Projekt I/1: RECHENFEHLER

- Teilprojekt 1:
  Die Nullstellen x_1, x_2 des quadratischen Polynoms

$$x^2 + px + q \tag{1}$$

  sind durch

$$x_{1,2} = -\frac{p}{2} \pm \sqrt{\frac{p^2}{4} - q} \tag{2}$$

  gegeben.

(1.1) Quantifizieren Sie die relative Kondition der Nullstellen[1] und zeigen Sie, daß die Kondition schlecht ist, wenn x_1 und x_2 sehr nahe beieinanderliegen. Interpretieren Sie diese Tatsache geometrisch.

(1.2) Für welche Parameterkonstellationen tritt bei der Berechnung von x_1 und x_2 gemäß (2) Auslöschung auf? Schätzen Sie den absoluten und relativen Rechenfehler, der durch die Auswertung der obigen Formeln in einer Gleitpunktarithmetik entsteht ab, d.h. geben Sie einen Ausdruck für die Rundungsfehlerschranke abhängig von p, q und der Maschinengenauigkeit *eps* an. Diskutieren Sie die numerische Stabilität, indem Sie untersuchen, wann diese Schranke signifikant größer als die entsprechende Konditionsschranke ist. Geben Sie für die Berechnung von x_1 und x_2 alternative Varianten an, um solche Instabilitäten zu vermeiden.

(1.3) Bestätigen Sie experimentell die obigen Überlegungen. Vergleichen Sie auch die stabile Berechnungsart der kritischen Nullstelle mit der instabilen, aber in doppelter Genauigkeit ausgeführten Variante.

- Teilprojekt 2:

(2.1) Die beiden Ausdrücke

$$f(x) = \sqrt{x+1} - \sqrt{x}, \quad x > 0$$

und

$$g(x) = \frac{1}{\sqrt{x+1} + \sqrt{x}}, \quad x > 0$$

[1] Man betrachte nur den Fall zweier verschiedener, reeller Nullstellen.

sind algebraisch äquivalent. Geben Sie für f und g absolute und relative Rundungsfehlerschranken als Ausdrücke in x und der Maschinengenauigkeit *eps* an. Welche der beiden Auswerteformeln ist aufgrund dieser Rechenfehlerschranken vorzuziehen? Decken sich diese Ergebnisse mit der intuitiven Beurteilung bezüglich allfälliger Auslöschung?

(2.2) Für die Ausdrücke in der untenstehenden Tabelle untersuchen Sie:

(i) ob Auslöschungssituationen vorliegen,

(ii) ob im Falle der Auslöschung gute Kondition vorliegt,

(iii) wie man in gutkonditionierten Auslöschungsfällen stabile Auswertungsvarianten angeben kann.

Ausdruck	Bereich
$\sqrt{1+x} - \sqrt{1-x}$	x positiv und klein
$\sin(x+c) - \sin x$	$x \approx -\frac{\pi}{2}$, c klein
$\arctan(a+1) - \arctan a$	a klein
$\frac{1}{1+3x} - \frac{1-2x}{1+x}$	x groß
$\frac{1-\cos x}{x}$	$x \approx \pi$
$\frac{1}{\cosh x - \sinh x}$	x positiv und groß
$\frac{1}{\cosh x - \sinh x}$	x negativ und groß
$\frac{e^x - e^{-x}}{2}$	x groß
$1 - \sqrt{1 - 2^{-n}\pi}$	$n \in \mathbb{N}$ groß
$\sqrt{\lvert \tan^2 x - \cos^2 x \rvert}$	x klein

Bestätigen Sie experimentell die Überlegenheit der stabilen Auswertungsvarianten.

- Teilprojekt 3:
  Die Exponentialfunktion kann man durch ihr Taylorpolynom n-ten Grades[2] approximieren:

$$e^x \approx P_n(x) := \sum_{i=0}^{n} \frac{x^i}{i!}.$$

Um einen Eindruck von den Rundungsfehlern zu erhalten, die bei der Auswertung von $P_n(x)$ entstehen, führen Sie folgendes Experiment durch: Berechnen Sie die Werte von P_{10} und P_{50} für

$$x = 1,\ 10,\ 20,\ -10,\ -20,$$

indem Sie die Summe in natürlicher Reihenfolge ($i = 0, 1, \ldots, n$) und in umgekehrter Reihenfolge auswerten. Für jede Stelle x berechnen Sie die absoluten und relativen Fehler der so erhaltenen vier Näherungswerte. Als Referenzwert nehmen Sie jeweils das entsprechend genaue Ergebnis aus MAPLE.

Man sollte beobachten, daß die Approximation der Werte von e^x durch direkte Auswertung des Taylorpolynoms für betragsgroße x rechenfehleranfällig ist. Es liegt daher nahe, e^x durch $P_n(x)$ nur in einem geeignet gewählten Standardintervall, in dem die Auswertung des Taylorpolynoms keinen großen Rechenfehler verursacht, zu approximieren. Folgende Vorgangsweise realisiert diese Idee:

(i) Ist x im Standardintervall $[0, \ln 2]$, so wird e^x durch $P_{10}(x)$ ersetzt, wobei die Summe in der umgekehrten Reihenfolge ausgewertet wird.

(ii) Ist $x > \ln 2$, dann stellt man x als

$$x = u \ln 2 + r, \quad \text{mit } u \in \mathbb{N},\ 0 < r < \ln 2,$$

dar, und es gilt

$$e^x = e^{u \ln 2 + r} = 2^u e^r.$$

Man berechnet e^r mit der Prozedur aus (i) und führt anschließend die Multiplikation mit 2^u durch. Liegt eine Arithmetik mit der Basis $b = 2$ vor, so erfolgt diese Multiplikation sogar rechenfehlerfrei. Für eine andere Basis

[2] Man beachte, daß für betragsgroße Werte von x Taylorpolynome hohen Grades erforderlich sind, um die Exponentialfunktion gut zu approximieren.

b genügt es, in (i) das rechte Intervallende $\ln 2$ durch $\ln b$ zu ersetzen, um den gleichen Effekt zu erreichen.

(iii) Ist $x < 0$, so berechnet man zuerst $e^{|x|}$ nach (i) oder (ii) und bildet anschließend $e^x = \frac{1}{e^{|x|}}$.

Um die Qualität dieser Variante zu beurteilen plotten Sie den Verlauf von

$$\delta(x) := \frac{P_{10}(x) - e^x}{e^x}$$

für $x \in [0, \ln 2]$. Dabei werde $P_{10}(x)$ in einfacher Genauigkeit berechnet, d.h. die geplottete Größe $\delta(x)$ ist der relative Gesamtfehler, also der Rechen- und Approximationsfehler dieser Vorgangsweise.

- Teilprojekt 4:

(4.1) Nach Archimedes kann man π näherungsweise als halben Umfang u_n eines regelmäßigen 2^n-Ecks, das dem Einheitskreis eingeschrieben ist, berechnen. Es gilt

$$\lim_{n\to\infty} u_n = \pi = 3.14159265\ldots$$

Diese Überlegung führt auf folgendes Verfahren zur numerischen Berechnung von π :

$$\begin{aligned} u_1 &:= 2, \\ n \geq 1: \quad u_{n+1} &:= 2^{n+1}\sqrt{\frac{1}{2}\left(1 - \sqrt{1 - (2^{-n}u_n)^2}\right)}. \end{aligned}$$

Führen Sie die obige Iteration numerisch so lange durch, bis alle Stellen des Ergebnisses "stehen". Erklären Sie die Ursachen des instabilen Verhaltens, das sich im Zuge der Iteration einstellt.

(4.2) Beseitigen Sie die Instabilität durch geeignete Umformulierung der Iterationsvorschrift und begründen Sie diese ausführlich. Die so gewonnene Iteration führen Sie so lange durch bis sämtliche Stellen "stehen", und vergleichen Sie den so erhalteten Wert mit π.

(4.3) Führen Sie die beiden Iterationen in doppelter Genauigkeit durch und vergleichen Sie mit dem doppelt genauen Wert von π.

- Teilprojekt 5:
Zur numerischen Lösung linearer Gleichungsysteme $Ax = b$ soll MATLAB benützt werden. Experimentieren Sie mit solcher Software um Rundungsfehler zu beobachten. Konstruieren Sie dazu Testbeispiele mit Hilbertmatrizen steigender Dimension und mit bekannter Lösung x.

- Teilprojekt 6:
  Der Ausdruck

$$\phi(x) = \frac{1}{1+3x} - \frac{1-2x}{1+x} = \frac{6x^2}{(1+3x)(1+x)},$$

soll für kleine Werte von $x \approx 0$ berechnet werden.

a) Geben Sie die relative Konditionszahl $K_{\phi(x)\leftarrow x}$ als Funktion von x an. Schätzen Sie $K_{\phi(x)\leftarrow x}$ für alle $x \in [-\frac{1}{10}, \frac{1}{10}]$ ab, d.h., geben Sie eine von x unabhängige Konstante C an, so dass für alle $x \in [-\frac{1}{10}, \frac{1}{10}]$ die Ungleichung $|K_{\phi(x)\leftarrow x}| \leq C$ gilt. Welche Aussage kann man demgemäß für die Kondition des Problems machen?

b) Für die beiden Auswertungsvarianten[3]

$$\phi_1(x) = \frac{1}{1+3x} - \frac{1-2x}{1+x},$$

$$\phi_2(x) = \frac{6x^2}{(1+3x)(1+x)},$$

schätzen Sie den relativen Rechenfehler ab. Dabei wird vorausgesetzt, daß x ist eine Maschinenzahl aus dem Intervall $[-\frac{1}{10}, \frac{1}{10}]$ ist. Was kann man demgemäß überdie Stabilität der Auswertung von ϕ_1 bzw. ϕ_2 im angegeben x-Bereich sagen?

c) Welche der beiden Auswertungsvarianten ist für kleine Werte von x vorzuziehen? Begründen Sie Ihre Wahl und bestätigen Sie sie durch entsprechende numerische Experimente.

[3]Auswertung in Gleitpunktarithmetik

Projekt I/2: VERFAHRENSFEHLER

Thema 1: Numerische Differentiation

Näherungsformeln für die erste Ableitung $y'(t)$ können wie folgt gewonnen werden: Man interpoliert die Funktion y an den Stellen

$$t - lh,\ t - (l-1)h,\ \ldots, t,\ \ldots,\ t + (n-1)h,\ t + nh,\ \ l, n \in \mathbb{N}$$

mit einem Polynom P vom Grad $l + n$ und nimmt die Ableitung $P'(t)$ als Approximation für $y'(t)$. Diese Vorgangsweise führt im Falle $l = 0$ und $n = 1$ zum einseitigen Differenzenquotienten

$$D_e(h) = \frac{y(t+h) - y(t)}{h}, \tag{1}$$

der die erste Ableitung auf $O(h)$-Niveau approximiert, und im Falle $l = 1$ und $n = 1$ zum zentralen Differenzenquotienten

$$D_z(h) = \frac{y(t+h) - y(t-h)}{2h}, \tag{2}$$

der die erste Ableitung auf $O(h^2)$-Niveau approximiert. Entwickelt man die Größen $y(t + kh)$,
$k \in \mathbb{Z},\ -l \le k \le n$ um die Stelle t in die Taylorreihe, so erhält man unter entsprechenden Glattheitsvoraussetzungen an y asymptotische Entwicklungen des Verfahrensfehlers. Beispielsweise ergibt sich für D_z und $y \in C^7[a, b]$

$$D_z(h) - y'(t) = h^2 \cdot \frac{y^{(3)}(t)}{3!} + h^4 \cdot \frac{y^{(5)}(t)}{5!} + R_6(h) \tag{3}$$

mit $|R_6(h)| \le h^6 \cdot \frac{M_7}{7!}$, $M_7 = \max_{a \le t \le b} |y^{(7)}(t)|$.

Diese Entwicklungen bieten die Möglichkeit mit Hilfe der Extrapolation bessere Approximationen der Ableitung zu gewinnen. Extrapolationsformeln können auf zwei Arten konstruiert werden:

(i) Man betrachtet die asymptotische Fehlerentwicklung für mehrere h-Werte und eliminiert die führenden Terme dieser Entwicklung, um zu Formeln höherer Ordnung zu gelangen. Nehmen wir beispielsweise an, daß wir drei Werte von D_z für

die Schrittweiten $h, h/2, h/4$ berechnet haben. Dann gilt nach (2)

$$D_z(h) - y'(t) = h^2 \cdot \frac{y^{(3)}(t)}{3!} + h^4 \cdot \frac{y^{(5)}(t)}{5!} + R_6(h), \tag{4}$$

$$D_z\left(\frac{h}{2}\right) - y'(t) = \left(\frac{h}{2}\right)^2 \cdot \frac{y^{(3)}(t)}{3!} + \left(\frac{h}{2}\right)^4 \cdot \frac{y^{(5)}(t)}{5!} + R_6\left(\frac{h}{2}\right), \tag{5}$$

$$D_z\left(\frac{h}{4}\right) - y'(t) = \left(\frac{h}{4}\right)^2 \cdot \frac{y^{(3)}(t)}{3!} + \left(\frac{h}{4}\right)^4 \cdot \frac{y^{(5)}(t)}{5!} + R_6\left(\frac{h}{4}\right), \tag{6}$$

und wir können durch Bildung geeigneter Linearkombinationen dieser Gleichungen, was auf die Elimination der h^2- und h^4-Terme auf der rechten Seite hinausläuft, eine Näherung $D_{extr}(h)$ für $y'(t)$ gewinnen, für die

$$D_{extr}(h) - y'(t) = O(h^6)$$

gilt.

(ii) Der zentrale Differenzenquotient $D_z(h)$ ist eine Funktion in h mit der Eigenschaft

$$h = 0: \quad D_z(0) = y'(t).$$

Es liegt daher nahe die Funktion D_z mit einem Polynom Q zu interpolieren und den Wert dieses Interpolationspolynoms an der Stelle $h = 0$ als Näherung für $D_z(0) = y'(t)$ zu nehmen. Analog zu (2) gilt für $y \in C^{2s+3}[a, b]$

$$D_z(h) = y'(t) + h^2 \cdot \frac{y^{(3)}(t)}{3!} + h^4 \cdot \frac{y^{(5)}(t)}{5!} + \cdots + h^{2s} \cdot \frac{y^{2s+1}(t)}{(2s+1)!} + R_{2s+2},$$

d.h. D_z ist (bis auf das Restglied R_{2s+2}) ein Polynom in h^2. Daher setzt man Q ebenfalls als Polynom in h^2 an. Man beachte, daß für die Berechnung des Wertes $Q(0)$ das explizite Aufstellen von Q nicht erforderlich ist; man kann diesen Wert mit Hilfe des Neville-Schemas direkt berechnen.

- Teilprojekt 1.1:
  Für die beiden Näherungsformeln (1) und (2) betrachten Sie die Fehlerschranke

  $$S(\delta, eps; h) := \text{Verfahrensfehlerschranke} + \text{Rechenfehlerschranke}.$$

  Dabei ist δ die Schranke für die Auswertefehler von y und eps die Maschinengenauigkeit. Man beachte, daß S auch noch von gewissen Ableitungsschranken der Funktion y abhängt. Bestimmen Sie für (1) und (2) jene Schrittweite $h^* = h(\delta, eps)$, für die S minimal wird. Berechnen Sie dieses Minimum für

$\delta = eps = 10^{-8}$ unter der Annahme, daß die Beträge der auftretenden Ableitungen durch 1 abgeschätzt werden können.

Bestimmen Sie die Größe $D_{extr}(h)$ als Ausdruck in $D_z(h)$, $D_z(h/2)$ und $D_z(h/4)$ und geben Sie den Faktor in der Abschätzung

$$|D_{extr}(h) - y'(t)| \leq h^6 \cdot \text{Faktor} \tag{7}$$

als expliziten Ausdruck in M_7 an.

Implementieren Sie die Näherungsformeln $D_e(h)$, $D_z(h)$ und $D_{extr}(h)$ und vergleichen Sie an selbstgewählten Beispielen ihre Qualität. Dazu plotten Sie die entsprechenden Fehlerverläufe als Funktion von h.

- Teilprojekt 1.2:
  Wie oben ausgeführt, läßt sich die Extrapolation nicht nur durch Elimination der führenden Terme der asymptotischen Entwicklung begründen, sondern auch durch Aufstellung eines Interpolationspolynoms $Q(h^2)$, das D_z interpoliert. Realisieren Sie diese Idee, indem Sie ein Programm schreiben, das auf dem Neville-Schema beruht und aus den Größen $D_z(h_0), D_z(h_1), \ldots, D_z(h_m)$ eine Näherung $D_{int}(h_0, h_1, \ldots, h_m)$ für y' berechnet.

  Verifizieren Sie die Übereinstimmung von $D_{int}(h, h/2, h/4)$ mit der Größe $D_{extr}(h)$ aus Teilprojekt 1.1.

  Geben Sie eine auf der Interpolationsfehlerformel basierende Abschätzung des Fehlers $|D_{int}(h_0, h_1, \ldots, h_m) - y'(t)|$ an. Welche Voraussetzungen an die Glattheit von y werden dabei benötigt? Vergleichen Sie die Fehlerschranke für $|D_{int}(h, h/2, h/4) - y'(t)|$ mit jener, die bei Teilprojekt 1.1 erhalten wurde.

  Anhand von selbstgewählten Testbeispielen plotten Sie für verschiedene Schrittweitenfolgen die Extrapolationsfehlerverläufe in Abhängigkeit von der Maximalschrittweite.

- Teilprojekt 1.3:
  Leiten Sie eine Näherungsformel für $y'(t)$ der Ordnung $O(h^6)$ her, indem Sie die Werte $y(t + kh)$, $k \in \mathbb{Z}$, $-l \leq k \leq n$ interpolieren. Geben Sie den Verfahrensfehler dieser Formel an und vergleichen Sie ihn mit dem Verfahrensfehler

von D_{extr}, siehe Teilprojekt 1.1.

Implementieren Sie diese Formel und vergleichen Sie experimentell deren numerische Stabilität mit den Stabilitätseigenschaften von D_{extr} und $D_{int}(h_0, h_1, h_2) = D_{int}(h, c_1 h, c_2 h)$, vgl. Teilprojekt 1.2, mit verschiedener Wahl der positiven Konstanten c_1, $c_2 < 1$. Plotten Sie zu diesem Zweck die entsprechenden Fehlerverläufe.

Thema 2: Numerische Integration

Um eine Näherung für das bestimmte Integral

$$I = \int_a^b f(x)\,dx$$

zu berechnen, werden in der Numerischen Mathematik vorwiegend interpolatorische Quadraturformeln eingesetzt, d.h. man ersetzt den Integranden f durch eine stückweise polynomiale Interpolationsfunktion und nimmt den leicht zu berechnenden Integralwert der Interpolationsfunktion als Approximation für das gesuchte Integral. Die daraus resultierende Näherungsformel ist typischerweise eine Linearkombination von Funktionswerten $f(x_i)$, d.h. sie ist von der Form

$$I_h(f) := \sum_i c_i\, f(x_i), \;\; a \le x_i \le b,$$

wobei die Stellen x_i als "Knoten" und die Koeffizienten c_i als "Gewichte" der Integrationsformel bezeichnet werden. Den Verfahrensfehler solcher Formeln kann man leicht durch Integration des Interpolationsfehlers berechnen bzw. abschätzen.

Unterteilt man beispielsweise das Integrationsintervall $[a, b]$ in n gleiche Teile der Länge $h = \frac{b-a}{n}$, was zu einem äquidistanten Gitter mit den Gitterpunkten $x_i = a + ih$, $i = 0, 1, \ldots, n$ führt, und interpoliert man auf jedem der Teilintervalle die Daten $(x_{i-1}, f(x_{i-1})), (x_i, f(x_i))$ linear (Polynomgrad 1), so entsteht die sogenannte Trapezregel

$$I \approx T_h(f) := h \cdot \left[\frac{1}{2}f(a) + f(x_1) + \cdots + f(x_i) + \cdots + f(x_{n-1}) + \frac{1}{2}f(b)\right].$$

Interpoliert man bezüglich desselben Gitters auf den Teilintervallen

$$[a, x_2], \quad [x_2, x_4], \quad \ldots$$

jeweils die Datensätze

$$(a, f(a)), (x_1, f(x_1)), (x_2, f(x_2)), \quad (x_2, f(x_2)), (x_3, f(x_3)), (x_4, f(x_4)), \quad \dots$$

mit Polynomen 2. Grades, so ergibt sich die Simpsonregel

$$I \approx S_h(f) := h \cdot \left[\frac{1}{3}f(a) + \frac{4}{3}f(x_1) + \frac{2}{3}f(x_2) + \frac{4}{3}f(x_3) + \cdots + \frac{4}{3}f(x_{n-1}) + \frac{1}{3}f(b)\right].$$

Bei stückweise polynomialer Interpolation mit Polynomen vom Grad 4 ergibt sich die Milneregel

$$\begin{aligned} I \approx M_h(f) \;:=\; & h \cdot \left[\frac{14}{45}f(a) + \frac{64}{45}f(x_1) + \frac{24}{45}f(x_2) + \frac{64}{45}f(x_3) + \frac{28}{45}f(x_4) + \frac{64}{45}f(x_5) + \cdots + \right. \\ & \left. \frac{64}{45}f(x_{n-1}) + \frac{14}{45}f(b)\right]. \end{aligned}$$

Solche Näherungsformeln, die auf äquidistanter Interpolation beruhen heißen Newton-Cotes-Formeln.

Die bekanntesten auf nichtäquidistanter Interpolation basierenden Integrationsformeln sind die Gauß-Formeln. Bei der Konstruktion dieser Formeln wählt man die Gewichte *und* Knoten so, daß Polynome möglichst hohen Grades exakt integriert werden. Diese Formeln werden bezüglich des Standardintervalls $[-1, +1]$ angegeben. Die ersten Gau-Formeln lauten:

$$\int_{-1}^{+1} f(x)\,dx \approx G_1(f) := 2f(0),$$

$$\int_{-1}^{+1} f(x)\,dx \approx G_2(f) := f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right),$$

$$\int_{-1}^{+1} f(x)\,dx \approx G_3(f) := \frac{5}{9}\,f\left(-\sqrt{\frac{3}{5}}\right) + \frac{8}{9}f(0) + \frac{5}{9}\,f\left(\sqrt{\frac{3}{5}}\right).$$

Mit G_1 werden Polynome bis zum Grad 1, mit G_2 bis zum Grad 3 und mit G_3 bis zum Grad 5 exakt integriert.

Will man diese Formeln auf einem allgemeinen Intervall $[a, b]$ statt auf dem Standardintervall $[-1, +1]$ verwenden, so muß man sowohl die Integrationsgewichte als auch die Knoten entsprechend transformieren. Setzt man die Gauß-Formeln stückweise ein, so muß man sie auf entsprechende Teilintervalle des Integrationsintervalls transformieren.

Weitere Integrationsmethoden enstehen dadurch, daß man die Idee der Extrapolation (vgl. numerische Differentiation) für die interpolatorischen Quadraturformeln realisiert. Die auf der Trapezregel basierende Extrapolationsmethode wird als Romberg-Integration bezeichnet.

- Teilprojekt 2.1:
  Zeigen Sie, daß für den Verfahrensfehler der Trapezregel die folgende a-priori Abschätzung

$$\left| T_h(f) - \int_a^b f(x)\,dx \right| \leq \frac{M_2(b-a)}{12} h^2, \quad \text{für } \max_{a\leq x\leq b} |f''(x)| \leq M_2,$$

  gilt, und diskutieren Sie die Skalierungsinvarianz dieser Schranke, d.h. verifizieren Sie, daß sich bei Umskalierungen diese Schranke genauso verändert wie der Integralwert selbst.
  Um unterschiedliche Integrationsformeln bezüglich ihrer Effizienz zu vergleichen betrachten Sie das folgende Testbeispiel:

$$\int_0^1 e^x\,dx = e - 1.$$

  Berechnen Sie Approximationen dieses Integrals basierend auf fünf Funktionsauswertungen

  - mit der Trapezregel,
  - mit der Romberg-Integration,
  - mit der Milneregel,
  - mit G_5, siehe M. Abramowitz, I. Stegun: Handbook of Mathematical Functions,

  und vergleichen Sie ihre Genauigkeiten.

- Teilprojekt 2.2:
  Konstruieren Sie Testbeispiele für die numerische Integration mit Integranden die schwieriger, d.h. weniger glatt, als e^x sind (große Werte gewisser Ableitungen, Unstetigkeiten von Ableitungen). Integrieren Sie diese Funktionen mit den obigen vier Verfahren. Dabei wenden Sie diese Methoden auch stückweise an, d.h. auf Teilintervallen des gesamten Integrationsintervalls. Beobachten Sie die jeweiligen Verfahrensfehler. Was ist bezüglich der Ordnungen des Verfahrensfehlers in Zusammenhang mit verschiedenen Glattheitseigenschaften der Integranden zu erkennen?

- Teilprojekt 2.3:
  Bei der Implementierung numerischer Integrationsverfahren stattet man die oben vorgestellten Quadraturformeln mit zahlreichen Steuerungsmaßnahmen (Schrittweitensteuerung, Ordnungssteuerung, Fehlerschätzungen) aus. Ziel solcher Maßnahmen ist es in möglichst effizienter Weise ein durch Toleranzparameter vorgegebenes Genauigkeitsniveau zu erreichen. Stellen Sie anhand selbstgewählter Beispiele den Toleranzparameter dem tatsächlichen (Verfahrens-)Fehler gegenüber.

Projekt II/1: GAUSSELIMINATION

- Teilprojekt 1:
  Bekanntlich sind schlechtkonditionierte lineare Gleichungssysteme $Ax = b$ auch von höherer Rundungsfehlersensitivität. Erhärten Sie diese Tatsache mit numerischen Experimenten, indem Sie vorhandene Software einsetzen und bei konstruierten Testbeispielen Rechenfehler beobachten. Experimentieren Sie mit folgenden Matrizen:

  (i) A ist eine orthogonale Matrix z.B. eine Householdermatrix. Diese Matrix ist besonders gut konditioniert; die Konditionszahl bezüglich der $\|\cdot\|_2$-Norm ist gleich 1.

  (ii) Die Matrix A wird aus ihrer Singulärwertzerlegung $A = UDV^T$ aufgebaut. Durch ensprechende Wahl singulärer Werte kann man beliebige Nähe zu singulären Matrizen erreichen.

  (iii) Die Matrix A wird aus ihrer Eigenwertzerlegung $A = X\Lambda X^{-1}$ aufgebaut. Hier ist nicht nur der Fall betragskleiner Eigenwerte, sondern auch die Situation mit schlechtkonditionierten Transformationsmatrizen X interessant.

  (iv) A ist eine Vandermonde-Matrix, bei der man die Kondition über die Wahl der (Interpolation-)Knoten steuern kann.

- Teilprojekt 2:
  In der "reinen" Mathematik kann man zwischen regulären und singulären Matrizen scharf unterscheiden; beim numerischen Rechnen (Rundungsfehler!) geht die Schärfe dieser Unterscheidung verloren. Software zur Lösung linearer Gleichungssysteme meldet kritische Fälle mit "Matrix numerisch singulär" oder ähnlichen Fehlermeldungen und bricht die Berechnungen ab.
  Provozieren Sie solche Meldungen mit Testbeispielen, die gemäß (ii) – (iv) von Teilprojekt 1 konstuiert werden, indem Sie sich entsprechend nahe an die Singularität herantasten.

- Teilprojekt 3:
  Um die Gaußelimination numerisch stabil durchzuführen sind Maßnahmen wie Vorskalierung und Spaltenpivotsuche mit anschließendem Zeilentausch notwendig. Um die Auswirkungen dieser Maßnahmen zu studieren, experimentieren Sie mit Testbeispielen, die gemäß den bei Teilprojekt 1 vorgeschlagenen Konstruktionsmöglichkeiten gegeben sind, indem Sie zwei von Ihnen programmierte

Varianten der Gaußelimination heranziehen:
Variante 1: reine Gaußelimination,
Variante 2: Gaußelimination mit Vorskalierung und Spaltenpivotsuche.
Vergleichen Sie auch Ihre Programme mit den MATLAB Routinen.

- Teilprojekt 4:
  Wie Teilprojekt 3 aber mit Totalpivotsuche statt Spaltenpivotsuche. Vergleichen Sie den Fall der Totalpivotsuche nicht nur mit der reinen Gaußelimination sondern auch mit der Variante mit Spaltenpivotsuche. Was bringt die Totalpivotsuche bei Matrizen, die sehr nahe bei singulären Matrizen liegen?

- Teilprojekt 5:
  Eine Möglichkeit sich einen Eindruck von der Kondition gegebener Probleme zu beschaffen besteht darin, die Problemdaten künstlich d.h. mit Zufallsstörungen zu überlagern, die so enstandenen Probleme zu lösen und ihre Lösungen mit der Lösung des ungestörten Problems zu vergleichen (experimentelle Konditionsuntersuchung). Realisieren Sie diese Idee für lineare Gleichungssysteme, basierend auf Gauß-verteilten bzw. gleichverteilten Zufallsstörungen, um die Kondition von Testbeispielen zu schätzen. Stellen Sie Ihre Beobachtungen den Konditionsschätzungen aus MATLAB gegenüber. Experimentieren Sie auch mit schlechtkonditionierten Matrizen, z. B. Hilbertmatrizen.

Projekt II/2: LINEARER AUSGLEICH

Thema 1: Vergleich zwischen Gaußschen Normalgleichungen und QR-Zerlegung

Bekanntlich ist bei der numerischen Lösung eines Ausgleichsproblems,

$$Ax = b, \quad A \in \mathbb{R}^{m\times n}, \quad x \in \mathbb{R}^n, \ b \in \mathbb{R}^m,$$
$$\|Ax - b\|_2 \Rightarrow \text{Min!},$$

die QR-Zerlegung den Gaußschen Normalgleichungen in Hinblick auf numerische Stabilität überlegen. In den folgenden Teilprojekten soll dieser Aspekt anhand von Testbeispielen illustriert werden. Die Matrizen der Testbeispiele sollen aus ihrer Singulärwertzerlegung konstruiert werden. Diese Konstruktion ermöglichst es die Kondition der Matrizen zu steuern und mit Hilfe der Pseudoinversen die Referenzlösung zu erzeugen.

- Teilprojekt 1.1:
  Untersuchen Sie die Unterschiede in der Rundungsfehlersensitivität der Normalgleichungen und der QR-Zerlegung in praxisrelevanten Fällen, in denen die rechte Seite b fast im Bildraum von A liegt, und in Fällen wo wesentliche Anteile von b im Kern von A^T sind.

- Teilprojekt 1.2:
  Untersuchen Sie wie sich die Anzahl m der Gleichungen des Ausgleichsproblems auf die Stabilitätsunterschiede auswirkt. Dazu konstruieren Sie Testbeispiele mit der gleichen Spaltenanzahl n, mit den gleichen Singulärwerten, und mit der gleichen Lage von b bezüglich des Bildraumes von A, d.h. variieren Sie *nur* die Größe m.

- Teilprojekt 1.3:
  In MATLAB gibt es Varianten der QR-Zerlegung zur Lösung rangdefizienter Ausgleichsprobleme, $r(A) < n$. Rangdefiziente Testbeispiele sind über die Singulärwertzerlegung leicht zu konstruieren. Beobachten Sie dabei im speziellen, ob jene Fälle, bei denen der kleinste nicht verschwindende Singulärwert nahe Null liegt, besonders rundungsfehleranfällig sind.

Thema 2: Konvergenzeigenschaften linearer Ausgleichsprobleme

Die Daten von Ausgleichsproblemen erhält man oft aus Messungen, wobei man öfter mißt als es notwendig wäre um die gesuchten Größen festzulegen; Ziel dieser scheinbar redundanter Vorgangasweise ist es Meßfehler herauszumitteln. Es stellt sich die Frage, ob Lösungen von Ausgleichsproblemen, die nach und nach durch immer mehr Messungen definiert sind ($m \to \infty$), gegen die gesuchte wahre Lösung konvergieren.

- Teilprojekt 2.1:
  Wählen Sie ein festes Polynom vom Grad n,

  $$P_n(x) := c_0 + c_1 x + c_2 x + \cdots + c_n x^n,$$

  d.h. legen Sie für Ihr Experiment die Koeffizienten $c_i \in \mathbb{R}$ fest. Nehmen Sie an, daß in einer Anwendung die Koeffizienten unbekannt sind und daß sie jemand mit Hilfe von Messungen ermitteln möchte. Führt er m Messungen durch, d.h. ermittelt er für die Stellen x_i die Werte[4] $\tilde{f}_i \approx f_i = P_n(x_i), i = 1, 2, \ldots, m$, so hat er das folgende Ausgleichsproblem zu lösen:

  $$\begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ & & & & \\ \vdots & \vdots & \vdots & & \vdots \\ & & & & \\ 1 & x_m & x_m^2 & \cdots & x_m^n \end{pmatrix} \begin{pmatrix} \tilde{c}_0 \\ \tilde{c}_1 \\ \vdots \\ \tilde{c}_n \end{pmatrix} = \begin{pmatrix} \tilde{f}_1 \\ \tilde{f}_2 \\ \\ \vdots \\ \\ \tilde{f}_m \end{pmatrix}.$$

  Simulieren Sie die Messungen, d.h. erzeugen Sie die $\tilde{f}_i$ Werte, indem Sie die exakten Funktionswerte $f_i = P_n(x_i)$ mit Zufallsstörungen überlagern. Betrachten Sie Gauß-verteilte bzw. gleichverteilte Zufallsstörungen. Untersuchen Sie experimentell, ob für $m \to \infty$ die Lösungen der Ausgleichprobleme gegen die "wahre" Lösung $(c_0, c_1, \ldots, c_n)$ konvergieren. Was beobachten Sie bezüglich Konvergenzgeschwindigkeit, insbesondere für unterschiedliche den Störungen zugrundeliegende Verteilungsfunktionen?

- Teilprojekt 2.2:
  Betrachten Sie die im Projekt 2.1 beschriebene Situation und dieselbe Folge von Ausgleichsproblemen. Anstatt die Ausgleichsprobleme zu lösen, versuchen Sie folgenden, sehr aufwendigen, Lösungsweg um die Werte c_i zu approximieren: Für ein festes m wählen Sie auf alle möglichen Arten $n + 1$ Gleichungen aus, lösen Sie die so entstehenden linearen Gleichungssysteme mit Gaußelimination

[4] Die Schlange in $\tilde{f}$ soll auf die Meßfehler hinweisen.

und nehmen Sie das arithmetische Mittel all dieser Lösungen als Näherung für die gesuchten Werte c_i. Vergleichen Sie für feste Werte von m die Approximationsqualität der so erhaltenen Lösungen mit der Qualität der Lösung der entsprechenden Ausgleichsprobleme.

Projekt II/3: Extrapolation

Dieses Projekt hat zum Ziel, die Zusammenhänge zwischen dem Verfahrensfehler und dem Rechenfehler bei der Approximation der ersten Ableitung und bei der Berechnung des Integrals einer skalaren Funktion zu beleuchten. Dabei soll gelernt werden, wie man stabile Verfahren hoher Ordnung mit Hilfe der Extrapolation entwickelt und implementiert.

Beispiel 1

Die erste Ableitung $y'(t)$ kann durch den einseitigen oder den zentralen Differenzenquotienten

$$D_e(h) = \frac{y(t+h) - y(t)}{h}, \quad D_z(h) = \frac{y(t+h) - y(t-h)}{2h} \tag{1}$$

approximiert werden.

(1a) Konstruieren Sie ein Beispiel das zeigt, dass die Kondition der Aufgabe extrem schlecht ist, d. h., dass der Wert der Ableitung an der Stelle t sehr empfindlich auf kleine additive Störungen in der Datenfunktion y reagiert. Genauer gesagt: Betrachtet man den Wert der ersten Ableitung an der Stelle t der exakten Funktion $y(t) \in C^1[a,b]$ und der gestörten Funktion $\tilde{y}(t) := y(t) + \epsilon(t)$, wobei $\|\epsilon(t)\|_\infty$ als klein angenommen wird, so kann trotzdem $|\tilde{y}'(t) - y'(t)|$ beliebig groß werden. Machen Sie eine Skizze, die die Problematik illustriert.

(1b) Für beide der obigen Näherungsformeln leite man den Ausdruck für den Verfahrensfehler her, indem man entsprechende Glattheit (Differenzierbarkeit) von y voraussetze.

(1c) Für beide Formeln gebe man eine möglichst gute Abschätzung für den Rechenfehler an. Sie können dabei die „$(1+\epsilon)$“ Technik verwenden. Für den Gesamtfehler erhält man daraus eine Schranke,

$$|\text{Gesamtfehler}| \leq |\text{Verfahrensfehler}| + |\text{Rechenfehler}| \leq S(\delta, \mathit{eps}; h),$$

die von der Schranke δ für die Auswertefehler von y, von der Maschinengenauigkeit *eps*und der Schrittweite h abhängt. S hängt noch von der Schranke M_0 für die Werte von y und von der Schranke M_k für eine bestimmte höhere Ableitung $y^{(k)}$ ab.

Bestimmen Sie jene Schrittweite $h^* = h(\delta, \mathit{eps})$, für die S minimal wird. Welches Fehlerniveau kann daher mit $D_e(h)$ bzw. $D_z(h)$ für $\delta = \mathit{eps} = 10^{-8}$ und $M_0 = M_k = 1$ garantiert werden?

Setzt man voraus, dass $y \in C^7[a,b]$ gilt, so ergibt die Taylorreihenentwicklung von $y(t-h)$ und $y(t+h)$ um die Stelle t in D_z eine sogennante asymptotische Fehlerentwicklung für den Verfahrensfehler

$$D_z(h) - y'(t) = h^2 \cdot \frac{y^{(3)}(t)}{3!} + h^4 \cdot \frac{y^{(5)}(t)}{5!} + R_6, \tag{2}$$

wobei $|R_6| \leq h^6 \cdot \frac{M_7}{7!}$, $M_7 = \max_{a \leq t \leq b} |y^{(7)}(t)|$. Diese strukturelle Information über den Fehler kann man wie folgt ausnützen:
Nehmen wir an, dass wir drei Werte von D_z für drei verschiedene Schrittweiten z.B. $h, h/2, h/4$ berechnet haben. Dann gilt nach (2)

$$\begin{aligned}
D_z(h) - y'(t) &= h^2 \cdot \frac{y^{(3)}(t)}{3!} + h^4 \cdot \frac{y^{(5)}(t)}{5!} + R_6, && \text{(3a)}\\
D_z\left(\frac{h}{2}\right) - y'(t) &= \left(\frac{h}{2}\right)^2 \cdot \frac{y^{(3)}(t)}{3!} + \left(\frac{h}{2}\right)^4 \cdot \frac{y^{(5)}(t)}{5!} + R_6, && \text{(3b)}\\
D_z\left(\frac{h}{4}\right) - y'(t) &= \left(\frac{h}{4}\right)^2 \cdot \frac{y^{(3)}(t)}{3!} + \left(\frac{h}{4}\right)^4 \cdot \frac{y^{(5)}(t)}{5!} + R_6 && \text{(3c)}
\end{aligned}$$

und wir können durch Bildung geeigneter Linearkombinationen dieser Gleichungen, was auf die Elimination der h^2- und h^4-Terme auf der rechten Seite hinausläuft, eine Näherung $D_{excel}(h)$ für $y'(t)$ herleiten, für die

$$D_{excel}(h) - y'(t) = \mathrm{O}(h^6)$$

gilt.

2a) Leiten Sie die Formel (2) her.

2b) Bestimmen Sie $D_{excel}(h)$ und ihren Verfahrensfehler.

2c) Welchen Vorteil hat diese Näherungsformel gegenüber $D_z(h)$?

2d) Versuchen Sie für das obige Vorgehen eine Interpretation zu finden. Machen Sie sich dazu bewusst, dass die rechte Seite in

$$D_z(h) \approx y'(t) + h^2 \cdot \frac{y^{(3)}(t)}{3!} + h^4 \cdot \frac{y^{(5)}(t)}{5!}$$

als ein spezielles Polynom 2. Grades in h^2 aufgefasst werden kann.

Beispiel 2

Implementieren Sie die Näherungsformeln $D_e(h)$, $D_z(h)$ und $D_{excel}(h)$ und vergleichen Sie an selbstgewählten Beispielen ihre Qualität. Dazu kann man für eine Reihe von Schrittweiten h vergleichen, welche Genauigkeit mit den einzelnen Formeln zu erreichen ist. Versuchen Sie in diesem Vergleich auch den Aufwand (Anzahl der Funktionsauswertungen in $D_e(h)$, $D_z(h)$ und $D_{excel}(h)$) einzubeziehen.

Basierend auf der Formel $D_{excel}(h)$, entwickeln Sie ein Matlab-Programm zur Berechnung der ersten Ableitung einer skalaren Funktion $f(t)$ and der Stelle $t^\star$. Der Benützer übergibt als Eingangsdaten $f, t^\star$ und die Genauigkeitsforderung für den absoluten Fehler, TOL, das Programm liefert eine Näherung $D_h(t^\star)$ für $f'(t^\star)$ und eine Fehlerschätzung für den absoluten (und den relativen) Fehler von $D_h(t^\star)$, falls

$$|D_h(t^\star) - f'(t^\star)| \leq TOL$$

gilt, oder eine Fehlermeldung, falls die Genauigkeitsforderung nicht erfüllt werden konnte. Überlegen Sie wie Sie den absoluten Fehler schätzen und die Schrittweite anpassen wollen. Testen Sie Ihr Programm an mehreren Funktionen verschiedener Schwierigkeit.

Beispiel 3

Die Schranken für den Verfahrensfehler der Näherungformel $D_e(h)$, $D_z(h)$ und $D_{excel}(h)$ haben eine gewisse Struktur, die man grob wie folgt darstellen kann:

$$|D(h) - y'(t)| \leq \text{ Kenngrößen des Problems} \cdot h^p, \;\; p \geq 1, \tag{4}$$

wobei sich hinter den "Kenngrößen" des Problems Schranken für höhere Ableitungen von y verbergen. Beobachten Sie die Ordnung p des Verfahrens anhand der Ableitungswerte der Funktion $y(x) = e^x$. Dazu werte man die obigen Näherungsformel für eine Reihe feiner werdendender Schrittweiten h aus und berechne für jede dieser Schrittweiten den Verfahrensfehler $|D(h) - y'(t)|$. Aus dem Ansatz

$$|D(h) - y'(t)| \approx c \cdot h^p, \;\; h \to 0,$$

kann man für jeweils zwei aufeinanderfolgende Schrittweiten eine Schätzung für die, von h unabhängige, Fehlerkonstante c und die Ordnung p ausrechnen. Experimentieren Sie mit Funktionen y die weniger glatt sind als e^x (Werte der höheren Ableitungen sind groß, es treten Unstetigkeiten höherer Ableitungen auf). Was beobachten Sie insbesondere in jenen Fällen, wo die Ableitungen, die in (4) auftreten, in der Nähe von t groß oder unstetig sind?

Beispiel 4

Näherungsformeln für die erste Ableitung $y'(t)$ können nach dem folgenden allgemeinen Prinzip gewonnen werden: Man interpoliert die Funktion y an den Stellen

$$t - lh,\, t - (l-1)h,\, \ldots, t,\, \ldots,\, t + (n-1)h,\, t + nh,\;\; l, n \in \mathbb{N}$$

mit einem Polynom P vom Grad $l + n$ und nimmt die Ableitung $P'(t)$ als Approximation für $y'(t)$. Diese Vorgangsweise führt im Falle $l = 0$ und $n = 1$ zum einseitigen Differenzenquotienten $D_e(h)$, der die erste Ableitung auf $O(h)$-Niveau approximiert, und im Falle $l = 1$ und $n = 1$ zum zentralen Differenzenquotienten $D_z(h)$, der eine $O(h^2)$ Approximation für die erste Ableitung ist, vgl. (1).
Leiten Sie eine Näherungsformel für $y'(t)$ der Ordnung $O(h^6)$ her, indem Sie die Werte $y(t + kh)$, $k \in \mathbb{Z}$, $-3 \leq k \leq 3$, interpolieren. Geben Sie den Verfahrensfehler dieser Formel an und vergleichen Sie ihn mit dem Verfahrensfehler von D_{excel}, siehe Teilprojekt 2a).
Implementieren Sie diese Formel und vergleichen Sie experimentell deren numerische Stabilität mit den Stabilitätseigenschaften von D_{excel}. Plotten Sie zu diesem Zweck die entsprechenden Fehlerverläufe als Funktion von h.

Beispiel 5

Ein sehr simples numerisches Integrationsverfahren ist die gemäß

$$\begin{aligned}\int_a^b f(x)dx \approx T_h \;&:=\; \frac{h}{2}\Big[f(a) + f(a+h)\Big] + \frac{h}{2}\Big[f(a+h) + f(a+2h)\Big] + \ldots \\ &\quad \ldots + \frac{h}{2}\Big[f(b-h) + f(b)\Big] = \\ &=\; h\Big[\frac{1}{2}f(a) + f(a+h) + f(a+2h) + \ldots + \frac{1}{2}f(b)\Big] \qquad (5)\end{aligned}$$

definierte Trapezregel, für die die folgende a-priori Fehlerschranke für den Verfahrensfehler gilt:

$$\left|T_h - \int_a^b f(x)dx\right| \leq \frac{b-a}{2}\, M_2\, h^2, \qquad (6)$$

mit $f \in C^2[0,1]$ und $M_2 := \max_{a \leq x \leq b} |f''(x)|$. Die Trapezregel hat die Ordnung 2, d.h., für zweimal stetig differenzierbare Integranden f geht der Fehler wie h^2 gegen Null (für hinreichend kleine Schrittweiten h geht bei Halbierung der Schrittweite

der Fehler etwa auf $\frac{1}{4}$ zurück).

In dem Projekt soll nun der Verfahrensfehler - abhängig von der Schrittweite h - bezüglich dreier numerischer Näherungsverfahren beobachtet werden. Diese Verfahren werden zunächst bezüglich des Standardintervalls $[a, b] = [0, 1]$ dargestellt, bevor die Verallgemeinerung auf den allgemeinen Fall $[a, b]$ angedeutet wird.

Trapezsummenextrapolation

Für die Standardgitter $h = \frac{1}{4}$ ($t_0 = a = 0$, $t_1 = \frac{1}{4}$, $t_2 = \frac{1}{2}$, $t_3 = \frac{3}{4}$, $t_4 = b = 1$), dann $h = \frac{1}{8}$ ($t_0 = a = 0$, $t_1 = \frac{1}{8}$, $t_2 = \frac{1}{4}$,...), dann $h = \frac{1}{16}$ und schließlich noch $h = \frac{1}{32}$ sollen zunächst die Trapezsummen $T_{\frac{1}{4}}$, $T_{\frac{1}{8}}$, $T_{\frac{1}{16}}$, $T_{\frac{1}{32}}$ berechnet werden und anschließend noch weitere Näherungswerte $\hat{T}_h$ und $\tilde{T}_h$ gemäß

h	T_h	$\hat{T}_h$	$\tilde{T}_h$
$\frac{1}{4}$	$T_{\frac{1}{4}}$	$\hat{T}_{\frac{1}{4}} := \frac{4}{3}T_{\frac{1}{8}} - \frac{1}{3}T_{\frac{1}{4}}$	$\tilde{T}_{\frac{1}{4}} := \frac{16}{15}\hat{T}_{\frac{1}{8}} - \frac{1}{15}\hat{T}_{\frac{1}{4}}$
$\frac{1}{8}$	$T_{\frac{1}{8}}$	$\hat{T}_{\frac{1}{8}} := \frac{4}{3}T_{\frac{1}{16}} - \frac{1}{3}T_{\frac{1}{8}}$	$\tilde{T}_{\frac{1}{8}} := \frac{16}{15}\hat{T}_{\frac{1}{16}} - \frac{1}{15}\hat{T}_{\frac{1}{8}}$
$\frac{1}{16}$	$T_{\frac{1}{16}}$	$\hat{T}_{\frac{1}{16}} := \frac{4}{3}T_{\frac{1}{32}} - \frac{1}{3}T_{\frac{1}{16}}$	
$\frac{1}{32}$	$T_{\frac{1}{32}}$		

Fig. 1

Bezüglich der Ordnungen gilt

1. für die Trapezsummen (vgl. (5)):

$$T_h - \int_a^b f(x)dx = O(h^2),$$

d.h. wenn man in dem Dreiecksschema (vgl. Fig.1) in den Näherungswerten T_h, $\hat{T}_h$ und $\tilde{T}_h$ für das Integral die Fehler selbst einträgt, sollte man in der T_h-Spalte nach unten (bei Halbierung von h) allmählich immer besser einen Rückgang des Fehlers von T_h auf ein Viertel beobachten;

2. für die erste Extrapolationsspalte gilt:

$$\hat{T}_h - \int_a^b f(x)dx = O(h^4),$$

d.h. im Fehlertableau sollte man in der zweiten Spalte nach unten einen Rückgang des Fehlers auf $\frac{1}{16}$ beobachten;

3. für die dritte Extrapolationsspalte gilt:

$$\tilde{T}_h - \int_a^b f(x)dx = O(h^6),$$

d.h. im Fehlertableau sollte man in der dritten Spalte nach unten einen Rückgang des Fehlers auf $\frac{1}{64}$ beobachten.

Wenn man statt dem speziellen Integrationsintervall $[a, b] = [0, 1]$ an den allgemeinen Fall denkt, bleibt das Tableau unverändert gültig, man hat es jetzt nicht mit den Gittern $h = \frac{1}{4}$ ($t_0 = 0$, $t_1 = \frac{1}{4}$, $t_2 = \frac{1}{2}$, ...), $h = \frac{1}{8}$, $h = \frac{1}{16}$, $h = \frac{1}{32}$ zu tun, sondern stattdessen mit $h = \frac{b-a}{4}$ ($t_0 = a$, $t_1 = a + \frac{b-a}{4}$, ...), $h = \frac{b-a}{8}$, $h = \frac{b-a}{16}$, $h = \frac{b-a}{32}$. Statt $T_{\frac{1}{4}}$, $T_{\frac{1}{8}}$, ... treten im Tableau die Größen $T_{\frac{b-a}{4}}$, $T_{\frac{b-a}{8}}$, ... auf.

Überlegen Sie Steuerungsmaßnahmen für ein Quadraturpaket basierend auf der Trapezsummenextrapolation. Formulieren Sie einen numerischen Integrationsalgorithmus und programmieren Sie diesen. Testen Sie Ihr Programm anhand von Testbeispielen unterschiedlicher Schwierigkeit. Vergleichen Sie dei Qualität Ihrer Software mit der geeigneten Matlab Standardsoftware.

Projekt III/1: Ersatzfunktionen

- Teilprojekt 1:
  Aufgrund des Alternantensatzes ist es klar, daß die Bestapproximierende im Sinne von Chebyshev auch Interpolationsfunktion ist. Folgende Gegenüberstellung liegt daher nahe:

  (i) Lagrange-Interpolation auf äquidistantem Gitter mit einem Interpolationspolynom vom Grad n,

  (ii) Lagrange-Interpolation mit einem Polynom vom Grad n, wobei die Knoten die transformierten Nullstellen des Chebyshev-Polynoms vom Grad $n+1$ sind,

  (iii) Das im Sinne von Chebyshev bestapproximierende Polynom[5] vom Grad n.

  Es wäre auch interessant die Qualität der Approximation bezüglich verschiedener Normen zu untersuchen und deshalb auch die unten stehenden Approximation einzubeziehen:

  (iv) Die abgebrochene Fourierreihe bis zum Grad n basierend auf den Chebyshev-Polynomen.

  (v) Die abgebrochene Fourierreihe bis zum Grad n basierend auf den Legendre-Polynomen.

  Für verschiedene Testfunktionen $f(x)$, $x \in [a,b]$ und unterschiedliche Polynomgrade vergleichen Sie diese möglichen Konstruktionen von Ersatzfunktionen, indem Sie sowohl die Fehlerverläufe plotten als auch die maximale Abweichung angeben. Insbesondere soll beobachtet werden, wie in Abhängigkeit von der Intervallänge und dem Polynomgrad das Maximum der Abweichung von (i) bis (iii) abnimmt.

- Teilprojekt 2:
  Immer wenn man mit polynomialen Interpolationsfunktionen als Ersatzfunktionen arbeitet, muß man entscheiden ob man auf dem gesamten Intervall ein einziges Polynom hohen Grades verwendet oder stückweise Polynome niedereren Grades heranzieht. Diese beiden Möglichkeiten sollen verglichen werden:

  (i) Durch theoretische Überlegungen basierend auf der bekannten Darstellung des Interpolationfehlers[6].

  (ii) Durch Plotten entsprechender Fehlerverläufe. Dabei realisieren Sie folgende Experimente: Gehen Sie von dem Definitionsintervall $[a,b]$ aus und interpolieren Sie die Daten $(x_i, f(x_i))$, $x_i = a + i\frac{b-a}{n}$, $i = 0,1,\ldots,n$ mit einem Polynom vom Grad n. Verfeinern Sie das Gitter durch Halbierung der Schrittweite, d.h. betrachten Sie den Datensatz $(x_i, f(x_i))$, $x_i = a + i\frac{b-a}{2n}, i = 0,1,\ldots,2n$ und interpolieren Sie stückweise mit zwei Polynomen vom Grad n. Führen Sie diesen Prozeß fort, indem Sie zu stückweise Interpolation mit immer mehr Polynomen mit festem Grad n übergehen. Die gleichen Datensätze wie im Falle der stückweisen Interpolation interpolieren Sie auch mit jeweils nur einem Polynom, d.h. betrachten Sie eine Folge von Interpolationspolynomen mit steigendem Grad, und zwar n, $2n$, $4n$ u.s.w. Stellen Sie für obige Datensätze die Fehlerverläufe der stückweisen Interpolation jener mit einem Interpolationspolynom gegenüber. Führen Sie analoge

[5] Eine Sammlung von Bestapproximierenden wird in der Sprechstunde zur Verfügung gestellt.

[6] Versuchen Sie auch in der mathematischen Literatur Konvergenzaussagen zu finden, die sich auf folgende Situation beziehen: Es wird $f(x)$ auf einem festen Intervall $[a,b]$ nach und nach mit einem Interpolationspolynom immer höheren Grades interpoliert.

Experimente durch, indem Sie statt äquidistanten Interpolationsknoten die entsprechend transformierten Nullstellen der Chebyshev Polynome wählen. Beachten Sie, daß im Gegensatz zu Interpolation mit äquidistanten Knoten jetzt die Datensätze für die stückweise Interpolation und die Interpolation mit einem Polynom hohen Grades nicht übereinstimmen.

- Teilprojekt 3:
  Beim Vergleich von Interpolation basierend auf Spline-Funktionen mit stückweiser Lagrange-Interpolation fällt auf, daß bei gleicher asymptotischer Approximationsqualität der Aufwand bei der Aufstellung die Spline-Funktionen ungleich höher ist. Bei der üblichen Definition von Spline-Funktionen ist das Lösen von linearen Gleichungssystemen mit Bandmatrizen notwendig. Um diesen Nachteil zu vermeiden, aber gleichzeitig die attraktiven Glattheitseigenschaften von Spline-Funktionen aufrechtzuerhalten, wurden eigene Typen von Interpolationsfunktionen entwickelt, z. B. Spline-Funktionen, deren Basisfunktionen kompakte Träger haben (B-Splines, u.a.) oder Akima-Interpolation. Geben Sie unter Heranziehung mathematischer Literatur einen Überblick über

  – verschiedene Definitionsmöglichkeiten von Spline-Funktionen, auch bez. unterschiedlicher Randbedingungen und von Akima-Interpolierenden,
  – den algorithmischen Aufwand bei der Aufstellung und Auswertung,
  – ihre asymptotische Approximationqualität und Glattheitseigenschaften.

  Versuchen Sie die in (i) bis (iii) aufgelisteten Eigenschaften in Hinblick auf verschiedene Anwendungssituationen zu bewerten. Für geeignet gewählte Datensätze stellen Sie Spline-Funktionen und Akima-Interpolierende auf und vergleichen Sie Ihre Plots mit der den Daten zugrunde liegenden Funktion f.

- Teilprojekt 4:
  Vergleichen Sie die polynomiale Lagrange-Interpolation mit der rationalen Interpolation, indem Sie gleiche Datensätze $(x_i, f(x_i))$, $i = 0, 1, \ldots, n$ mit Lagrange-Interpolationspolynomen vom Grad n und rationalen Interpolationsfunktionen mit unterschiedlichem Zähler- und Nenner-Grad approximieren. Dabei betrachten Sie Datensätze, die durch Abtasten von glatten Funktionen und von Funktionen mit Polstellen gewonnen werden. Plotten Sie die Fehlerverläufe, und beobachten Sie die asymptotischen Approximationseigenschaften der Interpolationsfunktionen bezüglich feiner werdenden Gittern.

- Teilprojekt 5:
  Eine wichtige Aufgabe der graphischen Datenverarbeitung besteht darin, zu n gegebenen Punkten in der Ebene eine geschlossenen Kurve zu konstruieren, die diese Punkte durchläuft. Betrachten Sie eine Punktfolge (x_i, y_i), $i = 1(1)n$ mit $x_1 = x_n$, $y_1 = y_n$.

  Eine mögliche Vorgangsweise besteht darin, die Länge des Polygonzuges s_i zwischen den Punkten (x_1, y_1) und (x_i, y_i) für alle $i \geq 1$ als Parameter zu nehmen, was näherungsweise der Bogenlängenparametrisierung entspricht, d.h. die Größen s_i sind gemäß

$$s_1 = 0, \quad s_i = \sum_{k=2}^{i} \sqrt{(y_k - y_{k-1})^2 + (x_k - x_{k-1})^2}, \quad i = 2(1)n$$

definiert. Man bildet dann die periodischen kubischen Splinefunktionen zu den Daten

$$t_i = s_i, \quad S_x(t_i) = x_i, \quad i = 1(1)n,$$
$$t_i = s_i, \quad S_y(t_i) = y_i, \quad i = 1(1)n,$$

und gewinnt damit die Parameterdarstellung für die Kurve.

- Überlegen Sie verschiedene Varianten der eben beschriebenen Idee (andere Parametrisierungen der Kurve als mit der Länge des Polygonzugs und andere Interpolationsfunktionen als kubische Splines) und implementieren Sie einige jener Varianten, die Sie für besonders attraktiv halten.
- An selbstgewählten Beispielen stellen Sie diese Varianten vergleichend gegenüber. Untersuchen Sie den Einfluß der Interpolationsmethode und der Parametrisierung. Ziehen Sie insbesondere auch die periodische Akima-Interpolation und die trigonometrische Interpolation in einigen Varianten heran. Untersuchen Sie auch experimentell die Kondition, indem Sie den Einfluß von Störungen der Punktkoordinaten (x_i, y_i), beobachten. Betrachten Sie auch die Punkte (x_i, y_i), die durch Abtasten geschlossener Kurven gewonnen werden und vergleichen Sie ihre Interpolationsfunktionen mit den Kurven. Beobachten Sie in diesem Fall allfällige Konvergenzeigenschaften bezüglich steigender Punktezahl.

- Teilprojekt 6:
  Stellen Sie anhand geeigenter Testfunktionen durch Plotten der Approximationsfehlerverläufe verschiedene Varianten von Bestapproximierenden gegenüber. Unter "Varianten" verstehen wir hier verschiedene Fälle, die sich ergeben, wenn man die Norm der Abweichung unterschiedlich definiert, z. B. über die Maximumnorm (Chebyshev-Approximation), oder über Skalarproduktnormen für verschiedene Gewichtsfunktionen (abgebrochene Fourierreihe als Bestapproximierende).

Projekt III/2: Interpolation [7]

Dieses Projekt hat zum Ziel, verschiedene Wege zur Konstruktion von Ersatzfunktionen zu zeigen. Dabei soll gelernt werden, wie man gute Approximationsqualität mit möglichst geringem Aufwand garantieren kann.

Beispiel 1

Aufgrund des Alternantensatzes ist es klar, dass die Bestapproximierende im Sinne von Chebyshev auch Interpolationsfunktion ist. Folgende Gegenüberstellung liegt daher nahe:

(i) Lagrange-Interpolation auf äquidistantem Gitter mit einem Interpolationspolynom vom Grad n,

(ii) Lagrange-Interpolation mit einem Polynom vom Grad n, wobei die Knoten die transformierten Nullstellen des Chebyshev-Polynoms vom Grad $n+1$ sind,

(iii) Das im Sinne von Chebyshev bestapproximierende Polynom[8] vom Grad n.

Es wäre auch interessant die Qualität der Approximation bezüglich verschiedener Normen zu untersuchen und deshalb auch die unten stehenden Approximation einzubeziehen:

(iv) Die abgebrochene Fourierreihe bis zum Grad n basierend auf den Chebyshev-Polynomen.

(v) Die abgebrochene Fourierreihe bis zum Grad n basierend auf den Legendre-Polynomen.

Für verschiedene Testfunktionen $f(x)$, $x \in [a,b]$ und unterschiedliche Polynomgrade vergleichen Sie diese möglichen Konstruktionen von Ersatzfunktionen, indem Sie sowohl die Fehlerverläufe plotten als auch die maximale Abweichung angeben. Insbesondere soll beobachtet werden, wie in Abhängigkeit von der Intervallänge und dem Polynomgrad das Maximum der Abweichung von (i) bis (iii) abnimmt.

Beispiel 2

Immer, wenn man mit polynomialen Interpolationsfunktionen als Ersatzfunktionen arbeitet, muß man entscheiden, ob man auf dem gesamten Intervall ein einziges Polynom hohen Grades verwendet oder stückweise Polynome niedrigeren Grades heranzieht. Diese beiden Möglichkeiten sollen verglichen werden.

Eine Standardfunktion $f(x)$ soll auf einem geeignet gewähltem Intervall $[a,b]$ approximiert werden. Die Approximation soll eine auf Chebyshev-Knoten basierende Interpolation sein. Dabei sollen zwei Varianten untersucht werden.

(i) Die Funktion f wird auf dem gesamten Intervall mit einer vorgegebenen Toleranz TOL approximiert, d.h. die Approximierende $\phi(x)$ soll $\forall x \in [a,b],\ |f(x)-\phi(x)| < TOL$, erfüllen.

(ii) Die Funktion f wird stückweise mit derselben Toleranz TOL approximiert. Dabei soll die Anzahl der Teilintervalle von $[a,b]$ variabel sein.

[7] Jedes Beispiel kann von 2 Personen bearbeitet werden.

[8] Eine Sammlung von Bestapproximierenden wird in der Sprechstunde zur Verfügung gestellt.

Für eine selbstgewählte Standardfunktion führen Sie die folgenden Überlegungen und Experimente durch:

- Mit Hilfe der Abschätzung des Verfahrensfehlers bei der Interpolation mit dem Chebyshev-Polynom vom Grad n auf dem gesamten Intervall $[a, b]$ setze man n in Beziehung mit dem Toleranzparameter TOL. Berechnen Sie $n(TOL)$ für $TOL = 10^{-3}$, 10^{-6}, 10^{-9}.

- Passend zu dem sich oben ergebenden Grad $n(TOL)$ wählen Sie einen oder mehrere Grade $n_{low}(TOL)$ und versuchen Sie abzuschätzen mit wieviel äquidistanten Teilintervallen von $[a, b]$ Sie die gleiche Toleranzanforderung erfüllen können. Beachten Sie dabei, dass die sich ergebende stückweise Interpolierende an den Enden der Teilintervalle nicht mehr stetig differenzierbar ist.

- Überzeugen Sie sich experimentell, ob Ihre Vorgangsweise erfolgreich war. Plotten Sie dazu die Fehlerverläufe der Approximierenden (und ihrer Ableitung im Vergleich zu f') und vergleichen Sie diese mit TOL.

Beispiel 3

Analoge Überlegungen und Experimente wie bei Teilprojekt 2, jedoch für die erste Ableitung der Interpolationsfunktionen im Vergleich mit der Ableitung von f. Statt der Variante mit einem Polynom von Grad n soll eine weitere Variante der stückweisen Interpolation betrachtet werden, nämlich jene, bei der an den Enden der Teilintervalle von $[a, b]$ auch die Ableitungswerte von f vorgeschrieben werden.

Beispiel 4 [9]

In dieser Aufgabe geht es um die Näherung für die Funktion $f(x) = \ln x$ auf dem Intervall $[\epsilon, a]$ mit $\epsilon << 1$ und $a \geq 1$. Die Funktion f soll mit der Toleranz $TOL = 10^{-2}$, 10^{-4} und 10^{-6} approximiert werden, d.h. die Approximierende $\phi(x)$ soll $\forall x \in [a, b]$, $|f(x) - \phi(x)| < TOL$ erfüllen. Dazu sollen die folgenden alternativen Vorgangsmöglichkeiten verglichen werden:

- Approximation mit einem durchgehenden Polynom vom Grad n und mit den äquidistanten Interpolationsknoten.

- Approximation mit einem Polynom vom Grad n und mit den auf das Intervall $[\epsilon, a]$ transformierten Chebyshev-Knoten.

- Approximation mit einem Polynom vom Grad n und mit Interpolationsknoten, deren Dichte gegen das linke Intervallende exponentiell zunimmt.

- Statt mit einem Polynom sollen die obigen drei Varianten mit einer rationalen Funktion der Form
$$r(x) = \frac{p_{n-k}(x)}{x^k},$$
betrachtet werden, wobei im Zähler ein Polynom vom Grad $n - k$ steht. Untersuchen Sie, für welchen Wert von $1 \leq k \leq n$ Sie die besten Resultate erzielen können.

Plotten Sie die Fehlerverläufe der verschiedenen Approximationen und vergleichen Sie sie mit TOL.

[9] Kann auch von 3 Personen bearbeitet werden.

Projekt III/3: Approximation von Standardfunktionen

Dieses Projekt hat zum Ziel, einige der so genannten mathematischen Standardfunktionen wie

$$\sin x, \quad \ln x, \quad e^x, \ldots$$

numerisch zu approximieren.

Diese Standardfunktionen sind in den meisten Implementierungen der gängigen Programmiersprachen fix und fertig bereitgestellt, manche sogar bereits auf Hardwareebene, z.B. als Maschineninstruktion für $\sqrt{x}$. In den folgenden Beispielen sollen davon unabhängige numerische Approximationen gebastelt werden. Diese Aufgabenstellung stellt ein schönes Übungsgelände für verschiedene numerische Techniken dar, wie z.B. Summation, Argumentreduktion, partielle doppelt Genauigkeit etc., und man muss mit den dabei auftretenden Rechenfehlereffekten zurechtkommen. Außerdem ist es denkbar, dass man auf exotischeren Rechnerarchitekturen tatsächlich gezwungen ist, eigene Routinen für die Standardfunktionen zu entwickeln.

Manche der relevanten numerischen Techniken, wie Reihenentwicklung, sind bei jedem der einzelnen Beispiele von Bedeutung; manches andere ist Beispiel-spezifisch. Für die Approximation werden jeweils Polynome verwendet, die aus abgebrochenen Taylorreihen gewonnen werden. Diesem Zugang wird die polynomiale Approximationen gegenübergestellt.

Zu beachten:

- Grafische Visualisierung der Ergebnisse ist sinnvoll und erwünscht.
- Der subnormale Gleitpunktzahlenbereich soll bei diesem Projekt grundsätzlich außer Betracht bleiben.
- Mit $\mathbb{F}$ bezeichnen wir den zugrundeliegenden Bereich von Gleitpunktzahlen. Als Rechnerarithmetik wird grundsätzlich doppelt genaue IEEE-Arithmetik (d.h. etwa `double` in MATLAB auf einem PC, d.h. ca. 15 signifikante Dezimalstellen) vorausgesetzt, dies ist aber nicht wirklich wesentlich.
- Mit *eps* bezeichnen wir die relative Maschinengenauigkeit für diese Arithmetik. Die in den Beispielen zu konstruierenden Approximationen sollen möglichst bis auf einen relativen Fehler der Größenordnung 100 *eps* genau sein; aus Gründen des Rechenaufwandes kann aber auch eine moderatere Fehlertoleranz K *eps* gewählt werden, mit $K > 100$.

Beispiel 1: Exponentialfunktion

Es soll eine Funktionsprozedur für die Exponentialfunktion $\exp(x) = e^x$ entwickelt werden. Die Funktion wird dabei durch die ersten $n+1$ Summanden ihrer Taylorreihe bezüglich der Stelle $x_0 = 0$ ersetzt:

$$e^x \approx \sum_{i=0}^{n} \frac{x^i}{i!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \ldots + \frac{x^n}{n!}$$

(1.1) Man formuliere die Aufgabenstellung als numerisches Problem für einen konkreten Rechner:

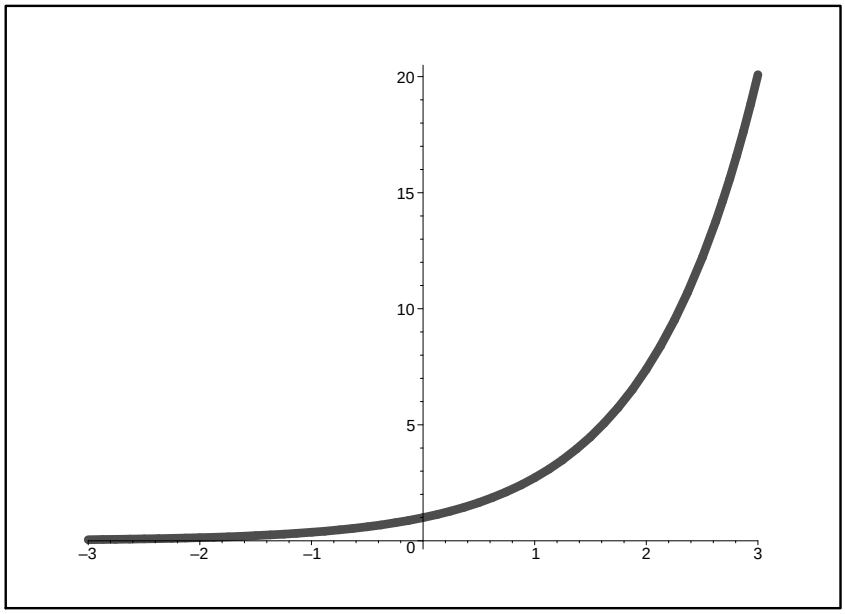


Abbildung 1: Exponentialfunktion e^x

- Genauigkeitsforderung: Es wird eine relative Genauigkeit von mindestens 100 *eps* angestrebt.
- Für welchen Bereich von x-Werten ist es (bei gegebener Arithmetik) überhaupt sinnvoll, eine Approximaton zu berechnen ('numerischer Definitionsbereich')? Welche Ausnahmesituation(en) liegt(liegen) außerhalb dieses Bereiches vor? Gibt es Symmetrien, so dass dieser Bereich weiter reduziert werden kann?
- Für welche speziellen Argumente $x \in \mathbb{F}$ ist der Funktionswert trivial?

(1.2) Wie viele Glieder der Taylorreihe muss man mindestens summieren, damit man die gewünschte Genauigkeit erreicht? Man führe eine Abschätzung für beliebige Argumente x aus dem numerischen Definitionsbereich durch und stelle diese Information in geeigneter Weise grafisch dar.

Dabei wird exakte Rechnung vorausgesetzt: Die Rechenfehler, die beim Aufsummieren der Reihe entstehen, werden vernachlässigt. Der Reihenrest,

$$e^x - \sum_{i=0}^{n} \frac{x^i}{i!} \;=\; \frac{x^{n+1}}{(n+1)!}\, e^{\theta x} \qquad \text{mit} \quad \theta \in [0,1],$$

entspricht dem absoluten Verfahrensfehler.

(1.3) Für einige konkrete Argumente $x \in \mathbb{F}$,

$$x \;=\; 1,\; -1,\; 10,\; -10,\; 20,\; -20,\; 100,\; -100$$

führe man die Summation dieser endlichen Reihen in natürlicher Reihenfolge durch und ermittle die relativen Fehler der so erhaltenen Resultate. (Als Bezugsgröße verwendet man den von der Standardprozedur gelieferten Wert.)

Wird eine relative Genauigkeit erreicht, die dem unter (1.2) angegebenen Verfahrensfehler entspricht?

(1.4) Man gebe eine Begründung für die beobachteten Phänomene und vergleiche die Qualität der obigen Resultate mit jener, die man bei Summation in umgekehrter Reihenfolge erhält. Hat man durch diese Maßnahme die Fehler bei den betragsgroßen negativen Argumenten in den Griff bekommen?

(1.5) Man entwickle folgende praktikable Variante, die auf dem Prinzip der *Argumentreduktion* beruht: Man approximiert die Funktion nur auf einem kleinen Standardintervall und verwendet spezielle Eigenschaften der Exponentialfunktion ($e^{x+y} = e^x \cdot e^y$), um andere Argumente auf diesen Fall zurückzuführen.

Im folgenden bezeichne $\mathbf{ln2} \in \mathbb{F}$ die numerische Approximation von $\ln 2$ in der gegebenen Arithmetik.

a) Ist $x \in [0, \mathbf{ln2}]$, so wird e^x mit dem obigen Algorithmus ermittelt, wobei die Taylorreihe in der umgekehrten Reihenfolge summiert wird. (Es empfiehlt sich ein zusätzlicher Summand als 'Genauigkeitsreserve'.) Speziell ist $e^0 = 1$.

b) Ist $x > \mathbf{ln2}$, dann stellt man x dar als

$$x = k\,\mathbf{ln2} + r, \qquad \text{mit} \quad k \in \mathbb{N}, \quad 0 \le r < \mathbf{ln2},$$

und verwendet die Identität

$$e^x = e^{k \ln 2 + r} = 2^k\, e^r.$$

Man berechnet also e^r mit der Prozedur aus a) und führt anschließend die Multiplikation mit 2^k durch. (Liegt eine Arithmetik mit der Basis $b = 2$ vor, so erfolgt diese Multiplikation sogar rechenfehlerfrei. Für eine andere Basis b genügt es, in a) das rechte Intervallende **ln2** durch $\ln b$ zu ersetzen, um den gleichen Effekt zu erzielen.)

c) Ist $x < 0$, so berechnet man zuerst $e^{|x|}$ nach a) oder b) und bildet anschließend $e^x = 1/e^{|x|}$.

(1.6) Mit dieser Variante berechne man neuerlich die Werte der Exponentialfunktion für $x = 1,\ -1,\ 10,\ -10,\ 20,\ -20,\ 100,\ -100$. Hat sich die Qualität der Werte für große negative Exponenten verbessert?

(1.7) Vergleichen Sie die Approximationsmethode, die unter (1.5) entwickelt wurde mit der folgenden Alternative:

(i) Lagrange-Interpolation zu äquidistanten Knoten,

(ii) Lagrange-Interpolation zu Chebyshev-Knoten.

Dabei wird wieder die relative Genauigkeit von 100 *eps* angestrebt und mit dem Standardintervall $x \in [0, \mathbf{ln2}]$, vgl. (1.5), gearbeitet. Vergleichen Sie die Qualität der drei Approximationfuntionen, indem Sie, sowohl die Fehlerverläufe plotten, als auch die maximalen Fehler angeben. Vergleichen Sie auch den Aufwand, der zur Konstruktion der Erstzfunktionen notwendig ist (Anzahl der Funtionsauswertungen).

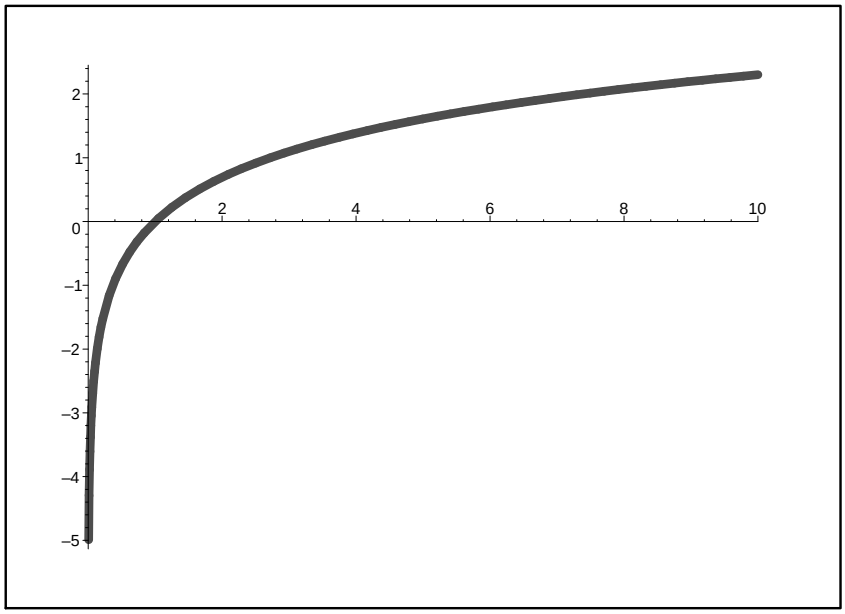


Abbildung 2: Natürlicher Logarithmus $\ln x$

Beispiel 2: Natürlicher Logarithmus

Es soll eine Funktionsprozedur für den natürlichen Logarithmus $\ln x$ (Inverse der Exponentialfunktion, definiert für $x > 0$) entwickelt werden. Die Funktion soll dabei durch die ersten n Summanden ihrer Taylorreihe bezüglich der Stelle $x_0 = 1$ ersetzt werden: Für $x = 1 + \delta$ ist

$$\ln(1+\delta) \approx \sum_{i=1}^{n} (-1)^{i-1} \frac{\delta^i}{i} = \delta - \frac{\delta^2}{2} + \frac{\delta^3}{3} - \frac{\delta^4}{4} + \ldots + (-1)^{n-1} \frac{\delta^n}{n},$$

diese Reihe konvergiert jedoch nur für $0 < x \leq 2$, d.h. für $|\delta| \leq 1$, $\delta \neq -1$ (siehe unten).

(2.1) Man formuliere die Aufgabenstellung als numerisches Problem für einen konkreten Rechner:

- Genauigkeitsforderung: Es wird eine relative Genauigkeit von mindestens 100 *eps* angestrebt.
- Für welchen Bereich von x-Werten ist es (bei gegebener Arithmetik) überhaupt sinnvoll, eine Approximaton zu berechnen ('numerischer Definitionsbereich')? Welche Ausnahmesituation(en) liegt(liegen) außerhalb dieses Bereiches vor? Gibt es Symmetrien, so dass dieser Bereich weiter reduziert werden kann?
- Für welche speziellen Argumente $x \in \mathbb{F}$ ist der Funktionswert trivial?

(2.2) Wir betrachten zunächst Argumente $0 < x \leq 1$, d.h. $\delta \in (-1, 0]$, für die die obige Taylorreihe konvergiert. Wie viele Glieder der Taylorreihe muss man mindestens summieren, damit man die gewünschte Genauigkeit erreicht? Man führe eine Abschätzung für beliebige Argumente $x \in (0, 1]$ durch und stelle diese Information in geeigneter Weise grafisch dar. Wo ist der 'problematische Bereich' von x-Werten?

Dabei wird exakte Rechnung vorausgesetzt: Die Rechenfehler, die bei dem Aufsummieren der Reihe entstehen, werden vernachlässigt. Der Reihenrest,

$$\ln(1+\delta) - \sum_{i=1}^{n} (-1)^{i-1} \frac{\delta^i}{i} \;=\; (-1)^n \frac{\delta^{n+1}}{n+1} \cdot \frac{1}{(1+\theta\delta)^{n+1}} \qquad \text{mit} \quad \theta \in [0,1],$$

entspricht dem absoluten Verfahrensfehler.

(2.3) Für einige konkrete Argumente $x \in \mathbb{F}$,

$$x \;=\; 2^{-k}, \quad k = 1, 2, \ldots, k_{max},$$

führe man die Summation dieser endlichen Reihen in natürlicher Reihenfolge durch und ermittle die relativen Fehler der so erhaltenen Resultate. (Als Bezugsgröße verwendet man den von der Standardprozedur gelieferten Wert.)

Wird eine relative Genauigkeit erreicht, die dem in (2.2) angegebenen Verfahrensfehler entspricht?

(2.4) Man gebe eine Begründung für die beobachteten Phänomene und vergleiche die Qualität der obigen Resultate mit jener, die man bei Summation in umgekehrter Reihenfolge erhält. Ist ein Unterschied erkennbar?

(2.5) Man entwickle folgende praktikable Variante, die auf dem Prinzip der *Argumentreduktion* beruht: Man approximiert die Funktion nur auf einem kleinen Standardintervall und verwendet spezielle Eigenschaften der Logarithmusfunktion um andere Argumente auf diesen Fall zurückzuführen.

Im folgenden bezeichne $\mathbf{e} \in \mathbb{F}$ die numerische Approximation von $e = \exp(1)$ in der gegebenen Arithmetik.

a) Ist $x \in [1/\mathbf{e}, 1]$, so wird $\ln x$ mit dem obigen Algorithmus ermittelt, wobei die Taylorreihe in der umgekehrten Reihenfolge summiert wird. (Es empfiehlt sich ein zusätzlicher Summand als 'Genauigkeitsreserve'.) Speziell ist $\ln 1 = 0$.

b) Ist $0 < x < 1/\mathbf{e}$, dann multipliziert man x so lange[10] mit dem Faktor $\mathbf{e}$, bis (nach k-facher Multiplikation mit $\mathbf{e}$) gilt

$$r \;:=\; \mathbf{e}^k \cdot x \in [1/\mathbf{e}, 1],$$

und verwendet die Identität

$$\ln x \;=\; \ln(r \cdot e^{-k}) \;=\; \ln r - k.$$

Man berechnet also $\ln r$ mit der Prozedur aus a) (mit $\delta = r - 1$) und subtrahiert anschließend k.

c) Ist $x > 1$, so berechnet man zuerst $\ln(1/x)$ nach a) oder b) und bildet anschließend $\ln x = -\ln(1/x)$.

Mit dieser Variante berechne man neuerlich die Werte der Logarithmusfunktion für Argumente $x = 2^{-k}$ (vgl. (2.3)) und bestimme den Fehler dieser Approximationen. Was ist zu beobachten? (... Genauigkeit, Rechenaufwand)

[10] Das geht auch etwas eleganter und genauer. Man überlege sich eine Variante – dabei darf aber natürlich keine Auswertung eines Logarithmus vorkommen!

(2.6) Für Argumente $x < 1/e$ nahe an $1/e$ tritt bei bei der Berechnung des zum reduzierten Argument $r = \mathbf{e}\,x$ gehörigen Wertes $\delta = r - 1$ *Auslöschung* auf, weil r mit einem Rundungsfehler behaftet ist und $r \approx 1$ ist.

Man diskutiere die Auswirkung dieses Auslöschungseffektes auf die Genauigkeit der resultierenden Approximation für $\ln x$, allenfalls unterstützt durch Tests. Besteht hier ein Handlungsbedarf?

(2.7) Man bestimme die relative Konditionszahl der Auswertung von $\ln x$ für Argumente $x < 1$ nahe an 1. Diese Konditionszahl ist groß (check!), aber für Maschinenzahlen $x \in \mathbb{F}$ (kein Datenfehler) erwartet man von einer guten Implementierung von ln, dass sie auch hier einen genauen Wert liefert.

Ist die Auswertung von $\ln x$ für $1 \approx x \in \mathbb{F}$ mittels der obigen Taylorreihe in diesem Sinne numerisch stabil? (Man betrachte insbesondere den Extremfall $x = 1 - O(eps)$, teste und analysiere.)

(2.8) Vergleichen Sie die Approximationsmethode, die unter (2.5) entwickelt wurde mit folgenden Alternativen:

(i) Lagrange-Interpolation zu äquidistanten Knoten,

(ii) Lagrange-Interpolation zu Chebyshev-Knoten.

Dabei wird wieder die relative Genauigkeit von 100 *eps* angestrebt und mit dem Standardintervall $x \in [1/\mathbf{e}, 1]$, vgl. (2.5), gearbeietet. Vergleichen Sie die Qualität der drei Approximationfuntionen, indem Sie, sowohl die Fehlerverläufe plotten, als auch die maximalen Fehler angeben. Vergleichen Sie auch den Aufwand, der zur Konstruktion der Erstzfunktionen notwenig ist (Anzahl der Funtionsauswertungen).

Beispiel 3: Sinusfunktion

Es soll eine Funktionsprozedur für die Sinusfunktion $\sin x$ entwickelt werden. Die Funktion wird dabei durch die ersten $n+1$ Summanden ihrer Taylorreihe bezüglich der Stelle $x_0 = 0$ ersetzt:

$$\sin x \;\approx\; \sum_{i=0}^{n} (-1)^i \frac{x^{2i+1}}{(2i+1)!} \;=\; x - \frac{x^3}{6} + \frac{x^5}{120} + \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$

(3.1) Man formuliere die Aufgabenstellung als numerisches Problem für einen konkreten Rechner:

- Genauigkeitsforderung: Es wird eine relative Genauigkeit von mindestens 100 *eps* angestrebt.
- Für welchen Bereich von x-Werten ist es (bei gegebener Arithmetik) überhaupt sinnvoll, eine Approximaton zu berechnen ('numerischer Definitionsbereich')? Welche Ausnahmesituation(en) liegt(liegen) außerhalb dieses Bereiches vor? Gibt es Symmetrien, so dass dieser Bereich weiter reduziert werden kann?
- Für welche speziellen Argumente $x \in \mathbb{F}$ ist der Funktionswert trivial?

(3.2) Wie viele Glieder der Taylorreihe muss man mindestens summieren, damit man die gewünschte Genauigkeit erreicht? Man führe eine Abschätzung für beliebige Argumente x aus dem numerischen Definitionsbereich durch und stelle diese Information in geeigneter Weise grafisch dar.

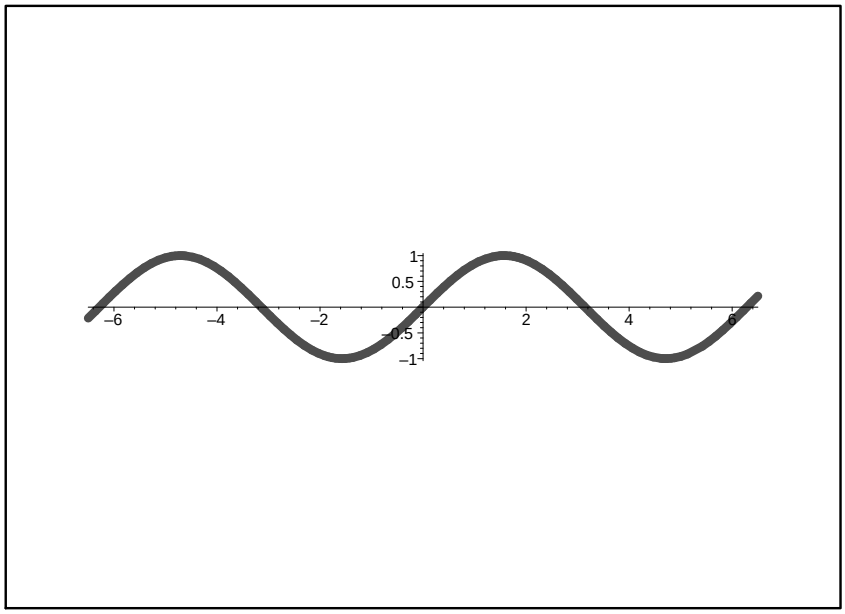


Abbildung 3: Sinusfunktion $\sin x$

Dabei wird exakte Rechnung vorausgesetzt: Die Rechenfehler, die bei dem Aufsummieren der Reihe entstehen, werden vernachlässigt. Der Reihenrest,

$$\sin x - \sum_{i=0}^{n} (-1)^i \frac{x^{2i+1}}{(2i+1)!} \;=\; (-1)^{n+1} \frac{x^{2n+3}}{(2n+3)!} \cos(\theta x), \qquad \text{mit} \quad \theta \in [0,1],$$

entspricht dem absoluten Verfahrensfehler.

(3.3) Für einige konkrete Argumente $x \in \mathbb{F}$,

$$x \;=\; 2^k, \quad k = 0, 1, 2, \ldots, k_{max},$$
$$x \;=\; 710,$$

führe man die Summation dieser endlichen Reihen in natürlicher Reihenfolge durch und ermittle die relativen Fehler der so erhaltenen Resultate. (Als Bezugsgröße verwendet man den von der Standardprozedur gelieferten Wert.) Dabei wähle man k_{max} 'vernünftig' (vgl. (3.2)!).

Wird eine relative Genauigkeit erreicht, die dem in (3.2) angegebenen Verfahrensfehler entspricht? Welche Argumentwerte sind besonders 'kritisch'?

(3.4) Man gebe eine Begründung für die beobachteten Phänomene und vergleiche die Qualität der obigen Resultate mit jener, die man bei Summation in umgekehrter Reihenfolge erhält. Hat man durch diese Maßnahme die Fehler bei den betragsgroßen Argumenten in den Griff bekommen?

(3.5) Man entwickle folgende praktikable Variante, die auf dem Prinzip der *Argumentreduktion* beruht: Man approximiert die Funktion nur auf dem Intervall $[0, 2\pi]$ und verwendet spezielle Eigenschaften der Sinusfunktion, insbesondere deren Periodizität, um andere Argumente auf diesen Fall zurückzuführen.

Im folgenden bezeichne **pi** $\in \mathbb{F}$ die numerische Approximation von π in der gegebenen Arithmetik.

a) Ist $x \in [0, \mathbf{pi}]$, so wird $\sin x$ mit dem obigen Algorithmus ermittelt, wobei die Taylorreihe in der umgekehrten Reihenfolge summiert wird. (Es empfiehlt sich ein zusätzlicher Summand als 'Genauigkeitsreserve'.)

b) Der Fall $x \in [\mathbf{pi}, 2\mathbf{pi}]$, wird unmittelbar auf Fall a) zurückgeführt (wie?).

c) Ist $x > 2\,\mathbf{pi}$, dann stellt man x dar als

$$x \;=\; 2\,k\,\mathbf{pi} + r, \qquad \text{mit} \quad k \in \mathbb{N}, \quad 0 \le r < \mathbf{pi},$$

und verwendet die Periodizität von sin:

$$\sin(2k\pi + r) \;=\; \sin r.$$

Man bestimmt also das betreffende k, berechnet $r \;:=\; x - 2\,k\,\mathbf{pi}$ und anschließend $\sin r$ gemäß $a), b)$.

d) Für $x < 0$ verwendet man die Schiefsymmetrie der Sinusfunktion: $\sin x = -\sin(-x)$.

(3.6) Mit der Variante (3.5) berechne man neuerlich die Werte der Sinusfunktion für die gleichen Argumente wie unter (3.3). Was hat sich verbessert, bzw. wo treten noch Ungenauigkeiten auf? Man achte auf die erreichte relative Genauigkeit!

Das spezielle Argument $x = 710$ steht stellvertretend für betragsgroße Argumente, bei denen $\sin x$ sehr betragsklein ist. (Beachte: $710 \approx 2\pi \cdot 113.000009\ldots$!) Diese Konstellation soll jetzt noch genauer untersucht werden.

- Wodurch ist bei solchen Argumenten die numerische Ungenauigkeit des unter (3.5) konstruierten Algorithmus verursacht? (Je kleiner $|\sin x|$ und je größer k, desto ausgeprägter ist diese Ungenauigkeit.)
- Kann man diese Ungenauigkeit dadurch beseitigen, dass man $\sin r$ 'exakt' auswertet (also mit der Standardprozedur für sin anstatt mit der selbstgebastelten Operation)?
- Welche Operation müsste man mit *höherer Genauigkeit* ausführen, um diese numerische Ungenauigkeit zu mildern bzw. zu beseitigen? (Begründung!)
- Für $\sin 710$ soll nun ein genauerer Wert in folgender Weise ermittelt werden: Man berechnet das reduzierte Argument r in erhöhter Genauigkeit (etwa mit 25 Dezimalstellen, mit Hilfe von MAPLE) übernimmt den sich daraus durch Rundung ergebenden Wert in das eigene Programm, und wertet dann sin aus.

  Man berechne die beiden Approximationen[11] für $\sin 710$, die sich mit diesen beiden Versionen von r ergeben ('normal' bzw. mit erhöhter Genauigkeit bestimmt), vergleiche diese mit dem 'exakten' Wert für $\sin 710$ und interpretiere die Resultate.
- Man bestimme auch die relative Konditionszahl der Auswertung von $\sin x$ für das 'kritische' Argument $x = 710$. Diese Konditionszahl ist groß (check!), aber da ja x exakt als Maschinenzahl gegeben ist (kein Datenfehler), erwartet man von einer guten Implementierung von sin, dass sie auch hier einen genauen Wert liefert.

(3.7) Vergleichen Sie die Approximationsmethode, die unter (3.5) entwickelt wurde mit folgenden Alternativen:

[11] Um den Einfluss der verschiedenen Fehlerquellen sauber zu trennen, soll bei diesem Vergleich der Wert von $\sin r$ der Einfachheit halber mit der 'exakten' Standardfunktion sin ausgewertet werden, und nicht mit der selbstgebastelten Approximation.

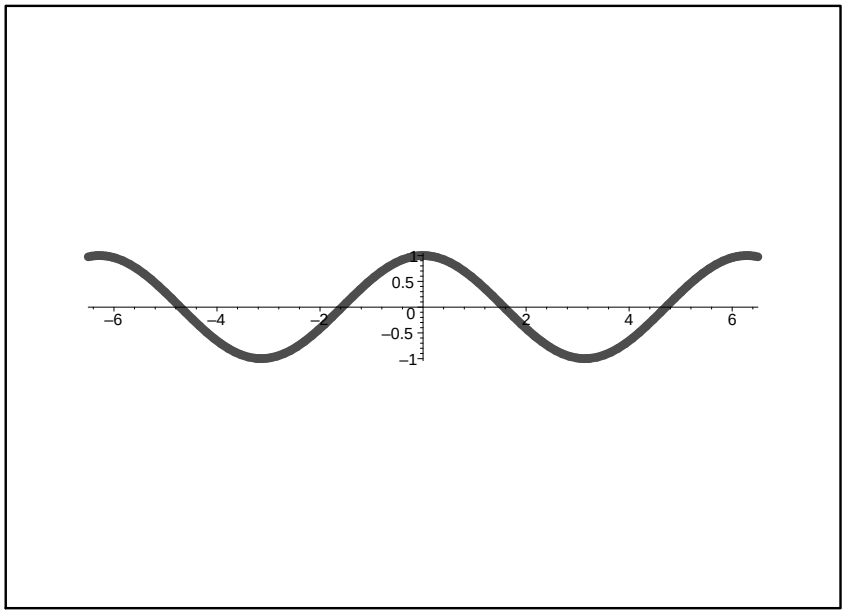


Abbildung 4: Kosinusfunktion $\cos x$

(i) Lagrange-Interpolation zu äquidistanten Knoten,

(ii) Lagrange-Interpolation zu Chebyshev-Knoten.

Dabei wird wieder die relative Genauigkeit von 100 *eps* angestrebt und mit dem Standardintervall $x \in [0, \mathbf{pi}]$, vgl. (3.5), gearbeietet. Vergleichen Sie die Qualität der drei Approximationfuntionen, indem Sie, sowohl die Fehlerverläufe plotten, als auch die maximalen Fehler angeben. Vergleichen Sie auch den Aufwand, der zur Konstruktion der Erstzfunktionen notwenig ist (Anzahl der Funtionsauswertungen).

Beispiel 4: Kosinusfunktion

Es soll eine Funktionsprozedur für die Kosinusfunktion $\cos x$ entwickelt werden. Die Funktion wird dabei durch die ersten $n+1$ Summanden ihrer Taylorreihe bezüglich der Stelle $x_0 = 0$ ersetzt:

$$\cos x \approx \sum_{i=0}^{n} (-1)^i \frac{x^{2i}}{(2i)!} = 1 - \frac{x^2}{2} + \frac{x^4}{24} + \ldots + (-1)^n \frac{x^{2n}}{(2n)!}$$

(4.1) Man formuliere die Aufgabenstellung als numerisches Problem für einen konkreten Rechner:

- Genauigkeitsforderung: Es wird eine relative Genauigkeit von mindestens 100 *eps* angestrebt.
- Für welchen Bereich von x-Werten ist es (bei gegebener Arithmetik) überhaupt sinnvoll, eine Approximaton zu berechnen ('numerischer Definitionsbereich')? Welche Ausnahmesituation(en) liegt(liegen) außerhalb dieses Bereiches vor? Gibt es Symmetrien, so dass dieser Bereich weiter reduziert werden kann?
- Für welche speziellen Argumente $x \in \mathbb{F}$ ist der Funktionswert trivial?

(4.2) Wie viele Glieder der Taylorreihe muss man mindestens summieren, damit man die gewünschte Genauigkeit erreicht? Man führe eine Abschätzung für beliebige Argumente x aus dem numerischen Definitionsbereich durch und stelle diese Information in geeigneter Weise grafisch dar.

Dabei wird exakte Rechnung vorausgesetzt: Die Rechenfehler, die bei dem Aufsummieren der Reihe entstehen, werden vernachlässigt. Der Reihenrest,

$$\cos x - \sum_{i=0}^{n} (-1)^i \frac{x^{2i}}{(2i)!} \;=\; (-1)^{n+1} \frac{x^{2n+2}}{(2n+2)!} \cos(\theta x), \qquad \text{mit} \quad \theta \in [0,1],$$

entspricht dem absoluten Verfahrensfehler.

(4.3) Für einige konkrete Argumente $x \in \mathbb{F}$,

$$\begin{aligned} &a) \quad x \;=\; 2^k, \quad k = 0, 1, 2, \ldots, k_{max}, \\ &b) \quad x \;=\; 887.5, \end{aligned}$$

führe man die Summation dieser endlichen Reihen in natürlicher Reihenfolge durch und ermittle die relativen Fehler der so erhaltenen Resultate. (Als Bezugsgröße verwendet man den von der Standardprozedur gelieferten Wert.) Dabei wähle man k_{max} 'vernünftig' (vgl. (3.2)!).

Wird eine relative Genauigkeit erreicht, die dem in (3.2) angegebenen Verfahrensfehler entspricht? Welche Argumentwerte sind besonders 'kritisch'? Was passiert bei $x = 887.5$?

(4.4) Man gebe eine Begründung für die beobachteten Phänomene und vergleiche die Qualität der obigen Resultate mit jener, die man bei Summation in umgekehrter Reihenfolge erhält. Hat man durch diese Maßnahme die Fehler bei den betragsgroßen Argumenten in den Griff bekommen?

(4.5) Man entwickle folgende praktikable Variante, die auf dem Prinzip der *Argumentreduktion* beruht: Man approximiert die Funktion nur auf dem Intervall $[0, 2\pi]$ und verwendet spezielle Eigenschaften der Kosinusfunktion, insbesondere deren Periodizität, um andere Argumente auf diesen Fall zurückzuführen.

Im folgenden bezeichne $\mathbf{pi} \in \mathbb{F}$ die numerische Approximation von π in der gegebenen Arithmetik.

a) Ist $x \in [0, \mathbf{pi}]$, so wird $\cos x$ mit dem obigen Algorithmus ermittelt, wobei die Taylorreihe in der umgekehrten Reihenfolge summiert wird. (Es empfiehlt sich ein zusätzlicher Summand als 'Genauigkeitsreserve'.)

b) Der Fall $x \in [\mathbf{pi}, 2\mathbf{pi}]$, wird unmittelbar auf Fall a) zurückgeführt (wie?).

c) Ist $x > 2\,\mathbf{pi}$, dann stellt man x dar als

$$x \;=\; 2\,k\,\mathbf{pi} + r, \qquad \text{mit} \quad k \in \mathbb{N}, \quad 0 \le r < \mathbf{pi},$$

und verwendet die Periodizität von cos:

$$\cos(2k\pi + r) \;=\; \cos r.$$

Man bestimmt also das betreffende k, berechnet $r := x - 2\,k\,\mathbf{pi}$ und anschließend $\sin r$ gemäß $a), b)$.

d) Für $x < 0$ verwendet man die Symmetrie der Kosinusfunktion: $\cos x = \cos(-x)$.

(4.6) Mit der Variante (4.5) berechne man neuerlich die Werte der Kosinusfunktion für die gleichen Argumente wie unter (4.3). Was hat sich verbessert, bzw. wo treten noch Ungenauigkeiten auf? Man achte auf die erreichte relative Genauigkeit!

Das spezielle Argument $x = 887.5$ steht stellvertretend für betragsgroße Argumente, bei denen $\cos x$ sehr betragsklein ist. (Beachte: $887.5 - \pi/2 \approx 2\pi \cdot 141.00001\ldots$!) Diese Konstellation soll jetzt noch genauer untersucht werden.

- Wodurch ist bei solchen Argumenten die numerische Ungenauigkeit des unter (4.5) konstruierten Algorithmus verursacht? (Je kleiner $|\cos x|$ und je größer k, desto ausgeprägter ist diese Ungenauigkeit.)
- Kann man diese Ungenauigkeit dadurch beseitigen, dass man $\cos r$ 'exakt' auswertet (also mit der Standardprozedur für cos anstatt mit der selbstgebastelten Operation)?
- Welche Operation müsste man mit *höherer Genauigkeit* ausführen, um diese numerische Ungenauigkeit zu mildern bzw. zu beseitigen? (Begründung!)
- Für $\cos 887.5$ soll nun ein genauerer Wert in folgender Weise ermittelt werden: Man berechnet das reduzierte Argument r in erhöhter Genauigkeit (etwa mit 25 Dezimalstellen, mit Hilfe von MAPLE), übernimmt den sich daraus durch Rundung ergebenden Wert in das eigene Programm, und wertet dann cos aus.

  Man berechne die beiden Approximationen[12] für $\cos 887.5$, die sich mit diesen beiden Versionen von r ergeben ('normal' bzw. mit erhöhter Genauigkeit bestimmt), vergleiche diese mit dem 'exakten' Wert für $\cos 887.5$ und interpretiere die Resultate.
- Man bestimme auch die relative Konditionszahl der Auswertung von $\cos x$ für das 'kritische' Argument $x = 887.5$. Diese Konditionszahl ist groß (check!), aber da ja x exakt als Maschinenzahl gegeben ist (kein Datenfehler), erwartet man von einer guten Implementierung von cos, dass sie auch hier einen genauen Wert liefert.

(4.7) Vergleichen Sie die Approximationsmethode, die unter (4.5) entwickelt wurde mit folgenden Alternativen:

(i) Lagrange-Interpolation zu äquidistanten Knoten,

(ii) Lagrange-Interpolation zu Chebyshev-Knoten.

Dabei wird wieder die relative Genauigkeit von 100 *eps* angestrebt und mit dem Standardintervall $x \in [0, \mathbf{pi}]$, vgl. (4.5), gearbeietet. Vergleichen Sie die Qualität der drei Approximationfuntionen, indem Sie, sowohl die Fehlerverläufe plotten, als auch die maximalen Fehler angeben. Vergleichen Sie auch den Aufwand, der zur Konstruktion der Erstzfunktionen notwenig ist (Anzahl der Funtionsauswertungen).

Beispiel 5: Arcustangens

Es soll eine Funktionsprozedur für die Arcustangensfunktion $\arctan(x)$ (Hauptzweig, mit Funktionswerten aus $(-\pi/2, \pi/2)$) entwickelt werden. Die Funktion wird dabei durch die ersten $n + 1$

[12] Um den Einfluss der verschiedenen Fehlerquellen sauber zu trennen, soll bei diesem Vergleich der Wert von $\cos r$ der Einfachheit halber mit der 'exakten' Standardfunktion cos ausgewertet werden, und nicht mit der selbstgebastelten Approximation.

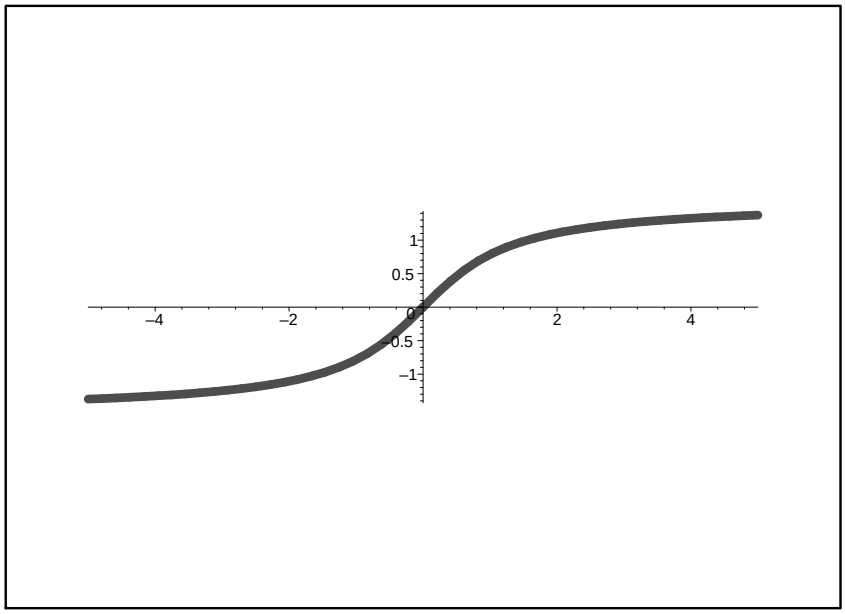


Abbildung 5: Arcustangens $\arctan x$

Summanden ihrer Taylorreihe bezüglich der Stelle $x_0 = 0$ ersetzt:

$$\arctan x \approx \sum_{i=0}^{n} (-1)^i \frac{x^{2i+1}}{2i+1} = x - \frac{x^3}{3} + \frac{x^5}{5} + \ldots + (-1)^n \frac{x^{2n+1}}{2n+1}$$

Diese Reihe konvergiert jedoch nur für $|x| \leq 1$. (Für $x = 1$ erhält man eine bekannte Reihenapproximation von $\pi/4$.)

(5.1) Man formuliere die Aufgabenstellung als numerisches Problem für einen konkreten Rechner:

- Genauigkeitsforderung: Es wird eine relative Genauigkeit von mindestens 100 *eps* angestrebt.
- Für welchen Bereich von x-Werten ist es (bei gegebener Arithmetik) überhaupt sinnvoll, eine Approximaton zu berechnen ('numerischer Definitionsbereich')? Welche Ausnahmesituation(en) liegt(liegen) außerhalb dieses Bereiches vor? Gibt es Symmetrien, so dass dieser Bereich weiter reduziert werden kann?
- Für welche speziellen Argumente $x \in \mathbb{F}$ ist der Funktionswert trivial?

(5.2) Wie viele Glieder der Taylorreihe muss man mindestens summieren, damit man die gewünschte Genauigkeit erreicht? Man führe eine Abschätzung für beliebige Argumente $x \in (0, 1]$ durch und stelle diese Information in geeigneter Weise grafisch dar.

Dabei wird exakte Rechnung vorausgesetzt: Die Rechenfehler, die beim Aufsummieren der Reihe entstehen, werden vernachlässigt. In diesem Fall gilt folgende Abschätzung für den Verfahrensfehler:[13]

$$\left| \arctan x - \sum_{i=0}^{n} \frac{x^{2i+1}}{2i+1} \right| \leq \frac{x^{2n+3}}{2n+3}$$

[13]Bei einer konvergenten alternierenden Reihe kann man den Abbruchfehler durch den Betrag des ersten nicht mehr berücksichtigten Reihengliedes abschätzen.

(5.3) Für eine Folge von konkreten Argumente $x \in \mathbb{F}$,

$$x = 1 - 2^{-k}, \qquad k = 1, 2, \ldots, k_{max},$$

führe man die Summation dieser endlichen Reihen in natürlicher Reihenfolge durch und ermittle die relativen Fehler der so erhaltenen Resultate. (Als Bezugsgröße verwendet man den von der Standardprozedur gelieferten Wert.)

Wird eine relative Genauigkeit erreicht, die dem unter (5.2) angegebenen Verfahrensfehler entspricht? Wie steht es mit dem Rechenaufwand (d.h. wieviele Reihenglieder mussten summiert werden?)

(5.4) Man gebe eine Begründung für die beobachteten Phänomene und vergleiche die Qualität der obigen Resultate mit jener, die man bei Summation in umgekehrter Reihenfolge erhält. Ist ein Unterschied erkennbar?

(5.5) Man entwickle folgende praktikable Variante, die für Argumente x nahe an 1 eine schneller konvergente Reihe verwendet. Mit Hilfe von MAPLE erhält man (Taylorreihe an $x_0 = 1$):

$$\arctan(1+\delta) \approx \frac{\pi}{4} + \frac{\delta}{2} - \frac{\delta^2}{4} + \frac{\delta^3}{12} - \frac{\delta^5}{40} + \frac{\delta^6}{48} - \frac{\delta^7}{112} + \frac{\delta^9}{288} - \frac{\delta^{10}}{320} + \frac{\delta^{11}}{704} + \ldots$$

(weitere Terme bzw. eine Restgliedabschätzung ermittle man bei Bedarf mittels MAPLE).

Man teste die numerische Konvergenz dieser Reihe für die gleiche Folge von Argumenten wie unter (5.2), und vergleiche insbesondere den Unterschied in der Genauigkeit bei umgekehrter Summationsreihenfolge, abhängig von der Anzahl der verwendeten Summanden.

(5.6) Schließlich implementiere und teste man folgende Approximation von $\arctan x$ für belienige x:

a) Ist $x \in [0, 1]$, so wird $\arctan x$ mit dem obigen Algorithmus ermittelt, wobei die jeweilige Taylorreihe in der umgekehrten Reihenfolge summiert wird. (Es empfiehlt sich ein zusätzlicger Summand als 'Genauigkeitsreserve'.)

b) Für $x > 1$ verwendet man die Identität

$$\arctan x = \frac{\pi}{2} - \arctan \frac{1}{x}.$$

c) Für $x < 0$ ist $\arctan x = -\arctan(-x)$.

(5.7) Vergleichen Sie die Approximationsmethode, die unter (5.6) entwickelt wurde mit folgenden Alternativen:

(i) Lagrange-Interpolation zu äquidistanten Knoten,

(ii) Lagrange-Interpolation zu Chebyshev-Knoten.

Dabei wird wieder die relative Genauigkeit von 100 *eps* angestrebt und mit dem Standardintervall $x \in [0, 1]$, vgl. (5.6), gearbeietet. Vergleichen Sie die Qualität der drei Approximationfuntionen, indem Sie, sowohl die Fehlerverläufe plotten, als auch die maximalen Fehler angeben. Vergleichen Sie auch den Aufwand, der zur Konstruktion der Erstzfunktionen notwenig ist (Anzahl der Funtionsauswertungen).

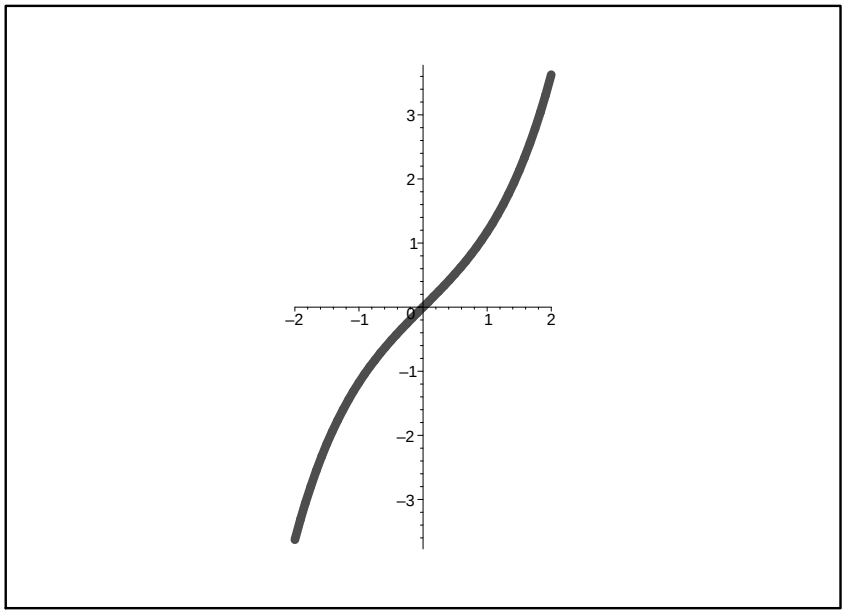


Abbildung 6: Sinus hyperbolicus $\sinh x$

Beispiel 6: Sinus hyperbolicus

Es soll eine Funktionsprozedur für den hyperbolischen Sinus $\sinh x$ entwickelt werden. Dazu werden zwei verschiedene Varianten in Betracht gezogen:

(i) Die Funktion wird durch die ersten $n+1$ Summanden ihrer Taylorreihe bezüglich der Stelle $x_0 = 0$ approximiert:

$$\sinh x \approx \sum_{i=0}^{n} \frac{x^{2i+1}}{(2i+1)!} = x + \frac{x^3}{6} + \frac{x^5}{120} + \ldots + \frac{x^{2n+1}}{(2n+1)!}$$

(ii) Unter der Annahme, dass eine Funktionsprozedur für die Exponentialfunktion $\exp(x) = e^x$ verfügbar ist, soll $\sinh x$ direkt auf Grund der Definition

$$\sinh x := \frac{e^x - e^{-x}}{2}$$

implementiert werden (kein Verfahrensfehler).

(6.1) Man formuliere die Aufgabenstellung als numerisches Problem für einen konkreten Rechner:

- Genauigkeitsforderung: Es wird eine relative Genauigkeit von mindestens 100 *eps* angestrebt.
- Für welchen Bereich von x-Werten ist es (bei gegebener Arithmetik) überhaupt sinnvoll, eine Approximaton zu berechnen ('numerischer Definitionsbereich')? Welche Ausnahmesituation(en) liegt(liegen) außerhalb dieses Bereiches vor? Gibt es Symmetrien, so dass dieser Bereich weiter reduziert werden kann?
- Für welche speziellen Argumente $x \in \mathbb{F}$ ist der Funktionswert trivial?

(6.2) Wie viele Glieder der Taylorreihe muss man mindestens summieren, damit man die gewünschte Genauigkeit erreicht? Man führe eine Abschätzung für beliebige Argumente x aus dem numerischen Definitionsbereich durch und stelle diese Information in geeigneter Weise grafisch dar.

Dabei wird exakte Rechnung vorausgesetzt: Die Rechenfehler, die bei dem Aufsummieren der Reihe entstehen, werden vernachlässigt. Der Reihenrest,

$$\sinh x - \sum_{i=0}^{n} \frac{x^{2i+1}}{(2i+1)!} = \frac{x^{2n+3}}{(2n+3)!} \cosh(\theta x), \qquad \text{mit} \quad \theta \in [0,1],$$

entspricht dem absoluten Verfahrensfehler.

(6.3) Für eine Folge von konkreten Argumenten $x \in \mathbb{F}$,

$$x = 2^{\pm k}, \quad k = 0, 1, 2, 3, \ldots \quad \text{(so weit wie möglich)}$$

führe man die Summation dieser endlichen Reihen in natürlicher Reihenfolge durch und ermittle den Fehler der so erhaltenen Resultate. (Als Bezugsgröße verwendet man den von der Standardprozedur gelieferten Wert.)

Wird eine relative Genauigkeit erreicht, die dem unter (6.2) angegebenen Verfahrensfehler entspricht? Man gebe eine Begründung für die beobachteten Phänomene und vergleiche die Qualität der obigen Resultate mit jener, die man bei Summation in umgekehrter Reihenfolge erhält.

(6.4) Man versuche zu beurteilen, für welchen Bereich von x-Werten Variante (i) bzw. (ii) aus (6.2) der anderen vorzuziehen ist, und führe entsprechende numerische Tests durch. (Rechenaufwand und numerisch erzielte Genauigkeit sind die entscheidenden Kenngrößen.)

Was passiert für sehr betragskleine Argumente x? Welche Konsequenz ist also hinsichtlich eine konkreten Implementierung zu ziehen?

(6.5) Für die auf der Exponentialfunktion basierende Variante führe man eine formale Rundungsfehleranalyse durch. Dabei soll angenommen werden, dass $\exp(x)$ stets mit einem relativen Rundungsfehler der Größenordnung $10\,eps$ ausgewertet wird.

(6.6) Auf Grund der unter (6.4) und (6.5) gewonnenen Erkenntnisse implementiere und teste man schließlich folgende Approximation von $\sinh x$:

a) Je nach Größenordnung des Argumentes wird mit der entsprechend langen Taylorreihe oder mit der Exponentialfunktion gearbeitet (ein zusätzlicher Term als Genauigkeitsreserve). Der 'Umschaltpunkt' zwischen den beiden Varianten ist geeignet zu wählen (vgl. (6.5)).

b) Für $x < 0$ ist $\sinh x = -\sinh(-x)$.

(6.7) Vergleichen Sie die Approximationsmethode, die unter (6.6) entwickelt wurde mit folgenden Alternativen:

(i) Lagrange-Interpolation zu äquidistanten Knoten,

(ii) Lagrange-Interpolation zu Chebyshev-Knoten.

Dabei wird wieder die relative Genauigkeit von 100 *eps* angestrebt und auf dem Intervall $x \in [0, \infty)$, vgl. (6.6), gearbeietet. Vergleichen Sie die Qualität der drei Approximationfuntionen, indem Sie, sowohl die Fehlerverläufe plotten, als auch die maximalen Fehler angeben. Vergleichen Sie auch den Aufwand, der zur Konstruktion der Erstzfunktionen notwenig ist (Anzahl der Funtionsauswertungen).

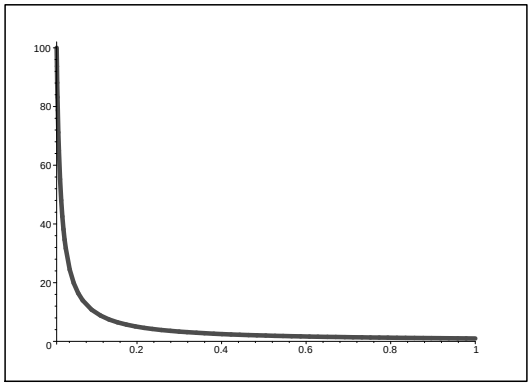
Projekt III/4: Interpolation einer Wertetabelle

Aus einer gegebenen Tabelle von Funktionswerten $(x_i, f(x_i))$ sollen (Näherungs-) Werte der betreffenden Funktion $f(x)$ an Zwischenstellen x durch lokale, stückweise polynomiale Interpolation (unter Verwendung zu x benachbarter Stellen x_i) gewonnen werden.

Dabei ergeben sich eine Reihe von Fragestellungen (siehe weiter unten).

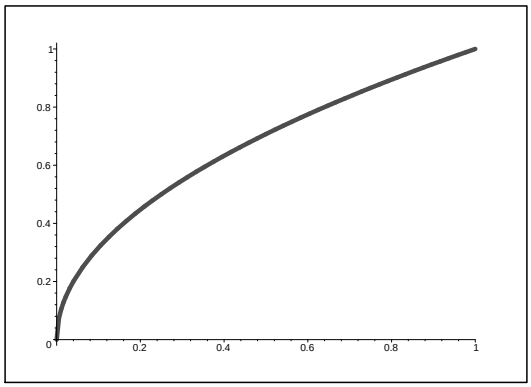
Die konkreten Beispiele:

Beispiel 1: $f(x) = 1/x,\ I = (0, 1]$

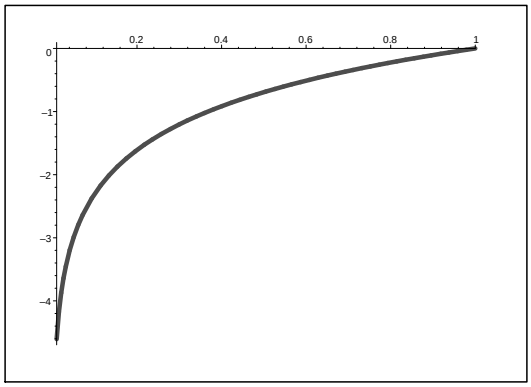


Das Beispiel ist durchaus nicht realitätsfremd. Division ist die 'schwierigste' arithmetische Operation, und unter Verwendung polynomialer Interpolation kann man sie (näherungsweise) auf die elementareren Rechenoperationen $+, -, *$ zurückführen. In Mikroprozessoren passiert (ansatzweise) ähnliches.
Achtung: $x = 0$?!.

Beispiel 2: $f(x) = \sqrt{x},\ I = [0, 1]$



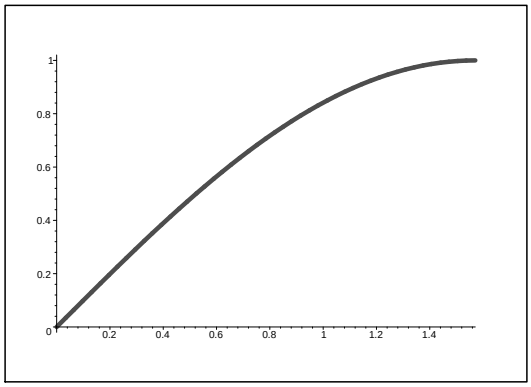
Achtung: $x = 0$?!.



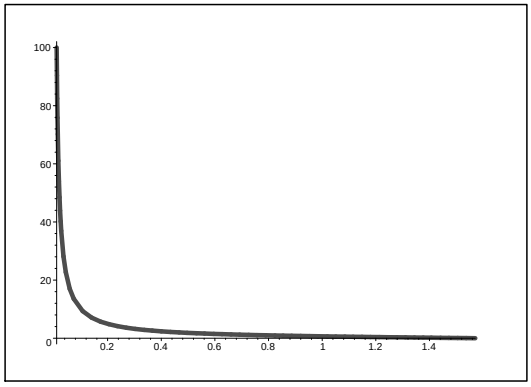
Beispiel 3: $f(x) = \ln x,\ I = (0, 1]$

Achtung: $x = 0$?!.

Beispiel 4: $f(x) = \sin x,\ I = [0, \frac{\pi}{2}]$

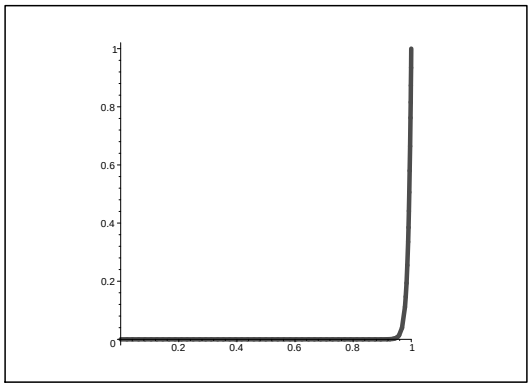


Beispiel 5: $f(x) = \cot x,\ I = [0, \frac{\pi}{2}]$



Achtung: $x = 0$?!.

Beispiel 6: $f(x) = x^{100},\ I = [0, 1]$



Zu beachten:

- Wenn im Folgenden von linearer ($m = 1$), quadratischer ($m = 2$) bzw. allgemein von polynomialer Interpolation mit Grad m die Rede ist, so bedeutet dies, dass für die Interpolation an einer Stelle x jeweils $m + 1$ benachbarte x_i herangezogen werden, so dass x im Inneren des von den x_i 'aufgespannten' Intervalls liegt (am besten 'möglichst in der Mitte').
- Grafische Visualisierung der Ergebnisse ist sinnvoll und erwünscht.

Hier die konkreten zu bearbeitenden Fragestellungen (für eines der konkreten $f(x),\ x \in I = [a, b]$ bzw. $I = (a, b]$):

- **Teilprojekt 1:**
  Es sei angenommen, dass die Tabelle exakte Funktionswerte enthält. Für ein beliebig vorgegebenes $h > 0$ gebe man, an welche absolute Genauigkeit mit linearer bzw. quadratischer Extrapolation mit Sicherheit erzielbar ist (Verfahrensfehler).

  Falls so eine Abschätzung nicht möglich ist, begründe man diese 'Unmöglichkeit' und betrachte dann ein vereinfachtes Problem in einem reduzierten Intervall $I = [a+\delta, b]$, mit (z.B.) $\delta = 0.1$.

  Könnte man durch lineare Interpolation basierend auf einer feineren Auflösung, d.h. mit $\tilde{h} <$ h einen kleineren Verfahrensfehler erzielen als mittels quadratischer Interpolation? Wie wäre dabei $\tilde{h}$ im Vergleich zu h zu wählen?

  Man implementiere die polynomiale Interpolation in obigem Sinne und teste für ein vorgegebenes h und verschiedene Stellen x die erreichte absolute und relative Genauigkeit.

- **Teilprojekt 2:**
  Man untersuche die Frage, wie h gewählt werden muss, damit bei linearer bzw. quadratischer Interpolation garantiert werden kann, dass der *relative* Verfahrensfehler für alle $x \in I$ unter einer vorgegebenen Toleranz $rtol = 10^{-k}$ bleibt.

  Man teste am Computer für $k = 3, 6, 9, 12, 15$, ob die geforderte Genauigkeit jeweils tatsächlich erreicht wird, und bewerte und interpretiere die Ergebnisse.

  Was würde für die Verwendung höherer Polynomgrade m sprechen?

Unter Zuhilfenahme der in Teilprojekt 1 erhaltenen theoretischen Fehlerschätzung überlege man, wie für lineare bzw. quadratische Interpolation (Interpolationspolynom $q(x)$) ein *nichtäquidistantes* Gitter $\{x_i\}$ gewählt werden sollte, so dass der *relative* Verfahrensfehler eine vorgegebene Toleranzforderung erfüllt, d.h.

$$\left| \frac{q(x) - f(x)}{f(x)} \right| \leq 10^{-k}$$

Man implementiere und teste diese Strategie. An welchen Stellen x treten hier Schwierigkeiten auf?

- **Teilprojekt 3:**
  Polynominterpolation kann auch für die Approximation von Ableitungen einer Funktion dienen (die Ableitung des Interpolationspolynoms wird als Approximation verwendet).

  Man untersuche experimentell, für verschiedene Stellen $x \in I$, die Approximationsqualität (absoluter Fehler), die sich für die Funktionswerte selbst (0. Ableitung) und die 1. und 2. Ableitung ergibt, unter Verwendung der Polynomgrade $1, 2$ und 3. Dabei bestimme man auch die 'Konvergenzgeschwindigkeit' für abnehmendes h, indem man die Berechnungen für verschiedene Werte von h ausführt (Wertetabelle mit immer feinerer Abtastung, z.B. $h = 1/2,\ 1/4,\ 1/8,\ \ldots$. Aus dem Ansatz

$$|q_h^{(k)}(x) - f^{(k)}(x)| \ = \ C \cdot h^p, \quad |q_{h/2}^{(k)}(x) - f^{(k)}(x)| \ = \ C \cdot (h/2)^p,$$

  bestimmt man die Fehlerkonstante C und die 'beobachtete' Ordnung p.

  Welche Systematik ist in den beobachteten Ordnungen p zu erkennen?

  Man versuche, zumindest für den Fall der linearen Interpolation, die für die 1. Ableitung beobachtete Approximationsordnung p theoretisch zu beweisen.

  Für die Approximation der 1. und 2. Ableitung mit Hilfe linearer und quadratischer Interpolation leite man eine Abschätzung für die Auswirkung des Datenfehlers her: Angenommen, q bzw. $\tilde{q}$ interpolieren die exakten Daten $(x_i, f(x_i))$ bzw. gestörte Daten $(x_i, f(x_i) + \delta_i)$, mit $|\delta_i| \leq \delta$, dann ist folgende Abschätzung gesucht:

$$| \, q(x) - \tilde{q}(x) \, | \ \leq \ ?? \qquad \text{(abhängig, von } h, \, \delta).$$

  Man kommentiere das Ergebnis.

  Welche verbesserte Abschätzung lässt sich angeben, wenn die Störungen δ_i nicht 'zufällig' sind, sondern als Werte einer hinreichend glatten (d.h. geeignet oft differenzierbaren) Funktion g interpretiert werden können, d.h. $\delta_i = g(x_i)$?

Projekt IV/1: Numerische Quadratur - Version 1

Entwickelt man Software zur numerischen Integration, so muß man sich nicht nur für ein numerisches Integrationsverfahren (bzw. für eine Klasse von Integrationsverfahren wie z.B. die Newton-Cotes Formeln) entscheiden, sondern man muß auch Steuerungsmaßnahmen (Schrittweitensteuerung, Abbruchkriterien, ...) vorsehen. Die meisten der folgenden Teilprojekte beziehen sich auf die Planung und Entwicklung solcher Steuerungsmaßnahmen. Greifen Sie bei Ihren Überlegungen auch auf die numerische Literatur zu dieser Thematik zurück!

- Teilprojekt 1:
  Zur Illustration der numerischen Faustformel
  "Geringe Genauigkeitsanforderungen $\Leftrightarrow$ Verfahren geringer Ordnung sind effizient; Strenge Genauigkeitsanforderungen $\Leftrightarrow$ Verfahren hoher Ordnung sind effizient" realisieren Sie folgende numerischen Experimente:

  Wählen Sie Testbeispiele, bei denen die Glattheit des Integranden (d. h. die Größenordnungen der Ableitungen) am ganzen Integrationsintervall nicht schwankt, sodaß äquidistante Gitter als sinnvoll erscheinen. Wählen Sie verschiedene numerische Quadraturverfahren (Newton-Cotes Formeln, Gauß Formeln, Kronrod Formeln) und integrieren Sie auf Folgen äquidistanter Gitter[14], d. h. zunächst auf dem gröbsten Gitter das für das betrachtete Verfahren möglich ist (z. B. Simpsonregel: $x_0 = a, x_1 = \frac{a+b}{2}, x_2 = b$; $a, b \ldots$ Integrationsgrenzen), dann auf doppelt so feinem Gitter, dann auf 4 mal so feinem Gitter, u.s.w. Beobachten Sie in all diesen Fällen den Verfahrensfehler. Stellen Sie fest, ab welcher Gitterfeinheit sich die Ordnung des Verfahrens beobachten läßt. Geben Sie eine Genauigkeitsforderung vor und prüfen sie, wieviele Funktionsauswertungen bei jedem einzelnen Verfahren nötig sind um das vorgegebene Genauigkeitsniveau zu erreichen. Experimentieren Sie auch mit weniger glatten Testbeispielen, d. h. mit Beispielen, bei denen an einzelnen Punkten Unstetigkeiten höherer Ableitungen auftreten. Konstruieren Sie Beispiele, bei denen nur wenige Unstetigkeitspunkte vorliegen (ev. nur einer!) und Beispiele, wo die entsprechende Ableitung relativ oft springt. Was beobachten Sie insbesondere in jenen Fällen, wo die Ableitungen, die in der Konvergenztheorie (d. h. beim Beweis der Ordnungsaussagen des Verfahrens) auftreten, in ihrer Ordnung kritisch nahe zu der Ordnung der unstetigen Ableitungen liegen?

- Teilprojekt 2:
  Überlegen Sie Steuerungsmaßnahmen für ein Quadraturpaket basierend auf den Newton-Cotes Formeln (eigene Überlegungen + Studium der einschlägigen numerischen Literatur). Falls Sie auch an automatische Ordnungssteuerung denken, sollten Sie die experimentellen Erfahrungen von Teilprojekt 1 in Ihre Überlegungen einbeziehen. Formulieren Sie – basierend auf diesen Überlegungen – einen numerischen Integrationsalgorithmus und programmieren Sie diesen. Testen Sie Ihr Programm anhand von Testbeispielen unterschiedlicher Schwierigkeit.

- Teilprojekt 3:
  Wie Teilprojekt 2, jedoch soll das zu entwickelnde Programm auf das Gauß-Kronrod Formelpaaren (statt auf den Newton-Cotes Formeln) beruhen.

[14] Bei den auf irrationalen Knoten beruhenden Gauß Formeln bezieht sich der Begriff "äquidistant" auf das "äußere" Gitter, bei dem in den einzelnen Gitterintervallen der Integrand jeweils durch *ein* Interpolationsproblem ersetzt wird.

- Teilprojekt 4:
  Wie Teilprojekt 2, jedoch soll das zu entwickelnde Programm auf der Trapezsummenextrapolation (statt auf den Newton-Cotes Formeln) beruhen.

- Teilprojekt 5:
  Testen Sie die in den Programmbibliotheken vorhandene Software zur numerischen Integration. Eine Auswahl von Testbeispielen wird in der Sprechstunde zur Verfügung gestellt. Vergleichen Sie die Qualität der professionellen Software mit der Qualität der in den Teilprojekten 2, 3, 4 entwickelten Pakete.

- Teilprojekt 6:
  Wenn man in den Endpunkten des Intergrationsintervalls Ableitungswerte des Integranden berechnet, kann man mit Hilfe der Euler-McLaurin'schen Summenformel Näherungen des Integrals gewinnen, die genauer (d.h. von höherer Ordnung) sind als die Trapezsumme. Eine oft sinnvolle Modifikation dieser Vorgangsweise besteht darin, die benötigten Ableitungswerte am Rand durch numerische Differentiation (asymptotisch gut genug) zu gewinnen. Basierend auf diesen Ideen schreiben Sie drei Programmversionen[15] zur numerischen Integration, allerdings ohne jegliche Steuerungsmaßnahmen.

  Version 1: Trapezsummenextrapolation der Ordnung 2k, basierend auf der Bulirschfolge[16].

  Version 2: An den Rändern modifizierte Trapezregel der Ordnung 2k, basierend auf exakten Ableitungsberechnungen.

  Version 3: An den Rändern modifizierte Traperegel der Ordnung 2k, basierend auf asymptotisch hinreichend genauen numerischen Differentiationen an den Rändern.

  Integrieren Sie selbstgewählte Testbeispiele mit diesen drei Versionen auf immer feineren Gittern und vergleichen Sie die dabei auftretenden Verfahrensfehler. Im Hinblick auf einen fairen Vergleich stellen Sie nur jene Fälle gegenüber, die bezüglich aller drei Versionen auf derselben Ordnung 2k und auf derselben Anzahl von Funktionsauswertungen beruhen.

  Diskutieren Sie, wie man die drei Versionen mit Hilfe von Steuermaßnahmen zu vollwertigen Programmen zur numerischen Integration erweitern kann. Was spricht im Hinblick auf solche Überlegungen für (bzw. gegen) die einzelnen Versionen?

[15] Genaugenommen drei Gruppen von Programmversionen, da man in jedem Fall auch unterschiedliche Konvergenzordnungen $2k(k = 1, 2, \ldots, k_{\max})$ betrachtet.

[16] Die Bulirschfolge ensteht aus der schnell abnehmenden Folge $h_n = \frac{(b-a)}{2^n}$ durch dazwischen schalten der Schrittweiten $\frac{(b-a)}{3 \cdot 2^n}$

Projekt IV/2: Numerische Quadratur - Version 2

Dieses Projekt hat zum Ziel, die Eigenschaften der Verfahrensfehler anhand der numerischen Quadratur (Integration) zu diskutieren. Dabei soll auch gelernt werden, wie man Verfahrensfehler schätzen kann.

Vorbemerkungen

Um eine Näherung für das bestimmte Integral

$$I(f) = \int_a^b f(x)\,dx$$

zu berechnen, werden vorwiegend interpolatorische Quadraturformeln eingesetzt, d. h., man ersetzt den Integranden f durch eine stückweise polynomiale Interpolationsfunktion und nimmt den leicht zu berechnenden Integralwert der Interpolationsfunktion als Approximation für das gesuchte Integral. Die daraus resultierende Näherungsformel ist typischerweise eine Linearkombination von Funktionswerten $f(x_i)$, d. h. sie ist von der Form

$$I_h(f) := \sum_i c_i\, f(x_i), \quad a \le x_i \le b,$$

wobei die Stellen x_i als "Knoten" und die Koeffizienten c_i als "Gewichte" der Integrationsformel bezeichnet werden. Den Verfahrensfehler solcher Formeln kann man leicht durch Integration des Interpolationsfehlers berechnen bzw. abschätzen.

Unterteilt man beispielsweise das Integrationsintervall $[a, b]$ in n gleiche Teile der Länge $h = \frac{b-a}{n}$, was zu einem äquidistanten Gitter mit den Gitterpunkten $x_i = a + ih$, $i = 0, 1, \ldots, n$ führt, und interpoliert man auf jedem der Teilintervalle die Daten $(x_{i-1}, f(x_{i-1})), (x_i, f(x_i))$ linear (Polynomgrad 1), so entsteht die sogenannte Trapezregel

$$I(f) \approx T_h(f) := h \cdot \left[\frac{1}{2}f(a) + f(x_1) + \cdots + f(x_i) + \cdots + f(x_{n-1}) + \frac{1}{2}f(b)\right].$$

Interpoliert man bezüglich desselben Gitters auf den Teilintervallen

$$[a, x_2], \quad [x_2, x_4], \quad \ldots$$

jeweils die Datensätze

$$(a, f(a)), (x_1, f(x_1)), (x_2, f(x_2)), \quad (x_2, f(x_2)), (x_3, f(x_3)), (x_4, f(x_4)), \quad \ldots$$

mit Polynomen 2. Grades, so ergibt sich die Simpsonregel

$$I(f) \approx S_h(f) := h \cdot \left[\frac{1}{3}f(a) + \frac{4}{3}f(x_1) + \frac{2}{3}f(x_2) + \frac{4}{3}f(x_3) + \cdots + \frac{4}{3}f(x_{n-1}) + \frac{1}{3}f(b)\right].$$

Bei stückweise polynomialer Interpolation mit Polynomen vom Grad 4 ergibt sich die Milneregel

$$\begin{aligned} I(f) \approx M_h(f) \;&:=\; h \cdot \left[\frac{14}{45}f(a) + \frac{64}{45}f(x_1) + \frac{24}{45}f(x_2) + \frac{64}{45}f(x_3) + \frac{28}{45}f(x_4)\right. \\ &+\; \left.\frac{64}{45}f(x_5) + \cdots + \frac{64}{45}f(x_{n-1}) + \frac{14}{45}f(b)\right]. \end{aligned}$$

Solche Näherungsformeln, die auf äquidistanter Interpolation beruhen, heißen Newton-Cotes-Formeln.

Die bekanntesten auf nichtäquidistanter Interpolation basierenden Integrationsformeln sind die Gauß-Formeln. Bei der Konstruktion dieser Formeln wählt man die Gewichte *und* Knoten so, dass Polynome möglichst hohen Grades exakt integriert werden. Diese Formeln werden bezüglich des Standardintervalls $[-1,+1]$ angegeben. Die ersten Gauß-Formeln lauten:

$$\int_{-1}^{+1} f(x)\,dx \approx G_1(f) := 2f(0),$$

$$\int_{-1}^{+1} f(x)\,dx \approx G_2(f) := f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right),$$

$$\int_{-1}^{+1} f(x)\,dx \approx G_3(f) := \frac{5}{9}\,f\left(-\sqrt{\frac{3}{5}}\right) + \frac{8}{9}f(0) + \frac{5}{9}\,f\left(\sqrt{\frac{3}{5}}\right).$$

Mit G_1 werden Polynome bis zum Grad 1, mit G_2 bis zum Grad 3 und mit G_3 bis zum Grad 5 exakt integriert.

Will man diese Formeln auf einem allgemeinen Intervall $[a,b]$ statt auf dem Standardintervall $[-1,+1]$ verwenden, so muss man sowohl die Integrationsgewichte als auch die Knoten entsprechend transformieren. Setzt man die Gauß-Formeln stückweise ein, so muss man sie auf entsprechende Teilintervalle des Integrationsintervalls transformieren.

Weitere Integrationsmethoden enstehen dadurch, dass man die Idee der Extrapolation für die interpolatorischen Quadraturformeln realisiert. Die auf der Trapezregel basierende Extrapolationsmethode wird als Romberg-Integration bezeichnet. Um diese Methode herzuleiten, geht man von der asymptotischen Fehlerentwicklung für den Fehler der Trapezregel aus,

$$T_h(f) = \int_a^b f(t)dt + c_1h^2 + c_2h^4 + c_3h^6 + \cdots + c_mh^{2m} + O(h^{2m+2}). \tag{1}$$

Dabei sind die Konstanten c_k von h unabhängig, im Term $O(h^{2m+2})$ tritt noch eine höhere Ableitung von f. Diese Darstellung des Fehlers mit der festen Länge (die von der Glattheit von f bestimmt wird) beschreibt sein Verhalten für $h \to 0$. Man kann zeigen, dass die Trapezregel eine glatte Funktion in h^2 ist, die klarerweise für $h = 0$ den Wert des Integrals annimt. Die Idee ist nun $T_h(f)$ durch ein Polynom in h^2 (!) zu approximieren und den Wert dieses Polynomes an der Stelle $h = 0$ als neue Approximation für den Integralwert zu nehmen. Dazu legt man *gedanklich* ein Polynom m-ten Grades durch die $m+1$ Wertepaare

$$(h_1^2, T_{h_1}(f)),\ (h_2^2, T_{h_2}(f)),\ \cdots,\ (h_m^2, T_{h_m}(f)),\ (h_{m+1}^2, T_{h_{m+1}}(f))$$

durch und wertet es an der Stelle $h = 0$ aus. Die so erhaltene Näherung approximiert das Integral auf dem Niveau $O(h^{2m+2})$, obwohl für alle $T_{h_k}(f)$ nur $O(h^2)$ gilt. Bei der praktischen Realisierung der Romberg-Integration stellt man das Polynom nicht wirklich auf. Es ist mit Hilfe des Neville-Schemas möglich, seinen Wert auf der Stelle $h = 0$ zu berechnen, ohne das man das Polynom explizit kennt.

Beispiel 1

Zeigen Sie, dass für den Verfahrensfehler der Trapezregel die folgende a-priori Abschätzung gilt:

$$\left| T_h(f) - \int_a^b f(x)\,dx \right| \leq \frac{M_2(b-a)}{12} h^2, \text{ für } \max_{a\leq x\leq b} |f''(x)| \leq M_2,$$

Dazu betrachten Sie zunächst den Verahrensfehler der Trapezregel auf dem Teilintervall $I_i := [x_i, x_{i+1}]$,

$$T_{h,I_i}(f) - \int_{x_i}^{x_{i+1}} f(t)dt =: \int_{x_i}^{x_{i+1}} P_1(t)dt - \int_{x_i}^{x_{i+1}} f(t)dt,$$

wobei mit P_1 das Polynom 1. Grades bezeichnet wird, das die Werte $f(x_i)$ und $f(x_{i+1})$ interpoliert. Weiters gilt die folgende Formel für den Interpolationsfehler einer Funktion $f \in C^2[x_i, x_{i+1}]$:

$$P_1(t) - f(t) = -\frac{f''(\xi_i)}{2!}(t - x_i)(t - x_{i+1})$$

für jedes t und ein festes ξ_i, beide in $[x_i, x_{i+1}]$.

Um unterschiedliche Integrationsformeln bezüglich ihrer Effzienz zu vergleichen, betrachten Sie die folgenden Testbeispiele:

$$\int_0^1 e^x\,dx = e - 1, \quad \int_0^1 e^{10x}\,dx = \frac{1}{10}\left(e^{10} - 1\right). \tag{2}$$

Berechnen Sie Approximationen dieses Integrals basierend auf 5 Funktionsauswertungen

- mit der Trapezregel,
- mit der Romberg-Integration,
- mit der Milneregel,
- mit G_5, siehe M. Abramowitz, I. Stegun: Handbook of Mathematical Functions,

und vergleichen Sie ihre absoluten und relativen Genauigkeiten.

Beispiel 2

Zur Illustration der numerischen Faustformel

Verfahren niederer Ordnung sind effizient für grobe Genauigkeitsanforderungen, Verfahren hoher Ordnung sind effizient für strenge Genauigkeitsanforderungen

realisieren Sie folgende numerischen Experimente:

Wählen Sie Testbeispiele mit bekannter Lösung, bei denen die Glattheit des Integranden (d. h. die Größenordnungen der Ableitungen) am ganzen Integrationsintervall nicht schwankt, sodass äquidistante Gitter als sinnvoll erscheinen. Als numerische Verfahren wählen Sie die Trapezregel $T_h(f)$ und zwei Extrapolationsverfahren, $T_{h,6}^{extr}(f)$ und $T_{h,8}^{extr}(f)$, der Ordnungen $O(h^6)$ und $O(h^8)$. Geben Sie eine Genauigkeitsforderung vor und prüfen sie, wieviele Funktionsauswertungen bei jedem einzelnen Verfahren notwendig sind, um das vorgegebene Genauigkeitsniveau zu erreichen. Da Sie über keine

Fehlerschätzung verfügen, vergleichen Sie den exakten Fehler mit der vorgegebenen Genauigkeitsforderung.

Experimentieren Sie auch mit weniger glatten Testbeispielen, d. h. mit Beispielen, bei denen Werte gewisser Ableitungen von f groß werden, oder Unstetigkeiten von Ableitungen auftreten.

Stellen Sie den Aufwand, gemessen durch die Anzahl der Funktionsauswertungen, als Funktion der absoluten Genauigkeit für die einzelnen Verfahren dar. Die drei Verfahrenskurven sollten dabei immer gemeinsam geplottet werden, um Vergleiche zu ermöglichen.

Beispiel 3

Bei der Implementierung numerischer Integrationsverfahren stattet man die oben vorgestellten Quadraturformeln mit zahlreichen Steuerungsmaßnahmen aus (Schrittweitensteuerung, Ordnungssteuerung, Fehlerschätzungen). Ziel solcher Maßnahmen ist es, in möglichst effizienter Weise ein durch Toleranzparameter vorgegebenes Genauigkeitsniveau zu erreichen. Stellen Sie anhand selbstgewählter Beispiele mit bekannter Lösung den Toleranzparameter dem tatsächlichen (Verfahrens-)Fehler gegenüber. Benützen Sie dabei die vorhandene numerische Software (z.B. aus MATLAB) zur Durchführung der Experimente.

Beispiel 4

Um zu überprüfen, ob ein Näherungswert für $I(f) := \int_a^b f(t)dt$, den man mit Hilfe von $T_h(f)$ berechnet hat, die vom Benutzer vorgeschriebene Toleranzanforderung erfüllt, benötigt man eine Fehlerschätzung für den unbekannten Fehler $T_h(f) - I(f)$. Eine solche a-posteriori Fehlerschätzung kann man wie folgt herleiten: Man berechnet $T_h(f)$ für drei verschiedene Schrittweiten, z. B. $h, h/2$ und $h/4$, und schreibt gemäß Formel (1),

$$T_h(f) = I(f) + c_1 h^2 + c_2 h^4 + O(h^6), \tag{3a}$$

$$T_{h/2}(f) = I(f) + c_1 \frac{h^2}{4} + c_2 \frac{h^4}{16} + O(h^6), \tag{3b}$$

$$T_{h/4}(f) = I(f) + c_1 \frac{h^2}{16} + c_2 \frac{h^4}{256} + O(h^6). \tag{3c}$$

Für den Fehler des *genaueren* Wertes $T_{h/2}(f)$ kann man durch eine entsprechende Manipulation von (11a) und (12) eine Fehlerschätzung der Form

$$T_{\alpha\beta}(f) = \alpha T_h(f) + \beta T_{h/2}(f)$$

herleiten. Geben Sie diese Formel an.

Beschreiben Sie weiters ihre asymptotische Qualität, d. h. geben Sie die Potenz q in der folgenden Abschätzung an:

$$\left| \underbrace{T_{h/2}(f) - I(f)}_{\text{Fehler}} - \underbrace{(\alpha T_h(f) + \beta T_{h/2}(f))}_{\text{Fehlerschätzung}} \right| \leq \text{Kenngrößen des Problems} \cdot h^q.$$

Eine Fehlerschätzung für $T_{h/4}(f)$ kann man herleiten indem man alle drei Formeln aus (11) einbezieht. Geben Sie auch diese Formel[17] an,

$$T_{\alpha\beta\gamma}(f) = \alpha T_h(f) + \beta T_{h/2}(f) + \gamma T_{h/4}(f),$$

und beschreiben Sie ihre asymptotische Qualität. Anhand selbstgewählter Beispiele testen Sie die Zuverlässigkeit der obigen Schätzformel. Versuchen Sie ein Beispiel zu konstruieren, wo ihre Fehlerschätzung versagt.

Beispiel 5

Man soll eine asymptotisch korrekte Fehlerschätzung für die Gaußformel $G_3(f)$ anbieten. Die Idee ist, als Schätzung für den unbekannten Fehler $G_3(t) - \int_a^b f(t)dt$ die Differenz $G_3(f) - G_5(f)$ zu nehmen. Zunächst stellen Sie experimentell fest mit welcher Potenz von h der Fehler von $G_3(f)$ für $h \to 0$ abnimmt. Dazu machen Sie für die beiden Testbeispiele (2) das folgende Experiment: Unterteilen Sie das Interval $[0, 1]$ in äquidistante Teilintervalle der Länge h. Wenden Sie darauf stückweise die entsprechend transformierte Formel $G_3(f)$ an. Das Ergebnis wird mit $G_h^3(f)$ bezeichnet, um die Abhängigkeit von h anzudeuten.

Führen Sie die soeben beschriebene Näherungsformel für eine Reihe feiner werdendenden Schrittweiten h aus und berechnen Sie für jede dieser Schrittweiten den Verfahrensfehler $|G_h^3(f) - \int_a^b f(t)dt|$. Aus dem Ansatz

$$\left| G_h^3(f) - \int_a^b f(t)dt \right| \approx c \cdot h^p, \quad h \to 0,$$

kann man für jeweils zwei aufeinanderfolgende Schrittweiten eine Schätzung für die, von h unabhängige, Fehlerkonstante c und die Ordnung p ausrechnen.

Wir wollen eine asymptotisch korrekte Fehlerschätzung anbieten, d. h. die Potenz q in

$$\left| \underbrace{G_h^3(f) - \int_a^b f(t)dt}_{\text{Fehler}} - \underbrace{(G_3(f) - G_5(f))}_{\text{Fehlerschätzung}} \right| = \left| G_h^5(f) - \int_a^b f(t)dt \right| \approx c \cdot h^q, \quad h \to 0,$$

muss $q \geq p + 1$ erfüllen. Überprüfen Sie diese Abschätzung durch analoge Experimente mit den Modelproblemem (2). Anhand selbstgewählter Beispiele verschiedener Schwierigkeitsgrade testen Sie die Zuverlässigkeit der soeben entwickelten Schätzprozedur.

Beispiel 6

Implementieren Sie eine der obigen Prozeduren zur Schätzung des Diskretisierungsfehlers bei der numerischen Integration und versuchen Sie eine Strategie zur Schrittweitensteuerung darauf aufzubauen. Eine solche Steuerung versucht mit möglich wenig Funktionsauswertungen die Toleranzforderung

$$|\text{Fehlerschätzung}| \leq c \cdot \max\{Tol_a, Tol_r\, |\text{Näherung}|\}$$

zu erfüllen. Dabei ist $0 < c \leq 1$ ein Sicherheitsfaktor, und Tol_a, Tol_r die vom Benützer vorgegebene Schranken für den absoluten und den relativen Gesamtfehler. Testen Sie Ihr Programm anhand selbstgewählter Beispiele. Versuchen Sie Situtionen zu finden, wo Ihr Programm versagt.

[17] Hier können die Koeffizienten α und β andere Werte haben als in der Formel für $T_{\alpha\beta}(f)$.

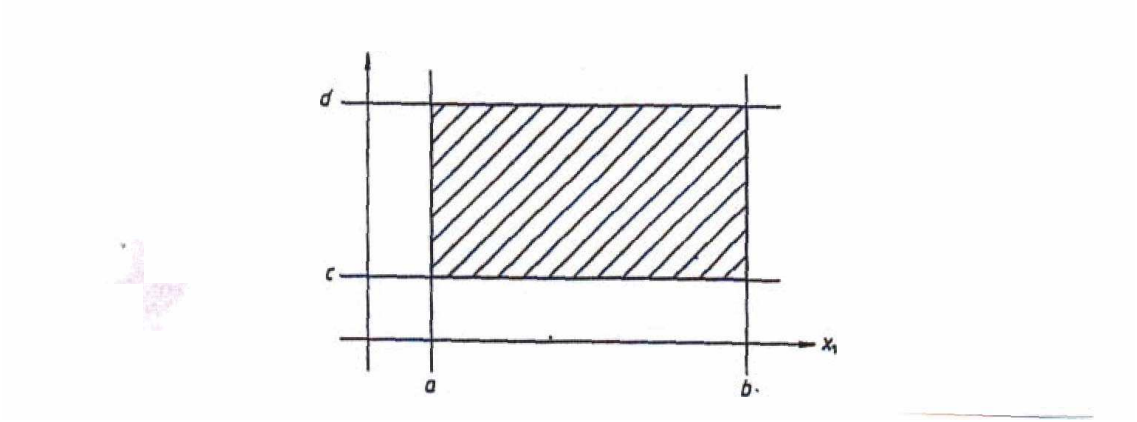
Projekt IV/3: Numerische Quadratur - zweidimensional

Sowohl bei Rechtecksbereichen als auch bei allgemeinen Gebieten läßt sich die mehrdimensionale Integration im Prinzip auf "verschachtelte" eindimensionale Integrationsaufgaben zurückführen. Z. B. im $\mathbb{R}^2$ gilt:

a) Rechtecksbereich:

$$\int_a^b \int_c^d f(x_1, x_2) dx_1 dx_2 = \int_a^b [\int_c^d f(x_1, x_2) dx_2] dx_1 = \int_a^b F(x_1) dx_1$$

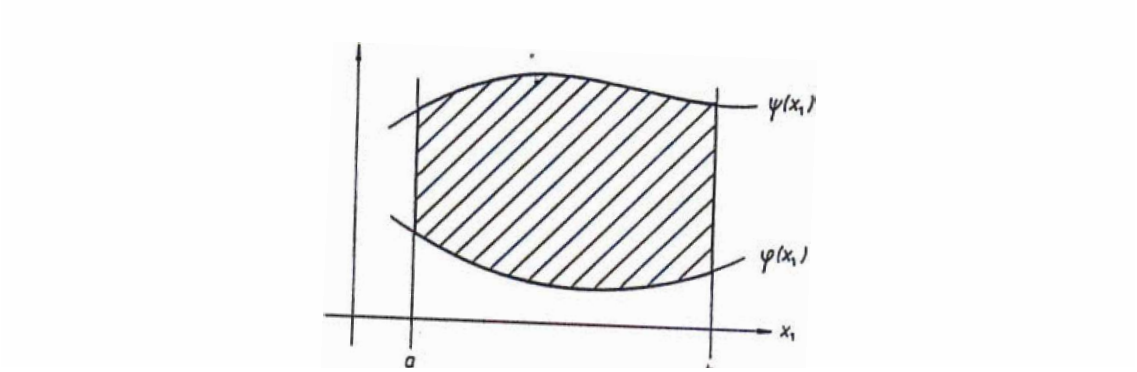
mit $F(x_1) := \int_c^d f(x_1, x_2) dx_2$.



b) Allgemeines Gebiet:

$$\int_a^b \int_{\varphi(x_1)}^{\psi(x_1)} f(x_1, x_2) dx_1 dx_2 = \int_a^b [\int_{\varphi(x_1)}^{\psi(x_1)} f(x_1, x_2) dx_2] dx_1 = \int_a^b F(x_1) dx_1$$

mit $F(x_1) := \int_{\varphi(x_1)}^{\psi(x_1)} f(x_1, x_2) dx_2$. Aufgrund dieser Tatsache lassen sich sofort numerische In-



tegrationsformeln für mehrdimensionale Integrale aufstellen, indem man auf die üblichen Verfahren für die eindimensionale Integration zurückgreift: Z. B. ergibt sich die "zweidimensionale Trapezregel" gemäß

$$\begin{aligned}
&\int_a^b \int_c^d f(x_1,x_2)dx_1dx_2 = \int_a^b F(x_1)dx_1 \approx \\
\approx\ & h(\frac{1}{2}F(a) + F(x_{1,1}) + \ldots + F(x_{1,\nu}) + \ldots \frac{1}{2}F(b)) = \\
=\ & h(\frac{1}{2}\int_c^d f(a,x_2)dx_2 + \int_c^d f(x_{1,1},x_2)dx_2 + \ldots + \int_c^d f(x_{1,\nu}x_2)dx_2 + \ldots \\
+\ & \frac{1}{2}\int_c^d f(b,x_2)dx_2)
\end{aligned}$$

mit $x_{1,\nu} = a + \nu \cdot h, h = \frac{b-a}{n}, \nu = 0(1)n$, wobei man jetzt jedes eindimensionale Integral mit der Trapezregel berechnet:

$$\int_c^d f(x_{1,\nu}x_2)dx_2 \approx k(\frac{1}{2}f(x_{1,\nu},c) + f(x_{1,\nu},x_{2,1}) + \ldots + f(x_{1,\nu},x_{2,\mu}) + \ldots \frac{1}{2}f(x_{1,\nu},d))$$

mit $x_{2,\mu} = c + \mu \cdot k, k = \frac{d-c}{m}, \mu = 0(1)m$.

Insgesamt entsteht eine Integrationsformel

(i) mit den Gewichten $h \cdot k$ an allen inneren Gitterpunkten $(1 \leq \nu \leq n-1, 1 \leq \mu \leq m-1)$,

(ii) mit den Gewichten $\frac{hk}{2}$ an allen Randpunkten

$$(\nu = 0, n; 1 \leq \mu \leq m-1 \quad \text{oder} \quad 1 \leq \nu \leq n-1, \mu = 0, m;)$$

(iii) mit den Gewichten $\frac{hk}{4}$ an den 4 Eckpunkten

$$(\nu = 0, \mu = 0; \nu = 0, \mu = m; \nu = n, \mu = 0; \nu = n, \mu = m) \ .$$

Auch Verfahrensfehlerabschätzungen lassen sich sofort auf die bekannten Abschätzungen im eindimensionalen Fall zurückführen: der Verfahrensfehler der "inneren" Integration läßt sich als Datenfehler für die "äußere" Integration interpretieren. Die Ideen für die Steuerungsmaßnahmen (Schrittweitensteuerung, Abbruchkriterium, u.s.w.) für den eindimensionalen Fall lassen sich ebenfalls in natürlicher Weise auf den mehrdimensionalen Fall übertragen.

- Teilprojekt 1:

  (i) Formulieren Sie die Trapezregel für n-dimensionale Integrale über Rechtecksbereiche.

  (ii) Formulieren Sie einige Newton-Cotes Formeln für zweidimensionale Integrale über Rechtecksbereiche.

  (iii) Geben Sie für die in (i) und (ii) entwickelten Integrationsmethoden Verfahrensfehlerschranken an.

  (iv) Formulieren Sie (basierend auf eigenen Überlegungen und unter zuhilfenahme der einschlägigen numerischen Literatur) sinnvolle Steuerungsmaßnahmen für die in (ii) entwickelten Verfahren.

(v) Schreiben Sie ein Programm, das auf den in (ii) entwickelten Verfahren basiert (ev. mit Steuerungsmaßnahmen) und testen Sie es anhand von selbstkonstruierten Testbeispielen.

- Teilprojekt 2:

  Wie Teilprojekt 1 (ohne (i)), aber statt auf den Newton-Cotes Formeln auf Gauß-Formeln beruhend.

- Teilprojekt 3:

  Wie Teilprojekt 1, aber auf der Trapezsummenextrapolation basierend. In diesem Fall soll auch auf die Verfahrensfehlerabschätzungen verzichtet werden.

- Teilprojekt 4:

  (i) Entwickeln Sie ein Verfahren zur zweidimensionaler Integration über krummlinige Bereiche.

  (ii) Geben Sie eine Verfahrensfehlerschranke an.

  (iii) Überlegen Sie sinnvolle Steuerungsmaßnahmen.

  (iv) Programmieren Sie Ihr Verfahren und testen Sie es.

- Teilprojekt 5:

  Testen Sie die in den Programmbibliotheken vorhandene Software für mehrdimensionale Integration. Ziehen Sie bei den Experimenten auch hochdimensionale Integrale auf Rechtecksbereichen und auf krummlinigen Bereichen heran.

Projekt IV/4: Nichtlineare Modelle - Version 1

- Teilprojekt 1:
  Untersuchen Sie experimentell mit Software aus den Programmbibliotheken die Konvergenzeigenschaften von Verfahren zur Lösung von Nullstellenproblemen der Form $F(x) = 0$, $F : \mathbb{R}^n \to \mathbb{R}^n$. Zu diesem Zweck konstruieren Sie Testbeispiele mit bekannter Lösung x^*. Variieren Sie die Schwierigkeit der Testbeispiele, indem Sie

  (i) die Dimension n variieren ($n = 2$ oder 3, bzw. $n \approx 10$),

  (ii) F mehr oder weniger glatt wählen (F hat moderate Ableitungen oder gewisse Ableitungen von F werden groß oder existieren nicht),

  (iii) den Separiertheitsgrad der Nullstelle x^* variieren (es gibt eine nicht zu kleine Umgebung von x^* mit keiner weiteren Nullstelle, oder es gibt weitere Nullstellen von F sehr nahe an x^*, oder x^* ist eine mehrfache Nullstelle).

  Bei zweidimensionalen Testbeispielen zeichnen Sie die Einzugsbereiche der Nullstelle x^*. In höherdimensionalen Fällen, wo die experimentelle Gewinnung der Einzugsbereiche zu aufwendig wäre, geben Sie einen Eindruck vom Konvergenzverhalten, indem Sie einige von x^* weggehende Richtungen auswählen, und prüfen wie weit Sie in diese Richtungen gehen können bis die Konvergenz ausbleibt.

- Teilprojekt 2:
  Bekanntlich besitzt das gedämpfte Newton-Verfahren bessere Konvergenzeigenschaften als das reine Newton-Verfahren, d.h. der Einzugsbereich einer Nullstelle sollte beim gedämpften Newton-Verfahren größer sein.

  (i) Geben Sie aufgrund des Vorlesungsskriptums und unter Heranziehung der einschlägigen numerischen Literatur einen Überblick über die unterschiedlichen algorithmischen Varianten des gedämpften Newton-Verfahrens und über die theoretischen Aussagen, die deren bessere Konvergenzeigenschaften sicherstellen sollen. Untersuchen Sie die Skalierungsinvarianz des reinen Newton-Verfahrens und des gedämpften Newton-Verfahrens nach Vorskalierung mit der Inversen der lokalen Jacobi-Matrix (im Gegensatz zum nichtvorskalierten gedämpften Newton-Verfahren).

  (ii) Programmieren Sie das reine und eine Variante des gedämpften Newton-Verfahrens (mit Vorskalierung).

  (iii) Vergleichen Sie die Einzugsbereiche dieser beiden Varianten. Ziehen Sie dazu auch die Beispiele aus Teilprojekt 1 heran.

- Teilprojekt 3:
  Konstruieren Sie Testbeispiele der Form $F(x) = 0$ und betrachten Sie Umskalierungen dieser Nullstellenprobleme vom Typ $AF(x) = 0$ mit verschiedenen Matrizen A (Koeffizienten von A groß oder klein, A gut oder schlecht konditioniert, etc.). Testen Sie, ob sich bei Verwendung der Nullstellenprogramme aus den Bibliotheken Einzugsbereiche von der Nullstelle x^* einstellen, die i.w. invariant bezüglich solcher Umskalierungen sind.

- Teilprojekt 4:
  Polynome $P_n(x)$ vom Grad n mit den reellen Nullstellen $x_1, \ldots, x_n$ können gemäß

$$P_n(x) = (x - x_1)(x - x_2) \cdots (x - x_n)$$

  dargestellt werden. Schreibt man dasselbe Polynom in der Standardschreibweise

$$P_n(x) = x^n - c_1 x^{n-1} + \cdots + (-1)^n c_n, \tag{1}$$

  so gelten bekanntlich folgende Beziehungen:

$$\begin{aligned} c_1 &= x_1 + x_2 + \cdots + x_n, \\ c_2 &= x_1x_2 + x_1x_3 + \cdots + x_1x_n + \cdots + x_{n-1}x_n, \\ c_3 &= x_1x_2x_3 + x_1x_2x_4 + \cdots + x_{n-2}x_{n-1}x_n, \\ &\vdots \\ c_n &= x_1x_2 \cdots x_n. \end{aligned} \tag{2}$$

  Konstruieren Sie Testbeispiele durch Vorgabe der Nullstellen. Kritische Fälle ergeben sich durch Wahl mehrfacher Nullstellen bzw. durch nahe zusammen liegende Nullstellen.

  Es gibt u.a. folgende Möglichkeiten die Nullstellen von P_n numerisch zu berechnen:

  (i) Die einzelnen Nullstellen werden nach und nach mit dem skalaren Newton-Verfahren berechnet und anschließend abdividiert.

  (ii) Die Nullstellen werden simultan, durch Lösen von (2) mit dem Newton-Verfahren für ein System nichtlinearer Gleichungen bestimmt.

  Vergleichen Sie diese beiden Möglichkeiten, vor allem hinsichtlich ihrer numerischen Stablitität in kritischen Fällen, indem Sie beide Varianten selbst programmieren und anhand von Testbeipielen unterschiedlichen Schwierigkeitsgrades testen. Vergleichen Sie Ihre Programme auch mit numerischer Software zur Ermittlung von polynomialen Nullstellen bzw. zur Lösung des nichtlinearen Gleichungssystems (2).

- Teilprojekt 5:
  Beim reinen Newton-Verfahren wird bekanntlich die Jacobi-Matrix in jedem Schritt neuerlich ausgewertet. Da einerseits das Aufstellen der Jacobi-Matrix, d.h. das Berechnen sämtlicher partieller Ableitungen und deren Eingabe als Funktionsprozedur unbequem ist und es andererseits rechenaufwendig ist, bei der Neuauswertung der Jacobi-Matrix in jedem Schritt eine neuerliche Dreieckszerlegung durchzuführen, gibt es auch Varianten des Newton-Verfahrens, bei denen die Jacobi-Matrix durch numerische Differentiation näherungsweise berechnet wird und (oder) über mehrere (alle) Newton-Schritte eingefroren bleibt (was einige Dreieckszerlegungen erspart).

  Überlegen Sie sinnvolle Steuermaßnahmen der eben beschriebenen Varianten des Newtonverfahrens z.B.: Wie soll die numerische Differentiation zu Berechnung der partiellen Ableitungen realisiert werden? Kann man mit ausreichender Genauigkeit numerische Approximationen dieser Ableitungen berechnen, die ausschließlich auf Funktionsauswertungen $F(x_i)$, $i = 0, 1, 2, \ldots$

beruhen (wobei mit x_i, die Folge der Näherungsvektoren bezeichnet wird, die das Newton-Verfahren liefert) oder sind für die numerische Differentiation zusätzliche Funktionsauswertungen nötig? Sind a-posteriori Fehlerschätzungen bez. der numerischen Differentiation von Aufwand und allfälligem Nutzen her vertretbar? Versuchen Sie ähnliche Fragestellungen – nicht nur bez. der numerischen Differentiation, sondern auch bez. allfälligen Einfrierens der Jacobi-Matrix – zu formulieren und suchen Sie nach plausiblen Antworten! Was finden Sie zu dieser Thematik in der numerischen Literatur? Realisieren Sie – als Ergebnis Ihrer Überlegungen – eine Variante des Newton-Verfahrens mit numerisch näherungsweiser Berechnung der Jacobi-Matrix und ihrem (teilweise) Einfrieren und überprüfen Sie anhand selbstgewählter Beispiele die Zweckmäßigkeit Ihrer Variante. Programmieren Sie auch das Standard Newton-Verfahren und vergleichen Sie Ihre Routine damit. Was finden Sie bezüglich vereinfachter Varianten des Newton-Verfahrens in den Programmbibliotheken? Gibt die Dokumentation ausreichende Informationon über den Ablauf und die Steuerung dieser Varianten?

Projekt IV/5: Nichtlineare Modelle - Version 2

- Teilprojekt 1. Das folgende skalare Randwertproblem für eine gewöhnliche Differentialgleichung zweiter Ordnung,

$$y''(t) = f(t, y(t)), \quad t \in [a, b], \tag{1a}$$

$$y(a) = \alpha, \quad y(b) = \beta, \tag{1b}$$

soll numerisch möglichst effizient gelöst werden. Dazu wird zunächst ein equidistantes Gitter

$$\Delta_h := \{t_i = a + ih, i = 0, 1, \ldots, N, \;\; h = (b-a)/N\}$$

eingeführt und die obige Gleichung wie folgt diskretisiert. Sei $y_i \approx y(t_i)$, dann lautet das diskrete Schema für den unbekannten Vektor $Y_h := (y_1, y_2, \ldots, y_{N-1})^T$,

$$F_h(Y_h)_i := \frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} - f(t_i, y_i) = 0, \quad i = 1, 2, \ldots, N-1, \tag{2a}$$

$$y_0 = \alpha, \quad y_N = \beta. \tag{2b}$$

Schreiben Sie dieses Schema in Form eines nichtlinearen Gleichungssystems für Y_h um,

$$F_h(Y_h) = 0. \tag{3}$$

Dieses System wird mit Hilfe des Newton Verfahrens gelöst. Dazu linearisieren Sie (3) und leiten Sie die Gestalt der Koeffizientenmatrix L_h in dem sich aus der Linearisierung ergebenden linearen Gleichungssystem. Dazu nehmen Sie an, dass die aktuelle Näherung für Y_h mit Y_h^k gegeben ist.

Beweisen Sie: Für entsprechend kleine Schrittweite h, und für $f_y \geq 0$, bzw. für $|f_y| \leq \kappa < \frac{8}{(b-a)^2}$ ist die Matrix L_h invertierbar und für ihre Inverse gilt,

$$\|L_h^{-1}\|_\infty \leq \begin{cases} \frac{(b-a)^2}{8}, & f_y \geq 0, \\ \frac{1}{\frac{8}{(b-a)^2} - \kappa}, & |f_y| \leq \kappa < \frac{8}{(b-a)^2}. \end{cases} \tag{4}$$

Implementieren Sie die im Skriptum vorgeschlagene Variante des Newton Verfahren, um den Vektor Y_h zu berechnen und testen Sie Ihr Programm an Beispielen mit bekannten Lösungen. Speziell, verkleinern Sie die Schrittweite kohärent so lange bis die für den absolution Fehler vorgeschriebene Toleranz erreicht wurde. Stellen Sie für verschiedene Testbeispiele die erreichte Genauigkeit dem Aufwand gegenüber. Testen Sie insbesondere die unten stehenden Randwertprobleme:

$$y''(t) = \mu \sinh(\mu t), \quad 0 \leq t \leq 1, \tag{5a}$$

$$y(0) = 0, \quad y(1) = 1, \tag{5b}$$

und

$$y''(t) = \frac{-3\varepsilon y(t)}{(\varepsilon + t^2)^2}, \quad -0.1 \leq t \leq 0.1, \tag{6a}$$

$$y(-0.1) = -\frac{0.1}{\sqrt{\varepsilon + 0.01}}, \quad y(0.1) = \frac{0.1}{\sqrt{\varepsilon + 0.01}}, \tag{6b}$$

für groß werdende Werte von μ und klein werdende Werte von ε, und verschiedene Toleranzen.

- Teilprojekt 2. Für das im Teilprojekt 1 vorgestellte numerische Schema, beobachten sie die Ordnung des Verfahrens für $h \to 0$. Konstruieren Sie dazu Testbeispiele, mit verschieden schwierigen Lösungen. Lösen Sie zunächst das System (2) auf dem Gitter Δ_{h_1}, anschliessend, auf dem Gitter Δ_{h_2}. Aus dem Ansatz

$$\|y(\Delta_h) - Y_h\|_\infty \approx c\, h^p, \quad h \to 0,$$

wobei $y(\Delta_h) = (y(t_1), y(t_2), \ldots, y(t_{N-1}))^T$, $c \neq c(h)$ und $p > 0$, schätzen Sie die Werte von c und p für $h \to 0$. Experimentieren Sie auch mit weniger glatten Lösungen, d. h. mit Beispielen, bei denen höhere Ableitungen der Lösung groß werden. Der Nachteil, der in (2) beschrieben Verfahrens ist seine niedrige Ordnung. Im Falle von entsprechend glatten Daten, kann man für den globalen Fehler der Diskretizierung (2) die folgende Fehlerdarstellung, die asymptotische Fehlerentwicklung zeigen:

$$y_i - y(t_i) = h^2 e_1(t_i) + h^4 e_2(t_i) + h^6 e_3(t_i) + \ldots + O(h^{2k}), \quad h \to 0, \tag{7}$$

mit glatten, von h unabhängigen Funktionen $e_j(t)$. Nutzen Sie diese strukturelle Eigenschaft, um mit Hilfe der Extrapolationsidee Verfahren 4. und 6. Ordnung zu entwickeln. Für alle drei Verfahren plotten Sie Graphen, wo die erreichte Genauigkeit dem Aufwand gegenüber gestellt wird. Welche Erkentnisse folgen aus diesen Vergleichen?

Nun wollen wir die von Ihnen beobachtete Ordnung des Schemas (2), *im linearen Fall*, $f(t, y) = f(t)$, beweisen. Dazu nehmen wir an, dass (1) eine Lösung hat. Zeigen Sie zuerst die *Konsistenz* des Verfahrens, d.h. die Eigenschaft

$$\|F_h(y(\Delta_h))\|_\infty = O(h^q), \quad h \to 0. \tag{8}$$

Welche Potenz q erhalten Sie? Was müssen Sie voraussetzen, um (8) zu beweisen?
Jetzt zeigen Sie, die *Stabilität* des Schemas: Die Abschätzung

$$\|Y_h^1 - Y_h^2\|_\infty \leq S \|F_h(Y_h^1) - F_h(Y_h^2)\|_\infty \tag{9}$$

gilt mit einer Konstanten $S \neq S(h)$ (für h klein genug). Kombinieren Sie diese beiden Eigenschafen, um zu zeigen, dass

$$\|Y_h - y(\Delta_h)\|_\infty = O(h^q), \quad h \to 0. \tag{10}$$

- Teilprojekt 3. Es gibt zwei Hauptprinzipien, um die a-posteriori Fehlerschätung durchzuführen. Das erste Prinzip basiert auf der Halbierung der Schrittweite. Eine solche a-posteriori Fehlerschätzung kann man wie folgt herleiten: Man berechnet Y_h für zwei verschiedene Schrittweiten, z. B. h und $h/2$ und schreibt gemäß Formel (7),

$$y_i^h = y(t_i) + e_1(t_i) h^2 + O(h^4), \tag{11a}$$

$$y_i^{h/2} = y(t_i) + e_1(t_i) \left(\frac{h}{2}\right)^2 + O(h^4). \tag{11b}$$

Für den Fehler des *genaueren* Wertes $y_i^{h/2}$ kann man durch eine entsprechende Manipulation von (11a) und (12) eine Fehlerschätzung der Form

$$y_i^{\alpha\beta} = \alpha y_i^h + \beta y_i^{h/2}$$

herleiten. Geben Sie diese so Formel an, dass in der folgenden Abschätzung,

$$\left| \underbrace{y_i^{h/2} - y(t_i)}_{\text{Fehler}} - \underbrace{(\alpha y_i^h + \beta y_i^{h/2})}_{\text{Fehlerschätzung}} \right| \leq \text{ Kenngrößen des Problems} \cdot h^q,$$

$q = 4 \geq p + 1$ gilt. In diesem Fall nennt man die Fehlerschätzung asymptotisch korrekt.

Die zweite Idee basiert auf dem Vergleich von Lösungen, die mit Verfahren verschiedenener Ordnungen errechnet wurden. Nehmen wir, dass wir auf dem Gitter Δ_h zwei Lösungen errechnet haben, Y_h^1 und Y_h^2,

$$y_i^1 = y(t_i) + e_1^1(t_i)h^p + O(h^{r_1}), \quad r_1 \geq p+1, \tag{12a}$$
$$y_i^2 = y(t_i) + e_1^2(t_i)h^q + O(h^{r_2}), \quad r_2 \geq q+1, \tag{12b}$$

mit $p < q$. Wir wollen eine asymptotisch korrekte Fehlerschätzung anbieten, d. h. die Potenz q in

$$\left| \underbrace{y_i^1 - y(t_i)}_{\text{Fehler}} - \underbrace{(y_i^1 - y_i^2)}_{\text{Fehlerschätzung}} \right| = \left| y_h^2 - y(t_i) \right| \approx c \cdot h^q, \;\; h \to 0,$$

muss $q \geq p + 1$ erfüllen. Identifizieren Sie Y_h^1 mit dem Schema (2) der Ordnung $p = 2$ und Y_h^2 mit dem Schema nach dem ersten Extrapolationsschritt, der Ordnung $q = 4$ und Uberprüfen Sie experimentell die obige Abschätzung.

Anhand selbstgewählter Beispiele verschiedener Schwierigkeitsgrade testen Sie die Zuverlässigkeit der soeben entwickelten Schätzprozeduren.

- Telprojekt 4. In diesem Teilprojekt geht es um die numerische Lösung von Systemen nichtlinearer gewöhnlicher Differentialgleichungen der Form (Anfangwertaufgabe),

$$y'(t) = f(t, y(t)), \quad t \in [a, b], \quad y(a) = \alpha, \tag{13}$$

wobei $f : [a, b] \times \mathbb{R}^n \to \mathbb{R}^n$ und $\alpha \in \mathbb{R}^n$ gegeben sind. Die numerischen Näherungen y_i für die Werte der exakten Lösung $y(t_i)$, werden auf dem equidistanten Gitter,

$$\Delta_h := \{t_i = a + ih, i = 0, 1, \ldots, N, \;\; h = (b-a)/N\}$$

bereitgestellt. Dabei betrachten wir die folgenden Einschrittverfahren:

$$
\begin{aligned}
X_1 &:= y_{i-1} + h(a_{11}f(\tau_1, X_1) + a_{12}f(\tau_2, X_2) + \cdots + a_{1s}f(\tau_s, X_s)),\\
X_2 &:= y_{i-1} + h(a_{21}f(\tau_1, X_1) + a_{22}f(\tau_2, X_2)) + \cdots + a_{2s}f(\tau_s, X_s),\\
&\vdots\\
X_s &:= y_{i-1} + h(a_{s1}f(\tau_1, X_1) + a_{s2}f(\tau_2, X_2)) + \cdots + a_{ss}f(\tau_s, X_s),\\
y_i &:= y_{i-1} + h(b_1 f(\tau_1, X_1) + b_2 f(\tau_2, X_2)) + \cdots + b_s f(\tau_s, X_s),\\
i &= 1, \ldots, N, \quad y_0 = \alpha,
\end{aligned}
\tag{14}
$$

wobei $\tau_k = t_{i-1} + c_k h$, $k = 1, 2, \ldots, s$. Ist die Matrix $A = (a_{ij})$, $i, j = 1, 2, \ldots, s$ und die Vektoren $b = (b_1, b_2, \ldots, b_s)^T$ und $c = (c_1, c_2, \ldots, c_s)^T$ gegeben, so stellt (14) eine Vorschrift dar, wie man ausgehend von der Näherungslösung (t_{i-1}, y_{i-1}) zur Näherungslösung (t_i, y_i) gelangt. Dabei heißen die Vektoren X_k die Stufen des Verfahrens.

Für die Experimente wählen wir die folgenden drei Verfahren:

Verfahren 1: Implizites Euler Verfahren, Ordnung 1:

$$
\frac{y_i - y_{i-1}}{h} - f(t_i, y_i) = 0, \quad i = 1, \ldots, N, \quad y_0 = \alpha. \tag{15}
$$

Das Verfahren entspricht den Daten $s = 1$, $A = (1)$, $c = (1)$, $b = (1)$, und kann in der Konvention von (14) formal, als folgendes Schema geschrieben werden:

$$
\begin{aligned}
X_1 &:= y_{i-1} + hf(t_i, X_1) \\
y_i &:= y_{i-1} + hf(t_i, X_1) && (16)\\
i &= 1, \ldots, N, \quad y_0 = \alpha, && (17)
\end{aligned}
$$

Verfahren 2: Explizites Runge-Kutta Verfahren, Ordnung 4:

$$
A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} \frac{1}{6} \\ \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{6} \end{pmatrix}, \quad c = \begin{pmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \\ 1 \end{pmatrix}. \tag{18}
$$

Verfahren 3: Implizites Gauß Verfahren, Ordnung 4:

$$
A = \begin{pmatrix} \frac{1}{4} & \frac{1}{4} - \frac{1}{6}\sqrt{3} \\ \frac{1}{4} + \frac{1}{6}\sqrt{3} & \frac{1}{4} \end{pmatrix}, \quad b = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}, \quad c = \begin{pmatrix} \frac{1}{2} - \frac{1}{6}\sqrt{3} \\ \frac{1}{2} + \frac{1}{6}\sqrt{3} \end{pmatrix}. \tag{19}
$$

Implementieren Sie alle drei Verfahren und testen Sie ihre Zuverlässigkeit und Effizienz anhand von selbestgewählten Beispielen mit bekannten Lösungen. Dazu verfeinern Sie h so lange, bis die für den absoluten Fehler der Lösung vorgeschriebene Toleranz erfüllt ist,

$$
\|y_N - y(b)\|_\infty \leq TOL.
$$

Bereiten Sie die Egebnisse graphisch auf, indem Sie die erreichte Genauigkeit mit dem Aufwand (die Anzahl der Auswertungen des Vektors f) vergleichen. Was beobachten Sie?
Testen Sie insbesondere das Prothero-Robinson Problem,

$$y'(t) = A(y(t) - g(t)) + g'(t), \quad y(a) = g(a).$$

Dabei sei A eine Diagonalmatrix, $A = \mathrm{diag}(\lambda_1\, \lambda_2 \ldots \lambda_n)$ bzw. sei A diagonalisierbar mit den Eigenwerten λ_k. Lassen Sie λ_k groß und negativ werden.