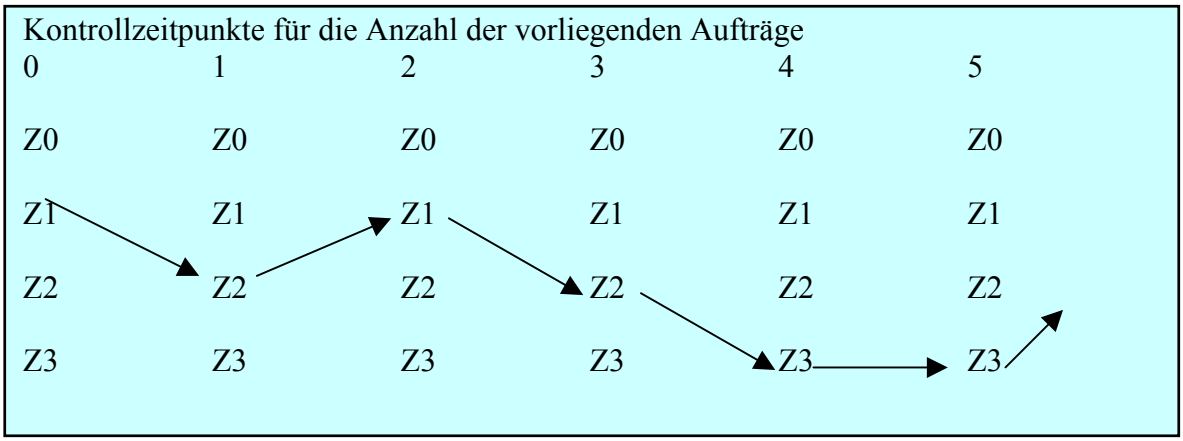


Abb.3.3.3-b: Einer von vielen möglichen Wegen durch die Warteschlange ist die ***Zustandskette*** (Z1, Z2, Z1, Z2, Z3, Z3, ...)

Zwanglos ergeben sich hieraus schon einige Fragen,

- wie z.B. die nach der Wahrscheinlichkeit für die dargestellte Kette bis zu einem gewissen Kontrollzeitpunkt oder gar
- die Frage nach einer Formel dafür.

Das sind Fragen, die die Schüler selbst gestellt haben.

Modellbildung, Schritt 2
Die Elemente der Übergangsmatrix, Veranschaulichung

Wie in der Abituraufgabe vorgeführt, sind hier für die einzelnen Elemente Überlegungen gemäß der vorliegenden Situation durchzuführen. Diese führen dann zu den Übergangswahrscheinlichkeiten zwischen den einzelnen Zuständen.
Die Übergangsmatrix weist interessante Regelmäßigkeiten auf, die sich bei weiteren vorliegenden Aufträgen sinngemäß fortsetzen würden! Zur Veranschaulichung kann man den ***Zustandsgraphen*** (Abb. 3.3.3-a) verwenden.

Modellbildung, Schritt 3
Wie entwickelt sich das System langfristig?

Von einem Kontrollzeitpunkt zum nächsten Kontrollzeitpunkt gilt immer wieder die gleiche Übergangsmatrix S – übrigens eine gewagte Modellannahme! Abbildung 3.3.3-c zeigt, dass zahlreiche 2-stufige Übergänge möglich sind, nämlich von

Z0 über (Z0,Z1,Z2,Z3) zu (Z0,Z1,Z2,Z3)	16 Übergänge
Z1	16 Übergänge
Z2	16 Übergänge
Z3	16 Übergänge
	das sind insgesamt 4^3 = 64 Übergänge.

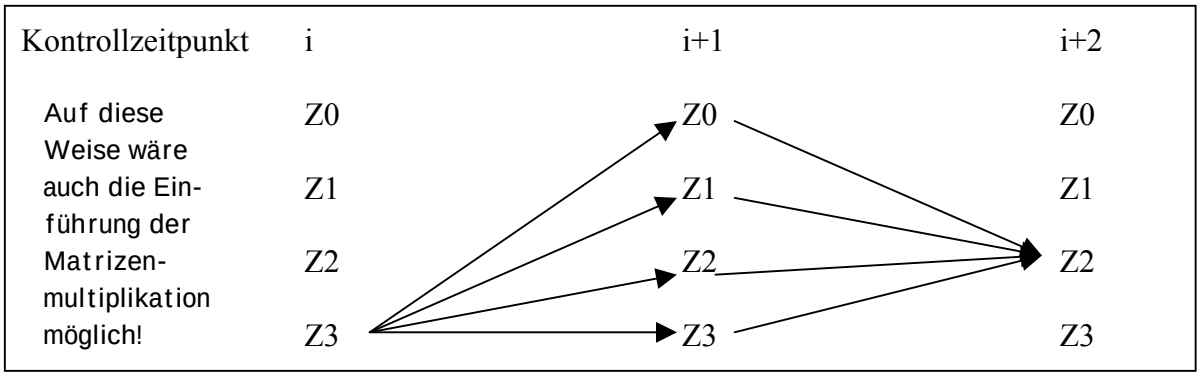


Abb.3.3.3-c: Denkbare Wege von Z3 nach Z2 mit einem Zwischen-Kontrollpunkt (Hinweis: Einige Wahrscheinlichkeiten können auch den Wert 0 haben, werden hier jedoch der Vollständigkeit halber eingezeichnet)

In der Abbildung wird der 2-stufige Übergang von Z3 nach Z2 zu beliebigen Kontrollzeitpunkten i, i+1, i+2 dargestellt. Der Übergang kann über jeden der 4 Zustände als Zwischenzustand erfolgen. Verfolgen wir die einzelnen Wege von Z3 nach Z2, so haben wir folgende Wahrscheinlichkeiten zu bilden:

P(Z3 nach Z0) * P(Z0 nach Z2) =	0	*	0	Hier kann ggf. die Einführung des Skalarprodukts erfolgen.
P(Z3 nach Z1) * P(Z1 nach Z2) =	0	*	0.2	
P(Z3 nach Z2) * P(Z2 nach Z2) =	0.3	*	0.5	
P(Z3 nach Z3) * P(Z3 nach Z2) =	0.7	*	0.3	

Addition aller Ergebnisse: 4.Zeile von S *„mal“* 3.Spalte von S (Skalarprodukt)

Durch Addition aller Ergebnisse erhält man die ***zweistufige Übergangswahrscheinlichkeit*** P(Z3 nach Z2, 2 Stufen) = 0 + 0 + 0.15 + 0.21 = 0.36. In der Sprache der linearen Algebra: Wir haben das ***Skalarprodukt*** gebildet aus dem Vektor in der 4. Zeile von S mit dem Vektor in der 3.Spalte von S. Bildet man auf diese Weise alle möglichen 16 Skalarprodukte („Zeile * Spalte“), so erhält man die vollständige zweistufige Übergangsmatrix als ***Matrizenprodukt*** der Übergangsmatrix S mit sich selbst: ***Zweistufige Übergangsmatrix*** $S*S = S^2$.
Auf diese Weise kommen ***Matrizenpotenzen*** ins Spiel.

Modellbildung, Schritt 4
Der Einsatz eines Tools – Computeralgebrasystem (CAS)

Für die weitere Bearbeitung wird als Hilfsmittel das CAS des Taschencomputers TI-92 (oder ein anderes CAS) benutzt. Die Übergangsmatrix S wird eingegeben (mit Zahlenwerten oder auch schon allgemein), anschließend wird der oben schon benutzte ***Baustein*** unter dem Namen „vers(a, b, n)” eingegeben.

Modellbildung, Schritt 5a1 – Bildung der Potenzen der Übergangsmatrix

Für diesen Schritt wird auf die obige Darstellung der Abituraufgabe verwiesen.

Modellbildung, Schritt 5a2 – Interpretation von Ergebnissen

Bei der zehnstufigen Übergangsmatrix – das System läuft also jetzt 10 Kontrollpunkte lang – fällt unter Beachtung der hier ausgegebenen 6 Nachkommastellen auf:

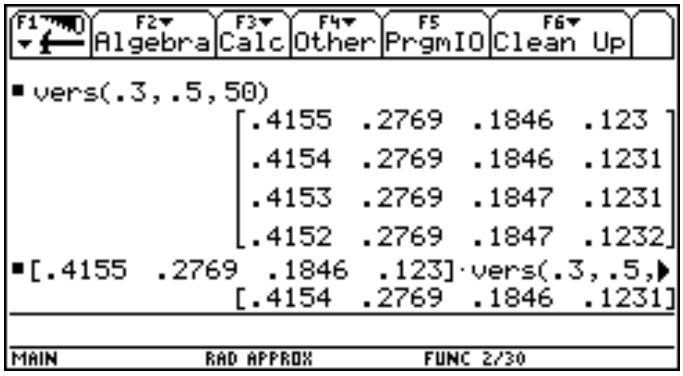
a) Alle Elemente einer Spalte sind gleich.
b) Die Zeilen der Matrix stimmen überein.
c) Alle Elemente liegen im Intervall [0,1].
d) Die Summe der Elemente jeder Zeile ist stets gleich 1.

Die Eigenschaften c) und d) sind die bekannten Eigenschaften von stochastischen Matrizen. Es gilt der leicht (auch allgemein) nachrechenbare Satz:

> Das Produkt zweier stochastischer Matrizen (also auch das Produkt einer stochastischen Matrix mit sich selbst) ist wieder eine stochastische Matrix.

e) Bei den folgenden Potenzen ändern sich die Werte (bei 6 Nachkommastellen!) nicht mehr. Es gilt anscheinend $S^{51} \approx S^{50}$, allgemeiner $S^{n+1} \approx S^n$ für große n.

Wir nehmen eine Zeile von S^{50} und rechnen mit dem TI-92 nach:



- So lässt sich vermuten, dass sich die Matrixpotenzen S^n immer mehr einer festen Matrix G nähern.

- $\lim_{n\to\infty} S^n = G$.

- Wenn es eine Grenzmatrix G gibt, so muss gelten G*S=G.

Bezeichnen wir die (untereinander gleichen) Zeilenvektoren von G mit g, so gilt damit auch:
- Es gibt einen Zeilenvektor g (Grenzverteilung) mit der Eigenschaft g*S = g.

Die Anfangsverteilung

Bisher wurde bei den Überlegungen an keiner Stelle der Anfangsvektor (0 1 0 0) benutzt – es liegt anfangs ein Auftrag vor!

Bilden wir nun (0 1 0 0)*G, so ergibt sich die 2. Zeile von G. Nimmt man als Anfangsvektor z. B. (0 0 0 1), so liefert (0 0 0 1)*G die 4. Zeile von G. Da aber alle Zeilen von G gleich sind, kann festgehalten werden:

- Das Verhalten des Systems ist offenbar unabhängig vom Anfangsvektor.

Modellbildung, Schritt 5b

Ein zweiter Lösungsweg: Stationäre Verteilung – Fixvektor – Lösung eines LGS

Diese Überlegungen eröffnen nun eine weitere Möglichkeit für die Untersuchung des langfristigen Systemverhaltens. Der Vektor g spielt offenbar die Rolle eines ***Fixvektors***. Wenn ein Vektor f mit der Eigenschaft f*S = f tatsächlich existiert, dann kann man ihn aus einem linearen Gleichungsystem errechnen:

- f * S = f, ausführlich [f0, f1, f2, f3] * S = [f0, f1, f2, f3]
  mit der Bedingung f0 + f1 + f2 + f3 = 1.

Hinweis: Im Unterrichsablauf wird diese Zusatzbedingung meistens vergessen. Dann ist das LGS zunächst nicht eindeutig lösbar und man kommt im Nachhinein auf diese Bedingung.
Wir können nun unter Beachtung der Matrixtypen z. B. folgendermaßen rechnen:

- $f_{(1,4)} * S_{(4,4)} = f_{(1,4)}$ mit f = [f0, f1, f2, f3] und f0+ f1+ f2+ f3 = 1

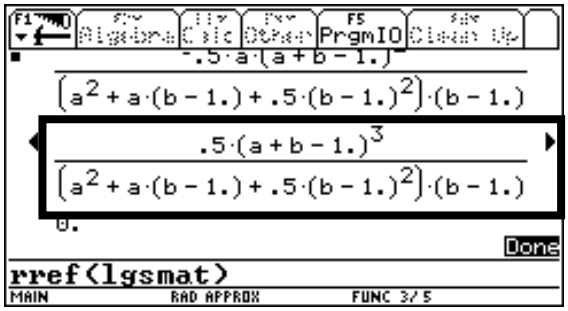
  $f_{(1,4)} * (S_{(4,4)} - E_{(4,4)}) = O_{(1,4)}$,
  wobei $E_{(4,4)}$ eine (4,4)-Einheitsmatrix und $O_{(1,4)}$ eine (1,4)-Nullmatrix ist.

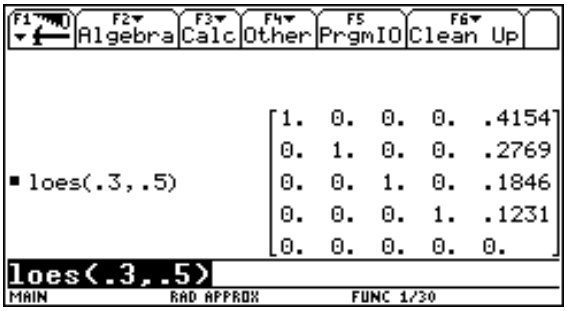
Nehmen wir nun noch die zusätzliche Gleichung f0 + f1 + f2 + f3 = 1 in die Matrixgleichung auf, so erhalten wir das lineare Gleichungssystem:

f0	f1	f2	f3	rechte Seite des LGS
(a+b)–1	a	0	0	0
1–(a+b)	b–1	a	0	0
0	1–(a+b)	b–1	a	0
0	0	1–(a+b)	1–a–1	0
1	1	1	1	1

= M

Dieses kann als erweiterte Koeffizientenmatrix M in das CAS des TI-92 eingeben werden und liefert mit **rref(M)** die allgemeine Lösung des LGS (recht umfangreiche Terme, die hier nicht vollständig abgedruckt werden). Damit ist die ***stationäre Verteilung*** der Markow-Kette berechnet. Für Sonderfälle (Nenner = 0) existiert diese nicht.





Zum Beispiel ergibt sich für f3 der oben dick umrahmte Term. Speichert man die allgemeine Lösung unter loes(a,b), so liefert der Aufruf loes(0.3, 0.5) die von oben bekannte stationäre Verteilung.

Simulation des Versandproblems

Wie bei vielen komplexen Problemen bietet sich auch bei dem vorliegenden Versandproblem eine Untersuchung durch Simulation an. Die Bezüge zur Informatik sind bei Simulationen besonders eng, werden doch in der Regel entsprechende Simulationsprogramme benötigt. Aber auch Computeralgebrasysteme können herangezogen werden.

Beim Versandproblem ist die Simulation nicht ganz einfach. Die Übergänge zwischen den Zuständen sind von den Übergangswahrscheinlichkeiten abhängig. Deren Werte sind also passend zu berücksichtigen. In *[Leh78], S. 227–242]* wird eine Formel für die Simulation von Markow-Ketten entwickelt. Sie lautet:

(*) $T = Int(R+1-p_0) + Int(R+1-\sum_{i=0}^{1} p_i) + ... + Int(R+1-\sum_{i=0}^{n-1} p_i)$ bzw. noch kompakter

(**) $T = \sum_{k=0}^{n} Int(R+1-\sum_{i=0}^{k} p_i)$. Die Formel errechnet den jeweiligen Folgezustand T.

Dabei ist R eine Zufallszahl aus dem Intervall [0, 1[, die für eine Berechnung fest bleibt. p_0, p_1, ..., p_{n-1} sind die Wahrscheinlichkeiten einer Zeile der Übergangsmatrix. Die Zustände sind Z0, Z1, ..., Z(n–1). Sie werden hier durch die ganzen Zahlen 0, 1, 2,.., n–1 ausgedrückt). *Int* steht für die Integerfunktion (Vorkommastelle). Also z. B. int(3.21) = 3.

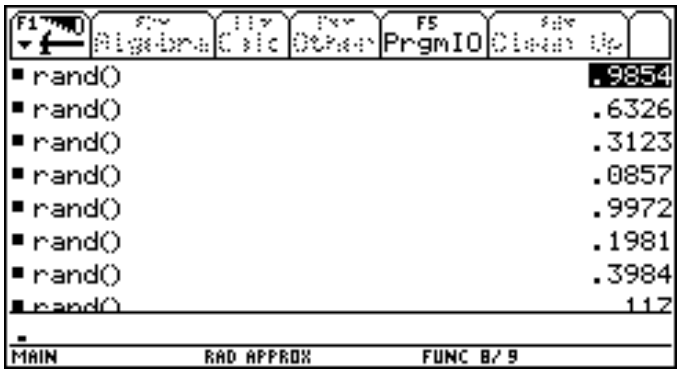
Für die Versandmatrix ergibt sich also gemäß (*):
T(Zeile Z0) = Int(R+1–0.8) + Int(R+1–(0.8+0.2) + Int(R+1–(0.8+0.2+0)) + Int(R+1–(0.8+0.2+0+0))
*T(Zeile Z0) = Int(R+0.2) + 3*Int(R),*
T(Zeile Z0) = Int(R+0.2); hier können nur die Zustände Z0 und Z1 erreicht werden, da sich je nach Wert von R nur die Werte 0 oder 1 ergeben können.

T(Zeile Z1) = Int(R+1–0.3) + Int(R+1–(0.3+0.5) + Int(R+1–(0.3+0.5+0.2)) + Int(R+1–(0.3+0.5+0.2+0))
*T(Zeile Z1) = Int(R+0.7) + Int(R+0.2) + 2*Int(R)*
T(Zeile Z1) = Int(R+0.7) + Int(R+0.2); hier können je nach Wert von R die Zustände Z0, Z1, Z2 erreicht werden.

T(Zeile Z2) = Int(R+1–0) + Int(R+1–(0+0.3) + Int(R+1–(0+0.3+0.5)) + Int(R+1–(0+0.3+0.5+0.2))
T(Zeile Z2) = Int(R+1) + Int(R+0.7) + Int(R+0.2) + Int(R)
T(Zeile Z2) = 1 + Int(R+0.7) + Int(R+0.2); erreichbare Zustände sind hier Z1, Z2, Z3.

T(Zeile Z3) = Int(R+1–0) + Int(R+1–0) + Int(R+1–(0+0+0.3)) + Int(R+1–(0+0+0.3+0.7))
*T(Zeile Z3) = 2*Int(R+1) + Int(R+0.7) + Int(R)*
T(Zeile Z3) = 2+ Int(R+0.7); erreichbare Zustände sind hier Z2, Z3.

Beispiel:
Wir starten beispielweise mit dem Zustand Z2 und erzeugen nacheinander folgende Zufallszahlen R:



Rechnen mit	Neuer Zustand
T(Zeile 2) = 3	Z3
T(Zeile 3) = 3	Z3
T(Zeile 3) = 2	Z2
T(Zeile 2) = 1	Z1

usw.

Mit dem im Folgenden kurz dargestellten älteren MS-DOS-Programm *MARKOW (1991)* lässt sich die Simulation schneller durchführen.

```
  M A R K O W - K E T T E N                    (c) Eberhard Lehmann 1.9.91
 ──────────────────────────────────────┬──────────────────────────────────────
  BEREITSTELLEN VON MATRIZEN, HILFEN   │ RECHNEN MIT STOCHASTISCHEN MATRIZEN
 ──────────────────────────────────────┼──────────────────────────────────────
   1 Matrix eingeben / laden           │  8 Transponieren
   2 Matrix ändern                     │  9 A*B Produkt
   3 Ausgabeformat ändern              │ 10 A=B Gleichheit prüfen
   4 Matrix löschen                    │ 11 Potenz
   5 Stochastische Matrix erzeugen     │ 12 Potenzen/Verteilungen von..bis
                                       │
   6 Kurzdokumentation lesen           │ 13 Eigenwerte für (2,2),(3,3)-Matrix
   7 Aufgaben/Notizen  lesen/schreiben │ 14 Eigenwerte, charakter. Gleichung
 ──────────────────────────────────────┼──────────────────────────────────────
  MARKOW-KETTEN, NICHT-ABSORBIEREND    │ MARKOW-KETTEN, ABSORBIEREND
 ──────────────────────────────────────┼──────────────────────────────────────
  17 Art der Kette feststellen        │ 21 Ordnen nach absorbier. Zuständen
  18 Simulation                       │ 22 Simulation
  19 Stationäre Verteilung            │ 23 Stationäre Verteilung
                                      │ 24 Grenzmatrix, Mittelwerte
 ──────────────────────────────────────┼──────────────────────────────────────
  Auswahl mit CURSOR AUF / AB         │ 25 Zum Hauptmenü        26 Ende
```

Und hier sind die Ergebnisse:

Anzahl der Übergänge zwischen den Zuständen (bei 10000 Schritten)

nach	1	2	3	4
von 1:	3358	852	0	0
von 2:	852	1472	538	0
von 3:	0	537	834	371
von 4:	0	0	370	816

Zustand	absolute -	relative Häufigkeit
1	4210	0.4210
2	2861	0.2861
3	1742	0.1742
4	1187	0.1187

Erwartungswert = 1.9906

Anzahl der Übergänge zwischen den Zuständen: (bei 20000 Schritten)

nach	1	2	3	4
von 1:	6508	1650	0	0
von 2:	1651	2758	1173	0
von 3:	0	1173	1850	742
von 4:	0	0	742	1753

Zustand	absolute -	relative Häufigkeit	
1	8159	0.4079	**0.4153**
2	5581	0.2791	**0.2769**
3	3765	0.1882	**0.1846**
4	2495	0.1247	**0.1230**

Erwartungswert = 2.0298

Die Übergangsmatrix war =

0.8000	0.2000	0.0000	0.0000
0.3000	0.5000	0.2000	0.0000
0.0000	0.3000	0.5000	0.2000
0.0000	0.0000	0.3000	0.7000

Zur Erinnerung: Die oben errechneten Ergebnisse.

Die Simulation liefert also (je nach Anspruch) recht ordentliche Ergebnisse.

Ein Blick in das Programmsystem MARKOW

Die informatischen Bezüge des Themas Markow-Ketten werden besonders deutlich bei der Programmierung. Das oben benutzte System MARKOW enthält beispielsweise neben vielen anderen Prozeduren die folgenden Anweisungen:

Ausgewählte Einblicke in mathematische Programme

```
(* MARKOW-KETTEN A *)
UNIT markow_u;
INTERFACE
uses     TXT91_U, bild91_u, MTR89D_4, MTR89P_4, MATIO_89, MTASK, DOS, CRT, LEH_SOFT,
         ino91_u,balk91_u, gauss_u;  { für stationäre Verteilung}

type intfeld20=array[1..20] of integer;
var quadratisch,stochastisch:boolean;
   absorbierend:          boolean;
   absorbanzahl:          integer;
   absorbfeld  :          intfeld20;
   absorbfeld_neu:        intfeld20;

{Hauptprozedur der Unit}
procedure markow_kette_mit_n_zustaenden;



REPEAT (* ZUSTAENDE BERECHNEN *)
   z:=zustand;
   jetzt:=zustand;
   zustand:=0;
   zufall:=random;

   for S:=1 to matz.s do
   zustand:=TRUNC(zufall+matt.wert[Z,S])+ zustand;

   zustand:= zustand +1;
   danach:= zustand;
   if ausgaben IN ['j','J'] then
   begin
     if simulationen mod 15 = 0 then
     begin
       writeln; write(simulationen:5,'  ');
     end;
     if (simulationen div 150 >=1) and (simulationen mod 150=0) then
     c:=readkey;
     write(zustand:2,'  ');
   end
   else
   begin
     gotoxy(75,25); write(simulationen:5);
   end;
   absh[zustand]:=absh[zustand ]+1;
   simulationen:=simulationen+1;
   matz.wert[jetzt,danach]:=matz.wert[jetzt,danach]+1;
 UNTIL SIMULATIONEN >= anzahl_simulationen;
```

In diesem Programmteil ist u. a. die häufige Benutzung von Modulen (Units) erkennbar. Diese Idee wurde inzwischen auf die Mathematik übertragen, siehe u. a. Kapitel 2.1.5-2.1.7.

Hier erfolgt die Anwendung der oben angegebenen Simulationsformel (*).

Zusammenfassung

Oben ist die langfristige Systementwicklung auf sehr unterschiedliche Arten mit sich gegenseitig bestätigenden Ergebnissen untersucht worden:

- über die Folge der Matrizenpotenzen der Übergangsmatrix S,
- über ein lineares Gleichungssystem mit 4 Variablen und 5 Gleichungen,
- durch Simulation.

Es dürfte deutlich geworden sein, dass Markow-Ketten ausgezeichnet geeignet sind, auch informatische Aspekte im Mathematikunterricht zu berücksichtigen.

3.3.4 Das Crap-Spiel – Markow-Kette und endlicher Automat

Das Crap-Spiel ist eine schönes Beispiel für die Verbindung von Mathematik und Informatik über die Idee des endlichen Automaten.

Mittels einer Unterrichtseinheit im Leistungskurs Stochastik (2001) über das Crap-Spiel wird nun gezeigt, wie sich dabei mathematische und informatische Inhalte mischen. Außerdem wird deutlich werden, wie man unterschiedliche Medien bei projektartiger Arbeit einsetzen kann. Damit wird ein weiterer informatischer Aspekt berücksichtigt. Ich beschränke mich im fachlichen Teil auf wenige Ansätze und Ergebnisse. Eine detaillierte Unterrichtsplanung mit Lösungen kann in *[Leh97], S. 49–62* nachgelesen werden.

(A) Die Spielregeln des Crap-Spiels

CRAP ist ein Würfelspiel. Man kann es alleine oder zu zweit (oder mit noch mehr Personen) spielen.

(1)
2 Würfel werden geworfen (oder auch ein Würfel zweimal). Dann wird die Augensumme gebildet. Wir nennen diese im 1.Wurf geworfene Summe S'.
a) Wenn man die Augensumme 7 oder 11 wirft, hat man sofort gewonnen!
b) Wenn man die Augensumme 2 oder 3 oder 12 wirft, hat man sofort verloren!
c) In allen anderen Fällen geht es weiter!

(2)
Wir würfeln erneut mit den beiden Würfeln und bilden wieder die Summe. Wir nennen sie S.
a) Wenn man die Summe S = 7 hat, hat man verloren.
b) Wenn man eine Summe S = S' hat (also wie im ersten Wurf), hat man gewonnen.
c) In allen anderen Fällen geht das Spiel weiter bei (2)
(Hinweis: S' ist also immer die Summe des ersten Wurfes)

Abb. 3.3.4-a: CRAP, Spielregeln

(B) Unterrichtsplanung – Ist das Spiel fair?

- Kern aller Bearbeitungen sind Beantwortungen der Frage: Ist das Spiel fair, d. h. sind die Wahrscheinlichkeiten für Gewinn und Verlust gleich?

Zur Gesamtplanung wird hier eine Planungsbild aus dem oben genannten Buch gezeigt (Abb. 3.3.4-b), aus dem dann die Einteilung der Schüler (in diesem Fall waren es 19) unter Beachtung verschiedener Formen des Medieneinsatzes und von Hilfsmitteln erwachsen ist. Die Gruppen sind auch unter dem Aspekt der Leistungsfähigkeit der Schüler eingerichtet worden, d. h. die Aufgabenstellungen sind unterschiedlich in Schwierigkeitsgrad und Aufwand.

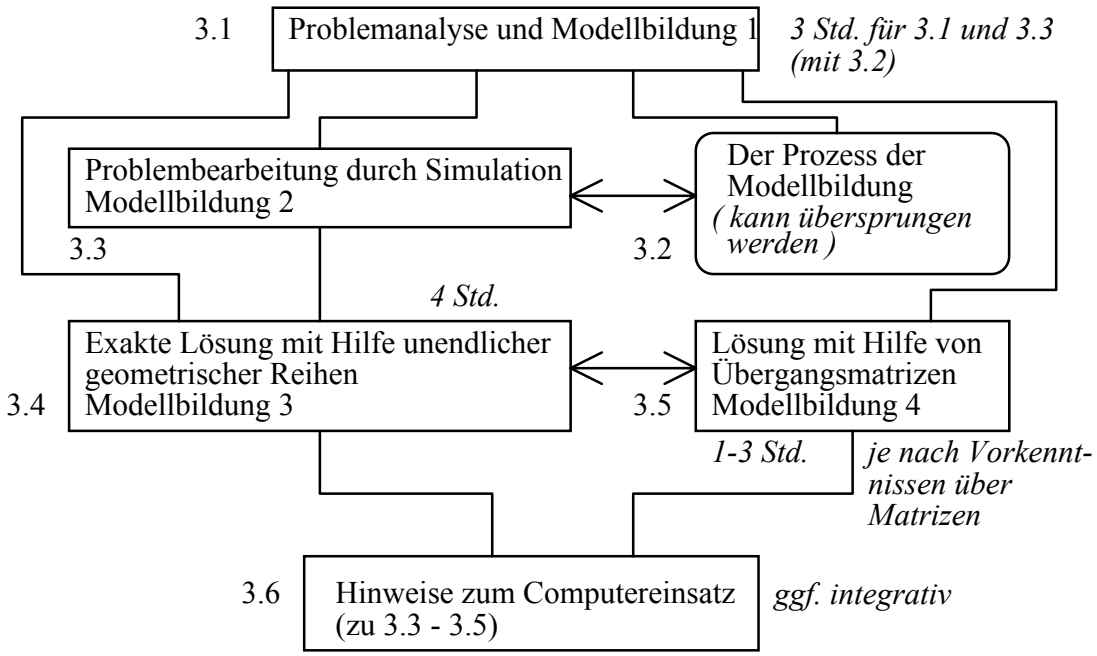


Abb. 3.3.4-b: Phasen der Modellbildung bei der Unterrichtsplanung „Crap-Spiel" (Kopie aus *[Leh97], S. 51*) – Vernetzung zwischen Mathematik und Informatik

C) Gruppeneinteilung – die Themen stellte der Lehrer

Am 19.3.01:

- Gruppe 1: Man berechne die Gewinn- und Verlustwahrscheinlichkeit des Crap-Spiels.
- Gruppe 2: Bearbeiten Sie das Problem durch Simulation.
- Gruppe 3: Bearbeiten Sie das Problem mit Hilfe von Markow-Ketten.
- Gruppe 4: Bearbeiten Sie die Ausführungen zum Crap-Spiel im Buch *Lehmann, E.: Wahrscheinlichkeitsrechnung, Volk und Wissen Verlag, 1997* mit der Problemlösung über unendliche geometrische Reihen.
- Gruppe 5: Schreiben Sie ein Struktogramm zu den vorliegenden Spielregeln und analysieren Sie die Arbeit des Programms CRAPAUTO.EXE
- Gruppe 6: Schreiben Sie ein Simulationsprogramm für das CRAP-Spiel in DELPHI.
- Gruppe 7: WWW-Recherche zum Thema CRAP-Spiel (Bemerkung für den Leser: Hierzu finden sich tatsächlich Ausführungen auf amerikanischen Webseiten)

Bezogen auf den Medieneinsatz ergab sich damit folgende Situation:

Gruppe	**Auftrag**	**Medien** (für Mathematik und Informatik), alle Gruppen dokumentieren mit Textverarbeitung WORD
1	Gewinn- und Verlustwahrscheinlichkeiten	beliebig
2	Simulation	Handsimulation, TI-92
3	Markow-Ketten (Matrizen)	TI-92 (notfalls Buch)
4	Unendliche geometrische Reihen	Buch, CAS (TI-92, DERIVE)
5	Struktogramm (Algorithmus), Programmanalyse	vorgelegtes PASCAL-Programm
6	Simulationsprogramm in DELPHI	PC mit Objekt-PASCAL
7	Informieren Sie sich im WWW zum Thema „CRAP".	WWW

Abb. 3.3.4-c: Planung für die Gruppenarbeit

Innerhalb der Gruppen kam es noch zu diversen Aufgabenverteilungen und dabei auch zur Dokumentation der Arbeit unter Benutzung der Textverarbeitung WORD.

(D) Einige Problemlösungen

Im Anschluss an die Erarbeitungen könnten die Ergebnisse zusammengefügt werden, z. B. zu einem „**Crap-Buch"**. Hier wurden die Ergebnisse teilweise vorgetragen, oder es wurde von den anderen Gruppen in die einzelnen Word-Dateien eingesehen. Die Programme wurden demonstriert. Insgesamt ergibt sich sehr beeindruckend, wie ein mathematisches Problem durch unterschiedliche Lösungsansätze bearbeitet werden und mit einer Vielfalt von Arbeitsmethoden angegangen werden kann.

(D1) Problembearbeitung durch Simulation

Auswahl aus den Ergebnissen von Gruppe 2:

Modellbildung – Protokoll führen und auswerten

Simulation

w Weiter würfeln, V Verloren, G Gewonnen

w V w G G w V w w w V G w w V G w w w G V w w w w G w w V w G w w w w w w
w w w w w w w w V V w w V G w w G w V w V G w V V G w w V w w w w V w V G
w w w w w V G V w w w w G w w w G G w w w w w w G w w w w V w w w w G w V
w w w w V w w w V w G w w G w w w w w w w V w w w w V w w G w w V w V G w
V w V

Ergebnisse:

Spielanzahl	=	50		Weiter gespielt	=	101
Gewinne	=	22	44 %	Wurfanzahl insgesamt	=	151
Verluste	=	28	56 %			

Modellbildung – Simulation mit Hilfe des Computers

Simulation 1 *Anzahl der Wiederholungen <=10000 ? 10000*
STATISTIK *Relative Häufigkeit*

Spielanzahl	=	*10000*		*10000*	
Gewinne	=	*4956*	*0.4956*	*4913*	*0.4913*
Verluste	=	*5044*	*0.5044*	*5087*	*0.5087*
Weiter gespielt	=	*23527*		*23893*	
Wurfanzahl insgesamt	=	*33527*		*33893*	

(D2) Exakte Lösung mit Hilfe von unendlichen geometrischen Reihen

Teillösungen von Gruppe 4:

Mathematisch-informatisches Modell

Modellbildung – Übergangsgraph

Wenn man eine exakte Problemlösung angehen will, ist ein neuer Schritt in der Modellbildung nötig. Dieser besteht zunächst darin, die Wahrscheinlichkeiten für die einzelnen Paarwürfe zu bestimmen. Man sieht leicht, dass gilt:

Augensumme S	2	3	4	5	6	7	8	9	10	11	12
Wahrscheinlichkeit	1/36	2/36	3/36	4/36	5/36	6/36	5/36	4/36	3/36	2/36	1/36

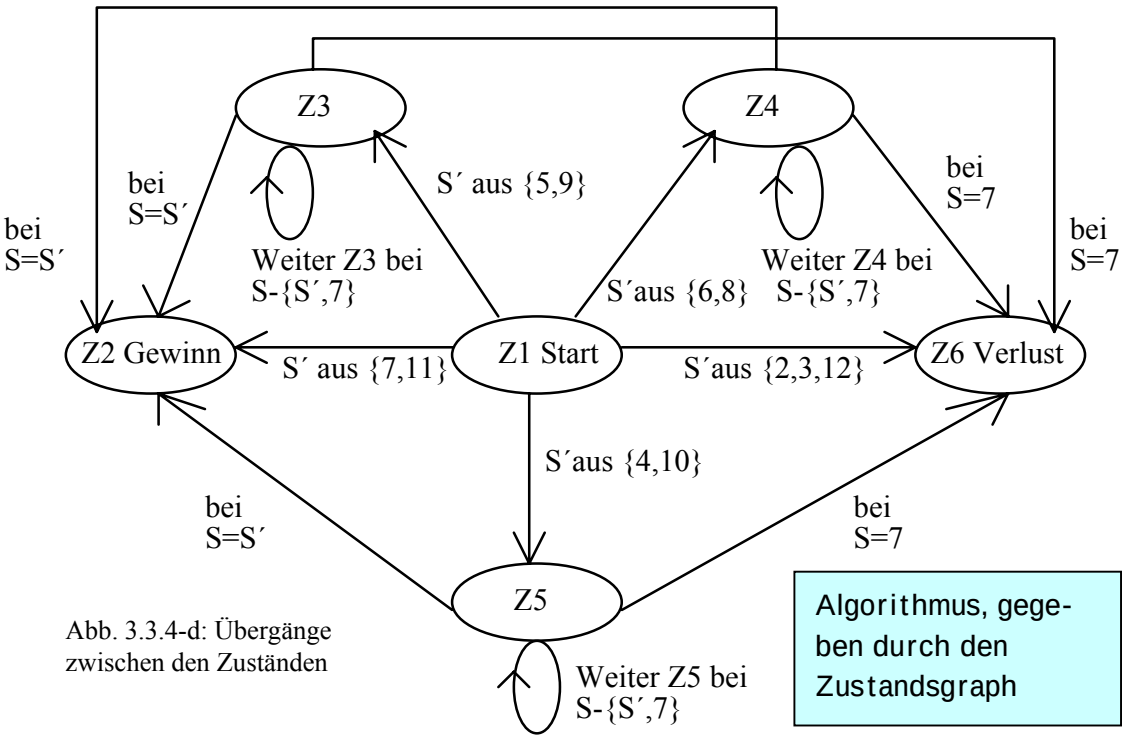


Abb. 3.3.4-d: Übergänge zwischen den Zuständen

Algorithmus, gegeben durch den Zustandsgraph

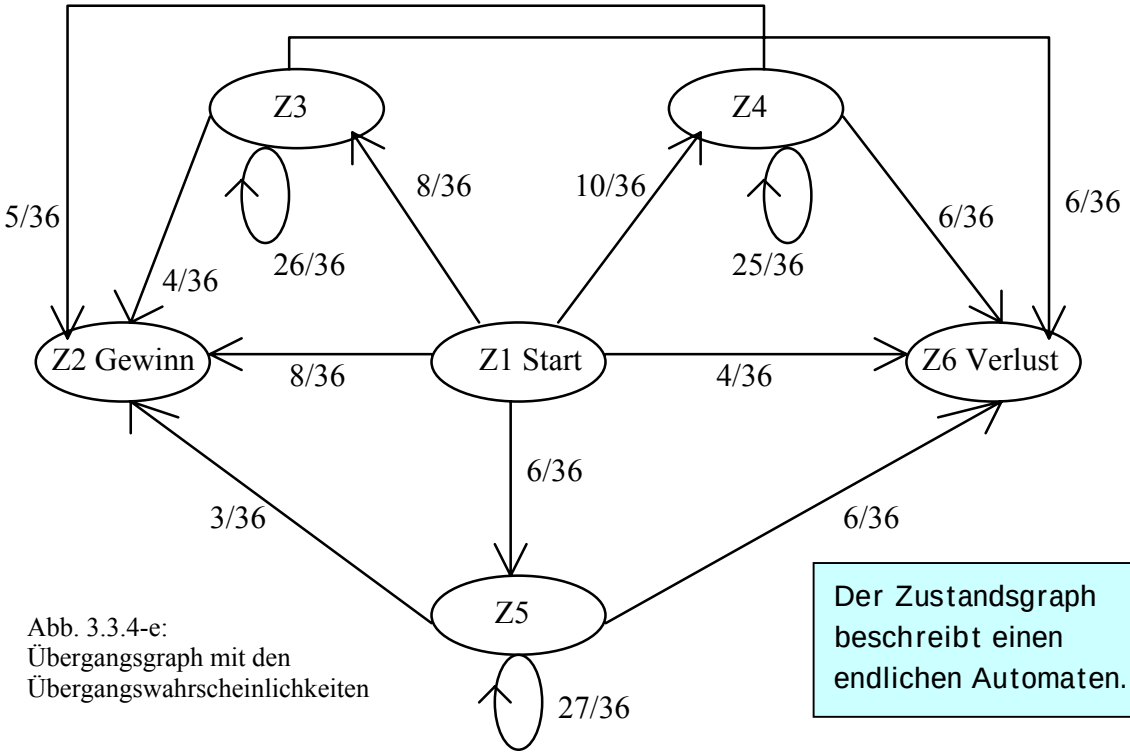


Abb. 3.3.4-e:
Übergangsgraph mit den Übergangswahrscheinlichkeiten

Der Zustandsgraph beschreibt einen endlichen Automaten.

Einsatz unendlicher geometrischer Reihen – Berechnung der Gewinnwahrscheinlichkeit

Übergang von Z5 nach Z2, siehe Abb. 3.3.4-e:

Unendliche geometrische Reihen

Anzahl der Schritte	1	2	3	4	...	n
Wahrscheinlichkeit	3/36	27/36 *3/36	$(27/36)^2$ *3/36	$(27/36)^3$ *3/36		$(27/36)^{n-1}$ *3/36

Das ist eine geometrische Reihe mit dem Quotienten q = 27/36. Als deren Summe ergibt sich

$s = \frac{3}{36} : \frac{1-\frac{27}{36}}{1} = \frac{1}{3}$, siehe unten ***.

Ausgehend von Z1 können wir nach Z2 auf verschiedenen Wegen gelangen:

Weg	Wahrscheinlichkeit		
Von Z1 direkt nach Z2	8/36		
Von Z1 über Z3 nach Z2	8/36 *	2/5	
Von Z1 über Z4 nach Z2	10/36 *	45/99	
Von Z1 über Z5 nach Z2	6/36 *	1/3	

Das sind die Summen der unendlichen geometrischen Reihen, siehe z. B. oben bei ***.

Diese Wahrscheinlichkeiten müssen nun noch aufsummiert werden (Additionsregel):
g(Z1→Z2) = 244/495 = 0.49292929... Gewinnwahrscheinlicheit des Crap-Spiels
g(Z1→Z6) = 1 - 244/495 = 251 /495 = 0.507171... Verlustwahrscheinlicheit des Crap-Spiels

Das Spiel ist nicht fair!

(D3) Lösung mit Hilfe von Übergangsmatrizen

Teilergebnisse von Gruppe 3

	Z1	Z2	Z3	Z4	Z5	Z6	
Z1	0	8/36	8/36	10/36	6/36	4/36	→ Zeilensumme = 1
Z2	0	1	0	0	0	0	→ Z2 ist Endzustand
Z3	0	4/36	26/36	0	0	6/36	
Z4	0	5/36	0	25/36	0	6/36	
Z5	0	3/36	0	0	27/26	6/36	
Z6	0	0	0	0	0	1	→ Z6 ist Endzustand

= Matrix A(1)
(eine Stufe)

↓

Übergänge nach Z1 sind nicht möglich, da Z1 Anfangszustand.
Für die Rechnungen könnte man auch auf Z1 verzichten.

Die längerfristige Entwicklung der Kette wird durch die Matrizenpotenzen ausgedrückt:

	Z1	Z2	Z3	Z4	Z5	Z6
Z1	0	0.4929	0	0	0	0.5071
Z2	0	1	0	0	0	0
Z3	0	0.4	0	0	0	0.6
Z4	0	0.4545	0	0	0	0.5455
Z5	0	0.3333	0	0	0	0.6667
Z6	0	0	0	0	0	1

.

Matrizenpotenzen

= Matrix A^{100}
(nach 100 Schritten)

Aus dieser Matrix kann man die Grenzmatrix mit schon recht großer Genauigkeit ablesen:

Bei Start in Zustand Z1 gewinnt man (Endzustand Z2) mit einer Wahrscheinlichkeit von 0.4929 und verliert (Endzustand Z6) mit einer Wahrscheinlichkeite von 0.5071. Das Spiel ist nicht fair!

(D4) Programmierung

Teilergebnisse der Gruppen 5 und 6

Simulationsprozedur

```
PROCEDURE simulation;
BEGIN
   FOR i:=1 TO anzahl DO
   BEGIN
      s:=RANDOM(6)+1+RANDOM(6)+1;
      IF s IN [7,11]          THEN  gewonnen:=gewonnen+1;
      IF s IN [2,3,12]        THEN  verloren:=verloren+1;
      IF s IN [4,5,6,8,9,10]  THEN  weiter:=weiter+1;
      t:=s;
      REPEAT
         s:=RANDOM(6)+1+RANDOM(6)+1;
         IF s=7 THEN verloren:=verloren+1;
         IF s=t THEN gewonnen:=gewonnen+1;
         IF s IN [2..12]-[7,t] THEN weiter:=weiter+1;
      UNTIL ( (s=7) OR (s=t) );
   END;
END;
```

Das DELPHI-Programm von Gruppe 6

```
unit Unit1;
interface

uses
  Windows, Messages, SysUtils, Classes,
  Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Label1: TLabel;
    Button2: TButton;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private-Deklarationen }
  public
    { Public-Deklarationen }
  end;

var
  Form1: TForm1;
  i,j,x,x1,g,v,g1,v1,g2,v2,anz: integer;

implementation

{$R *.DFM}

procedure TForm1.Button2Click
(Sender: TObject);
begin
 close;
end;
```

DELPHI-Programm

```
procedure TForm1.Button1Click
(Sender: TObject);
begin
 randomize;
 i:=random(6)+1;
 j:=random(6)+1;
 x:=i+j;
 anz:=anz+1;
 if x1 = 0 then begin
  x1:=x;
  label1.caption:='S`= '+inttostr(x1);
  label2.caption:='S = ';
  label3.caption:='';
  if (x1=7)or(x1=11)then begin
   label3.caption:='Winner';
   g2:=g2+anz;
   anz:=0;
   g:=g+1;
   x1:=0;
  end
  else if (x1=2)or(x1=3)or(x1=12)then begin
   label3.caption:='Looooooooooser';
   v2:=v2+anz;
   anz:=0;
   v:=v+1;
   x1:=0;
  end;
 end
 else begin
  label2.caption:= label2.caption + inttostr(x) + ', ';
  if x = x1 then begin
   label3.caption:='Winner';
   g2:=g2+anz;
   anz:=0;
   g:=g+1;
   x1:=0;
  end
  else if x=7 then begin
   label3.caption:='Looooooooser';
   v2:=v2+anz;
   anz:=0;
   v:=v+1;
   x1:=0;
  end;
 end;
 label6.caption:=inttostr(g);
 label7.caption:=inttostr(v);
 if g>0 then label10.caption:=floattostr((g2/g));
 if v>0 then label11.caption:=floattostr((v2/v));
 if (v>0) or (g>0) then
 label13.caption:=floattostr((v2+g2)/(v+g));
end;
end.
```

Zusammenfassung: Die Vielfalt der Lösungsmöglichkeiten zeigt die besondere Eignung des Themas „Crap-Spiel” für den Unterricht - auf Grund des unterschiedlichen Schwierigkeitsgrads können einige Lösungen auch schon in der Sekundarstufe 1 gefunden werden.

3.4. Ideen für weitere mathematisch-informatische Themen

3.4.1 Ausgewählte mathematische Funktionen der Informatik unter mathematisch-informatischen Aspekten

In der Informatik sind einige Funktionsarten von besonderer Bedeutung. Behandelt man derartige Funktionen im Mathematikunterricht, so liefert die Informatik das Anwendungsgebiet (das man dann im Unterricht auch zumindest kurz behandeln sollte), und es ergibt sich fächerübergreifender Unterricht.

Im Folgenden sollen einige derartige Funktionen genannt oder auch veranschaulicht werden, und es soll kurz auf Anwendungsbereiche hingewiesen werden.

Hinweise zur unterrichtlichen Verwendung der Themen

(A) Modulo-Funktion	(B) Effizienz von Algorithmen	(C) Zufallszahlen – Glücksräder
Befehle wie mod(19,12) können schon in der Unterstufe u. a. bei der Teilbarkeitslehre eingesetzt werden.	Einsetzbar in Klasse 11 oder in Analysiskursen bei Untersuchung von Funktionseigenschaften, insbesondere, wenn Informatikschüler im Kurs sind.	Dieses Thema kann sehr flexibel eingesetzt werden – im Rahmen von Stochastikunterricht beiden Sekundarstufen, aber auch in anderen Einzelstunden.
In der Sekundarstufe 1 sind mod(x,4) und Abwandlungen davon Beispiele für stückweise definierte Funktionen mit Anwendungsbezug, etwa bei Codierungsproblemen.		Siehe auch Kapitel 3.4.4 (Simulationen)
In der Sekundarstufe 2 können die Anwendungsmöglichkeiten von „modulo" in der Informatik zur Sprache kommen (in Klasse 11–13). Das Thema ist für Informatikkurse von besonderem Interesse.		Das Thema wird auch für Informatikkurse empfohlen.

(A) Modulo-Funktion

- Zum Beispiel ist mod(32, 27) = 5, denn bekanntlich gilt 32:27 = 1*27 + 5 (bzw. 32:27 = 1 Rest 5, Division mit Rest). 2. Beispiel: mod(4, 9) = 4.
- Auch Derive oder das CAS des TI-92 verstehen Anweisungen wie mod(32, 27). Damit ergeben sich diverse Möglichkeiten für eine mathematisch-informatische Unterrichtseinheit über die Modulo-Funktion und ihre Anwendungen.
- Anstelle von mod(a, b) schreibt man auch a mod b.

Es kann allgemein definiert werden: Für alle $a, b \in \mathbb{N}$ gilt:

Falls a < b gilt a mod b = a,

falls a > b gilt a mod b = r, wobei a = p*b + r und $0 \le r < b \text{ mit } r, p \in \mathbb{N}$,

falls a = b gilt a mod b = 0, also r = 0.

Anwendungsbereiche sind u. a.

- Kryptologie, z. B. RSA-Algorithmus,
- Verschlüsselung von Texten mit Matrizen,
- Programmierung von Zeilenumbrüchen bei Datenausgaben,
- Programmierung von Kongruenz-Zufallsgeneratoren (siehe Kapitel 3.4.4).

Benutzung eines Demonstrationsprogramms (Subtraktionsalgorithmus)

Berechnung von mod(a,b): a = 123 b = 7

116 109 102 95 88 81 74 67 60
53 46 39 32 25 18 11 4

mod(123, 7) = 4 bzw. 123: 7 = 17*7 *Rest 4*

Weiter (j, n) = j

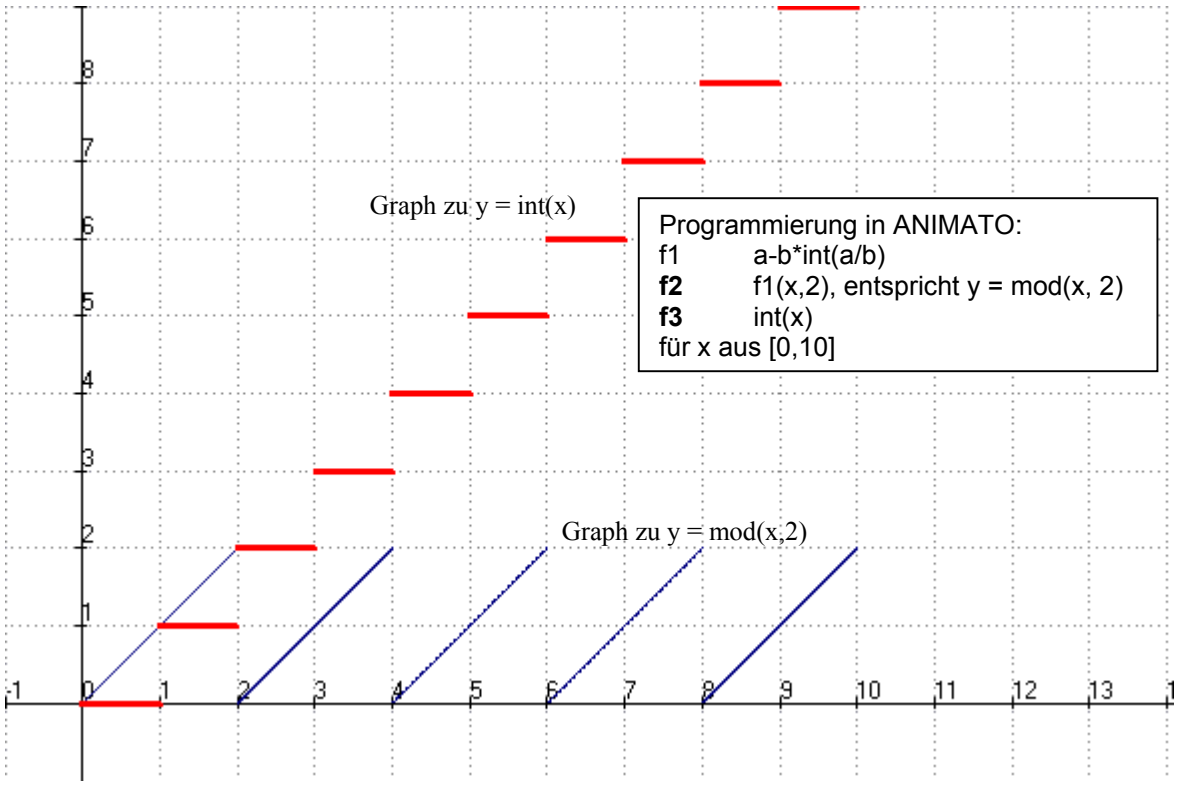


Abb.3.4.1-a : Die Graphen der Funktionen y = mod(x,2) und y = int(x). y = int(x) wird ebenfalls häufig benutzt, der Graph dient hier zum Vergleich mit der Modulo-Funktion
(int(x) liefert die Vorkommastelle von x, Beispiele: int(3.1416) = 3 und int(7) = 7)

Über Unterricht mit der Modulo-Funktion in der Sekundarstufe 1 kann man nachlesen in
[Lehmann, E.: Das Projekt Modulo, Projektarbeit im Wahlpflichtfach Mathematik, Klasse 10, in MU, Der Mathematikunterricht, 1999, Heft 6, S. 32f., Friedrich-Verlag]

(B) Effizienz von Algorithmen

In der Informatik sind Untersuchungen zur Effizienz von Algorithmen von großer Bedeutung. Beispielsweise interessieren bei Sortierverfahren u. a. die Anzahlen der notwendigen Vergleiche je zweier Elemente und die Anzahlen der Bewegungen (Zuweisungen) von Elementen. Hierzu eine Beispiel aus dem bekannten Buch von *Wirth [N. Wirth: Algorithmen und Datenstrukturen, Teubner-Verlag 1975, S. 94]*. Bei der Analyse des Sortierens durch direktes Einfügen gibt *Wirth* folgende Werte an:

Vergleiche	**Bewegungen**
C(min, schon sortiert) = $n-1$	M(min) = $2(n-1)$
C(mittlere Anzahl) = $\frac{n^2+n-2}{4}$	
C(max, ungünstigster Fall) = $\frac{n^2+n}{2}-1$	M(max) = $\frac{n^2+3n-4}{2}$

Zur Bewertung der in der Tabelle angegebenen Ergebnisse ist eine Veranschaulichung überaus nützlich. In Abb. 3.4.1-b erkennt man u. a. den erheblichen Unterschied zwischen der minimalen und maximalen Anzahl der Vergleiche, abhängig von der Vorsortierung der gegebenen Sortierfolge.

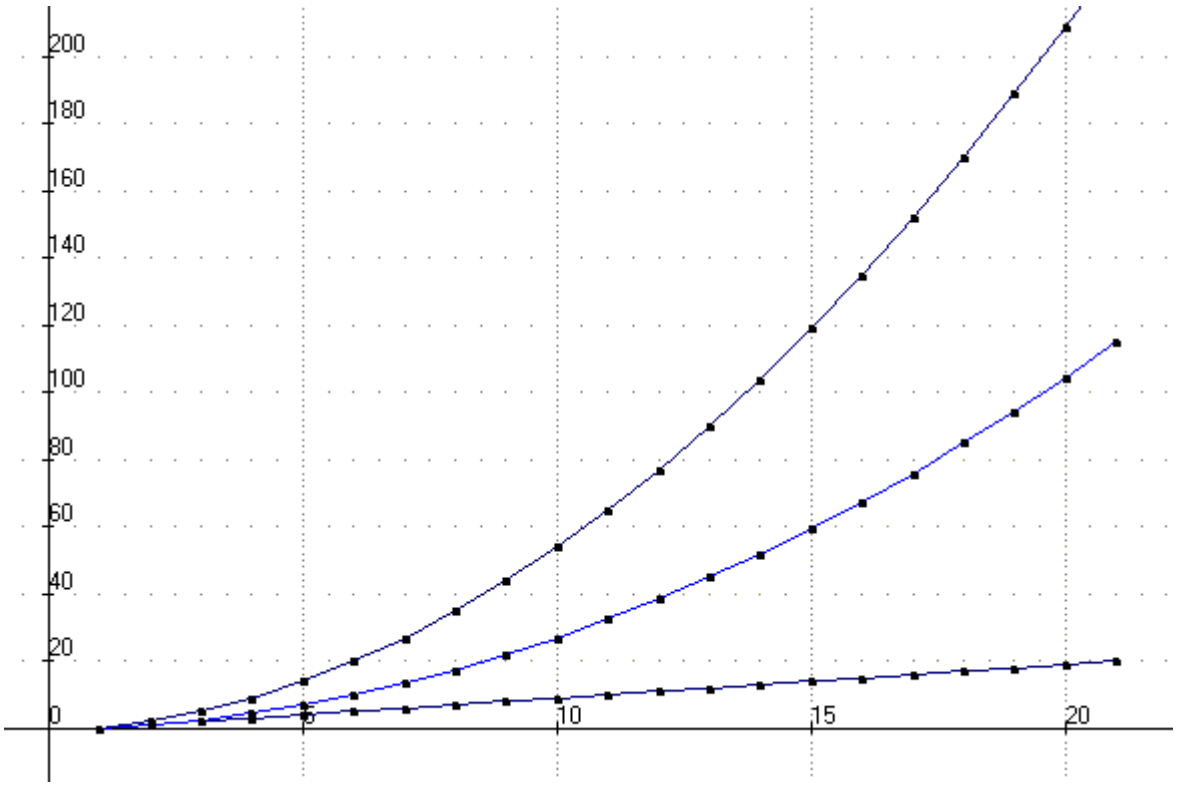


Abb. 3.4.1-b : Sortieren durch direktes Einfügen.

f1: Unten: Minimale Anzahl von Vergleichen bei n zu sortierenden Elementen
f3: Oben: Maximale Anzahl
f2 Mitte: Mittlere Anzahl

n	f1	f2	f3	n	f1	f2	f3
1	0	0	0	12	11	38.5	77
2	1	1	2	13	12	45	90
3	2	2.5	5	14	13	52	104
4	3	4.5	9	15	14	59.5	119
5	4	7	14	16	15	67.5	135
6	5	10	20	17	16	76	152
7	6	13.5	27	18	17	85	170
8	7	17.5	35	19	18	94.5	189
9	8	22	44	20	19	104.5	209
10	9	27	54	21	20	115	230
11	10	32.5	65				

(C) Zufallszahlen – Glücksräder

Dieses Thema ist auch eine eigene Unterrichtsreihe wert, siehe Kapitel 3.4.4. Hier soll insbesondere der funktionale Aspekt betont werden. In Kapitel 3.3.3 wurde bei der Simulation eines Versandproblems die Formel

$T = Int(R+1-p_0) + Int(R+1-\sum_{i=0}^{1} p_i) + ... + Int(R+1-\sum_{i=0}^{n-1} p_i)$ benutzt. Sie kann auch als Formel zur Simulation eines Glücksrads aufgefasst bzw. benutzt werden, dessen Sektoren die Wahrscheinlichkeiten $p_0, p_1, ..., p_{n-1}$ haben.

Betrachten wir etwa die Zeile 2 zur Simulation des Versandproblems:
T(Zeile Z2) = Int(R+1) + Int(R+0.7) + Int(R+0.2) + Int(R), erreichbare Zustände: Z1, Z2, Z3.
T(Zeile Z2) = 1+ Int(R+0.7) + Int(R+0.2) + 0.

Für $0 \le R < 0.3$ ist T = 1, für $0.3 \le R < 0.8$ ist T = 2, für $0.8 \le R < 1$ ist T = 3.

Die erreichten T-Werte 1, 2, 3 entsprechen den Zuständen Z1, Z2, Z3. In Abbildung 3.4.1-c erkennt man deutlich, dass der Übergang zu Zustand 2 am häufigsten auftritt. Das ist immer dann der Fall, wenn gilt $0.3 \le R < 0.8$ (Wahrscheinlichkeit 0.5).

Bei der Simulation von Problemen mit Hilfe von Zufallszahlen ist es von entscheidender Bedeutung, die richtige Transformation der durch die Random-Funktion gewonnenen Zufallszahlen aus dem Intervall [0, 1[zu finden.

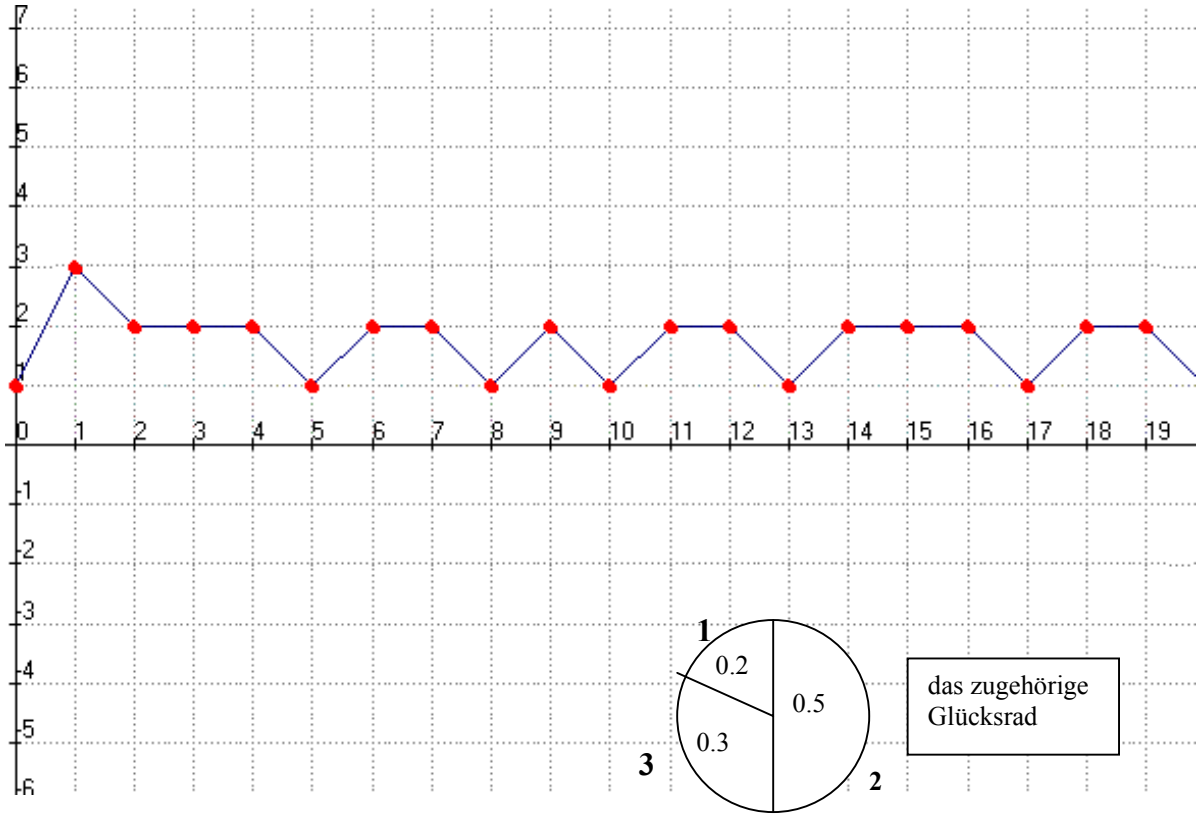


Abb.3.4.1-c: Visualisierung der Zustandsfolge mit Hilfe von ANIMATO

Int(R+1) + Int(R+0.7) + Int(R+0.2) + Int(R) = 1 + Int(R+0.7) + Int(R+0.2)

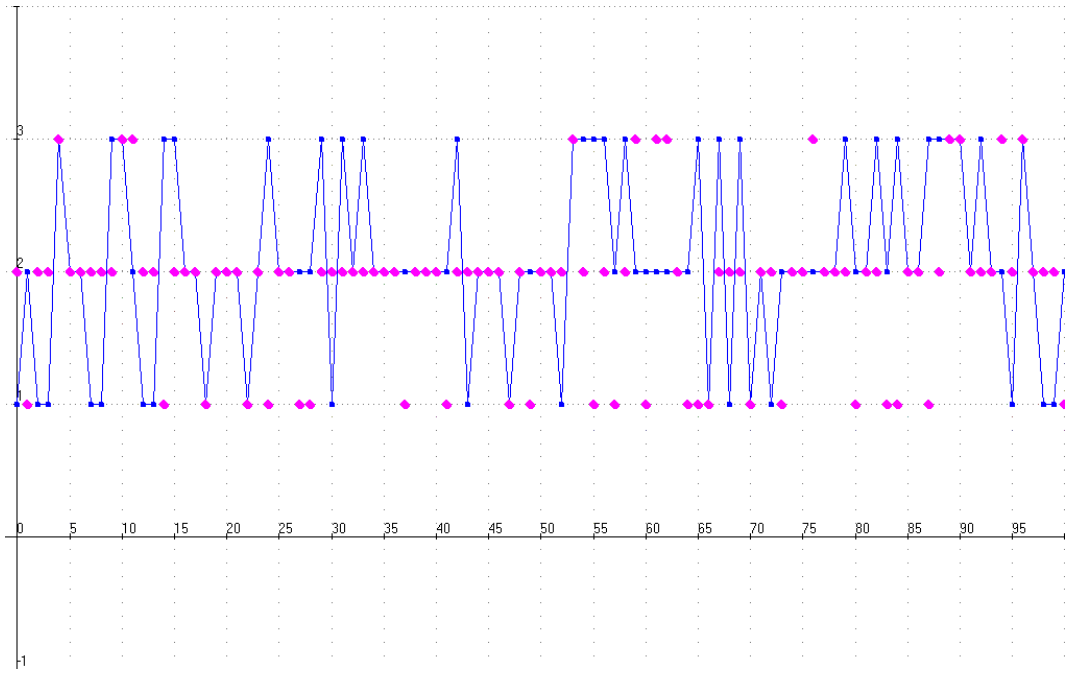


Abb. 3.4.1-d: Zwei Simulationen, jetzt je 100 Werte (zum Auszählen). Bei der ersten Simulation sind die Werte verbunden, die zweite Simulation wird durch die einzelnen dicker gezeichneten Punkte dargestellt.

3.4.2 Der (3a+1)-Algorithmus – ein Projekt für wenige Stunden

In Kapitel 1.3 wurden bereits einige interessante Algorithmen erwähnt, die geeignet sind, um mathematische und informatische Fragestellungen miteinander zu verknüpfen. Hier wird der dort ebenfalls genannte (3a+1)-Algorithmus (dort Beispiel 2) noch einmal aufgegriffen.

Hinweise zur unterrichtlichen Verwendung des Themas

A) Schon in der Unterstufe kann man diesen interessanten Algorithmus zum Üben des Zahlenrechnens mit positiven Zahlen benutzen. Hierzu kann z. B. ein passendes Spiel erfunden werden.	B) In der Sekundarstufe 1 (ab Klasse 7) ergeben sich Einsatzmöglichkeiten, wenn es um besondere Funktionsterme bzw. Folgen geht.	(C) In Klasse 11 werden in der Regel Folgen betrachtet. Der (3a+1)-Algorithmus lädt mittels leichter Abwandlungen geradezu zum Experimentieren ein. Die Programmierung in einem CAS ist leicht.	(D) Wer erzeugt die längste Folge? Diese Frage kann im Informatik-, aber auch im Mathematikunterricht zum „Halteproblem" führen.

Einleitung des Projekts durch 3 Arbeitsaufträge

Der (3a+1)-Algorithmus
Dieser Algorithmus, beginnend mit a, erzeugt Zahlenfolgen, die nach einer endlichen Anzahl von Schritten enden (oder auch nicht?). Dargestellt in einem Struktogramm, lautet der Algorithmus:

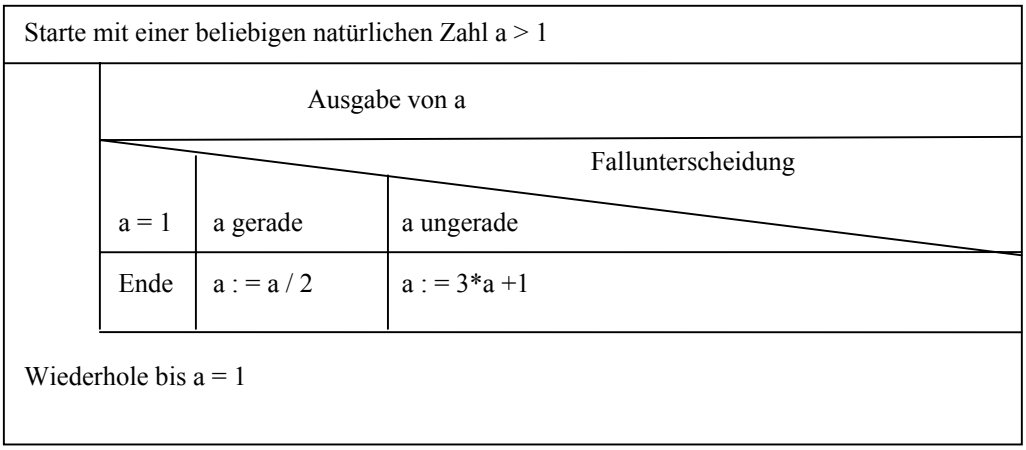


Abb. 3.4.2-a: Struktogramm zum (3a+1)-Algorithmus

Startet man beispielsweise mit a = 3, so ergibt sich die Zahlenfolge 3, 10, 5, 16, 8, 4, 2, 1, und damit endet der Algorithmus.

Arbeitsauftrag 1
Führe den Algorithmus für verschiedene a-Werte durch.

Arbeitsauftrag 2
In der folgenden Abbildung wird der Algorithmus für a =3 und für a = 7 im Koordinatensystem veranschaulicht.
a) Erläutere die Abbildungen 3.4.2-b und 3.4.2-c.
b) Für die Abbildungen wurde das Animationsprogramm ANIMATO benutzt. Schreibe ein Programm für ANIMATO oder für eine andere zur Verfügung stehenden Software.
c) Wiederhole mit deinem Programm die obigen Ergebnisse der Abbildungen 3.4.2-b und 3.4.2-c.

Arbeitsauftrag 3

In der Literatur findet man das folgende BASIC-Programm *[Engel, A.: Elementarmathematik vom algorithmischen Standpunkt, Klett-Studienbücher, Stuttgart 1977, S. 11]*. Erläutere das Programm und teste es. Alternative für einige Schüler: Schreibe ein Programm in der von dir verwendeten Programmiersprache.

```
10    INPUT A                          60    GOTO 20
20    PRINT A;                         70    A = 3A+1
30    IF A=1 THEN 90                   80    GOTO 20
40    IF  A/2 < > INT(A/2) THEN 70     90    END
50    A = A/2
```

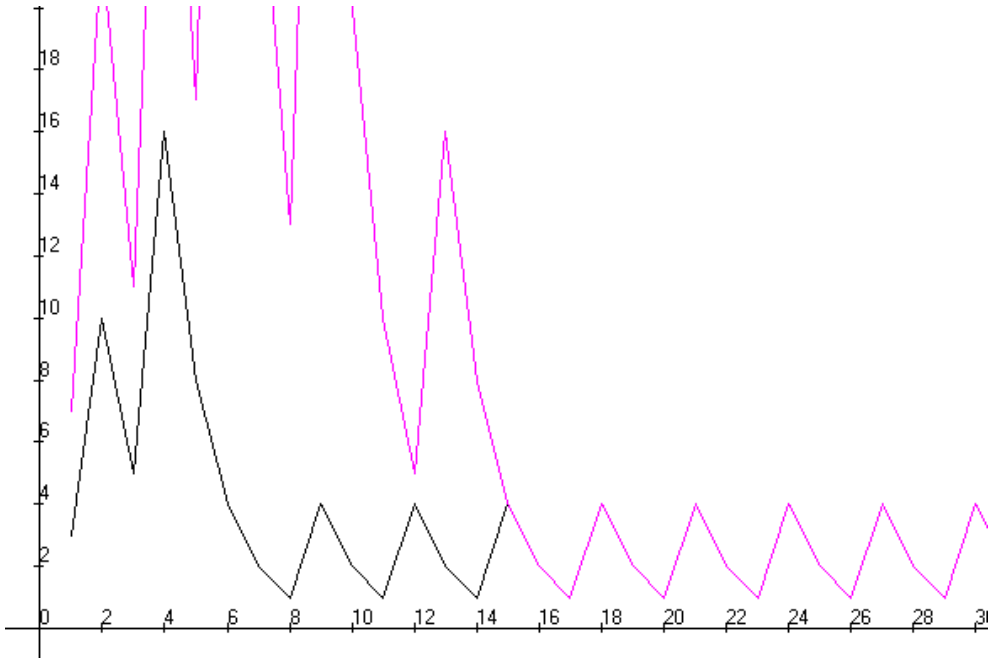


Abb. 3.4.2-b: Veranschaulichungen des (3a+1)-Algorithmus, Startwerte sind 3 und 7

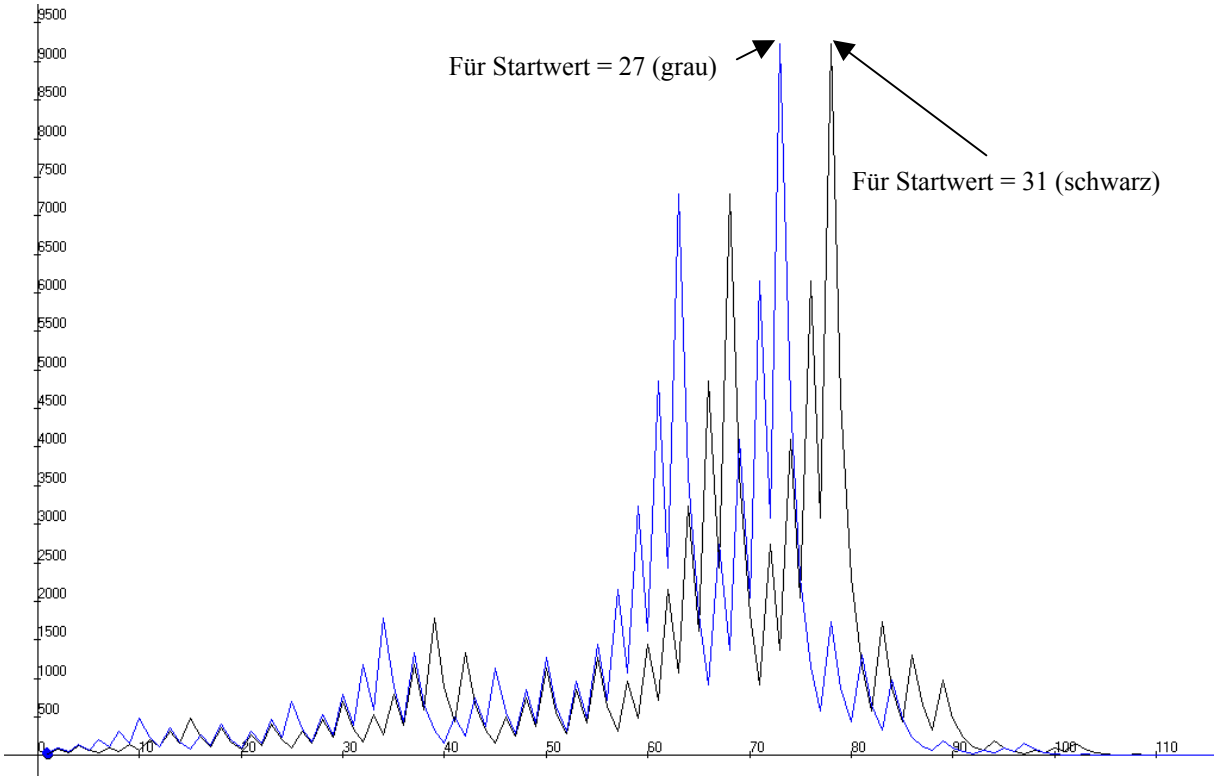


Abb. 3.4.2-c: Veranschaulichungen des (3a+1)-Algorithmus, Startwerte sind hier 27 bzw. 31

Projektfortsetzung

Die Fortsetzung des Projekts kann nun unter starker Berücksichtigung von Schülereigentätigkeit auf verschiedene Weise erfolgen. Hierfür bieten der (3a+1)-Algorithmus und geeignete Aufgabenvariationen diverse Angriffspunkte für experimentelle Arbeit, für Entdeckungen und Vermutungen. Hierzu sei noch einmal auf *[Eng77], S. 148* verwiesen. Naheliegend ist z.B. die Frage:

Wie ist das Verhalten der Folge bei anderen Startwerten. Notiere deine Feststellungen!

Einige Ansätze ergeben sich aus dem folgenden Text.

Lösungen – Bemerkungen zur Problemstellung – Der informatische Anteil

- In ANIMATO kann man so programmieren:

Programmzeilen	Erläuterungen
f1=3	Startwert der Folge
f2={n=1:f1:{(f2(n-1)/2=int(f2(n-1)/2)):f2(n-1)/2:3*f2(n-1)+1}}	Wenn n=1, dann f1 nehmen, andernfalls: wenn der Vorgänger ungerade, dann die Hälfte des Vorgängers nehmen, andernfalls: das (3-Fache des Vorgängers +1) nehmen.
f3=7 f4={n=1:f3:{(f4(n-1)/2=int(f4(n-1)/2)):f4(n-1)/2:3*f4(n-1)+1}}	Entsprechend wie oben.

Der ANIMATO-Algorithmus führt zu der folgenden Wertetafel

x,t	f2	f4	x,t	f2	f4	x,t	f2	f4
1	3	7	7	2	26	13	2	16
2	10	22	8	1	13	14	1	8
3	5	11	9	4	40	15	4	4
4	16	34	10	2	20	16	2	2
5	8	17	11	1	10	17	1	1
6	4	52	12	4	5	18	4	4
						usw. (Wiederholung der Teilfolge 4 - 2 - 1)		

- Lösung für den Taschencomputer (VOYAGE 200 oder für TI-92-Plus):

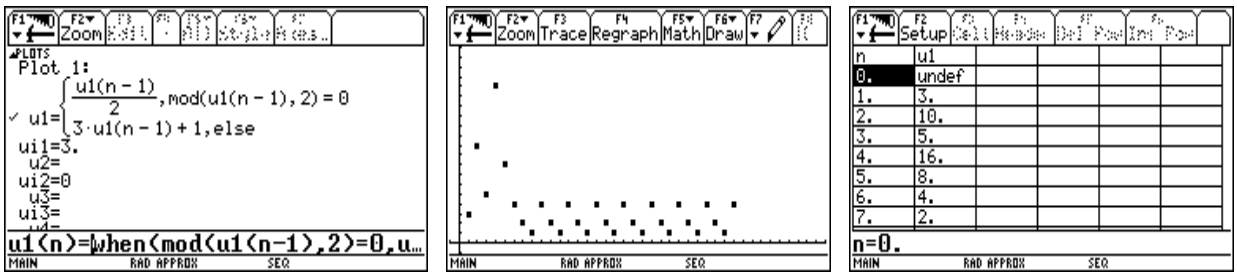


Abb. 3.4.2-d: when(mod(u1(n-1),2)=0, u1(n-1)/2, 3*u1(n-1)+1), Startwert ist hier gleich 3

Weitere Anregungen zum Experimentieren:

- Startwert – Anzahl der Schritte bis zum Abbruch (bei a=3 ist s=8, bei a=7 ist s=17, Startwert mitgezählt).
- Systematisches Vorgehen – den Startwert laufen lassen.
- Maximales Element der jeweiligen Folge?
- Arbeit mit anderen Folgen und ähnlichen Algorithmen.
- Gesetzmäßigkeiten für das Stoppen des Algorithmus.

Engel schreibt in seinem oben genannten Buch (1977) auf Seite 15 f.:
„Es ist nicht bekannt, ob die Folge für passende Startwerte nach ∞ divergiert oder in eine Schleife hineinläuft. Das Artificial Intelligence Laboratory am M.I.T. (Memo 239, 1972) hat alle A aus $-10^8 \leq A \leq 6*10^7$ untersucht. ... Die Folge lief stets in eine der folgenden fünf Schleifen hinein: ... D.J.Selfridge (Berkeley) hat nachgeprüft, daß der Algorithmus für alle $A \leq 2^{29}$ stoppt. Nach einer unbestätigten Meldung gilt dies auch für alle $A \leq 10^{40}$."

Die Frage des Stoppens des Algorithmus ist aus Informatiksicht besonders interessant in Verbindung mit dem sogenannten Halteproblem. Diese Problematik wurde bereits bei der Darstellung des Busy-Beaver-Problems behandelt, siehe Kapitel 3.3.

Hinweis: Mit den erwähnten Themen ließe sich auch eine mathematisch-informatische Unterrichtseinheit „Halteprobleme" entwickeln: a) der (3a+1)-Algorithmus, b) das Busy-Beaver-Problem usw.

Zusammenfassung:

Mit kleinen Programmen kann viel erreicht werden. Insbesondere ergibt sich die Möglichkeit experimentellen Arbeitens:

- Experimentieren mit verschiedenen Eingaben,
- Ausgaben auswerten,
- Vermutungen aufstellen und ggf. verifizieren.

Diese Arbeitsweise ist heute hoch aktuell und mit den jetzt zur Verfügung stehenden Mitteln leichter als damals zu realisieren. In der vorliegenden Arbeit finden sich zu dieser Methode mehrere Beiträge.

3.4.3 Mathematische Aspekte aus der Kryptologie

Fragestellungen aus der Kryptologie sind zur Zeit sowohl in der Informatik als auch in der Mathematik „hoch im Kurs". Die Informatik interessiert sich hierbei für die Algorithmen und Fragen der Datensicherheit, die Mathematik ist mehr an den mathematischen Hintergründen (Zahlentheorie) und den Algorithmen interessiert, findet aber in den informatischen Fragen ihr Anwendungsgebiet.
Angesichts der Fülle informativer Veröffentlichungen zur Kryptologie – auch für Umsetzungen in der Schule – wird hier auf eine ausführlichere Darstellung verzichtet und u. a. auf das Literaturverzeichnis, z. B. *[Ber94], [Sch94] und [Sin02]* verwiesen. Jedenfalls handelt es sich um ein auch für die Zielsetzungen dieser Arbeit sehr attraktives Thema. Außer den Literaturangaben, wird hier noch der zur Zeit (2002) in Berlin geltende *Rahmenplan für das Wahlpflichtfach Mathematik, Klasse 10*, *zur Unterrichtseinheit „Kryptologie"* abgedruckt. Dieser Plan vermittelt einen ersten Eindruck über unterrichtsrelevante Themenstellungen, die sich mathematisch, aber auch informatisch füllen lassen.

Wahlpflichtfach Mathematik Klasse 10

Rahmenplan

10.4 Kryptologie

Methoden der Chiffrierung und Dechiffrierung haben mit der Nutzung elektronischer Medien (z.B. Telefon über Satellit, elektronischer Geldtransfer, Versenden von Nachrichten in Netzen, elektronische Unterschrift) große Bedeutung erhalten. Einige mathematische Grundlagen der Kryptologie können somit praxisbezogen, handlungsorientiert, aber auch im historischen Zusammenhang im Unterricht behandelt werden, ohne jeweils vollständig präzisiert werden zu müssen; häufig genügt statt eines Beweises beispielhafte Verdeutlichung. Der RSA-Algorithmus kann nur an kleinen Primzahlen demonstriert werden.

Unterrichtsziele

Die Schüler sollen Texte verschlüsseln und entschlüsseln nach verschiedenen Methoden. Einige Aspekte zur Sicherheit der Verfahren sind zu untersuchen. Die historische und aktuelle Bedeutung der Kryptologie soll verdeutlicht werden.

Lerninhalte	Hinweise
1. Klassische Verfahren	
Monoalphabetische Chiffrierungen: - Transpositionschiffre (Verschiebeschiffre) - Verwendung eines Schlüsselwortes und eines Schlüsselbuchstabens.	Es sollte deutlich werden, daß im Unterschied zur Codierung mit öffentlichen Schlüsseln die Schlüssel nur den Sendern und Empfängern bekannt sind. Bei der Behandlung verschiedener Kryptosysteme sind jeweils auch die Sicherheit und damit die Möglichkeiten der Entschlüsselung ohne Kenntnis des Schlüssels zu untersuchen (Kryptoanalyse). Monoalphabetisch chiffrierte Texte lassen sich elementar durch Bestimmung der Zeichenhäufigkeit entschlüsseln.
Polyalphabetische Chiffrierungen.	Mindestens ein klassisches polyalphabetisches Verfahren (z.B. der Vigenère-Algorithmus und dessen Kryptonalyse mit Hilfe des Kasiski-Tests) soll behandelt werden.
2. Public-Key-Kryptosysteme	
Das RSA-Verfahren: - Schlüsselerzeugung - Verschlüsselung - Entschlüsselung.	Die Idee ist an einem einfachen Beispiel zu verdeutlichen (z.B. Koffer mit Namen und Schloß/Schlüssel). Hier bietet sich die Möglichkeit, einige Aspekte der elementaren Zahlentheorie zu behandeln. Es ist notwendig, auf einige Eigenschaften von Primzahlen hinzuweisen.

[Senatsverwaltung für Schule, Jugend, Sport: Vorläufiger Rahmenplan für Unterricht und Erziehung in der Berliner Schule, Klassen 9 und 10, Gymnasium, Wahlpflichtfach Mathematik, Berlin 1996, Seite 15].

3.4.4 Zufallszahlen – Grundlage für Simulationen

Zufallszahlen und Simulationen wurden in dieser Arbeit bereits an verschiedenen Stellen verwendet, siehe z. B. Kapitel 3.3.3, 3.3.4, 3.4.3. Dabei wurden die vom Programm bereitgestellten Zufallsgeneratoren benutzt, die in der Regel mit Befehlen wie random, random(5) usw. aufgerufen werden können. Das Thema wird im Unterricht immer dann relevant, wenn vom Rechner erzeugte Zufallszahlen benötigt werden.
Die Thematik „Zufallszahlen" wirft einige naheliegende Fragen auf, z. B.:

- Wie findet man mit dem Rechner „gute" (Pseudo-)Zufallszahlen? „Gut" bedeutet für Pseudozufallszahlen u. a. das Vorhandensein einer sehr langen Periode und das Bestehen diverser anderer Tests (z. B. des Pokertests). Wie kann man Gütetests durchführen?
- Wie kann man Zufallszahlengeneratoren programmieren?
- Zufallszahlen müssen häufig gemäß der gerade vorliegenden Aufgabenstellung transformiert werden. Wie geht das?

Diese und andere Fragen berühren Mathematik und Informatik. Hier soll nur der folgende Aspekt herausgegriffen werden:

Visualisierung von Zufallszahlen und einigen Transformationen
Einige Abbildungen sollen Anregungen zu Visualisierungen von Zufallszahlen geben. Man vergleiche hierzu auch das Deckblatt der Dissertation mit Zufallspunkten in einem Quadrat. Benutzt wird hier wieder die Software ANIMATO. Das Programm stellt Zufallszahlen mit den folgenden Befehlen zur Verfügung:

random (bzw. random(1)) , random(2), random(3) usw.	Jeder Aufruf liefert eine neue Zufallszahl.
rand (=rand(1)), rand(2), rand(3) usw.	Diese Variante sorgt dafür, dass die gerade ermittelte Zufallszahl z beim Aufruf verschiedener Funktionen gleich bleibt. In den Aufrufen f1(n, z), f2(n, z) usw. wird also immer dasselbe z verwendet. Erst ein neuer Aufruf mit einem anderen Wert für n benutzt eine neue Zufallszahl (aber für dieses n dann wieder immer dieselbe).

Abbildung 3.4.4-a zeigt die Visualisierung zweier Kongruenz-Zufallsgeneratoren. Hierzu wurde ANIMATO folgendermaßen programmiert.

Programm	Erläuterung
f1=0,1,1,1,1,0	zeichnet den Rand des Einheitsquadrats
f2={n=0:71124:frac((b*f3(n-1)+c)/d)}	ein Teil des in f3 definierten Bausteins (die mod-Funktion wird in ANIMATO anders ausgedrückt)
f3={n=0:f2(0,b,c,d):f2(n,b,c,d)*d}	Baustein für Kongruenz-Zufallsgeneratoren, Startwert 71124 (ein von mir erprobter Wert)
f4=f2(n-1,3^9,7,10^6+1),f2(n,3^9,7,10^6+1)	Visualisierung des Bausteins: Zwei aufeinanderfolgende Werte (f2(n-1), f2(n)) dienen als Punktepaar (x,y) Bausteinaufrufe. Der Zufallsgenerator heißt $x(n) := (3^9 * x(n-1) + 7) \pmod{10^6+1}$
f9=f2(n-1,3^2,7,10^2+1),f2(n,3^2,7,10^2+1)	Siehe f4, hier ein schlechter Zufallsgenerator $x(n) := 9*x(n-1) + 7 \pmod{101}$

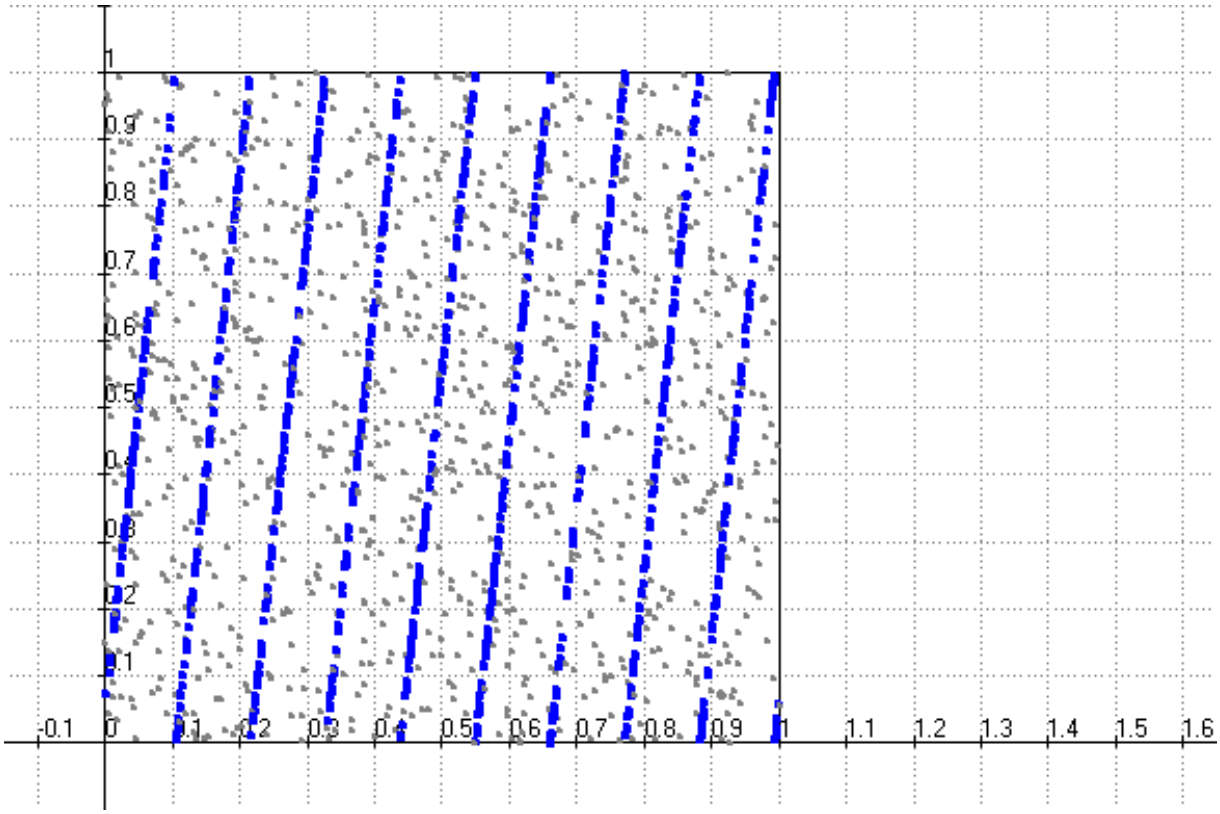


Abb. 3.4.4-a: Visualisierung der Ergebnisse zweier Kongruenz-Zufallsgeneratoren
(grau f4: gut, schwarz f9: schlecht)

Wertetafel für die Zufallsgeneratoren

x,t	f1	f4	f9	
0	0	?	?	
1	0	71124	71124	Startwert
2	0	0.93229907	0.85148515	
3	0	0.44255656	0.73267327	
4	0	0.84076265	0.66336634	
5	0	0.73120147	0.039603961	
6	0	0.23846826	0.42574258	
7	0	0.77076211	0.90099012	
8	0	0.91058633	0.17821799	
9	0	0.070815919	0.67326883	
10	0	0.86974132	0.12872644	
		scheint gut brauchbar zu sein	nicht brauchbar (siehe Abb. 3.4.4-a)	

Der Aufruf f4 zeigt – zumindest bei der gewählten Anzahl von Zufallszahlen – eine recht gleichmäßige Verteilung von Punkten über das Einheitsquadrat – allerdings ist das nicht das einzige Kriterium für die Güte des Zufallsgenerators. Der Aufruf f9 dagegen liefert sehr schnell Punkte auf parallelen Strecken, ein mögliches Kennzeichen für einen unbrauchbaren Zufallsgenerator.

Transformation von Zufallszahlen (Punktepaare)

f2 = int(4*random+1), int(4*random+1)	Zufallspunkte mit x, y aus {1,2,3,4}
f3 = int(6*random+1), -int(5*random+1)	Zufallspunkte mit x aus {1,2,3,4,5,6}und y aus {–1,–2,–3,–4,–5}
f4 = -int(3*random+1), int(3*random+1)	Zufallspunkte mit x aus {–1,–2,- 3} und y aus {1,2,3}
f5 = -int(2*random+1),-int(2*random+1)	Zufallspunkte mit x, y aus {–1,–2}
range0= 0,1,200	201 Punkte

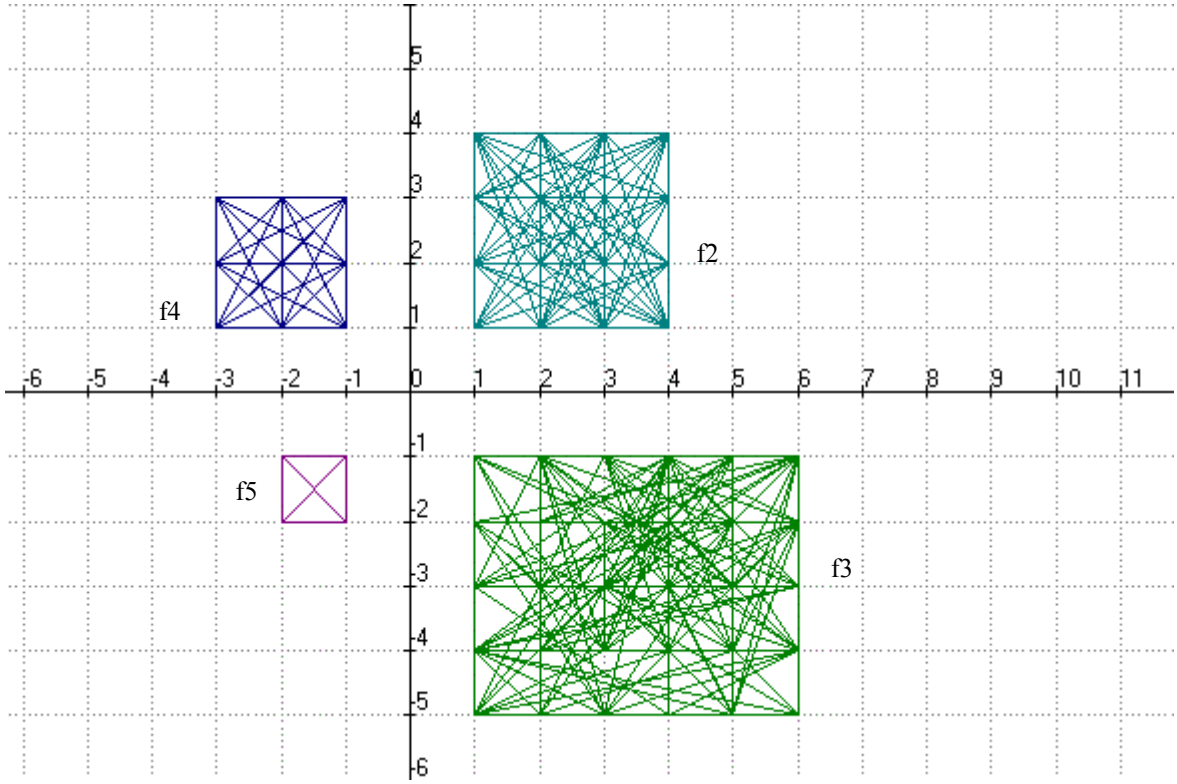


Abb. 3.4.4-b: Vier Transformationen f2. f3. f4. f5, Punktepaare

Drehen eines Glücksrads mit den Sektoren 1, 2, 3, 4

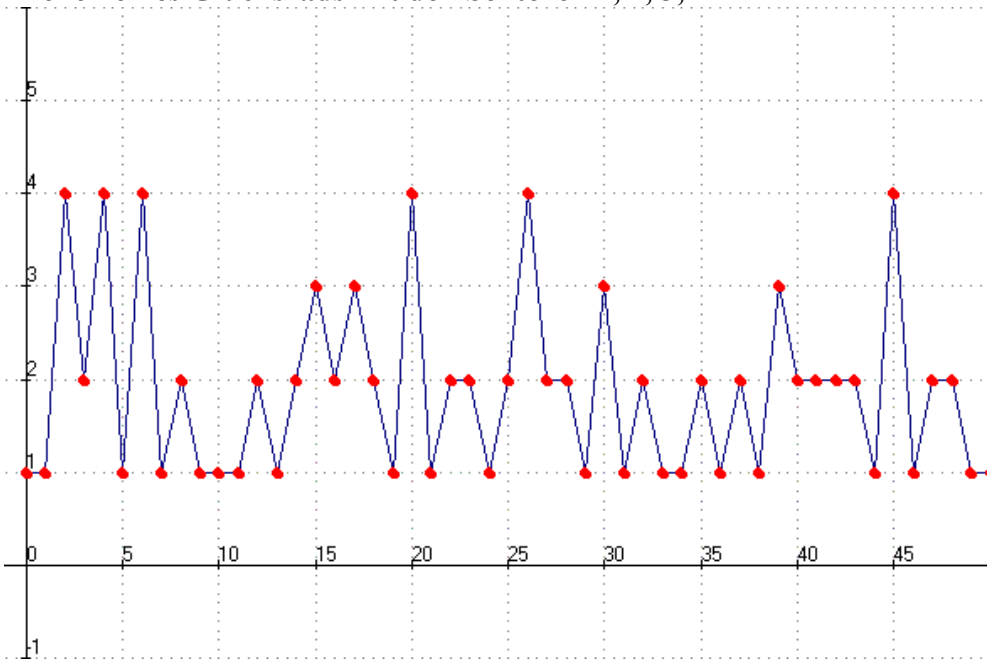


Abb. 3.4.4-c: Drehen eines Glücksrads mit den Sektoren 1, 2, 3, 4.
Erzeugende Formel: int(rand+1)+int(rand+0.6)+int(rand+0.2)+int(rand+0.1)+int(rand).

Die Visualisierung von Zufallszahlen und ihren Transformationen dient dem besseren Verständnis der Abläufe bei Problemlösungen durch Simulationen. Diese sollten heute Bestandteil jedes Stochastikkurses sein und können oft auch schon in der Sekundarstufe 1 als leicht einsichtige Lösungen schwierig aussehender Probleme verwendet werden.

3.4.5 Einige Elemente der Computergrafik

Computergrafik ist ein für Schüler besonders attraktives Thema, das auch viele Bezüge zwischen Mathematik und Informatik bietet. Im Folgenden werden einige Bereiche angedeutet, die sich für verschiedenen Unterrichtssituationen eignen.

(A) Abbildungsgeometrie in der Sekundarstufe 1: Bereits ab Klasse 9 können (2,2)-Matrizen dazu verwendet werden einfache Objekte (Gerade, Dreiecke, Parabeln usw.) abzubilden. Dabei können die aus der Geometrie bekannten Abbildungen wie Spiegelungen an Achsen, am Nullpunkt, Scherungen usw. nun unter analytischen und programmiersprachlichen Aspekten (funktionales Programmieren) aufgegriffen werden.	(B) Abbildungsgeometrie, nun Sekundarstufe 2 in Analysis oder Lineare Algebra/Analytische Geometrie-Kursen: Siehe Ausführungen unter (A). Aber nun werden komplexere Objekte (z.B. Kreis, Ellipse) und Abbildungen genommen; siehe Kapitel 3.4.5.1.	(C) Die in (A) und (B) genannten Hinweise können auf Abbildungsgeometrie im dreidimensionalen Raum übertragen werden (Einbettung in den Unterricht zur Analytischen Geometrie).

(D) Untersuchung von Darstellungsproblemen bei geometrischen Objekten auf dem Bildschirm (z. B. Geradendarstellung, hidden-line-Problem), siehe Kapitel 3.4.5.2.	(E) Benutzen und Analysieren von Elementen fertiger Software zur Analytischen Geometrie, zum Ray-Tracing und von CAD-Systemen. Die Einbettung kann ebenfalls im Kurs Analytische Geometrie erfolgen, siehe Kapitel 3.4.5.2.	(F) Gestaltung von Animationen mathematischer Zusammenhänge, siehe z. B. Anwendungen des Programms ANIMATO, siehe Kapitel 2.1.7.

3.4.5.1 Abbildungsgeometrie mit Matrizen

Matrizen sind ein wesentliches Hilfsmittel für Probleme der Computergrafik. Damit wird ein direkter Bezug zu algebraischen und geometrischen Fragestellungen im Kurs „Lineare Algebra und Analytische Geometrie" hergestellt. Dieser gewinnt damit wesentlich an Attraktivität. Als Beispiel wird eine von mir gestellte Abituraufgabe aus dem Jahr 2001 gewählt. Die Aufgabe zeigt u. a.

- Verwendung moderner Technologie im Mathematikunterricht
- Bezüge zur Informatik durch Computergrafik und Programmieren mit Funktionen
- Umgehen mit einer offenen Teilaufgabe (siehe Aufgabenteil 3.1) auch im Abitur
- Arbeit an vorgegebenem Material

Die Problemstellungen sind ohne Mühe auf den Unterricht übertragbar.

Leistungskurs Mathematik (Lehmann) – Abitur 2001, Abbildungsgeometrie, Kurs MA-3 Vorschlag 2, Aufgabe 3 – 27 Bewertungseinheiten

Die Anlage enthält eine Grafik, erstellt mit dem Funktionenplotter PLOT11, sowie Daten zur Erstellung der Grafik.

3.1 Erläutern Sie die Grafik und ihren Entstehungsprozess unter Aspekten der Abbildungsgeometrie. (ca. 30´)

Wichtige Hinweise:

- Beginnen Sie mit der Beschreibung der Ausgangsellipsen (diese jeweils farbig markieren).
- Notieren Sie in Ihrer Bearbeitung Terme und Bausteine in der üblichen mathematischen Notation (Matrizen usw.).
- Strukturieren Sie Ihre Erläuterungen durch geeignete Nummerierungen der betrachteten Aspekte.

3.2 Gegeben ist die Matrix A = [[3, –4] [4, 3]] (TI-92-Notation). (ca. 40´)

a) Zerlegen Sie die Matrix so, dass die Abbildungseigenschaften deutlich werden.

b) Die Matrix wird nun auf die Parabel mit der Gleichung $y = -x^2$ mit x aus [–1, 1] angewendet. Zeichnen Sie die Parabel und ihr Bild mit dem TI-92 (parametric) und übernehmen Sie die Zeichnung und Daten in gewohnter Weise in Ihre Klausurarbeit. Verdeutlichen Sie in der Zeichnung die Abbildungseigenschaften (beachten Sie hierzu Teil c)).

c) Die Ausgangsparabel enthält u. a. die Punkte P1(–1, –1) und P2(1, –1). Überprüfen Sie ihre Zeichnung durch Berechnung der beiden Bildpunkte P1´und P2´. Tragen Sie alle Punkte in ihre Zeichnung ein.

Anlage zu Vorschlag 2, Aufgabe 3 (Kurs MA-3, Abbildungsgeometrie)

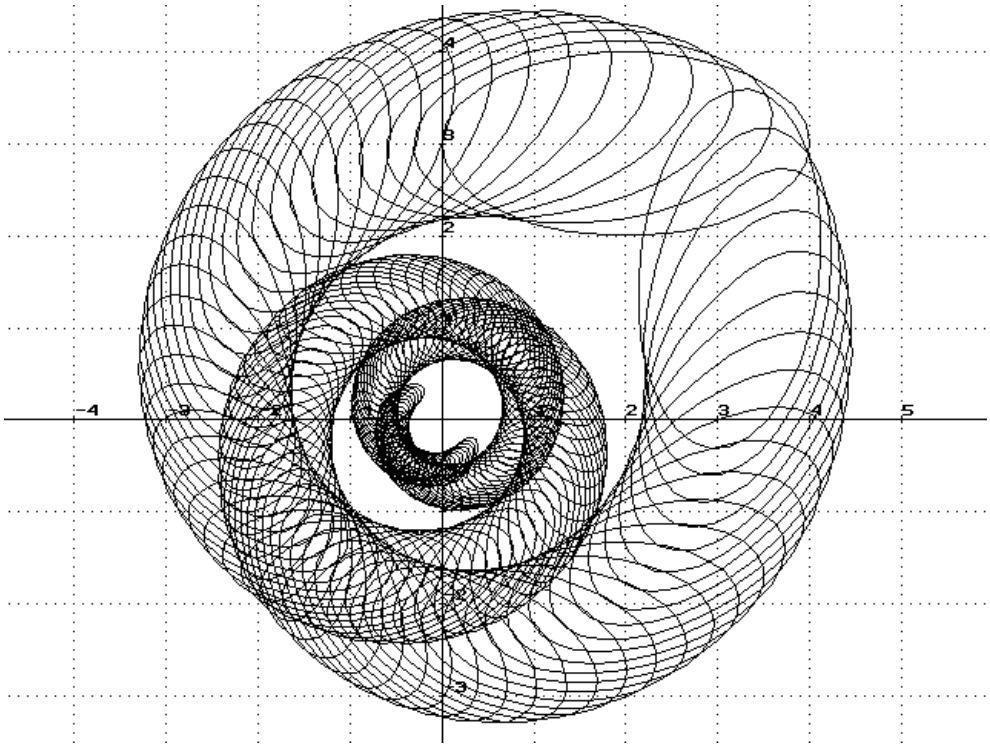
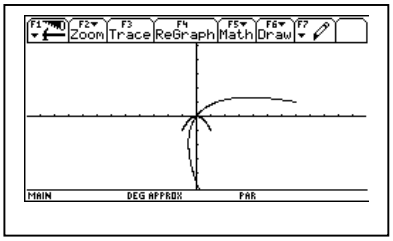


Abb.3.4.5.1-a

f1 0.98 f2 pi/30	f3 f1^n*cos(n*f2) f4 f1^n*sin(n*f2)	f5 2cos(t)+2 f6 sin(t)+3 f7 f5,f6	f8 f3(u)f5-f4(u)f6, f4(u)f5+f3(u)f6 f9 f4(u)f5+f3(u)f6,f3(u)f5-f4(u)f6	Laufbereiche der Variablen t,x 0.00E+00 6.28E+00 2.09E-01 30 u bzw. n 1.00E+00 1.00E+02 1.00E+00 99

Abb. 3.4.5.1-b: Programmierung

Leistungskurs Mathematik – Abitur 2001
Vorschlag 2, Aufgabe 3 – Erwartungshorizont, Lösungen

Aufgabenteil, Lösungsskizzen, Erwartungen	Bewertungseinheiten, Anforderungsbereiche, Erläuterungen
Aufgabenteil 3.1 f1 0.98 *Streckfaktor* f2 pi/30 *Drehwinkel 6°* f3 f1^n*cos(n*f2) *0.98^n*cos(n*6°)* f4 f1^n*sin(n*f2) *0.98^n*sin(n*6°)* *Drehstreckmatrix, Elemente (1,1) und (2,1)* f5 2cos(t)+2 *Urbild x-Wert* f6 sin(t)+3 *Urbild y-Wert*, f7 f5,f6 um vx=2, vy=3 *verschobene Ellipse* f8 f3(u)f5-f4(u)f6, f4(u)f5+f3(u)f6 f9 f4(u)f5+f3(u)f6,f3(u)f5-f4(u)f6 *Matrix D^n * Urbild, [[f3, -f4][f4, f3]] * Urbild* t,x 0.0000000000E+00 6.2800000000E+00 Schrittweite 2.0933333333E-01 30 *Laufbereich für die Winkel t in f5,f6, f7* u bzw. n 1.0000000000E+00 1.0000000000E+02 1.0000000000E+00 99 n läuft von 1 bis 100 (100 Ellipsen)	**AB1 AB2 AB3** Für die Erläuterung der Zeichnung sind ca. 30 Minuten vorgesehen. – Es handelt sich um die Drehstreckung einer Ellipse (f7) mit Hilfe der ersten 100 Potenzen der Drehstreckmatrix 0.98^n*[[cos(n*6°), –sin(n*6°] [sin(n*6°), cos(n*6°]]. Die Schüler erkennen das aus dem gegebenen Plot-Programm in Verbindung mit der Abbildung und beschreiben die Auswirkung in Textform unter Benennung der Terme und der Verknüpfung der einzelnen Programmelemente. Hierfür werden **insgesamt 12 BE** vergeben **4 BE, AB 1** für Erkennen grundlegender Elemente **5 BE, AB 2** für Erkennen der Zusammenhänge **3 BE, AB 3** für schwierige Zusammenhänge, wie eigenständiges Erkennen, welche Auswahl von f1 bis f9 hier gezeichnet wird, worin sich f9 von f8 unterscheidet, warum die Ellipsen immer kleiner werden
Aufgabenteil 3.2 a) Drehstreckung: Streckfaktor k=5, Drehwinkel aus cos(t)=3/5, t=53,13° b) Arbeit in Parameterdarstellung am TI-92, Zeichnung erstellen und mit Daten auf Papier übernehmen. Drehwinkel und Streckung verdeutlichen. Lösung im parametric-Editor xt1 t yt1 –t^2 xt2 3t-4(-t^2) yt2 4t+3(-t^2) window tmin –1 tmax 1 c) Matrix * Punkte ergibt Bildpunkte, Punkte markieren, P1´(1, –7), P2´(7,1)	**3 BE, AB 1** **4 BE, AB 2** **4 BE, AB 2** **4 BE, AB 1**  TI-92-Zeichnung zu 3.2 b.
Summe / % **27 BE (100 %)**	**AB 1 AB 2 AB 3** **11 (41 %) 13 (48 %) 3 (11 %)**

Weitere Erläuterungen zu der Aufgabe: Die Angabe der Bearbeitungszeiten ist bei dieser Aufgabe nötig, um bei Teil 3.1 den erforderlichen Umfang der Erläuterungen in etwa zu charakterisieren.

3.4.5.2 Weitere Probleme aus der Computergrafik

Wie entsteht eine Gerade bzw. Strecke auf dem Bildschirm?

Für die Untersuchung dieses Problems bietet sich der Bresenham-Algorithmus an – eine überzeugende Verbindung zwischen Mathematik und Informatik. Diesen Algorithmus findet man u. a. ausführlich erläutert unter der unten angegebenen Internet-Adresse. Weitere Adressen erhält man durch Eingabe des Stichworts „Bresenham".

Literaturangaben

- Bresenham-Algorithmus:
  http://www.uni-paderborn.de/fachbereich/AG/agdomik/computergrafik/cg_skript/html/node34.htm

- *Werner u. a.: Taschenbuch der Informatik, Fachbuchverlag Leipzig, 2.Auflage 1995, S. 534 f.* – dort auch mehr über Computergrafik

Konstruktion fotorealistischer Szenen

Ray-Tracing-Programme ermöglichen die Programmierung fotorealistischer Darstellungen auf dem Bildschirm. Sie sind eine wesentliche Bereicherung für den Kurs „Lineare Algebra und Analytische Geometrie", da hier auch andere als die üblichen Unterrichtsobjekte Geraden, Ebenen, Kugeln im Raum verwendet werden können, wie Kegel, Zylinder usw. Die Programmierung ist durch die Verwendung von Bausteinen für solche Objekte relativ leicht möglich. Für die Gestaltung der Oberflächen (Beleuchtung, Farbe, Oberflächenzustand, ...) stehen einfache Anweisungen zur Verfügung. POVRAY ist eines dieser Programme. Es ist leicht im Internet zu finden und kann von dort heruntergeladen werden.

Beispiel für eine POVRAY-Anwendung:

Spalte 1: 1. Programmteil

```
#include "colors.inc"

camera{ location <0,0,-6>
      look_at <0,0,0>       }

light_source {<-10,10,-5> color White}


  sphere {       <0,0,0>,1
      pigment {Yellow}
      finish {phong 1}      }

light_source {<10,5,-4> color Gray50}

sphere{       <2.2,0,-1.2>,0.25
      pigment {Red}
      finish {phong 1}      }
```

Spalte 2: Fortsetzung

```
sphere{       <1.2,-1.2,0>,0.5
     pigment {Yellow}
     finish {phong 1}       }

sphere{       <-1,1,1.1>,0.2
     pigment {Red}
     finish {phong 1}       }

cylinder{ <-2,-2,-2>,<6,2,2>,0.04
      pigment {Red}
     finish {phong 1}       }

plane { <0.5, -0.7,0.51>, -1
     pigment {
     checker color Gray, color Blue    } }
```

3.4.6 Unerwartetes in Bildern

Eine Unterrichtseinheit über unerwartete Ergebnisse am Computer sollte nicht fehlen. Hierfür gibt es diverse Möglichkeiten. Hier werden drei unerwartete Fälle dargestellt, die im Unterricht vermutlich meistens übergangen werden.
Das Hauptziel derartiger Beispiele ist es, das Vertrauen in den Computer zu erschüttern. Aus der Informatik erhalten wir die Antworten, wenn wir der Frage nachgehen: **Wie rechnet der Computer?** Passende Beispiele hierzu findet man u. a. in dem Beitrag *„Per Kopf oder Knopf? Rechnen können oder lassen?" von Ingmar Lehmann in „Medien verbreiten Mathematik", Bericht über die 19. Arbeitstagung des Arbeitskreises „Mathematikunterricht und Informatik" in der GdM, 2001 in Dillingen (Hrsg. Herget, Sommer, Weigand, Weth), Franzbecker-Verlag 2002.*

Problem 1: Unerwartete Graphen: y = sin(x) und y = sin(119x)

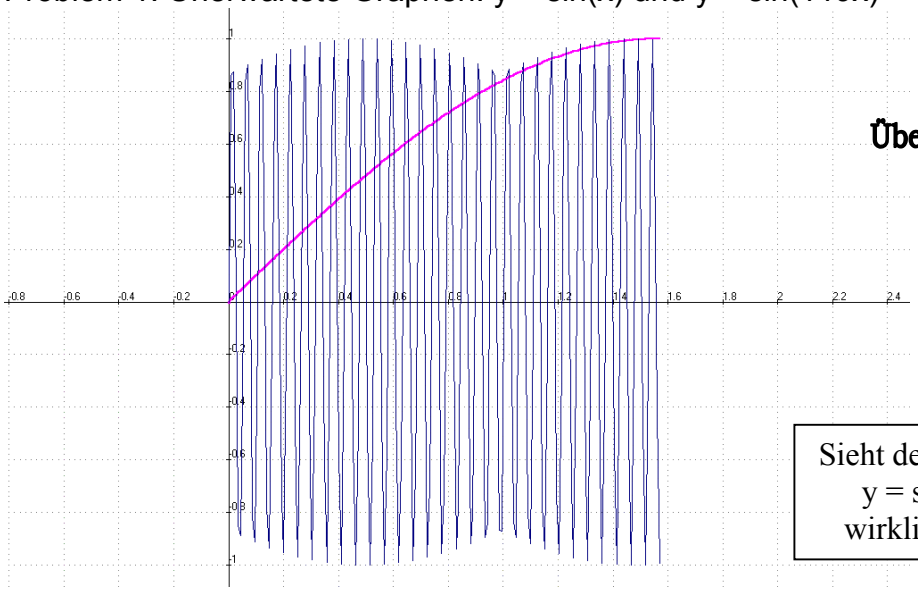


Abb.3.4.6-a: y = sin(x) und y = sin(119x) , x läuft von 0 bis 1.57, 180 Werte

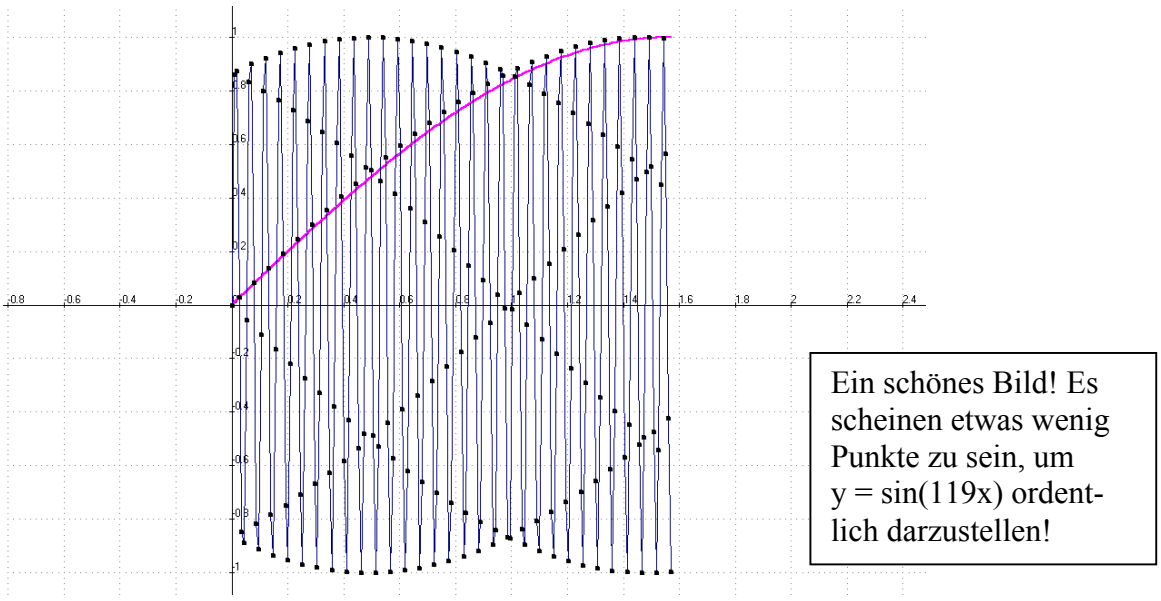


Abb.3.4.6-b: y = sin(x) und y = sin(119x) , x von 0 bis 1.57, 180 Werte. Für y = sin(119x) wird noch einmal gezeichnet, jedoch ohne die Punkte miteinander zu verbinden.

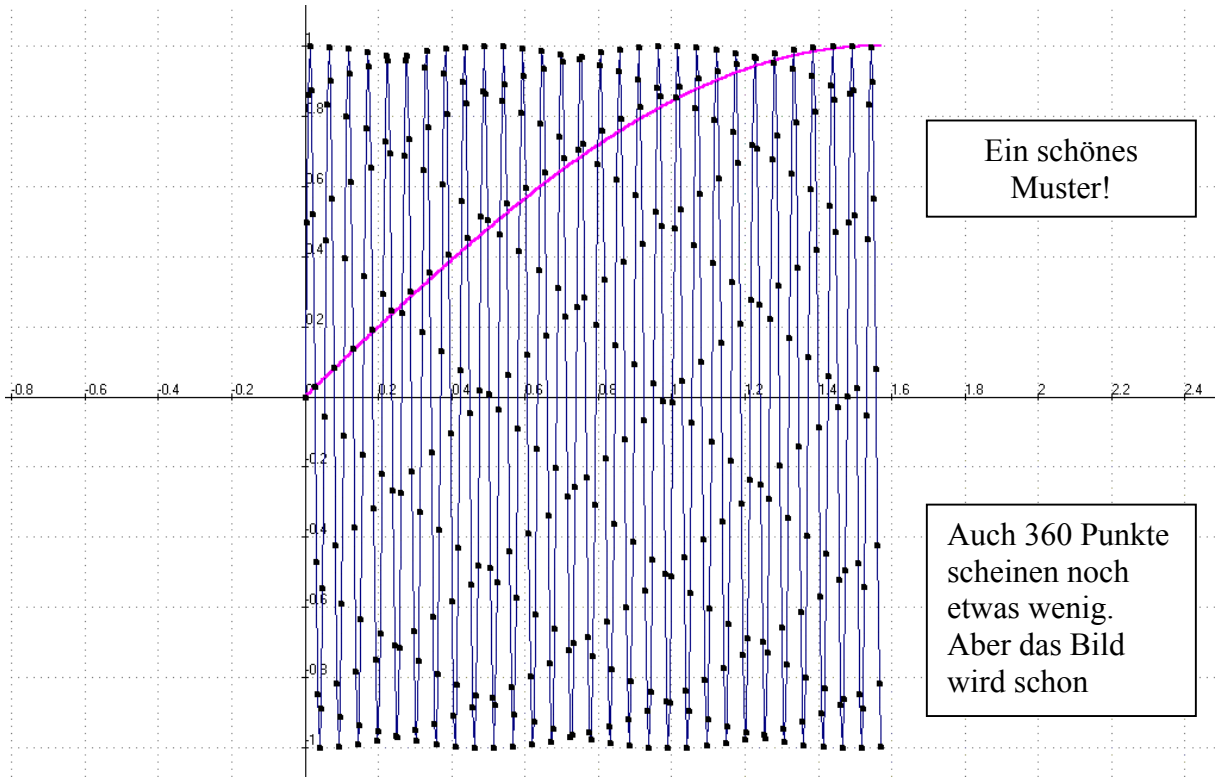


Abb.3.4.6-c: y = sin(x) und y = sin(119x) , x von 0 bis 1.57, 360 Werte

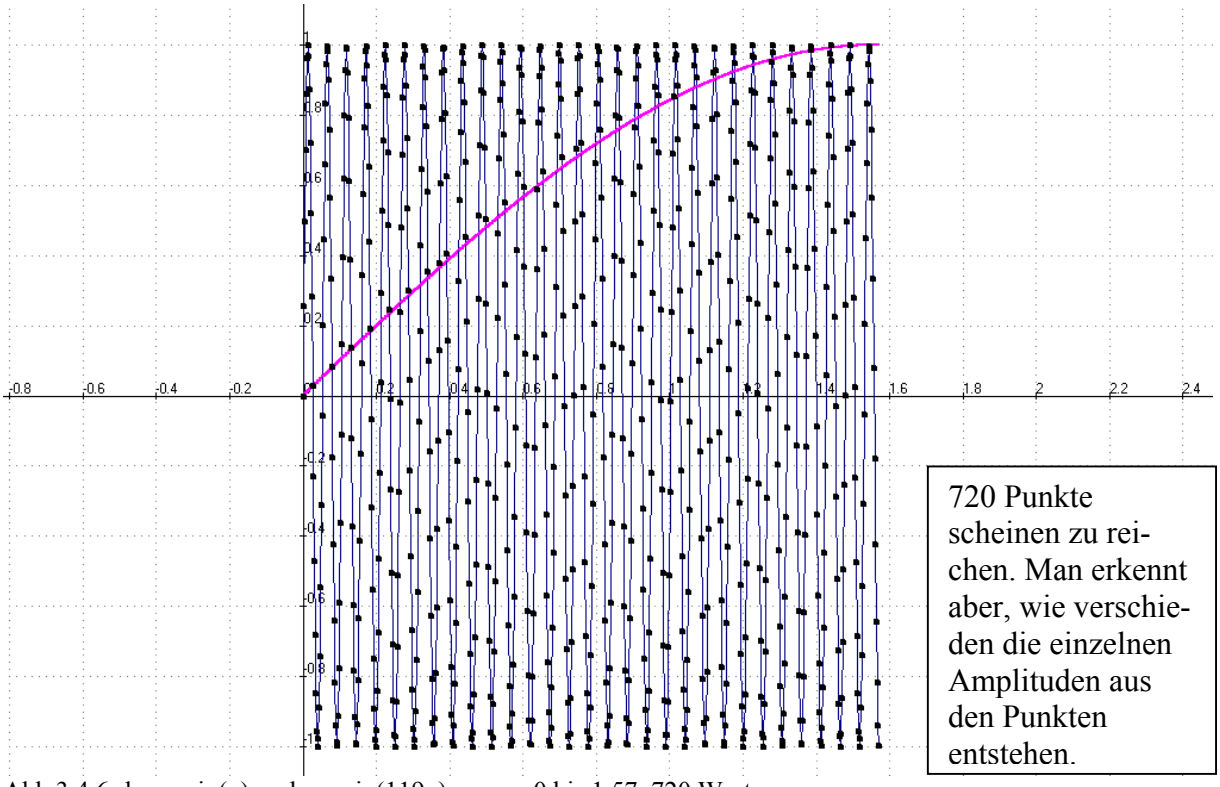


Abb.3.4.6-d: y = sin(x) und y = sin(119x) , x von 0 bis 1.57, 720 Werte.

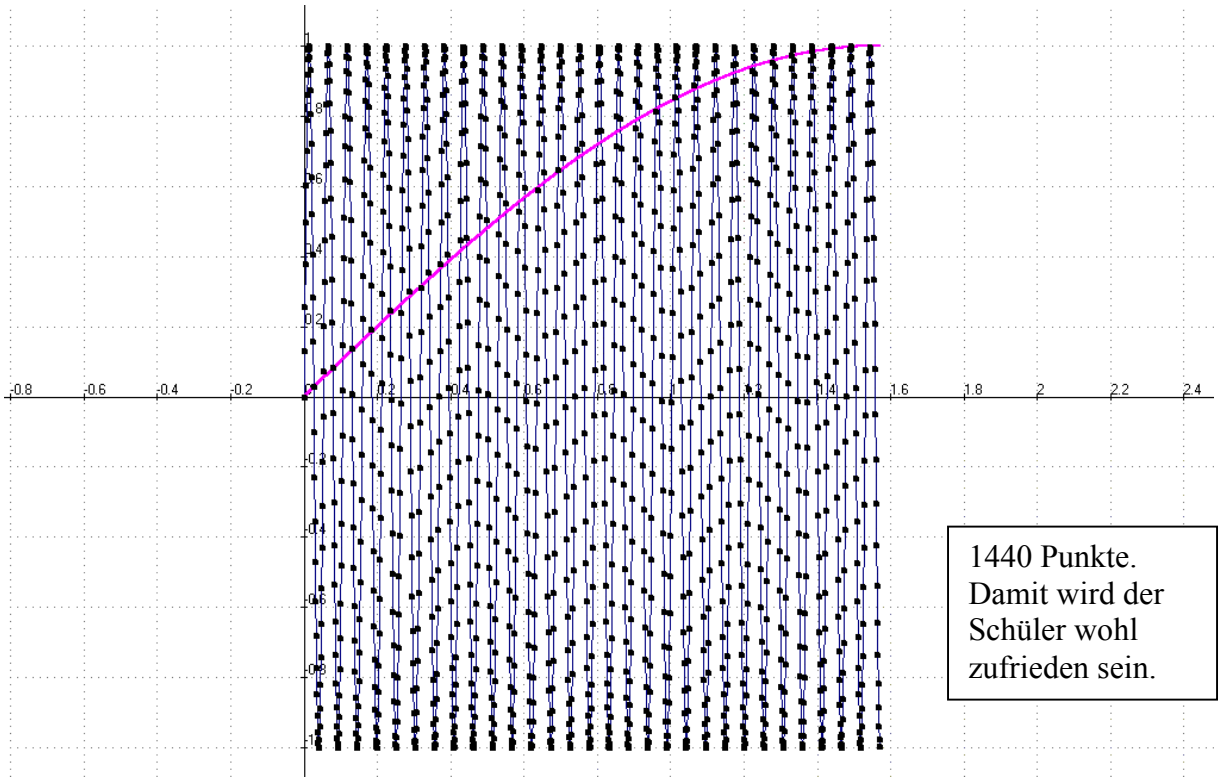


Abb.3.4.6-e: y = sin(x) und y = sin(119x) , x von 0 bis 1.57, 1440 Werte.

Also Vorsicht mit der Darstellung von Graphen! Nicht zu wenig Punkte nehmen!

Problem 2: Der Differenzenquotient von y = sin(x)

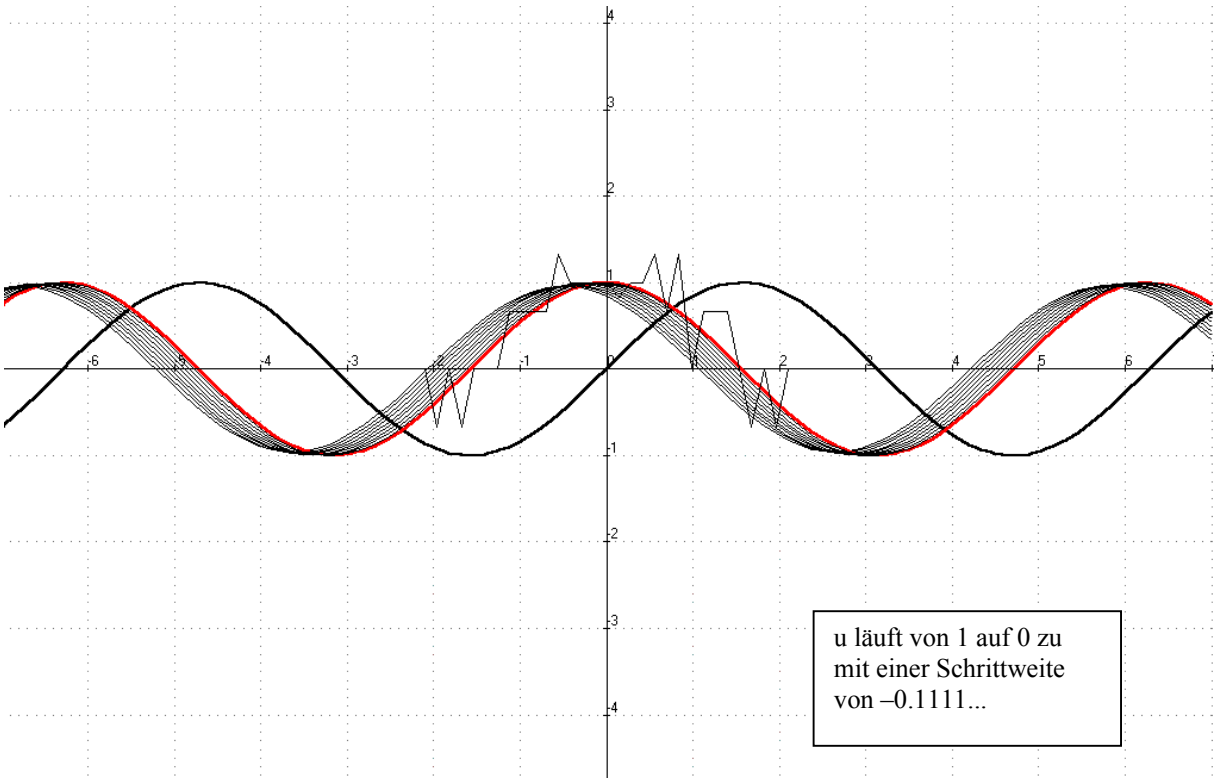


Abb. 3.4.6-f: Differenzenquotienten-Funktion $d(x,u)=\frac{\sin(x+u)-\sin(x)}{u}$ zu y = sin(x)

Man kann wunderschön verfolgen, wie sich der Graph des Differenzenquotienten von

y = sin(x) erwartungsgemäß immer mehr dem Graphen von y = cos(x) nähert. Doch zum Schluss plötzlich diese Zacken! Das ist Anlass zum Experimentieren mit kleinem u. Dazu wird ein Baustein zum bequemen Ändern von Werten definiert, zum Beispiel $u = 10^{-12}$. Zum Unterscheiden von den schon vorhandenen Graphen lässt man jetzt Punkte zeichnen. Deren Graph sieht noch recht gut aus!

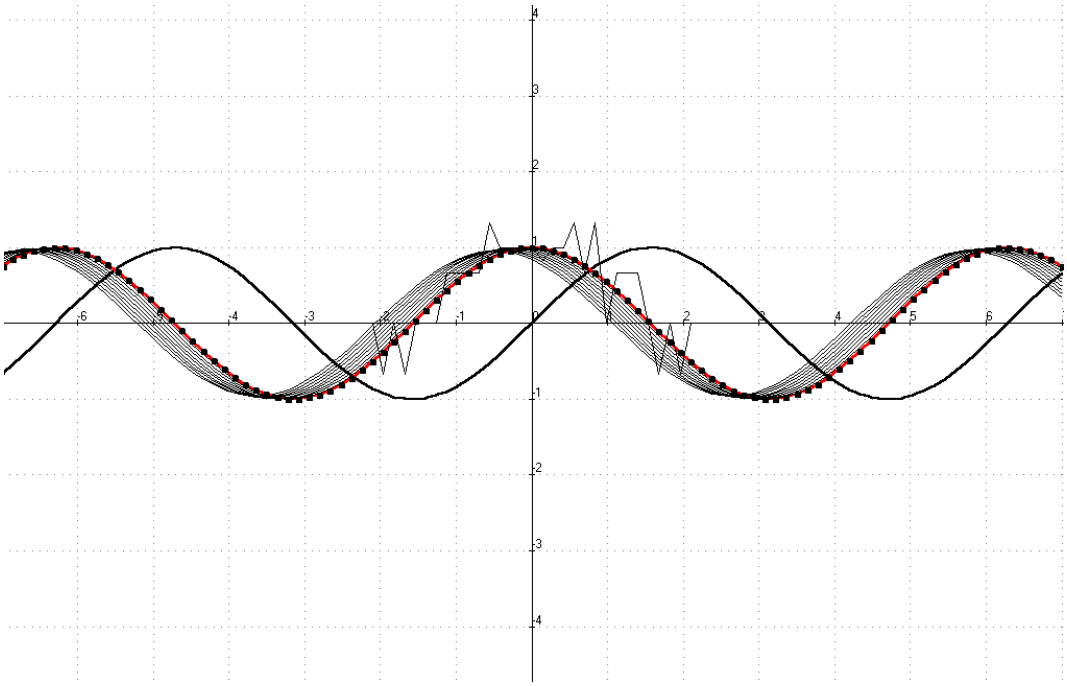


Abb. 3.4.6-g: Noch kann man mit den Punkten zufrieden sein.Weiter experimentieren! Bei $u = 10^{-15}$ wird es kritisch:

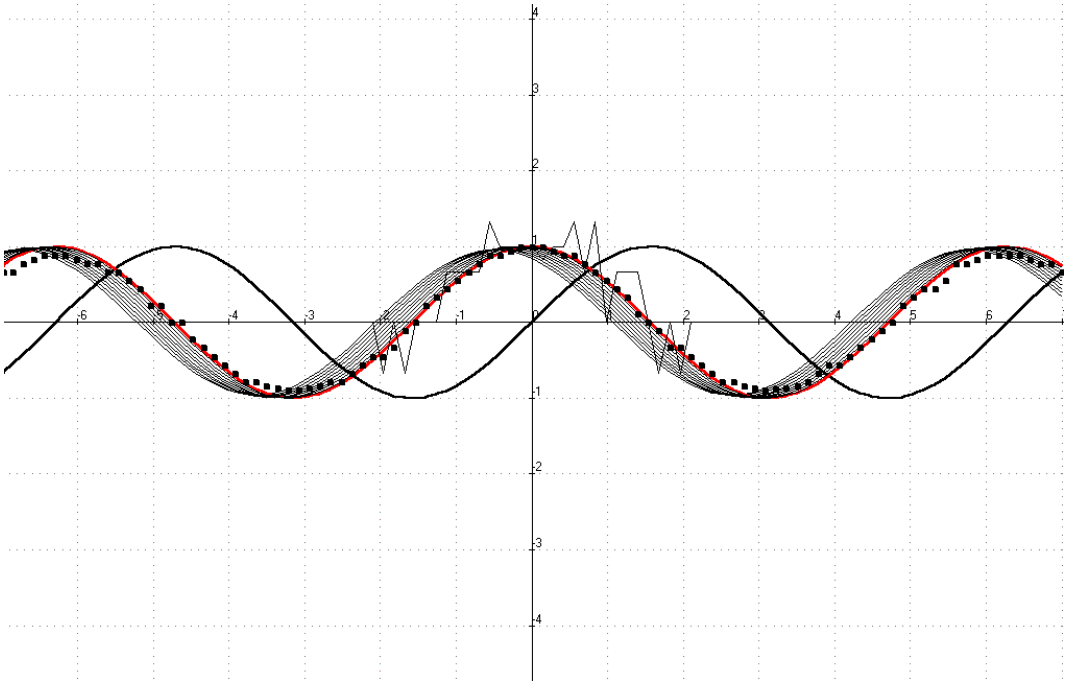


Abb. 3.4.6-h: Nun sieht auch der Punktegraph schon etwas seltsam aus! Jetzt wird $u = 10^{-16}$ gewählt.

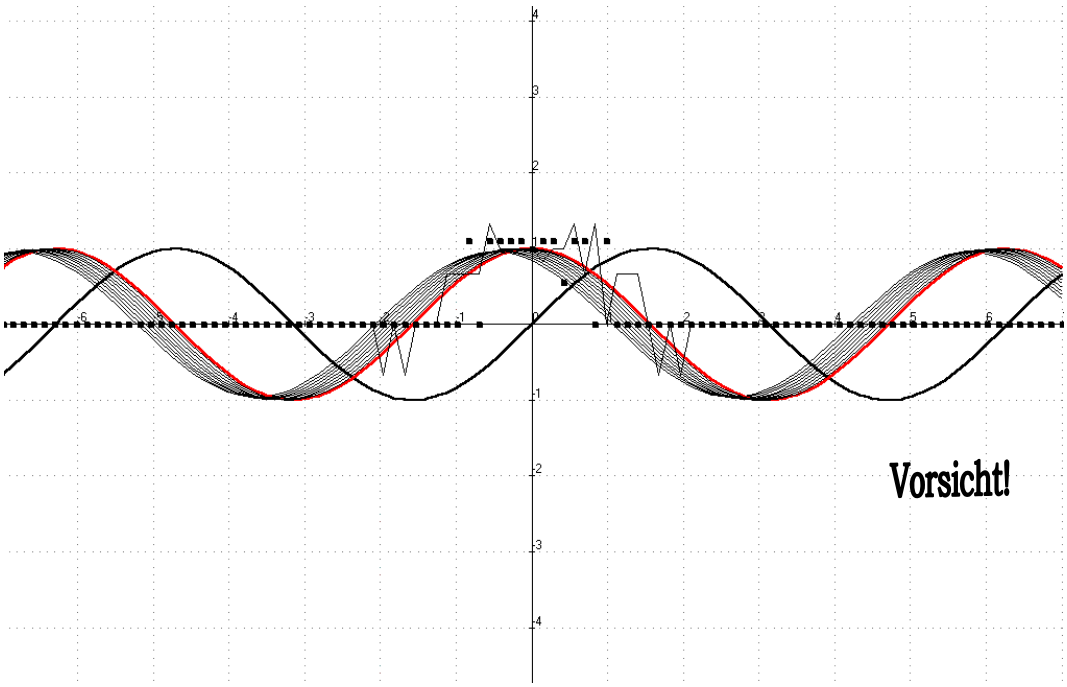


Abb. 3.4.6-i: Das hat nun nichts mehr mit der cos-Funktion zu tun!

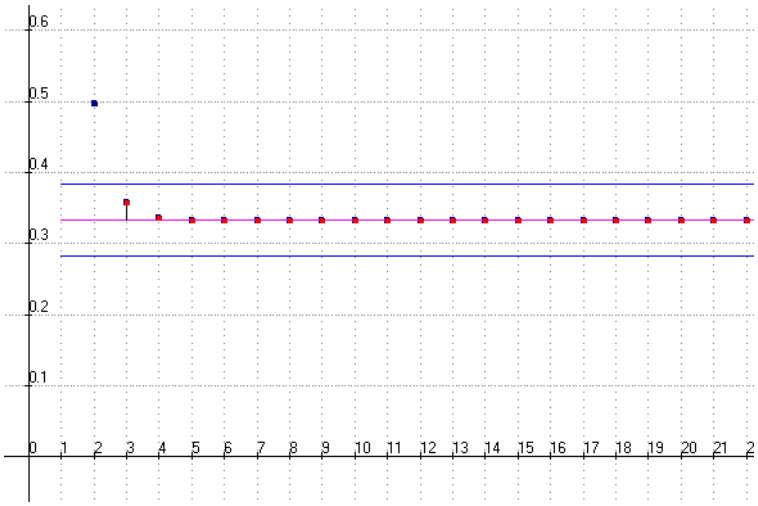
- Also Vorsicht mit der Veranschaulichung von Grenzwerten!

Woran liegt es? Es wird die Division durch sehr kleine Werte sein.

- Vorsicht bei der Division durch sehr kleine Zahlen!
- Weitere Informationen zum Thema „Konvergenzverhalten" an Beispielen passend ausgesuchter Folgen finden man z. B. bei *[Her90], S. 49–55] oder bei [Koe95], S. 65f.*.

Das folgende Beispiel stammt aus diesem Aufsatz, wird hier jedoch auf andere Weise veranschaulicht.

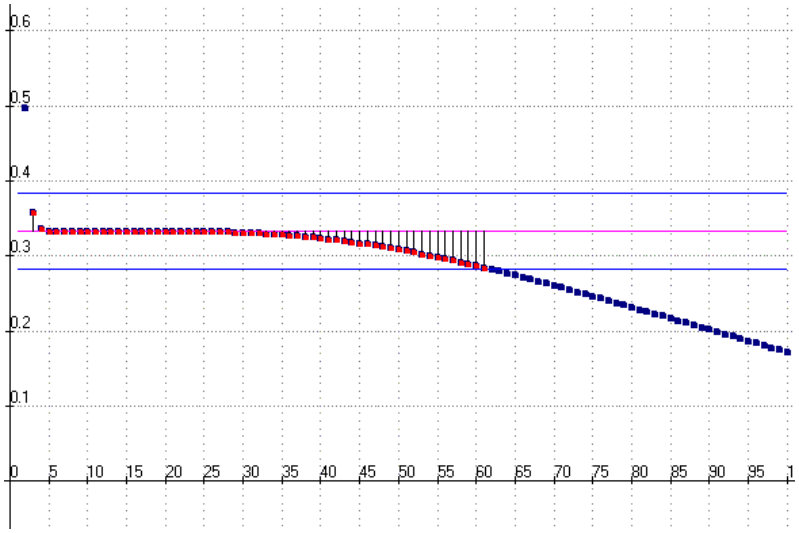
Problem 3: Folgengrenzwert



$$\lim_{n\to\infty}\left(\frac{1}{3}+\frac{(11-n)^9}{(9+n)^9}\right)$$ D

er Grenzwert scheint 1/3 zu sein, die Folgenglieder liegen in der Epsilon-Umgebung für ε =0.05.

Abb.3.4.6-j: Darstellung der Folgenglieder bis n=22, die Epsilonumgebung ist hier 0.05



Die Folgenglieder verlassen die Epsilonumgebung!

Abb.3.4.6-k: Darstellung der Folgenglieder bis n=100, Epsilon ist hier 0.05

Abb.3.4.6-l: Darstellung der Folgenglieder bis n=1000

Jetzt sollte doch lieber gerechnet werden – von Hand oder mit dem CAS!

$$\lim_{n\to\infty}\left(\frac{1}{3}+\frac{(11-n)^9}{(9+n)^9}\right)=\lim_{n\to\infty}\left(\frac{1}{3}+\frac{n^9\left(\frac{11}{n}-1\right)^9}{n^9\left(\frac{9}{n}+1\right)^9}\right)=\lim_{n\to\infty}\left(\frac{1}{3}+\frac{\left(\frac{11}{n}-1\right)^9}{\left(\frac{9}{n}+1\right)^9}\right)=\frac{1}{3}-\frac{1}{1}=-\frac{2}{3}.$$

Eine erneute Darstellung der Folge berücksichtigt dieses Ergebnis und setzt diesen Grenzwert an. n läuft nun bis 5001. Offenbar gelangen dabei die Folgenglieder wieder in die Epsilon-umgebung.

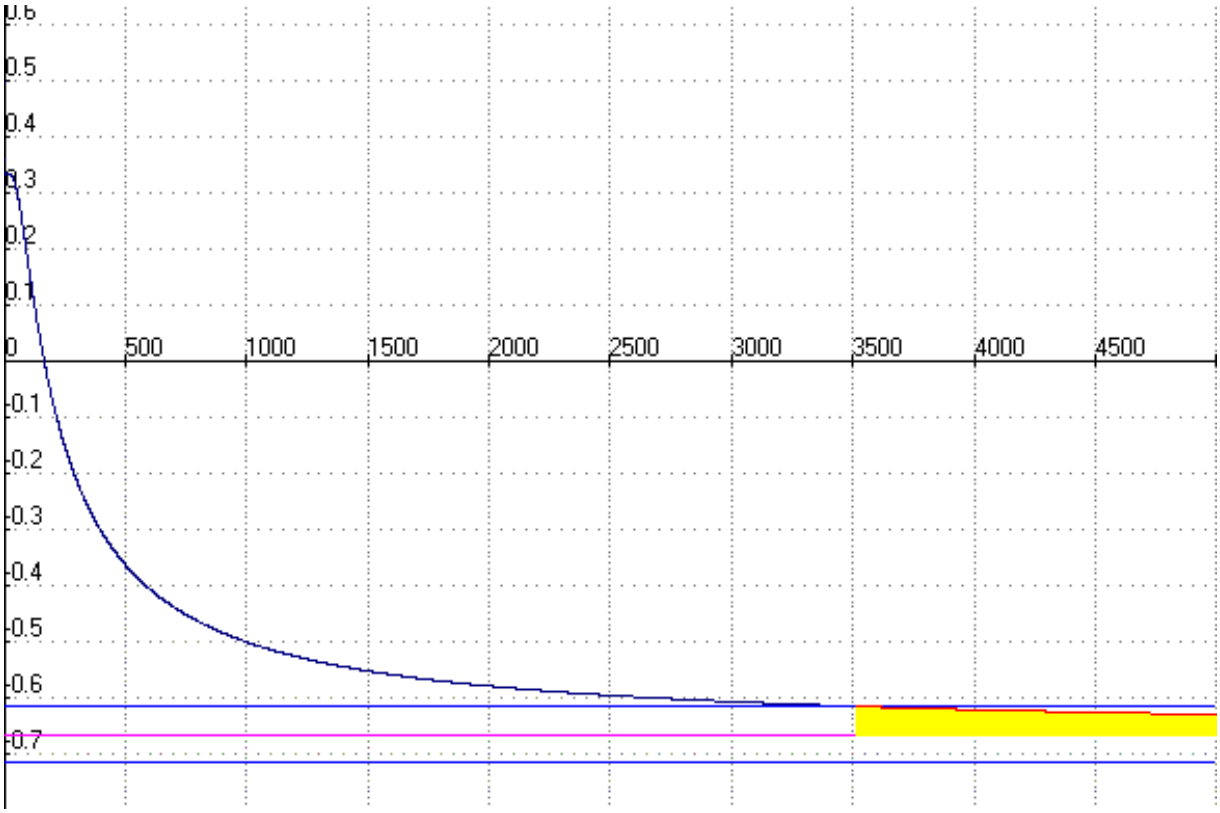


Abb.3.4.6-m: Darstellung der Folgenglieder bis n=5000, die Epsilonumgebung ist 0.5, der vermutete Grenzwert ist –2/3.

Wertetafeln und Rechnungen mit dem CAS bestätigen diese Ergebnisse.

Zur grafischen Darstellung wurde hier wieder im Programm ANIMATO funktional programmiert.

```
f1=1/3+(11-n)^9/(9+n)^9                    der Folgenterm
f2=-2/3+0*n                                der vermutete Grenzwert
f3=0.05                                    das gewählte Epsilon
f4=f2+f3                                   Epsilon-Umgebung
f5=f2-f3                                   Epsilon-Umgebung
f6=n,{abs(f1(n)-f2)<f3:f1(n):undef}        Wenn in der Epsilon-Umgebung,
f7=n,f2,n,{abs(f1(n)-f2)<f3:f1(n):undef}  dann senkrechte Strecken zeichnen.
```

In diesem Zusammenhang wird verwiesen auf
Lehmann, E.: Epsilon-Delta, ein neuer Weg zum Verständnis des Grenzwertbegriffs durch Veranschaulichung mit dem Computer, (Praxis der Mathematik, 1993, Heft 5)

- Was lernt man an dem Beispiel? – Keine vorschnellen Schlüsse über Folgengrenzwerte!

Problem 4: Wo ist der Schnittpunkt?

Man löse die sich nur geringfügig unterscheidenden linearen Gleichungssysteme:

a) 0.1000x + 1y = 2 und b) 0.10000x + 1y = 2
0.1001x + 1y = 2.001 **0.10001x** + 1y = 2.001.

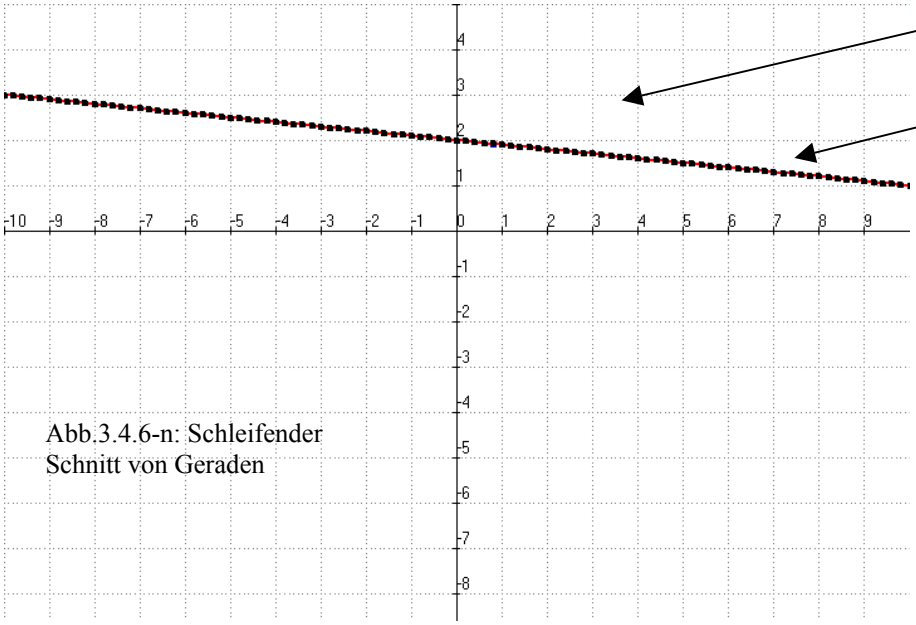


Abb.3.4.6-n: Schleifender Schnitt von Geraden

Geradenpaar a
y=2.000–0.10000x
y=2.001–**0.10010x**
L={(10, 1)}

Geradenpaar b
y=2.000–0.10000x
y=2.001–**0.10001x**
L={(100, –8)}

Die Überlegungen lassen sich auf umfangreichere Gleichungssysteme übertragen.

- Bei schlecht konditionierten LGS ist höchste Vorsicht geboten!

a) hat offenbar die Lösung L = {(10, 1)}, für b) ist L = {(100, –8)}. Obwohl die Gleichungssysteme sich nur an einer Stelle geringfügig unterscheiden (oben fett markiert), sind die Lösungsmengen arg unterschiedlich. Eine grafische Darstellung (Abb. 3.4.6-j) zeigt die Situation: Alle Geraden liegen bei dem gewählten Maßstab fast aufeinander – sie haben einen „schleifenden Schnitt".
Die Berechnung der Lösungsmenge eines schlecht konditionierten LGS kann bei der Wahl sehr kleiner Pivotelemente (etwa nach dem Gauss-Algorithmus) zu Schwierigkeiten führen. Über die Auswirkungen, die kleinste Änderungen an einem System auslösen können, kann man u. a. in dem Buch von *Peitgen* nachlesen: *[Pei92], Seite 52f.*

Oben wurden bereits einige Beispiele erläutert, die die Frage nach der Exaktheit der vom Computer erzeugten Ergebnisse als sehr relevant nachwiesen.

Zu dem Problem von „Black-Box-Software" ein Zitat aus dem oben genannten Werk von *Peitgen u. a.*:
„Immer häufiger werden Berechnungen heutzutage mit Hilfe von Black-Box-Software Paketen ausgeführt, deren genaue Funktionsweise verborgen bleibt. Diese Pakete werden manchmal von renommierten wissenschaftlichen Zentren entwickelt und scheinen deshalb sehr zuverlässig zu sein. In der Tat sind sie es auch. Das schließt aber nicht aus, daß die hochwertigste Software manchmal völligen Unsinn produziert, und es ist eine Kunst für sich, zu verstehen und vorauszusagen, wann und weshalb dies geschieht. Außerdem haben die Benutzer oftmals keine Möglichkeit, eine Fehleranalyse durchzuführen, einfach deshalb, weil sie keinen Zugang zum Black-Box-Algorithmus haben."

3.5 Lineare Algebra – ein Kurskonzept mit Matrizen, Computereinsatz und informatischen Anteilen

Die weitreichendste Form einer Verknüpfung von Mathematik und Informatik im Unterricht ist die durchgehende Berücksichtigung informatischer Aspekte in einem gesamten Kurs. Als Beispiel für ein derartiges Kurskonzept wird die Lineare Algebra (und Analytische Geometrie) gewählt. Ausgangspunkt der Überlegungen ist das Buch *[Lehmann, E.: Lineare Algebra mit Matrizen und Vektoren, Metzler-Verlag, 1990]*.

Das folgende Inhaltsverzeichnis (Spalte 1) wird in Spalte 2 mit einigen Angaben zum Computereinsatz, insbesondere mit CAS, versehen. Die Angaben ermöglichen eine erste Orientierung und werden durch einige Anmerkungen in Spalte 3 zu informatischen Aspekten ergänzt.

Inhaltsverzeichnis Man beachte die durchgehende Verwendung von Matrizen und die diversen Anwendungsbeispiele aus der Linearen Algebra	**Computereinsatz / Software**	**Informatische Aspekte**
1. Tabellen - Matrizen	Die Eingabe von Matrizen erfolgt bei einem CAS (TI-92-Plus) über einen Tabelleneditor oder in Form von Listen.	Bereits hier erfolgt die erste Begegnung mit informatischen Aspekten, wenn man an die **Datenspeicherung von Tabellen** denkt (zweidimensionale Felder oder Listen). Insbesondere wird später das Problem auftreten, dünn besetzte Matrizen oder Matrizen in spezieller Gestalt platzsparend zu speichern. *Siehe auch Kapitel 3.1.4.*
2 Skalarprodukt, Matrizenmultiplikation 2.1 Materialverflechtung, Marktforschung 2.2 Einige besondere Matrizen 2.3 Matrizen in der Abbildungsgeometrie 2.4 Materialverflechtung, Modellerweiterung 2.5 Gesetze für das Rechnen mit Matrizen	Die Verwendung eines CAS ermöglicht eine Reduzierung des händischen Rechnens. Damit ergibt sich die zusätzliche Möglichkeit des experimentellen Arbeitens durch Parametervariation. Anwendungen führen oft auf besondere Matrizenformen (z. B. Drehstreckmatrizen, stochastische Matrizen, Bandmatrizen, ...), die sich mit dem CAS gut erforschen lassen. – Für abbildungsgeometrische Fragestellungen gibt es zusätzlich zum CAS spezielle Programme, z. B. *ANIMATO,* siehe Kapitel 3.4.5. – Mit dem CAS lassen sich Matrizengesetze etwa für (3,3)-Matrizen leicht nachrechnen.	In diesem Kapitel sind es in erster Linie die diversen **Algorithmen**, die den Bezug zur Informatik herstellen. Beispielsweise lässt sich eine Prozedur für die Matrizenmultiplikation nachprogrammieren. Wie in der linken Spalte nachlesbar, werden Matrizen in diversen wirtschaftlichen Anwendungen verwendet. Sie lassen sich aber auch z. B. in der **Kryptologie** zum Verschlüsseln verwenden (enger Bezug zur Informatik) oder sind wertvoll bei der Arbeit mit magischen Quadraten (siehe Kapitel 3.1). Für den in Kapitel 1 des Lehrgangs definierten Matrizen-Datentyp lassen sich nun diverse Verknüpfungen definieren und entsprechende **Prozeduren** erstellen. So entsteht insgesamt ein **abstrakter Datentyp „Matrix".** Die verschiedentlich angesprochenen **Modellierungsprozesse** sind ein weiteres Bindeglied zwischen Mathematik und Informatik.

Inhaltsverzeichnis	**Computereinsatz / Software**	**Informatische Aspekte**
3 Analytische Geometrie 3.1 Matrizen - Vektoren - Geraden - Ebenen - Linearkombinationen 3.2 Skalarprodukte, Abstands- und Winkelberechnungen	Für die gängigen Objekte der Analytischen Geometrie lassen sich im CAS Bausteine definieren, mit denen dann rationell gearbeitet werden kann. Damit lassen sich die Schnittaufgaben zwischen Geraden, Ebenen usw. auf ein Minimum reduzieren. Auch das Programm ANALYGEO (Kaese-Software) hilft dabei. Andere Objekte können in einem Ray-Tracing-Programm (zum Beispiel POVRAY) betrachtet werden: Kegel, Zylinder, usw.	Die Verwendung eines **Ray-Tracing**-Programms, wie z. B. POVRAY, führt in den Programmierbereich, in dem wieder **Bausteine für diverse Objekte** zum Einsatz kommen. Der **Entwurf foto-realistischer Szenen** ist in erster Linie eine informatische Aufgabe. Weiterhin drängen sich Fragen nach der **Darstellung der Objekte auf dem Bildschirm** auf (hidden-line-Problem, Bresenham-Algorithmus zur Liniendarstellung usw.).
4. Lineare Gleichungssysteme 4.1 Probleme, die auf LGS führen 4.2 Eliminationsverfahren nach Gauß 4.3 Rang einer Matrix, Lösungskriterien für LGS 4.4 Anwendungen linearer Gleichungssysteme, u. a. aus der Analytischen Geometrie 4.5 Homogene und inhomogene LGS 4.6 Probleme bei der Lösung von LGS	Bei linearen Gleichungssystemen empfiehlt sich Computereinsatz in besonderem Maße, da er das lästige Handrechnen vermeiden hilft. Beim CAS sind es u. a. Befehle wie Solve(...) und Rref(), die die Gleichungslehre stark beeinflussen. Andere Möglichkeiten zur Benutzung eines CAS finden sich bei der schrittweisen Ermittlung der Lösungen, siehe *[Lehmann, E: Lineare Algebra mit dem TI-92, Texas Instruments 1999].*	In der Schule wird der **Gauss-Algorithmus** benutzt, ggf. in verschiedenen Varianten. Hier lohnen sich Überlegungen zur Programmierung, insbesondere wie der Computer auf Sonderfälle reagieren soll (beispielsweise Zeilen- oder Spaltentausch). Von Interesse sind dabei auch schlecht-konditionierte Gleichungssysteme und sehr umfangreiche LGS (Anwendung u. a. in der Tomographie, siehe *[Thomas Sonar: Angewandte Mathematik, Modellbildung und Informatik – Vieweg-Verlag, 2001, S. 123-144].*
5. Vektorräume 5.1 Magische Quadrate, Vektorräume 5.2 Lineare Abhängigkeit 5.3 Zeilenrang und Spaltenrang einer Matrix 5.4 Basis, Dimension, Basis-transformation	Ein besonders motivierendes Vektorraum-Modell sind magische Quadrate, siehe auch Kapitel 3.1. Ein CAS dient zum Forschen, Entdecken, Formulieren von Vermutungen und deren Begründung. Dabei erweisen sich wieder CAS-Bausteine mit Parameter als sehr nützlich.	Kapitel 3.1 zeigt die Verbindungen dieses Themas zur Informatik. Die **Arbeit mit Bausteinen** und ihren Parametern ist hier besonders einleuchtend.

Inhaltsverzeichnis	**Computereinsatz / Software**	**Informatische Aspekte**
6. Inverse Matrizen 6.1 Begriff, Berechnung, Sätze 6.2 Stücklistenproblem 6.3 Input-Output-Analyse	Wieder ist ein CAS geeignet, mit dem sich u.a. Modellrechnungen an den genannten Anwendungen durchführen lassen. U. a. treten hier Terme der Form $x = (E - T)^{-1} \cdot y$ auf. Interessant sind hier auch die Auswirkungen der Inversenbildung bei abbildungsgeometrischen Fragestellungen unter Benutzung der oben genannten Software. Mit dem CAS lässt sich z. B. auch die Formel $(A \cdot B)^{-1} = B^{-1} \cdot A^{-1}$ entdecken.	Inverse Matrizen können auf verschiedene Weise berechnet werden. Ein leicht durchzuführender Algorithmus ist z. B. der von Faddejev. Seine Begründung ist allerdings schwieriger. Sie erfolgt über die Eigenwerttheorie und ist daher nur für Leistungskurse möglich, Literatur siehe unten. Ein **Effizienzvergleich der Algorithmen** bietet sich an. *[E.Lehmann: Lineare Algebra mit dem Computer, Teubner-Verlag 1983, S. 57 f. (u. a. Struktogramm zum Algorithmus von Faddejev)]*
7. Matrizenpotenzen, mehrstufige Prozesse 7.1 Maschinenüberwachung, Irrfahrten 7.2 Aus der Populationsdynamik 7.3 Stochastische Matrizen	Das Berechnen von Matrizenpotenzen ist einfach, aber zeitaufwendig. Angesichts der vielen Anwendungen für Matrizenpotenzen kommt ein CAS gerade recht. Für spezielle Matrizen lassen sich für die Potenzen Formeln entwickeln.	Über die Bezüge zur Informatik wird in Kapitel 3.3 ausführlich berichtet.
8. Computereinsatz in der linearen Algebra 8.1 Der "Lineare Algebra-Matrizen-Rechner" MATRIX 8.2 Programmierung mit Hilfe von Prozeduren aus der UNIT M90_U 8.3 Matrizen aus der Sicht der Informatik - ausgewählte Matrizenprozeduren 8.4 Aus der Computergrafik	Heute haben CAS die seinerzeitigen Lösungen verdrängt. Computergrafik wurde bereits oben mehrfach erwähnt.	1990 waren CAS noch nicht verbreitet. So entstand seinerzeit im Informatikunterricht eine **Matrizen-Software**, in der bereits **zahlreiche Bausteine** enthalten waren. Auch heute noch sollte in einem Leistungskurs der eine oder andere Baustein, der im CAS als Black Box vorliegt, analysiert oder sogar nachprogrammiert werden. Siehe Kapitel 3.4.5.

Lange Zeit bestand ein Kurs „Lineare Algebra und Analytische Geometrie" überwiegend aus dem Durchführen langwieriger (und langweiliger) Rechnungen: Lösen linearer Gleichungssysteme, Bearbeiten von sogenannten „Hieb- und Stichaufgaben", Bestätigen von Vektorräumen usw. Heute gewinnt dieser Kurs besondere Attraktivität, z. B., wenn man den oben beschriebenen Aufbau wählt. Abbildung 4.5-b fasst die Ideen zusammen, die sich je nach Schwerpunktsetzung zu verschiedenen Kursabläufen verbinden lassen.

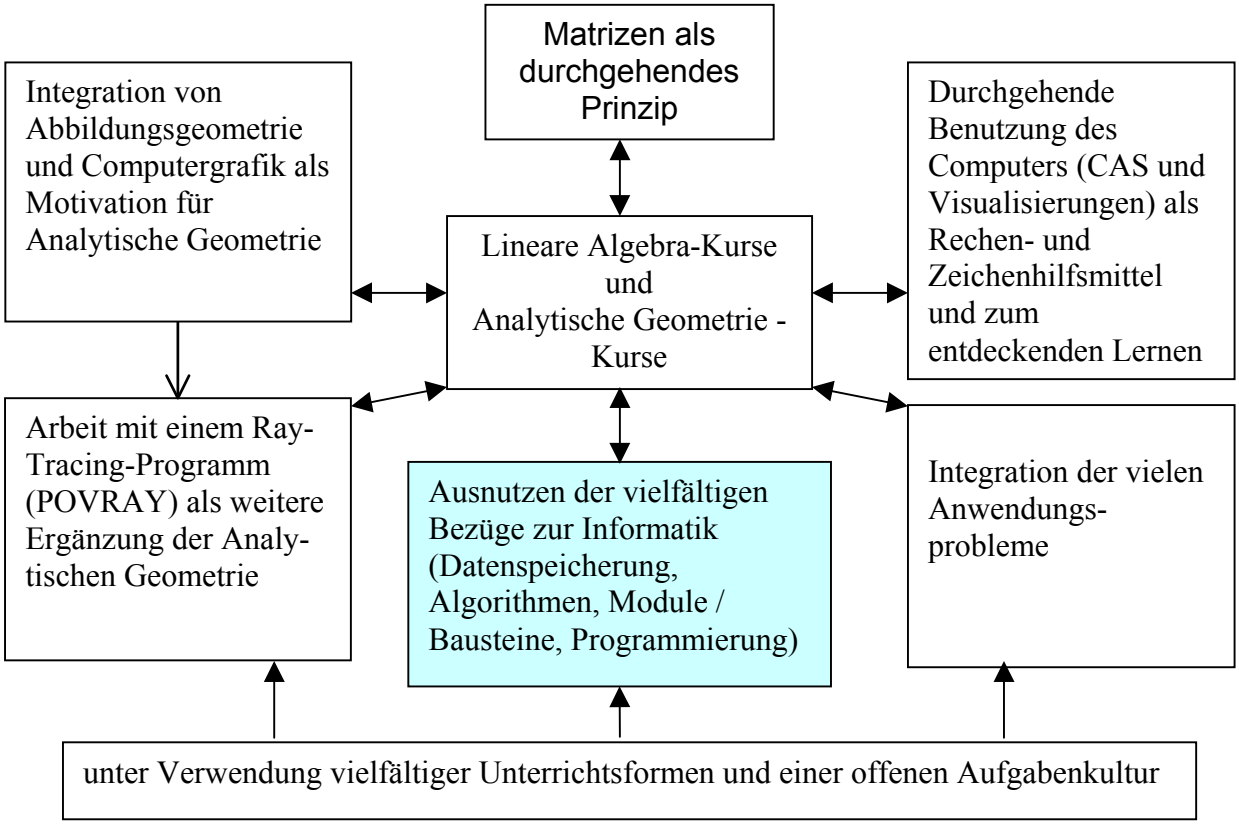


Abb. 4.5-b: Ein Lineare-Algebra-Kurs auf der Basis des Matrizenkalküls als durchgehendem Prinzip

4. Zusammenfassung

Nach der Darstellung der Grundlagen in den Kapiteln 1 und 2 wurden in Kapitel 3 verschiedene Unterrichtsvorschläge mit mathematisch-informatischen Inhalten unterbreitet. Für die konkrete Unterrichtssituation wurden dabei mehrere Ansätze verfolgt. Sie werden auf der linken Seite der folgenden Tabelle aufgelistet. Die rechte Seite der Tabelle verweist auf die vorgelegten Themen und ihren Standort in der Arbeit.

- Mathematik-Unterricht mit informatischen Inhalten und Methoden

Unterrichtssituation Hinweis: Das eine oder andere Thema passt zu mehreren Situationen	**Angebot in der vorliegenden Arbeit**
A Ein gesamter Mathematikkurs wird durchgehend mit Berücksichtigung informatischer Querverbindungen unterrichtet.	A Für diesen Ansatz steht Kapitel 3.5: Lineare Algebra – ein Kurskonzept mit Matrizen, Computereinsatz und informatischen Anteilen *(Kurs Lineare Algebra / Analytische Geometrie)*
B Einige Kursinhalte werden mit informatischen Fragestellungen angereichert	B Für diese Situation werden die folgenden Vorschläge unterbreitet: 3.2 Eine mathematisch-informatische Entdeckungsreise, Teilverhältnisse auf Dreiecksseiten – ein weiteres Projekt für wenige Stunden *(z. B. Klasse 11, rekursiv definierte Folgen)* 3.3.3 Ein Versandproblem – Markow-Ketten *(Kurs Lineare Algebra oder Stochastik)* 3.3.4 Das Crap-Spiel – Markow-Kette und endlicher Automat *(Kurs Lineare Algebra oder Stochastik)* 3.4.2 Magische Quadrate *(z. B. Kurs Lineare Algebra)* 3.4.4 Zufallszahlen – Grundlage für Simulationen *(Stochastikunterricht)* 3.4.1 Ausgewählte mathematische Funktionen der Informatik unter mathematisch-informatischen Aspekten *(Klasse 11 oder Kurs Analysis)* 3.4.5 Einige Elemente der Computergrafik *(u. a. Kurs Lineare Algebra)*

Unterrichtssituation Hinweis: Das ein oder andere Thema passt zu mehreren Situationen	**Angebot in der vorliegenden Arbeit**
C Neue, im Kursplan in der Regel nicht vorgesehene mathematisch-informatische Inhalte. Teilthemen können auch als Übungsaufgaben zu verschiedenen Gebieten aufgefasst werden.	C 3.1 Der (3a+1)-Algorithmus - ein Projekt für wenige Stunden 3.4.3 Mathematische Aspekte aus der Kryptologie 3.4.6 Unerwartetes in Bildern *(Unterrichtliche Verwendung in verschiedenen Gebieten)*
D Mathematisch-informatische Themen für Projekttage oder für zusätzliche Kurse. Teilthemen können auch als Übungsaufgaben zu verschiedenen Gebieten aufgefasst werden. Hinweis: Hier können selbstverständlich auch Themen aus A bis C (eventuelle mehrere) bearbeitet werden.	D 3.3 Zustandsgraphen in Informatik und Mathematik – ein längeres Projekt 3.3.1 Endliche Automaten und Markow-Ketten 3.3.2 Fleißige Biber – das Busy-Beaver-Problem – Turingmaschinen 3.3.3 Ein Versandproblem – Markow-Ketten 3.3.4 Das Crap-Spiel – Markow-Kette und endlicher Automat *(Die unterrichtliche Verwendung erfolgt z. B. in den Kursen „Lineare Algebra" oder „Stochastik" oder in Projekten aus verschiedenen Anlässen.)*

Abb. 4-1: Mathematik-Unterricht mit informatischen Inhalten und Methoden

- Zur didaktisch-methodischen Aufbereitung der oben genannten Themen stehen vielfältige Unterrichtsmethoden zur Verfügung (siehe u. a. Kapitel 1.4.1.2 und 2.1.4), insbesondere bieten sich häufig projektartige Arbeitsformen an.
- Weiterhin empfiehlt es sich, die Leitlinie „Arbeit mit Modulen/Bausteinen" möglichst oft und frühzeitig zu verfolgen.
- Die Ausführungen in dieser Arbeit und insbesondere die oben ausgearbeiteten Unterrichtsbeispiele zeigen, dass kein Mangel an Themen besteht, um mathematische Inhalte mit informatischen Inhalten zu vernetzen.
- Ein größeres Problem dürfte dagegen die häufig fehlende Bereitschaft von Lehrern für ein derartiges Vorgehen sein. Erfahrungen zeigen z. B., dass selbst Informatiklehrer, die sich ja durchaus mit dem Computer auskennen und auch Mathematik unterrichten, den Computer im Mathematikunterricht nicht einsetzen, weil sie von den Möglichkeiten dazu zu wenig wissen.

Unter Bezug auf die Überblicksdarstellungen in Kapitel 1.2.1 und Kapitel 1.5 stellt sich die Ausarbeitung des Konzepts nun wie folgt dar:

- Das erarbeitete Konzept im Überblick

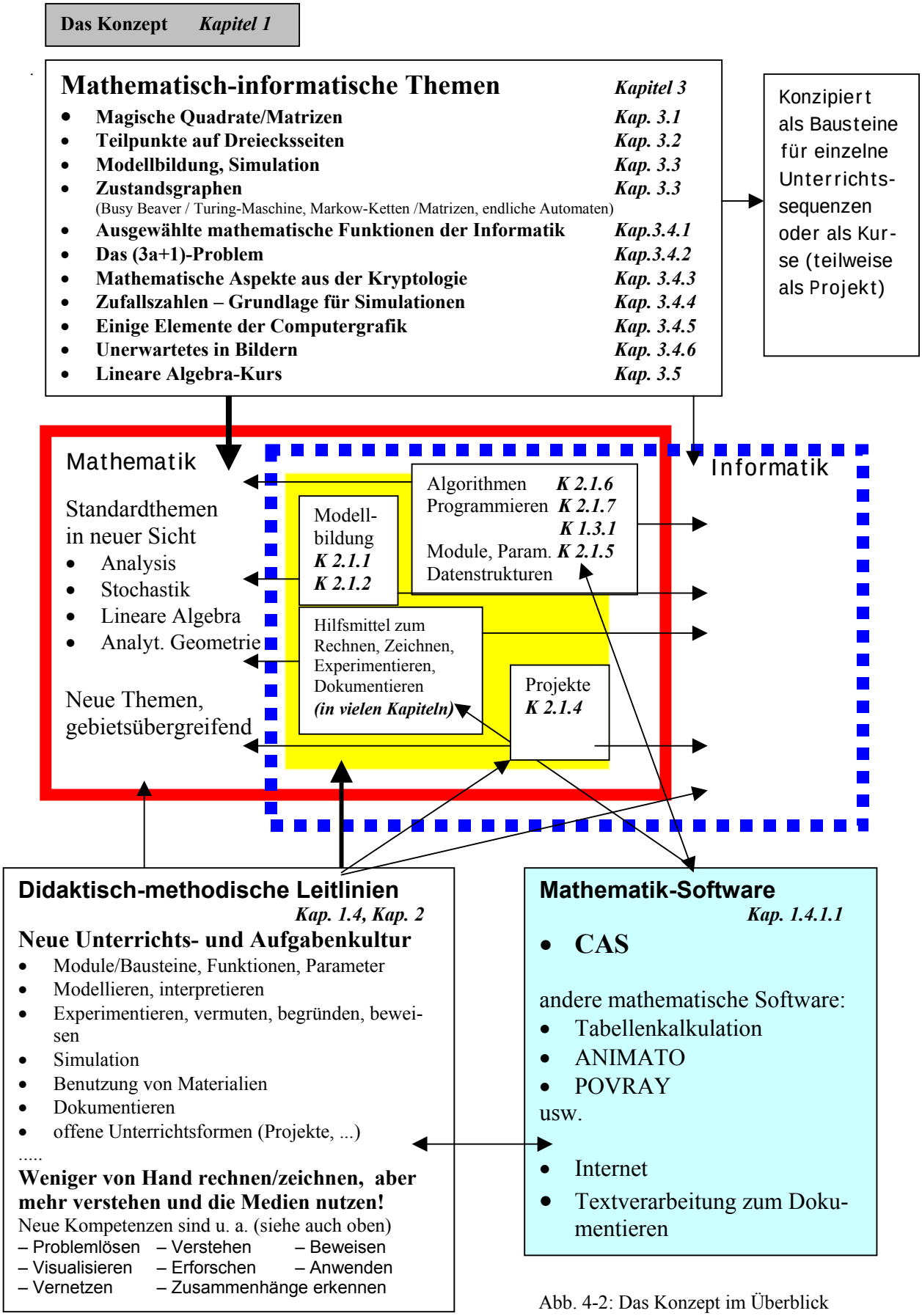


Abb. 4-2: Das Konzept im Überblick

- **Ein Ablaufplan für die Entwicklung von Unterrichtseinheiten für Mathematikunterricht mit Computereinsatz und informatischen Methoden und Inhalten**

  1. Wähle das mathematisches Gebiet aus.

  2. Untersuche die Inhalte des Gebietes auf Ansatzpunkte für informatische Inhalte. Kennzeichen dafür können sein:
  - Das Gebiet ist rechenintensiv und enthält wichtige Algorithmen.
  - Das Gebiet ist datenintensiv.
  - Es werden komplexere Datentypen verwendet.
  - Das Gebiet ist zeichenintensiv. Es kann häufig mit Computergrafik gearbeitet werden.
  - Mathematische Sachverhalte können durch Programmierung verdeutlicht werden.
  - Die verwendeten Funktionen können auch in der Informatik eine Rolle spielen.
  - Auftretende Modelle können simuliert werden.
  - Das Gebiet ist besonders für entdeckendes Arbeiten am Computer geeignet.
  - Für das Gebiet gibt es spezielle (eventuell auch programmierbare) Software.

  3. Wähle passende Vernetzungen für den Unterricht aus.

  4. Wähle aus der Vielfalt der Möglichkeiten jeweils passende Unterrichtsformen aus; beachte und benutze dabei die Schülerkompetenzen – auch aus der Informatik.

  5. Beachte die Bedeutung von Dokumentationen und die Hervorhebung wichtiger Ergebnisse. Vergiss nicht die notwendigen mathematischen Zwischen- und Endzusammenfassungen.

- **Was bringt die Einbeziehung informatischer Methoden und Inhalte in den Mathematikunterricht?**

Über Unterrichtsziele eines neuzeitlichen Mathematikunterrichts wurde bereits in den Kapiteln 1.4 und 1.5 nachgedacht. Was kann man zusätzlich von den fächerübergreifenden Ansätzen aus dem Informatikunterricht erwarten?

Sichtweisen

Es ist bekannt, dass unterschiedliche Sichtweisen auf einen Sachverhalt das Verständnis desselben wesentlich fördern können. Dabei kann die mehr innermathematische Sicht ergänzt werden, beispielsweise durch
- verschiedenartige Darstellung von Algorithmen,
- weitere Visualisierungen,
- Beispiele aus der Informatik.

Arbeitsweisen aus dem Informatikunterricht

Aus dem Informatikunterricht bekannte Arbeitsweisen eignen sich häufig auch für den Mathematikunterricht. So ist z. B. das Programmieren im Informatikunterricht trotz aller vorherigen Planung häufig auch ein Experimentieren und Suchen nach der besten Realisierung – ein Arbeiten mit einer Lösungsidee und der häufig nötigen Abwandlung und Verbesserung der Idee. Auf diese Weise können aber auch manche mathematische Problemstellungen angegangen werden: Experimentieren – vermuten – begründen – beweisen.

Schülerkompetenz durch Computernutzung

Aus dem Informatikunterricht ist der Schüler in der Regel gewohnt, sehr eigenständig am Computer zu arbeiten und Ergebnisse sinnvoll festzuhalten. Hiervon profitiert der Mathematikunterricht, sofern der Lehrer diese Fähigkeiten für seinen Unterricht auch beachtet. Durch die häusliche Arbeit am Computer besitzen die Schüler weitere Kompetenzen: Im Computerhandling und in speziellen anderen Bereichen. Auch diese können für den Mathematikunterricht genutzt werden.

Die meisten der in der vorliegenden Arbeit verwendeten Beispiele sind im Verlauf vieler Jahre unterrichtlich erprobt. Dabei handelte es sich um verschiedene Lerngruppen: Informatikkurse ab Klasse 11, Grundkurse und Leistungskurse der gymnasialen Oberstufe, Wahlpflichtfachkurse Klasse 9 und 10. So beruht mein Erfahrungshintergrund zwar auf insgesamt vielen Schülern, die Sammlung empirischer Daten größeren Umfangs war jedoch angesichts der großen zeitlichen Streuung und der unterschiedlichen Voraussetzungen nicht möglich und wäre auch nicht sinnvoll gewesen. Interessante, gut durchdachte Äußerungen von Schülern meines letzten Leistungskurses (2001) zur Arbeit mit CAS-Bausteinen finden sich in *[Leh02b], S.155–166*. Hieraus ein Zitat des Schülers *Thomas Kolonka*:

„Im Allgemeinen sind Bausteine eine sehr sinnvolle Anwendung, wenn es darum geht, ***ein Problem schnell und immer wieder zu lösen****. Da der Baustein vom Benutzer selbst erarbeitet werden muss und* ***der Baustein in der Regel eine allgemeine Lösung*** *ist (Baustein mit Variablen), ist es eine wunderschöne Übung, um allgemeine Lösungsansätze herauszufinden. Mit einem Baustein lässt es sich wunderbar* ***experimentieren, d.h. mit wenigen Handgriffen kann man verfolgen, wie sich eine Funktion oder Anderes verändern****, wenn man cos(x) statt sin(x) oder* e^x *einsetzt. Hier ist für den Benutzer eine große Möglichkeit gegeben, um das Verhalten von Funktionen zu studieren. Auch die* ***Möglichkeit, zwei Bausteine miteinander zu verbinden****, lässt für den Benutzer eine Reihe von Möglichkeiten offen.*

Ein Nachteil ... ist, dass das ***Rechnen von Hand vernachlässigt*** *werden kann. Auch, dass* ***viele Ergebnisse nicht auf dem Papier*** *stehen könnten, sondern nur auf dem TI, ist eine Gefahr für den Schüler, da er es entweder löschen kann oder nach einigen Monaten die Syntax nicht mehr versteht. Bausteine werden zwar von den Benutzern erarbeitet und eingetippt, aber es kann leicht passieren, dass nach monatelanger Anwendung eines bestimmten Bausteins der Schüler zwar noch weiß, was das Ergebnis zu bedeuten hat und auch die Eingabe ist klar, aber* ***leider gibt ein CAS keine Zwischenergebnisse aus****, so dass die Herleitung an Bedeutung verliert. ... Die Frage,* ***wie das CAS Aufgaben rechnet, ist ein weiteres Problem****, denn der Benutzer sollte es schon wissen, wieso das Ergebnis herauskommt."*

Über den Einsatz von CAS in der Sekundarstufe 1 – ohne besondere informatische Inhalte – habe ich jedoch gerade ein Projekt abgeschlossen, an dem 16 Lehrer und ca. 450 Schüler aus 16 neunten und dann zehnten Klassen (2001 - 2003) beteiligt waren. Jeder Schüler hatte dabei einen Taschencomputer TI-92 ständig zur Verfügung. In der Evaluation werden mehrere statistische Umfragen bei Lehrern und Schülern vorgestellt und ausgewertet, die über die breite Akzeptanz des Taschencomputereinsatzes im Mathematikunterricht informieren, siehe *[Leh02], S. 82–94* .

Die Darstellung endet in Kapitel 6 mit „Zusammenfassung wichtiger Ergebnisse und Empfehlungen für Computer-Projekte im Mathematikunterricht". Die Projektlehrer stellen dabei u. a. einen hohen Arbeitsaufwand für die Lehrer fest. Eine Antwort auf diesen Hinweis, die auch im Rahmen der vorliegenden Dissertation gültig ist, findet sich auf Seite 109 der Evaluation:

- „Hoher Arbeitsaufwand für den Lehrer

Dieser Feststellung kann nicht widersprochen werden. Tatsächlich erfordert ein Unterricht mit neuen Unterrichtsformen, neuen Aufgabenformen und Medieneinsatz wesentlich mehr Aufwand und damit auch Vorbereitungszeit für den Lehrer. Mit wachsender Kompetenz des Lehrers wächst aber auch die Chance eines souveränen Umgangs mit den Gegenständen, besonders dann, wenn der Eigentätigkeit der Schüler und deren Kompetenzzuwachs große Bedeutung beigemessen wird. Der Lehrer kann dadurch immer mehr zum Unterrichtsmanager werden. Er spart Kraft im Unterricht und kann auf die eine oder andere von früher gewohnte Unterrichtsvorbeitung verzichten! ”

Inhaltliche Bereicherung

In Kapitel 3 werden Unterrichtsinhalte genannt, die viele Vernetzungsmöglichkeiten zwischen Mathematik und Informatik aufzeigen. Klassische Inhalte werden durch informatische Aspekte ergänzt, weitere Inhalte kommen ins Blickfeld. Kapitel 2 liefert die Grundlagen für diese Ansätze und stellt methodisches Rüstzeug bereit. Insgesamt ergeben sich wesentliche didaktische und methodische Bereicherungen für den Mathematikunterricht durch die Einbeziehung informatischer Aspekte. Dadurch entstehende zeitliche Ausweitungen können durch Einsparungen insbesondere bei den übertriebenen und unnötigen Rechen- und Zeichenübungen vermieden werden, siehe u. a. *[Her01]*.

Software, Programmierung

Neuzeitlicher Mathematikunterricht benutzt CAS und weitere Software. Im Informatikunterricht ist u. a. das Hinterfragen von Software, beispielsweise von darin enthaltenen Algorithmen, wichtig. Das fördert den verständigen Umgang der Schüler mit Software. Besonders kann der Mathematikunterricht von den Schülerkenntnissen in der im Informatikunterricht verwendeten Programmiersprache profitieren – allerdings nur dann, wenn es dem Lehrer gelingt, diese Kenntnisse sinnvoll (in engen Grenzen, siehe u. a. Kapitel 1.3) in den Mathematikunterricht einzubauen. Die vielen Beispiele haben gezeigt, dass hierbei wegen Arbeitsweise in CAS insbesondere Grundkenntnisse in der funktionalen Programmierung von Bedeutung sind.

Ausblick

Aus Abbildung 4.2 erwächst auch die Fragestellung:

- Was bringt die Einbeziehung mathematischer Methoden und Inhalte in den Informatikunterricht?

Dieser Frage konnte (und sollte) in dieser Arbeit nicht nachgegangen werden. Nachdem anfangs Informatikunterricht stark durch mathematische Probleme bestimmt war, gab es über viele Jahre eine Abkehr davon, häufig zugunsten gesellschaftlicher Fragestellungen. Ältere derartige Probleme, wie z. B. nach der Umstrukturierung von Betrieben und der gesellschaftlichen Relevanz von Computern sind nicht mehr so aktuell. Stattdessen werden u. a. Probleme der Datensicherheit und damit auch das mathematisch-informatische Thema „Kryptologie” anhaltend diskutiert. Auch mit der zunehmenden Einrichtung von Informatik-Leistungskursen (seit kurzer Zeit auch in Berlin) wächst der Anspruch an die fachlichen Grundlagen und der Informatikunterricht wird sich damit auch einer verstärkten Einbeziehung mathematischer Fragestellungen stellen müssen!

Literatur

Hinweise: Die Literaturangaben sind in der üblichen Weise nach Autorennamen geordnet. Für einige Themenbereiche werden die Angaben gesondert ausgewiesen, um dem Leser die Literatursuche zu erleichtern. Das trifft u. a. zu auf die wichtigen Grundlagenthemen „TIMSS und PISA" und auf einige Themen aus Kapitel 3, die ich besonders hervorheben möchte.

- [Ame02] Amelung, Udo u. a. (Hrsg.): Neues Lernen – neue Medien – Blick über den Tellerrand, Tagungsdokumentation Westfälische Wilhelms-Universität Münster, 5.–8. Juni 2001, ZKL-Texte, Münster, 2002

- [Bar99] Bartke, Peter: Funktionale Programmierung, ein aufgabenorientierter Lehr-gang, Kursunterlage Informatik-Workshop MNU / FU Berlin, Version 8.9.1999

- [Bau98] Baumann, R.: Analysis 1, Ein Arbeitsbuch mit Derive, Ernst Klett-Verlag, Stuttgart 1998

- [Bau01] Baumann, R.: Analysis 2, Ein Arbeitsbuch mit Derive, Ernst Klett-Verlag, Stuttgart 2001

- [Ber94] Berendt, G.: Elemente der Kryptologie, in: Schulz, R.-H. (Hrsg.): Mathematische Aspekte der angewandten Informatik, BI, Wissenschaftsverlag, Mannheim 1994

- [Dud93] Duden, Informatik (1993), Lektorat des B.I.-Wissenschaftsverlags unter Leitung von H. Engesser, bearbeitet von Volker Claus und Andreas Schwill, Dudenverlag, Mannheim 1993

- [Eng77] Engel, Arthur: Elementarmathematik vom algorithmischen Standpunkt, Klett-Verlag, Stuttgart 1977

- [Eng91] Engel, Arthur: Mathematisches Experimentieren mit dem PC, Klett-Verlag, Stuttgart 1991

- [Feh94] Fehr, E.: Mathematische Aspekte der Programmiersprachen, in: Schulz, R.-H. (Hrsg.): Mathematische Aspekte der angewandten Informatik, BI Wissenschaftsverlag, Mannheim 1994

- [Gra85] Graf, K.-D. (Hrsg.): Computer in der Schule, Perspektiven für den Mathematikunterricht, Teubner-Verlag, Stuttgart 1985

- [Gra90] Graf, K-D. (Hrsg.): Computer in der Schule 3, Materialien für den Mathematik- und Informatikunterricht, Teubner-Verlag, Stuttgart 1990

- [Hen97] Henn, Hans-Wolfgang: Realitätsnaher Mathematikunterricht mit DERIVE, Dümmler-Verlag, Bonn 1997

- [Her01] Herget,W., Heugl, H., Kutzler, B., Lehmann, E.: Welche handwerklichen Rechenkompetenzen sind im CAS-Zeitalter unverzichtbar? In MNU 2001, Heft 8

- [Her93] Herget, W.: Ziele und Inhalte des Informatikunterrichts – zum Vergleich. In Hischer, H. (Hrsg.): Mathematikunterricht und Computer, neue Ziele oder neue Wege zu alten Zielen? Bericht über die 11. Arbeitstagung des Arbeitskreises "Mathematikunterricht und Informatik" in der Gesellschaft für Didaktik der Mathematik e.V. 1993 in Wolfenbüttel, S.28

- [Heu95] Heugl, Helmut: Computeralgebrasysteme im Mathematikunterricht der Allgemeinbildenden Höheren Schulen (Gymnasien) in Österreich, in Der Mathematikunterricht, 1995, Heft 4 (DERIVE-Projekte im Mathematikunterricht, verantw. Koepf, Wolfram)

- [Iwa94] Iwainsky, A. u. Wilhelmi, W.: Lexikon der Computergrafik und Bildverarbeitung, Verlag Vieweg, Braunschweig 1994

- [Ker85] Kerner, I. O.: Numerische Mathematik mit Kleinstrechnern, VEB Deutscher Verlag der Wissenschaften, Berlin 1985

- [Ker95] Kerner, I. O.: Beitrag von Theorieanteilen der Informatik zur Allgemeinbildung, Teil 1 in PM Praxis der Mathematik 1995, Heft 1, S. 21–25; Teil 2 in PM Praxis der Mathematik 1995, Heft 4, S. 168–175

- [Kli75] Klingen, Laubsch, Neufang, Roth: Informatik (Themenhefte der Mathematik), Klett-Verlag, Stuttgart 1975

- [Koe93] Koepf, W., Ben-Israel, A., Gilbert, R. P.: Mathematik mit DERIVE, Vieweg, Braunschweig/Wiesbaden 1993.

- [Koe94] Koepf, W.: Höhere Analysis mit DERIVE, Vieweg, Braunschweig/Wiesbaden 1994.

- [Koe95] Koepf, W.: „Zur Einführung", S. 3f. und „Was ist 0^0?" S. 65f. – In: MU, Der Mathematikunterricht, DERIVE-Projekte im Mathematikunterricht (Hrsg. Koepf, W.), Heft 4, 1995

- [Koe96] Koepf, W.: DERIVE für den Mathematikunterricht, Vieweg, Braunschweig/Wiesbaden, 1996.

- [Kop94] Koppelberg, S.: Was können Algorithmen? – In Schulz, R.-H. (Hrsg.): Mathematische Aspekte der angewandten Informatik, BI, Wissenschaftsverlag, Mannheim 1994

- [Kut97] Kutzler, Bernhard: Einführung in DERIVE FÜR WINDOWS, Hagenberg, Austria 1997

- [Leh83] Lehmann, E.: Lineare Algebra mit dem Computer, Teubner-Verlag, Stuttgart 1983

- [Leh88] Lehmann, E.: Mathematik-Unterricht mit Computer-Einsatz,
  Band 1: Grundlagen, didaktische und methodische Hinweise für die Sekundarstufen I und II, Dümmler-Verlag 1988
  Band 2: Anwendungen, Unterrichtsbeispiele für die Sekundarstufen I und II, Dümmler-Verlag 1988

- [Leh94a] Lehmann, E. und Schüler: Projektbericht: Potenzen besonderer (2,2)-Matrizen, in MU (Der Mathematikunterricht), Heft 6, 1994

- [Leh94b] Lehmann, E.: Mathematik und Informatik – Konkurrenten oder Partner? In: Hischer, H. & Weiß M. (Hrsg.): Fundamentale Ideen. Zur Zielorientierung eines künftigen Mathematikunterrichts unter Berücksichtigung der Informatik, Bericht über die 12. Arbeitstagung des Arbeitskreises "Mathematikunterricht und Informatik" in der Gesellschaft für Didaktik der Mathematik e.V. 1994 in Wolfenbüttel

- [Leh95] Lehmann, E.: Komplexe Systeme, eine fundamentale Idee im Informatik-Unterricht, in LOGIN 1995, Heft 1

- [Leh96] Lehmann, E.: Komplexe Systeme, Teil 2 – Komplexe Systeme auf Schulniveau reduzieren, in LOGIN 1996, Heft 2

- [Leh98] Lehmann, E.: Wieviel White-Box und wann Black-Box? - Mathematik mit Computeralgebra-Bausteinen des TI-92 - Mathematik in der Schule, Heft 3/1998, Pädagogischer Zeitschriftenverlag Berlin

- [Leh99a] Lehmann, E.: Grundlagen von Projektarbeit, Der Mathematik-Unterricht, 1999, Heft 6

- [Leh99b] Lehmann, E.: Neues Lernen, neue Medien - selbständige Schüler / innen. In Tagungsdokumentation 25.-28.5.1999: Neues Lernen mit neuen Medien – Mathematikunterricht der Zukunft, Westfälische Wilhelms-Universität, ZKL-Texte 8, Münster 1999

- [Leh99c] Lehmann, E.: Mathematik mit Bausteinen und ihren Parametern - PM, Praxis der Mathematik, Heft 3/1999, Aulis-Verlag Köln

- [Leh02a] Lehmann, E.: Berliner CAS-Projekt Sekundarstufe 1 im Rahmen des BLK-Sinus-Projekts, Senatsverwaltung für Bildung, Jugend und Sport, Berlin, August 2002

- [LeH01] Lehmann, Hergen: Animationsplotter ANIMATO, Berlin 2001

- [LeI93] Lehmann, Ingmar: Eulersche Quadrate, in: Mathematik in der Schule 31(1993), S. 612–617

- [LeI96] Lehmann, Ingmar: Zum Lösen von Gleichungen – mit Tafel und Kreide oder Computer? In: Mathematik in der Schule
  Teil 1: Exaktes und näherungsweises Lösen 34(1996) 11, S. 623–632
  Teil 2: Computer-"Fallen" und Fehler 34 (1996) 12, S. 684–696

- [LeI00] Lehmann, Ingmar: Für und Wider von Termumformungen mit einem CAS. In: Herget, W. / Weigand, H.-G. / Weth, T. (Hrsg.): Standardthemen des Mathematikunterrichts in moderner Sicht, Hildesheim, Franzbecker, 2000, S. 67 - 75

- [LeI01] Lehmann, Ingmar: Per Kopf oder Knopf? Rechnen können oder lassen? In: Beiträge zum Mathematikunterricht 2001, Hildesheim, Franzbecker, 2001, S. 376–379

- [Lud97] Ludwig, M.: Projekte im Mathematikunterricht des Gymnasiums, Dissertation, Würzburg 1997

- [MNU86] Empfehlungen zur Gestaltung von Lehrplänen für die informationstechnische Bildung in der Sekundarstufe I bzw. II und für den Computer-Einsatz im Mathematikunterricht der Sekundarstufe II, in MNU 1986, Heft 2.

- [Schup02] Schupp, Hans: Thema mit Variationen, Aufgabenvariation im Mathematikunterricht, Franzbecker-Verlag, Hildesheim 2002

- [Sch87] Schulz, R.-H.: Übersetzen von Nachrichten für die digitale Übertragung – ausgewählte Aspekte der Quellencodierung, in MU (Der Mathematikunterricht), Heft 3 (Codieren und Chiffrieren) /1987

- [Sch94] Schulz, R.-H.: Informations- und Codierungstheorie – eine Einführung, in: Schulz, R.-H. (Hrsg.): Mathematische Aspekte der angewandten Informatik, BI, Wissenschaftsverlag, Mannheim 1994

- [Sch91] Schulz, R.-H.: Codierungstheorie, Eine Einführung, Vieweg-Verlag, Braunschweig / Wiesbaden 1991

- [Schw94] Schwill, A.(1994): Fundamentale Ideen in Mathematik und Informatik. In Hischer, H. & Weiß, M. (Hrsg.): Fundamentale Ideen. Zur Zielorientierung eines künftigen Mathematikunterrichts unter Berücksichtigung der Informatik, Bericht über die 12. Arbeitstagung des Arbeitskreises "Mathematikunterricht und Informatik" in der Gesellschaft für Didaktik der Mathematik e.V. 1994 in Wolfenbüttel

- [Sen73] Der Senator für Schulwesen: Neugestaltung der gymnasialen Oberstufe, Vorläufiges Grundprogramm für das Fach Mathematik, Berlin, 1973

- [Sen85] Der Senator für Schulwesen, Berufsausbildung und Sport: Vorläufiger Rahmenplan für Unterricht und Erziehung in der Berliner Schule, gymnasiale Oberstufe, Fach Informatik, 1985 (der dritte Informatiklehrplan in Berlin)

- [Sen93] Senatsverwaltung für Schule, Berufsbildung und Sport: Vorläufiger Rahmenplan für Unterricht und Erziehung in der Berliner Schule, gymnasiale Oberstufe, Fach Informatik, gültig ab Schuljahr 1993/94 (der vierte Informatiklehrplan in Berlin)

- [Sen95] Senatsverwaltung für Schule, Berufsbildung und Sport: Vorläufiger Rahmenplan für Unterricht und Erziehung in der Berliner Schule, Gymnasiale Oberstufe, Fach Mathematik, Berlin, 1995

- [Sen96] Senatsverwaltung für Schule, Berufsbildung und Sport:
  Vorläufiger Rahmenplan für Unterricht und Erziehung in der Berliner Schule, Klassen 9 und 10, Gymnasium, Wahlpflichtfach Mathematik, 10.4 Kryptologie,Berlin 1996

- [Sin02] Singh, Simon: Codes, die Kunst der Verschlüsselung, die Geschichte, die Geheimnisse, die Tricks, Hanser-Verlag, 2002, 301 Seiten

- [Sob00] Sobich, N.: Variationen des immer Gleichen – Modulmöbel machen den privaten Wohnbereich flexibel und mobil, in DER TAGESSPIEGEL (28.Juli 2000, S.14), Berlin

- [Son01] Sonar, T.: Angewandte Mathematik, Modellbildung und Informatik, Vieweg Verlag, Wiesbaden 2001

- [Tex95] Texas Instruments: TI-92 Handbuch, (1995)

- [Tie97] Tietze, Klika, Wolpers: Mathematikunterricht in der Sekundarstufe II,
  Band 1: Fachdidaktische Grundfragen, Didaktik der Analysis, Friedr. Vieweg & Sohn, Braunschweig 1997

- [Ulm02] Ulm, V.: Pädagogische Schulentwicklung im Mathematikunterricht – Wege zur Umsetzung des Konzepts von H. Klippert. Ein Beitrag zum BLK-Programm „Steigerung der Effizienz des mathematisch-naturwissenschaftlichen Unterrichts" vom Lehrstuhl für Mathematik und ihre Didaktik an der Universität Bayreuth, ohne Jahresangabe (vermutlich 2002)

- [Wei97] Weigand, H.-G.: Veränderungen des Mathematikunterrichts aufgrund des Einflusses der Informatik. In Hischer, H. (Hrsg.): Geometrie und Computer, Bericht über die 15. Arbeitstagung des Arbeitskreises "Mathmeatikunterricht und Informatik" in der Gesellschaft für Didaktik der Mathematik e.V. 1997 in Wolfenbüttel

- [Wel02] Weller, Hubert: Computeralgebra in der Schule – „wie ein Tropfen auf den heißen Stein...", in: Computeralgebra Rundbrief, Oktober 2002, S. 12-15

- [Wet97] Weth, T.: Was bringt der Computer "wirklich" Neues für den Geometrieunterricht? In Hischer, H. (Hrsg.): Geometrie und Computer, Bericht über die 15. Arbeitstagung des Arbeitskreises "Mathematikunterricht und Informatik" in der Gesellschaft für Didaktik der Mathematik e.V. 1997 in Wolfenbüttel

- [Win88] Winter, H.: Divergentes Denken und quadratische Gleichungen, in: mathematik lehren, 1988, Heft 28, S. 54–55

- [Wit98] Witten, H. u. a.: RSA & Co. in der Schule - Moderne Kryptologie, alte Mathematik, raffinierte Protokolle, Teil 1 LOGIN 1998 Heft 3/4, Teil 2 LOGIN 1998, Heft 5

- [Zub01] Zuber, J.: Kryptologie - Ein Wahlthema im Schuljahrgang 13, in LOGIN 2001, Nr.3-4

Spezielle Literatur zu TIMSS und PISA

- [Bau97] Baumert, J. et al: TIMSS – Mathematik-naturwissenschaftlicher Unterricht im internationalen Vergleich. Deskriptive Befunde. Opladen: Leske + Budrich 1997

- [Blu98] Bluhm, W. / Neubrand, M.: TIMSS und der Mathematikunterricht, Informationen, Analysen, Konsequenzen, Schroedel-Verlag, Hannover, 1998

- [Bru02] Bruder, R. u.a.: PISA und kein Ende – oder: Dies ist erst der Anfang, in: Zeitschrift „Mathematiklehren", Friedrich-Verlag, August 2002, Heft 113, S. 64

- [Pis02] Deutsches PISA-Konsortium /Hrsg.): PISA 2000 – Die Länder der Bundesrepublik Deutschland im Vergleich, Leske + Budrich, Opladen 2002

- [Hen99] Henn, H.-W.: Mathematikunterricht im Aufbruch, eine Veröffentlichung des Landesinstituts für Erziehung und Unterricht Stuttgart, Schroedel-Verlag, Hannover 1999

- [Kno02] Knoche, N. u. a.: (Deutsche PISA-Expertengruppe Mathematik, PISA-2000): Die PISA-2000-Studie, einige Ergebnisse und Analysen, in Journal für Mathematik-Didaktik, Heft 3/4, 2002.

Kapitel 2.1.5
Literatur zur Arbeit mit Bausteinen und Parametern

- [Böh02] Böhm, Josef : How I Learned Loving Parameters, in: The Derive-Newsletter #47, Würmla, September 2002

- [Coh93] Cohors-Fresenborg, E. / Kaune,C. / Griep, M.: Vertragswerke über den Umgang mit Zahlen, Handbuch für Lehrer, 2. überarbeitete Auflage, Osnabrück: Forschungsinstitut für Mathematikdidaktik 1993.

- [Coh95] Cohors-Fresenborg, E. / Kaune,C. / Griep, M.: Funktionenlehre Klasse 10, Osnabrück 1995

- [Fil01] Filler, A.: Dreidimensionale Computergrafik als Anwendung der Analytischen Geometrie im Mathematikunterricht der Sekundarstufe II, in: Beiträge zum Mathematikunterricht, Verlag Franzbecker, Hildesheim 2001, S. 181–184

- [Kau95] Kaune, C.: Der Funktionsbegriff als ein Fundament für den gymnasialen MU in der S1. In: Steiner/Vollrath (Hrsg.): Neue problem- und praxisbezogene Ansätze in der mathematik-didaktischen Forschung, Köln 1995

- [Leh96] Lehmann, E.: Lineare Gleichungssysteme. In: Hischer, H. & Weiß M. (Hrsg.): Rechenfertigkeit und Begriffsbildung – zu wesentlichen Aspekten des Mathematikunterrichts vor dem Hintergrund von Computeralgebrasystemen, Bericht über die 13. Arbeitstagung des Arbeitskreises "Mathematikunterricht und Informatik" in der Gesellschaft für Didaktik der Mathematik e.V. 1994 in Wolfenbüttel, Franzbecker-Verlag, Bad Salzdetfurth 1996

- [Leh98a] Lehmann, E.: Wieviel White-Box und wann Black-Box? – Mathematik mit Computeralgebra-Bausteinen des TI-92 – Mathematik in der Schule, Heft 3/1998, Pädagogischer Zeitschriftenverlag Berlin

- [Leh98b] Lehmann, E.: Lineare Algebra mit dem TI-92 (Handreichung mit weitgehender Verfolgung des Bausteinprinzips) - Texas Instruments 1998

- [Leh99a] Lehmann, E.: Mathematikunterricht mit einem Computeralgebrasystem - Analyse des Bausteins Binobau(a,b,n):=(a+b)^n – in: MNU, Der mathematische und naturwissenschaftliche Unterricht, Heft 5, 1999, Dümmler-Verlag Köln

- [Leh99b] Lehmann, E.: Ein Projekt unter Verwendung eines Computeralgebrasystems und Computergrafik - Der CAS-Baustein Trap(a,b,h), in: MU, Der Mathematik-Unterricht, Heft 6, 1999, Projekte im Mathematik-Unterricht

- [Leh02b] Lehmann, E.: Mathematiklehren mit Computeralgebrasystem-Bausteinen, Franzbecker-Verlag, Hildesheim 2002

- [Leh02c] Lehmann, E.: Mathematikunterricht mit Parametern in der Sekundarstufe 1, Schroedel-Verlag, Hannover 2002

- [Mye82] Myers Roy E.: Mikrocomputer Grafik, Addison-Wesley Publishin Company, Amsterdam 1982

- [Schm96] Schmidt, T. / G. / S.: Numerische Verfahren mit dem TI-92, Texas Instruments, Freising 1996

- [Schum01] Schumann, H.: Modulares Arbeiten im Geometrieunterricht, in: MNU, 2001, Heft 6, S. 332–336

Kapitel 3.3
Literatur zu Zustandsgraphen / Übergangsgraphen, Turing-Maschinen

- [Bie79] Biess, G.: Graphentheorie, BSB B.G.Teubner Verlagsgesellschaft, Leipzig 1979

- [Bre95] Brecht, W.: Theoretische Informatik, Vieweg-Verlag, Braunschweig 1995

- [Dew95] Dewdney, A.K.: Der Turing-Omnibus – eine Reise durch die Informatik mit 66 Stationen, Springer-Verlag, Berlin-Heidelberg, 1995

- [Eng76] Engel, A.: Wahrscheinlichkeitsrechnung und Statistik, Band 2, Klett-Studienbücher, Stuttgart 1976

- [Gas92] Gasper, Leiß, Spengler, Stimm: Technische und theoretische Informatik, Bayerischer Schulbuch-Verlag, München 1992

- [Leh99] Lehmann, E.: Die Turingmaschine im Anfangsunterricht – ein Bericht von den ersten Stunden eines Informatikkurses in Klasse 11, in: LOGIN 1999, Heft 6, S. 44–52

- [Nol76] Noltemeier, H.: Graphentheorie (mit Algorithmen und Anwendungen), Walter de Gruyter, Berlin, New York 1976

- [Wer95] Werner u. a.: Taschenbuch der Informatik, Fachbuchverlag Leipzig 1995

Kapitel 3.3.3
Literatur zu Markow-Ketten

- [Hel78] Heller u.a.: Stochastische Systeme, Walter de Gruyter, Berlin-New York 1978

- [Hen78] Hengartner / Theodorescu: Einführung in die Monte-Carlo-Methode, Hansa-Verlag 1978

- [Krü75] Krüger, S.: Simulation – Grundlagen, Techniken, Anwendungen, Walter de Gruyter, Berlin-New York 1975

- [Leh78] Lehmann, E.: Simulation von endlichen Markoff-Ketten, in: Didaktik der Mathematik, 1978, Heft 3, S. 227–242)

- [Leh86a] Lehmann, E.: Markow-Ketten, in: Der Mathematikunterricht, 1986, Heft 5, S. 60–92

- [Leh86b] Lehmann, E.: Fallstudien mit dem Computer, Markow-Ketten und weitere Beispiele aus der Linearen Algebra und Wahrscheinlichkeitsrechnung, Teubner-Verlag, Stuttgart 1986

- [Leh86c] Lehmann, E.: Programmsystem „MARKOW" (für MSDOS-Rechner), Berlin 1986 (neuere Version 1997) in: [Leh86b]

- [Leh97] Lehmann, E.: Wahrscheinlichkeitsrechnung, problemorientierte Unterrichtseinheiten, Verlag Volk und Wissen, Berlin 1997

Kapitel 3.4.6
Literatur zu „Unerwartetes in Bildern"

- [Her90] Herget, W.: Konvergenz-Experimente mit dem Computer?" - in der Zeitschrift „mathematik lehren", 1990, Heft 39, S. 49–55.

- [Her02] Herget, W. / Malitte, E.: Sinus-Schwächen und Rechner-Grenzen, in Exponential- und Winkelfunktionen (Hrsg. Herget, W. / Lehmann, E.), S. 57–64

- [LeI02] Lehmann, Ingmar: „Per Kopf oder Knopf? Rechnen können oder lassen? In: Medien verbreiten Mathematik (Hrsg. Herget, Sommer, Weigand, Weth), Bericht über die 19. Arbeitstagung des Arbeitskreises „Mathematikunterricht und Informatik" in der GdM, 2001 in Dillingen, Franzbecker-Verlag 2002

- [Leh93] Lehmann, E..: Epsilon–Delta, ein neuer Weg zum Verständnis des Grenzwertbegriffs durch Veranschaulichung mit dem Computer, in: Praxis der Mathematik, 1993, Heft 5

- [Pei92] Peitgen, H.-O. u. a.: Bausteine des Chaos, Springer-Verlag 1992, Seite 52 f..

- [Ros02] Rosebrock, S.: Die Folgenmaschine, in: MNU 2002, Jahrgang 55, Heft 7, S. 403