Vorlesungskonzeption „Mathematik-orientierte Computernutzung“

M. Roczen

11. September 2009

Zusammenfassung

Die Vorlesung setzt sich das Ziel, den Umgang mit dem Computer aus Sicht des Mathematikers für (künftige) Mathematiklehrer darzustellen. Beginnend mit dem Grundverständnis für das technische Gerät wird – vom Einfacheren zum Schwierigeren – die Arbeit mit verschiedenen Programmiersprachen und ihre Anwendung auf mathematische Fragestellungen trainiert. Weiterer Schwerpunkt ist die wissenschaftliche Kommunikation und Dokumentation, die im Mathematik- und Informatikunterricht noch an Bedeutung gewinnen dürfte.

Diese Konzeption entspricht weitgehend meiner Vorlesung im Sommersemester 2009.

0 Vorbemerkungen

Die unten folgende Zusammenfassung der Vorlesungsinhalte und Übungsaufgaben kann als Empfehlung für eine Gliederung der Vorlesung angesehen werden. Sie ersetzt nicht die Modulbeschreibung, sondern stellt sich die Aufgabe, diese weiterzuentwickeln; ein Vorschlag zur Neufassung findet sich im Anhang.

Die bisher vorliegende Modulbeschreibung ist so allgemein gehalten, dass Teile eines Informatikstudiums damit gestaltet werden könnten. Das sollte auch so bleiben, damit dem Lesenden eine gewisse Flexibilität und das Eingehen auf aktuelle Entwicklungen erlaubt bleiben. Tatsächlich reicht die verfügbare Zeit aber nur, Grundkompetenzen und Problembewusstsein zu vermitteln: Die Vorlesung sollte deutlich auf anwendungsbereites und für den künftigen Berufsalltag relevantes Wissen ausgerichtet sein. Dabei ist es entscheidend, prinzipielle Möglichkeiten der Rechnernutzung anhand konkreter Beispiele zu erarbeiten und einen Überblick zu verschaffen, der eine sinnvolle Zusammenarbeit auch mit Vertretern anderer Fächer ermöglicht. Mit der Vermittlung einer einzigen höheren Programmiersprache ist es dabei m.E. nicht getan.

1 Motivation: Fragen und Antworten

Ausgangspunkt sollen in vielfältiger Weise vorgetragene Fragen sein, wozu ein Mathematiklehrer eigentlich wissen soll, was in dieser Vorlesung vermittelt wird. Ausreichend bekannt sind derartige Diskussionen bereits von Grundvorlesungen zur Analysis und linearen Algebra, die bei Studienbeginn zu belegen sind.

Während dort die Fragen nach kurzer Zeit verstummen (die betreffenden Teilnehmer sind dann nicht mehr dabei), hat sich im 4. Semester in manchen Fällen ein Selbstbewusstsein entwickelt, das den eigenen Leistungsstand nicht realistisch widerspiegelt: Bereits durch Prüfungsabschluss belegte Kenntnisse etwa der linearen Algebra werden (für immer?) als erledigt angesehen, die „Mathematik-orientierte Computernutzung“ als eine lästige Verpflichtung, die nebenbei noch zu erledigen ist. Es gibt Studierende, die offen aussprechen, dass sie die in der Modulbeschreibung vorgegebene Zeit für Vorlesungsnachbereitung und Übungsaufgaben nicht aufbringen wollen.

Das Fach ist jedoch nicht so nebensächlich, wie es sich in den Vorstellungen mancher Hörer darstellt. Ein zeitgemäßer Mathematikunterricht erlaubt keine Lehrer, die mit dem Computer auf Kriegsfuß stehen. Vielmehr sollte mit Nachdruck darauf hingewirkt werden, dass sich das allgemeine Niveau erhöht.

Mangelhafte Formulierungen bei der Lösung von Übungsaufgaben liegen auf der Linie solcher Probleme, die uns vom Studienanfang an vertraut sind. Diese Vorlesung bietet im größeren Rahmen die letzte Möglichkeit, darauf während des Studiums noch Einfluss zu nehmen.

Während theoretische Grundkenntnisse der Mathematik bereits vom ersten Semester an zu erwerben sind, spreche ich hier noch ein fachspezifisches Problem an, das einige Teilnehmer bis zu 2/3 ihrer Übungszeit kostet – das ist der uneffektive Umgang mit der Tastatur. Ich meine, diese Verzweiflung mit Kollegen zu teilen, die entsprechende Veranstaltungen halten. Bevor darüber nachgedacht wird, ob als letzte Lösung ein vorheriger Nachweis der Qualifikation im *10-Finger-Blindschreiben* zu fordern ist, könnte geprüft werden ob es hilft, diese Fertigkeit lediglich als prinzipielle Anforderung in die Modulbeschreibung aufzunehmen (Gliederungspunkt: Voraussetzungen für die Teilnahme). Die mangelnde Bereitschaft, das Schreiben zu üben, deutet letztlich auch auf ein Motivationsproblem hin.

Die folgenden Zeilen werden hoffentlich von den betroffenen Studierenden zur Kenntnis genommen. Wir kommen zum zentralen Problem, das etwa in der folgenden, kaum verständlichen Meinung gipfelt: „Ich will Mathematiklehrer werden, da brauche ich doch nicht zu wissen, wofür der Informatiklehrer zuständig ist.“

Damit wird tatsächlich eine Vorlesung zur Mathematik-orientierten Computernutzung generell in Frage gestellt. Es zeigt,

- welche Vorstellung von einer künftigen Kooperation mit Kollegen des Fachs „Informatik“ besteht,
- die mangelnde Bereitschaft, später in entsprechenden Entscheidungsgremien fundiert zu einer Meinungsbildung beizutragen, etwa wenn es um den Erwerb von Rechnern und Software für den Unterricht geht,
- mangelnde Motivation für eine kompetente berufliche Nutzung elektronischer Kommunikations- und Dokumentationsmöglichkeiten im Internet,
- dass noch nicht klar geworden ist, welche Anerkennung Mathematiklehrer bei ihren Schülern finden, wenn ihnen begabte, auch recht junge Schüler in der Kenntnis von Betriebssystemen, im Umgang mit Computeralgebrasystemen, bei der Programmierung von Webseiten überlegen sind,

– dass nicht klar wurde, wie bald sich heutige Standardaufgaben aus dem Unterricht von den Schülern mit frei zugänglichen Programmen aus dem Internet erledigen lassen und welche Konsequenzen sich daraus für realistische Leistungsbewertungen ergeben,

– dass nicht erkannt wurde, welche Möglichkeiten der Computer auch mittels selbst programmierter Aufgaben bietet, sinnvolle Leistungskontrollen durchzuführen oder mit Schülern mathematisch interessante Probleme zu diskutieren,

– dass nicht verstanden wurde, wie sehr die schnelle Entwicklung der technischen Möglichkeiten eine das ganze Berufsleben andauernde Herausforderung darstellt, wofür im Studium die Grundlagen zu erarbeiten sind.

Der Computer ist nicht nur ein Gerät, auf dem von Zeit zu Zeit Andere eine neue Software mit „vielen bunten Knöpfen“ installieren – die dann aber nach 3 Jahren unter der neuen Version des Betriebssystems nicht mehr funktioniert, und bei der die Nachfolgeversion wieder „bunte Knöpfe“ hat – leider mit ganz anderen Funktionen.

Was ein mathematischer Beweis ist, wird gerade durch die elektronische Rechentechnik neu abzugrenzen sein: Es stellt sich die Frage, inwiefern ein mit Computer-Unterstützung (z.B. durch Formelmanipulation) „bewiesener“ Satz aus mathematischer Sicht wirklich als bewiesen angesehen werden kann. Geringe Akzeptanz dürfte ein Beweis finden, der auf der Korrektheit des nichtpublizierten Quellcodes eines Systems beruht.

Ein generelles Problem ist die Kontroverse um freie oder kommerzielle Software. „Sachzwänge“, die für eine Anschaffung der letzteren sprechen, sollten nicht allein in der Unkenntnis der Nutzer bestehen. Es kann nicht Aufgabe der Schule sein, bei Schülern Präferenzen zur Nutzung bestimmter kommerzieller Software anzulegen.

Warum Dateiformate, die auch nach Computer-Generationen noch interpretierbar sind? Wer einmal Manuskripte nach 10 Jahren weggeworfen hat, weil keine Software mehr existiert, mit der diese gelesen oder gar bearbeitet werden können, kann es sich vorstellen.

Wer die hier diskutierte Vorlesung besucht hat, sollte sich mit derartigen Fragen hinreichend befasst haben, um qualifizierte Antworten zu geben. In dem Sinne fügt sich das Fach als unverzichtbarer Bestandteil in die Ausbildung von Mathematiklehrern ein. Darüber hinaus können Grundlagen für anspruchsvolle Bachelor-Arbeiten geschaffen werden, bei denen der souveräne Umgang mit dem Rechner sowohl für mathematische Inhalte als auch zur formalen Gestaltung erforderlich ist.

2 Vorlesungsinhalte und Übungsaufgaben

Hier sollte zunächst Einblick in die „Evolution“ des Computers als technisches Gerät sowie für typische Strukturen in Programmiersprachen gegeben werden. Es ist das Verständnis dafür zu wecken, dass es sich dabei um Mathematik handelt, dass es keineswegs überraschend ist, wenn die Computeralgebra fast

hundert Jahre älter ist als der erste funktionstüchtige programmierbare Computer.

Es folgt ein Gliederungsvorschlag für den zeitlichen Ablauf:

- Rechnerarchitektur und Grundverständnis für Möglichkeiten und Grenzen des Computers (Halteproblem); eine maschinennahe Programmiersprache (Registermaschine)
- Betriebssystem (Unix / Linux): Verzeichnisse und Dateien; Arbeiten mit der Shell
- Zahldarstellungen im Rechner
- Anwendung einer höheren Programmiersprache auf Probleme der linearen Algebra und Graphentheorie (hier die leicht erlernbare Skriptsprache des Computeralgebra-Systems Singular)
- Datenstrukturen, Sortieren und Komplexität anhand programmierter Beispiele
- Kommunikation zwischen Computern, das WWW und die Auszeichnungssprache HTML;
- wissenschaftliche Dokumentation: die erweiterbare Auszeichnungssprache XML, einfachste Anwendungen von CSS und XSL; die mathematische Textverarbeitung LaTeX; XHTML
- objektorientierte Programmierung mit Java: Grundkonzepte und ihre Darstellung anhand von konkreten Programmieraufgaben

Eine detaillierte Dokumentation zur Gestaltung der Vorlesung im Sommersemester 2009 entsprechend den angegebenen Schwerpunkten findet sich unter

http://www.math.hu-berlin.de/∼roczen/teaching/2009/coma09.html

im Gliederungspunkt „Stoff der Vorlesung“.

Der Schwerpunkt liegt bei allen Themen auf selbst zu programmierenden Beispielen, die mathematisch überschaubar sein sollten, auch wenn anspruchsvolle Probleme behandelt werden. Der Umfang der Vermittlung einer Programmiersprache sollte jeweils von einem „hello world“-Programm bis zu einer Anwendung reichen, die erkennbar berufsrelevant ist.

Die wöchentlichen Aufgabenserien bestehen in der Regel sowohl aus schriftlichen Aufgaben als auch aus Programmieraufgaben, bei denen die Quelltexte in elektronischer Form vorzulegen sind. Unter Umständen können Themen teilweise im Rahmen von Übungsaufgaben behandelt werden, so dass die Vorlesung entlastet wird.

3 Empfehlungen zur technischen Durchführung

Eine prüfungsähnliche Klausur Mitte des Semesters sollte als Pflichtbestandteil zum Erhalt des Übungsscheins die Semesternoten objektivieren. Nicht selbst

programmierter (korrekter) Code erhält so ein geringeres Gewicht für die Semesterauswertung. Dadurch werden die Hausaufgaben keineswegs abgeschafft, sondern anders gewichtet. Entsprechende personelle Unterstützung der Übung erlaubt es darüber hinaus festzustellen, ob Teilnehmer ihre vorgelegten Programme erklären können und damit wenigstens prinzipiell als Autoren infrage kommen.

Die Veranstaltung aus wöchentlich einer Vorlesung und einer Übung wird z.Z. im großen Hörsaal des Instituts (Vorlesung) und in den Räumen des Computer-Pools (Übung) durchgeführt. Es ist denkbar, auch die Vorlesung in einem Raum durchzuführen, der mit Computer-Arbeitsplätzen in ausreichender Anzahl ausgestattet ist; dadurch wird eine größere Flexibilität bei der Stoffaufteilung zwischen Vorlesung und Übung erreicht: Dies wird offenbar künftig für etwa die Hälfte der Teilnehmer technisch machbar sein. Es bietet sich an, die Vorlesung in zwei parallele Veranstaltungen aufzuteilen, etwa eine fortgeschrittene und eine Anfängergruppe. Die Erfahrung im Sommersemester '09 zeigt, dass dies sinnvoll sein kann, denn das „mittlere" Leistungsniveau war - insbesondere zum Semesterbeginn - kaum vertreten.

Die Abschlussprüfung könnte mit nachhaltigerem Erfolg mündlich durchgeführt werden. Dabei wäre ein Rechner ohne Netzwerkanbindung zu verwenden, auf dem mit einer z.B. 30-minütigen Vorbereitungszeit Programmierkenntnisse nachzuweisen sind. Neben einer präzisen Einschätzung der Leistung dürfte davon auch eine positive „Fernwirkung" ausgehen.

Anhang: Die Modulbeschreibung – ein Änderungsvorschlag

Für den Modul 5 (Mathematik-orientierte Computernutzung) wird folgende Neufassung des Gliederungspunktes „Inhalte" vorgeschlagen:

1. Einführung in die Rechnernutzung.
2. Zahldarstellung und Rechnerarithmetik. Komplementdarstellung ganzer Zahlen, Gleitkommadarstellung, Rechnergenauigkeit, Konsequenzen bei der Realisierung des Gauß-Algorithmus
3. Grundverständnis für Programmiersprachen. Aktives Programmieren in (wenigstens) einer höheren Programmiersprache.
4. Datenstrukturen, Sortieren, Komplexität.
5. Kommunikation und wissenschaftliche Dokumentation. (X)HTML und XML
6. Einführung in wissenschaftliche Software (z. B. Mathematica, LaTeX, ...).
7. Anwendungen in diskreter Mathematik oder linearer Algebra.