

Intervall-Arithmetik

- In manchen Fällen genügt es, statt einer aufwendigen exakten Arithmetik ein **Resultat mit Fehlerabschätzung**
- Jede reelle Zahl x wird in **Intervall-Notation** $x = [x_{\min}, x_{\max}]$ mit $x_{\min} \leq x \leq x_{\max}$ dargestellt, im Falle von $x_{\min} = x_{\max}$ spricht man von einem Punktintervall
- Bei bekanntem Fehler ε_x kann das Intervall $x = [x - \varepsilon_x, x + \varepsilon_x]$ generiert werden
- Wenn f stetig ist, dann ist $f(x) = f([x_{\min}, x_{\max}])$ ebenfalls ein Intervall $[y_{\min}, y_{\max}]$ mit $y_{\min} = \min\{f(x) \mid x \in [x_{\min}, x_{\max}]\}$ und $y_{\max} = \max\{f(x) \mid x \in [x_{\min}, x_{\max}]\}$
- Für die **Addition** gilt: $x + y = [x_{\min} + y_{\min}, x_{\max} + y_{\max}]$
- Für die **Subtraktion** gilt: $x - y = [x_{\min} - y_{\max}, x_{\max} - y_{\min}]$

Intervall-Arithmetik (Forts.)

- Für die **Multiplikation** gilt: $x \cdot y = [\min(P), \max(P)]$
mit $P = \{x_{\min} \cdot y_{\min}, x_{\min} \cdot y_{\max}, x_{\max} \cdot y_{\min}, x_{\max} \cdot y_{\max}\}$
- Für die **Division** gilt: $x / y = [\min(Q), \max(Q)]$
mit $Q = \{x_{\min} / y_{\min}, x_{\min} / y_{\max}, x_{\max} / y_{\min}, x_{\max} / y_{\max}\}$
Problem: was ist bei $y_{\min} < 0 < y_{\max}$?
- In der Intervall-Arithmetik gelten Kommutativ-, Assoziativ- und Distributivgesetz
Anmerkung: es gibt kein additives und multiplikatives inverses Element!
- Implementierung mit **Gleitkommazahlen**: bei Berechnung von $z = [z_{\min}, z_{\max}] = f(x, y)$ muss zur Berücksichtigung möglicher Rundungsfehler z_{\min} abgerundet (*rounding towards $-\infty$*) und z_{\max} aufgerundet (*rounding towards $+\infty$*) werden!
Anmerkung: Unterlauf und Überlauf können ggf. berücksichtigt werden!

Intervall-Arithmetik (Forts.)

- Intervall-Arithmetik liefert zuverlässige, wenn auch etwas zu **pessimistische** Fehlerabschätzungen
- Für arithmetische Operationen wird nur ca. die doppelte Zeit im Vergleich zur herkömmlichen Arithmetik benötigt
- Intervall-Arithmetik auf Gleitkommazahlen ist in heutigen Hochsprachen nicht direkt implementierbar, da die Wahl einer Rundungsart hier nicht unterstützt wird
- alternative Möglichkeiten der Implementierung:
 - mit Maschinensprache
 - mit Mathematik-Softwarepaketen wie z.B. **Mathematica** oder **Maple**
 - mit **Bibliotheken für C++** (z.B. bietet LEDA den Datentyp `interval` und entsprechende Operationen)
 - mit **JAVA** bei Verwendung von `BigDecimal`