Übungen zur Computerphysik
SS 2020

T. Luu, A. Nogga, M. Petschlies, A. Wirzba

Übungsblatt 2 **16. - 22. April 2020**

A.2: Grundlegende mathematische Operationen in der Numerik

Wir entwickeln unser Repertoire numerischer Verfahren: diese werden dann (in Kombination) den Schlüssel zum Lösen vielfältiger physikalischer Problemstellungen bilden. Wir beginnen mit je einem Beispiel zur numerischen Ableitung, Nullstellensuche und Integration.

1. Numerische Ableitung

   (a) Geben Sie die Berechnungsvorschrift für die symmetrische 1., 2. und 3. numerische Ableitung. Als Anwendung betrachten Sie die Ableitungen der Funktion

   $$f(x) = \log(x) \tag{1}$$

   bei $x_0 = 3$.

   Die symmetrischen Ableitungen sind gegeben durch

   $$\Delta^{(1)} f(x) = \frac{1}{2h} \left(f(x+h) - f(x-h)\right)$$
   $$\Delta^{(2)} f(x) = \frac{1}{h^2} \left(f(x+h) + f(x-h) - 2f(x)\right)$$
   $$\Delta^{(3)} f(x) = \frac{1}{2h^3} \left(\left(f(x+2h) - f(x-2h)\right) - 2\left(f(x+h) - f(x-h)\right)\right)$$

   (b) Welche maximale Genauigkeit erreichen Sie dabei für die Berechnung in einfacher, doppelter (*optional:* vierfacher) Genauigkeit ?

   Vgl. z.B. Programm A2_abl.c.
   Ableitungsfunktionen deriv_sym_1/2/3 . In main werden die numerischen Ableitungen in Abhängigkeit von der Schrittweite h mit Schleife über kleiner werdendes h berechnet. Genauigkeit = absolute Abweichung vom exacten (= analytische berechneten) Wert.
   Darstellung im h - Genauigkeit - Plot.

   (c) Und übrigens: Welches ist die größte Zahl ϵ mit $1 + \epsilon = 1$ in einfacher, doppelter (*optional:* vierfacher) Genauigkeit ? Bestimmen Sie den Wert analytisch und demonstrieren Sie numerisch.
   (Hinweis: einfache / doppelte / vierfache Genauigkeit mit 23 / 52 / 112 Bit Mantisse; vierfache Genauigkeit mit gcc z.B. über quadmath-Bibliothek mit Datentyp

__float128)

Mit m-Bit Mantisse ist die kleinste darstellebare Zahl 2^{-m}, also

$$\begin{aligned}\epsilon\,(\text{float}) &= 5.960464 \times 10^{-8}\\ \epsilon\,(\text{double}) &= 1.110223 \times 10^{-16}\\ \epsilon\,(\text{__float128}) &= 9.62965 \times 10^{-35}\end{aligned}$$

Aber: Diese Grenze gilt nur für die Additonsoperation. Für zusammengesetzte Operationen (z.B. Funktionsauswertungen) werden Fehler akkumuliert und es gelten i.A. kleinere Genauigkeitsgrenzen. Vgl. auch Aufgabe (b) oben.

2. Approximative Nullstellenbestimmung
   Nullstellensuche ist ein Allerweltsproblem; in vielen Formen zur Lösung nicht-linearer Gleichungen, für Optimierungsprobleme, ...
   Im Beispiel hier betrachten wir ein Extremalwertproblem.

   (a) Benutzen Sie Ihre Implementation aus 1. und das Sekanten-Verfahren zur Bestimmung der Stelle des Extremums der Funktion

   $$f(x) = \log(x)^{\frac{1}{x}} \tag{2}$$

   im Intervall $[1\,,\,6]$.
   (Hinweis: *optional* Cross-check durch Auffinden des Maximums mit z.B. Bisektion)

   Vgl. z.B. Programm A2_nst.c. Startwerte für die Iteration des Sekantenverfahrens durch Ansicht des Funktionsgraphen.

3. Integration
   Wir wollen das Trapez-Verfahren für die numerische Integration benutzen, und zwar mit adaptiver Schrittweite.

   (a) Das Ziel ist die näherungsweise Berechnung der Eulerschen Gammafunktion aus der Integraldarstellung

   $$\Gamma(z) = \int_0^\infty dt \exp(-t)\; t^{z-1} \tag{3}$$

   für reelles, nicht-negatives z.
   Implementieren Sie die Trapez-Integration. Iterieren Sie die Integration für Schrittweiten h, $h/2$, $h/4$, ... bis zum Unterschreiten einer vorgegebenen absoluten / relativen Änderung der approximierten Integralwerte.
   (Hinweis: Nutzen Sie auch die Eigenschaften der Gammafunktion aus.)

   Vgl. Programm A2_trapez.c.
   Formeln für die Iteration wie in der Vorlesung. Cross-check z.B. über ganzzahliges Argument mit bekanntem Wert $\Gamma(z) = (z-1)!$ für $z \in \{0, 1, 2, \dots\}$.
   Dank $z\,\Gamma(z) = \Gamma(z+1)$ brauchen wir die obere Integrationsgrenze nur für z.B. $1 \le z < 2$ einstellen. Für alle anderen Werte können wir die o.g. Eigenschaft ausnutzen.

(b) Was wird im Allgemeinen der numerisch teuerste Teil dieser Berechnung sein? Versuchen Sie, diesen Posten in der Iteration zu minimieren.

Das teure, und vor allem auch unkontrollierbar teure ist die Funktionauswertung. Die Anzahl von Funktionsauswertungen möchte man daher so klein wie möglich halten.
Mit der Folge $h_k = h_0/2^k$ für $k = 0, 1, \ldots$ kann man ein "Restart" machen, also in jedem neuen Schritt lediglich die hinzukommenden Zwischenpunkte dazunehmen. (Vgl. die Romberg-Folge aus der Vorlesung.)