4.1.8. Weitere Interpolationsansätze

(a) Hermite-Interpolation:

Um Oszillationen zu vermeiden will man auch die Ableitungen an den Stützstellen kontrollieren, im einfachsten Fall genau die ersten Ableitungen:

Vorgegeben sind (x_j , y_j , y'_j) , $j=0,...,n$

Gesucht ein Polynom vom Grad $2n+1$ mit

$$p(x_j) = y_j \quad \text{und} \quad p'(x_j) = y'_j \text{ , } j=0,1,...,n$$

(1) Lösung durch Aufstellen des dadurch gegebenen linearen $2n+2 \times 2n+2$ Gleichungssystems für die gesuchten *Koeffizienten des Polynoms:*

$$\begin{pmatrix} 1 & x_0 & \cdots & x_0^{2n+1} \\ 0 & 1 & \cdots & (2n+1)x_0^{2n} \\ 1 & x_1 & \cdots & x_1^{2n+1} \\ \vdots & \vdots & & \vdots \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{2n+1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_0' \\ y_1 \\ \vdots \end{pmatrix}$$

(2) Lösung durch ‚Lagrange'-Polynome, die genau an einer Stützstelle z.B. Wert 0 und Ableitung 1 haben, sonst an allen anderen Stützstellen Wert und Ableitung 0.

Setze dazu

$$L_j(x) := (x - x_0)^2 \cdots (x - x_{j-1})^2 (x - x_{j+1})^2 \cdots (x - x_n)^2$$

1. Aufgabe:

Finde Polynom vom Grad 2n+1, das

an der Stelle x_j den Wert 1 und Ableitung 0 hat, und

an allen anderen Stützstellen Wert und Ableitung 0:

Ansatz: $G_j(x) = L_j(x) \cdot (\alpha + \beta x)$

α und β sind daher so zu wählen, dass

$L_j(x_j)(\alpha + \beta x_j) = 1$ *gilt* *und* $L_j'(x_j)(\alpha + \beta x_j) + L_j(x_j)\beta = 0$

Dies liefert zwei Bedingungen für α und β → $G_j(x)$

2. Aufgabe:
Finde Polynom $H_j(x)$, das an der Stelle x_j den Wert 0 und Ableitung 1 hat, an allen anderen Stützstellen Wert und Ableitung 0.

Dann sind α und β so zu wählen, dass

$$L_j(x_j)(\alpha+\beta x_j)=0 \quad \textit{gilt} \quad \textit{und} \quad L_j'(x_j)(\alpha+\beta x_j)+L_j(x_j)\beta=1$$

Lösung liefert α und β → $H_j(x)$

Damit erhält man als Lösung der Hermite-Interpolation:

$$p(x)=\sum_{j=0}^{n}\left(y_j G_j(x)+y'_j H_j(x)\right)$$

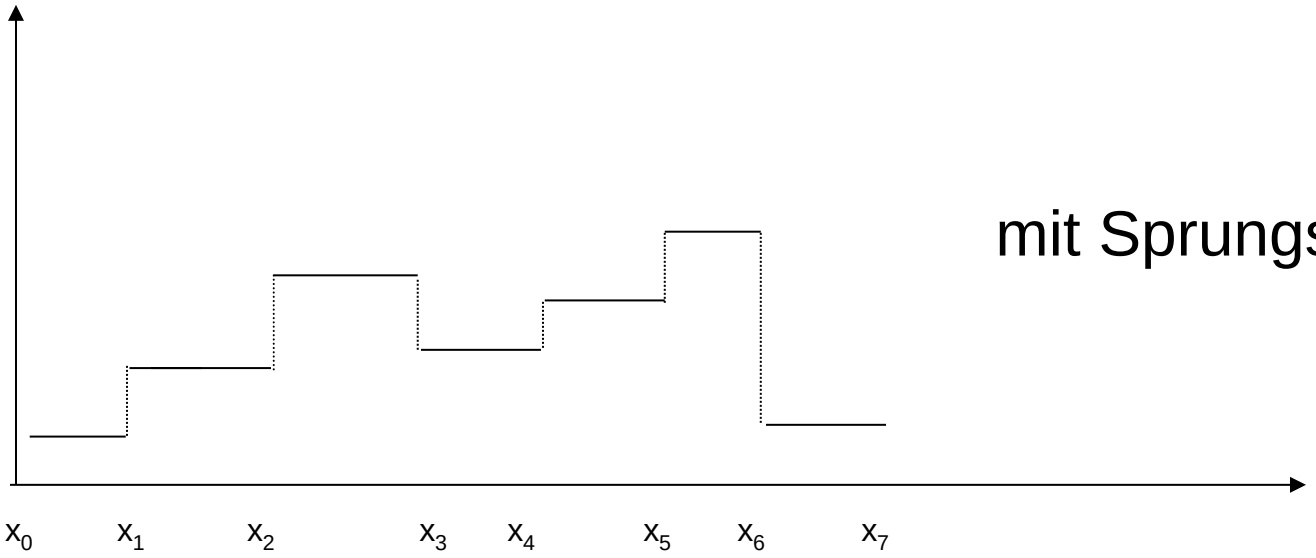
(b) Spline-Interpolation:

Versuche, die Oszillationen zu verhindern, indem man die interpolierende Funktion stückweise aus Polynomen niedrigen Grades zusammenbaut.

Definition einer Splinefunktion S vom Grade k zu Stützstellen x_0 ,..., x_n :

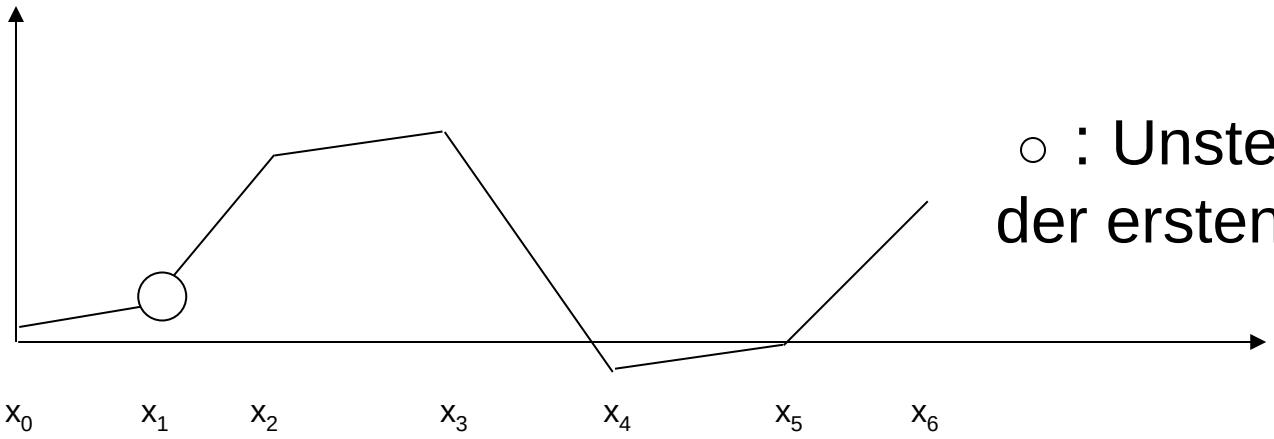
S(x) sei insgesamt (k-1)-mal stetig diff'bar und auf den einzelnen Teil-Intervallen $[x_i , x_{i+1}]$ jeweils ein Polynom k-ten Grades.

Für k=0 stückweise konstante Funktionen (Treppenfunktionen):



mit Sprungstellen

Für k=1 stetig, stückweise linear:



○ : Unstetigkeitsstelle der ersten Ableitung

Für k=2 stückweise quadratische Polynome, stetig diff'bar verbunden.

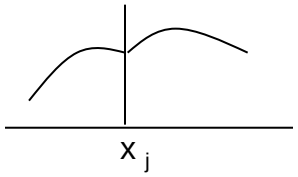
quadratisch	quadratisch	quadtratisch	quadratisch	.....

Für k=3 (kubische Splines) stückweise Polynome dritten Grades; an den Nahtstellen sind die Funktion samt erster und zweiter Ableitung stetig. Diese Splines werden am häufigsten verwendet.

Das Interpolationsproblem mit Splinefunktionen lautet dann: Bestimme Spline S(x) zu k so, dass $S(x_j)=y_j$, j=0,...,n

Dies führt auf lineares Gleichungssystem für die Parameter, die S(x) bestimmen, abhängig von k.

Neben den n+1 Interpolationsbedingungen gehen noch die Bedingungen an den Nahtstellen ein (stetig, stetig diff'bar,...), z.B. in der Form



$$p_{links}(x_j) = p_{rechts}(x_j)$$

$$p'_{links}(x_j) = p'_{rechts}(x_j) \quad \text{, usw.}$$

p sollen dabei Polynome k-ten Grades sein und S, S',... stetig bis einschließlich (k-1)-te Ableitung.

Bestimmt werden die Koeffizienten der Teilpolynome vom Grad k, so dass die Interpolationsbedingungen erfüllt sind und die Übergänge entsprechend stetig sind.

Anzahl Unbekannte: n Polynome vom Grad k: n(k+1) Koeff.

Anzahl Bedingungen: n+1 Interpol.-Bedingungen $S(x_j)=y_j$

und (n-1)k stetige Übergänge $p_j^{(i)}(x_j) = p_{j+1}^{(i)}(x_j)$

für i=0,…,k-1 an den inneren Stellen j=1,…,n-1

Insgesamt also nk+n-k+1=n(k+1)+(1-k) Bedingungen.

Füge noch k-1 neue Bedingungen hinzu, um ein quadratisches lineares Gleichungssystem mit genauso vielen Gleichungen wie Unbekannten zu erhalten,

z.B. - Bedingungen über Ableitungen an den Endpunkten,
oder - Periodizität, o.ä.
also $S''(a)=S''(b)=0$, oder $S^j(a)=S^j(b)$.

Dann ergibt sich dann eine n(k+1) x n(k+1) - Matrix.

Geschickter Ansatz:

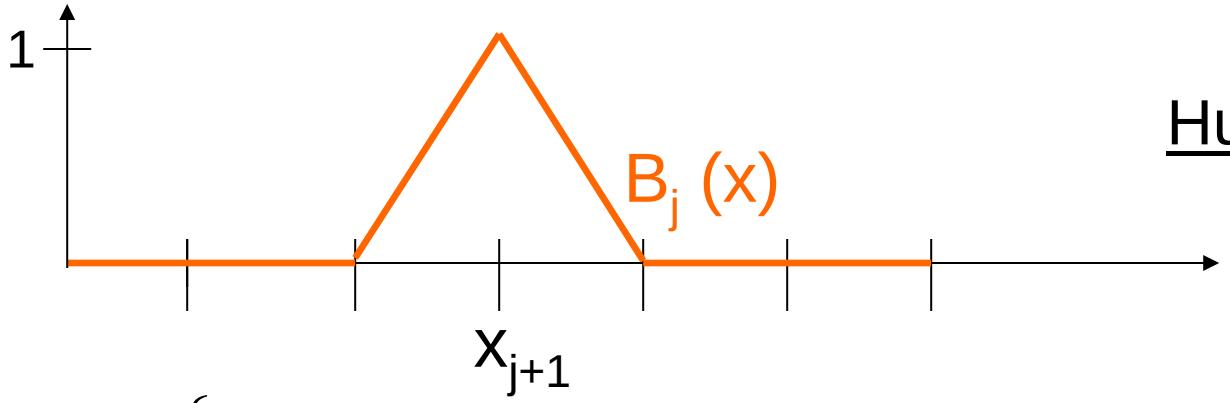
Bestimme wieder geeignete ***Basis*** von Splinefunktionen und löse das Interpolationsproblem in dieser Basis →

B-Spline-Basis (analog zu Lagrange-Polynomen):

Elementare Splinefunktion vom Grade k, die
- höchstens an einer Stützstelle den Wert 1 hat,
- und an ‚fast' allen anderen 0.

Der Fall k=1:

Gesucht: stückweise lineare, stetige Funktion, die genau bei x_{j+1} den Wert 1 annimmt, und an anderen Stützstellen 0.

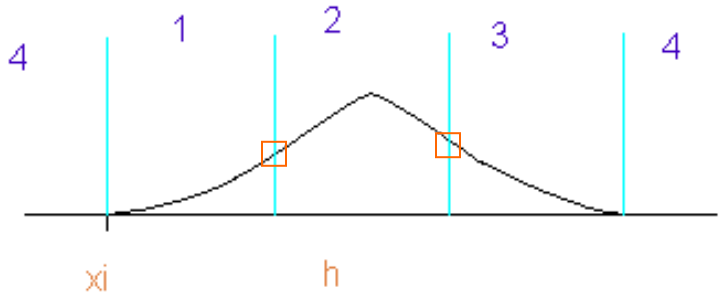


Hut-Funktion

$$B_j(x) := \begin{cases} \dfrac{x - x_j}{x_{j+1} - x_j} & \text{für} \quad x \in [x_j, x_{j+1}] \\ \dfrac{x - x_{j+2}}{x_{j+1} - x_{j+2}} & \text{für} \quad x \in [x_{j+1}, x_{j+2}] \\ 0 & \text{sonst} \end{cases}$$

Lösung der Interpolations-Aufgabe: $S(x) = \sum_{j=0}^{n} y_j B_{j-1}(x)$

B-Spline für k=2 (quadratisch):



$$B_i(x) := \frac{1}{2h^2}\begin{cases} (x - x_i)^2 & \textit{für} \quad x \in]x_i, x_{i+1}[\quad (1) \\ h^2 + 2h(x - x_{i+1}) - 2(x - x_{i+1})^2 & \textit{für} \quad x \in [x_{i+1}, x_{i+2}] \quad (2) \\ (x_{i+3} - x)^2 & \textit{für} \quad x \in]x_{i+2}, x_{i+3}[\quad (3) \\ 0 & \textit{sonst} \quad (4) \end{cases}$$

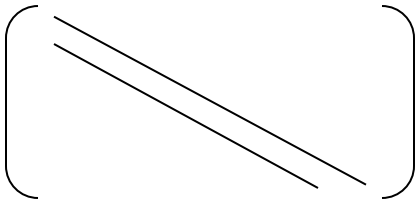
Stetig diff'bar, stückweise quadratisch.

Interpolationsproblem:

Ansatz: $$p(x) = \sum_{i=-1}^{n-1} a_i B_i(x), \quad a_i = ?$$

$$y_j = p(x_j) = \sum_{i=-1}^{n-1} a_i B_i(x_j) = a_{j-2} B_{j-2}(x_j) + a_{j-1} B_{j-1}(x_j) \quad \text{für j=0,1,...,n.}$$

Ergibt bidiagonales lineares Gleichungssystem zur Berechnung der Koeffizienten a_j :

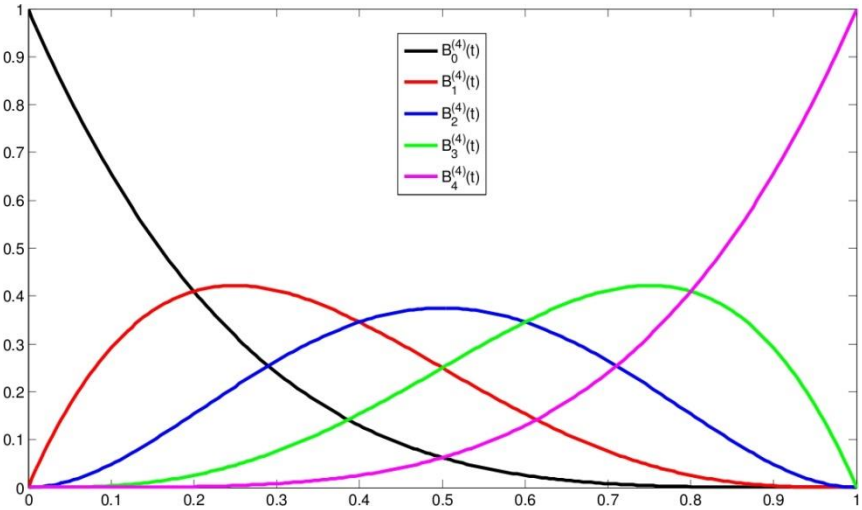


Trivial zu lösen! (Hier $B_{-2}(x) \equiv 0$, und x_{-1} dazu)

(c) Bezier-Kurven: (keine Interpolation)

Gesucht sind Kurven, die nicht exakt durch die Stützpunkte laufen, aber sich durch die Stützpunkte gut kontrollieren lassen.
(lokale Änderungen sollen nur lokal wirken)

Ausgangspunkt: Bernsteinpolynome in [0,1]



$$B_i^{(n)}(t) := \binom{n}{i} \cdot (1-t)^{n-i} t^i \qquad \textit{für} \quad i = 0,1,\ldots,n$$

Für n=4 sind dies also 5 Polynome.

Es gilt:
$$\sum_{i=0}^{n} B_i^{(n)}(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i = 1$$

Jedes $B_i^{(4)}(t)$ hat einen Teilbereich, indem es die anderen Polynome dominiert!

Gegeben: Stützpunkte $\vec{b}_i \in R^k$, i=0,...,n, als Vektoren im R^k .

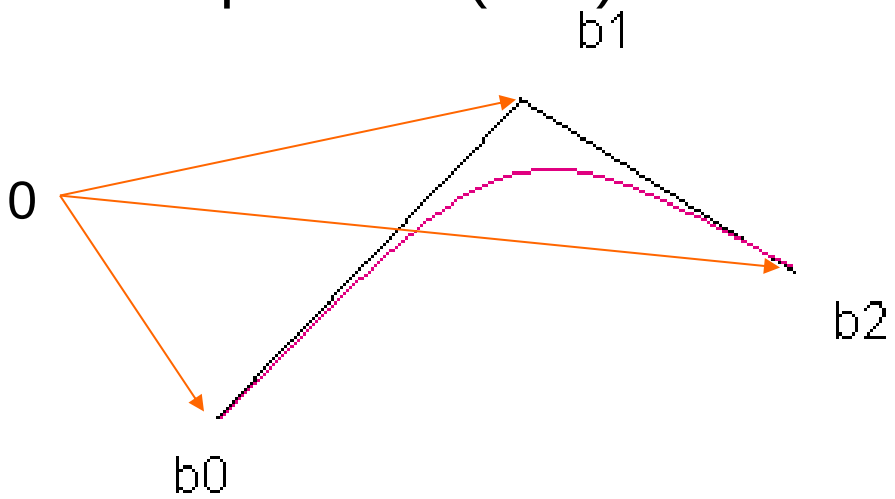
Die Bezierkurve zu diesen Punkten ist definiert durch:

$$X(t) := \sum_{i=0}^{n} \vec{b}_i B_i^{(n)}(t) \qquad \textit{für} \quad t \in [0,1]$$

Beginnend mit t=0 hat der Vektor X(t) zunächst den Wert b_0 , und fährt dann für wachsendes t der Reihe nach die anderen Kontrollpunkte b_i näherungsweise ab,
bis er für t=1 bei b_n anlangt.

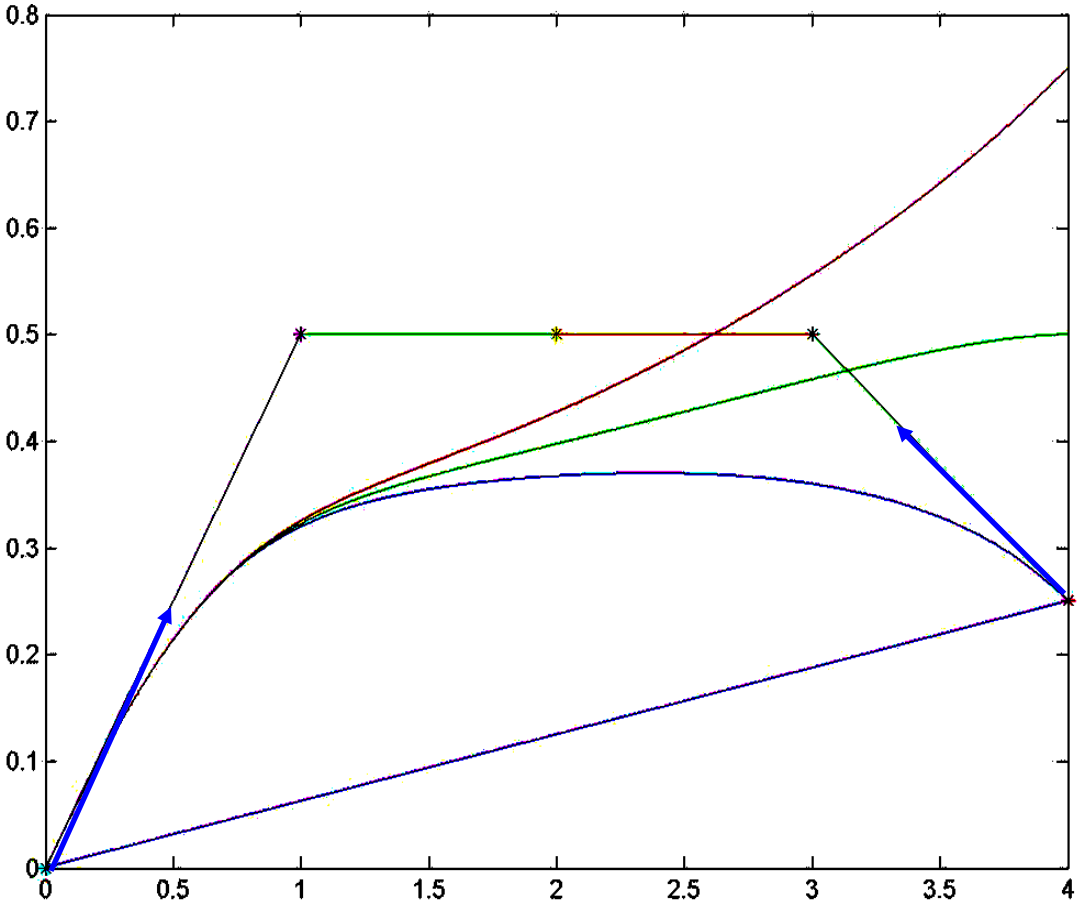
Dabei überwiegt in einem Teilbereich von [0,1] jeweils eines der Bernsteinpolynome $B_i(x)$ die anderen, und sorgt dafür, dass die Kurve zu dem entsprechenden Kontrollpunkt b_i gezogen wird.

Verlauf bei drei Kontrollpunkten (n=2):



Die sich ergebende Bezier-Kurve ist eine gewichtete Summe von Vektoren!
Ergibt keine interpolierende Kurve, aber gut steuerbar → Computergraphik (Word,CorelDraw,…)

Beispiel Bezierkurve (n=3):



Hier sind drei verschiedene Bezierkurven dargestellt, die sich nur im vierten Kontrollpunkt unterscheiden.

- Die Kurve verläuft stets innerhalb der konvexen Hülle der Kontrollpunkte (als Polygon eingezeichnet für den Fall, dass der rechte Kontrollpunkt den kleinsten Wert hat.).

- Am Anfangs-, bzw. Endpunkt zeigt die Tangente in Richtung des nächsten Nachbarpunktes (blauer Pfeil).

- Bei Veränderung des rechten Kontrollpunktes bleibt die linke Kurve fast unverändert; d.h. lokale Änderungen wirken sich kaum auf den Rest der Kurve aus!

Bezier-Applets: MATLAB bezier.m
http://www.theparticle.com/applets/nyu/BezierApplet/
http://sunsite.ubc.ca/LivingMathematics/V001N01/UBCExamples/Bezier/bezier.html
http://www.fh-friedberg.de/users/mlutz/JavaKurs/applets/Bezier/Bezier.html

Verbesserung:

Ersetze Bernsteinpolynome durch B-Splines

Vorteile:

- bessere Approximationseigenschaften
- einstellbare Differenzierbarkeit (k-mal stetig diff'bar)
- bessere Lokalitätseigenschaften

(lokale Änderungen wirken wirklich nur lokal)

B-Splines Applets:

http://www.cs.technion.ac.il/~cs234325/Applets/applets/bspline/GermanApplet.html

http://www.cs.uwaterloo.ca/~r3fraser/splines/bspline.html

4.1.9. Eines der Hauptprobleme der Numerik: Finde geeignete Ansatzfunktionen (Basis)

Klassisch: Polynome p(x),
rationale Funktionen p(x)/q(x) ,
Taylor-, Potenz-, Laurent-Reihe , z.B. $\sum a_k x^k$
Fourierreihen, cos(kx), sin(kx) , z.B. $\sum a_k e^{ikx}$

Analytische Darstellung

Heutzutage erforderlich:
Algorithmische Darstellung

Anforderungskatalog:

- Variable Diff'barkeit, angepasst an das gestellte Problem
- Variable Approximationsgüte, kleiner Fehler
- Adaptivität, Auflösung angepasst an (lokale) Anforderungen
- Lokalität, d.h. lokale Änderungen wirken nur lokal
- Schnelle Algorithmen, dünne Gleichungen
  - *Analysemöglichkeiten, vgl. Fouriertransformation Kap. 3*

Die ersten Punkte sind z.B. durch B-Splines erfüllt.
Es fehlt noch die Möglichkeit einer **‚Frequenzanalyse'**, die wir bei der Fouriertransformation kennenlernen werden → Multiskalendarstellung, Wavelet, Multigrid

Vergleich der Interpolationsverfahren:

- Standard-Polynom-Interpolation:

  Gegeben (x_j , y_j), $j=0,1,...,n$

  Gesucht interpolierendes Polynom vom Grad n

- Spline-Funktionen:

  stückweise Polynome vom Grad k, (k-1)-mal stetig diff'bar, aneinandergeheftet, so dass sie die Interpolations-bedingungen erfüllen.

- Hermite-Interpolation:

  Gegeben (x_j , y_j , y'_j), $j=0,1,...,n$

  Gesucht interpolierendes Polynom vom Grad 2n+1

- Nicht-interpolierende Kurven der Form

  $\sum_{i=1,...,n} b_j \cdot B_j(t)$, $t \in I$, mit Kontrollpunkten b_j und Gewichten $B_j(t)$ (z.B. Bezier-, B-Spline-Gewichte).

- Stückweise Hermite-Interpolation, d.h.

  Gegeben (x_j , y_j , y'_j) , $j=0,1,...,n$

  Gesucht stückweise quadratische, interpolierende Polynome

- Stückweise Bezierkurven, z.B. fasse für n=3 jeweils 4 Kontrollpunkte zusammen zur Bestimmung einer Teil-Bezierkurve. Hefte Teilkurven ‚glatt' aneinander.

4.1.10. Zweidimensionale Interpolation

Problem:

Gegeben Punkte (x_j , y_j) , Werte z_j, j=0,...,N

Gesucht Polynom in x und y mit $p(x_j , y_j) = z_j$, j=0,...,N,

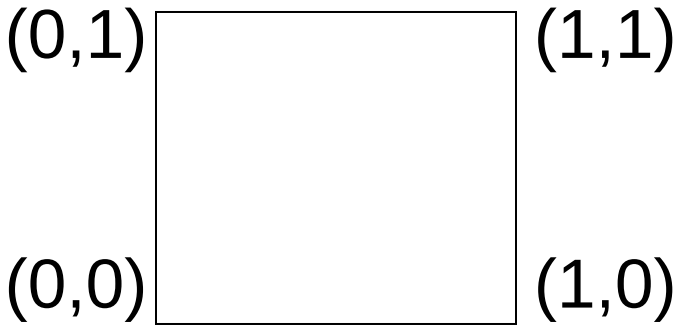
z.B. $$p(x,y) = \sum_{r=0}^{R} \sum_{s=0}^{S} a_{r,s} x^r y^s$$ mit (R+1)(S+1)=N+1

Ergibt N+1×N+1 lineares Gleichungssystem für die unbekannten Koeffizienten $a_{r,s}$.

Spezialfall ‚konstant': R=S=0, N=0;
Eine Stützstelle (x_0 , y_0) mit Wert z_0
Lösung: $p(x,y) \equiv z_0$

Spezialfall ‚bilinear': R=S=1, N=3:
4 Stützstellen, z.B. (0,0), (1,0), (0,1), (1,1)
bilineares Polynom p(x,y)=a+b·x+c·y+d·xy

Lösung der Interpolationsaufgabe :

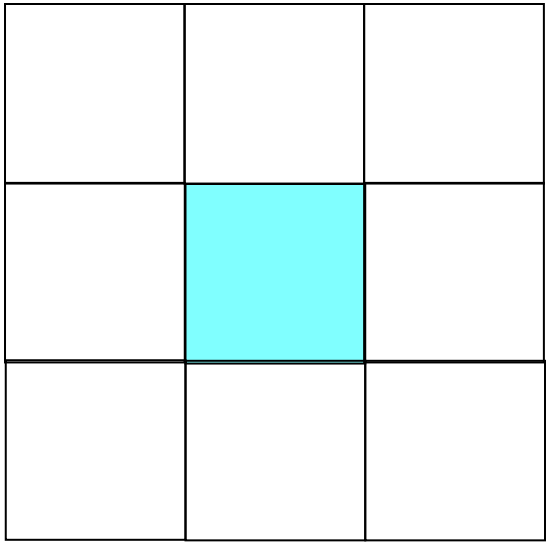


$$\begin{aligned} a &= z_0 \\ a+b &= z_1 \\ a+c &= z_2 \\ a+b+c+d &= z_3 \end{aligned} \quad \Leftrightarrow \quad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix}$$

$$p(x,y) = z_0 + (z_1 - z_0)x + (z_2 - z_0)y + (z_3 + z_0 - z_1 - z_2)xy$$

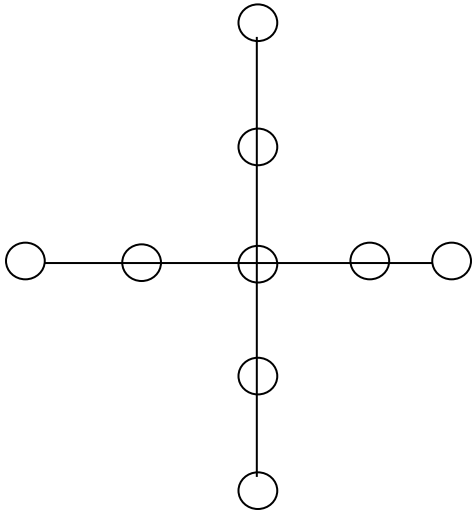
Spezialfall R=S=3, N=15,
16 Stützstellen:

bikubisch



Spezialfall R=S=2, N=8,
9 Stützstellen:

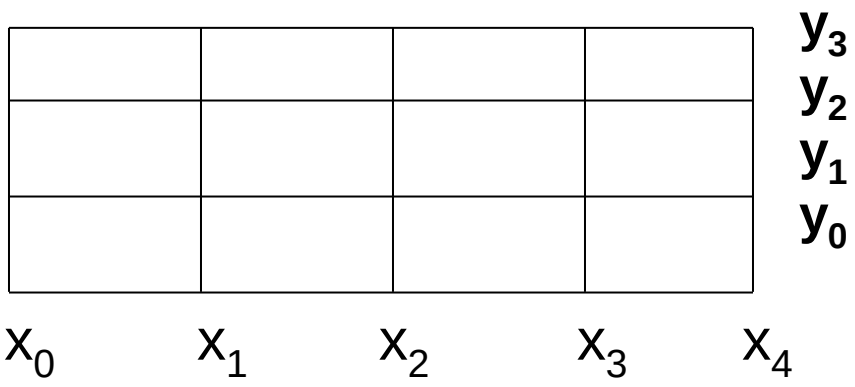
biquadratisch



Spezielle Problemstellung: Funktion auf Rechteck.

$(x_0 , ..., x_n)$ und $(y_0 , ..., y_m)$ führen zu Stützstellen auf Gitter

(x_j , y_k) , j=0,...,n und k=0,…,m mit Stützwerten $z_{j,k}$



Lösung durch 1D-Lagrange-Polynome:

$$L_{j,k}(x,y) := L_{x_j}(x) \cdot L_{y_k}(y)$$

in der Form:

$$p(x,y) = \sum_{j=0}^{n} \sum_{k=0}^{m} z_{j,k} \cdot L_{j,k}(x,y)$$

Andere Problemstellung: Grad r + s <= n (Gesamtgrad <= n)

$$p(x,y) = \sum_{r+s \le n} a_{r,s} x^r y^s$$

z.B. n=2: $p(x,y) = a + bx + cy + dx^2 + exy + fy^2$

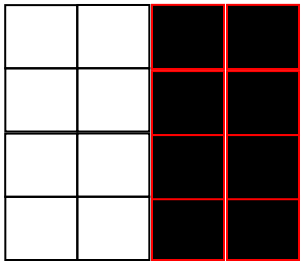
Hier ist der Gesamtgrad r+s aller Terme $x^r y^s$ durch n beschränkt!

Bestimme Koeffizienten wieder aus linearem Gleichungssystem.

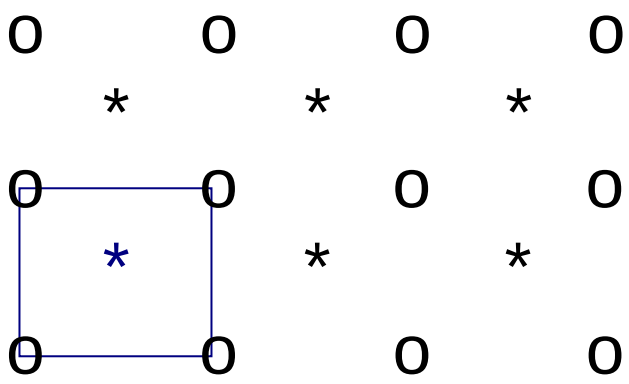
4.1.11. Beispiele aus Bildverarbeitung:

Bild als Matrix von Werten zwischen 0 und 255.
Jeder Wert gibt Grauwert an einer Stelle (einem Pixel) an.

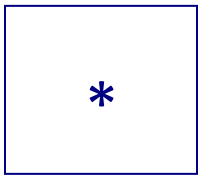
255 255 0 0
255 255 0 0 →
255 255 0 0 →
255 255 0 0



(a) Umskalierung, Drehung, Verschiebung führt auf neues Gitter!
Wie werden Pixelwerte des neuen Gitters bestimmt?



Bestimmung des neuen Wertes *
z.B. durch bilineare Interpolation mit



(b) Registrierung von Bilddaten:

Man verfügt über zwei verschiedene Aufnahmen desselben Objekts, z.B. zu verschiedenen Zeiten, von verschiedenen Aufnahmemedien, oder von verschiedenen Blickwinkeln aus aufgenommen.

Ziel ist es, diese Bilder zur Deckung zu bringen.

Z.B. eine neue Aufnahme mittels einer geeigneten Transformation (Verschiebung, Drehung) in das Referenzobjekt abzubilden.

Bestimme dazu a, b, und φ, so dass

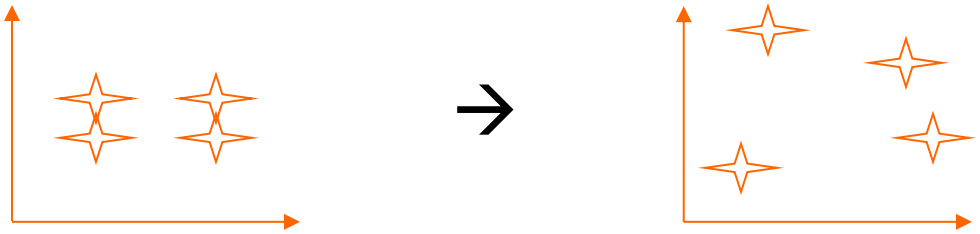
$$\min_{\varphi,a,b} \left\| \left(\begin{pmatrix} \cos(\varphi) & -\sin(\varphi) \\ \sin(\varphi) & \cos(\varphi) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix} \right) - \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} \right\|$$

(c) Kalibrierung von Bilddaten:

Wieder zwei verschiedene (aber deckungsgleiche) Aufnahmen desselben Objekts.

Nun ist aber eine der Aufnahmen verzerrt, z.B. im Falle von Röntgenaufnahmen durch Erdmagnetfeld, usw.

Wir nehmen an, dass wir Markierungspunkte in beiden Aufnahmen haben, die zur Übereinstimmung gebracht werden sollen, diesmal aber mit einer nichtlinearen Transformation.



Transformiere die Koordinaten der verzerrten Markierungspunkte möglichst gut auf die originalen Positionen,

z.B. durch zwei Polynome dritten Grades und Minimierung des Fehlers über alle Punkte:

(x_d , y_d) verzerrte Position von (x_p , y_p):

Minimiere über alle Markierungspunkte den Abstand

$$\begin{pmatrix} x_p \\ y_p \end{pmatrix} - \begin{pmatrix} a_0 + a_1 x_d + a_2 y_d + a_3 x_d y_d + a_4 x_d^2 + a_5 y_d^2 + a_6 y_d x_d^2 + a_7 x_d y_d^2 + a_8 x_d^3 + a_9 y_d^3 \\ b_0 + b_1 x_d + b_2 y_d + b_3 x_d y_d + b_4 x_d^2 + b_5 y_d^2 + b_6 y_d x_d^2 + b_7 x_d y_d^2 + b_8 x_d^3 + b_9 y_d^3 \end{pmatrix}$$

Betrachte z.B. $x_p = a_0+a_1x_{d,k}+\ldots+a_9y_{d,k}$ über alle Markierungspunkte:
ergibt Normalgleichungen zur Bestimmung der optimalen Koeffizienten $a_0,\ldots,a_9$.
Bestimme $b_0,\ldots,b_9$ genauso.

4.2. Quadratur

Integration ←→ *Flächenberechnung*

Problem:

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

In einigen Fällen kann das Integral mittels Stammfunktion berechnet werden:
Stammfunktion von x^k ist $x^{k+1}/(k+1)$;

Daher

$$\int_a^b x^k\ dx = \left[\frac{x^{k+1}}{k+1}\right]_a^b = \frac{b^{k+1} - a^{k+1}}{k+1}$$

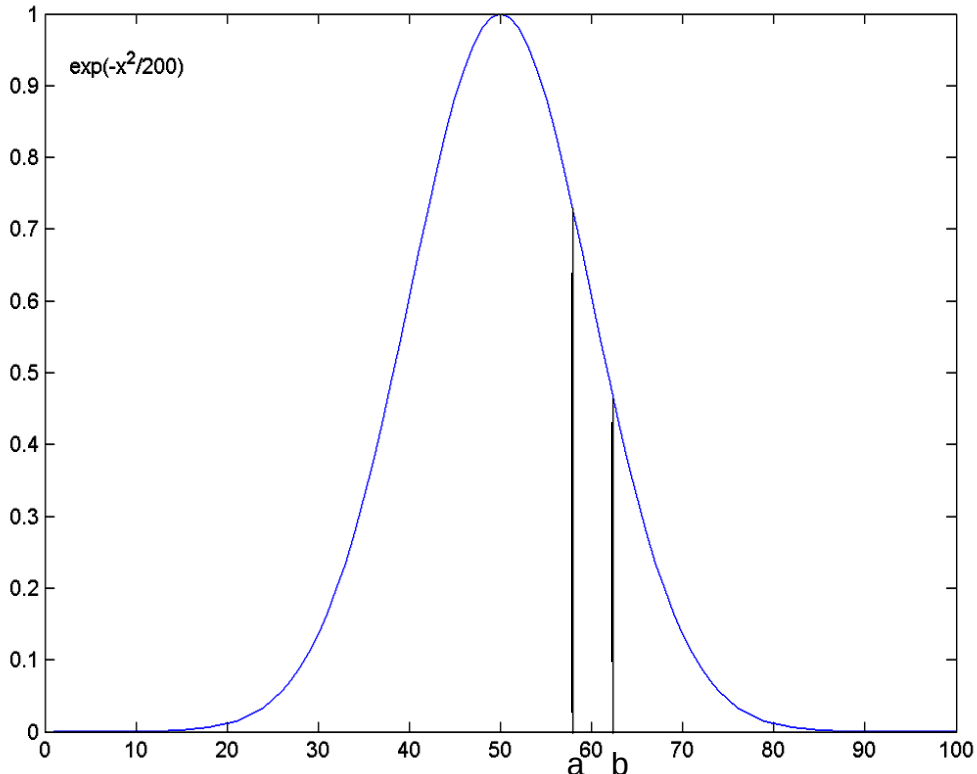
Stammfunktion von exp(x) ist exp(x), usw. (Formelsammlung)
In vielen Fällen existiert keine explizite Formel für die Stammfunktion!
Dann muss das Integral numerisch angenähert werden (vgl. Riemann'sche Summen, Obersumme, Untersumme)

4.2.1. Beispiel aus der Wahrscheinlichkeitsrechnung:

Gauss-Verteilung mit Mittelwert m und Standardabweichung σ; gesucht ist die Wahrscheinlichkeit dafür, dass ein gemessener Wert zwischen a und b liegt:

$$\frac{1}{\sigma\sqrt{2\pi}} \cdot \int_a^b \exp\left(-\frac{(x-m)^2}{2\sigma^2}\right) dx$$

Entspricht der Fläche unter der Kurve:



4.2.2. Allgemeine Form numerischer Quadraturregeln zur näherungsweisen Berechnung solcher Integrale:

$$I(f) = \int_a^b f(x)\ dx \approx \sum_{i=0}^{n} w_i f(x_i)$$

$$I(f) = \int_a^b f(x)\ dx \approx \sum_{i=0}^{n} w_i f(x_i)$$

Bestandteile:
- Stützstellen x_i mit Werten $f(x_i)$
- Gewichte w_i

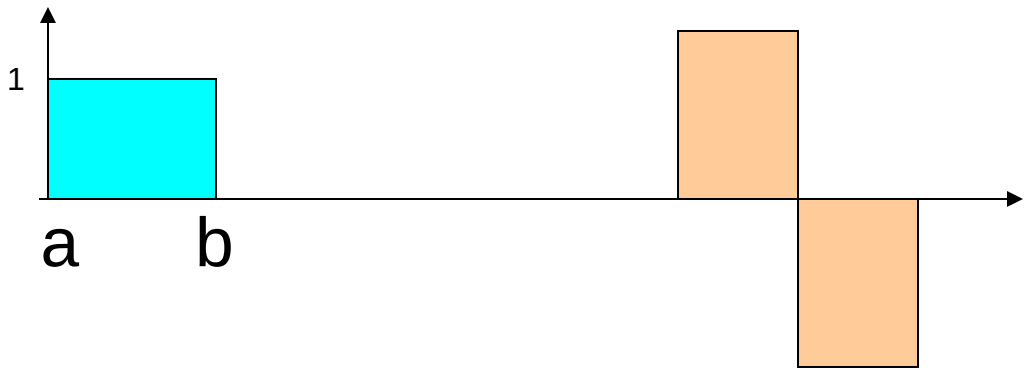
Frage: Wie sind die Stützstellen und Gewichte zu wählen, damit der Näherungswert möglichst gut ist.

Die Gewichte sollten dabei größer oder gleich 0 sein, um Rundungsfehler durch Auslöschung in der obigen Summe zu vermeiden!

Möglichst wenige Funktionsauswertungen für möglichst genaue Näherung!

4.2.3. Quadratur als Flächenberechnung

Das Integral zwischen a und b gibt den Flächeninhalt der Kurve an, die zwischen f(x) und der x-Achse eingeschlossen ist.



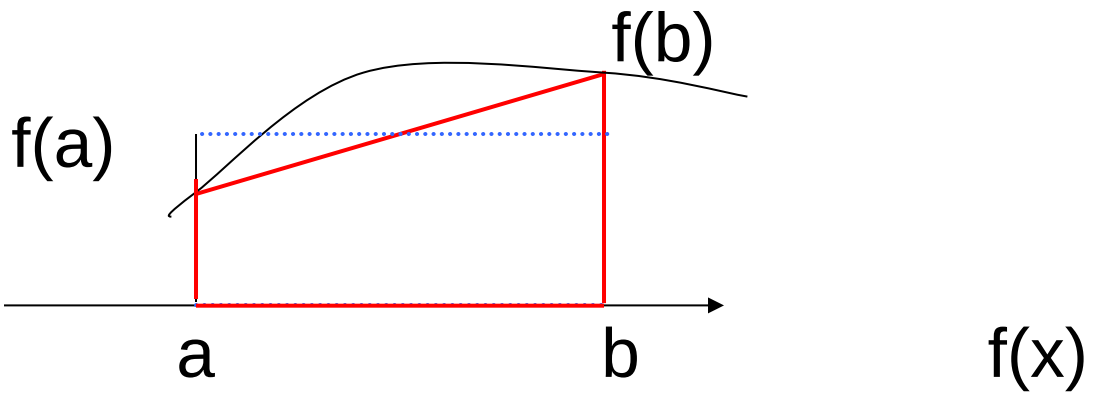
Die blaue, linke Fläche entspricht (b-a)*1
Und die rosa, rechte Fläche entspricht dem Wert 0, da obere und untere Fläche sich weg heben.

Trapezregel (n=1):

$$I(f) = \int_a^b f(x)dx \approx \frac{f(a)+f(b)}{2} \cdot (b-a) =$$

$$= \frac{b-a}{2} f(a) + \frac{b-a}{2} f(b)$$

Stützstellen a,b; Gewichte $w_0 = w_1 = (b-a)/2$

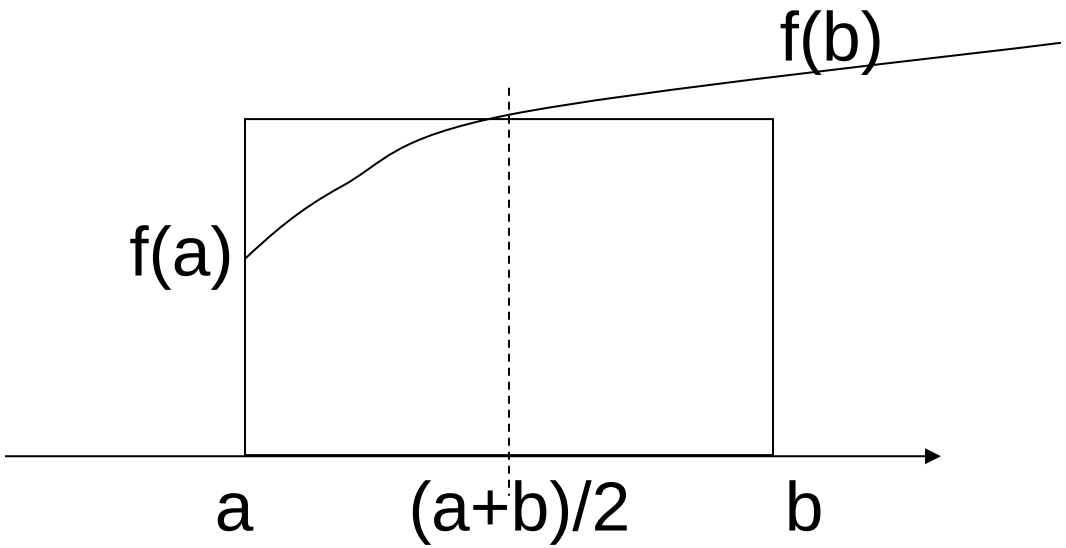


Mittelspunktregel (n=0):

Rechteck, begrenzt durch die Punkte (a,0), (b,0) und ((a+b)/2, f((a+b)/2)

$$I(f) \approx (b-a) \cdot f\left(\frac{a+b}{2}\right)$$

Also mit n=0 nur eine Stützstelle (a+b)/2 und ein Gewicht w_0 =(b-a).



4.2.4. Regeln aus der Interpolation

Nähere f(x) durch einfach zu integrierende Funktion an, z.B. durch interpolierendes Polynom.

Der Einfachheit halber wählen wir äquidistante Stützstellen

$$x_i = a + ih, \qquad i = 0,1,\ldots,n, \qquad h = (b-a)/n$$

Zu bestimmen sind noch passende Gewichte w_i .

Sie ergeben sich aus der Integration des f(x) interpolierenden Polynoms in der Lagrangedarstellung:

$$p(x) = \sum_{i=0}^{n} f(x_i) L_i(x)$$

also $$\int_a^b f(x)dx \approx \int_a^b p(x)dx = \sum_{i=0}^{n} f(x_i) \int_a^b L_i(x)dx = \sum_{i=0}^{n} f(x_i) w_i$$

Dadurch lassen sich allgemein Integrationsregeln herleiten.

Für n>6 ergeben sich auch negative Gewichte!
Numerisch problematisch (Auslöschung).

Daher ist dieses Vorgehen nur für kleine n sinnvoll.

1.Fall: n=1, nur x_0=a, x_1=b, h=b-a:

Das linear interpolierende Polynom ist

$$p(x) = f(a) + \frac{f(b) - f(a)}{b - a}(x - a)$$

Daher ergibt sich als Näherung:

$$\int_a^b f(x)dx \approx \int_a^b p(x)dx = \int_a^b \left(f(a) + \frac{f(b) - f(a)}{b - a}(x - a) \right) dx =$$

$$= f(a)(b - a) + \frac{f(b) - f(a)}{b - a}\left(\frac{b^2 - a^2}{2} - a(b - a) \right) =$$

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

Also Gewichte $w_0 = w_1 = h/2 = (b-a)/2$.

Dies ist genau die Trapezregel.

Fehlerabschätzung nach Kap. 4.1.6:

$$\left| \int_a^b \left(f(x) - p(x) \right) dx \right| = \left| \int_a^b \left(\frac{f^{(2)}(\chi)}{2} (x-b)(x-a) \right) dx \right| \leq$$

$$\leq \frac{M_2}{2} \int_a^b (x-a)(b-x)dx = \frac{M_2}{2} \cdot \frac{(b-a)^3}{6} = \frac{M_2 (b-a)^3}{12} = \frac{M_2}{12} h^3$$

mit $M_k := \max_{x \in [a,b]} \left| f^{(k)}(x) \right|$

Daher ist die Trapezregel exakt für Polynome vom Grad 1 (da dann $M_2 = 0$),
d.h. Näherungswert und exakter Wert sind gleich falls f(x) eine Polynom vom Grad <= 1 ist.

n=2, Simpson-Regel:

$$x_0 = a, \qquad x_1 = \frac{a+b}{2}, \qquad x_2 = b, \qquad h = \frac{b-a}{2}$$

$$w_0 = w_2 = \frac{h}{3} = \frac{b-a}{6}, \qquad w_1 = \frac{4}{3}h = \frac{4(b-a)}{6}$$

$$I(f) \approx \frac{h}{3} \cdot \left(f(a) + 4 f\left(\frac{a+b}{2}\right) + f(b) \right)$$

Fehlerabschätzung (ohne Beweis):

$$\left| \int_a^b \left(f(x) - p(x) \right) dx \right| \leq \frac{M_4 (b-a)^5}{2880} = \frac{M_4}{90} h^5$$

Daher ist die Simpsonregel sogar exakt für Polynome dritten Grades ($M_4 = 0$).

Allgemein:

$$\int_a^b f(x)dx \approx \int_a^b p_{0,1,\ldots,n}(x)dx = h\sum_{i=0}^{n} \alpha_i f(a+ih)$$

Beispiel: kmquad.ppt

4.2.5. Gauss-Quadratur

Bis jetzt waren die Stützstellen vorgegeben (äquidistant), und nur die Gewichte wurden optimal gewählt.

Aus Interpolation wissen wir aber, dass am Rand mehr Stützstellen sein sollten!

Daher neue Problemstellung:

Wie sind Gewichte und Stützstellen zu wählen, dass sich eine möglichst ‚gute' Quadratur-Regel ergibt.

‚Gut' heisst: die Regel soll Polynome möglichst hohen Grades exakt integrieren!

(Trapezregel: 2 Stützstellen, Grad 1 exakt,
Simpsonregel: 3 Stützstellen, Grad 3 exakt)

Finde $x_i \in [a,b]$, w_i, i=0,...,n, so dass für j=0,...,m:

$$\frac{b^{j+1}-a^{j+1}}{j+1} = \int_a^b x^j dx = I(x^j) \overset{!}{=} \sum_{i=0}^{n} w_i x_i^j$$

Dies sind m+1 nichtlineare Gleichungen für die 2n+2 Unbekannten x_i und w_i .

Fallstudie n=1:

$$\frac{b^{j+1}-a^{j+1}}{j+1} = \int_a^b x^j dx \overset{!}{=} w_0 x_0^j + w_1 x_1^j$$

$j=0:$	$b-a$	$=$	$w_0 + w_1$
$j=1:$	$\frac{b^2-a^2}{2}$	$=$	$w_0 x_0 + w_1 x_1$
$j=2:$	$\frac{b^3-a^3}{3}$	$=$	$w_0 x_0^2 + w_1 x_1^2$
$j=3:$	$\frac{b^4-a^4}{4}$	$=$	$w_0 x_0^3 + w_1 x_1^3$
$(j=4:)$	$\frac{b^5-a^5}{5}$	$=$	$w_0 x_0^4 + w_1 x_1^4$

Aus den ersten vier Gleichungen ergibt sich
(Lösung per Hand)

$$x_{0,1} = \frac{a+b}{2} \pm \frac{b-a}{2\sqrt{3}}, \qquad w_{0,1} = \frac{b-a}{2}$$

Gleichung j=4 ist mit diesen Werten nicht erfüllt!

Also werden durch diese Parameter Polynome bis zum Grad 3 exakt integriert.

Im Spezialfall: $\int_{-1}^{1} f(x)\frac{dx}{\sqrt{1-x^2}}$ ergeben sich als optimale Stützstellen, die Polynome möglichst hohen Grades exakt integrieren, wieder die Nullstellen der Tchebycheff-Polynome

$$\cos\left(\frac{(2j+1)\pi}{2n+2}\right), \text{ j=0,1,...,n}$$

Allgemein lässt sich zeigen, dass die optimalen Stützstellen gerade die Nullstellen von Orthogonalpolynomen sind, z.B. Legendre-, Hermite-, Laguerre-Polynome.
(siehe Formelsammlung)

Vorteil der Gauss-Quadratur mit optimalen Stützstellen :

Für festes n ergibt sich optimales Ergebnis!

Nachteil: Falls Resultat nicht genau genug, will man mehr Stützstellen verwenden; die neuen Stützstellen sind dann aber Nullstellen eines anderen Polynoms, also muss man an allen Stellen $f(x_i)$ neu berechnen.

Ausweg: Behalte die n alten Stützstellen. Suche nun n neue Stützstellen und 2n Gewichte zu den 2n Stützstellen, so dass Polynome möglichst hohen Grades exakt integriert werden.

4.2.6. Monte-Carlo-Methoden:

Wähle n Zufallspunkte aus dem Gebiet $Q \in R^n$,
berechne Funktionswerte an diesen Stellen und setze

$$I = \int_Q f(\vec{x}) d\vec{x} \approx \sum_{i=1}^{n} f(x_i) \frac{|Q|}{n}$$

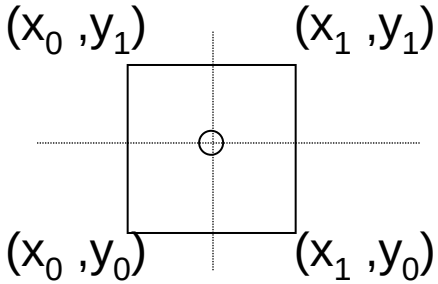
Besonders nützlich bei hochdimensionalen Integralen!

Stützstellen: Zufallsstellen $f(x_i)$

Gewichte: immer $|Q| / n$.

4.2.7. Zweidimensionale Integration

Mittelpunktsregel:



$$\int_Q f(x,y)dxdy \approx (y_1 - y_0)\int_{x_0}^{x_1} f(x, \frac{y_0 + y_1}{2})dx \approx$$

$$\approx (x_1 - x_0)(y_1 - y_0) f\left(\frac{x_1 + x_0}{2}, \frac{y_1 + y_0}{2}\right)$$

Allgemein Quadratur-Regeln aus der 2D-Interpolation:

Bestimme zu f(x,y) ein interpolierendes Polynom p(x,y), so dass Stützstellen und Grad zusammenpassen

(vgl. Kap. 4.1.10).

Gesucht: Integral über Quadrat Q mit Ecken (x_0, y_0), (x_0, y_1), (x_1, y_0), (x_1, y_1)

$$\int_Q f(x,y)dxdy \approx \int_Q p(x,y)dxdy = \int_Q \sum_{r,s} a_{r,s} x^r y^s dxdy =$$

$$= \sum_{r,s} a_{r,s} \int_{x_0}^{x_1} x^r dx \int_{y_0}^{y_1} y^s dy = \sum_{r,s} \frac{a_{r,s}\left(x_1^{r+1} - x_0^{r+1}\right)\left(y_1^{s+1} - y_0^{s+1}\right)}{(r+1)(s+1)}$$

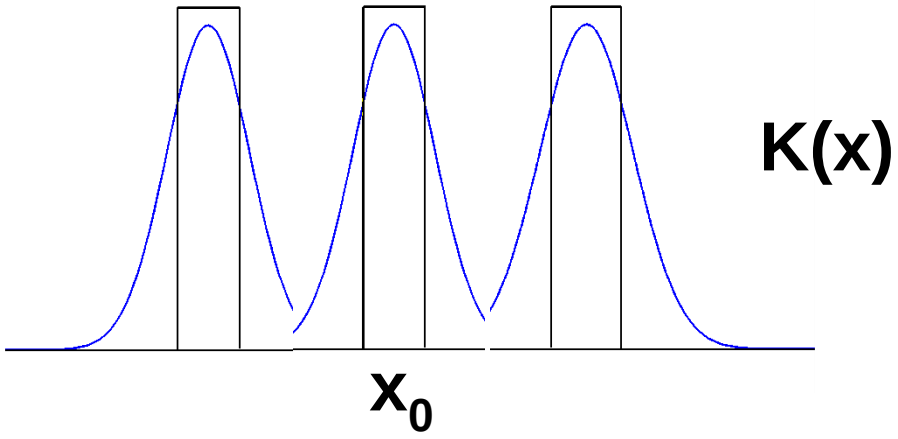
Andere Möglichkeit: Lagrange-Ansatz wie 4.1.10.

4.2.8. Deblurring bei verschmierten Bildern, Gauss'scher Weichzeichner:

Modell: Jeder Pixelpunkt wird entsprechend einer Gauss-Verteilung, verschmiert, z.B. Aufnahme durch Atmosphäre:

(Zur Vereinfachung hier nur eindimensional)

An der Stelle x_0 wird der ursprüngliche Wert $f(x_0)$ ersetzt durch gestörten Wert $g(x_0)$, der sich aus der Überlagerung von Nachbarpunkte ergibt.



Das beobachtete Bild g entsteht durch eine ‚Faltung' des ursprünglichen Bildes f mit dem Gauss-Kern $K(x)=\exp(-x^2/(2\pi\sigma^2))$, der genau die obige Funktion beschreibt

Um ein diskretes Modell zu erhalten, ersetze man das Faltungs-Integral an jeder diskreten Stelle x_j z.B. durch die Mittelpunktsumme (oder Trapezsumme) →

$$g(x_j) = \int \exp\left(-\frac{(x_j - t)^2}{2\pi\sigma^2}\right) \cdot f(t)dt \approx$$

$$\approx h\sum_k \exp\left(-\frac{(x_j - x_k)^2}{2\pi\sigma^2}\right) \cdot f(x_k)$$

Dies führt auf ein lineares Gleichungssystem, das die Beziehung zwischen ursprünglichen und gestörten Werten beschreibt:

$$\left(\exp\left(-\frac{(j-k)^2}{2\pi\sigma^2 n^2} \right) \right)_{j,k=1}^{n} \cdot \vec{f} = \vec{g}$$

Durch Lösung des Gleichungssystems kann man aus den gestörten Daten $g(x_j)$ das Originalbild $f(x_k)$ wieder gewinnen.

Aber das Gleichungssystem $K \cdot \vec{f} = \vec{g}$

ist sehr schlecht konditioniert.

Daher verwendet man zusätzlich Regularisierung, z.B.:

$$\left(K^T K + \rho^2 I \right) \cdot \vec{f} = K^T \vec{g}$$

Umgekehrt beschreibt die Beziehung

$$\left(\exp\left(-\frac{(j-k)^2}{2\pi\sigma^2 n^2} \right) \right)_{j,k=1}^{n} \cdot \vec{f} = \vec{g}$$

einen sog. Weichzeichner.

„Verwasche“ die Pixel durch Mittelwertbildung mit Nachbarn,

z.B. $x_k \rightarrow (x_{k-1} + 2x_k + x_{k+1}) / 4$

oder Maske $\begin{bmatrix} & 1 & \\ 1 & 4 & 1 \\ & 1 & \end{bmatrix} /8$

bild→blur