TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Numerische Methoden und Simulation

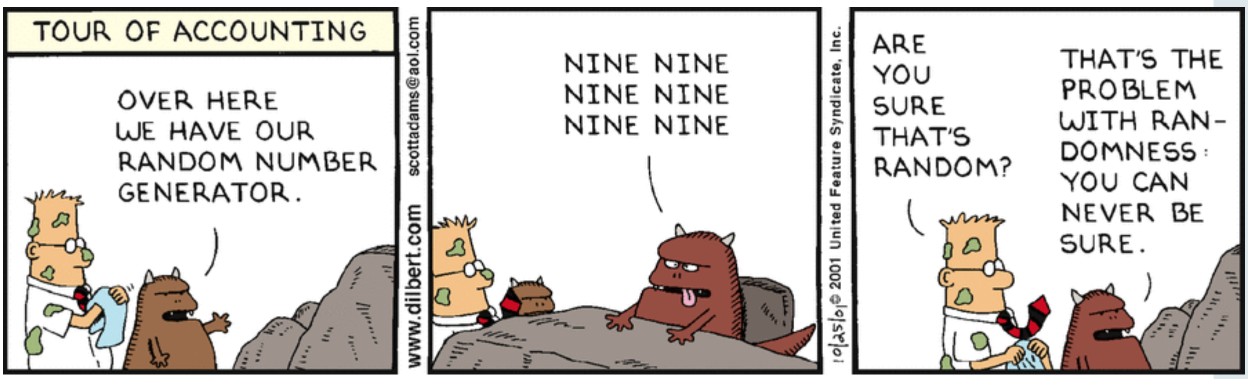
Kapitel 2: Datenanalyse und -modellierung

2.1 (Verteilung von) Zufallszahlen

2.2 Anpassungsprobleme (χ^2-fitting)

2.3 Interpolation (Polynom-, Spline-Interpolation)

2.1 Zufallszahlen



http://www.dilbert.com

2.1 Zufallszahlen

- üblicherweise nur Pseudo-Zufallszahlen
- **vertrauen Sie niemals** einem Zufallszahl-Generator (RNG), der nicht hinreichend getestet ist (NIST-STS, DIEHARD, CISC-library, …)
- **Warnung**: Umsetzung des RNG in manchen standard C- und Fortran Routinen (z.B. **`rand`** und **`srand`**) ist nicht standardisiert → unbekannte Qualität!
- historisches Beispiel einer schlechten Umsetzung: **RANDU** (auf früheren IBM-Großrechnern)

„klassische" Zufallszahl-Generatoren

lineare Kongruenz Generatoren:

$$a_{i+1} = (a_i \cdot b + c) \bmod m$$

randu:
$b = 65535 = 2^{16} - 1; c = 0; m = 2^{31} - 1$

Fibonacci Generatoren:

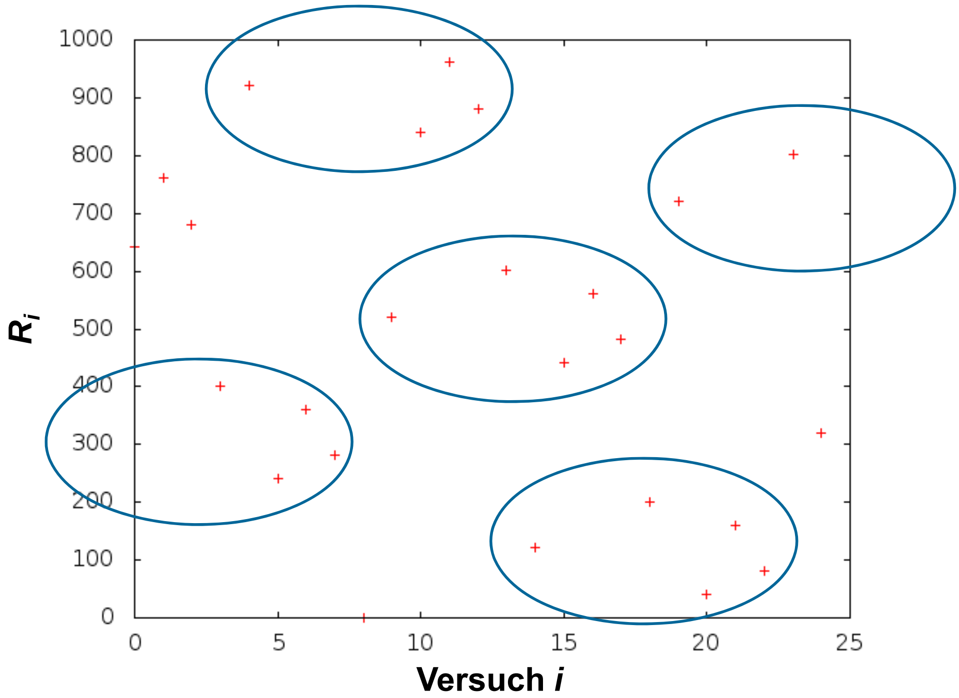
$$a_{i+1} = (a_i + a_{i-1}) \bmod m$$

von Neumann Generatoren:

$$a_{i+1} = \tilde{a}_i \cdot \tilde{a}_i$$

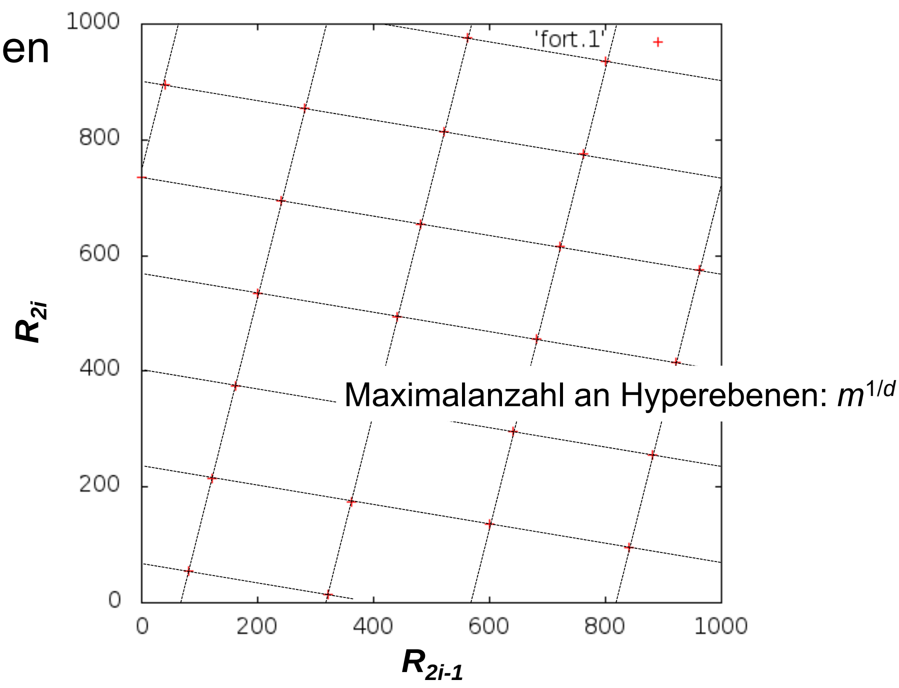
$\tilde{a}_i$ … Sequenz aus Mitte von a_i

lineare Kongruenz $R_{i+1} = (R_i \cdot 279 + 23456) \bmod 1000$



lineare Kongruenz $R_{i+1} = (R_i \cdot 279 + 23456) \bmod 1000$

Korrelationen



moderne Zufallszahl-Generatoren

pseudo-Zufallszahlen:

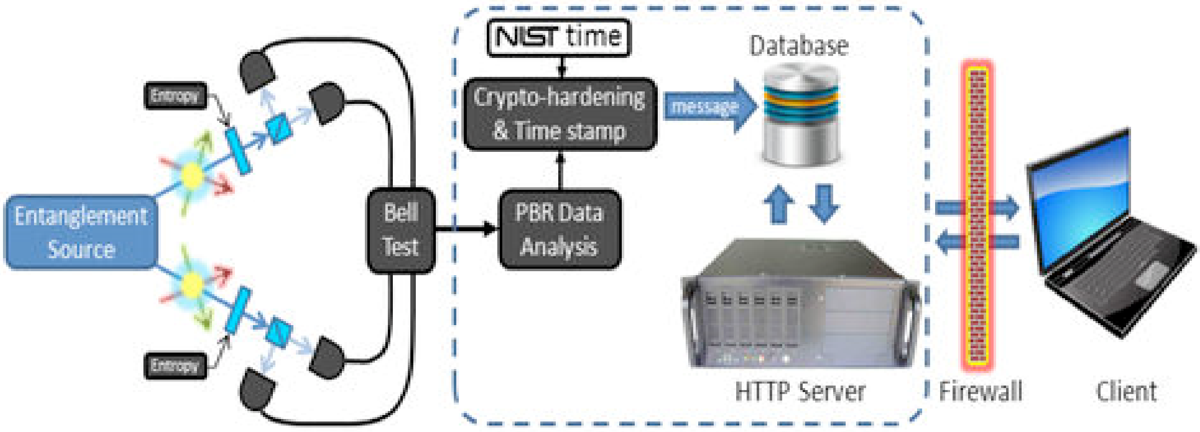
- XOR-shift Generatoren
- multiply-with-carry Generatoren
  + lineare Kongruenz

  } "KISS" Generator
- Mersenne twister (+ kleiner Bruder TT800)
- WELL (well equidistributed long-period linear) Generator

echte Zufallszahlen (aus physikalischen Prozessen):

- radioaktiver Zerfall
- thermisches Rauschen
- Radio-/Mikrophon-Rauschen
- CCD-Sensor-Rauschen

NIST randomness beacon (https://beacon.nist.gov/home)



512 bit/60 s

Natur: selten Gleichverteilungen

Rekonstruktion der Verteilungsfunktion $p(x)$ aus Messdaten:

a) berechne n-tes Moment $\langle X^n \rangle$

b) berechne $\phi_X(k) = \sum_{n=0}^{\infty} \frac{(ik)^n \langle X^n \rangle}{n!}$

c) → $\phi_X(k)$ ist Fouriertransformierte von $p(x)$

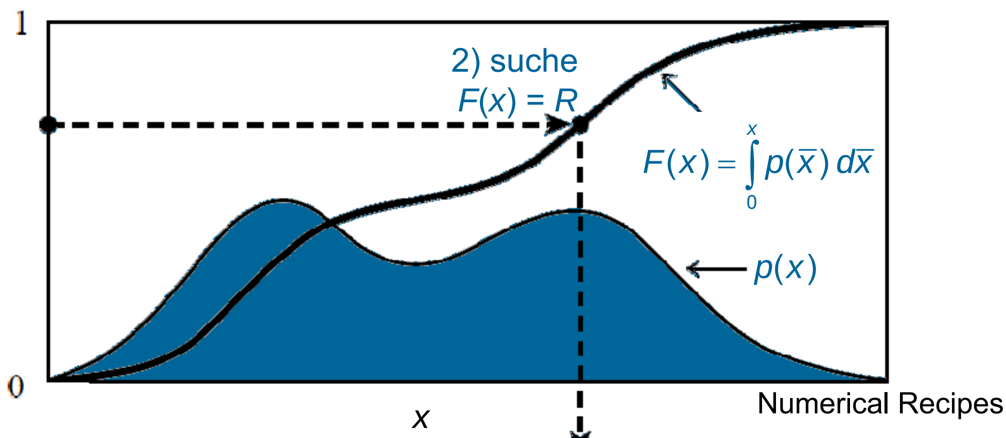
unbekannte Verteilungsfunktion $\rightarrow$ $\langle X^n \rangle = \frac{1}{N} \sum_i \left(x_i - \langle x \rangle\right)^n$

bekannte Verteilungsfunktion $\rightarrow$ $\langle X^n \rangle = \int x^n p(x)\, dx$

Erzeugung nicht-gleichverteilter Zufallszahlen

a) Transformationsmethode

1) ermittle gleichverteilte Zufallszahl $R \in [0,1)$



3) $x = F^{-1}(R)$

gilt für normalisierte Verteilungsfunktionen!

normalisierte Verteilung: $p(x)$; $\int p(x)dx = 1$

$$|p(x)dx| = |p(R)dR| \quad \rightarrow \quad p(x) = p(R)\left|\frac{dR}{dx}\right|$$

= 1 falls $p(R)$ gleichverteilt

nicht-gleichverteilte Zufallszahl x gegeben durch

$$\boxed{x(R) = F^{-1}(R)}$$

Bsp.: $p(x) = e^{-x} \quad \rightarrow \quad F(x) = \int_0^x e^{-\bar{x}} d\bar{x} = 1 - e^{-x} = R$

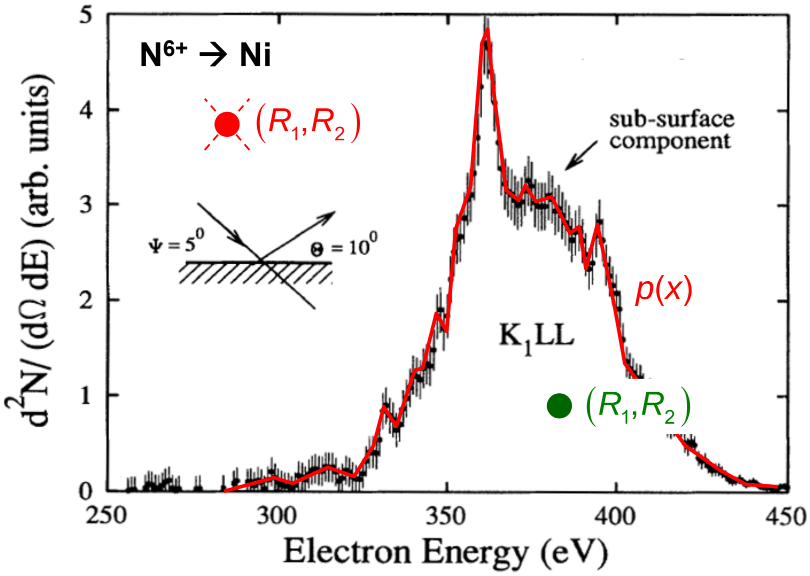
$$\rightarrow \quad x(R) = F^{-1}(R) = -\ln(1-R) = -\ln(R)$$

falls F^{-1} nicht analytisch bekannt

→ tabellierte Funktion $F(x)$ → direkter Vergleich mit R

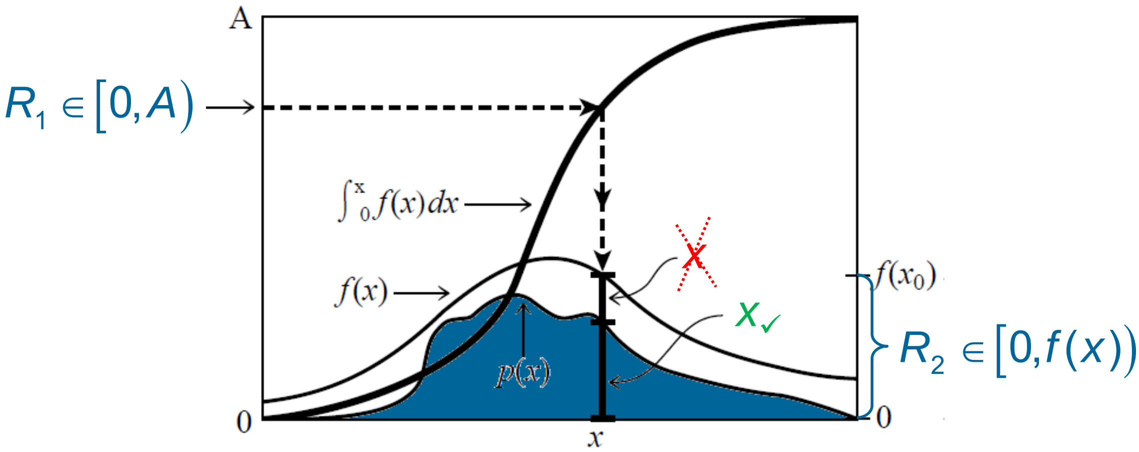
b) Zurückweisungsmethode

1) generiere 2 Zufallszahlen (als Koordinaten)



2) falls Punkt (R_1,R_2) unter Verteilung $p(x)$ → $x = R_1$

Beschleunigung: Vergleich mit (analytisch invertierbarer) Verteilungsfunktion $f(x) > p(x)$



1) bestimme x aus R_1 mithilfe der Transformationsmethode
2) verwende x als Zufallszahl falls $R_2 \le p(x)/f(x)$

Vorteil: weniger Zurückweisungen von Paaren (R_1, R_2)

Gaussverteilte Zufallszahlen:

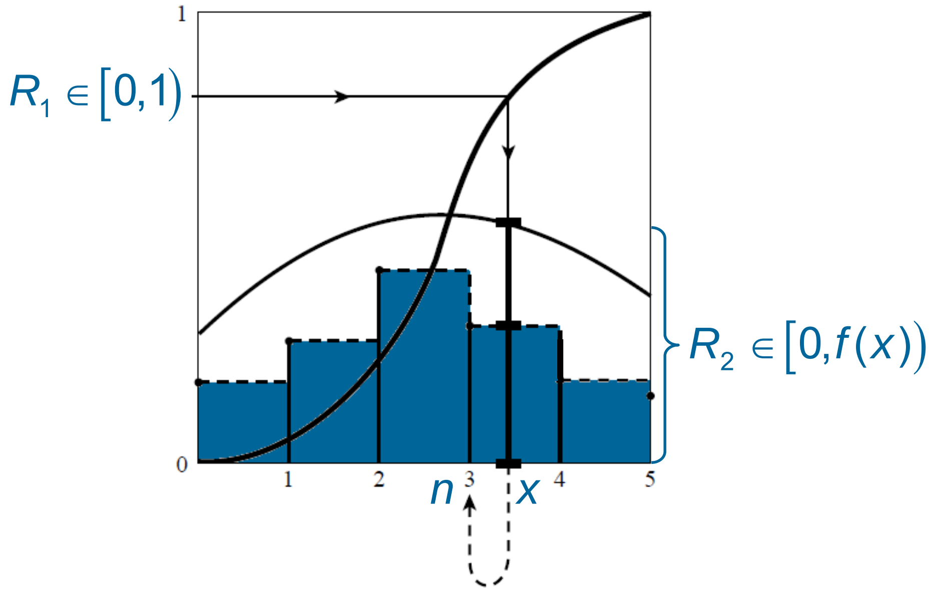
Box-Muller-Algorithmus:

- generiere $R_1 \in [0,1); R_2 \in [0,1)$
- berechne $\omega = 2\pi R_1;\ \rho = -2\ln R_2$
- berechne $x_1 = \sqrt{\rho}\cos\omega;\ x_2 = \sqrt{\rho}\sin\omega$

Polarmethode:

- generiere $R_1 \in [0,1); R_2 \in [0,1)$
- berechne $S = 2R_1 - 1; T = -2R_2 + 1$
- falls $W^2 = S^2 + T^2 > 1 \quad \rightarrow$ Neustart
- sonst $x_1 = (S/W)\sqrt{-2\ln W^2};\ x_2 = (T/W)\sqrt{-2\ln W^2}$

ganzzahlige Verteilungen (z.B. Poisson):



erweitere $p(n)$ auf Intervall $[n,n+1)$ und verwende **`n=int(x)`** als Zufallszahl (oder **`nint(x),floor(x)`**)

2.2 χ^2-Anpassung (fitting)

Definition:

linearer Fit: linear in Fitparametern

$$f(x) = \sum_{k=1}^{M} a_k X_k(x)$$

z.B.: $f(x) = a_1 + a_2 x + \ldots + a_M x^{M-1}$

$$f(t) = \sum_k a_k \sin(\omega_k t)$$

nicht-linearer Fit

$$f(x) = \sum_k a_k X_k(b_k x)$$

z.B.: $f(x) = a_1 \cdot \exp[-b_1 x] + a_2 \cdot \exp[-b_2 x]$

Ziele einer Anpassungsprozedur:

1) berechne Fitparameter

2) Fehlerabschätzung für jeden Parameter

3) Abschätzung der Qualität der Anpassungsfunktion

Wofür ist sie NICHT gedacht?

- Datenpunkte durch glatte Linie verbinden ($\rightarrow$ Interpolation)
- eine optisch ansprechende Linie durch Datenpunkte legen ($\rightarrow$ graphische Aufbereitung von Daten)

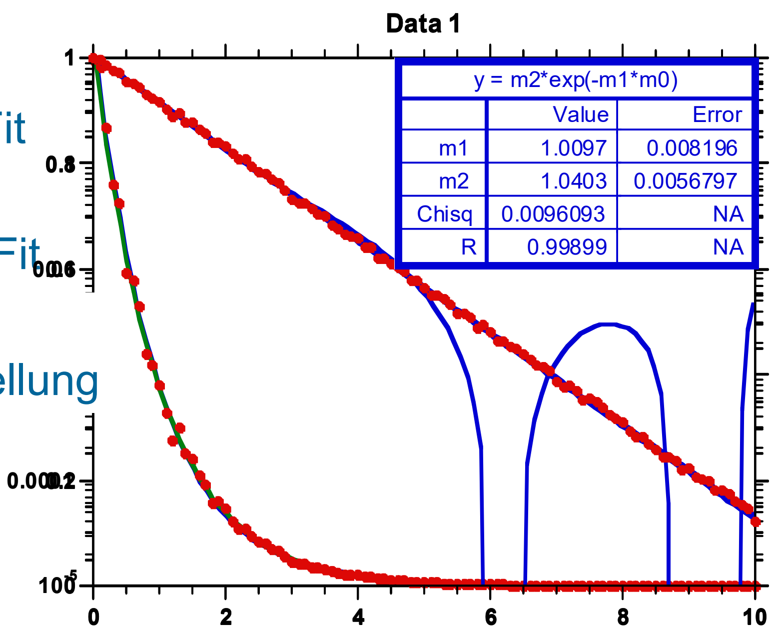
beide Punkte erlauben keine Aussage über physikalischen Prozess!

Beispiel: radioaktiver Zerfall

1. Versuch:
   Polynomial-Fit

2. Versuch:
   Exponential-Fit

Entscheidung:
   lin-log Darstellung



Maß für Abweichung der Daten von der Fitfunktion:

Messdaten

Fitfunktion

$$\chi^2 = \sum_{j=1}^{N} \left(\frac{y_j - f(x_j; a_1, \dots, a_M)}{\sigma_j} \right)^2$$

evtl. Messfehler

→ finde Minimum von χ^2

→ finde Nullstellen der *M* ersten Ableitungen

$$0 = 2 \cdot \sum_{j=1}^{N} \left(\frac{y_j - f(x_j; a_1, \dots, a_M)}{\sigma_j} \right) \left(\frac{\partial f(x_j; a_1, \dots, a_M)}{\partial a_k} \right)$$

Umordnung → System linearer Gleichungen

$$\left(\mathbf{A}^T \cdot \mathbf{A} \right)_{MxM} \cdot \vec{a}_M = \mathbf{A}^T_{MxN} \cdot \vec{b}_N$$

$$\mathbf{A}_{jk} = \frac{X_k(x_j)}{\sigma_j} \qquad b_j = \frac{y_j}{\sigma_j}$$

falls σ_j unbekannt → setze $\sigma_j = \sigma = 1$
nach Lösung des Fitproblems kann mittlerer Messfehler σ ermittelt werden:

$$\sigma^2 = \frac{\sum_{j=1}^{n} \left[y_j - f(x_j; a_1, \dots, a_M) \right]^2}{N - M}$$

Ziele der χ^2-Anpassung:

1. Fitparameter a_i ✓
2. Abschätzung des Fehlers der Fitparameter ✓

$$\sigma^2(a_k) = \left(\mathbf{A}^T \mathbf{A}\right)^{-1}_{kk}$$

3. Abschätzung für Qualität der Fitfunktion ✓

$$Q(\alpha, x) = \frac{1}{\Gamma(\alpha)} \int_x^\infty e^{-t} t^{\alpha-1}\, dt \quad \text{mit} \quad \begin{cases} \alpha = (N-M)/2 \\ x = \chi^2/2 \end{cases}$$

Anwendung: lineare Regression $y(x) = a + b \cdot x$

$$\chi^2 = \sum_{j=1}^{N} \left(\frac{y_j - a - bx_j}{\sigma_j} \right)^2 \rightarrow \begin{cases} \dfrac{\partial \chi^2}{\partial a} = 0 = -2 \displaystyle\sum_{j=1}^{N} \frac{y_j - a - bx_j}{\sigma_j^2} \\ \dfrac{\partial \chi^2}{\partial b} = 0 = -2 \displaystyle\sum_{j=1}^{N} \frac{x_j (y_j - a - bx_j)}{\sigma_j^2} \end{cases}$$

mit

$$S \equiv \sum_{j=1}^{N} \frac{1}{\sigma_j^2};\ S_x \equiv \sum_{j=1}^{N} \frac{x_j}{\sigma_j^2};\ S_y \equiv \sum_{j=1}^{N} \frac{y_j}{\sigma_j^2};\ S_{xx} \equiv \sum_{j=1}^{N} \frac{x_j^2}{\sigma_j^2};\ S_{xy} \equiv \sum_{j=1}^{N} \frac{x_j y_j}{\sigma_j^2}$$

$$\boxed{\left.\begin{aligned} aS + bS_x &= S_y \\ aS_x + bS_{xx} &= S_{xy} \end{aligned}\right\} \rightarrow \begin{cases} a = \dfrac{S_{xx} S_y - S_x S_{xy}}{S S_{xx} - S_x^2} \\ b = \dfrac{S S_{xy} - S_x S_y}{S S_{xx} - S_x^2} \end{cases}}$$

oft: $\sigma_j = \sigma = 1$
→ $S = N$

2.3 Interpolation

Behauptung: Es existiert nur ein Polynom vom Grad n oder weniger das n+1 reell-wertige Datenpunkte $f(a_i)$ an Abszissenwerten $a_0 \ldots a_n$ interpoliert.

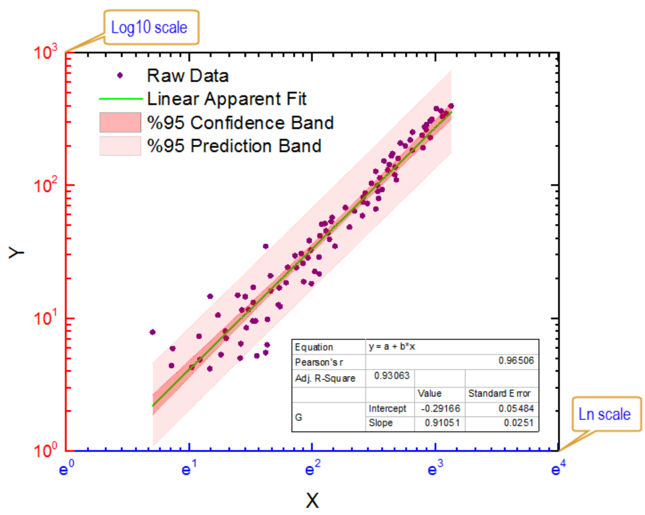
Beweis: Annahme: $\exists$ 2 Polynome $p_n^{(1)}(x), p_n^{(2)}(x)$

$\rightarrow$ $p_n^{(1)}(x) - p_n^{(2)}(x)$ hat n+1 Nullstellen an $a_0 \ldots a_n$, muss aber vom Grad n sein
→ Widerspruch

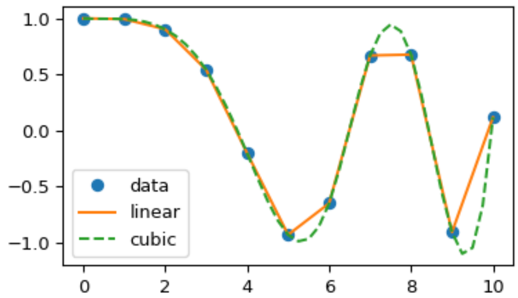
→ alle diskutierten Methoden liefern identische Lösung
→ Auswahl des Algorithmus passend zum Problem!

zur Unterscheidung:

Anpassung



Interpolation



direkte Bestimmung eines Funktionswerts von $p_n(x)$:

Basisfunktionen $P_i(x) = x^i \; ; \quad i = 0,\dots,n$

$$p_n(x) = \sum_{i=0}^{n} c_i \cdot P_i(x) = c_0 + c_1 x + c_2 x^2 + \dots + c_n x^n$$

$$p_n(a_i) = c_0 + c_1 a_i + c_2 a_i^2 + \dots + c_n a_i^n = f(a_i) \quad \rightarrow$$

$$\begin{pmatrix} 1 & a_0 & a_0^2 & \dots & a_0^n \\ 1 & a_1 & a_1^2 & \dots & a_1^n \\ 1 & a_2 & a_2^2 & & \vdots \\ \vdots & \vdots & & \ddots & \\ 1 & a_n & \dots & & a_n^n \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ \\ c_n \end{pmatrix} = \begin{pmatrix} f(a_0) \\ f(a_1) \\ \vdots \\ \\ f(a_n) \end{pmatrix}$$

Vandermonde-Matrix

Lagrange-Interpolation

$$p_n(x) = \sum_{k=0}^{n} f(a_k) \ell_k(x) \qquad \text{mit} \qquad \ell_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - a_i}{a_k - a_i}$$

wegen $\ell_k(a_j) = \delta_{jk} \quad \rightarrow \quad p_n(a_j) = f(a_j)$

Bsp.: 2 Punkte (lineare Interpolation)

$$p_1(x) = \sum_{k=0}^{1} f(a_k) \ell_k(x) = f(a_0) \frac{x - a_1}{a_0 - a_1} + f(a_1) \frac{x - a_0}{a_1 - a_0}$$

$$\boxed{= f(a_0) + (x - a_0) \frac{f(a_1) - f(a_0)}{a_1 - a_0}}$$

Berechnung von $p_n(x)$ benötigt mindestens

- O(n^2) Additionen
- O(n^2) Multiplikationen
- O(n) Divisionen (je ~4 Rechenoperationen)

vgl. Berechnung von $p_n(x)$ mit bekannten Koeffizienten c_i:

- n Additionen
- n Multiplikationen

durch Rekursion $\boxed{p_i(x) = p_{i-1}(x) \cdot x + c_{n-i}}$

(Horner-Schema)

Bsp.: $p_4(x) = \Big(\big((c_4 x + c_3)x + c_2\big)x + c_1\Big)x + c_0$

Newton-Interpolation

Basisfunktionen $N_0(x) = 1, N_i(x) = \prod_{j=0}^{i-1} (x - a_j)$

$$p_n(x) = \sum_{i=0}^{n} c_i \cdot N_i(x) = c_0 + c_1(x - a_0) + c_2(x - a_0)(x - a_1) + \ldots$$

$$p_n(a_i) = f(a_i) \quad \rightarrow$$

$$\begin{pmatrix} 1 & & & & 0 \\ 1 & N_1(a_1) & & & \\ 1 & N_1(a_2) & N_2(a_2) & & \\ \vdots & \vdots & & \ddots & \\ 1 & N_1(a_n) & \ldots & & N_n(a_n) \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ \\ c_n \end{pmatrix} = \begin{pmatrix} f(a_0) \\ f(a_1) \\ \vdots \\ \\ f(a_n) \end{pmatrix}$$

Berechnung der c_i → „dividierte Differenzen"

Definition:

$$f[a_0] = f(a_0)$$

$$f[a_0, a_1] = \frac{f[a_1] - f[a_0]}{a_1 - a_0}$$

$$f[a_0, a_1, a_2] = \frac{f[a_1, a_2] - f[a_0, a_1]}{a_2 - a_0}$$

$$\ldots$$

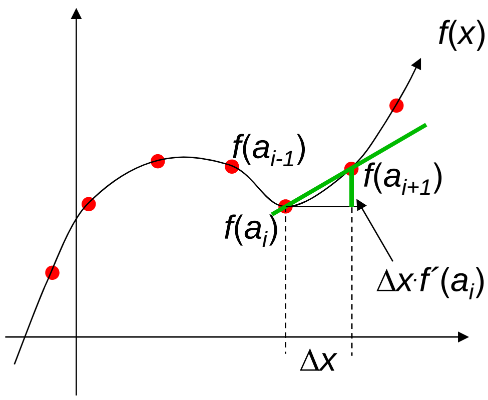
$$f[a_0, \ldots, a_n] = \frac{f[a_1, \ldots, a_n] - f[a_0, \ldots, a_{n-1}]}{a_n - a_0}$$

	$f[a_i, \ldots, a_j]$				
a_0	$f[a_0] = c_0$				
a_1	$f[a_1]$	$f[a_1, a_0] = c_1$			
a_2	$f[a_2]$	$f[a_2, a_1]$	$f[a_2, a_1, a_0] = c_2$		
$\ldots$	$\ldots$	$\ldots$	$\ldots$		
a_n	$f[a_n]$	$f[a_n, a_{n-1}]$	$f[a_n, a_{n-1}, a_{n-2}]$	$\ldots$	$f[a_n, \ldots, a_0] = c_n$

einmalige Berechnung der Koeffizienten c_i: $O(n^2)$

Berechnung von Funktionswerten $p_n(x)$: $O(n)$
→ effizient zur Berechnung von $p_n(x)$ für viele Werte von x
Numerischer Aufwand der zusätzlichen dividierten Differenzen für zusätzlichen Datenpunkt (alle $f[a_i \ldots a_j]$ gespeichert): $O(n)$

Ähnlichkeit zu Taylorreihen-Entwicklung:



$$f'(a_i) \approx \frac{f(a_{i+1}) - f(a_i)}{a_{i+1} - a_i}$$

$$f''(a_i) \approx \frac{f(a_{i-1}) - 2f(a_i) + f(a_{i+1})}{\Delta x^2}$$

Grenze $a_j \rightarrow a_0$:

$$f(x) = f[a_0] + (x - a_0) f[a_0, a_1] + (x - a_0)(x - a_1) f[a_0, a_1, a_2] + \ldots$$

$$= f(a_0) + (x - a_0)\; f'(a_0) \;+\; (x - a_0)^2 \;\frac{1}{2} f''(a_0) \;+ \ldots$$

ähnlich: Neville-Algorithmus

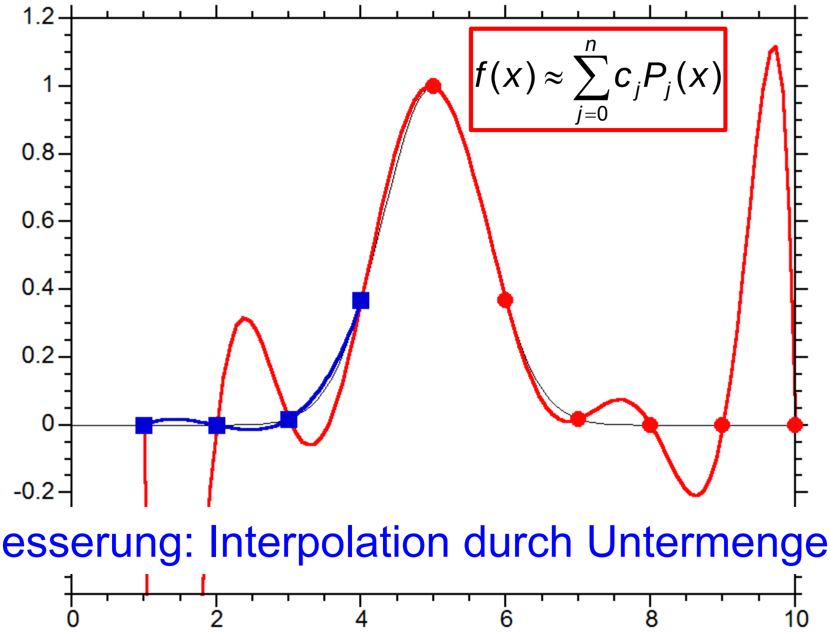
$p_k(x)$ bekannt an k+1 Abszissenwerten: $p(f \mid \underbrace{a_i, \ldots, a_j}_{k+1})$

$$p(f \mid a_i, \ldots, a_j) = \frac{(x - a_i) p(f \mid a_{i+1}, \ldots, a_j) - (x - a_j) p(f \mid a_i, \ldots, a_{j-1})}{a_j - a_i}$$

$$\begin{array}{llll}
p(f \mid a_0) & & & \\
p(f \mid a_1) & p(f \mid a_0, a_1) & & \\
p(f \mid a_2) & p(f \mid a_1, a_2) & p(f \mid a_0, a_1, a_2) & \\
\vdots & \vdots & \ddots & \ddots \\
p(f \mid a_n) & p(f \mid a_{n-1}, a_n) & \ldots & p(f \mid a_0, \ldots, a_n)
\end{array}$$

Effizient zur Berechnung weniger Funktionswerte

Problem: Oszillationen von Polynomen hohen Grades



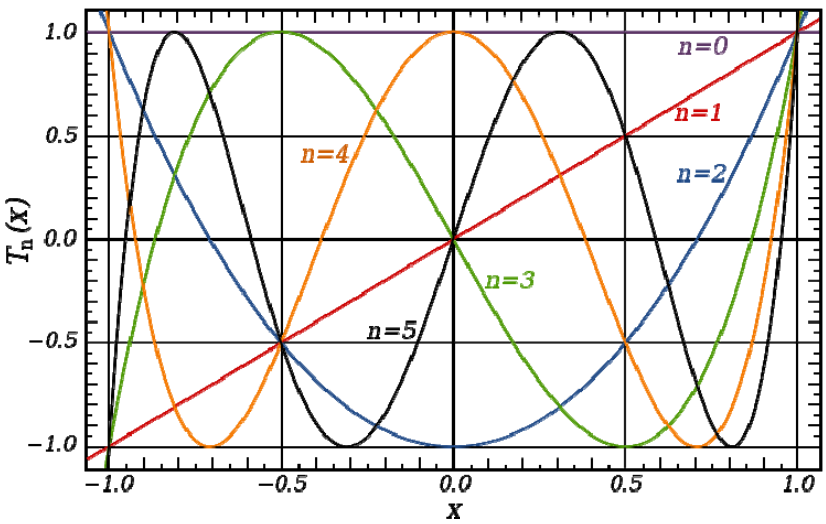
Verbesserung: Interpolation durch Untermenge $\{f(a_i)\}$

Entwicklung in orthogonale Basisfunktionen:

z.B. Tschebyscheff-Polynome:

$$T_n(x) = \cos\left[n \cdot \operatorname{acos}(x)\right] = 0 \quad \rightarrow \quad x_k = \cos\left(\frac{\pi(k+1/2)}{n}\right)$$

Rekursion: $T_{n+1}(x) = 2x \cdot T_n(x) - T_{n-1}(x); \quad T_0 = 1, T_1 = x$



Orthogonalität:

$$\int_{-1}^{1} \frac{T_i(x)T_j(x)}{\sqrt{1-x^2}}\,dx = \begin{cases} 0 & i \neq j \\ \pi/2 & i = j \neq 0 \\ \pi & i = j = 0 \end{cases}$$

$$\sum_{k=1}^{n+1} T_i(x_k)T_j(x_k) = \begin{cases} 0 & i \neq j \\ (n+1)/2 & i = j \neq 0 \\ n+1 & i = j = 0 \end{cases}$$

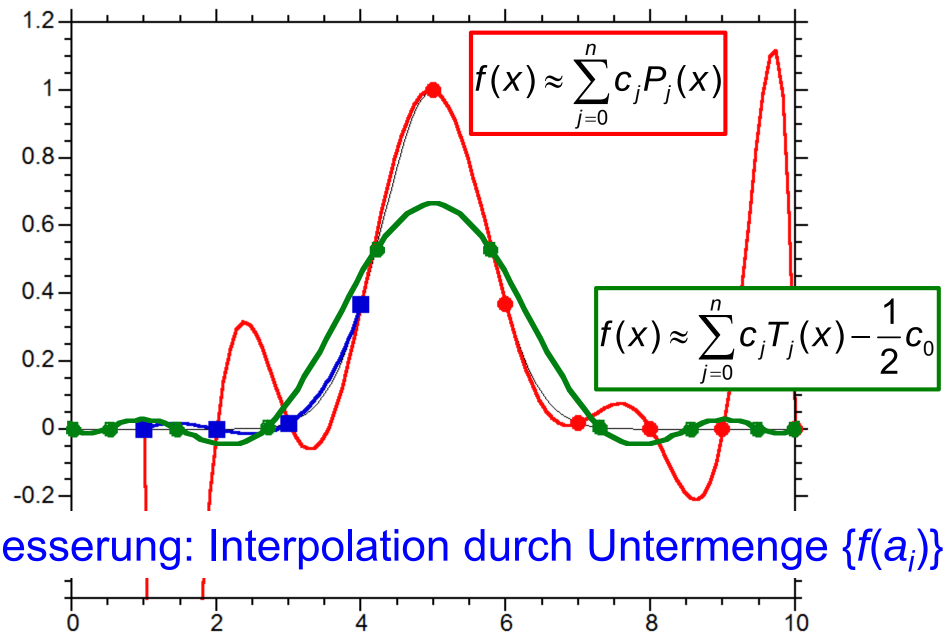
$$f(x) \approx \sum_{j=0}^{n} c_j T_j(x) - \frac{1}{2}c_0$$

$$c_j = \frac{2}{n+1}\sum_{k=1}^{n+1} f(x_k)T_j(x_k)$$

oft Konvergenz mit wenigen Koeffizienten c_j
numerisch effizient, wenn $f(x)$ analytisch bekannt!

$$[a,b] \neq [-1,1] \quad \rightarrow \quad x_k = \frac{b+a}{2} + \frac{b-a}{2}\cos\left(\frac{(2k-1)\pi}{2(n+1)}\right)$$

Problem: Oszillationen von Polynomen hohen Grades



Verbesserung: Interpolation durch Untermenge $\{f(a_i)\}$

(kubische) Spline-Interpolation

$f(x)$ bekannt an $n+1$ Abszissenwerten $a_0 < a_1 < \ldots < a_n$

Idee: $f(x) \approx p_n^{(i)}(x)$ im Intervall $[a_i, a_{i+1}]$
zusätzlich: zweimal stetig differenzierbar an Intervallgrenzen

meistens: n = 3 (→ kubischer Spline)

$$\boxed{\begin{aligned} p_3^{(i)}(x) = c_3^{(i)}(x-a_i)^3 + c_2^{(i)}(x-a_i)^2 + c_1^{(i)}(x-a_i) + c_0^{(i)} \\ i = 0, \ldots, n-1 \end{aligned}}$$

Bestimmung der Koeffizienten aus Stetigkeitsbedingungen

$$p_3^{(i)}(x) = c_3^{(i)}(x-a_i)^3 + c_2^{(i)}(x-a_i)^2 + c_1^{(i)}(x-a_i) + c_0^{(i)}$$

Stetigkeit:

$$p_3^{(i)}(a_i) = f(a_i) = \boxed{f_i = c_0^{(i)}} = p_3^{(i-1)}(a_i); \quad i = 1, \ldots, n-1$$

$$p_3'^{(i)}(a_i) = c_1^{(i)} = p_3'^{(i-1)}(a_i); \quad i = 1, \ldots, n-1$$

$$p_3''^{(i)}(a_i) = \boxed{2c_2^{(i)} = S_i} = p_3''^{(i-1)}(a_i); \quad i = 1, \ldots, n-1$$

$$p_3''^{(i)}(a_{i+1}) = 2c_2^{(i)} + 6c_3^{(i)}\Delta_i = S_{i+1} \quad \rightarrow \quad \boxed{c_3^{(i)} = \frac{1}{6\Delta_i}\left(S_{i+1} - S_i\right)}$$

$$p_3^{(i)}(a_{i+1}) = f_{i+1} = \frac{\Delta_i^2}{6}\left(S_{i+1} - S_i\right) + \frac{S_i}{2}\Delta_i^2 + c_1^{(i)}\Delta_i + f_i$$

$\Delta_i = a_{i+1} - a_i$

$$\rightarrow \quad \boxed{c_1^{(i)} = \frac{f_{i+1} - f_i}{\Delta_i} - \frac{\Delta_i}{6}(S_{i+1} + 2S_i)}$$

$$p_3^{(i)}(x) = c_3^{(i)}(x-a_i)^3 + c_2^{(i)}(x-a_i)^2 + c_1^{(i)}(x-a_i) + c_0^{(i)}$$

Stetigkeit der 1. Ableitung an a_i

$$p_3'^{(i)}(a_i) = c_1^{(i)} = 3c_3^{(i-1)}\Delta_{i-1}^2 + 2c_2^{(i-1)}\Delta_{i-1} + c_1^{(i-1)} = p_3'^{(i-1)}(a_i)$$

einsetzen und umordnen

$$\boxed{\begin{aligned} 6\left(\frac{f_{i+1}-f_i}{\Delta_i} - \frac{f_i - f_{i-1}}{\Delta_{i-1}}\right) = \bar{y}_i &= \Delta_{i-1}S_{i-1} + 2(\Delta_{i-1}+\Delta_i)S_i + \Delta_i S_{i+1} \\ 6\, f\left[a_{i-1}, a_i, a_{i+1}\right] = y_i &= \frac{\Delta_{i-1}}{\Delta_{i-1}+\Delta_i}S_{i-1} + 2S_i + \frac{\Delta_i}{\Delta_{i-1}+\Delta_i}S_{i+1} \end{aligned}}$$

n-1 Gleichungen für n+1 Unbekannte → unterbestimmt

→ tridiagonales Gleichungssystem:

$$\begin{pmatrix} \frac{\Delta_0}{\Delta_0+\Delta_1} & 2 & \frac{\Delta_1}{\Delta_0+\Delta_1} & & 0 \\ & \ddots & \ddots & & \\ & \frac{\Delta_{i-1}}{\Delta_{i-1}+\Delta_i} & 2 & \frac{\Delta_i}{\Delta_{i-1}+\Delta_i} & \\ & & \ddots & \ddots & \\ 0 & & \frac{\Delta_{n-2}}{\Delta_{n-2}+\Delta_{n-1}} & 2 & \frac{\Delta_{n-1}}{\Delta_{n-2}+\Delta_{n-1}} \end{pmatrix} \begin{pmatrix} S_0 \\ \vdots \\ S_{i-1} \\ S_i \\ S_{i+1} \\ \vdots \\ S_n \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_i \\ \vdots \\ y_{n-1} \end{pmatrix}$$

fehlende 2 Gleichungen: freie Wahl von S_0, S_n

numerisch effiziente Lösung des Gleichungsproblems: siehe VO zu Numerik der linearen Algebra

Wahl von S_0, S_n

- „natürlicher Spline": $S_0 = S_n = 0$
- $S_0 = S_1$, $S_{n\text{-}1} = S_n$
- Extrapolation: $S_2, S_1 \to S_0$, $S_{n\text{-}2}, S_{n\text{-}1} \to S_n$
- Schätzwert für S_0, S_n
- „vollständiger Spline": Definition zusätzlicher y_0, y_n

  $$y_0 = 6\frac{f[a_0,a_1] - p_3'^{(0)}(a_0)}{\Delta_0}; \quad y_n = 6\frac{p_3'^{(n-1)}(a_n) - f[a_{n-1},a_n]}{\Delta_{n-1}}$$
- „periodischer Spline": 1. und 2. Ableitung gleich am Rand $p_{3,0}'^{(0)} = p_{3,n}'^{(n-1)}$; $p_{3,0}''^{(0)} = S_0 = S_n = p_{3,n}''^{(n-1)}$
  Vorsicht: Matrix nicht mehr tridiagonal!