Mathematische Anwendungen

WS 2001/02 Prof. L. Griebl

Erstell von Reinhard Ostermeier

Angaben zur Richtigkeit des Skripts

Alle Angaben in diesem Skript sind ohne Gewähr und es wurde nicht genauer auf Fehler überprüft.

Außerdem wird keine Gewähr für die Vollständigkeit gegeben.

Das Benutzen des Skripts ist auf eigene Gefahr.

Es können weder rechtlichen Schritte gegen den Autor des Skripts (Reinhard Ostermeier), noch gegen den vortragenden (Prof. L. Griebl) eingeleitet werden.

Inhaltsverzeichnis

1 Zahlendarstellung

1.1 Gleitpunktzahlen

R ist vollständig archimedisch angeordneter Körper:
Folgerung (u.a.) ***R*** hat unendlich viele Elemente.
Es gibt keine Möglichkeit ***R*** auf einer endlichen Maschine darzustellen! (sogar für $N \in R$ nicht möglich)

Computer arbeiten mit "Maschinen-Zahlen" der Form:
$x = \pm y*b^z$ oder $x = 0$
b Basis ($b \in N \setminus \{1\}$)
y Mantisse
z Exponent (zur Basis b)

Dabei ist die Mantisse eine "Festpunktzahl"

$$y = \sum_{i=m}^{n} y_i \cdot b^{-i} \quad \text{mit } y \neq 0$$ (entspricht Normalisierung)

Bem.: Kann auch $n \leq m$ sein, auch $n \leq m \leq 0$

Also y ist eindeutig durch $y_m, y_{m+1}, \ldots y_n$ bestimmt (da b, n, m fest).
d.h. Mantisse besteht aus n - m + 1 signifikanten Ziffern.

Exponent $z \in Z$ wird meist auch in der Basis b abgespeichert (auch nur endlich viele Stellen)!

Bsp.: Taschenrechner:
Extern wird Mantisse 7-Stellig mit m = 0 dargestellt.
d.h. Angezeigte Mantissen (für $x \neq 0$) im Bereich
$1 \leq y \leq 9.999.999$
Exponent z: $-00 \leq z \leq 99$
z.b.: -2,123456e-14

Intern arbeitet der Taschenrechner aber mit längerer Mantisse (z.B. 10).
Man sieht: Es lassen sich ziemlich große/kleine Zahlen darstellen (aber nur endlich viele).

Große Herausforderung an Ingenieure:
- Wie sollen Elementare Rechnungen (+, -, *, /) realisiert werden?
- Wie soll 0 dargestellt werden?

Im wesentlichen gilt:
Das Ergebnis aller Gleitpunktoperationen ist gerundet!

Etwas Formaler:
$M := M_C :=$ Zahlen $\subset R$, welche auf Computer C darstellbar sind.
$x, y \in M$, dann gilt im Allgemeinen nicht das $x +,-,*,/ y \in M$.

also: (M, +) nicht abgeschlossen.

Sei $(x * y)^*$ die Näherung von x * y im M, dann gute Ingenieur-Leistung wenn:
$|(x * y)^* - x * y|$ so klein wie möglich.

Codierung von y, z (Mantisse, Exponent) geschieht heute nach einem seit 1984 existierendem Standard (IEEE 754) für handelsübliche Prozessoren (Pentium, Athlon, ...)

Typisch: Basis z = 2 (Dual)

Zahlentypen:

			VZ	VZ + Exp	Mantisse	
4 Byte	float	short real	1	1 + 7	23	bit
8 Byte	double	long real	1	1 + 10	52	bit
10 Byte	temp-real	(extended)	1	1 + 14	64	bit

Untersuchung zu float:

4 Byte sind aufgeteilt in:

Exponent: 1 Byte

Mantisse: 3 Byte

$$x \pm y \cdot b^z = \pm y \cdot 2^z$$

also $$y = \sum_{i=0}^{22} y_i \cdot 2^i \quad y_i \in \{0,1\}$$

$$z = \sum_{i=0}^{6} z_i \cdot 2^i \quad z_i \in \{0,1\}$$

Interpretation Dezimalsystem:

zur Mantisse: Die größte darstellbare Zahl ist: $2^{23} - 1 = 8.388.607$

$\Rightarrow$ ca. 7 signifikante Ziffern bei float.

zum Exponenten: Größter Exponent zur Basis 2: $z = 2^7 - 1 = 127$

Welchem Exponent entspricht das im Dezimal-System?

$$10^t = 2^{127}; \quad \ln(10^t) = \ln(2^{127}) \Rightarrow t \cdot \ln(10) = 127 \cdot \ln(2)$$

$$\Rightarrow t = 127 \cdot \frac{\ln(2)}{\ln(10)} \approx 38{,}23$$

größte darstellbare Zahl (float) $\approx 10^{38}$

(Nach Doku.: 0x7f7fffff entspricht Float.maxValue (JAVA))

1.2 Probleme aus der Endlichkeit von Computern

(M, +, *) ist ein Körper.

1. Bsp.: $1 \in M$, $2 \in M$, $1/3 \in M$ (1/3 = 0.3333...)

2. Bsp.: Suche Experiment um zu zeigen:

$(a + b) + c \neq a + (b + c)$

$a = 1$; $b = 4 * 10^{-12}$; $c = 4 * 10^{-12}$

3. Bsp.: Im Körper gilt:

$1 + x = 1$ hat eindeutige Lösung ($x = 0$)

$1 + 0 = 1$

$1 + 10^{-1} = 1$ (Widerspruch!)

4. Bsp.:
$f(x) := x^3 - 3$ stetig auf beliebigem Intervall [a, b]
Wenn $f(a) < 0$ und $f() > 0 \Rightarrow \exists\, x_0 \in [a, b]$ mit $f(x_0) = 0$

aber: f(1.44224957031) = 0.0...02
f(1.44224957030) = -0.0...5
$\Rightarrow$ offensichtlich keine Nullstelle in ***M***!

Bem.:
Auch noch größere Mantissen- / Exponenten-Längen lösen dieses Problem nicht. Selbst winzige Ungenauigkeiten schaufeln sich in den vielen Mio. numerischen Operationen zu unbrauchbaren Ergebnissen auf.

Die dabei auftretenden Probleme lassen sich unterteilen in:

- Auslöschung
- Smearing (Verwischen)
- numerische Instabilität
- schlecht konditioniertes Problem

1.2.1 Auslöschung

Tritt bei Subtraktion von fast gleichen Zahlen auf.

also: a, b Gleitpunktzahlen mit Exponent z.

Wenn a - b berechnet wird, werden die beiden Mantissen subtrahiert.

Dabei entstehen führende Nullen. Da aber normalisiert sein muss (1. Ziffer ≠ 0), werden die Zahlen nach links geschoben und hinten mit Nullen aufgefüllt. Diese Nullen haben eigentlich keine Bedeutung - sind eigentlich falsch.

Bsp:
$$\begin{array}{r} 1.2345 * 10^0 \\ \underline{- 1.2344 * 10^0} \\ 1.0000 * 10^{-4} \end{array}$$

Die Nullen nach dem Dezimalpunkt sind nur dann richtig, wenn die beiden Operanden richtig wahren.

Behebung: Benutze: $a - b = \dfrac{a^2 - b^2}{a + b}$
Dies kann manchmal helfen, muss aber nicht.

Dann formuliere den Ausdruck um:

Bsp.:

1. $\sin\left(\frac{\alpha}{2}\right)=\sqrt{\frac{1-\cos(\alpha)}{2}}=\sqrt{\frac{1-\sqrt{1-\sin^2(\alpha)}}{2}} \quad 0\le\alpha\le 2\pi$

2. $\sin\left(\frac{2\pi}{6}\right)=\frac{1}{2}\sqrt{3}$

3. $s_n := \sin\left(\frac{2\pi}{n}\right); \quad n\in \boldsymbol{N}$

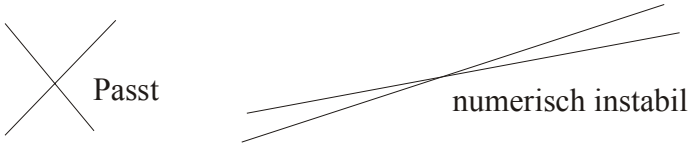
$$\begin{cases} s_6=\frac{1}{2}\sqrt{3} \\ s_{2^n\cdot 6}=\sqrt{\frac{1-\sqrt{1-s_{2^n\cdot 6}^2}}{2}} & n=1,2,3\cdots \end{cases}$$

Rekursikonsformel zur Berechnung von s_6, s_{12}, s_{24}, ... wobei nur +, *, √ benutzt wird.

1.2.2 Smearing (Verwischen)

Falls Zwischenergebnis mit kleinem Fehler behaftet, kann sich das eventuell katastrophal auf das Endergebnis auswirken.

1. Bsp.:
Suche Schnitt zweier Geraden. (Dies führt zur Lösung eines Linearen Gleichungssystems)



2. Bsp.: Berechne:

$$y_n=\int_0^1\frac{x^n}{a+x}dx \quad \text{mit } a>1 \quad n=0,1,2,\cdots 10$$

$$y_n=\int_0^1\frac{x^{n-1}\left((x+a)-a\right)}{x+a}dx=\int_0^1 x^{n-1}dx-a\int_0^1\frac{x^{n-1}}{x+a}dx$$

$$=\frac{x^n}{n}\Big|_0^1-a\cdot y_{n-1}=\frac{1}{n}-a\cdot y_{n-1}$$

$$y_0=\int_0^1\frac{1}{a+x}dx=\ln(a+x)\Big|_0^1=\ln(a+1)-\ln(a)=\ln\left(\frac{a+1}{a}\right)$$

Damit Rekursikonsformel:

$$\begin{cases} y_0=\ln\left(\frac{a+1}{a}\right) \\ y_n=\frac{1}{n}-a\cdot y_{n-1} & n=1,2,\cdots \end{cases}$$

Sei $\breve{y}_n$ der Fehlerbehaftete Wert von y_n

$$\breve{y}_0 = y_0 + \varepsilon_0$$

$$\breve{y}_n = -a\breve{y} + \frac{1}{n}$$

also: $r_n = \breve{y}_n - y_n$

$$r_0 = \varepsilon_0$$

$$r_n = \breve{y}_n - y_n = -a\breve{y}_{n-1} + \frac{1}{n} + ay_{n-1} - \frac{1}{n} = a\left(-\breve{y}_{n-1} + y_{n-1}\right) = -a\left(\breve{y}_{n-1} - y_{n-1}\right) = -a \cdot r_{n-1}$$

also: $\boxed{r_n = (-a)^n \cdot \varepsilon_0}$

$\Rightarrow$ Jeder Schritt bewirkt Vergrößerung des fehleres um Faktor a.

2 Nichtlineare Gleichungen

2.1 Problem

Geg.: f: A→***R***; A ⊂ ***R***; suche $f^{-1}(0)$ (Nullstellen)

Bsp.: $f(x) = -x^2 - 5$ (Nullstellen: $\pm\sqrt{5}$)
Berühmter Satz zur Existenz von Nullstellen:
Zwischenwertsatz: f:[a, b]→ ***R*** Stetig mit f(a) * f(b) < 0
⇒ ∃ s ∈]a, b[mit f(s) = 0 (keine Aussage über Eindeutigkeit)

Dann folgt numerisches Verfahren **"Bisektionsverfahren"**:

```
x := ( a + b ) / 2;

while( a < x && x < b )
{
   if( f( x ) * f( b ) > 0 ) b = x else a = x;
x = (a + b ) / 2;
}
```

2.2 Fixpunktverfahren

Ges.: Nullstelle von f:[a, b] → ***R***

also suche s ∈ [a, b] mit f(s) = 0

Versuche diese Gleichung nach x umzuformen, so dass x = F(x)

Def.: Sei F Funktion wenn gilt: F(s) = s, so heißt s **Fixpunkt von F**.

Bsp.:

$$f(x) := x^2 \cdot c \quad c > 0$$

$$f(x) = 0 \Rightarrow x^2 \cdot c = 0 \Rightarrow \underbrace{x + x^2 - x}_{F(x)} = x$$

Tatsächlich: Nullstellen von f(x) ($\pm\sqrt{c}$) sind Fixpunkte von F(x) und umgekehrt.

oder $x = \frac{1}{2}\left(x + \frac{c}{x}\right)$
also Umformung nicht eindeutig!
⇒ Anstelle von Nullstellen können auch Fixpunkte gesucht werden!

2.3 Der Banachsche Fixpunktsatz

(Stephan Banach 1892 - 1945)

Def.: F:A $\rightarrow$ ***R***, A $\subset$ ***R*** heißt **Kontraktion**, wenn es eine Konstante $0 \le k < 1$ gibt, so dass
$|F(x) - F(y)| \le k^* |x - y| \; \forall \; x, y \in A$
(k heißt **Kontraktionskonstante**, 2 Bildpunkte liegen näher beieinander als ihre Urpunkte)

Bem.: F Kontraktion $\Rightarrow$ F stetig!

Theorem: (Banachscher Fixpunktsatz, skalare Version)
Sei F: A $\rightarrow$ A Kontraktion und A abgeschlossene Teilmenge von ***R***.

Dann gilt:

a) es gibt genau einen Fixpunkt in A
b) sei x_0 aus A beliebig, konstruiere nun Folge:
   $x_n := F(x_{n-1})$; $n \ge 1$
   Dann konvergiert diese Folge gegen den Fixpunkt s
   $\lim_{n\to\infty}(x_n) = s$ mit F(s) = s
c) es gelten die Abschätzungen

$$|x_n - s| \le \underbrace{\frac{1}{1-k}|x_{n+1} - x_n|}_{\text{a-postpriori}} \le \underbrace{\frac{k^n}{1-k}|x_1 - x_0|}_{\text{a-priori}}$$

Algorithmus:
f(x) = 0: Forme auf Fixpunktgleichung x = F(x) um.
Finde A mit F: A $\rightarrow$ A und prüfe ob F Kontraktion.
Dann wähle $x_0 \in$ A beliebig.

```
do
{
   x1 = x0;
   x0 = f( x1 );
} while( abs( x1 - x0 ) > epsilon );

return x0;
```

Probleme:

- wie Umformen?
- wie A finden?
- wie sieht man Kontraktion?

Satz:
F:A $\rightarrow$ A, F stetig diffbar und es existiere $k := \max\{ |F'(\xi)| \} \; \xi \in$ A mit $0 \le k < 1$
Dann ist F Kontraktion mit Kontraktionskonstante k.

Satz:
F: B $\rightarrow$ ***R*** stetig diffbar mit Fixpunkt s (F(s) = s), B offen.
Wenn |F'(s)| < 1 dann gibt es ein abgeschlossenes Intervall A $\subset$ B, so dass
F: A $\rightarrow$ A Kontraktion ist.

d.h.: Wenn Iteration nur nahe genug beim Fixpunkt beginnt, dann erfolgreich!

Satz:
F: A $\rightarrow$ ***R*** sei Kontraktion mit Kontraktionskonstante k,
also $|F(x) - F(y)| \leq k \; \forall \; x, y \in A$

Sei $s \in A$ Fixpunkt und F stetig diffbar, dann gilt $|F'(s)| \leq k$
(also F'(s) liefert eine untere Schranke für Kontraktionskonstante.)

Satz:
B offen in ***R***; F: B $\rightarrow$ ***R***
stetig diffbar mit $s \in B$

Wenn |F'(s)| < 1, dann gibt es ein abgeschlossenes Intervall A in B, so dass
$s \in A$ und F_A: A $\rightarrow$ A Kontraktion.

Bedeutung: Wenn Iteration nur nahe genug bei s beginnt, dann konvergiert sie auch!

Was bedeutet dies praktisch?
Löse f(x) = 0

1. Formuliere diese Gleichung in eine Fixpunktgleichung um
   x = F(x)
2. Finde A abgeschlossen, so dass F: A $\rightarrow$ A Kontraktion ist.
   dazu : suche max $|F'(\xi)| =: k; \xi \in A$
   Wenn k < 1, dann ist F eine Kontraktion.
   Weiterhin muss gelten: F(A) $\subset$ A

Bsp.:
$f(x) := x^3 + x - 1;$ f(0) = ?
Da f(0) = -1 und f(1) = 1
$\Rightarrow$ f hat in [0, 1] eine Nullstelle.

Weitere Versuche: 0,6 < x < 0,7 (Nullstelle in [0,6, 0.7])

Umformen in Fixpunktgleichnug:
1. Versuch: $x^3 + x - 1 = 0$
$x = 1 - x^3$ F:[0,6, 0,7] $\rightarrow$ ***R***
$x \mapsto F(x) = 1-x^3$

Nun $F'(x) = -3x^2 \Rightarrow |F'(x)| = |3x^2|$ ist monoton steigend auf [0,6, 0,7]

$\Rightarrow |F'(x)| \geq |F'(0,6)| = 1.08 \Rightarrow$ Geht nicht!

2. Versuch: $x^3 + x - 1 = 0 \Leftrightarrow x = (1 + x + x^3) / 2$
$F'(x) = (1 - 3x^2) / 2$ F' ist für $x > 0$ monoton fallend.
$|F'(x)| < |F'(0,7)| = 0,235...$ auf $[0,6, 0,7]$

Wenn nahe genug bei Nullstelle begonnen wird $\Rightarrow$ erfolgreich!

Suche geeignetes Intervall A:
$A := [0,6, 0,7]$ passt, denn $F(A) \subset A$

damit gefunden: $F:[0,6, 0,7] \rightarrow [0,6, 0,7]$
$x \mapsto 1/2 (1 + x - x^3)$
ist Kontraktion ($k = 0,235$)

$x_0 = 0,7$
$x_1 = F(x_0) = 0,6785$
$x_2 = F(x_1) = 0,68307$
$x_3 = F(x_2) = 0,68217$
…

2.4 Konvergenzbeschleunigung

$|F'(s)|$ ist untere Schranke für Kontraktionskonstante.
Klar ist, ja kleiner k, desto schnellere Konvergenz.

Def.:
Gegeben sei ein Verfahren, welches eine Folge (x_i) konstruiert, welche gegen s konvergiere. Das Verfahren heißt von **q-ter Ordnung**, wenn es eine Konstante k gibt, so dass für hinreichend große Indizes gilt:

$$|x_{i+1} - s| \leq k * |x_i - s|^q$$

(Wunsch: q groß)

Bem.:
Banch-Iteration (mindestens) lineare Ordnung.
gut währe $F'(s) = 0$.

noch besser:

Satz:
$F: A \rightarrow A$ sei q-Fach stetig diffbare Kontraktion auf A mit Fixpunkt s.

Es gelte $F'(s) = F''(s) = ... = F^{(q-1)}(s) = 0$
$x_0 \in A$; $x_{i+1} = F(x_i)$

Dann gilt: $\left|x_{i-1} - s\right| \leq \frac{1}{q!} \cdot \max_{s \in A} \left|F^{(q)}(\xi)\right| \cdot \left|x_i - s\right|^q$

d.h. Verfahren dann von q-ter Ordnung.

2.5 Das Newton-Verfahren

Ges.: Nullstelle von f(x) = 0

Natürlicher Ansatz für Fixpunktgleichung:
F(x) := x + α * f(x) mit α noch zu bestimmen.

Um superlineare Konvergenz zu erreichen, währe nötig F'(s) = 0
oder $1 + \alpha * f'(s) = 0 \Rightarrow \alpha = -\frac{1}{f'(s)}$)

d.h. gute Fixpunktgleichung währe $F(x) = x - \frac{f(x)}{f'(s)}$

Nachteil: s ist nicht bekannt!

Wähle Näherung für f'(s) in der Nähe von s : f'(x)

Also Newton-Verfahren:
f sei in Umgebung der Nullstelle von f(x) = 0 dreimal diffbar.
Wähle Startpunkt in der Nähe der Lösung, und Iteriere

$$x_{i+1} = F(x_i) = x_i - \frac{f(x_i)}{f'(x_i)}$$

Diese Folge konvergiert Quadratisch gegen s (wenn x_0 nahe s)

Nachteil: Man braucht Ableitung.

2.6 Das Seffensen-Verfahren

Wenn f(x) klein (also in der Nähe einer Nullstelle), dann:

$$\frac{f(x + f(x)) - f(x)}{f(x)} \approx f'(x)$$

Ersetzte in Newton f'(x) durch diese Näherungsfunktion:

$$F(x) = x - \frac{f(x)^2}{f(x + f(x)) - f(x)}$$

Vorteil: Keine Ableitung nötig.

2.7 Höhere Verfahren

$F_3(x) := x - \frac{f(x)}{f'(x)} - \frac{f(x)^2 \cdot f''(x)}{2 \cdot f'(x)^3}$ (Konvergenz 3. Ordnung)

$F_4(x) := F_3(x) + f(x)^3 \cdot \left(\frac{f'''(x)}{f(x)^4} - \frac{f''(x)^2}{2 f'(x)^5} \right)$ (4. Ordnung)

Höhere Ordnungen sind praktisch nicht relevant!

2.8 Sekanten-Verfahren

$$x_{i+1} = x_i - f(x_i)\frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} \quad i \geq 1$$

Wähle 2 Startpunkte x_0, x_1 und konstruiere dann die x_{i+1} konvergent mit Ordnung

$$p = \frac{(\sqrt{5}+2)}{2} \approx 1{,}618\,.$$

Dies nennt man Mehrschrittverfahren, da mehr Vorgänger benötigt werden. - im Gegensatz zum obigem Einschritverfahren.

2.9 Zusammenfassung

Lösung von Gleichung f(x) = 0 lässt sich (hoffentlich) in Fixpunktgleichung F(x) = x umformen.

Zur Umformung haben wir formale Methoden kennen gelernt:

z.B.: $F(x) := x - \frac{f(x)}{f'(x)}$

Untersuche F: A $\rightarrow$ A, ob dies eine Kontraktion ist:

- suche abgeschlossenes A $\in$ ***R*** mit F(A) $\subset$ A
- stelle fest, ob max(|F'(ξ)| =: k < 1; $\xi \in$ A

Dann sichert der Banachsche Fixpunktsatz die Eindeutigkeit eines Fixpunktes in A $\Rightarrow$ s.
Weiterhin hilft dieser Satz, dass die Folge $(x_n)_{n \in \mathbb{N}0}$ ist

$n0 \in$ A beliebig

$x_{n+1} = F(x_n)$

gegen s konvergiert (Einzelschritverfahren).
Dieser Satz gibt zusätzlich Auskunft über Güte der Näherung für s.

3 Polynome

Polynome spielen eine Rolle bei:

- Approximation von beliebigen Funktionen (Taylor)
- Näherung für Punktwolken (Interpolation)
- Lineare DGL n-ter Ordnung führen zu charakteristischen Polynomen n-ter Ordnung

3.1 Auswertung von Polynomen

Geg.: $p(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$

$x \in \boldsymbol{R}$ $a_i \in \boldsymbol{R}$ $i = 0 \ldots n$ $n = \text{grad}(p(x))$

Frage: Wert von $p(x) = ?$

Beurteilungskategorien: Anzahl der Add / Sub
Anzahl der Mul / Div

1. Algorithmus:

```
double xpot = x;
double p = a[0] + a[1] * x;

for( i = 2; i < a.length; ++i )
{
   xpot *= x;
   p += a[I] * xpot;
}
return p;
```

Anzahl (+; -): n
(*; /): 2n - 1

2. Algorithmus:

Idee: $[... ((a_n x + a_{n-1})x + a_{n-2})x + ... + a_1]x + a_0$

```
n = a.length - 1;
p = a[n];

for( I = n - 1; I >= 0; --j )
{
   p = p * x + a[I];
}
```

Anzahl (+; -): n
(*; /): n

Horner auf alle Fälle obigem Algorithmus vorzuziehen!

Frage:

- Gibt es eine theoretische Untergrenze für Berechnung von p(x)?
- Gibt es noch schnellere Algorithmen?

Hornerschema:
Geg.: Polynom $p(x) = a_n x^n + \dots a_0$

Berechne p(a); p'(a); p''(a) … $a \in \boldsymbol{R}$

Folgende Rekursion leistet diese Berechnung, wobei gilt:
$p(a) = b_n$; $p'(a) = c_{n-1}$; $p''(a) = d_{n-2}$; ...

Rekursion:

$b_0 := a_n$;	$b_i := b_{i-1} * a + a_{n-i}$;	i = 1…n
$c_0 := b_0$;	$c_i := c_{i-1} * a + b_i$;	i = 1…n-1
$d_0 := 2c_0$;	$d_i := d_{i-1} * a + 2c_i$;	i = 1…n-2
$e_0 = 3d_0$;	$e_i := e_{i-1} * a + 3d_i$;	i = 1…n-3

Bem.:
Die anfallenden Koeffizienten sind von a abhängig

Die b_i haben folgende Bedeutung:
$p(x) = (x - a) * (b_0 x^{n-1} + b_1 x^{n-2} + \dots + b_{n-2} x + b_{n-1}) + b_n$

Also: Hornerschema ist auch numerisches Verfahren, um Polynomdivision zu berechnen!

denn sei a Nullstelle von p(x), dann berechnet das Schema:

$$\frac{p(x)}{x-a} = b_0 x^{n-1} + b_1 x^{n-2} + \dots + b_{n-1}$$

Prüfungsaufgabe: Wie berechnet das Hornerschema p'(a)?
Bsp.:
$p(x) = 3x^2 - 5x + 4$
p'(2) = ?

	3	-5	4
a = 2		6	2
	3	1	6 = p(2)

und es gilt:
p(x) = (x - a) q(x) + p(a)
mit q(x) = 3x + 1

Beh.: q(a) = p'(a)

a = 2		6
	3	7 = p'(2)

tatsächlich:
p'(x) = (x - a) * q'(x) + q(x)
p'(a) = q(a)

3.2 Abschätzung zur Lage der Nullstellen

Geg.: $p(x) = a_n x^n + ... + a_0$ mit $a_i \in \boldsymbol{C}$ besitzt n Nullstellen in $\boldsymbol{C}$

Es besitzt Linearfaktorzerlegung:
$p(x) = x_0 * (x - x_1)(x - x_2)\ldots(x - x_n)$ x_i Nullstellen

falls: $a_i \in \boldsymbol{R}$, dann treten komplexe Nullstellen immer paarweise konjugiert komplex auf.

Frage: Lässt sich ein gebiet in $\boldsymbol{C}$ angeben, wo alle Nullstellen zu finden sind?

$$p(x) = a_n x^n + \cdots + a_0 = a_n x^n \left(1 + \frac{a_{n-1}}{a_n x} + \frac{a_{n-2}}{a_n x^2} + \cdots + \frac{a_0}{a_n x^n} \right)$$

Damit lassen sich Abschätzungen finden, wo Nullstellen liegen können.
(x groß genug $\Rightarrow$ keine Nullstellen mehr)

Satz:
Für die Nullstellen $\xi \in \boldsymbol{C}$ des Reellen oder komplexen Polynoms $p(x) = a_n x^n + ... + a0$ gilt:

1. $$|\xi| \le \max\left\{ \left|\frac{a_0}{a_n}\right|, 1 + \left|\frac{a_1}{a_n}\right|, 1 + \left|\frac{a_2}{a_n}\right|, \cdots 1 + \left|\frac{a_{n-1}}{a_n}\right| \right\}$$
2. $$|\xi| \le \max\left\{ \left|\frac{a_0}{a_n}\right|, 2 \cdot \left|\frac{a_1}{a_n}\right|, 2 \cdot \left|\frac{a_2}{a_n}\right|, \cdots 2 \cdot \left|\frac{a_{n-1}}{a_n}\right| \right\}$$
3. $$|\xi| \le 2 \cdot \max\left\{ \sqrt[n]{\left|\frac{a_0}{a_n}\right|}, \sqrt[n-1]{\left|\frac{a_1}{a_n}\right|}, \cdots \sqrt[2]{\left|\frac{a_{n-2}}{a_n}\right|}, \left|\frac{a_{n-1}}{a_n}\right| \right\}$$

Nullstellen-Suche:

Satz: (Konvergenz des Newtonverfahrens für Polynome)
$p(x)=a_n x^n + ... + a_0$ mit $a_i \in \boldsymbol{R}$; $n \ge 2$

Es seien: $\alpha_1 \ge \alpha_2 \ge \alpha_3 \ge \ldots \alpha_k$ die reellen Nullstellen und
$\xi_1, \xi'_1, ... \xi_L, \xi'_L$ die Komplexen Nullstellen
(α_1 ist die größte Nullstelle)

wenn gilt: $\alpha_1 \ge \max\{ \mathrm{Re}(\xi_1), ... \mathrm{Re}(\xi_L) \}$,
dann konvergiert die Newtoniteration

$$x_{i+1} = x_i - \frac{p(x_i)}{p'(x_i)}$$

für jeden Startwert $x_0 > \alpha_1$ streng monoton fallend gegen die größte reelle Nullstelle α_1.

Damit: Wenn größte reelle Nullstelle größer als Realanteile der komplexen Nullstellen dann lässt sich diese mit Newton berechnen. Startwert genügend groß (siehe Satz oben).

Bem.:
Analog für kleinste reelle Nullstelle.

Wenn nach diesem Verfahren eine Reelle Nullstelle gefunden ist, so kann man den Linearfaktor abspalten, und auf dem verbleibenden Polynom wieder den Algorithmus anwenden.
Wenn von Polynom p(x) bekannt, dass nur reelle Nullstellen vorhanden sind, dann ist dies ein effizientes Verfahren.

Das abspalten von Linearfaktoren (Deflation) nach Horner ist ineffizient und numerisch instabil.
Viel besser:

Es soll allgemein das Faktorpolynom
$\omega(x) = (x - \alpha_1)(x - \alpha_2)...(x-\alpha_j)$
vom Polynom p(x) abgespalten werden.

Wie sieht die Newton-Iteration für das Restpolynom aus?
$p(x) = \omega(x) \cdot \breve{p}(x)$

Wie lautet Newton-Iteration für $\breve{p}(x)$?
$$\breve{p}(x) = \frac{p(x)}{\omega(x)} = \frac{p(x)}{(x-\alpha_1)\cdots(x-\alpha_j)}$$

Newton-Iteration für $\breve{p}(x)$:
$$F(x) = x - \frac{\breve{p}(x)}{\breve{p}'(x)} = x - \frac{p(x)}{p'(x) - p(x)\cdot\frac{\omega'(x)}{\omega(x)}} = x - \frac{p(x)}{p'(x) - p(x)\cdot\sum_{k=1}^{j}\frac{1}{x-\alpha_k}}$$

3.3 Zusammenfassung

Algorithmus: Newton-Meahly
gesucht: Nullstellen des Reellen Polynoms p(x):
Wähle Startpunkt x_0 größer als die Nullstellen von p(x)

a) die Newtoniteration
$x \mapsto x - \frac{p(x)}{p'(x)}$ mit Startwert x_0
konvergiert monoton fallend gegen die größte Nullstelle α_1, falls diese größer als die Realteile der Komplexen Nullstellen ist.

b) Wenn $\alpha_1 \geq \alpha_2 \geq \alpha_3 \geq ... \alpha_j$ schon gefunden sind, so konvergiert die modifizierte Newton-Iteration:
$$x_{i+1} = F(x_i) \text{ mit } F(x) = x - \frac{p(x)}{p'(x) - p(x)\sum_{k=1}^{j}\frac{1}{x-\alpha_k}}$$
für alle Startwerte $x_0 \geq \alpha_j$ monoton fallend gegen die nächst kleinere reelle Nullstelle α_{i+1} falls diese immer noch größer als die Realanteile der komplexen Nullstellen ist.

4 Graphentheorie

Def.: M sei Menge. $[M]^k$ sei die Menge aller k-Elementigen Teilmengen von M.

Bsp.: M = {1, 2, 3} $[M]^2$ = { {1, 2}, {1, 3}, {2, 3} }

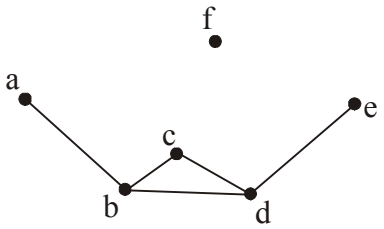
Erinnerung: $\left|[M]^k\right| = \binom{|M|}{k}$

4.1 Graphen

Def.:
Ein **Graph** ist ein Paar G = (V, E) mit $E \subset [V]^2$
d.h. E sind 2-Elementige Teilmengen einer Menge V.
V nennt man **Ecken** (vertice, Knoten)
E nennt man **Kanten** (edge)

Wir werden hauptsächlich endliche Graphen behandeln, d.h. V ist endliche Menge.

1. Bsp.:



V = {a, b, c, d, e, f}

E = { {a, b}, {b, c}, {b, d}, {c, d}, {d, e} }

2. Bsp.:
S = {1, … 10} R sie Relation auf S
x R y :⇒ x ≠ y und x teilt y
dann V = {1, … 10} E = { {x, y} | x R y}

4.2 Digraphen (gerichtete Graphen)

Def.:
Ein **Digraph** G=(V, E) mit V endlich und $E \subset V \times V$ ($V \times V$ geordnet!)

Darstellung: $v_1 \bullet\!\!\longrightarrow\!\!\bullet v_2$ wenn $\{v_1, v_2\} \in E$

(ein Digraph wird auch **gerichteter Graph** genannt!)

Bem.: Es sei für Digraph ausgeschlossen, dass {a, a} ∈ E

Schreibweisen:
G = (V, E)
(v, w) ∈ E ⇒ $\overrightarrow{vw}$:=(v, w) bei Digraph
$\overline{vw}$:= {v, w} = {w, v} = $\overline{wv}$ bei Graph

Fragen:
- Kürzeste Flugzeit München - Moskau?
- Billigster Flug München - Moskau?
- Wenn ein Flughafen wegen schlechten Wetter geschlossen, gibt es einen anderen Weg?
- Wie viel Verkehr kann zwischen 2 Ecken fliegen?
- Ist eine gegebene Relation transitiv (a R b, b R c, a R c) ?
- Hat ein Flowchart Loops?
- Wie kann man Schaltung mit wenig Draht verdrahten?

Def.:
- Graph G' = (V', E') heißt **Subgraph** von G = (V, E) wenn $V' \subset V$ und $E' \subset E$ (analog Digraph)
- a, b ∈ V heißen **benachbart**, wenn $\{a, b\} \in E$
- Ein **Pfad** (**Weg**) ist von v nach w in einem Graph (Digraph) ist eine Sequenz $\{v_0, v_1\}, \{v_1, v_2\}, \ldots \{v_{i-1}, v_i\}, \ldots \{v_{k-1}, v_k\}$ mit $v_0 = v$ und $v_k = w$ und $v_0, \ldots v_k$ verschieden. Länge des Pfades = k.
- Ein **gewichteter Graph** ist ein Tripel (V, E, W), wobei (V, E) ein Graph ist und $W : E \rightarrow \mathbb{N}_0$ ist. W(e) heißt Gewicht von Kante e (e ∈ E)
  Darstellung: $v_1 \xrightarrow{W(e)} v_2$ (Weglänge, Kosten, ...)

4.3 Computerdarstellung von Graphen

Bisherige Vorstellung von Graphen:
1. "Bild" mit Ecken und Kanten ()
2. Ecken und Kanten als Mengen "hingeschrieben"

Nun am Compi:
Sei |V| = n (n Ecken)
|E| = m (m Kanten)

$V = \{v_1, \ldots v_n\}$

1.Darstellung:
Graph G = (V, E) kann als n × n Matrix $A = (a_{ij})$ (i = 1...n, j = 1...n)

$$\text{mit } a_{ij} = \begin{cases} 1 & wenn\ \overline{v_i v_j} \in E \\ 0 & sonst \end{cases} \text{ (Nachbarschaftsmatrix)}$$

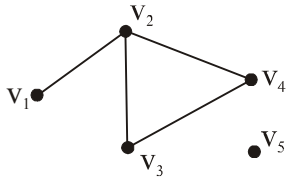
G Gewichteter Graph G(V, E, W)

$$a_{ij} = \begin{cases} W\left(\overline{v_i v_j}\right) & wenn\ \overline{v_i v_j} \in E \\ C & sonst \end{cases} \text{ wobei } C \notin W$$

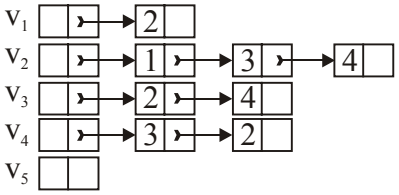
Bem.:
A ist symmetrisch bei Graph, nicht symmetrisch bei Digraph!

2. Darstellung: (linked List)
Zu jeder Ecke v ∈ V gibt es eine linked List, welche die Nachbarn von v beschreibt.
also:
Lege ein Feld der Länge |V| an, wobei jeder Eintrag dieses Feldes auf eine Liste der Nachbarn von v verweist.

Bsp.:



Bei gewichteten Graphen gebe zusätzlich das Gewicht zu jedem Listenelement an.



4.4 Minimal Spanning Tree

Def.:

- Ein Graph G = (V, E) heißt **zusammenhängend**, wenn zu je zwei Ecken v, w ∈ V ein Pfad von v nach w existiert
- Ein Graph G = (V, E) heiß **Baum**, wenn es zu je zwei Ecken v, w ∈ V genau einen Pfad von v nach w gibt. (Baum → zusammenhängend)
- Ein **Spanning-Tree** eines Graphen G = (V, E) ist ein Subgraph von G welcher ein Baum ist und alle Ecken V von G enthält.
- Wenn G gewichtet ist, und T ein spanning-Tree von G ist, so heißt die Summe der Gewichte der Kanten von T **Gewicht von T**.
- Ein **Minimal-Spannig-Tree** ist ein spanning-Tree mit minimalem Gewicht.

Wie findet man einen Minimal-Spanning-Tree?

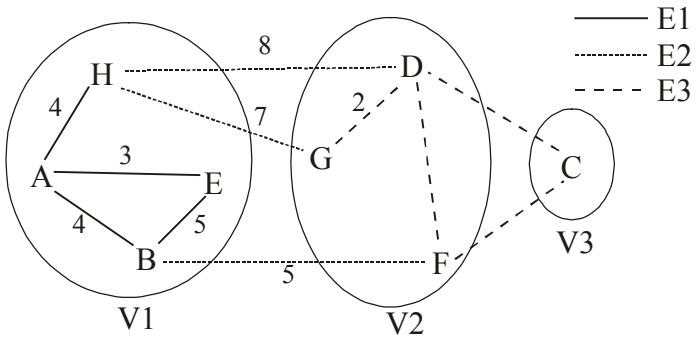
4.5 Minimal Spanning Tree Algorithmus

Während des folgenden Alorithmus lassen sich die Kanten um Ecken in 3 Gruppen einteilen:
V1: Ecken welche zu einem Subgraph ein Minimal-Spanning-Tree gehören.
E1: Kanten, welche einem Minimal-Spanning-Tree der in V1 liegenden Ecken bilden.
V2: Ecken, welche nicht in V1 liegen, aber zu einer Ecke in V1 benachbart sind.
E2: Zu jeder Ecke in V2 wähle diejenige Kante, welche minimale Verbindung zu einer Ecke in V1 ist, und füge diese Kante zu E2 hinzu.
V3 Restliche Ecken.
E3 Restliche Kanten.

Ziel: Vergrößere V1 und E1 so lange, bis alle Ecken des Graphen in V1 liegen.

Frage: Wie vergrößert man V1 und E1?

Klar ist, dass eine neue Ecke aus V2 kommen muss.



Idee: Wähle aus E2 die Kante, mit minimalem Gewicht.

Frage: Ist der so vergrößerte Baum auch weiterhin teil eines Minimal-Spanning-Tree ?

Theorem: G = (V, E, W) gewichteter Graph.
und E1 $\subset$ E Teilmenge eines Minimal-Spanning-Tree .
V1 seien die zu diesem Minimal-Spanning-Tree gehörenden Ecken.

Wenn $\overline{xy}$ Kante mit minimalem Gewicht ist, wobei x $\in$ V1 und y $\notin$ V1, denn ist E1 $\cup$ $\{\overline{xy}\}$ Teilmenge eines Minimal-Spanning-Tree.

Algorithmus:
Eingabe: G = (V, E, W) gewichteter Graph
Ausgabe: E1 $\rightarrow$ Kanten eines Minimal-Spanning-Tree

```
x = beliebige Ecke;
V1 = {x}; V2 = {}; V3 = V1 \ {x};
E1 = E2 = {};

while ( V1 != V )
{
   for( xy mit y ∉ V1 ( y zu x benachbart ) )
   {
      if( W( xy ) < W( Kante e in E2 mit y benachbart ) )
      {
         E2 = ( E2 \ { e } ) ∪ { xy }
      }
   }

   if( E2 == {} )
   {
      break; // no Spanning-Tree
   }

   Suchen nun in E2 Kante e mit minimalem Gewicht;
   x = Ecke von e welche in V2 liegt;
   E1 = E1 ∪ { e };
   E2 = E2 \ { e };
   V1 = V1 ∪ { x };
   V2 = V2 \ { x };
}
```

5 Gaus'scher Algorithmus

5.1 Prinzip (Gaus'sches Eliminisationsverfahren)

Gleichungssystem der Form:

$A_x = a$ mit

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \quad a = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}$$

habe eine eindeutig bestimmte Lösung (z.B. det(A) ≠ 0, rg(A) = n)

Führe dieses System über in "Zeilen-Stufen-Form":

$B_x = b$

$$B = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ 0 & b_{22} & & \vdots \\ 0 & \cdots & 0 & b_{nn} \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

Dann lassen sich beginnend mit der letzten Zeile die x_i rekursiv berechnen.

5.2 Konstruktion des Verfahrens

Geg.: (A/a) mit det(A) ≠ 0

(*) Vertausche die Zeilen so, dass die Zeile mit dem betragsmäßig größten Element a_{x1} in der 1. Zeile steht. Die so entstehende Matrix lautet:

$(A^0/a^0) = (a^0_{ij}/a_i)$

Eliminiere nun aus den Zeilen 2 bis n die x_1-Komponenten. Dazu multipliziere die 1. Gleichung mit $-\frac{a^0_{i1}}{a^0_{11}}$ und addiere sie zur i-ten Zeile.

Wiederhole dies (*) mit einer um die 1. Zeile und Spalte reduzierten Matrix.

Bem.:
Vertauschung der Zeilen erfolgt wegen unnötigem Anwachsen von Rundungsfehlern. (Großer Nenner ⇒ kleine Zahlen)
⇒ Pivotsuche (beliebig komplizierte Verfahren sind hier möglich!)

6 Approximation von Funktionen

In der Praxis sollen Funktionen durch einfachere Funktionen ersetzt werden, oder vorhandene Punkte sollen durch Funktionen angenähert werden.

Bsp.: sin(x)
Nur für einige Werte kann sin(x) genau berechnet werden. (0, π, ...)

Man kann zeigen: sin(x) $\notin \boldsymbol{Q}\ \forall$ x ≠ 0, x ∈ $\boldsymbol{Q}$

Berechnung nach $\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \cdots$ kann nicht durchgeführt werden, da unendlich viele Summanden. Also sin(x) approximieren.

Wunsch: Funktion durch "einfachere" ersetzen in der nur +, -, *, / vorkommen.
Zum Beispiel durch Polynome.

Bem.: Es gibt eine Unmenge von Möglichkeiten Funktionen zu approximieren.

6.1 Zwei klassische Approximations-Theoreme

Theorem 1: (Weiastrass Approximations-Theorem)
f : [a, b] → $\boldsymbol{R}$ stetig ε > 0 beliebig vorgegeben.

Dann gibt es ein Polynom p(x), so dass
|f(x) - p(x)| < ε ∀ x ∈ [a, b]

Bem.:
Es gibt mehrere Beweise dafür (auch konstruktiv). Aber der Grad dieser Polynome ist für die Praxis meist zu hoch!

Bei mehreren Vorraussetzungen (z.B. diffbar)
⇒ bessere Polynome für Praxis

Theorem 2: (Erinnerung Taylor)
f : [a, b] → $\boldsymbol{R}$ (a, b evt. auch ± ∞)
f sei n+1 mal stetig diffbar
$x_0 \in$ [a, b]

$$p(x) := f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n$$

Dann gibt es ein ξ zwischen x und x_0, so dass

$$f(x) = p(x) + \underbrace{\frac{f^{(n+1)}(x_0)}{(n+1)!}(x - x_0)^{n+1}}_{Fehler} \qquad p(x) = \text{Taylerpolynom (n – jet)}$$

Bem.: Typisch für $j^n_{x0}(f)$ (n-jet von f an der Stelle x_0):
f und $j^n_{x0}(f)$ haben bei x_0 die gleiche Ableitung bis zur Ordnung n.

Bsp.:
Approximiere f(x) = sin(x) auf Intervall $[0, \pi/2]$ mit relativen Fehler $< 10^{-14}$
f(0) = 0 exakt!

Wähle Taylorpolynom bei $x_0 = 0$

$$\Rightarrow p(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots + (-1)^{n-1} \frac{x^{2n-1}}{(2n-1)!}$$

Der Fehler der dabei entsteht lautet:

$$\left|\frac{\sin(x) - p(x)}{\sin(x)}\right| < 10^{-14} \quad \text{oder} \quad \left|\frac{x^{2n} \cdot \sin(\xi)}{2n \cdot \sin(x)}\right| \le 10^{-14}$$

Da $0 < \xi < x$ und sin(x) monoton steigend im Intervall $[0, \pi/2]$ ist dies erfüllt wenn:

$$\left|\frac{x^{2n}}{(2n)!}\right| < 10^{-14} \ \forall\, x \in \left[0, \frac{\pi}{2}\right] \quad \text{d.h.} \ \frac{\left(\frac{\pi}{2}\right)^{2n}}{(2n)!} < 10^{-14}$$

durch versuche finde: ab n = 10 ist dies der Fall!

$\Rightarrow$ Die gewünschte Approximation mir relativem Fehler $< 10^{-14}$ ist erreicht durch Taylerpolynom vom Grad 2n - 1 = 19.

Nachteile:
- Im allgemeinen ist Näherung in der Nähe von x0 besser als weiter weg.
  $\Rightarrow$ nicht einheitlich
- Portable Software sollte beliebige Fehlertoleranz vorgeben:
  $\Rightarrow$ Bei Taylor: $n \to \infty$

6.2 Interpolations-Polynome

Taylor hat mit Funktion die Ableitungen in einem Punkt gemeinsam.

Ein Interpolations-Polyom hat mit Funktion n+1 Punkte gemeinsam.

$f : I \to \boldsymbol{R}$ Funktion auf Intervall I
$x_0 < x_1 < x_2 < \ldots < x_n$ und $f_k := f(x_k)$ (n+1 Punkte)

Problem:
Finde Polynom p(x) (Grad(p) ≤ n), welches f an den n+1 vorgegebenen Punkten (x_k, f_k) interpoliert.
d.h.: $p(x_k) = f(x_k)$ k = 0, 1, ... n

Theorem:
Es gibt genau ein Polynom p(x) mit Grad(p) ≤ n welches f an den n+1 Punkten $x_0, \ldots x_n$ interpoliert.

Def.:
m = 0, 1, ... n (n+1 Punkte)

$L_m(x) := \prod_{\substack{k=0 \\ k \neq m}}^{n} \frac{x - x_k}{x_m - x_k}$ "Lagrange-Interpolations-Koeffizienten"

hat Eigenschaft $L_m(x) = \begin{cases} 1 & k = m \\ 0 & k \neq m \end{cases}$

nun: $\boxed{p(x) := \sum_{m=0}^{n} f_m \cdot L_m(x)}$

dann gilt: Grad(p) ≤ n und $p(x_k) = f_k = f(x_k)$

Also: Polynom ist mit obiger Formel "berechenbar"

Ordnung dieser Auswertung: $O(n^2)$
Umformung möglich (Aitken-Neville), Barycentrische Darstellung ⇒ O(n)

Heute wird Polynominterpolation seltener benutzt, da bei größeren n Oszilationen auftreten.

6.3 Spline-Interpolation

Bis ca. 1950 wurden Funktionen durch Polynome stückchenweise (niedrige Ordnung) approximiert (z.B. linear).

Technik des Splines benötigt zwar mehr Rechenaufwand, hat aber enorme Vorteile:

- Numerisch stabil
- Ausgezeichnet konditioniert
- Sehr verbreitet in den letzten Jahren
- Verschiedene Zugänge möglich

Geg.: $x_0 < x_1 < x_2 < ... < x_n$
$f_0, f_1, f_2, \ldots f_n$ } n+1 Interpolationspunkte
$f_0 = f(x_0), \ldots f_n = f(x_n)$

Finde Funktion g : $[x_0, x_n] \rightarrow R$ welche f approximiert.

Def.:
$x_0 < ... < x_n$ n+1 Punkte (C^x := x-mal stetig diffbar)
Raum der (polynomialen) Splinefunktionen vom Grad k
$S^k_{[x_0,x_n]} := \left\{ S \in C^{k-1}([x_0, x_n]) \,\middle|\, S|_{[x_i, x_{i+1}]} \text{ ist Polynom Grad} \leq k \right\}$

Bem.:

1. Auf Teilintervalle durch Polynome darstellbar, und an den Stützstellen (k-1)-mal stetig diffbar. ("Glatheitsbedingung", nicht so glatt wie Polynom über $[x_0, x_n]$!)
2. $S^k_{[x_0,x_n]}$ ist Vektorraum über $\boldsymbol{R}$
3. $S^k_{[x_0,x_n]}$ ist endlich dimensionaler Vektorraum

   $$\dim\left(S^n_{[x_0,x_n]}\right) \le \dim\left(\underbrace{R^{n+1}\times\cdots\times R^{n+1}}_{n-mal}\right)$$

   $$\Rightarrow \dim\left(S^n_{[x_0,x_n]}\right) \le n\cdot(n+1)$$
4. Und es gilt dim($S^k_{[x_0,x_n]}$) = n + k
5. Zur Lösung von Interpolationsaufgabe mit Splinefunktionen suche zu vorgegebenen Punkten:
   $(x_0, f_0), (x_1, f_1), ...(x_n, f_n)$ mit $x_0 < x_1 < ... < x_n$ eine Funktion
   $S \in S^k_{[x_0,x_n]}$ mit $S(x_i) = f_i$ (i = 1, ... n)
   (d.h. S soll auf $[x_0, x_n]$ k-1 stetig diffbar sein und auf den Teilintervallen ein Polynom k-ter Ordnung sein.)

Nun: Die Bedingung $S(x_i) = f$; i = 0, ... n liefert n+1 Gleichungen. d.h. es fehlen zur eindeutigen Bestimmung von S noch (n+k) - (n+1) = k-1 Gleichungen. Diese fehlenden Bedingungen sind meistens durch k-1 Bedingungen an den Rändern bei x_0 und x_n anzugeben.

Damit vorläufiger Algorithmus zur Bestimmung von Splinepunkten:

1. Lege k fest (meist k = 3, kubische Splines)
2. Einlesen der (x_i, f_i) i = 0, ... n
3. Angabe von k-1 Bedingungen (Randbedingungen)
4. Löse zur Bestimmung der k+1 Koeffizienten der Polynome auf den n Teilintervallen ein großes Gleichungssystem.

6.4 Kubische Splines (k=3)

(Für Praxis wichtigster Fall!)

$$S^3_{[x_0,x_n]} = \left\{ S \in C^2[x_0,x_n] \,\middle|\, S \text{ ist auf den Teilintervallen jeweils ein Polynom vom Grad 3} \right\}$$

Wenn die n+1 Bedingungen $S(x_i)=f_i$ i = 0, ... n vorgegeben, dann fehlen noch 3 - 1 0 2 Bedingungen zur eindeutigen Bestimmung des gesamten S.

Dazu bieten sich in der Praxis folgende 4 Möglichkeiten an:

1. Natürliche Randbedingung: $S''(x_0) = 0$; $S''(x_n) = 0$
2. Vollständige Randbedingung: $S'(x_0) = f'_0$; $S'(x_n) = f'_n$
3. Periodische Randbedingung: $S(x_n) = S(x_0)$; $S'(x_n) = S'(x_0)$; $S''(x_n) = S''(x_0)$
4. not-a-knot Randbedingung: An den stellen x_1 und x_{n-1} wird Stetigkeit von S''' gefordert. Dies läuft darauf hinaus, dass S auf den Doppelintervallen $[x_0, x_2]$ und $[x_{n-2}, x_n]$ ein Polynom 3. Ordnung ist, so dass x_1, x_{n-1} eigentlich nicht mehr zu den Stützstellen gehören (not-a-knot).

6.5 Berechnung der kubischen Splines

Auf dem Intervall $x \in \{x_i, x_{i+1}\}$ $i \in \{0, ... n-1\}$ sei gesuchtes S(x):

$S(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$

$h_i = (x_{i+1} - x_i)$ (Breite des Intervalls)

Bestimme alle Koeffizienten a_i, b_i, c_i, d_i $i = 0, \ldots n-1$

1. Feststellung:

Die zu suchenden Koeffizienten sind alle berechenbar, wenn man die Werte

$f_i = S(x_i)$ und

$f_i'' = S''(x_i)$ an den Stellen $x_0 \ldots x_n$ kennt.

$$a_i = f_i \qquad b_i = \frac{f_{i+1} - f_i}{h_i} - \frac{h_i}{6}\left(f''_{i+1} + 2f''_i\right)$$

denn

$$c_i = \frac{f''_i}{2} \qquad d_i = \frac{f''_{i+1} - f''_i}{6h_i}$$

Damit: Wenn man das Gleichungssystem findet, welches die n+1 Unbekannten f_i'' berechnet (f_i sind vorgegeben), dann kann man die a_i, b_i, c_i, d_i leicht berechnen.

2.Feststellung:

Es gibt folgendes Gleichungssystem:

$$h_{i-1} \cdot f''_{i-1} + 2(h_{i-1} + h_i) f''_i + h_i \cdot f''_{i+1} = \underbrace{6\frac{f_{i+1} - f_i}{h_i} - 6\frac{f_i - f_{i-1}}{h_{i-1}}}_{=\delta_i} \qquad i = 1 \cdots n-1$$

3. Feststellung:

Bei den fehlenden Gleichungen bestimme aus den Randbedingungen (4 Fälle).

6.6 Gleichungssysteme für die Unbekannten f_i'' (i = 0 ... n)

Nach Feststellung 2 haben wir bereits n-1 Gleichungen für die Unbekannten f_0'', ... f_n''.

a) Natürliche Randbedingungen: $f_0'' = 0$, $f_n'' = 0$

Damit lautet die Matrix-Schreibweise für die Unbekannten f_0'' ... f_n'':

$$\begin{pmatrix} 1 & 0 & & & \cdots & & & & 0 \\ h_0 & 2(h_0+h_1) & h_1 & 0 & \cdots & & & & 0 \\ 0 & h_1 & 2(h_1+h_2) & h_2 & 0 & \cdots & & & 0 \\ \vdots & & & \ddots & & & & & \vdots \\ 0 & & & & \cdots & 0 & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & & & & \cdots & & 0 & 1 & \end{pmatrix} \begin{pmatrix} f''_0 \\ f''_1 \\ f''_2 \\ \vdots \\ f''_{n-1} \\ f''_n \end{pmatrix} = \begin{pmatrix} 0 \\ \delta_1 \\ \delta_2 \\ \vdots \\ \delta_{n-1} \\ 0 \end{pmatrix}$$

b) Vollständige Randbedingungen:

$S'(x_0) = f_0'$ $S'(x_n) = f_n'$ f_0', f_n' beliebig

$$\begin{pmatrix} 2h_0 & h_0 & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \cdots & 0 & h_{n-1} & 2h_{n-1} \end{pmatrix} \begin{pmatrix} \quad \\ \quad \\ \quad \\ \quad \\ \quad \end{pmatrix} = \begin{pmatrix} \delta_0 := 6\dfrac{f_1 - f_0}{h_0} - 6f'_0 \\ \vdots \\ \delta_n := 6f'_n - 6\dfrac{f_n - f_{n-1}}{h_{n-1}} \end{pmatrix}$$

c) Periodische Randbedingungen:

$$\begin{pmatrix} 2(h_{n-1}+h_0) & h_0 & 0 & \cdots & 0 & h_{n-1} & 0 \\ \vdots & & & \ddots & & & \vdots \\ 1 & 0 & & \cdots & & 0 & -1 \end{pmatrix} \begin{pmatrix} \quad \\ \quad \\ \quad \\ \quad \\ \quad \\ \quad \\ \quad \end{pmatrix} = \begin{pmatrix} \delta_0 := 6\dfrac{f_1 - f_0}{h_0} - 6\dfrac{f_0 - f_{n-1}}{h_{n-1}} \\ \vdots \\ \delta_n := 0 \end{pmatrix}$$

d) "not-a-knot":

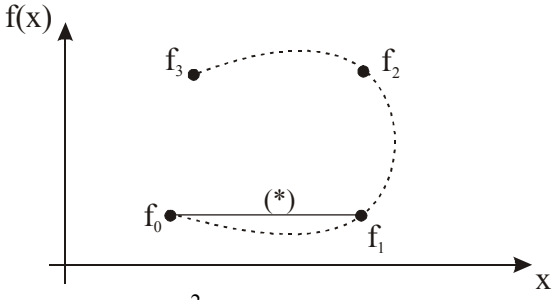
$$\begin{pmatrix} h_1 & -(h_0+h_1) & h_0 & 0 & \cdots & 0 \\ \vdots & & \ddots & & & \vdots \\ 0 & \cdots & 0 & h_{n-1} & -(h_{n-2}+h_{n-1}) & h_{n-2} \end{pmatrix} \begin{pmatrix} \quad \\ \quad \\ \quad \\ \quad \\ \quad \\ \quad \end{pmatrix} = \begin{pmatrix} \delta_0 := 0 \\ \vdots \\ \delta_n := 0 \end{pmatrix}$$

Damit Algorithmus:

1. Dateneingabe (x_i, f_i)
2. Wähle passende Randbedingungen
3. Löse Gleichungssystem mit den n+1 Unbekannten f_i'
4. Berechne alle Koeffizienten a_i, b_i, c_i, d_i für die Polynome auf den Intervallen $[x_i, x_{i+1}]$
5. Zur Berechnung von Funktionswert S(x):
   a) Suche Intervall mit $x \in [x_i, x_{i+1}]$
   b) Berechne aus den zugehörigen Koeffizienten:
   $S(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$

6.7 Spline-Interpolation von Kurven

Geg.: n+1 Vektoren (Punkte) des $\boldsymbol{R}^k$



$f_0, f_1, \ldots f_n \in \boldsymbol{R}^2$

$$f_0 = \begin{pmatrix} f_{01} \\ f_{02} \end{pmatrix}; \quad f_1 = \begin{pmatrix} f_{11} \\ f_{12} \end{pmatrix}; \quad \cdots f_n = \begin{pmatrix} f_{n1} \\ f_{n2} \end{pmatrix}$$

Problem: Suche Kurve, welche durch $f_0, \ldots f_n$ geht (interpoliert)

Kurve: $S : [a,b] \rightarrow \boldsymbol{R}^2$ mit $f_i \in S([a, b])$

Idee:
$x_0 := 0$
$x_i := x_{i-1} + \|f_i - f_{i-1}\|^{(*)} \quad i = 1 \ldots n$
$x_0 < x_1 < \ldots < x_n$

Nun löse 2 Interpolationsaufgaben:
1. $(x_0, f_{01}), (x_1, f_{11}), \ldots (x_n, f_{n1})$ 1. Komponenten von f_i
2. $(x_0, f_{02}), (x_1, f_{12}), \ldots (x_n, f_{n2})$ 2. Komponenten von f_i

Die Lösungen seien:
$S_1[x_0, x_n] \rightarrow \boldsymbol{R}$ und
$S_2[x_0, x_n] \rightarrow \boldsymbol{R}$

dann ist:
$S : [x_0, x_n] \rightarrow \boldsymbol{R}^2$

$$x \mapsto \begin{pmatrix} S_1(x) \\ S_2(x) \end{pmatrix}$$

Frage: wie sollen die Randbedingungen von S_1, S_2 gewählt werden?
Ja nach praktischer Anwendung:

z.B.: geschlossener Spline → Periodische Randbedingungen

Analog: $\boldsymbol{R}^3$, $\boldsymbol{R}^n$

7 Numerische Integration

Aufgabe: Berechne Numerisch $\int_a^b f(x)dx$

weil z.B.:

- f nur für Messwerte gegeben oder
- Stammfunktion geschlossen nicht angegeben werden kann oder
- ...

7.1 Simpson-Quadratur-Formel

Geg.: a < b f : [a, b] → ***R*** (integrierbar)

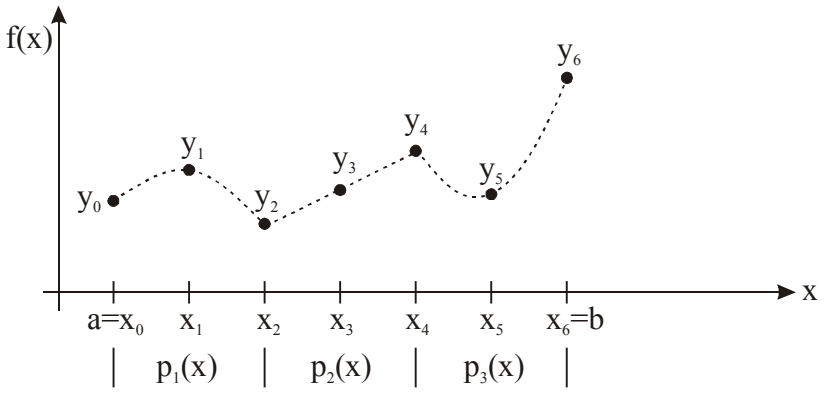
Ges.: Näherung für $\int_a^b f(x)dx$

Idee: Unterteile [a, b] in n Teilintervalle gleicher Länge

$$a = x_0 < x_1 < \cdots < x_n = b; \quad h := \frac{b-a}{h}; \quad x_i = a + h \cdot i; \quad i = 0 \cdots n$$

Wähle n gerade!
Interpoliere jeweils 3 Punkte (x_0, x_1, x_2), (x_2, x_3, x_4), ... (x_{n-2}, x_{n-1}, x_n) mit Polynom ≤ 2 und Integriere diese Parabeläste.

Summe dieser Integrale liefert Näherung für $\int_a^b f(x)dx$



Wie lautet die Gleichung für $p_1(x)$ = Polynom vom Grad ≤ 2 auf dem Teilintervall [x_0, x_2] mit Mittelpunkt x_1?

Behauptung:

$$p_1(x) = y_0 + \frac{4y_1 - y_2 - 3y_0}{2h}(x - x_0) + \frac{y_2 - 2y_1 + y_0}{2h^2}(x - x_0)^2$$

ist das gesuchte interpolierende Polynom.

Allgemein:

$$p_j(x) = y_{2j-2} + \frac{4y_{2j-1} - y_{2j} - 3y_{2j-2}}{2h}\left(x - x_{2j-2}\right) + \frac{y_{2j} - 2y_{2j-1} + y_{2j-2}}{2h^2}\left(x - x_{2j-2}\right)^2 \qquad j = 1 \cdots n$$

Nun berechne Integral von $p_j(x)$ über das Intervall $[x_{2j-2}, x_{2j}]$ mit Mittelpunkt x_{2j-1}.
Hier nur für $p_1(x)$:

$$\int_{x_0}^{x_2} p_1(x)dx = \int_{x_0}^{x_0+2h} p_1(x)dx = y_0 \cdot 2h + \frac{4y_1 - y_2 - 3y_0}{2h} \cdot \frac{(2h)^2}{2} + \frac{y_2 - 2y_1 + y_0}{2h^2} \cdot \frac{\left(2h^2\right)}{3}$$

$$= \frac{h}{3}\left(y_0 + 4y_1 + y_2\right)$$

oder allgemein:

$$\int_{x_{2j-1}}^{x_{2j}} p_j(x)dx = \frac{h}{3}\left(y_{2j-2} + 4y_{2j-1} + y_{2j}\right)$$

Damit:
Satz: Simson-Quadratur-Formel

$$\int_a^b f(x)dx \approx \frac{h}{3}[\quad f(x_0) + 4f(x_1) + f(x_2) +$$
$$f(x_2) + 4f(x_3) + f(x_4) +$$
$$\vdots$$
$$f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)]$$

$$= \frac{h}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + 2f(x_4) + \cdots + 2f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)\right]$$

Satz:
Sei von f bekannt, dass $f \in C^4([a, b])$, dann gilt für den Fehler den man bei der Simpson-Regel erhält:

$$\left|\text{Fehler}\right| = \left|R(h)\right| \le \frac{h^4}{180} \cdot \max_{x \in [a,b]} \left|f''''(x)\right| \cdot (b - a)$$

7.2 Verallgemeinerung (Newton-Codes)

1. Nehme k+1 Stützstellen
2. Lege Polynom k-ter Ordnung durch diese Stützstellen
3. Integriere diese auf den Teilintervallen
4. Summiere auf

Diese Formeln heißen **Newton-Codes**

Bsp.: k=3

$$\int_a^b f(x)dx \approx \frac{3}{8}h[f(x_0) \quad +3f(x_1)+3f(x_2)+2f(x_3)$$
$$+3f(x_4)+3f(x_5)+2f(x_6)$$
$$+ \quad \cdots \qquad\qquad +f(x_k)]$$

Fehler dabei:

$|r| \le \frac{n^4}{80} \cdot \max_{x\in[0,6]}(|f''''(x)|) \cdot (b-a)$ (Newtonsche $^3/_8$-Regel)

Ganz andere Idee: Gaus'sche Integrationsmethode:

Wunsch: $\int_0^1 f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i)$

Problem: Finde w_i und x_i (Stützstellen, so dass gute Näherung erreicht wird
$\Rightarrow$ "Legandre-Polynome"

8 Gewöhnliche Differentialgleichungen

Geg.:
Anfangswertaufgabe: Dgl 1. Ordnung y'=f(x, y)
Anfangswert: $y(x_0) = y_0$

Ges.: Lösung für Anfangswertaufgabe
d.h. suche Funktion y(x) mit:

$y'(x) = f(x, y(x))$
$y(x_0) = y_0$

Versuch / Idee:
wenn gelten soll, dass y'(x) = f(x, y(x)), dann

$$\int_{x_0}^{x_0+h} y'(x)dx = \int_{x_0}^{x_0+h} f(x, y(x))dx$$

$$y(x_0+h) - y(x_0) \Rightarrow \underbrace{y(x_0+h)}_{\text{neuerWert}} = \underbrace{y(x_0)}_{\substack{\text{vorgegebener}\\ \text{Wert}}} + \underbrace{\int_{x_0}^{x_0+h} f(x, y(x))dx}_{\substack{\text{leider nicht berechenbar}\\ \text{da } y(x) \text{ auf } [x_0, x_0+h]\\ \text{nicht bekannt}}}$$

aber: Näherung für $\int_{x_0}^{x_0+h} f(x, y(x))dx$ suchen.

Nach Simpson: (s.o.)
Für kleines h:

$$\int_{x_0}^{x_0+h} f(x, y(x))dx \approx \frac{1}{3} \cdot \frac{h}{2}\left[f(x_0, y(x_0)) + 4f(x_0 + h/2, y(x_0 + h/2)) + f(x_0 + h, y(x_0 + h))\right]$$

$$= \frac{h}{6}[\quad f(x_0, y(x_0))$$
$$+ 2 \cdot f(x_0 + h/2, y(x_0 + h/2))$$
$$+ 2 \cdot f(x_0 + h/2, y(x_0 + h/2))$$
$$+ f(x_0 + h, y(x_0 + h)) \quad]$$

leider $y(x_0 + h/2)$ und $y(x_0 + h)$ nicht bekannt.

setze für diese Näherungswerte:

$$k_1 := f(x_0, y_0)$$
$$k_2 := f(x_0 + h/2, y_0 + h/2 \cdot k_1)$$
$$k_3 := f(x_0 + h/2, y_0 + h/2 \cdot k_2)$$
$$k_4 := f(x_0 + h, y_0 + h \cdot k_3)$$
$$y_1 := y_0 + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

ist Näherung für gesuchtes $y_0(x_0 + h)$.

Satz: Ruuge-Kutta-Verfahren zur Schrittweite h
Geg.: Anfangswertaufgabe:
$y' = f(x, y); \quad y(x_0) = y_0$

Def.: Folge

$$x_k := x_0 + k \cdot h \quad k = 0,1,2,\cdots$$

$$y_{k+1} := y_k + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

mit

$$k_1 := f(x_k, y_k)$$
$$k_2 := f(x_k + {}^{h}/_{2}, y_k + {}^{h}/_{2} \cdot k_1)$$
$$k_3 := f(x_k + {}^{h}/_{2}, y_k + {}^{h}/_{2} \cdot k_2)$$
$$k_4 := f(x_k + h, y_k + h \cdot k_3)$$

dann gilt : $y(x_0 + k \cdot h) \approx y_k$

Bsp.: Löse:
$y' = -2xy^2$ mit $y(0) = 1$
$h = {}^{1}/_{10}$
$f(x, y) = -2xy^2$; $x_0 = 0$; $y_0 = 0$

Berechne Näherung für $y({}^{1}/_{10})$:

$$k_1 = 0$$
$$k_2 = f({}^{1}/_{20}, 1 + {}^{1}/_{20} \cdot 0) = -{}^{1}/_{10}$$
$$k_3 = f({}^{1}/_{20}, 1 + {}^{1}/_{20} \cdot (-{}^{1}/_{10})) = f({}^{1}/_{20}, {}^{99}/_{100}) = -0,0990025$$
$$k_4 = -0,196059503$$
$$y_1 = 1 + {}^{1}/_{60}(0 + 2 \cdot (-{}^{1}/_{10}) + 2(-0,0990025) + (-0,196059503))$$
$$y_1 = 0,99009893$$

exakte Lösung:

$$y(x) = \frac{1}{x^2 + 1}$$
$$y({}^{1}/_{10}) = \frac{1}{{}^{1}/_{100} + 1} = 0,9900990099$$

Fehler $\approx 8 \cdot 10^{-8}$

9 Index