Nichtlineare Modelle, Regression, Interpolation, numerische Integration

7. Vorlesung

170.021 Numerische Methoden 1

(170.026 Numerische Methoden I)

Clemens Brand, Erika Hausenblas, Alexander Steinicke

Montanuniversität Leoben

27. April 2023

(Nachtrag: Material der 6. Vorlesung, das noch nicht behandelt wurde)

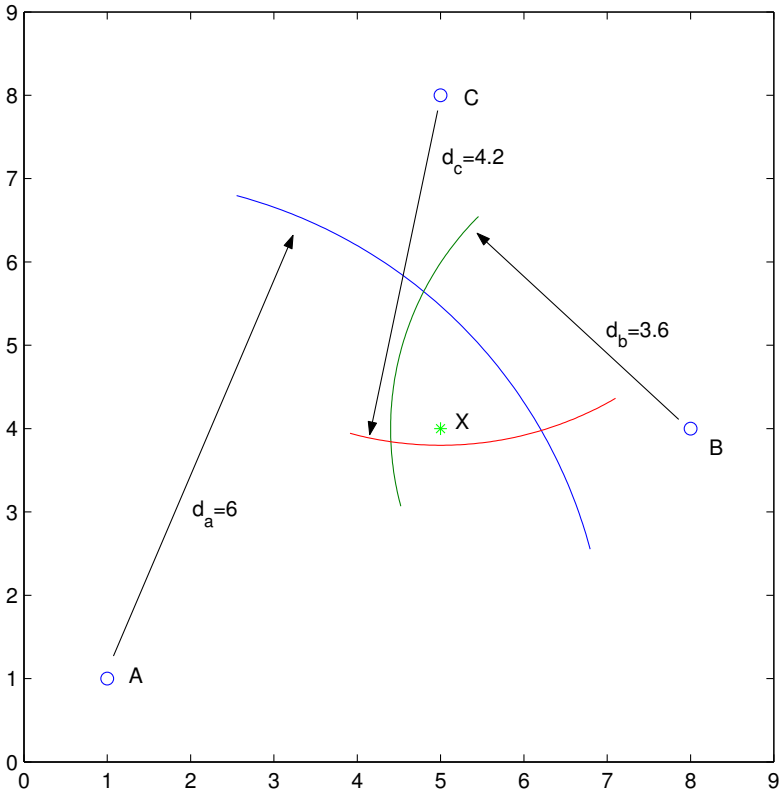
Überbestimmte nichtlineare Systeme

Beispiel: Standortbestimmung durch Trilateration

Die Abstände von drei festen Punkten A, B, C zu einem unbekannten Punkt X sind (etwas ungenau) bekannt. Gesucht ist eine möglichst gute Positionsbestimmung.

$$\begin{aligned}\sqrt{(x_1-1)^2+(x_2-1)^2} &= 6\\ \sqrt{(x_1-8)^2+(x_2-4)^2} &= 3.6\\ \sqrt{(x_1-5)^2+(x_2-8)^2} &= 4.2\end{aligned}$$

Den drei Gleichungen entsprechen drei Kreise im $\mathbb{R}^2$. Sie haben keinen gemeinsamen Schnittpunkt.



Überbestimmte nichtlineare Systeme

Lösung durch Linearisierung und Iteration

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}\ , \quad \mathbf{x} \in \mathbb{R}^n\ , \quad \mathbf{f}(\mathbf{x}) \in \mathbb{R}^m, \quad m > n$$

Ausgehend von Startvektor $\mathbf{x}^{(0)}$ bestimmt man eine Korrektur $\Delta\mathbf{x}$.
Die Rechenvorschrift des Newton-Verfahrens für $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ ergibt ein überbestimmtes lineares System mit der Jacobimatrix D_f

$$D_f \cdot \Delta\mathbf{x} = -\mathbf{f}(\mathbf{x})$$

Verbesserte Lösung $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \Delta\mathbf{x}$.

Für die Konvergenz der Iteration kann Unterrelaxation (Dämpfung) notwendig sein:
$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} + \omega\Delta\mathbf{x}$ mit Unterrelaxationsfaktor $0 < \omega \leq 1$.

Rechenbeispiel von vorhin

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \sqrt{(x_1-1)^2+(x_2-1)^2} - 6 \\ \sqrt{(x_1-8)^2+(x_2-4)^2} - 3.6 \\ \sqrt{(x_1-5)^2+(x_2-8)^2} - 4.2 \end{bmatrix}, D_f = \begin{bmatrix} \frac{x_1-1}{\sqrt{(x_1-1)^2+(x_2-1)^2}} & \frac{x_2-1}{\sqrt{(x_1-1)^2+(x_2-1)^2}} \\ \frac{x_1-8}{\sqrt{(x_1-8)^2+(x_2-4)^2}} & \frac{x_2-4}{\sqrt{(x_1-8)^2+(x_2-4)^2}} \\ \frac{x_1-5}{\sqrt{(x_1-5)^2+(x_2-8)^2}} & \frac{x_2-8}{\sqrt{(x_1-5)^2+(x_2-8)^2}} \end{bmatrix}$$

Mit Startvektor $\mathbf{x} = \begin{bmatrix} 5 \\ 4 \end{bmatrix}$ erhält man

$$\mathbf{f}\left(\begin{bmatrix} 5 \\ 4 \end{bmatrix}\right) = \begin{bmatrix} -1 \\ -3/5 \\ -1/5 \end{bmatrix}, D_f = \begin{bmatrix} \frac{4}{5} & \frac{3}{5} \\ -1 & 0 \\ 0 & -1 \end{bmatrix}, \text{ lin. Syst. } \begin{bmatrix} \frac{4}{5} & \frac{3}{5} \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 3/5 \\ 1/5 \end{bmatrix}$$

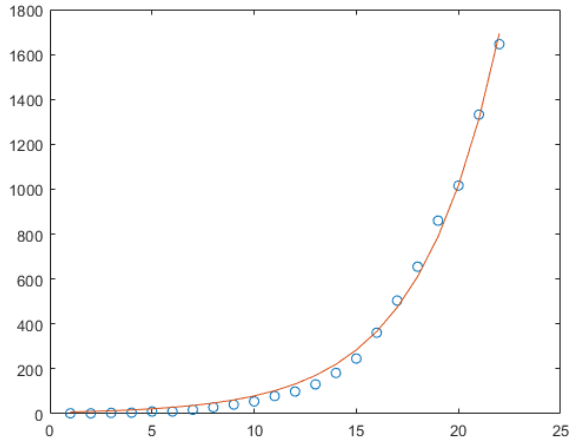
Ergibt $\Delta x_1 = 1/25, \Delta x_2 = 7/25 \quad \longrightarrow$ verbesserte Position $[5.04; 4.28]$.

Noch ein Beispiel: Datenmodell anpassen

Datenpunkte und Modellgleichungen sind gegeben, Parameter sind gesucht

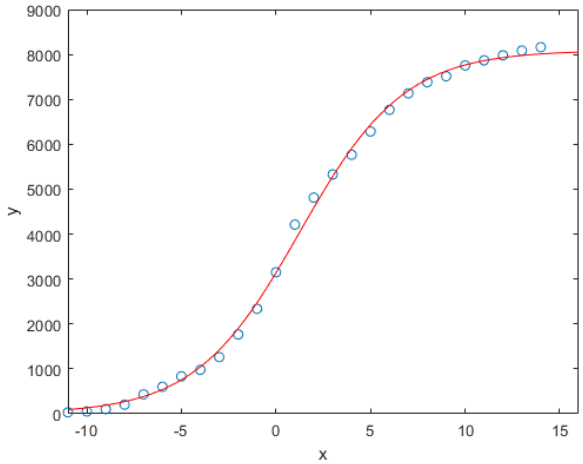
$$y = ae^{bx}$$

$$a, b = ?$$



$$y = \frac{a}{1 + e^{-b(x-c)}}$$

$$a, b, c = ?$$



(Siehe Zusatzmaterial `GaussNewtonCoronaFit.m`)

Datenmodell anpassen

Gauß-Newton-Verfahren

Datenpunkte

x	y
1	2
2	2
3	4
$\vdots$	$\vdots$
20	1016
21	1332
22	1646

Gleichungen

$a\exp(bx)$	$= y$
$a\exp(b\ 1)$	$= 2$
$a\exp(b\ 2)$	$= 2$
$a\exp(b\ 3)$	$= 4$
$\vdots$	$\vdots$
$a\exp(b\ 20)$	$= 1016$
$a\exp(b\ 21)$	$= 1332$
$a\exp(b\ 22)$	$= 1646$

Datenmodell anpassen

Gauß-Newton-Verfahren

Fehler $\mathbf{f} =$

$$\begin{bmatrix} a\exp(bx) - y \\ \hline \\ \exp(1b) - 2 \\ \exp(2b) - 2 \\ \exp(3b) - 4 \\ \vdots \\ \exp(20b) - 1016 \\ \exp(21b) - 1332 \\ \exp(22b) - 1646 \end{bmatrix}$$

Jacobi-Matrix $J =$

$$\begin{bmatrix} \exp(bx) & ax\exp(bx) \\ \hline \\ \exp(1b) & 1a\exp(1b) \\ \exp(2b) & 2a\exp(2b) \\ \exp(3b) & 3a\exp(3b) \\ \vdots & \vdots \\ \exp(20b) & 20a\exp(20b) \\ \exp(21b) & 21a\exp(21b) \\ \exp(22b) & 22a\exp(22b) \end{bmatrix}$$

Startvektor

$$\mathbf{x}^{(0)} = \begin{bmatrix} 2 \\ 0.3 \end{bmatrix}$$

J und $\mathbf{f}$ für $\mathbf{x}^{(0)}$ auswerten

Korrektur $\Delta\mathbf{x}$ aus überbestimmtem System

$$J\Delta\mathbf{x} = -\mathbf{f}$$

nächste Näherung

$$\mathbf{x}^{(1)} = \mathbf{x}^{(1)} + \omega\Delta\mathbf{x}$$

mit $\omega \approx 0.5$

Polynomiale Regression: Aufgabenstellung

Gesucht ist ein Polynom, das die Datenpunkte möglichst gut approximiert

Gegeben

$m+1$ Wertepaare (x_i, y_i), $\quad i = 0, \ldots, m$

Gesucht

$p(x)$, ein Polynom n-ten Grades, $n < m$, so dass die Summe der Fehlerquadrate

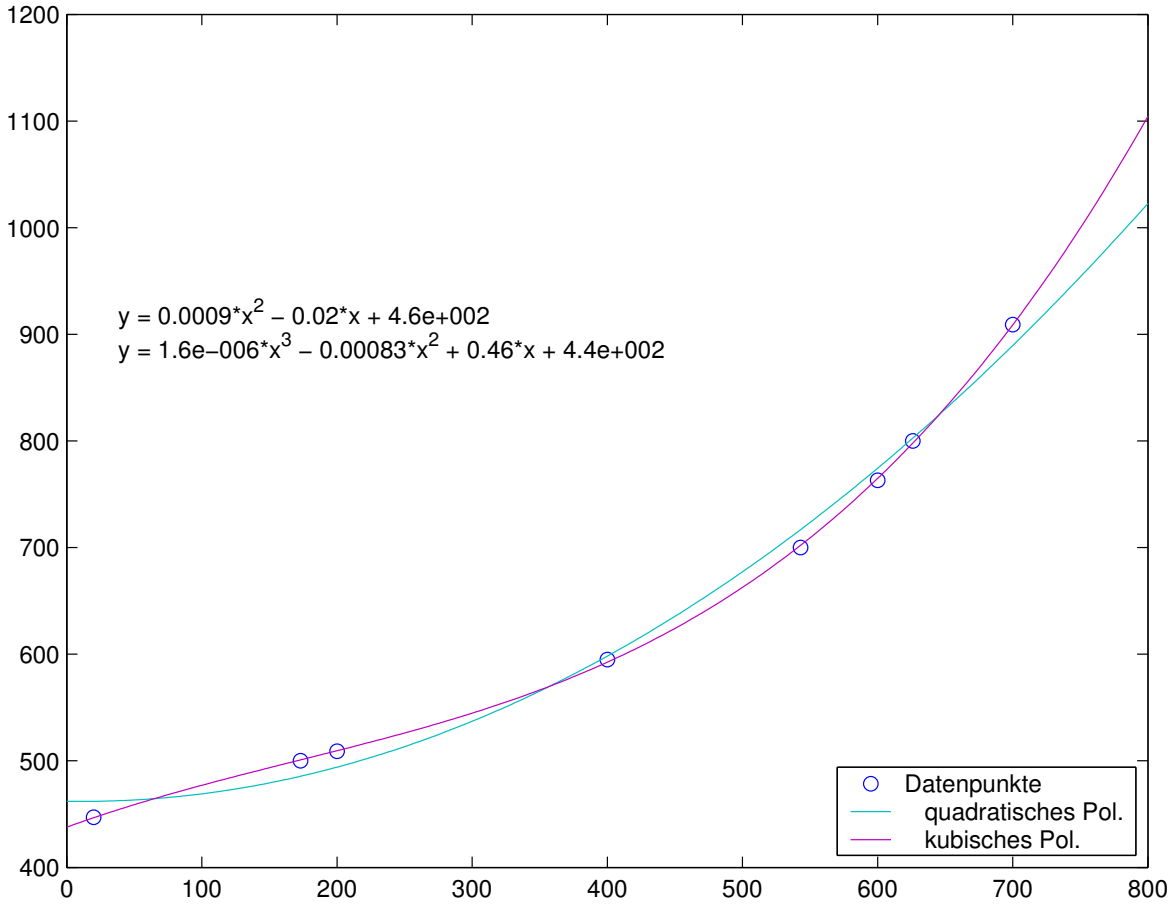
$$\sum_{i=0}^{m} (p(x_i) - y_i)^2$$

minimal wird.

Anpassen eines Polynoms an Datenpunkte

Spezifische Wärmekapazität von kohlenstoffarmem Stahl in $\mathrm{J/kg\,K}$ für $20\,\mathrm{C} \leq T \leq 700, \mathrm{C}$

T	c_p
20	447
173	500
200	509
400	595
543	700
600	763
626	800
700	909



Die Abbildung illustriert polynomiale Regression (quadratisch und kubisch) an die gegebenen Datenpunkte.

Polynomiale Regression

Zusammenfassung, grob vereinfacht

- macht man, weil Polynome die erstbesten Funktionen sind, die sich für Datenmodelle anbieten.
- ist ein Spezialfall der Anpassung von linearen Modellen, die bereits behandelt wurde. (Ansatzfunktionen sind nichtlinear, aber die gesuchten Koeffizienten treten nur linear auf!)
- für die Normalengleichungs-Matrix gibt es eine einfache Formel
- für Polynome hohen Grades (ab $n \approx 15 - 20$) ist der naive Ansatz $a_0 + a_1 x + a_2 x^2 + \cdots x^n$ völlig ungeeignet. Abhilfe:
  - Für Anfänger: Lassen Sie 's bleiben!
  - Für Fortgeschrittene: Verwenden Sie Orthogonalpolynome!

Direkter Lösungsweg

Ansatz des Polynoms mit unbestimmten Koeffizienten

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n \ .$$

- Einsetzen der gegebenen Wertepaare führt auf ein System von m linearen Gleichungen in den $n+1$ unbekannten Koeffizienten $a_0, a_1, \ldots, a_n$.
- Die Matrix A hat eine spezielle Form *(Vandermonde-Matrix)*:

$$A = \begin{bmatrix} 1 & x_0 & x_0^2 & x_0^3 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & x_1^3 & \ldots & x_1^n \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & x_m & x_m^2 & x_m^3 & \ldots & x_m^n \end{bmatrix}$$

- Standard-Lösung am Rechner durch QR-Zerlegung
- Bei kleinen Problemen und Rechnung mit Papier und Stift: klassisch nach der Methode der Normalengleichungen.

Formel für die Normalengleichungen

Bei polynomialer Regression haben die Normalengleichungen spezielle Form; man kann die Koeffizienten direkt angeben.

$$\begin{bmatrix} s_0 & s_1 & \dots & s_n \\ s_1 & s_2 & \dots & s_{n+1} \\ \vdots & \vdots & & \vdots \\ s_n & s_{n+1} & \dots & s_{2n} \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} t_0 \\ t_1 \\ \vdots \\ t_n \end{bmatrix}$$

$$\text{mit} \quad s_k = \sum_{i=0}^{m} x_i^k, \quad t_k = \sum_{i=0}^{m} x_i^k y_i$$

Praktisch nur bei linearer oder vielleicht noch quadratischer Regression sinnvoll. Moderner Lösungsweg: Vandermonde-Matrix aufstellen, QR-Lösung

Normalengleichungen, Spezialfall Ausgleichsgerade

Gleichung der Ausgleichsgeraden: $y = a_0 + a_1 x$

Das 2×2-Gleichungssystem für die Koeffizienten a_0 a_1 lautet

$$\begin{bmatrix} s_0 & s_1 \\ s_1 & s_2 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} t_0 \\ t_1 \end{bmatrix} \qquad \text{mit}$$

$$s_0 = m + 1,\ s_1 = \sum_{i=0}^{m} x_i,\ s_2 = \sum_{i=0}^{m} x_i^2,\ t_0 = \sum_{i=0}^{m} y_i,\ t_1 = \sum_{i=0}^{m} x_i y_i$$

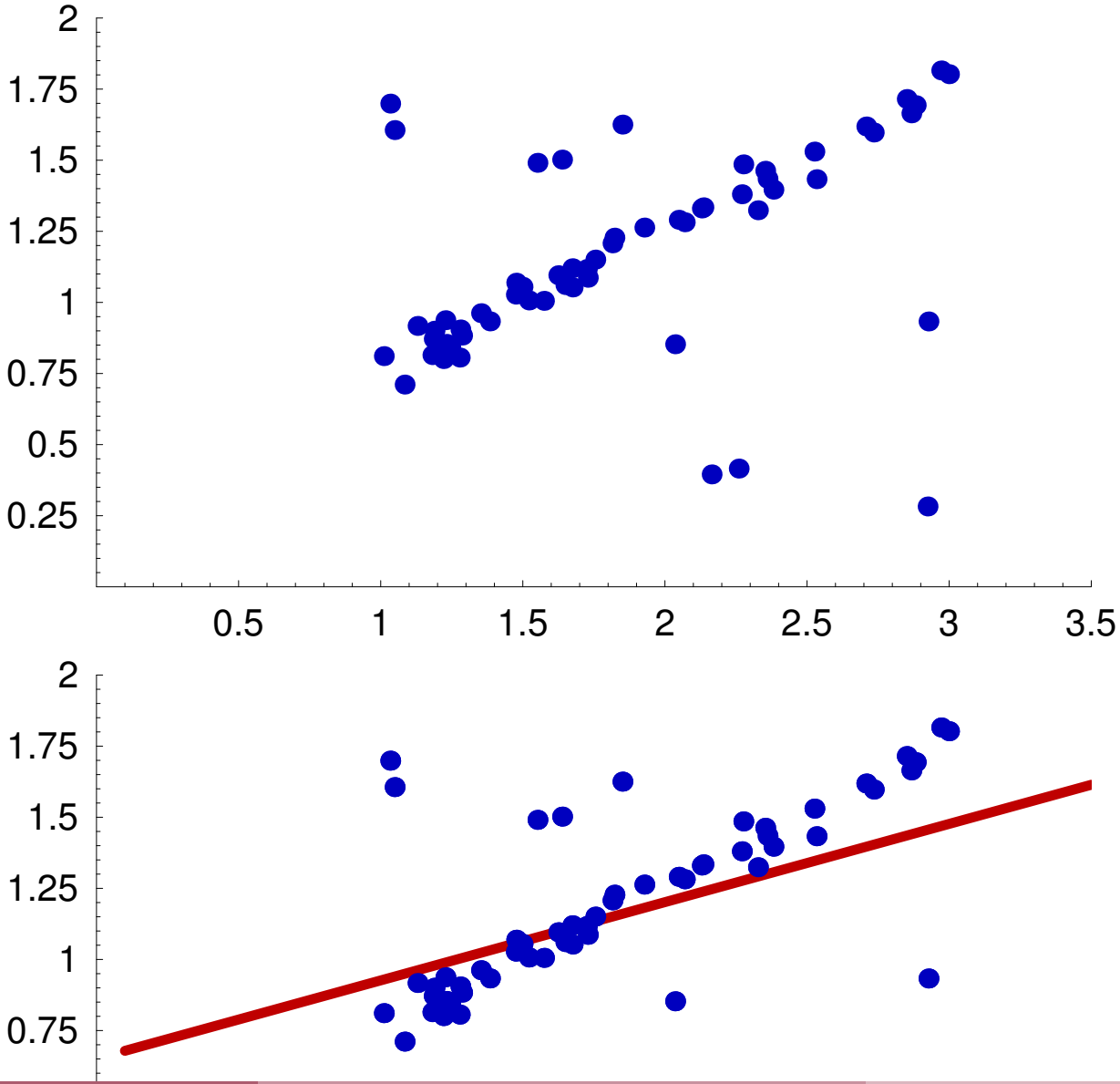
Lösung des Gleichungssystems:

$$a_0 = \frac{s_2 t_0 - s_1 t_1}{s_0 s_2 - s_1^2} \qquad a_1 = \frac{s_0 t_1 - s_1 t_0}{s_0 s_2 - s_1^2}$$

(Je nachdem, wo Sie nachschauen, finden Sie unterschiedliche Schreibweisen dieser Formeln.
Beachten Sie hier: Anzahl der Datenpunkte $= m + 1$)

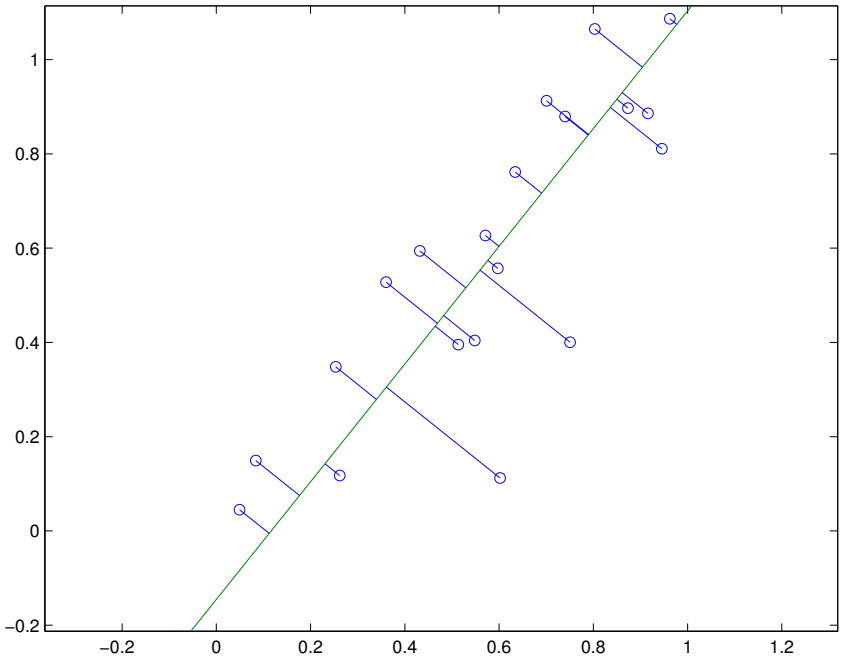
Ausgleichsgerade anpassen

Einfacher Spezialfall der polynomialen Regression



Total Least Squares mit SVD

Standardverfahren minimiert Summe der Abstandsquadrate in y-Richtung, TLS minimiert Quadratsumme der Normalabstände



Bestimme Schwerpunkt $[\bar{x}, \bar{y}]$ der Daten.

$$\bar{x} = \frac{1}{n} \sum_{i=1,n} x_i, \quad \bar{y} = \frac{1}{n} \sum_{i=1,n} y_i$$

Verschiebe die Daten

$$\Delta x_i = x_i - \bar{x}, \quad \Delta y_i = y_i - \bar{y}$$

Bilde Singulärwertzerlegung

$$U \cdot S \cdot V^T = \begin{bmatrix} \Delta x_1 & \Delta y_1 \\ \vdots & \vdots \\ \Delta x_n & \Delta y_n \end{bmatrix}$$

TLS-Gerade geht durch den Schwerpunkt in Richtung des ersten Spaltenvektors von V .

Schätzen von Modellparametern

Die Methode der kleinsten Quadrate ist das Standardverfahren zur Schätzung von Modellparametern. Unter bestimmten Annahmen liefert sie eine „beste“ Schätzung. Nämlich, im Spezialfall polynomiale Regression:

- Modell $y_i = a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_n x_i^n + \epsilon_i$
  Die $a_0, \ldots a_n$ sind unbekannte Parameter, die aus Beobachtungen $\{[x_i, y_i],\ i = 0, \ldots, m\}$ geschätzt werden sollen.
  Die Daten sind durch unbekannte zufällige Störgrößen ϵ_i verrauscht.
- $\mathbb{E}[\epsilon_i] = 0$ Alle Störgrößen haben Erwartungswert 0.
- $\mathbb{E}[\epsilon_i^2] = \sigma^2$ gleiche Varianz.
- $\mathbb{E}[\epsilon_i \epsilon_j] = 0$ für $i \neq j$ unkorellierte Störgrößen

Unter diesen Annahmen gilt der

Satz von Gauß-Markow

Der Kleinste-Quadrate-Schätzer ist ein bester linearer erwartungstreuer Schätzer, englisch: *Best Linear Unbiased Estimator*, kurz: BLUE

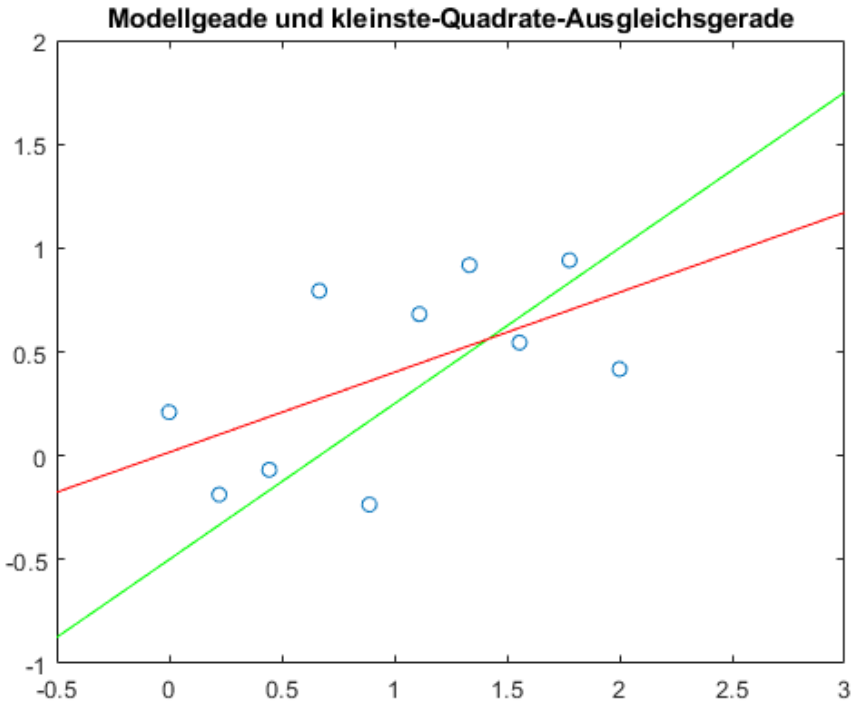
Beispiel zur Parameter-Schätzung

Dazu gibt es eine MATLAB-Datei `linRegKonfid.m`

Annahme: linearer Zusammenhang

$$y = a + bx \quad \text{mit } a = -\frac{1}{2},\ b = \frac{3}{4}$$

Zehn Messpunkte, mit normalverteilten Störgrößen verrauscht, $\sigma = \frac{1}{2}$.
Ausgleichsgerade, geschätzte
$\hat{a} = 0{,}016\,;\ \hat{b} = 0{,}385\,;\quad \hat{\sigma} = 0{,}390$



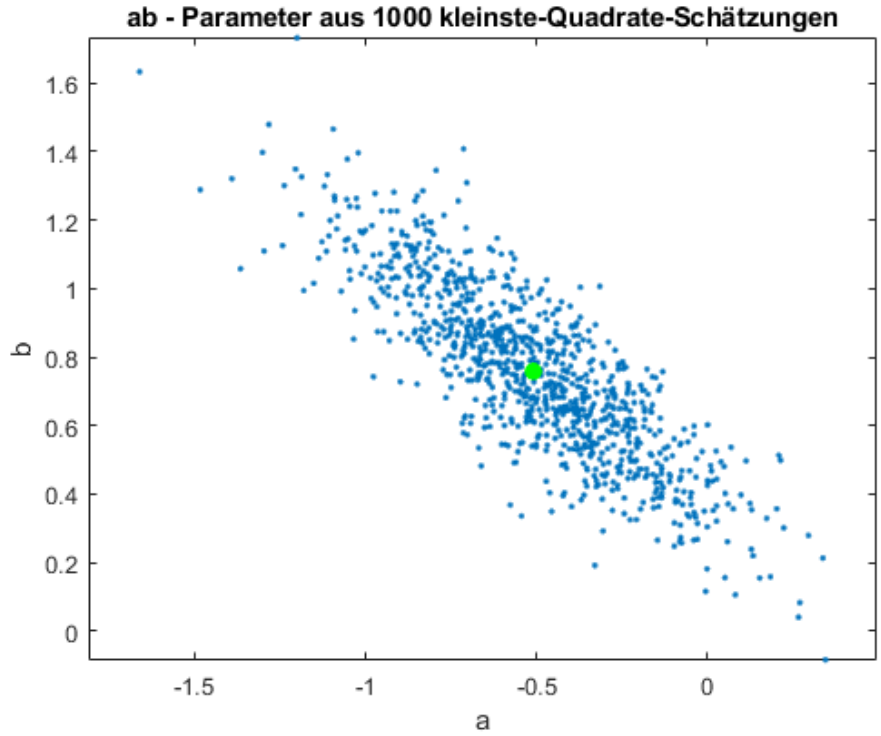
Jede Wiederholung der Messung mit anders gestörten Datenpunkten würde andere Schätzungen für a und b liefern. Was lässt sich über die Unsicherheit der Schätzungen aussagen?

Beispiel zur Parameter-Schätzung: 1000 Experimente

Annahme: linearer Zusammenhang

$$y = a + bx \quad \text{mit } a = -\frac{1}{2},\ b = \frac{3}{4}$$

Zehn Messpunkte, normalverteilt mit $\sigma = \frac{1}{2}$ verrauscht.
Gezeigt sind *ab*-Schätzungen aus 1000 Wiederholungen. Die Werte streuen in einer elliptischen Region
Die Ellipse mit Hauptachsen je eine Standardabweichung ist eingetragen.



Die Streuung entlang der *a*- bzw. *b*-Achse lässt sich aus der Kovarianzmatrix ablesen:

$$\sqrt{Cov_{11}} = 0{,}294\,;\ \sqrt{Cov_{22}} = 0{,}248$$

Kovarianzmatrix $Cov =$

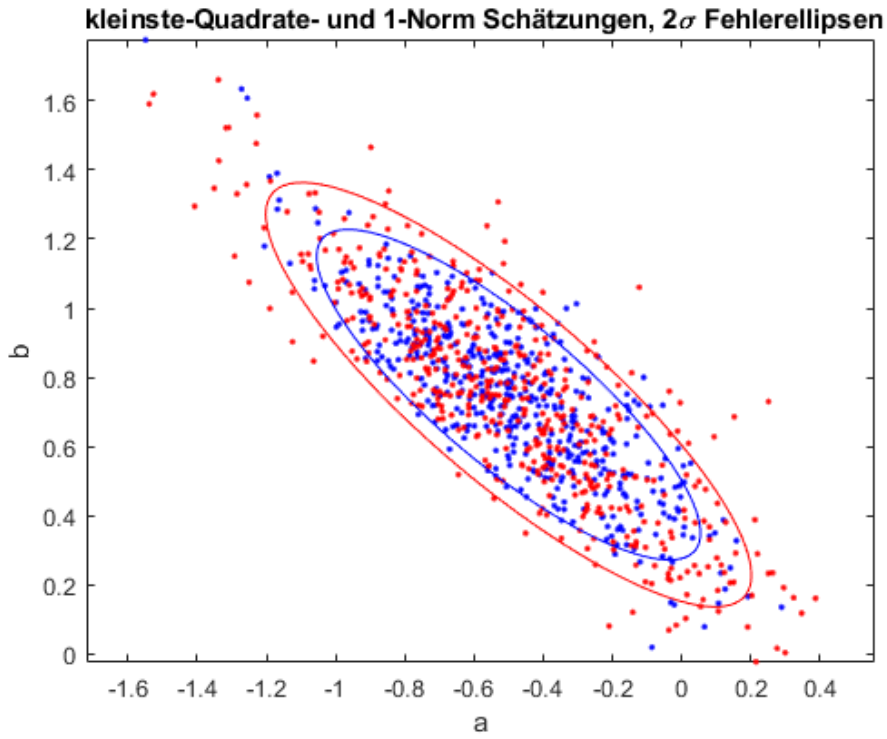
$$\sigma^2 (A^T A)^{-1} = \begin{bmatrix} 0.086 & -0.061 \\ -0.061 & 0.061 \end{bmatrix}$$

Parameter-Schätzung und Gauß-Markov Theorem

Annahme: linearer Zusammenhang

$$y = a + bx \quad \text{mit } a = -\frac{1}{2},\ b = \frac{3}{4}$$

Zehn Messpunkte, normalverteilt mit $\sigma = \frac{1}{2}$ verrauscht.
Gezeigt sind ab-Schätzungen aus 500 Wiederholungen mit den 2σ Fehlerellipsen. Blau: Kleinste Quadrate, Rot: Kleinste 1-Norm.



Satz von Gauß-Markow

Der Kleinste-Quadrate-Schätzer ist ein bester linearer erwartungstreuer Schätzer, englisch: *Best Linear Unbiased Estimator*, kurz: BLUE

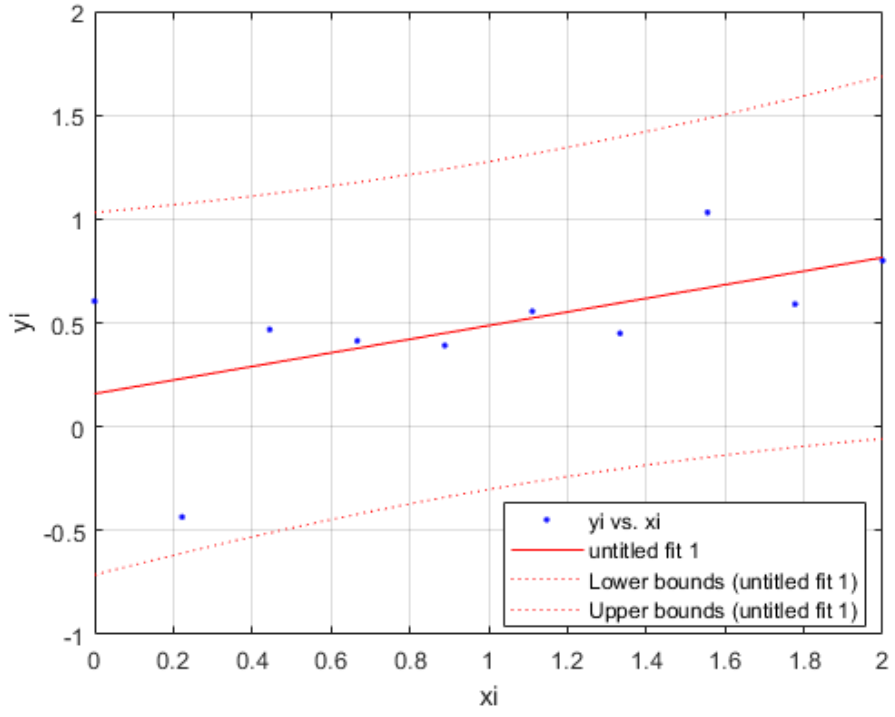
Parameter-Schätzung: Konfidenzintervalle

Die MATLAB-Datei in den Unterlagen zeigt sowohl explizit die Formeln als auch den Aufruf des *curve-fitting tools*

```
Linear model Poly1:
fitresult(x) = p1*x + p2
Coefficients (with 95% confidence bounds):
p1 = 0.3279 (-0.04482, 0.7007)
p2 = 0.1589 (-0.2833, 0.6011)
```

werden aus der geschätzten Varianz mittels Student-t-Verteilung bestimmt.

Die strichlierten Grenzen, *prediction bounds*, beziehen sich *nicht* auf die Genauigkeit der Modell-Voraussagen, sondern sagen den Streubereich weiterer Messungen voraus: mit 95%iger Wahrscheinlichkeit liegt ein weiter Messpunkt innerhalb der Grenzen.



Übrigens: zufällig ist das ein Fall, wo das Konfidenzintervall die tatsächlichen Werte $a = -0.5\,;\ b = 0.75$ nicht enthält. Sollte nur in 5% der Fälle passieren, aber: *Demonstrationseffekt*!

Interpolation

Definition der Aufgabenstellung

Gegeben:

Datenpunkte

Gesucht:

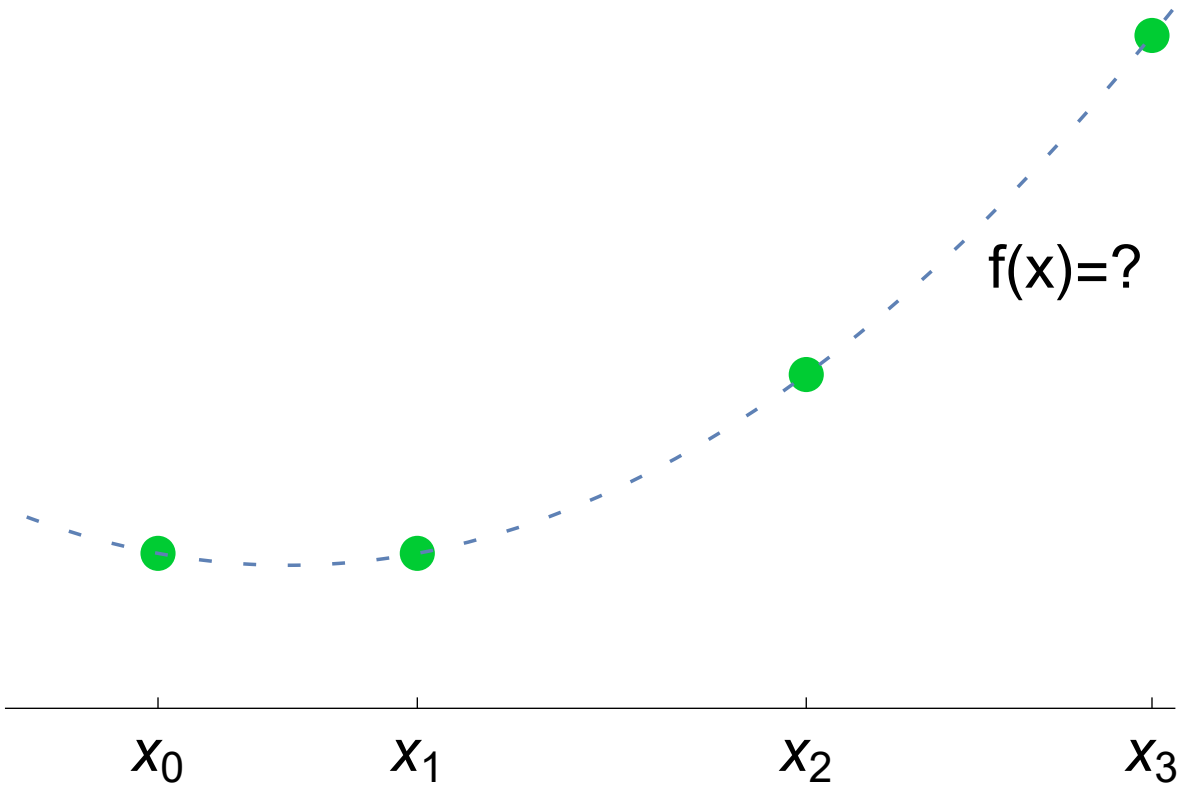
- Eine Funktion, die durch die gegebenen Datenpunkte verläuft.
- Ein Wert zwischen den Datenpunkten
- Trend über den gegebenen Datenbereich hinaus: Extrapolation

Anwendung:

Zwischenwerte in Tabellen, „glatte“ Kurven für Graphik...

Interpolation

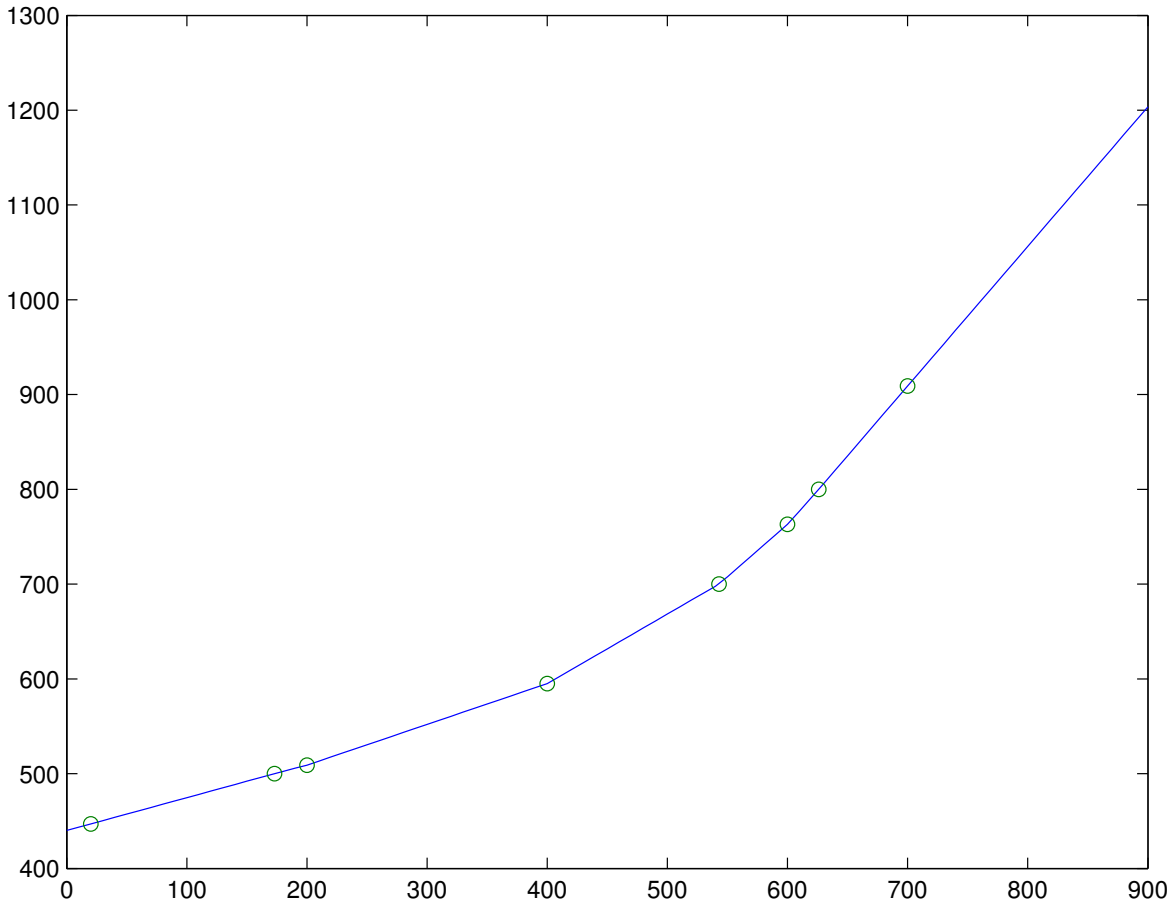
Gegeben: Datenpunkte. An Stützstellen x_i liegen Werte y_i vor.
Gesucht: Funktion f mit $f(x_i) = y_i$



Beispiel: Interpolation in Tabellen

Spezifische Wärmekapazität von kohlenstoffarmem Stahl in $\mathrm{J/kg\,K}$ für $20\,\mathrm{C} \leq T \leq 700, \mathrm{C}$

T	c_p
20	447
173	500
200	509
400	595
543	700
600	763
626	800
700	909



Die Abbildung illustriert stückweise lineare Interpolation zwischen den Stützstellen und Extrapolation bis 900 C.

Polynomiale Interpolation

Die einfachsten Interpolations-Funktionen sind Polynome...

Durch zwei Punkte der xy-Ebene geht genau eine Gerade. Durch drei beliebige Punkte lässt sich eindeutig eine Parabel legen. Durch $n+1$ Punkte ist ein Polynom n-ten Grades eindeutig bestimmt. (Ausnahme, wenn x-Werte zusammenfallen)

Aufgabenstellung:

- gegeben $n+1$ Wertepaare $(x_i, y_i),\ i = 0, \ldots, n$, wobei die x_i paarweise verschieden sind.
- gesucht ist das eindeutig bestimmte Polynom n-ten Grades p, das durch die gegebenen Datenpunkte verläuft:

$$p(x_i) = y_i \quad \text{für } i = 0, \ldots, n$$

.

Rechenverfahren zur polynomialen Interpolation

- Direkter Ansatz in der Standard-Form, Gleichungssystem mit Vandermonde-Matrix. Mehr Rechenaufwand als bei den folgenden Methoden.
- Lagrangesches Interpolationspolynom: Eine Formel, die das Polynom direkt hinschreibt.
- Newtonsches Interpolationspolynom: besonders rechengünstig.
- Es gibt noch einige andere Rechenschemen (im Skript: Neville-Verfahren; wir lassen es heuer aus)

Trotz unterschiedlicher Namen und Schreibweisen liefern alle Verfahren dasselbe (eindeutig bestimmte) Polynom.

Ansatz in der Standard-Darstellung

Das Interpolationspolynom in der Standard-Form

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

Gesucht sind die Koeffizienten $a_0, \ldots, a_n$.

Die Gleichungen $p(x_i) = y_i$ ergeben ein Gleichungssystem mit Vandermonde-Matrix

$$\begin{bmatrix} 1 & x_0 & x_0^2 & x_0^3 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & x_1^3 & \ldots & x_1^n \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & x_n^3 & \ldots & x_n^n \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Die gute Nachricht: Falls alle x-Werte verschieden sind, ist das Gleichungssystem eindeutig lösbar.
Die schlechte Nachricht: Diese Matrix kann sehr hohe Konditionszahl haben. Bei Polynomen höheren Grades entstehen gravierende Rundungsfehler. MATLAB's Befehl `polyfit` verwendet diesen Ansatz (mit all seinen Vor- und Nachteilen).

Lagrangesche Interpolationsformel

Das Interpolationspolynom durch die $n+1$ Wertepaare (x_i, y_i), $i = 0, \ldots, n$ ist gegeben durch

$$p(x) = L_0(x)y_0 + L_1(x)y_1 + \ldots + L_n(x)y_n,$$

wobei

$$L_i(x) = \frac{(x - x_0)(x - x_1)\ldots(x - x_{i-1})(x - x_{i+1})\ldots(x - x_n)}{(x_i - x_0)(x_i - x_1)\ldots(x_i - x_{i-1})(x_i - x_{i+1})\ldots(x_i - x_n)}$$

Es ist für die rechnerische Durchführung nicht ratsam, nach Einsetzen der Datenpunkte die $L_i(x)$ durch symbolisches Ausmultiplizieren noch weiter zu „vereinfachen“. Die x-Werte direkt einsetzen!

Polynome in verschiedenen Basis-Darstellungen

Ein Polynom $p(x)$ lässt sich auf unterschiedliche Art als Summe von Termen der Form *Koeffizient mal Basisfunktion* anschreiben.

Darstellungen

- Standardbasis $p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$
  p ist Linearkombination der Basis-Polynome $1, x, x^2, \ldots, x^n$
- Lagrange-Basis $p(x) = y_0 L_0(x) + y_1 L_1(x) + \cdots + y_n L_n(x)$
  p ist Linearkombination der Lagrange-Polynome $L_0, L_1, \ldots, L_n$
- Newton-Basis
  $p(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots + c_n(x - x_0)\cdots(x - x_{n-1})$
  p ist Linearkombination der Basis-Polynome $1, (x - x_0)$,
  $(x - x_0)(x - x_1)$, $(x - x_0)(x - x_1)(x - x_2), \ldots,$
  $(x - x_0)(x - x_1)\cdots(x - x_{n-1})$

Newtons Interpolations-Algorithmus

- Ansatz mit *Newton-Basisfunktionen*
- Das *Schema der dividierten Differenzen* berechnet mit wenig Aufwand die Koeffizienten
- Auswertung des Polynoms effizient mit *Horner-Schema*.

Newtons Interpolations-Algorithmus

Ansatz mit *Newton-Basisfunktionen*

$$p(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots + c_n(x - x_0)\cdots(x - x_n)$$

Gesucht sind die Koeffizienten $c_0, \ldots, c_n$.

Die Gleichungen $p(x_i) = y_i$ ergeben ein Gleichungssystem mit unterer Dreiecksmatrix. Die Koeffizienten $c_0, \ldots, c_n$ lassen sich einfach berechnen.

$$\begin{bmatrix} 1 & & & & 0 \\ 1 & (x_1 - x_0) & & & \\ 1 & (x_2 - x_0) & (x_2 - x_0)(x_2 - x_1) & & \\ \vdots & \vdots & & \ddots & \\ 1 & (x_n - x_0) & \cdots & & \prod_{i=0}^{n-1}(x_n - x_i) \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ \\ c_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ \\ y_n \end{bmatrix}$$

Newtons Interpolations-Algorithmus

Beispiel: kubisches Polynom

Daten: Ansatz:

$$\begin{aligned} p(x) =& c_0 + c_1 \cdot (x - x_0) + \\ &+ c_2 \cdot (x - x_0)(x - x_1) + \\ &+ c_3 \cdot (x - x_0)(x - x_1)(x - x_2) \end{aligned}$$

Einsetzen der vier Wertepaare x_i, y_i gibt Gleichungssystem der Form

$$\begin{aligned} c_0 &= y_0 \\ c_0 + c_1(x_1 - x_0) &= y_1 \\ c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1) &= y_2 \\ c_0 + c_1(x_3 - x_0) + c_2(x_3 - x_0)(x_3 - x_1) + c_3(x_3 - x_0)(x_3 - x_1)(x_3 - x_2) &= y_3 \end{aligned}$$

Auflösen von oben nach unten:

$$c_0 = y_0, c_1 = \frac{y_1 - y_0}{x_1 - x_0}, \ldots$$

Newtons Interpolations-Algorithmus

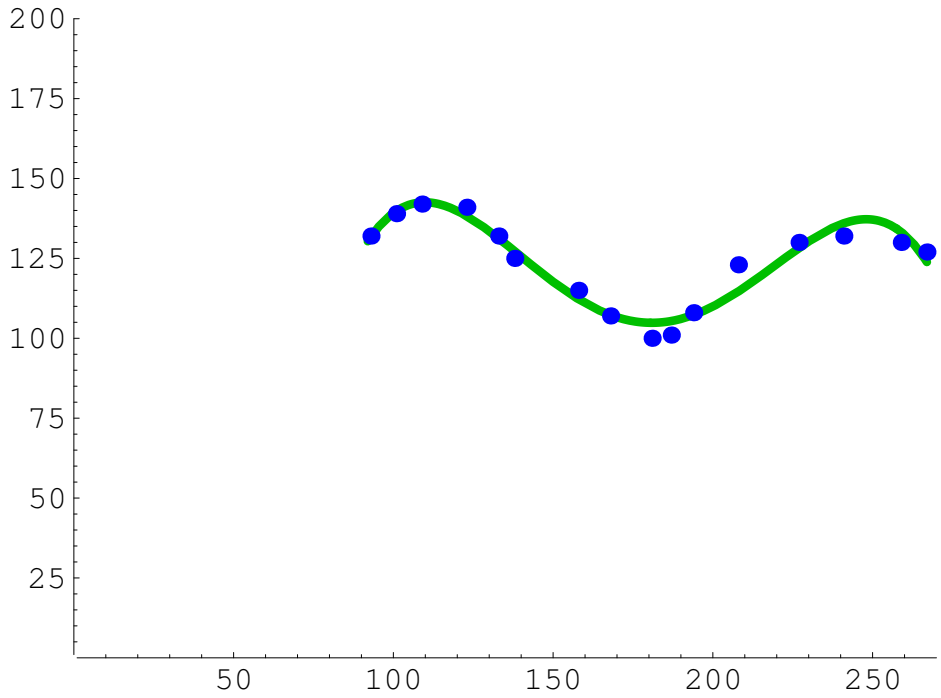
Das Schema der dividierten Differenzen ist ein optimierter Rechenablauf zur Lösung des Gleichungssystems mit unterer Dreiecksmatrix

Mit Papier und Stift organisiert man die Rechnung am besten in Tabellenform nach folgendem Schema

$$
\begin{array}{cccccccc}
 & & & x_0 & y_0 & & & \\
 & & x_1 - x_0 & & & [x_0, x_1] & & \\
 & x_2 - x_0 & & x_1 & y_1 & & [x_0, x_1, x_2] & \\
 & & x_2 - x_1 & & & [x_1, x_2] & & \\
\ldots & x_3 - x_1 & & x_2 & y_2 & & [x_1, x_2, x_3] & \ldots \\
 & & x_3 - x_2 & & & [x_2, x_3] & & \\
 & x_4 - x_2 & & x_3 & y_3 & & [x_2, x_3, x_4] & \\
 & & x_4 - x_3 & & & [x_3, x_4] & & \\
 & & & x_4 & y_4 & & &
\end{array}
$$

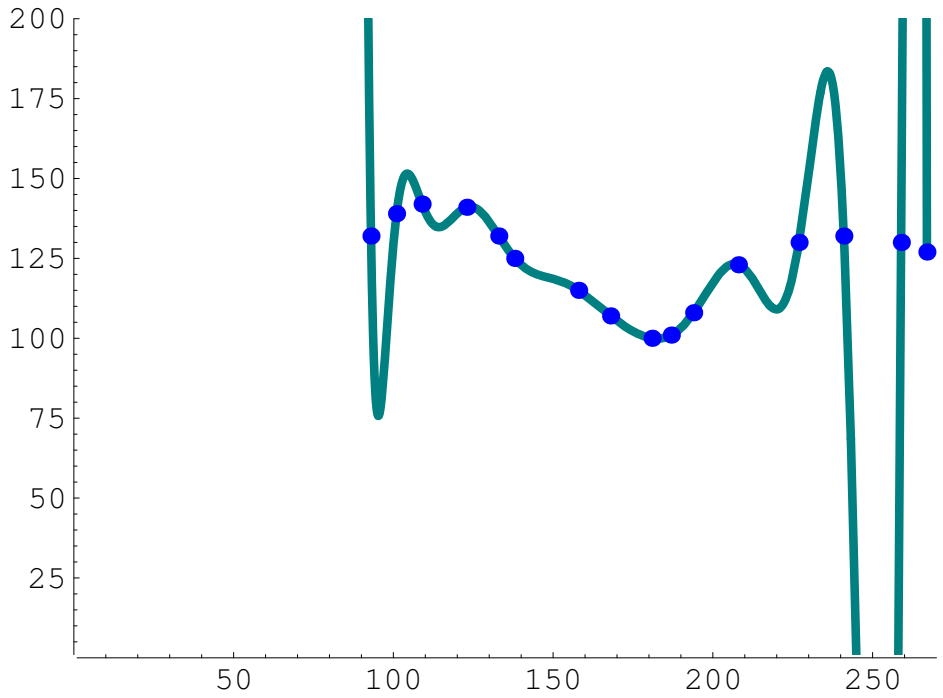
Beispiele siehe Skript!

Warnung vor zu hohem Grad!



16 Datenpunkte sind gegeben. Gesucht ist eine „schöne“ Kurve durch diese Punkte. Die hier gezeichnete Kurve approximiert, aber interpoliert nicht!
Ein Polynom 15. Grades könnte die Daten exakt modellieren, aber...

Warnung vor zu hohem Grad!



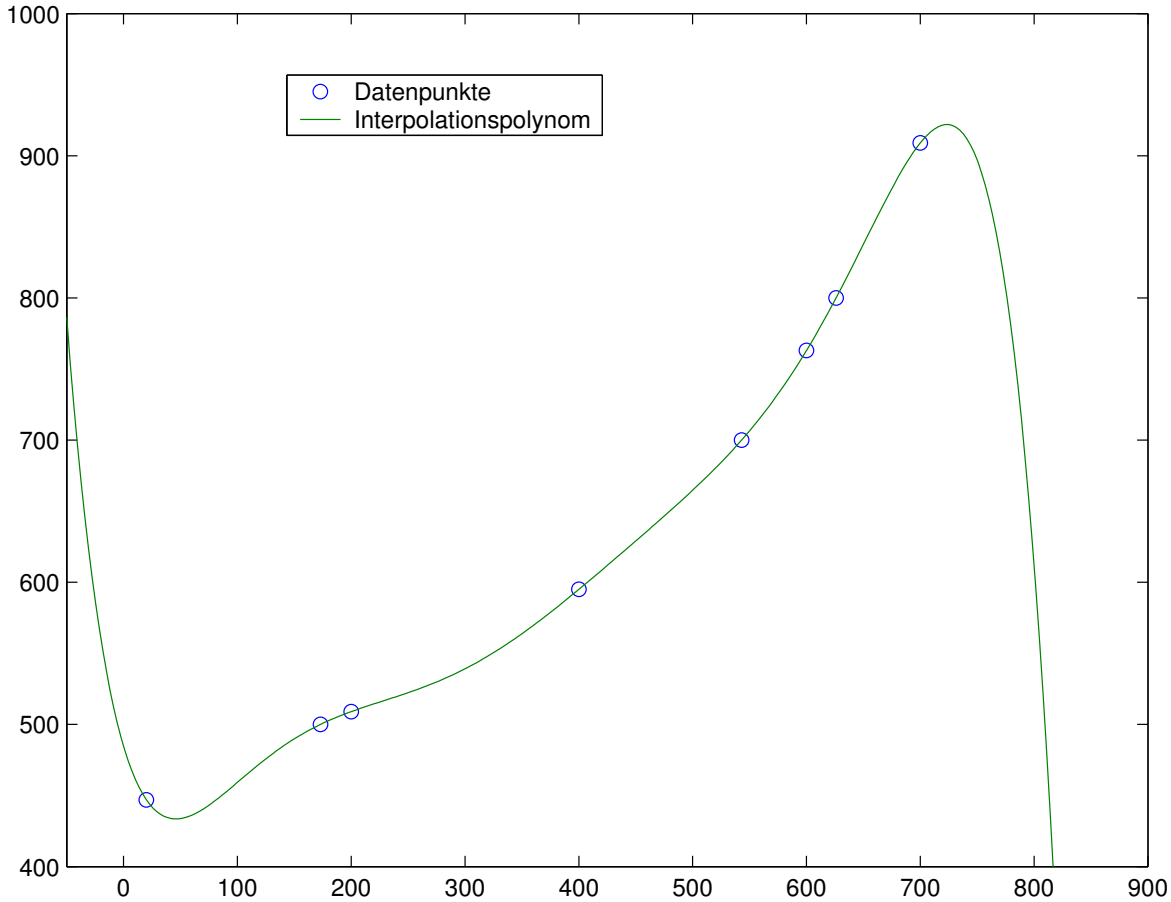
Kein Fehler an den Datenpunkten, aber dazwischen oszilliert das Polynom heftig. Typisch für Polynome hohen Grades. Sie oszillieren besonders zu den Rändern hin, wenn man Sie durch vorgegebene Datenpunkte zwingt.

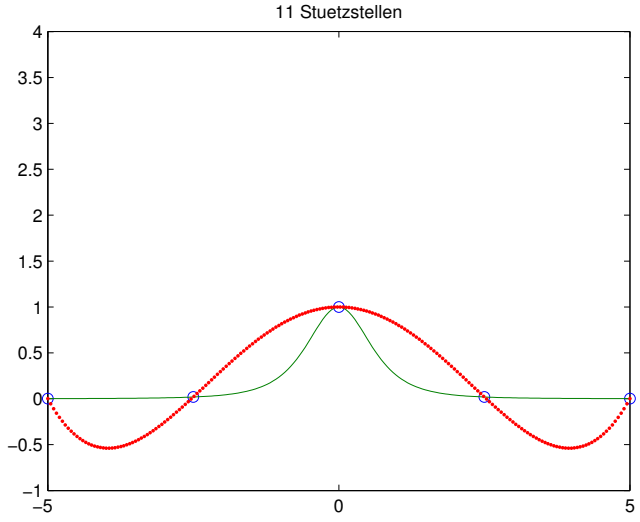
Nochmal: Warnung vor zu hohem Grad!

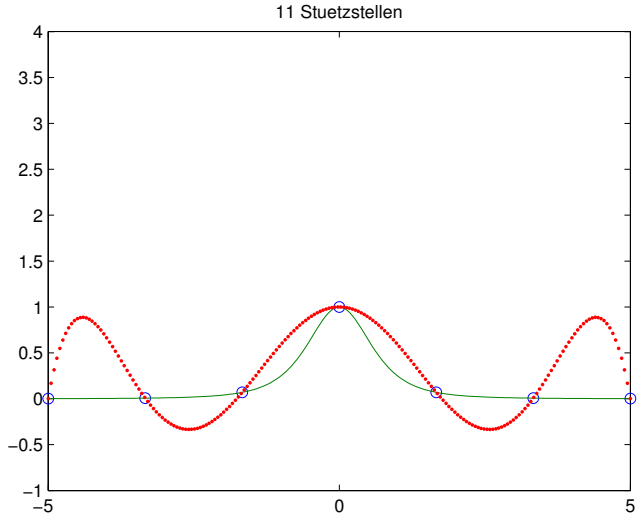
Beispiel von vorhin: spezifische Wärme c_p bei Temperatur T.

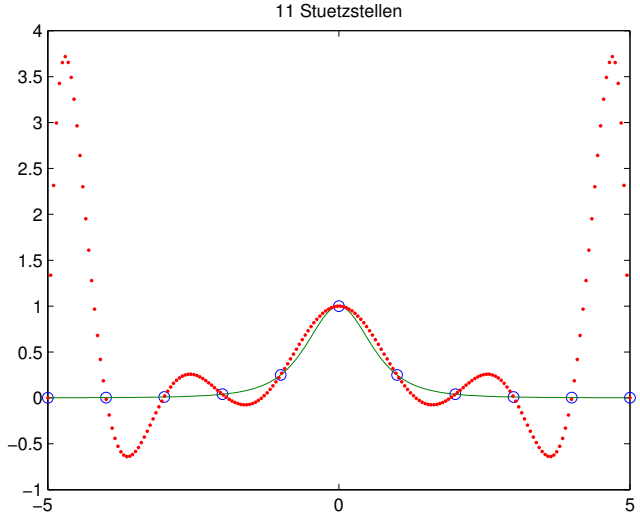
Durch die acht Datenpunkte lässt sich ein Polynom siebten Grades exakt durchlegen.

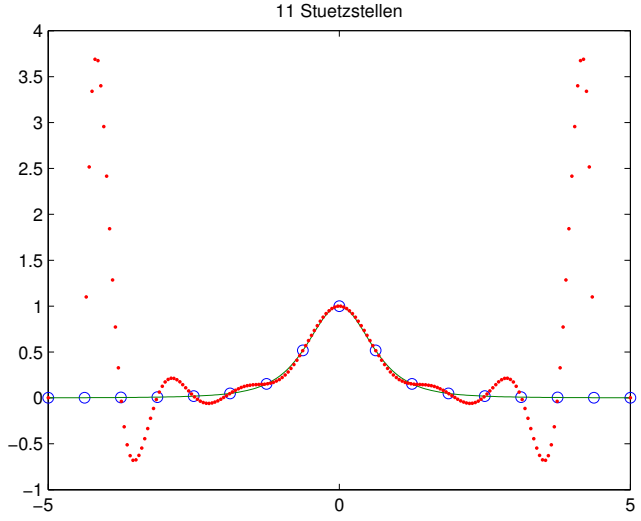
Aber:
Polynome so hohen Grades neigen zu Oszillationen und zu extrem unrealistischer Extrapolation











Interpolationspolynome hohen Grades sind ungeeignet!

Approximation der Funktion $f(x) = \frac{1}{1+x^2}$, $x \in [-5, 5]$ mit 3, 7, 11 und 17 Stützstellen:

„Interpolationspolynome hohen Grades sind ungeeignet“

Vorsicht, Übertreibung!

Die vorhergehenden Folien gebrauchen das rhetorische Stilmittel der Übertreibung. Es gilt nämlich

Approximationssatz von Weierstraß

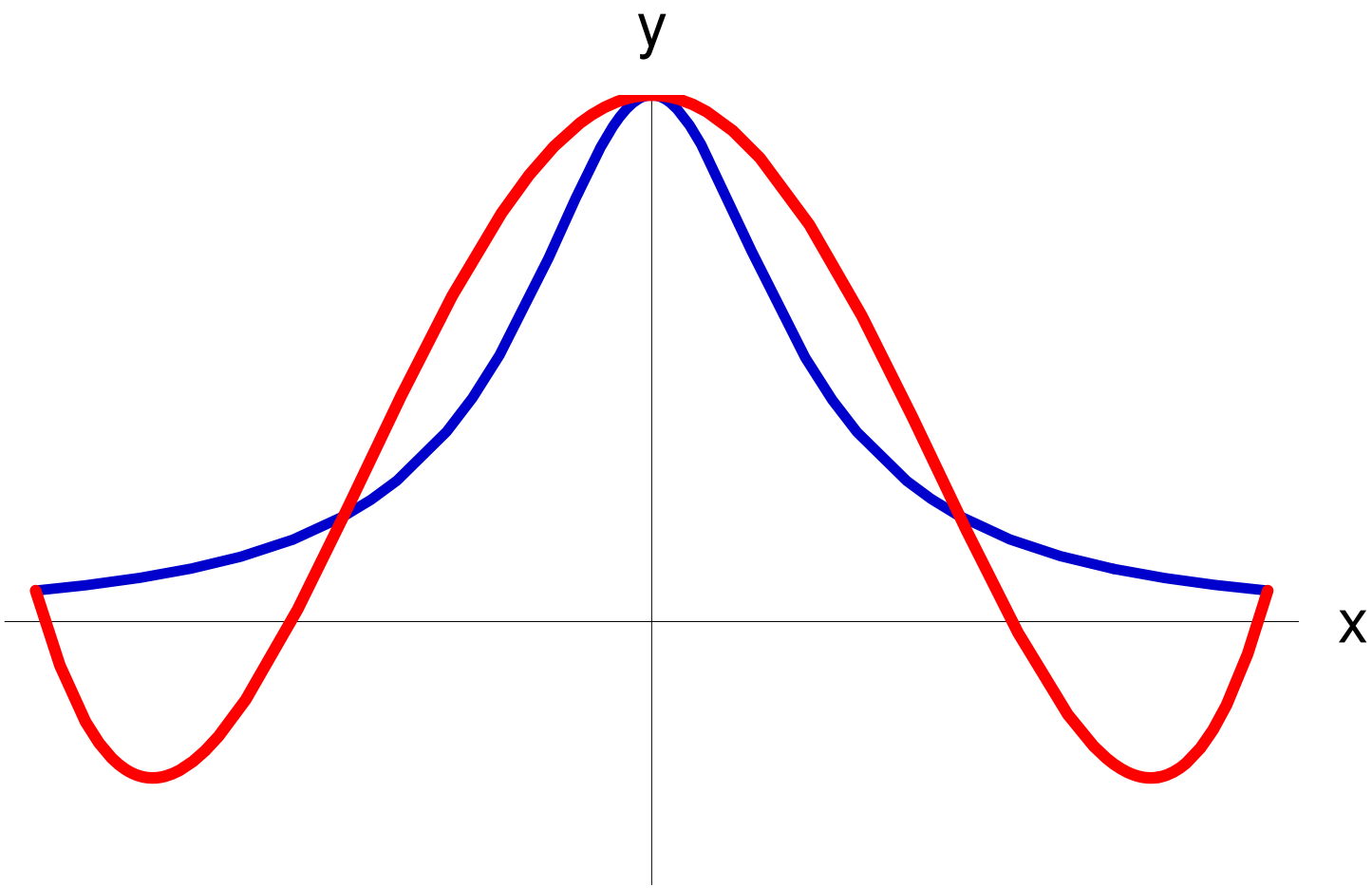
Jede stetige Funktion lässt sich auf einem abgeschlossenen Intervall beliebig genau durch Polynome approximieren.

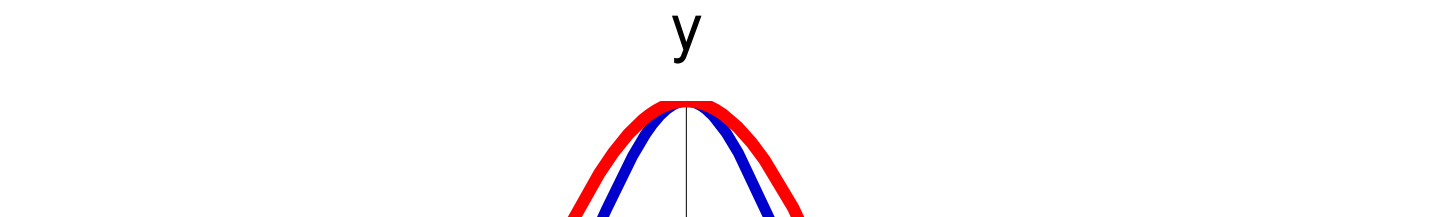
Die Probleme treten bei Interpolation auf, wenn das Polynom durch *fix vorgegebene* Stützstellen gehen soll. Es gibt aber speziell verteilte *Tschebyschow-Stützstellen*, für die Interpolationspolynome besonders günstige Eigenschaften haben.

Runge-Phänomen

Interpolationspolynome mit äquidistenten Stützstellen konvergieren nicht unbedingt

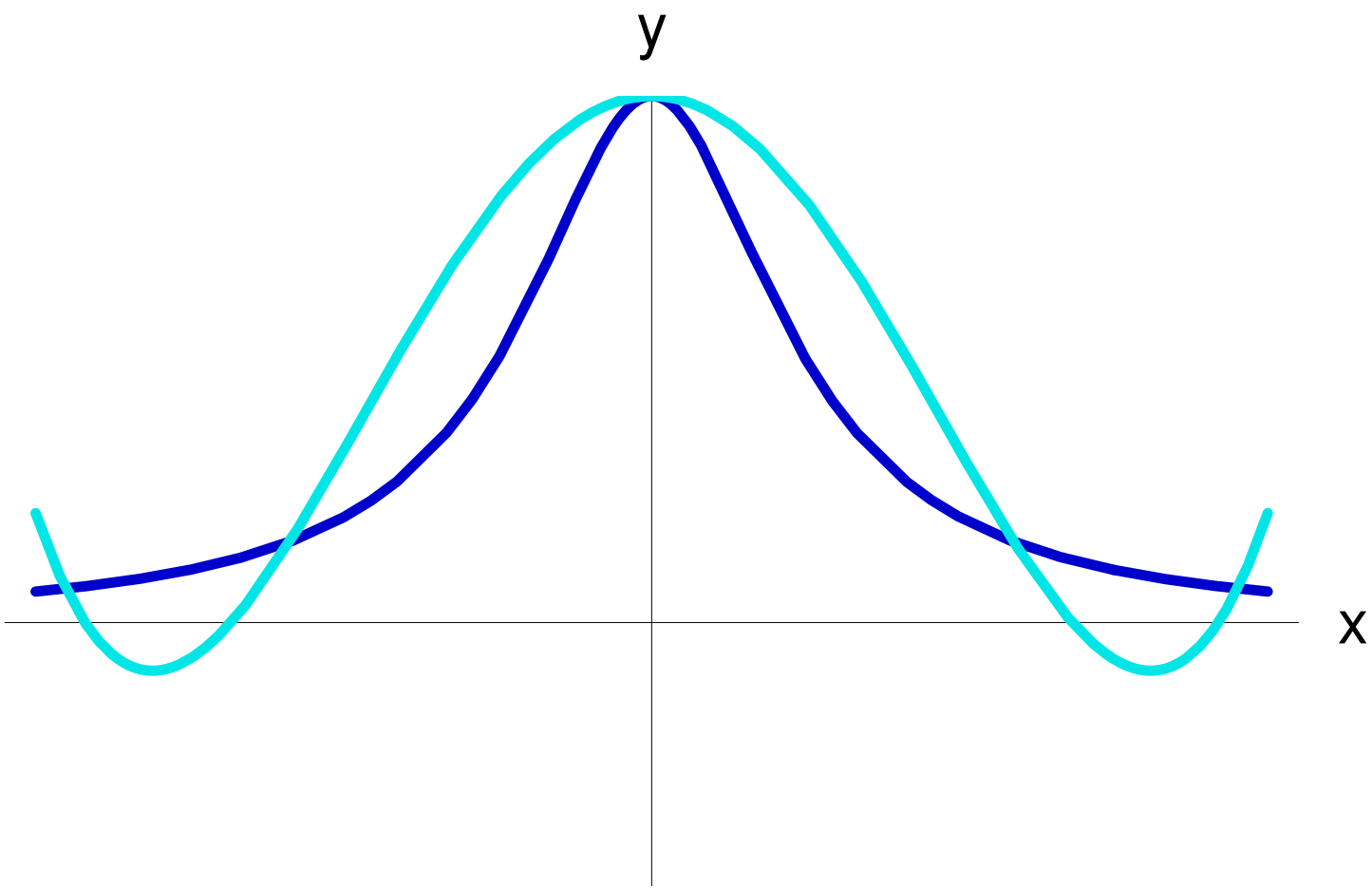
Noch einmal am Beispiel von $f(x) = \frac{1}{1+x^2}$

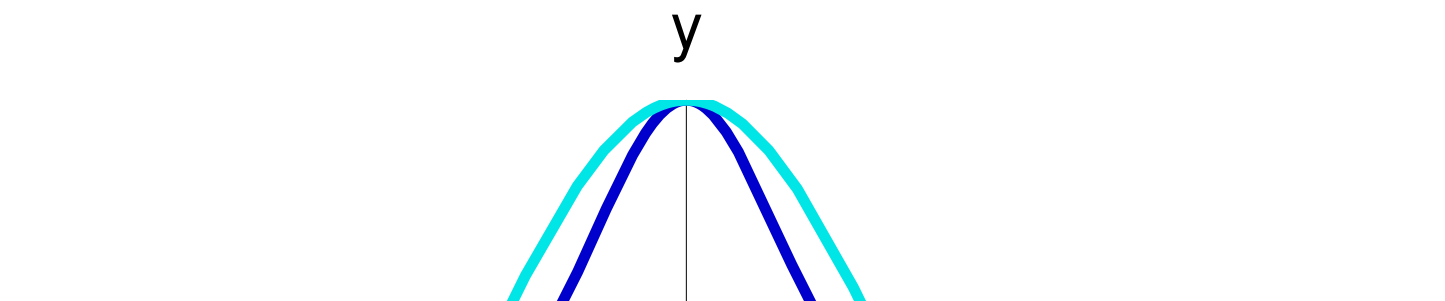




Tschebyschow-Stützstellen

Nicht äquidistant, sondern zu den Rändern hin dichter

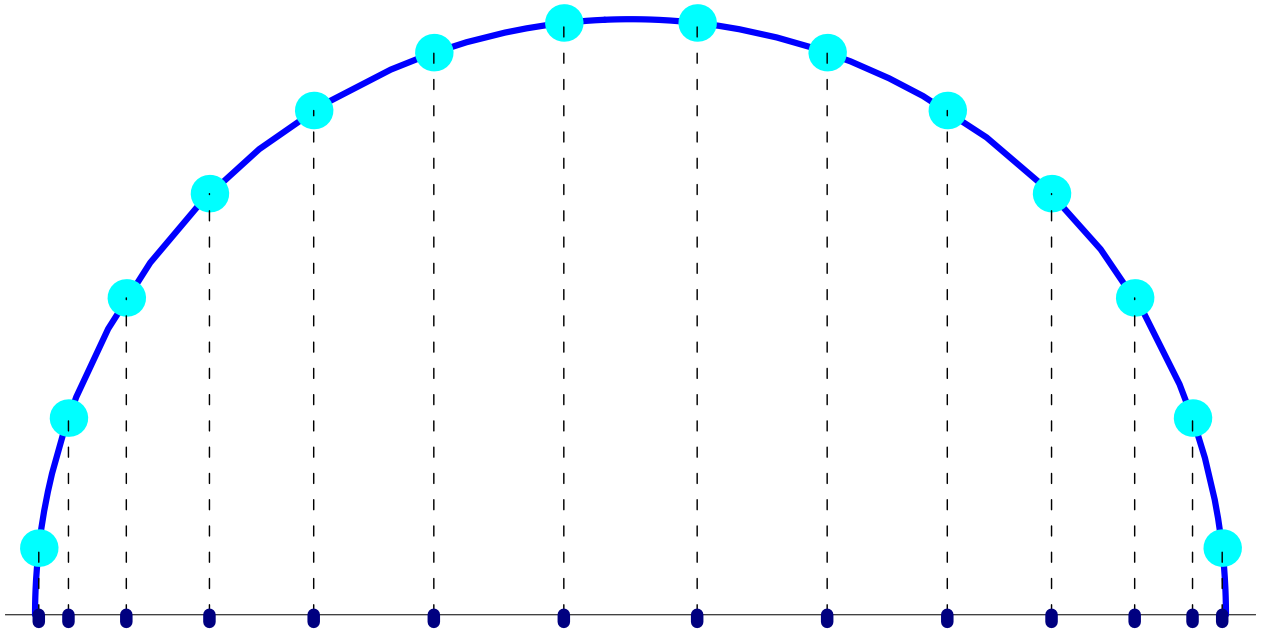




Optimale Lage der Stützstellen

Projektion von n gleichmäßig am Halbkreis verteilten Punkten

Bei dieser Wahl der Stützstellen („Tschebyschow-Stützstellen“) ist die maximale Abweichung Funktion–Polynom am kleinsten



Lage auf Intervall $[-1;1]$

$$x_i = \cos\left(\frac{2i-1}{2n}\pi\right) \qquad \text{für } i = 1,\ldots,n$$

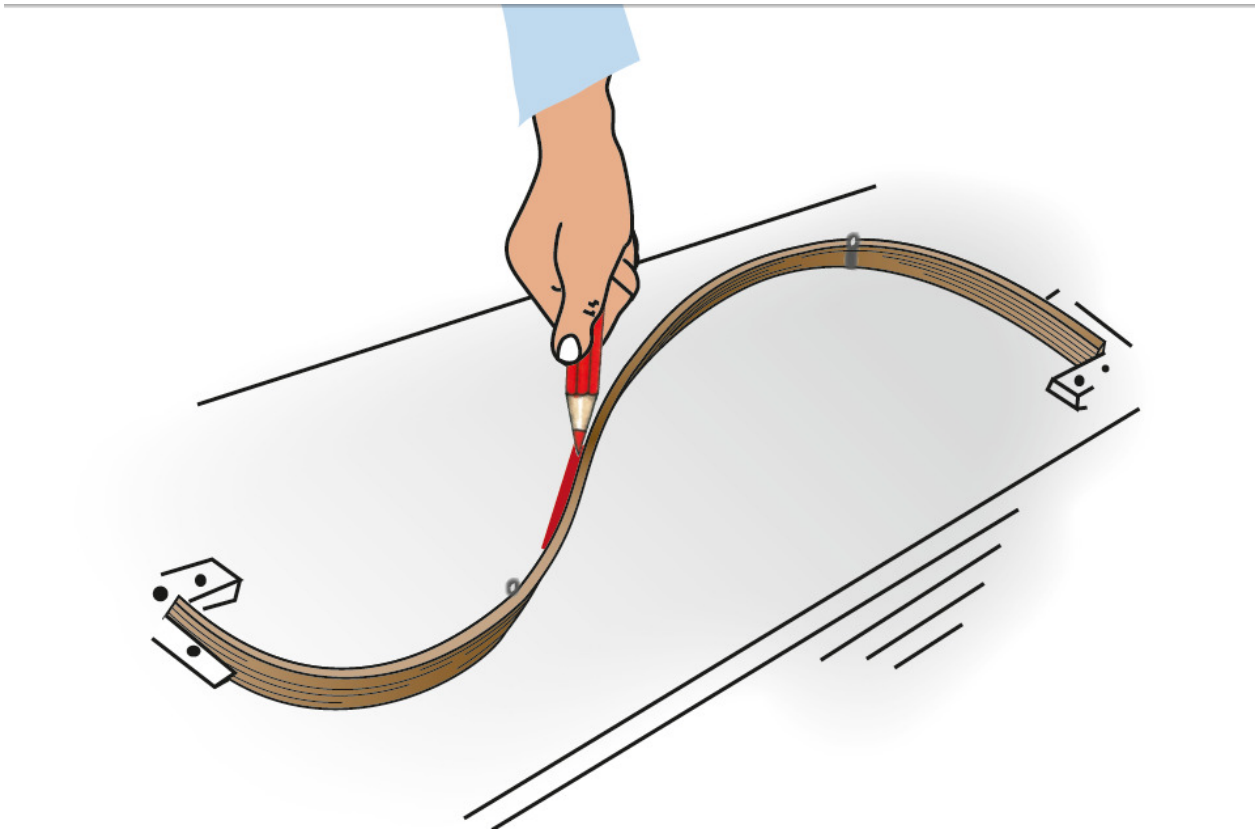
Weitere Interpolationsverfahren

Klassische Newton- oder Lagrange-Interpolation ist nicht immer optimal

Andere wichtige Verfahren sind

- Spline-Interpolation (Kubisch, MATLAB pchip,...)
- Rationale Interpolation
- Trigonometrische Interpolation
- Interpolation in zwei oder mehr Dimensionen

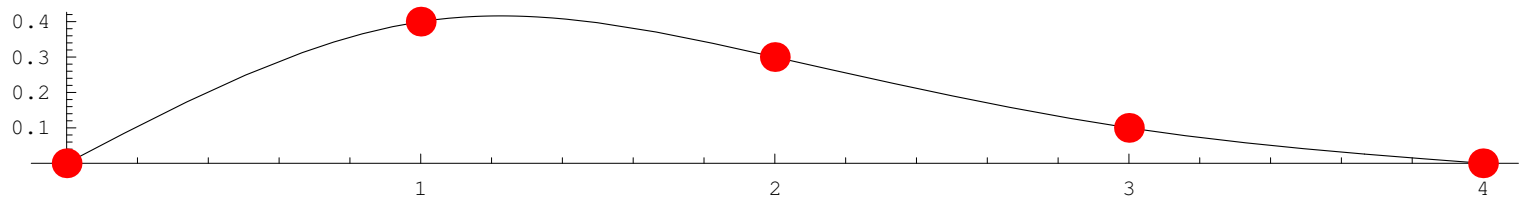
Biegsame Latte als Kurvenlineal



Ursprünglich im Schiffsbau verwendet, heißt Straklatte, englisch Spline.
Funktionen, die das Verhalten biegsamer Latten nachbilden: Natürliche kubische Spline-Funktionen.

Natürlicher kubischer Spline

Eine dünne, an einzelnen Punkten festgehaltene Latte biegt sich in der Form eines kubischen Splines



Natürlicher kubischer Spline

Ein natürlicher kubischer Spline $s(x)$ durch die $n+1$ Wertepaare (x_i, y_i), $i = 0, \ldots, n$ ist folgendermaßen charakterisiert:

- In den einzelnen Intervallen (x_{i-1}, x_i) ist $s(x)$ jeweils ein kubisches Polynom
- An den Intervallgrenzen stimmen die Funktionswerte, die ersten und die zweiten Ableitungen rechts- und linksseitig überein.
- Zweite Ableitung an den Rändern wird Null gesetzt.

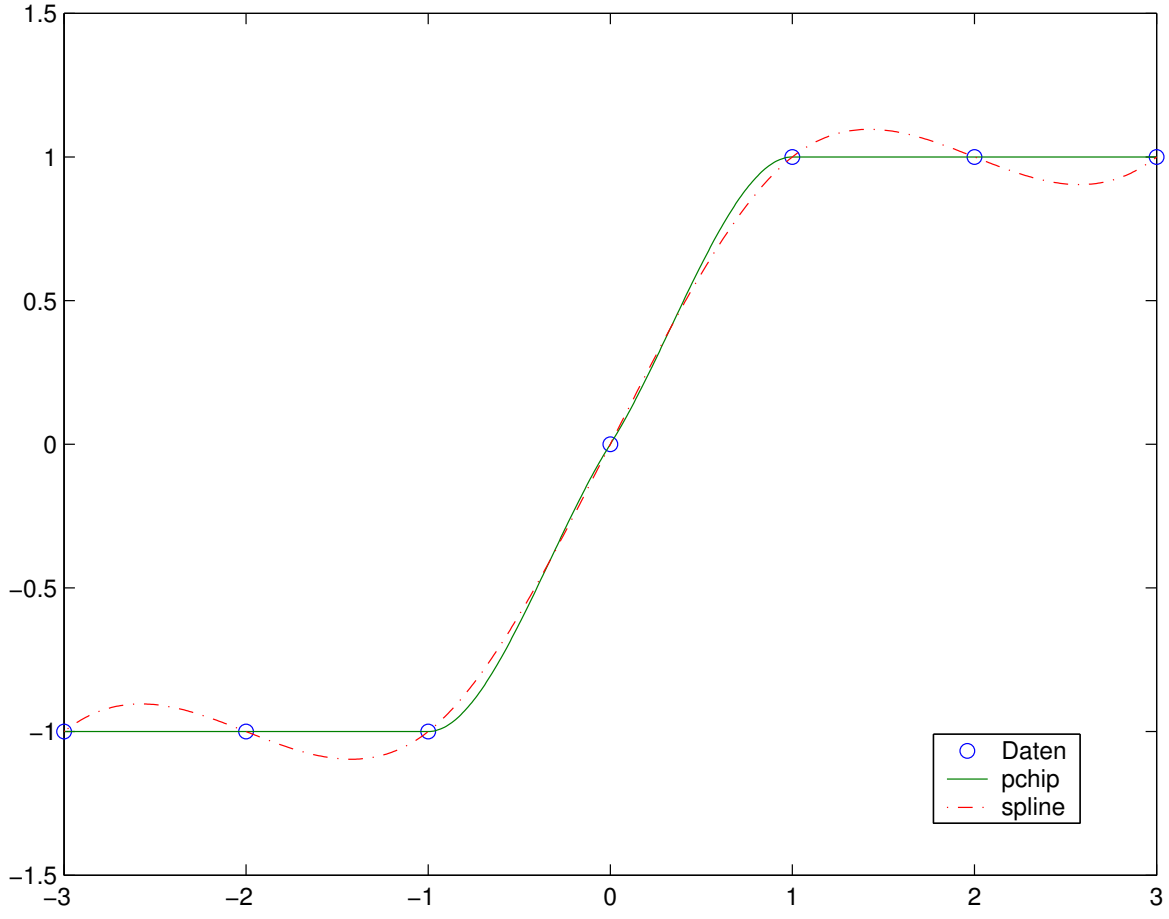
Andere kubische Splines

Ein Spline ist stückweise aus einzelnen Polynomen zusammengesetzt. Je nach Spline-Typ werden weitere Bedingungen gewählt. MATLAB's `pchip` erfüllt:

- In den einzelnen Intervallen (x_{i-1}, x_i) ist $s(x)$ jeweils ein kubisches Polynom
- An den Intervallgrenzen stimmen die Funktionswerte und die ersten Ableitungen rechts- und linksseitig überein. Die zweiten Ableitungen können unterschiedlich sein.
- Der Spline respektiert das Monotonieverhalten der Datenpunkte – kein Überschwingen

Interpolation in Matlab: spline und pchip

```
x = -3:3;
y = [-1 -1 -1 0 1 1 1];
t = -3:.01:3;
p = pchip(x,y,t);
s = spline(x,y,t);
plot(x,y,'o',t,p,'-',
     t,s,'-.')
```



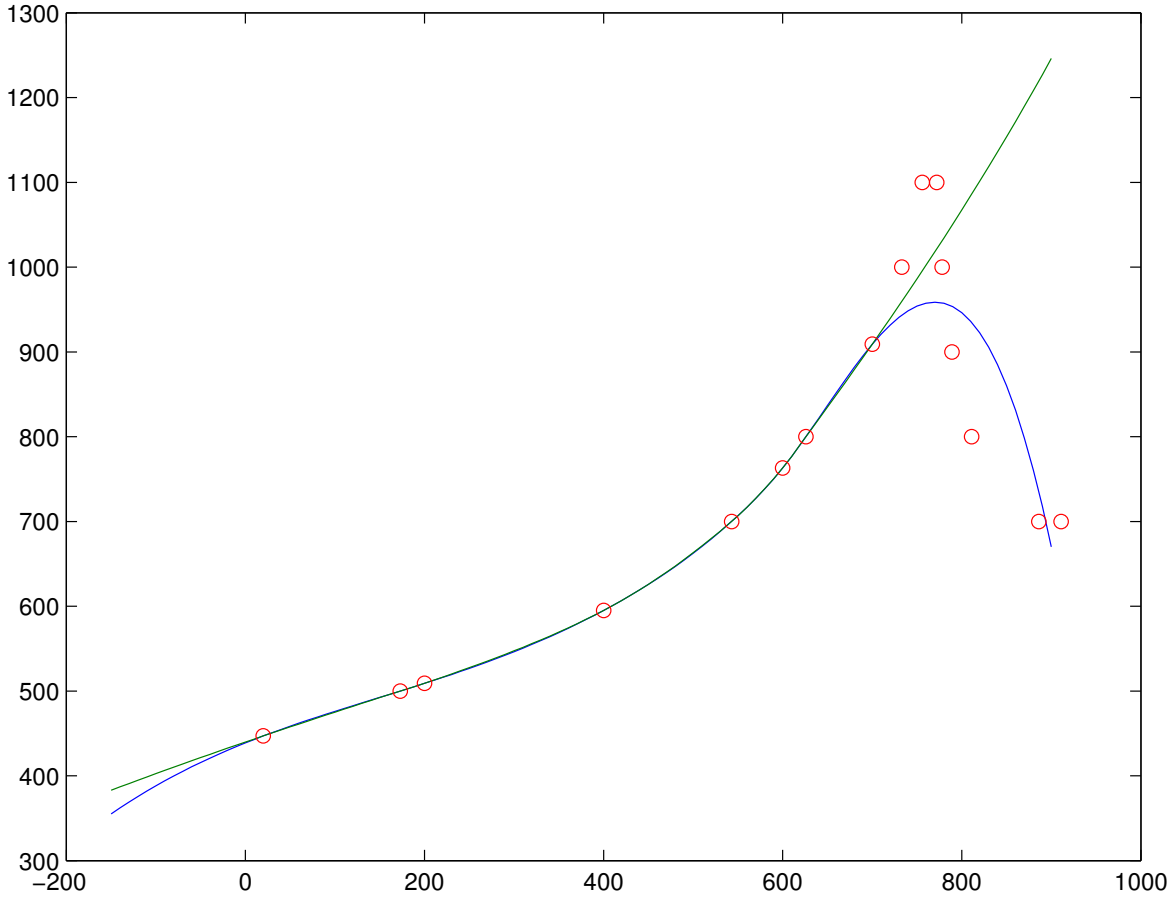
MATLAB bietet verschiedene stückweise kubische Interpolationsverfahren.
`spline` ist für glatte Daten genauer.
`pchip` überschwingt nicht und neigt weniger zu Oszillationen.

Beispiel: c_p-Daten mit `spline` und `pchip`

Innerhalb des Datenbereiches stimmen beide Verfahren sichtlich überein.

Extrapolation ist ein wesentlich riskanteres Geschäft. . .

. . . wie man sieht: hier sind weitere Datenpunkte eingetragen. Keine der beiden Methoden extrapoliert den tatsächlichen Verlauf der spez. Wärme korrekt.



Interpolationsverfahren

Übersicht und Zusammenfassung

- Newton-Polynom: rechengüstig; zusätzliche Datenpunkte lassen sich leicht anfügen
- Lagrange-Polynom: y-Werte lasen sich leicht ändern
- Polynome mit hohem Grad und äquidistanten Stützstellen: Ungeeignet! (Runge-Phänomen).
- Tschebyschow-Stützstellen: optimale Approximation
- Natürliche kubische Splines: meist bessere Anpassung als Polynome.
- Es gibt viele weitere Spline-Varianten

Numerische Integration

Gegeben:

eine Funktion $f(x)$ in einem Intervall $a \leq x \leq b$.

Gesucht:

deren Integral $\int_a^b f(x)dx$

Oft lässt sich das Integral nicht durch elementare Funktionen ausdrücken, oder die Funktion selbst ist nur tabellarisch gegeben. $\longrightarrow$ Numerische Verfahren

Integrationsformeln nach Newton-Cotes

Gegeben:

Eine Funktion $f(x)$ in einem Intervall (a, b) durch ihre Werte f_i an $n+1$ äquidistanten Stützstellen,

$$f_i = f(a + ih), \quad \text{mit } h = \frac{b-a}{n}, \quad i = 0, \ldots, n.$$

Prinzip:

Interpoliere $f(x)$ durch ein Polynom $p(x)$. Nähere das Integral von f durch das Integral von p. Die Näherung ist gegeben als gewichtete Summe der f_i,

$$\int_a^b f(x)dx \approx (b-a) \sum_{i=0}^{n} \alpha_i f_i$$

mit fixen Gewichten α_i

Newton-Cotes-Formeln

Klassische Beispiele — mit Fehlertermen

Trapezregel

$$\int_a^b f(x)dx = \frac{b-a}{2}\left(f(a)+f(b)\right) \qquad -\frac{(b-a)^3}{12}f''(\xi)$$

Simpson-Regel

$$\int_a^b f(x)dx = \frac{b-a}{6}\left(f(a)+4f(\frac{a+b}{2})+f(b)\right) \qquad -\frac{(b-a)^5}{2880}f^{IV}(\xi)$$

3/8-Regel

$$\int_a^b f(x)dx = \frac{b-a}{8}\left(f(a)+3f(a+h)+3f(a+2h)+f(b)\right) \qquad -\frac{(b-a)^5}{6480}f^{IV}(\xi)$$

Newton-Cotes-Formeln

Herleitung: Simpson-Regel integriert Parabel durch Datenpunkte

Simpson-Regel

$$\int_a^b f(x)dx = \frac{b-a}{6}\left(f(a) + 4f(\frac{a+b}{2}) + f(b)\right)$$

1. Interpoliere Polynom durch (a, f_0), $(\frac{a+b}{2}, f_1)$, (b, f_2) z. B. nach Newton:

$$p(x) = f_0 + (x-a)\frac{2(f_0-f_1)}{a-b} + (x-a)(x-\frac{a+b}{2})\frac{2(f_0-2f_1+f_2)}{(a-b)^2}$$

2. Integrieren des Polynoms von a bis b ergibt $\frac{1}{6}(b-a)(f_0+4f_1+f_2)$

Newton-Cotes-Formeln

Klassische Beispiele — mit Fehlertermen

3/8-Regel

Cotes: „pulcherrima et utilissima“, sehr schön und nützlich

$$\int_a^b f(x)dx = \frac{b-a}{8}\left(f(a) + 3f(a+h) + 3f(a+2h) + f(b)\right) \qquad -\frac{(b-a)^5}{6480} f^{IV}(\xi)$$

Idee: Wie vorher, nur betrachtet man ein Polynom über vier Stützstellen!

Newton-Cotes-Formeln, allgemeines Prinzip

Alle Integrationsformeln haben die Gestalt

$$\text{Basislänge} \times \text{mittlere Höhe der Funktion}$$

Zum Beispiel bei der 3/8-Regel: Der Term

$$\left(\frac{1}{8}f(a) + \frac{3}{8}f(a+h) + \frac{3}{8}f(a+2h) + \frac{1}{8}f(b)\right)$$

ist ein gewichtetes Mittel der Funktionswerte.

Zusammengesetzte N.-C.-Formeln

Wenn feinere Intervallteilung vorliegt

Achtung bei Intervallbreite h! Es sind $n+1$ Datenpunkte, und um eins weniger Intervalle. n= Anzahl Datenpunkte-1, und $h = (b-a)/n$

Zusammengesetzte Trapezregel

$$\int_a^b f(x)dx = \frac{h}{2}(f_0 + 2f_1 + 2f_2 + \cdots + 2f_{n-1} + f_n) + E$$

Zusammengesetzte Simpson-Regel

$$\int_a^b f(x)dx = \frac{h}{3}(f_0 + 4f_1 + 2f_2 + 4f_3 + \cdots + 2f_{n-2} + 4f_{n-1} + f_n) + E$$

(Nur für gerades n möglich!)

Weitere Quadraturformeln

Gauß-Quadratur, Gauß-Lobatto-Formeln

Nicht äquidistante Stützstellen, dafür höhere Genauigkeit. (Typische Anwendung: Finite Elemente)

Romberg-Verfahren

berechnet mit zusammengesetzter Trapezregel mehrere Werte zu verschiedenen h und extrapoliert auf $h = 0$.

MATLAB

bietet zwei Verfahren: `quad` (adaptive Simpson-Regel) und `quadl` (trickreichere Gauß-Lobatto Methode)