Wissenschaftliches Rechnen: Alles ist Approximation

oder:

'geschickt falsch' und 'richtig falsch' rechnen als Paradigma des Scientific Computing

Winfried Auzinger

Institut für Analysis und Scientific Computing
Technische Universität Wien
www.math.tuwien.ac.at/∼winfried

Woche der freien Bildung
Resselpark, Wien, 18. Mai 2010

Überblick

Einleitung: ‘Alles ist Zahl’ am Digitalcomputer

Ganze Zahlen (`integer`)

- Ganze Zahlen am Computer (Digitalrechner) intern in festen Format dargestellt – nicht dezimal sondern *binär*:
- Es gibt nur zwei Ziffern (= bits): `0` und `1`.
- Beispiel:

  dezimal `123` = $1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$ = ...

  ... = binär `1111011` =

  $1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$
- Standardformat: z.B. 32 bit – maximal etwa 10-stellige Dezimalzahlen darstellbar.
- $+, -, *$ eingeschränkt auf diesem Bereich exakt – darüber hinaus nicht durchführbar.
- Genügt für einfache Zahlenrechnungen, Buchhaltung, etc. ...

Einleitung: ‘Alles ist Zahl’ am Digitalcomputer

Exakte Arithmetik

- ¿ Kann man am Computer *exakt* rechnen ?
  - *Im Prinzip ja* – nur dauert es unendlich lange und benötigt unendlich viel Speicher.
- Kleine Einschränkung: Mit *rationalen* Zahlen (Brüchen) kann man in endlicher Zeit und mit endlichem Speicheraufwand exakt rechnen, z.B.

$$\frac{1231}{98098} \cdot \frac{98877}{6565} = \frac{121717587}{644013370}$$

- Das Problem: Die Brüche werden immer komplizierter – für viele Rechenoperationen steigt Rechen- und Speicheraufwand extrem schnell an. Man kann keine festen Zahlenformate verwenden.
- Wurzeln, π, etc. sind rational zu approximieren.

Einleitung: ‘Alles ist Zahl’ am Digitalcomputer

Gleitpunkt-Arithmetik

- ¿ Ist es *notwendig*, am Computer exakt zu rechnen ?
  - *Im Prinzip nein* – aber im Parteiprogramm steht ‘ja’.
- ¿ Was tut der Klassenfeind ?
  - Er verwendet Gleitpunktarithmetik – ein durch und durch reaktionäres Konzept.
- ¿ Ist das *exakt* ?
  - *Im Prinzip nein* – aber die Numeriker behaupten, das ist O.K.
- ¿ Hat der Parteitag Beschlüsse gefasst ?
  - *Im Prinzip ja* – nur kümmert sich niemand darum. Alle verwenden Gleitpunktarithmetik.

Einleitung: ‘Alles ist Zahl’ am Digitalcomputer

Gleitpunkt-Arithmetik als Approximation

- ¿ Was heißt überhaupt *exakt* ?
- Jede praktische Berechnung geht von *Daten* aus (Messwerte, Materialparameter, Naturkonstanten)
- ‘Exakte’ Daten hat man so gut wie nie. Wozu mit inexakten Daten exakt rechnen?
- *Gleitpunktarithmetik* bedeutet:
  - Verwende ein festes Zahlenformat
  - Von jeder Zahl wird nur eine gewisse Anzahl von Stellen berücksichtigt. Z.B.:

$$x = \frac{2142714234234}{7698798765} = 278.31799474680775085826002882931568610755862508$$

wird ersetzt durch

$$\tilde{x} = 278.3179947468078 = 0.2783179947468078 \cdot 10^3$$

  - Arithmetik ist nicht exakt: Jedes Einzeloperation wird auf gewähltes Format *gerundet*. Abgesehen von Spezialfällen ist jede einzelne Basisoperation eine (sehr genaue) *Approximation*.

Einleitung: 'Alles ist Zahl' am Digitalcomputer

Gleitpunkt-Arithmetik: Industriestandard (`double`)

- Ca. 16 Dezimalstellen: Industriestandard gemäß ANSI/IEEE 754-1985; IEC 60559-1989
  … *double precision*, 64 bit Binärcodierung
- 'Stellen' und 'Exponent' (d.h. die Position des Dezimalpunktes) werden separat gespeichert
- Sehr, sehr kleine und sehr, sehr große Zahlen nicht darstellbar (Grenze ca. bei $10^{\pm 300}$)
- Auf modernen Computerchips: Arithmetik ($+, -, *, /, \sqrt{\cdot}, \ldots$) sind 'fest verdrahtete' Instruktionen (Maschinenbefehle)
  ($/$ und $\sqrt{\cdot}$ beruhen auf Approximationsverfahren!)
- Kompliziertere mathematische Funktionen ($\sin, \cos, \exp, \ldots$) sind als Softwarebausteine in Anwendungsprogramme integriert – auf Basis von analytisch/numerischen Approximationsalgorithmen.

Einleitung: 'Alles ist Zahl' am Digitalcomputer

Gleitpunkt-Arithmetik: Anmerkungen

- Frage: Ich habe mir heimlich einen Computer mit Gleitpunktarithmetik besorgt. Er kann in einer Sekunde eine Milliarde Rechenoperationen ausführen. Ich hab's probiert. Ist das Ergebnis O.K. ?
  - *Im Prinzip ja* – es sei denn, das Gegenteil trifft zu.
- ¿ Ist das eine Frage der Politik ?
  - Im Prinzip nein –
es sei denn, Sie halten Bildungspolitik für wichtig.
- ¿ Können Sie mich genauer instruieren?
  - *Im Prinzip ja* – aber wir haben jetzt Dienstschluss, und die Ministerin ist anderweitig beschäftigt.

Archimedes und die Berechnung von $\pi = 3.14159\ldots$

Wie war das nochmal mit der Zahl π ?

ˇ1

- Die Zahl

$$\pi = 3.141592653589793\ldots$$

  entspricht der Fläche bzw. dem halben Umfang des Kreises mit Radius 1.

  - π: ∞ viele Dezimalstellen (*transzendente Zahl*)
  - 16 Dezimalstellen $\approx$ verfügbare Rechengenauigkeit auf handelsüblichem Computer (*double*)
  - Fehler dieser Approximation ca. im Bereich von Milliardsteln von Milliardsteln (irrelevant für übliche praktische Anwendungen)

- π kann nicht exakt als endliche Dezimal- bzw. Binärzahl angegeben werden. Dies gilt aber bereits für die meisten rationalen Zahlen, z.B.: $1/3 = 0.3333333333333333\ldots$

Archimedes und die Berechnung von $\pi = 3.14159\ldots$

Eine Approximationsformel

- ¿Wie erhält man 16 Dezimalstellen von π (oder wieviele auch immer)?
- …viele Möglichkeiten. Auf **Archimedes von Syrakus** (287–212 v. Chr.) geht eine geometrische Konstruktion zurück (Beispiel für Vorläufer der modernen Integralrechnung):
- *Schreibe dem Kreis ein regelmäßiges 4-Eck, 8-Eck, 16-Eck, etc. ein, berechne die Flächen und verfeinere je nach gewünschter Genauigkeit*.
- Für die Flächen π_n der 2^n - Ecke gibt Archimedes eine *Rekursionsformel* an ($\pi_2 = 2\sqrt{2}$):

$$\pi_{n+1} = 2^n \sqrt{2\left(1 - \sqrt{1 - \left(\frac{\pi_n}{2^n}\right)^2}\right)}, \quad n = 2, 3, \ldots$$

Archimedes und die Berechnung von $\pi = 3.14159\ldots$

Anmerkungen zur Approximationsformel

- ...Das generiert die richtigen Flächenformeln
  Z.B. Fläche des 128-Ecks ($n = 7$) :

$$\pi_7 = 64\sqrt{2-\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2}}}}}} \approx 3.1412\ldots$$

  (= π auf 4 Stellen genau). Beachte die Asymptotik dieser Formel für $n \to \infty$ (eine der vielen bekannten asymptotischen Darstellungen für π).
- Dabei sind sind Wurzeln auszuwerten – geht nicht exakt. Für Archimedes war das Handarbeit; für den PC ist es Routine (Näherungsalgorithmus fest am Chip 'verdrahtet').
- Die Methode ist relativ 'langsam' (gute Genauigkeit erfordert eine sehr feine Abtastung des Kreises).

Archimedes und die Berechnung von $\pi = 3.14159\ldots$

Durchführung am Computer

- …Rechnung am PC mit 16-stelliger `double`-Arithmetik in jedem einzelnen Rechenschritt (inklusive den Wurzelauswertungen):
- …

```
n=15: 3.141592654807589
...
n=20: 3.141596553704820
...
n=25: 3.142451272494134
...
n=30: 0.000000000000000
```

- …9 richtige Dezimalstellen für $n = 15$; danach sukzessiver Genauigkeitsverlust bis hin zu $0.00\ldots$ Kein Programmierfehler!

Archimedes und die Berechnung von $\pi = 3.14159\ldots$

Kommentar

- *Also:* Genauigkeit des Endresultates *nicht automatisch garantiert*, obwohl jeder einzelne Schritt auf 16 Dezimalstellen korrekt gerechnet (mit je einem sehr kleinen Rundungsfehler).
- Hätten wir π_{30} sehr genau ausgewertet (d.h. mit sehr hoher Rechengenauigkeit – das geht nur mit spezieller Computeralgebra-Software und ist vergleichsweise ‘langsam’ bzw. ‘teuer’), hätten wir knapp 16 richtige Dezimalstellen von π erhalten.
- *Folgerung:* Ein – im Prinzip korrekter – Algorithmus kann am Computer total scheitern.
- Es gibt auch viele Beispiele für *exakte* Algorithmen (im Gegensatz zu Approximationsformeln wie bei Archimedes), die am Computer genauso scheitern. Beispiel folgt.

Archimedes und die Berechnung von $\pi = 3.14159\ldots$

Kann man das reparieren ?

- Archimedische Formel lässt sich (mathematisch äquivalent) umformen – dann funktioniert es perfekt.
- Wie und warum? – das lernt in den LVAs über Numerische Mathematik.
- Einige Schritte der Archimedes-Approximation umfassen nur ‘eine Handvoll’ Rechenoperationen.

  Zum Vergleich: Lösung eines Systems von $1{,}000$ Gleichungen in $1{,}000$ Unbekannten dauert am PC weit unter 1 Sekunde. Dabei werden mehr als $300{,}000$ Rechenoperationen ausgeführt – ¿ Endgenauigkeit ?
- Wie und warum das O.K. geht – auch das lernt man bei uns.

Das leidige Integral

Beispiel: Exakte Lösungsformel unbrauchbar

- Wir alle wissen:

$$\int_a^b \cos x \, dx = \sin b - \sin a$$

- Beispiel (in `double`-Arithmetik: $a = 1$, $b = 1 + \varepsilon \approx a$ ($\varepsilon \approx 10^{-10}$, b exakte `double`-Zahl)
- $\rightsquigarrow$
  `sin b - sin a =`
  `5.403022473871033`$\cdot 10^{-11}$
- ... nur 6 statt 16 richtige Stellen: 'Auslöschung'
- $\sin$ wurde korrekt ausgewertet (gerundet)!
- Einen *korrekten* `double`-Wert für das Integral erhält man durch geeignete *Approximation* der Fläche unter dem Integranden.
- **Also**: Die exakte Formel ist hier unbrauchbar. **Man muss approximieren** für ein korrektes Resultat.

Das kriechende Pendel

Einfachste mathematische Beschreibung

- Ein Anwendungsproblem:

  Berechne den Schwingsverlauf eines gedämpften ('abgebremsten') Pendels; Auslenkung (klein) und Drehimpuls am Anfang vorgegeben (Zeitpunkt $t = 0$)

- Mathematische Beschreibung ('Kraft = Masse × Beschleunigung') führt auf *Differentialgleichung* für den gesuchten Auslenkungswinkel $\varphi = \varphi(t)$ (z.B. Zeit t in s):

$$\ddot{\varphi}(t) + \rho\,\dot{\varphi}(t) + \omega^2\varphi(t) = 0$$

- $\dot{\varphi}(t)$: Winkelgeschwindigkeit

  $\ddot{\varphi}(t)$: Winkelbeschleunigung

  ρ charakterisiert Intensität der Dämpfung (gegeben)

  ω ergibt sich aus Länge und Masse des Pendels (= Eigenfrequenz bei Abwesenheit von Dämpfung)

Das kriechende Pendel

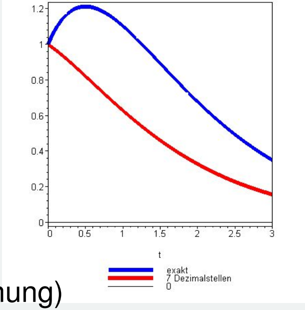
Exakte Lösung

- Je nach Intensität der Dämpfung:
  'Schwingen' oder 'Kriechen'
- Wir betrachten 'Kriechen': Für $\rho > 2\,\omega \;\rightsquigarrow\;$ Lösung
  $$\varphi(t) = c_1 e^{\lambda_1 t} + c_2 e^{\lambda_2 t}$$
- c_1, c_2 errechnen sich aus Anfangszustand
- 'Abklingparameter' λ_1, λ_2 :
  $$\lambda_{1,2} = -\frac{\rho}{2} \pm \sqrt{\frac{\rho^2}{4} - \omega^2} < 0$$
- $e^x = \exp(x) \ \ldots$ *Exponentialfunktion*
  (abklingend für $x < 0$)
- Anmerkung: 'Schwingen' wäre Kombination aus Exponentialtermen und periodischem Anteil (dargestellt mittels $\sin, \cos$)

Das kriechende Pendel

Lösung am Computer: Zahlenbeispiel

- Anmerkung: $\rho = 2\,\omega$ ist Grenzfall
  (andere Lösungsformel)
- Wähle $\rho = 2$ und $\omega = 1 - \varepsilon,\ 0 < \varepsilon \ll 1$
  (Kriechen, nahe am Grenzfall)
- Lösung ($\epsilon = 10^{-14}$), 'exakt' (sehr genau) und in single precision (ca. 7 Dezimalstellen) ausgewertet:



rot: qualitativ falsch (Auslöschung)

Das kriechende Pendel

Lösung am Computer: Folgerungen

- Eine mathematisch korrekt in ein Computerprogramm umgesetzte Lösungsformel garantiert nicht immer eine genaue Lösung.
- Es ist ‘zusätzliches Know-how’ zu investieren, etwa Fallunterscheidungen im Code.
- Manchmal helfen rein algebraische Umformungen (‘vgl. ‘Archimedes’). Im allgemeinen jedoch gilt

Satz (Auzinger, 2010)

Numerisch umsetzbare und verlässliche Lösungen für (insbesondere analytische, aber auch viele algebraische) Probleme erhält man nur mittels *Approximation.*

- Man *muss* sozusagen falsch rechnen. Aber wir rechnen ohnehin immer ‘falsch’ (siehe oben).
- Man muss *‘richtig falsch rechnen’*, d.h. numerisch korrekt und stabil approximieren.

Das kriechende Pendel

Lösung am Computer: Wie ist das mit ***Artificial Intelligence*** *?*

- Stand der Technik: Lösungsalgorithmen sind im Detail genau zu programmieren, mit verschiedensten Fallunterscheidungen je nach Problem, Problemgröße und -struktur, Hardware, ...
- Fallunterscheidung, Reagieren auf *exceptions* usw.
- Was braucht man dazu? $\rightsquigarrow$ Kompetenz in mathematischen Methoden, Numerischen Algorithmen, Programmierung.
- Später vielleicht (?) Der Rechner reagiert 'selbständig' auf Ausnahmesituationen und ergreift die richtige Maßnahme ... bedeutet Programmieren auf höherer Ebene.
- Beispiel: Moderne mathematische Software ist teilweise auch heute schon so so programmiert, dass sie viele Fallunterscheidungen automatisch richtig vornimmt.

Die Exponentialfunktion

oder: von der getakteten zur ‘infinitesimalen’ Verzinsung

- ¿ Was ist das eigentlich:
$$\exp(x) = e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \ldots \quad ?$$
- Nehme $E = E(0) = 1$ EUR und trage ihn auf die Bank, zu $100\,\%$ Verzinsung.
- Nach einem Jahr: Verzinstes Kapital
$E(1) = 1 \cdot (1 + 1) = 2$ EUR.
- Das Kapital liegt bleibt 1 Jahr lang ‘eingefroren’, dann wird verzinst.
- Der Zinssatz ist mir zu niedrig; einen höheren genehmigt mir die Bank nicht.
- Ich fordere jedoch:
Aufzinsung jedes halbe Jahr (mit halbem Zinssatz):
$$E(\tfrac{1}{2}) = \big(1 + \tfrac{1}{2}\big)E(0) = 1.5$$
$$E(1) = \big(1 + \tfrac{1}{2}\big)E(\tfrac{1}{2}) = \big(1 + \tfrac{1}{2}\big)^2 E(0) = 2.225$$

Die Exponentialfunktion

Denke Verzinsungsintervall beliebig klein

- Ich fordere: Aufzinsung jeden Monat (1/12 Zinssatz):

$$E(1) = \left(1 + \tfrac{1}{12}\right)^{12} E(0) \approx 2.613$$

- Denke Verzinsungsintervall beliebig klein: n mal verzinsen mit Zinsatz $1/n \cdot 100\,\%$; $n \to \infty$ $\rightsquigarrow$

$$E(1) = \lim_{n\to\infty} \left(1 + \tfrac{1}{n}\right)^n E(0) = 1+1+1/2+1/6+\ldots = e$$

mit $e = 2.7182818284590452353602874 71\ldots$ (Euler'sche Zahl).

- allgemein: 'kontinuierliche Verzinsung' eines Anfangskapitals $E(0)$ über allgemeinen Zeitraum x, mit Zinssatz $100\,p\,\%$:

$$E_p(x) = \lim_{n\to\infty} \underbrace{\left(1 + \frac{p\,x}{n}\right)^n}_{= 1+\frac{(p\,x)^2}{2}+\frac{(p\,x)^3}{6}+\ldots\ =\ e^{p\,x}} \cdot E(0)$$

Die Exponentialfunktion

Approximation von e^x

- e^x ist unendliche Reihe (bzw. unendliches Produkt); e^x mit $x < 0$ entspricht Abklingen statt Aufklingen
- Algebraische approximation mittels Abbruch für endliches n: O.K., aber nur brauchbar für kleine x
- Approximation nur *lokal* z.B. für $x = 0 \ldots 1$
- Fehler berechenbar (bzw. abschätzbar) – Analysis!
- Rückrechnung für beliebige x unter Ausnützung der Eigenschaften der Exponentialfunktion.
- ¿ 'lokale Genauigkeit O.K.' $\Rightarrow$ 'globale Genauigkeit O.K.' ?

  …erfordert *numerische Stabilität* der Rückrechnung.

  …ähnlich wie bei Pendel: kritische Sonderfälle.
- Lokale Approximation kann noch wesentlich effizienter ('billiger') erfolgen – ein Thema der *Approximationstheorie*.

Exponentialfunktion und Pendel

Beispiel für sinnvolle Abstraktion in der Mathematik

- Betrachte Winkel $\varphi = \varphi(t)$ aus Pendelbeispiel
- Fasse φ und $\dot{\varphi}$ zu einem Objekt zusammen ('Vektor'): $\Phi = (\varphi, \dot{\varphi})$
- $\rightsquigarrow$ Pendelgleichung in 'Vektorform':

  $$\dot{\Phi}(t) = P \cdot \Phi(t)$$

  mit einer 'Abbildung' (Matrix) P,
  die das Bewegungsgesetz repräsentiert. Lösung:

  $$\Phi(t) = e^{t\,P} \cdot \Phi(0)$$

  mit *matrix exponential* $e^{t\,P} = I + P + \frac{P \cdot P}{2} + \ldots$
- Approximation von $e^{t\,P}$ (ähnlich wie oben angedeutet für e^x) liefert *perfekte numerische Lösung*.
- *Hier:* Übergang zu abstrakterer Formulierung plus kompetente Approximation löst das Problem.

Approximation als Sparmaßnahme

Wenn der Rechenaufwand aus dem Ruder läuft . . .

- Beispiel: Berechnung einer dreidimensionalen Temperaturverteilung $T(t; x)$ unter gegebenen atmosphärischen Bedingungen (t = Zeit, x = Raumvariable); $\rightsquigarrow$ *partielle Differentialgleichung* für Wärmeleitungsprozess.
- Lösung unter vereinfachten Annahmen:

$$T(t; x) = e^{t P} \cdot T(0; x)$$

  mit noch abstrakter definiertem P und $e^{t P}$.
- 'Runterkochen' auf numerischen Algorithmus (per Approximation, insbesondere *Diskretisierung* der Raum- und Zeitvariablen) führt auf eine Folge von Standardproblemen.
- Das Standardproblem: Löse (z.B.) ein System von $N = 1{,}000{,}000$ linearen Gleichungen in $N = 1{,}000{,}000$ Unbekannten (in jedem Zeitschritt!).

Approximation als Sparmaßnahme

Ein Lösungsalgorithmus

- Das lineare Gleichungssystem: $M\,u = c$

  M ... Koeffizientenmatrix (gegeben, $N \times N$)

  c ... gegebener Vektor (Länge N)

  u ... *gesuchter* Vektor (Länge N)
- Wärmeleitungsproblem: M hat spezielle Struktur (*positiv definit*), und M ist 'dünn besetzt'.
- Algorithmus (*Hestenes, Lanczos, Stiefel,* ca. 1950):

Conjugate Gradient Alg. (CG):

$u_0 := 0;\ d_0 := r_0 := c;$

for $i = 0, 1, \ldots$ **do**

$\alpha_i := (r_i \cdot r_i) \,/\, (M\,d_i \cdot d_i);$

$u_{i+1} := u_i + \alpha_i\, d_i$

$r_{i+1} := r_i - \alpha_i\, M\,d_i;$

$\beta_i := (r_{i+1} \cdot r_{i+1}) \,/\, (r_i \cdot r_i);$

$d_{i+1} := r_{i+1} + \beta_i\, d_i;$

end do

Approximation als Sparmaßnahme

CG-Iteration ist konvergent

- CG: Beispiel dafür, wie die Existenz des Digitalrechners 'neue Mathematik generiert' hat
- CG ist ein exakter Algorithmus: liefert exakte Lösung nach N Schritten (abgesehen von Rundung).
- CG ist ein *konvergenter* Algorithmus:

  – Vernünftige Approximation auch bereits bei vorzeitiger Beendigung (nach $m \ll N$ Schritten)!

  – Ein Maß für die erzielte Genauigkeit wird in jedem Schritt mitgeliefert (Residuum).
- CG ist ein *selbstkorrigierender* Algorithmus:

  Rundungsfehler verzögern lediglich die Konvergenz, aber zerstören sie nicht.
- Schritte problemlos vektorisier- bzw. parallelisierbar.

Approximation als Sparmaßnahme

CG-Iteration: Nachteile und Optimierungspotential. Der 'Heilige Gral'

- Algorithmus 'nimmt an', dass vorhergehende Schritte exakt ausgeführt wurden.
  - Dies ist nicht der Fall (Rundung)
  - Wird jedoch durch 'Selbstkorrektur' entschärft.
- CG: Konvergenz *beliebig langsam* für 'hinreichend schwierige', realistische Probleme (typisch: $N \to \infty$)
- 'Heiliger Gral': Will Problem der Dimension N mittels $\mathcal{O}(N)$ bzw. Rechenoperationen lösen
  - Weniger geht nicht (auf jede Variable muss mindestens einmal zugegriffen werden)
  - Erfordert *Konvergenzgeschwindigkeit unabhängig von* N – ist für CG normalerweise nicht gegeben.
- *Abhilfe:* Löse ein modifiziertes Problem: Verschiedenste Techniken zur 'Vorskalierung' der Gleichungen. Erfordert präzises Verständnis der mathematischen Struktur der Aufgabenstellung.

Anhang

Eine Anekdote: Ariane 5, Release 1

- Was so alles passieren kann:
  Der Absturz der ersten **Ariane 5** der ESA.
- Ursache: Programmierfehler
  (eher: Schlamperei bzw. Versehen)
- Zitat:
  The internal SRI software exception was caused during execution of a data conversion from 64 bit floating point to a 16 bit signed integer value. The floating point number which was converted had a value greater than what could be represented by a 16 bit signed integer.*
- Was ist passiert: Eine unzulässige Datenkonversion im internen Kontrollsystem (integer overflow) führte zu einer *exception* und damit zum Absturz der Software und damit auch der Rakete.

Anhang

Anmerkungen zur Didaktik und Praxis der Mathematik

- Unverzichtbare Grundlagen:

  Lineare Algebra, Analysis, Geometrie, diskrete Mathematik, samt ihrer wichtigsten Algorithmen
- Numerische Methoden nur auf dieser Basis verstehbar.
- Formeln und Rechnen sind wichtig. Aber Formeln sind nur die 'Syntax' der Mathematik. Wichtiger:
- – Inhaltliches, soweit möglich anschauliches Denken
  – Abstraktion ist wertvoll, um allgemeine Strukturen zu erkennen, aber kontraproduktiv, wenn man statt über das Kaninchen nur mehr über den Hut redet ...
  – Denken in Analogien und Anwendungen (Bezug zur 'physikalischen' Realität)
  – Lebenslanges, berufsbegleitendes Lernen;
  Neugier, Interesse (an Neuem und Alten!);
  Demut gepaart mit Frustrationstoleranz und Hartnäckigkeit
- Kompetenter Einsatz des Computers: Der Rechner ist das 'Labor' des angewandten Mathematikers. Ein gesundes Maß an Misstrauen ist jedoch angebracht.
- *Kreatives Denken* – was ist das? ...ein schönes Thema ...