Softwareentwicklung in der Wissenschaft

Eine Einführung

Sandra Schröder

Department Informatik
Arbeitsbereich Wissenschaftliches Rechnen
Universität Hamburg

9.12.2010

Seminar: Softwareentwicklung in der Wissenschaft

Gliederung

Ausgangssituation

- Grundlage: Naturwissenschaftliche Fragestellung
- Modellierung in mathematisches Modell mithilfe von Gleichungen
- Interessiert an der (exakten) Lösung der Gleichungen
- Unterstützung durch den Rechner

$\Rightarrow$ Interaktion zwischen Mathematik, Informatik und Naturwissenschaften (MIN)

Definition MIN

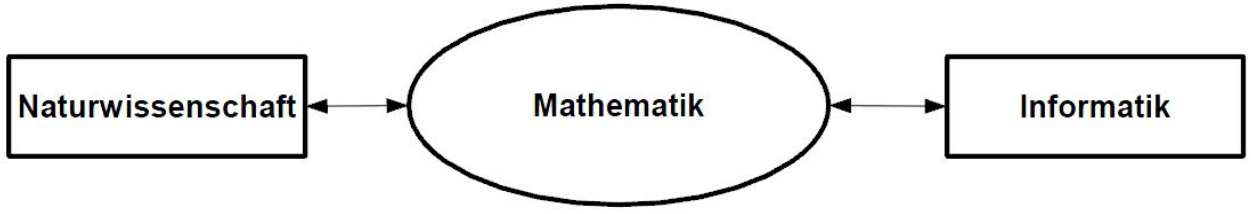

Abbildung: Interaktion

Mathematik Effektives Strukturieren und logisches Begründen von Erkenntnissen

Informatik Darstellung, Speicherung,Übertragung und Verarbeitung von Informationen

Naturwissenschaft Erforschung der belebten und auch unbelebten Natur: Beschreiben, Erfassen, Erklärung (wikipedia.de)

Modellierungs-Schema

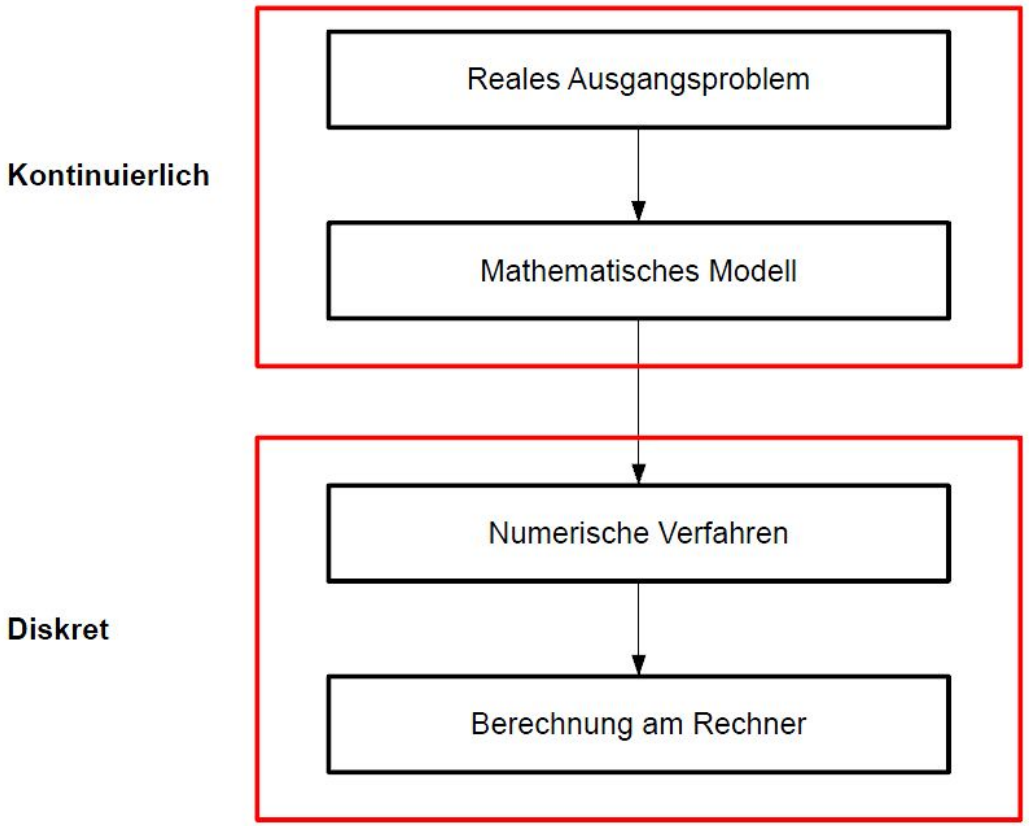

Abbildung: Abtastung

Wissenschaftliche Software - Übersicht

Wissenschaftliche Software

- Was, Warum, Wie?
- Anforderungen an wissenschaftliche Software
- Software Design:
  - Planen
  - Schreiben
  - Testen, Debuggen, Tunen
  - Dokumentieren

Das Dilemma der wissenschaftlichen Softwareentwicklung

- Nicht - wissenschaftliche Software ist “einfach” zu implementieren
- Ein Durchschnittsprogrammierer würde naturwissenschaftliche Zusammenhänge nicht verstehen
- $\Rightarrow$ Zusammenarbeit der Programmierer und Wissenschaftler

Klingt zwar einfach, aber. . .

Kommunikation

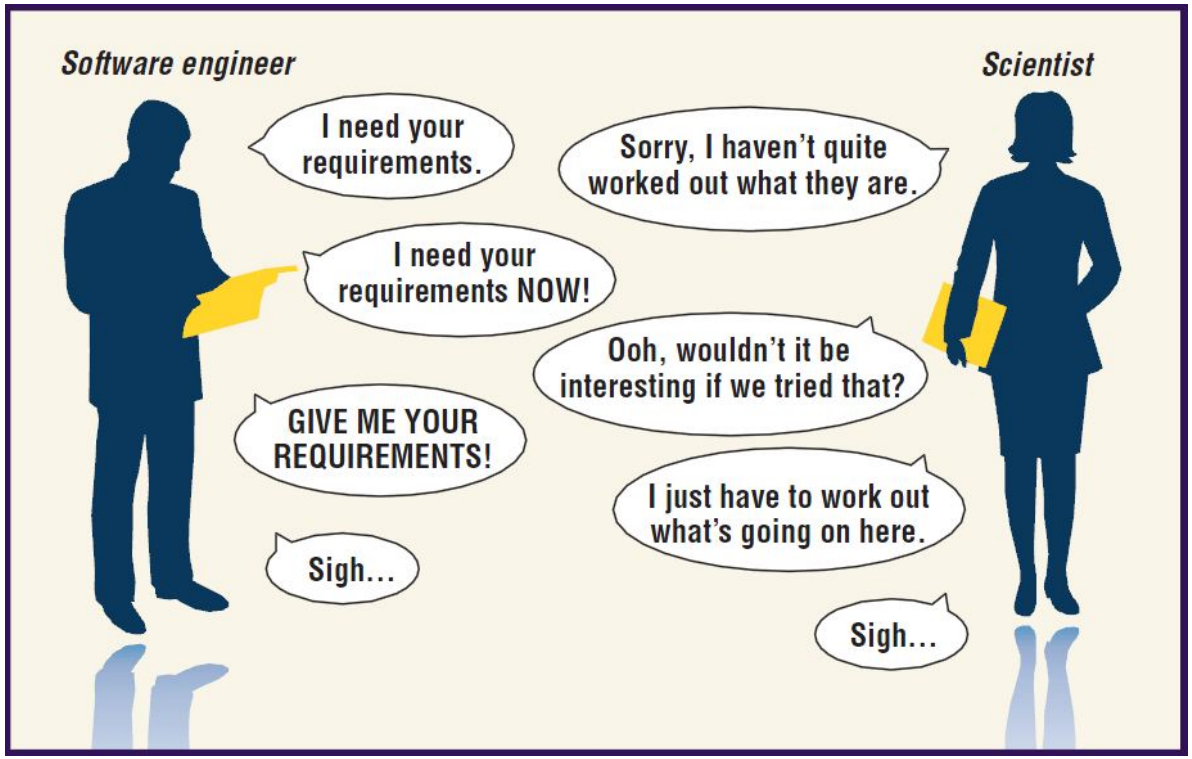

Abbildung: Kommunikation - [Quelle1]

So entwickeln Wissenschaftler ihre eigene Software

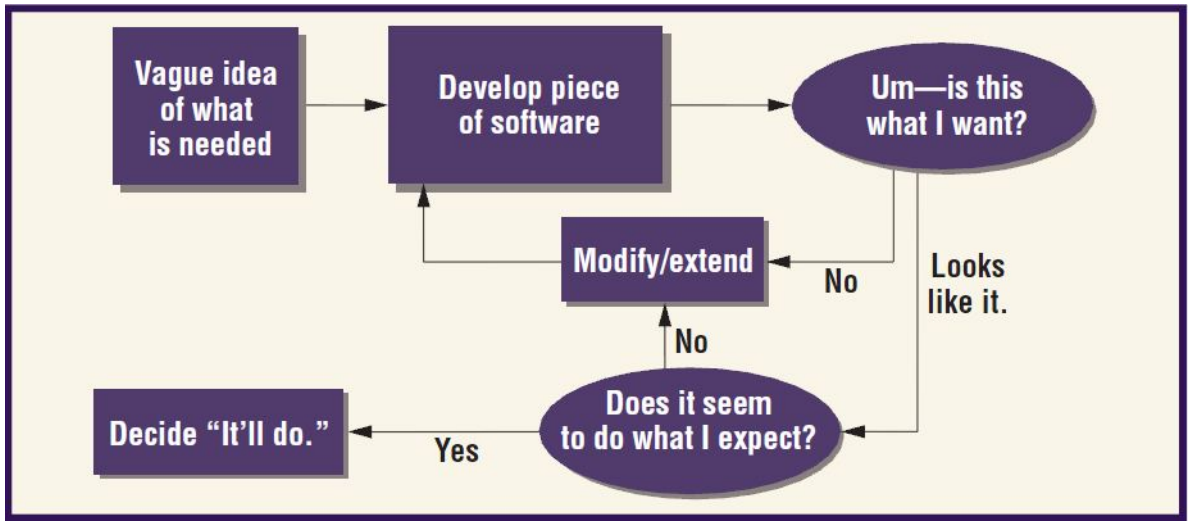

Abbildung: Softwareentwicklung nach naturwissenschaftlicher Art - [Quelle1]

Wissenschaftlich vs. Nicht - wissenschaftlich

- Wissenschaftlich:
  - Berechnungen komplizierter Gleichungen
  - Unterstützung bei Datenauswertung, Mess-, Steuer - und Regelungstechnik
  - Simulation
- Nicht - wissenschaftlich:
  - Kontoverwaltung
  - Hotelbuchung
  - Kino - Manager

Anforderungen an wissenschaftliche Software

- Zuverlässigkeit
  - Korrektheit
  - Robustheit
  - Genauigkeit
- Portabilität
- Effizienz
  - Laufzeit - Effizienz
  - Speicher - Effizienz

Was ist Zuverlässigkeit?

Grad der Wahrscheinlichkeit, mit dem das Programm seine Funktion erfüllt

Korrektheit *Wenn aus Eingabedaten richtige Ausgabedaten erzeugt werden, dann ist das Programm korrekt*

Robustheit *Grad eines Programmes, mit dem es Fehler erkennt, für den Benutzer verständlich reagiert, aber trotzdem seine Funktionsfähigkeit wahrt*

Genauigkeit *Grad der Übereinstimmung zwischen angezeigtem und exakten Wert*

Portabilität

- schnelle Entwicklung der Hardware
- Hardwareunabhängigkeit
- Softwareunabhängigkeit
- Anstreben einer hohen Portabilität

Effizienz

Zwei interessante Betrachtungen bei Effizienz:

- Laufzeit - Effizienz
- Speicher - Effizienz

Effizienz erreichen auf verschiedenen Designebenen:

1. Algorithmen und Datenstrukturen
2. Codeoptimierung
3. Systemsoftware
4. **Hardwareumgebung**

Effizienz und Zuverlässigkeit

- Unzuverlässige effiziente Software ist WERTLOS!
- Höhere Kosten durch unzuverlässige Software
- Einzelne, unzuverlässige Programmteile können Entwicklungsaufwand stark erhöhen

Effizienz ist notwendig!

Software Design - Das generelle Vorgehen

1. Planung
   - Herstellen einer Struktur
   - Bibliotheken
   - Programmiersprache(n) wählen
   - Portabilität sicherstellen
2. Schreiben
3. Debug
   - GNU - Debugger und Valgrind
   - Hardware: Joint Test Action Group
4. Test
5. Profiling
6. Dokumentation

Profiling

- Laufzeitverhalten analysieren
- “Performance - Tuning”
- Geschwindigkeit, Speicherausnutzung, Nebenläufigkeit (modern)
- Es existieren verschiedene Analysetechniken, u.a.
  - Statistische Auswertung
  - Instrumentierung
- Beispiel: gprof

Last But Not Least: Die Dokumentation

- Kommentare
- Testdokumentation
- Dokumentation der Entwicklung
  - Aufgabenstellung
  - Aufgabenlösung
  - Ablaufplan und Vorgehensmodell
- Dokumentation der Programmfunktionen
  - für den Benutzer
  - Verwendung des Softwareprodukts
  - mögliche Fehlermeldung und ihre Behandlung

Beispiele für Bibliotheken

Builtin C/C++ Bibliothek

- `stdio.h`
- `math.h`
- `stdlib.h`
- `time.h`

GNU Scientific Library

- Komplexe Zahlen Arithmetik
- Vektor und Matrizen Rechnung
- Interpolation, Differentiation, Integration
- Schnelle Fouriertransformation

Weitere Bibliotheken:

- Numerical Recipes

Programmiersprachen

- Highlevel:
  - C
  - FORTRAN
  - C++
- Skriptsprachen
  - Perl
  - python
  - shell
  - R
- Symbolische Sprachen
  - Matlab
  - Maple
  - Mathematica
- Graphische Sprachen
  - LabView

Compiler

- 2 Phasen werden durchlaufen:
  1. Analysephase
     - Scannen
     - Syntaxanalyse
     - semantische Analyse
  2. Synthesephase
     - Zwischencodeerzeugung
     - Programmoptimierung
     - Codegenerierung
- Beispiel : GNU Compiler Collection

Hardware - Übersicht

Hardware

- Effizienz auf Hardwareebene
- Speicherhierarchie und Lokalität

Effizienz auf Hardwareebene

Effizienz ist eine der wichtigsten Anforderungen an wissenschaftliche Software!

Wie erreicht man Effizienz auf Hardwareebene?

Interessant: *Speichereffizienz*

Speicherhierarchie

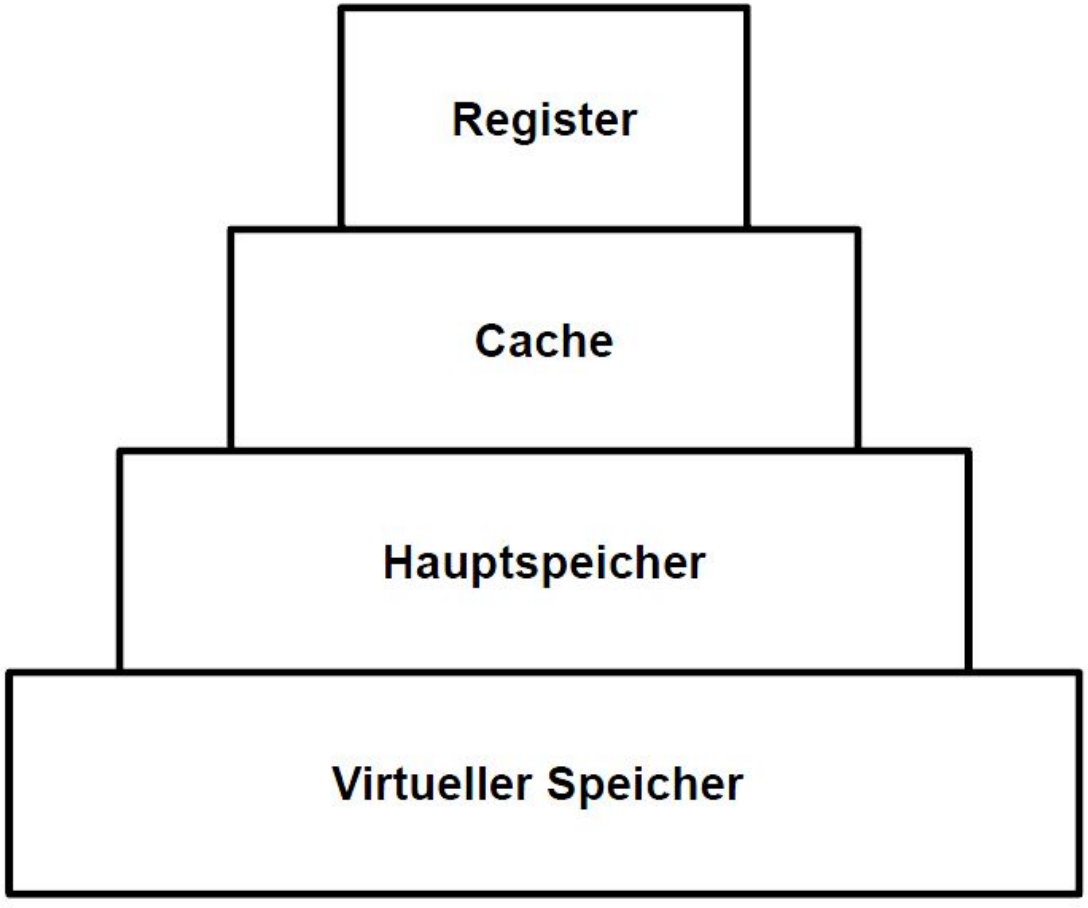

Abbildung: Speicherhierarchie

Rechnerarithmetik - Übersicht

Rechnerarithmetik

- Maschinenzahlen
- Exkurs - Stellenwertsysteme
- Gleitkommazahlen
- IEEE - Standard
- Rundung und Rundungsfehler

Rechnerarithmetik - Maschinenzahlen

Motivation:

- Reelle Zahlen sind im Rechner nicht exakt darstellbar
- Wissenschaftliche Berechnungen sind aber auf reelle Zahlen angewiesen
- Darstellung der reellen Zahlen als Maschinenzahlen

Definition (Maschinenzahlen)

Eine endliche Menge $\mathbb{M} \subset \mathbb{R}$ heißt die Menge der Maschinenzahlen.

Kleiner Exkurs - Wie war das nochmal?

Grundlage

- Gegeben: Menge mit b Symbolen $\mathbb{B} = \{0, 1, \ldots, b-1\}$
  - b ist die Basis/Grundzahl
  - bijektive Zuordnung der Ziffern
- Ziel: Mit Symbolen beliebig große Zahlen darstellen
- Wert einer Ziffer wird bestimmt durch ihre Position in der Zahl $\Rightarrow$ Stellenwertsystem

Die b-adische Entwicklung

Für natürliche Zahlen:

Definition

$$S_i = \sum_{i=0}^{n} a_i \cdot b^i = a_0 + a_1 \cdot b^1 + \ldots + a_n \cdot b^n$$

Für ganze Zahlen:

Definition

$$S_i = \pm \sum_{i=0}^{n} a_i \cdot b^i = \pm(a_0 + a_1 \cdot b^1 + \ldots + a_n \cdot b^n)$$

Für rationale Zahlen:

Definition

$$S_i = \pm \sum_{i=0}^{n} a_i \cdot b^i \pm \sum_{i=-\infty}^{-1} a_i \cdot b^i$$

wobei b die Basis der Zahl und a eine Ziffer aus dem Zeichenvorrat bezeichnet.

Und was ist mit reellen Zahlen?

Idee: Numerisch reelle Zahlen
$\Rightarrow$ Gleitkommazahlen

Darstellung mit:

- Vorzeichen
- Basis
- Mantisse
- Exponent

Erste Annäherung für eine Darstellung:

$$Z = Vorzeichen \cdot Mantisse \cdot Basis^{Exponent}$$

Eindeutigkeit

- Darstellung ist nicht eindeutig, z.B.:
  - $123 \cdot 10^0 = 12.3 \cdot 10^1 = 1.23 \cdot 10^2 = \ldots$

$\Rightarrow$ Normalisierung der Zahl:

Definition (Normalisierung)

Eine Zahl in der Darstellung

$$Z = \mathit{Vorzeichen} \cdot \mathit{Mantisse} \cdot \mathit{Basis}^{\mathit{Exponent}}$$

ist normiert, wenn gilt:

$$1 \leq \mathit{Mantisse} < \mathit{Basis}$$

Der IEEE - Standard

- Computer verwenden das Binärsystem : Basis b = 2
- Genauigkeit von 32 Bit (einfach)
- Genauigkeit von 64 Bit (doppelt)
- IEEE 754 ist bekanntestes Gleitkommasystem

	Mantisse	Exponent
Single	23 Bit	8 Bit
Double	52 Bit	11 Bit

Tabelle: IEEE 754 - Standard

- Mantisse ist normiert
- “Hidden Bit”
- Exponent in Exzess/Bias - Darstellung

Schlussbetrachtung

Erweiterung der Definition der Maschinenzahlen:

Definition (Maschinenzahlen II)

Eine reelle Zahl x wird im Computer als Gleitkommazahl $float(x) \in \mathbb{M}$ dargestellt in der Form:

$$float(x) = (-1)^s \cdot (0.a_1 a_2 \dots a_n) \cdot 2^e, a_1 \neq 0, s \in \{0, 1\}$$

Sie kann als 4 - Tupel beschrieben werden:

$$\mathbb{M}(2, n, MIN, MAX)$$

wobei n die Anzahl der Nachkommastellen der Mantisse und MIN und MAX die minimale/maximale darstellbare Zahl bezeichnen.

Rundung

Definition (Rundung)

Rundung ist eine Abbildung $rd : \mathbb{R} \to \mathbb{M}$ mit $\mathbb{M} \subset \mathbb{R}$, wenn die folgenden Bedingungen

$$rd(m) = m, \forall m \in \mathbb{M}$$
und
$$r \leq s \Rightarrow rd(r) \leq rd(s), \forall r, s \in \mathbb{R}$$

Rundungsfehler

Runden führt zu Rundungsfehlern

- absoluter Fehler: $rd(x) - x$
- relativer Fehler: $\frac{rd(x)-x}{x}$

Abschätzung für den relativen Fehler (ohne Beweis):

$$\frac{rd(x)-x}{x} \leq \sigma := \frac{1}{2} b^{1-t}$$

Maschinenepsilon

Definition

Sei $x \in \mathbb{R}$. Die kleinste positive Zahl c, die man zu $x = 1.0$ hinzuaddieren kann, sodass

$$rd(1.0 + c) > 1.0$$

gilt, heißt Maschinenepsilon $\varepsilon_{machine}$ und ist ein Maß für die Rechengenauigkeit.

Numerik - Übersicht

Numerik

- Was ist Numerik?
- Fehlerbewertungsmechanismen
  - Kondition
  - Stabilität
  - Konsistenz

Was ist Numerik?

- Teilgebiet der Mathematik
- Approximation
- Fehlerabschätzung - und Minimierung
- Algorithmen, ihre Bewertung und Optimierung
- Berechnung mithilfe des Computers

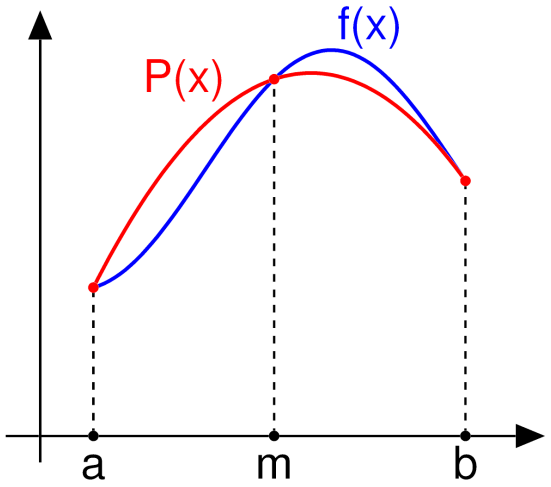

Abbildung: Numerische Integration mit Simpson - [Quelle2]

Fehlerabschätzung

Grundlegende Fehlerbewertungsmechanismen:

- **Kondition**
  - eines Problems
- **Stabilität**
  - eines Algorithmus
- **Konsistenz**
  - eines Algorithmus

Kondition eines Problems

Wie wirken sich Störungen in den Eingabedaten auf das Resultat unabhängig vom gewählten Algorithmus aus?

Gegeben: $f : D \to W$ mit $x \in D$

Verfälschtes x : $\tilde{x}$

Verfälschtes $f(x)$: $f(\tilde{x})$

Datenfehlereffekt: $|f(x) - f(\tilde{x})|$

- Kondition $\cong$ Empfindlichkeit der Lösung
- Gute Kondition
- Schlechte Kondition

Stabilität eines Algorithmus

Stabilität: Wie robust ist ein Algorithmus gegenüber von (kleinen) Störungen in den Eingabedaten?

- Abschätzen des Fehlers:

$$|f(\tilde{x}) - \tilde{f}(\tilde{x})|$$

- Vorwärtsanalyse
- Rückwärtsanalyse
- *Kondition und Stabilität hängen miteinander zusammen!*
- Dies lässt sich mathematisch zeigen...

Konsistenz eines Algorithmus

Konsistenz: Bearbeitet der Algorithmus tatsächlich das Problem oder ein anderes?

Gegeben:

- Kontinuierliche Problemstellung und dessen exakte Lösung
- Numerische Lösung
- Schrittweite

$\Rightarrow$ Ein Verfahren heißt konsistent, wenn es zu jedem Zeitpunkt eine Fehlerbeschränkung in Abhängigkeit von der gewählten Schrittweite gibt. (aus wikipedia.de/Konsistenz)

Zusammenfassung

- Die Informatik unterstützt Wissenschaften große Mengen von Informationen zu strukturieren, zu verarbeiten und aufzubereiten
- An wissenschaftliche Software gibt es unterschiedliche Anforderungen, wie z.B. Effizienz
- Verständnis über die Hardwareumgebung kann helfen, Programme effizienter zu gestalten
- Im Rechner werden reelle Zahlen als Gleitkommazahlen abgespeichert
- Oft muss gerundet werden, dabei entstehen Rundungsfehler
- Die Numerik bietet Fehlerbewertungsmechanismen, wie z.B. Kondition, Stabilität und Konsistenz

Diskussion/Fazit und Ausblick

Fazit

- Insgesamt: Sehr interessantes Thema
- Schwierig: Über (fast) alle Themen einen Überblick geben, ohne zuviel zu erzählen
- Numerik ist interessant, in einigen Bereichen jedoch schwer zu verstehen, z.B. die Fehlertheorie
- Hardwareteil ist eher klein ausgefallen

Ausblick

- Nicht geschafft: Hardware der Rechnerarithmetik - Wie erreicht man Bitgenauigkeit? $\Rightarrow$ Evtl. eigenes Seminarthema?
- Wie kann man die Zusammenarbeit zwischen Durchschnittsprogrammieren und Wissenschaftlern verbessern? Gibt es ein geeignetes Vorgehensmodell?

Danke für Eure Aufmerksamkeit!

Quellen

Bücher

Suley Oliveira, David Stewart (2006)
Writing Scientific Software - A Guide To Good Style
Cambridge University Press

Thomas Huckle, Stefan Schneider (2002)
Numerische Methoden
Springer, 2. Auflage

Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman
Compiler Prinzipen, Techniken und Werkzeuge
Pearson Studium, 2. Auflage

Quellen (2)

Internetseiten und Paper

Helmut G. Katzgraber
Scientific Software Engineering In A Nutshell
http://arxiv.org/PS_cache/arxiv/pdf/0905/0905.1628v2.pdf

Zahlendarstellungen
http://www.math.tu-berlin.de/CoMa/skript/Skript-I-Java/zahlendarstellungen.pdf

Numerische Software
http://www.dorn.org/uni/sls/kap05/e00.htm

Judith Segal, Chris Morris
Developing Scientific Software
www.computer.org/csdl

Quellen(3)

Bildquellen
[Quelle1]:
Judith Segal, Chris Morris
Developing Scientific Software
www.computer.org/csdl

[Quelle2]:
Simpson-Verfahren
http://www.sjmp.de/java/numerisches-integrieren-mit-dem-simpson-verfahren/