Computerorientierte Mathematik I

9. Vorlesung

Carsten Gräser

Freie Universität Berlin

13.12.2019

Rückblick: Vorlesung vom 06.12.2019

Freie Universität Berlin

Stabilitätsabschätzungen

Auswertungsbäume zur systematischen Stabilitätsabschätzung

- Auswertungsbaum: Knoten, gerichtete Kanten, Wurzel, Blätter
- Zerlegung in Teilbäume
- Von den Blättern zur Wurzel:
  rekursive Funktionsauswertung und Stabilitätsabschätzung
- Theoretische Grundlagen: Satz 7.6 und Satz 7.9
- Beispiele

Beispiel

Problem:

$$f(x_1, x_2) = (x_1 - x_2)^2 = (x_1^2 - 2(x_1 x_2)) + x_2^2$$

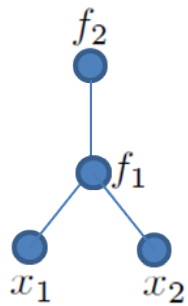
Algorithmus 1: $f(x_1, x_2) = f_2(f_1(x_1, x_2))$

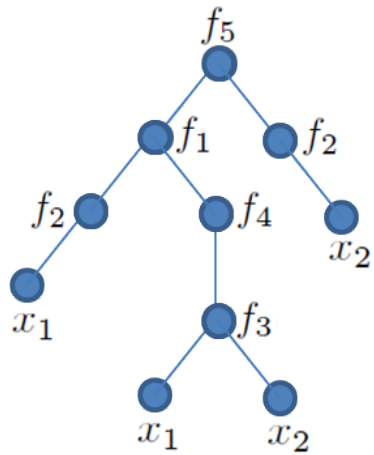
$$f_1(x, y) = x - y, \qquad f_2(x) = x^2$$

Algorithmus 2: $f(x_1, x_2) = f_5(f_1(f_2(x_1), f_4(f_3(x_1, x_2))), f_2(x_2))$

$$f_1(x, y) = x - y, \quad f_2(x) = x^2, \quad f_3(x, y) = xy, \quad f_4(x) = 2x, \quad f_5(x, y) = x + y$$

Algorithmus 1





Stabilitätsabschätzung (für $x_1 = 10^{11}$, $x_2 = 10^{11} - 1$)

- Teilbäume: $\beta_{21} = [\,], \beta_{22} = [\,], \beta_1 = [\beta_{21}, \beta_{22}], \beta_0 = [\beta_1]$
- Aus Satz 7.6 und 7.9: $\sigma \leq 1 + \kappa(f^{\beta_1}, z^{\beta_1})\sigma^{\beta_1}$
- Berechnung:

$$z^{\beta_1} = f_1(x_1, x_2) = 1, \quad \kappa(f^{\beta_1}, z^{\beta_1}) = \kappa_{\text{rel}}(f_2, 1) = \frac{|f_2'(1)||1|}{|f_2(1)|} = 2, \quad \sigma^{\beta_1} = 1$$

- Ergebnis: $\sigma = 1 + 2 \cdot 1 = 3$
- Keine Fehlerverstärkung durch innerste Funktionsauswertung f_1

Fehlerkonzepte

- Stabilität: Auswirkung inexakter Auswertung ohne Eingabefehler
- Kondition: Verstärkung von Eingabefehlern bei exakter Auswertung
- Gesamtfehler: Eingabefehlern und inexakte Auswertung

Fehlerkonzepte

- Stabilität: Auswirkung inexakter Auswertung ohne Eingabefehler
- Kondition: Verstärkung von Eingabefehlern bei exakter Auswertung
- Gesamtfehler: Eingabefehlern und inexakte Auswertung

Satz 7.5: Gesamtfehler $= \kappa_{\text{rel}} \cdot$ Eingabefehler $+ \sigma_{\text{rel}} \cdot$ Auswertungsfehler!

Fehlerkonzepte

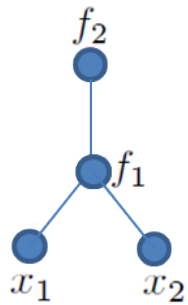
- Stabilität: Auswirkung inexakter Auswertung ohne Eingabefehler
- Kondition: Verstärkung von Eingabefehlern bei exakter Auswertung
- Gesamtfehler: Eingabefehlern und inexakte Auswertung

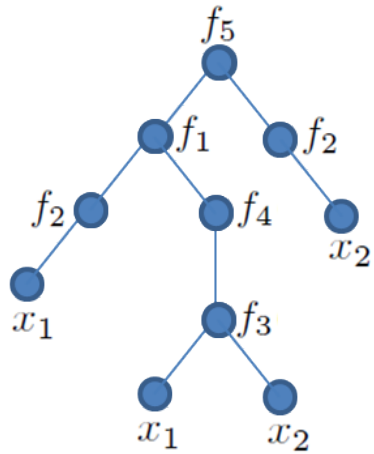
Satz 7.5: Gesamtfehler $= \kappa_{\text{rel}} \cdot$ Eingabefehler $+ \sigma_{\text{rel}} \cdot$ Auswertungsfehler!

Beispiel (Algorithmus 1)

- Wir benötigen: relative Kondition von $f(x_1, x_2) = (x_1 - x_2)^2$
- Funktion von mehreren Unbekannten
- Alternative: Direkte Abschätzung

Algorithmus 1





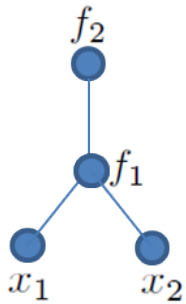
Gesamtfehler (für $x_1 = 10^{11}$, $x_2 = 10^{11} - 1$)

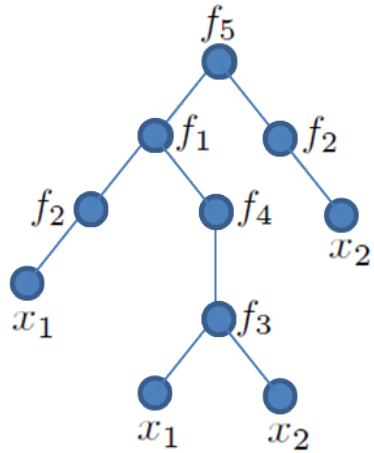
- Direkte Abschätzung. Idee: Schreibe $x_i = \tilde{Id}(x_i)$
- Ersetze σ^{β_1} durch $\sigma^{\beta_1}_{\text{ges}} = 1 + \kappa(f^{\beta_1})$
- Abschätzung: $\sigma_{\text{ges}} \leq 1 + \kappa(f^{\beta_1}, z^{\beta_1})\sigma^{\beta_1}_{\text{ges}} = 1 + 2 \cdot \sigma^{\beta_1}_{\text{ges}}$
- Berechnung von $\sigma^{\beta_1}_{\text{ges}}$

$$\sigma^{\beta_1}_{\text{ges}} = 1 + \kappa(f^{\beta_1}, (x_1, x_2)) = 1 + \frac{|x_1| + |x_2|}{|x_1 - x_2|} = 1 + \frac{2 \cdot 10^{11} - 1}{1} = 2 \cdot 10^{11}$$

- Ergebnis: $\sigma_{\text{ges}} \leq 1 + 4 \cdot 10^{11}$

Vergleich der Algorithmen





Gesamtfehlerverstärkung

- Algorithmus 1: $\sigma_{\text{ges}} \leq 1 + 4 \cdot 10^{11}$
- Algorithmus 2: $\sigma_{\text{ges}} \leq 44 \cdot 10^{22}$.

Systematische Stabilitätsanalyse

... kann der Computer übernehmen.

Automatische Fehleranalyse

- Eingabedaten: $x_1, \ldots, x_n$
- Elementarfunktionen: g_i
- Ableitungen: g_i'

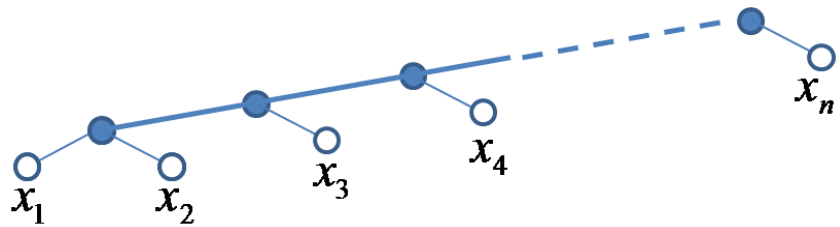
Auswertung der Summe positiver Zahlen

Freie Universität Berlin

$$S = x_1 + x_2 + \cdots + x_n = \sum_{k=1}^{n} x_k, \qquad x_k > 0 \quad \forall k$$

Rekursive Summation

```
>>> S = 0
>>> for k in xrange(len(x)):
>>>     S = S + x[k]
```



Rekursive Stabilitätsanalyse

$$\sigma^{\beta_{k+1}} = 1 + \max\{\sigma^{\beta_k}, 1\} = 1 + \sigma^{\beta_k}, \qquad k = 1, \ldots, n-1, \qquad \sigma^{\beta_1} = 1$$

Fehlerverstärkung bei rekursiver Summation: $\sigma_{\text{rek}} \leq n$

Auswertung der Summe positiver Zahlen

Freie Universität Berlin

$$S = a_1 + a_2 + \cdots + a_n = \sum_{k=1}^{n} a_k, \qquad a_k > 0 \quad \forall k$$

Hierarchische Summation

$$S = \big((x_1 + x_2) + (x_3 + x_4)\big) + \big((x_5 + x_6) + (x_7 + x_8)\big)$$

Rekursive Stabilitätsanalyse

$$\sigma^{\beta_i^{k+1}} = 1 + \max\{\sigma_{2i}^{\beta^k}, \sigma_{2i+1}^{\beta^k}\}, \quad k = 1, \ldots, \log_2(n), \quad \sigma^{\beta_i^1} = 1, \quad i = 0, \ldots$$

Fehlerverstärkung bei hierarchischer Summation: $\sigma_{\text{hier}} \leq \log_2(n)$

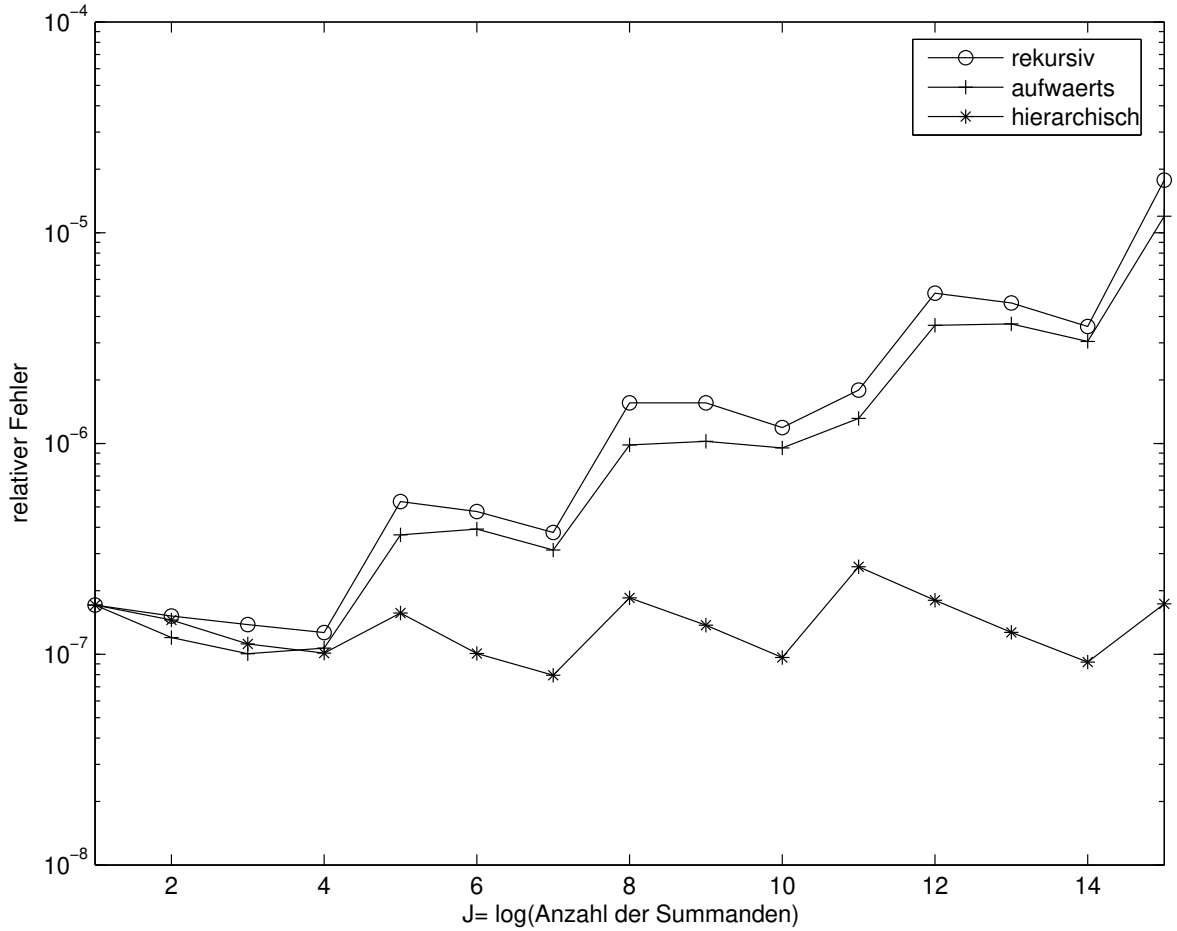
Numerisches Beispiel

Freie Universität Berlin

Genauigkeit: $\ell = 7$ Dezimalstellen

Problem:

Summation von $n_j = 2^J$ Zufallszahlen $a_k \in (0,1), \quad J = 1, \ldots, 15$



Kondition: Klassifizierung von Problemen

Freie Universität Berlin

Problem: Auswertung von $f(x)$

Definition: Auswirkung von Eingabefehlern auf das Ergebnis (Formel!)

Ausrechnen: Formeln!

Bedeutung: Grenzen der Genauigkeit (Formel!)

Beispiele

Stabilität: Klassifizierung von Algorithmen

Freie Universität Berlin

Algorithmus: $f(x) = g_n \circ \cdots \circ g_1(x)$

Definition: Auswirkung von sukzessivem Runden auf das Ergebnis (Formel!)

Elementarfunktionen: Modell für den tatsächlichen Ablauf im Rechner

Ausrechnen: Formeln!

Bedeutung: vermeidbare(?) Grenzen der Genauigkeit (Formel!)

Beispiele

Weitere Klassifizierung: Komplexität und Effizienz

Freie Universität Berlin

Wie lange muß ich auf das Ergebnis warten?

- Ist das Problem schwierig? (Komplexität)
- Ist mein Algorithmus zu langsam? (Effizienz)

Theoretische Informatik/Diskrete Mathematik

- Komplexitätstheorie, Berechenbare Funktionen

Aufwand (Laufzeit) eines Algorithmus

Freie Universität Berlin

Aufwand hängt ab von

- Algorithmus
- Eingabedaten („Größe“ des Problems)

Aufwandsmaß: reine Rechenzeit

- abhängig von weiteren Parametern
- Implementierung, Programmiersprache, Compiler, Prozessor, ...

Aufwandsmaß: Anzahl dominanter Operationen (problemabhängig!)

- implementierungsunabhängig

$$S = x_1 + x_2 + \cdots + x_n = \sum_{k=1}^{n} x_k, \qquad x_k > 0 \quad \forall k$$

Rekursive Summation

```
>>> S = 0
>>> for k in xrange(len(x)):
>>>     S = S + x[k]
```

Aufwandsmaß: Anzahl der Additionen

Aufwand des rekursiven Algorithmus: $n - 1$

Auswertung der Summe positiver Zahlen

$$S = a_1 + a_2 + \cdots + a_n = \sum_{k=1}^{n} a_k, \qquad a_k > 0 \quad \forall k$$

Hierarchische Summation

$$S = \Big((x_1 + x_2) + (x_3 + x_4)\Big) + \Big((x_5 + x_6) + (x_7 + x_8)\Big)$$

Aufwandsmaß: Anzahl der Additionen

Auswertung der Summe positiver Zahlen

$$S = a_1 + a_2 + \cdots + a_n = \sum_{k=1}^{n} a_k, \qquad a_k > 0 \quad \forall k$$

Hierarchische Summation

$$S = \big((x_1 + x_2) + (x_3 + x_4)\big) + \big((x_5 + x_6) + (x_7 + x_8)\big)$$

Aufwandsmaß: Anzahl der Additionen

Aufwand des hierarchischen Algorithmus: $n = 2^J$ bzw. $J = \log_2(n)$

$$\sum_{j=0}^{J-1} 2^j$$

$$S = a_1 + a_2 + \cdots + a_n = \sum_{k=1}^{n} a_k, \qquad a_k > 0 \quad \forall k$$

Hierarchische Summation

$$S = \big((x_1 + x_2) + (x_3 + x_4)\big) + \big((x_5 + x_6) + (x_7 + x_8)\big)$$

Aufwandsmaß: Anzahl der Additionen

Aufwand des hierarchischen Algorithmus: $n = 2^J$ bzw. $J = \log_2(n)$

$$\sum_{j=0}^{J-1} 2^j = \frac{2^J - 1}{2 - 1}$$

Auswertung der Summe positiver Zahlen

Freie Universität Berlin

$$S = a_1 + a_2 + \cdots + a_n = \sum_{k=1}^{n} a_k, \qquad a_k > 0 \quad \forall k$$

Hierarchische Summation

$$S = \Big((x_1 + x_2) + (x_3 + x_4)\Big) + \Big((x_5 + x_6) + (x_7 + x_8)\Big)$$

Aufwandsmaß: Anzahl der Additionen

Aufwand des hierarchischen Algorithmus: $n = 2^J$ bzw. $J = \log_2(n)$

$$\sum_{j=0}^{J-1} 2^j = \frac{2^J - 1}{2 - 1} = 2^J - 1 = n - 1$$

Komplexität der Summation positiver Zahlen

Kann es einen Algorithmus geben, der weniger als $n-1$ Additionen benötigt?

Nein!

Folgerung: Die Komplexität der Addition von n Zahlen ist $n-1$

Aufwand und Komplexität

Gegeben:

- Problem: (P)
- Menge aller Algorithmen zur Lösung von (P): $\mathcal{A}(P)$
- Eingabedaten der Länge n
- Referenzoperationen (problemspezifisch)

Definition (2.4)

Der Aufwand $T_A(n)$ eines Algorithmus $A \in \mathcal{A}(P)$ ist das

Maximum der benötigten Anzahl von Referenzoperationen
über alle zulässigen Eingabedaten der Länge n.

Definition (2.5)

Die Komplexität $\mathcal{K}_P(n)$ von (P) ist

$$\mathcal{K}_P(n) = \inf_{a \in \mathcal{A}(P)} T_A(n).$$

- Problem: (P)
- Algorithmus $A \in \mathcal{A}(P)$ zur Lösung von (P)

Ein Algorithmus A ist effizient, wenn

- Theoretische Informatik: $T_A(n) = O(n^p)$
- Numerische Mathematik: $T_A(n) = O(\mathcal{K}_P(n))$

Definition

Landau-Symbol $O(\cdot)$ für $f(n), g(n) \to \infty$ für $n \to \infty$

$$f(n) \in O(g(n)) \quad \text{für } n \to \infty \qquad \Leftrightarrow \qquad \limsup_{n\to\infty} \frac{f(n)}{g(n)} < \infty.$$

Beispiel: $18n^3 + 3n^2 + \sin(e^n) \in O(n^3)$