4.2. Quadratur

Integration ←→ *Flächenberechnung*

Problem: $$\boxed{I(f) := \int_a^b f(x) \quad dx}$$

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

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

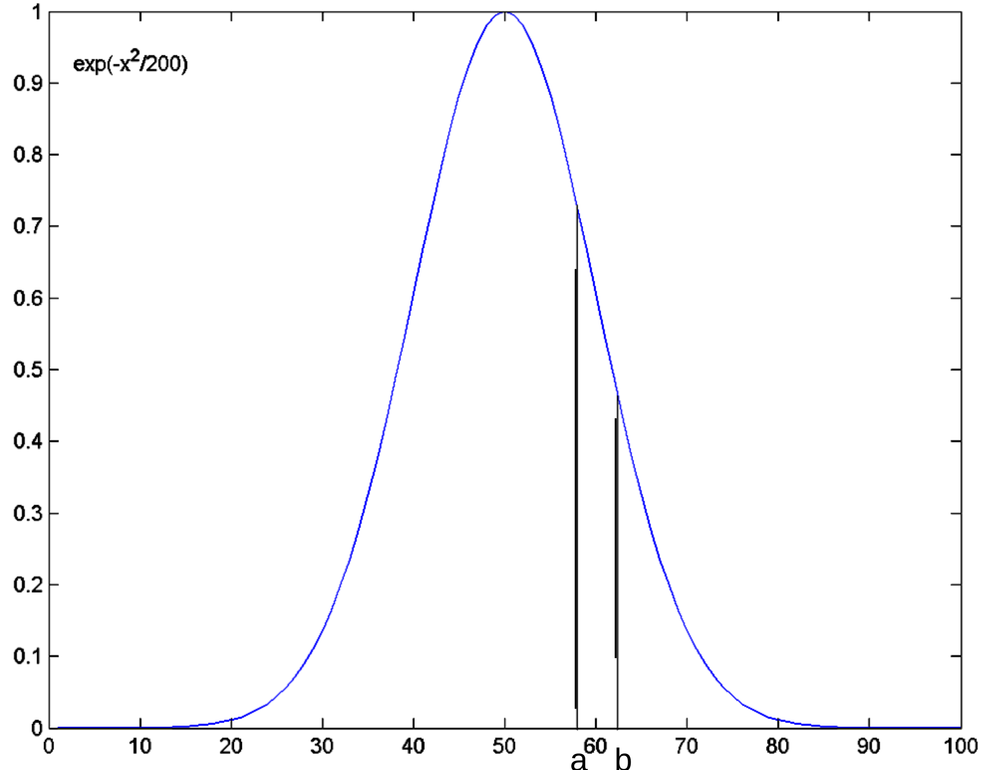
Stammfunktion von exp(x) ist exp(x), usw. (Formelsammlung)
In vielen Fällen existiert keine explizite Formel für die Stammfunktion!
Dann muss das Integral numerisch angenähert werden (vgl. Riemann'sche Summen, Obersumme, Untersumme)

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:

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

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

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

Frage: Wie sind die Stützstellen und Gewichte zu wählen,

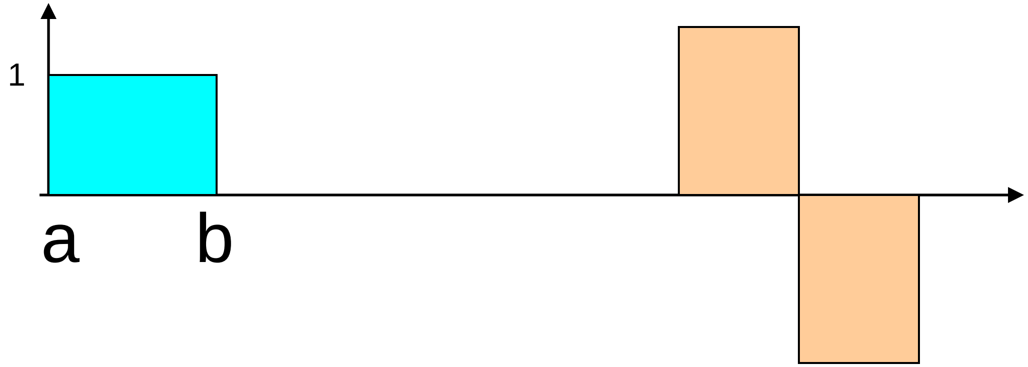
damit der Näherungswert möglichst gut ist.

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.

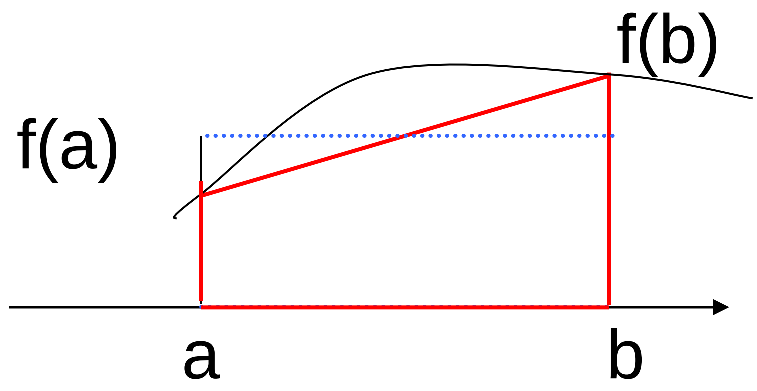


Die blaue, linke Fläche entspricht (b-a)*1
Und die rosa, rechte Fläche entspricht dem Wert 0, da obere und untere Fläche sich weg heben.

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 $w_0 = w_1 = (b-a)/2$



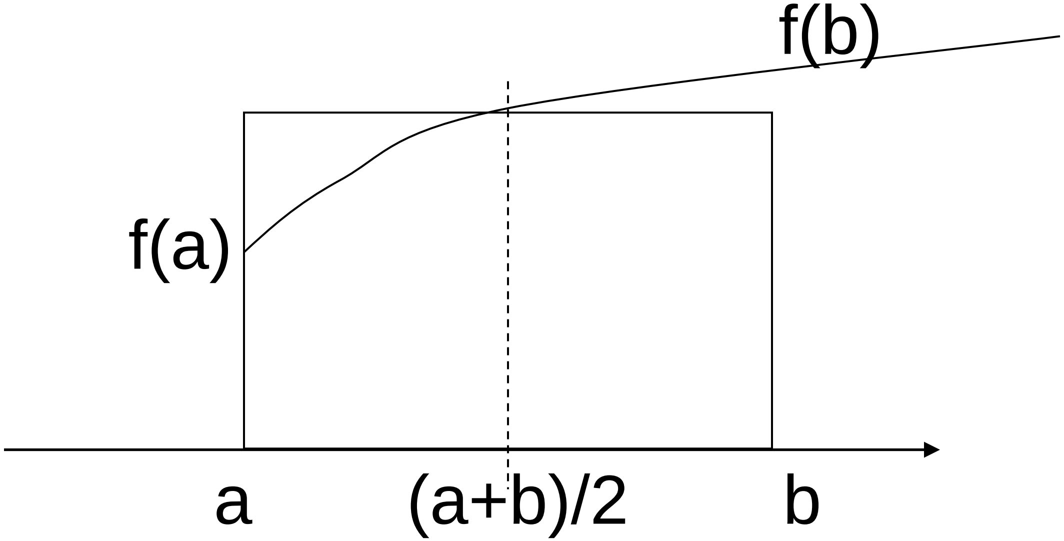
f(x)

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:

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

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

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

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:

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

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

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

Fehlerabschätzung (ohne Beweis):

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

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. <u>Gauss-Quadratur</u>

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 <u>und</u> 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, $i=0,\ldots,n$, so dass für $j=0,\ldots,m$:

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

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

Fallstudie n=1:

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

$$\boxed{\begin{array}{lccc} j=0: & b-a & = & w_0+w_1 \\ j=1: & \dfrac{b^2-a^2}{2} & = & w_0x_0+w_1x_1 \\ j=2: & \dfrac{b^3-a^3}{3} & = & w_0x_0^2+w_1x_1^2 \\ j=3: & \dfrac{b^4-a^4}{4} & = & w_0x_0^3+w_1x_1^3 \\ (j=4:) & \dfrac{b^5-a^5}{5} & = & w_0x_0^4+w_1x_1^4 \end{array}}$$

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

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

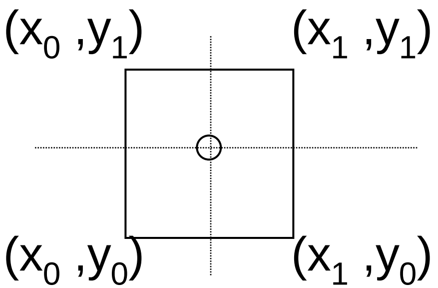
Besonders nützlich bei hochdimensionalen Integralen!

Stützstellen: Zufallsstellen $f(x_i)$

Gewichte: immer $|Q| / n$.

4.2.7. Zweidimensionale Integration

Mittelpunktsregel:



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

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

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)

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

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

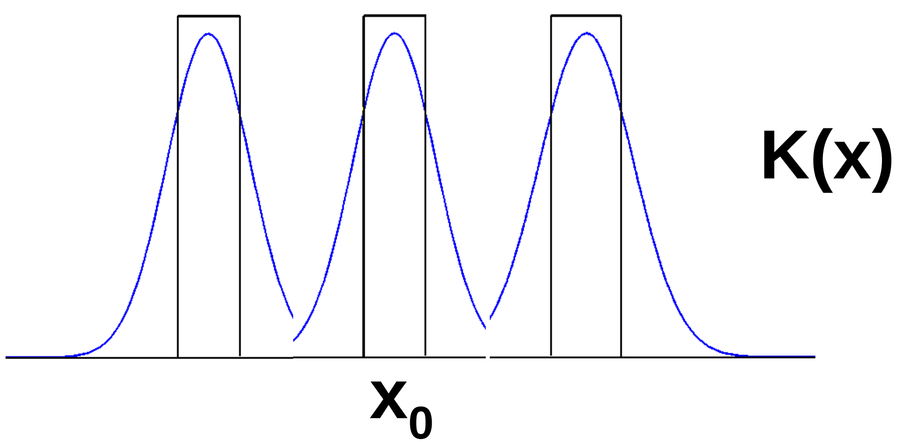
Andere Möglichkeit: Lagrange-Ansatz wie 4.1.10.

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

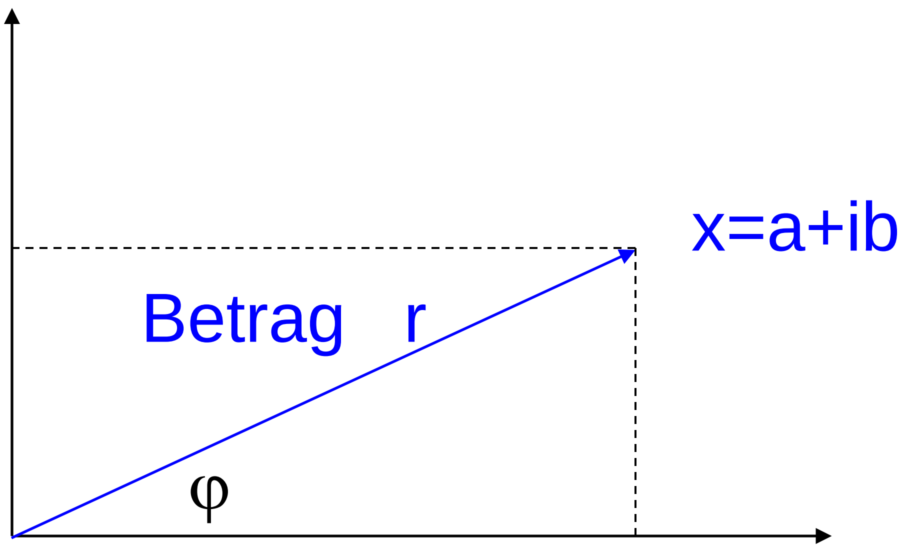
oder Maske $\begin{bmatrix} & 1 & \\ 1 & 4 & 1 \\ & 1 & \end{bmatrix} / 8$

bild→blur

5.1. Komplexe Zahlen und trigonometrische Interpolation

$$i = \sqrt{-1}$$

Imaginärteil
b=r·sin(φ)



a = r·cos(φ) = Realteil

Komplexe Zahl:

Vektor in der Ebene mit spezieller Multiplikation

Zwei mögliche Darstellungen:

x = a + i b = r · (cos(φ) + i · sin(φ))   = r · exp(i φ)

mit $$r = |x| = \sqrt{a^2 + b^2}, \qquad \tan(\varphi) = \frac{b}{a}$$

$$e^{i\varphi} = \sum_{n=0}^{\infty} \frac{(i\varphi)^n}{n!} = \sum_{k=0}^{\infty} \frac{(i\varphi)^{2k}}{(2k)!} + \sum_{k=0}^{\infty} \frac{(i\varphi)^{2k+1}}{(2k+1)!} =$$

$$= \sum_{k=0}^{\infty} \frac{(-1)^k \varphi^{2k}}{(2k)!} + i \cdot \sum_{k=0}^{\infty} \frac{(-1)^k \varphi^{2k+1}}{(2k+1)!} = \cos(\varphi) + i \cdot \sin(\varphi)$$

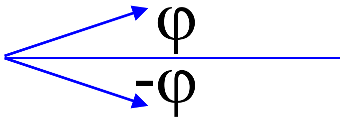
Multiplikation:

x * y = (a+ib)*(c+id) = (ac-bd)+i(ad+bc) =

= r exp(i φ) * s exp(i θ) = rs exp(i (φ+θ))

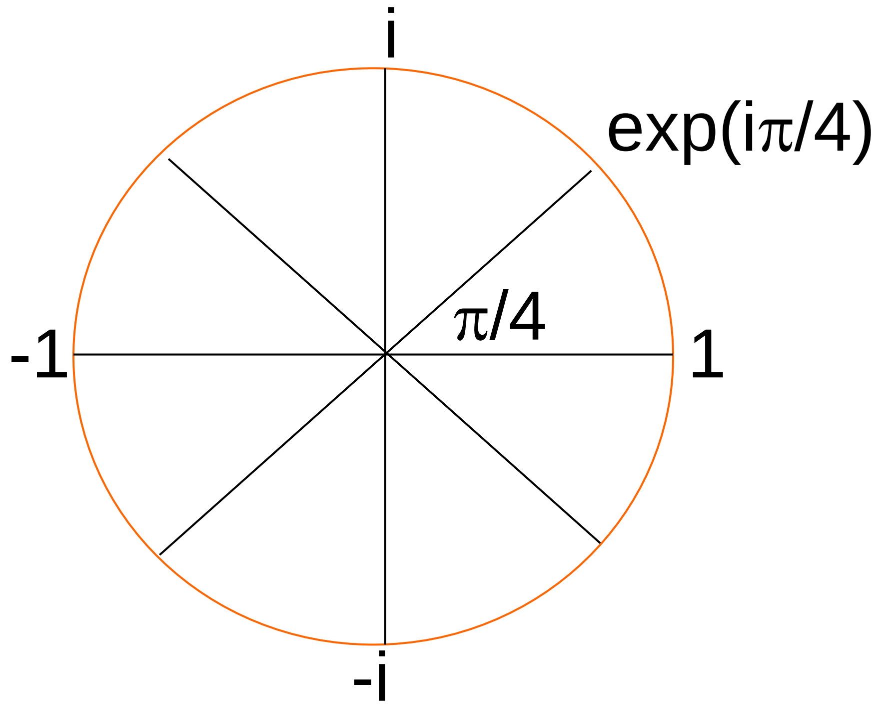
(Beträge multipliziert, Winkel addiert)

konjugiert komplexe Zahl: $\bar{x} = a - ib = r \cdot e^{-i\varphi}$



n-te Einheitswurzeln: $x^n = 1 = e^{2\pi i}$ hat n Lösungen

$$\exp\left(\frac{2\pi i \cdot j}{n}\right) = \cos\left(\frac{2\pi j}{n}\right) + i\sin\left(\frac{2\pi j}{n}\right), \qquad j = 0, \ldots, n-1$$



für n=8

$$\left(e^{(i\pi/4)}\right)^8 = e^{(2\pi i)} = 1$$

Wir betrachten nun ein komplexes Interpolationsproblem mit den n-ten Einheitswurzeln als Stützstellen:

$$\omega^j, \quad j = 0,1,\ldots,n-1 \quad mit \quad \omega := \exp(2\pi i/n)$$

Zu den n Stützstellen ω^j und vorgegeben Werten v_j, j=0,1,...,n-1, finde man das Polynom mit den Koeffizienten c_j, j=0,1,...,n-1, so dass gilt: $p(\omega^j) = v_j$, für j=0,1,...,n-1 ,
also

$$v_j = p\left(\exp(2ij\pi/n)\right) =$$

$$= c_0 + c_1 \cdot \exp(2ij\pi/n) + \ldots + c_{n-1} \cdot \exp(2ij\pi(n-1)/n) =$$

$$= \sum_{k=0}^{n-1} c_k \cdot \exp(2ijk\pi/n) = \sum_{k=0}^{n-1} c_k \cdot \omega^{jk} = \sum_{k=0}^{n-1} c_k (\omega^j)^k$$

für j=0,1,...,n-1

Wie üblich erhalten wir lineares n x n – Gleichungssystem zur Bestimmung der c_j:

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ \vdots \\ v_{n-1} \end{pmatrix}$$

, **F_n * c = v**

Diese Matrix F_n hat spezielle Eigenschaften:

$$F_n^T = F_n, \qquad F_n^H = \overline{F_n} = n \cdot F_n^{-1} = n \cdot inv(F_n)$$

(dabei ist A^H:= conj(A^T) das hermite'sche von A)

Damit erhält man die Lösung durch **c = inv(F_n)*v**

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{pmatrix} = \frac{1}{n} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \overline{\omega} & \overline{\omega}^2 & \cdots & \overline{\omega}^{n-1} \\ 1 & \overline{\omega}^2 & \overline{\omega}^4 & \cdots & \overline{\omega}^{2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \overline{\omega}^{n-1} & \overline{\omega}^{2(n-1)} & \cdots & \overline{\omega}^{(n-1)(n-1)} \end{pmatrix} \cdot \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ \vdots \\ v_{n-1} \end{pmatrix}$$

mit $\overline{\omega} = conj(\omega) = \exp(-2\pi i/n)$, also

$$\boxed{c_k = \frac{1}{n}\sum_{j=0}^{n-1} v_j \overline{\omega}^{jk}, \qquad k = 0,1,\ldots,n-1}$$

Dies beschreibt gerade die sog. Diskrete Fourier-Transformation, abgekürzt DFT, also

,

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{pmatrix} = DFT \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ \vdots \\ v_{n-1} \end{pmatrix} \qquad c_k = \frac{1}{n} \sum_{j=0}^{n-1} v_j \overline{\omega}^{jk}, \qquad k = 0,1,\ldots,n-1$$

Die inverse DFT entspricht dem ersten Matrix-Vektor-Produkt

$$\begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ \vdots \\ v_{n-1} \end{pmatrix} = IDFT \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{pmatrix} \qquad v_j = \sum_{k=0}^{n-1} c_k \omega^{jk}, \qquad j = 0,1,\ldots,n-1$$

DFT beschreibt den Zusammenhang zwischen Funktionswerten und (Polynom)-Koeffizienten

Normalerweise sind daher die Kosten für die Durchführung einer DFT gerade die Kosten eines Matrix*Vektor-Produkts, also $O(n^2)$.
Mittels ‚divide & conquer' werden wir ein rekursives Verfahren herleiten, um die DFT in $O(n \log(n))$ Operationen auszuführen, die sog. FFT = Fast Fourier Transform

Ein schneller Algorithmus ist sehr wichtig und nützlich, da die DFT sehr oft gebraucht wird (→ Frequenzanalyse)

Vorsicht: In der Literatur unterscheiden sich die Definition von DFT und IDFT manchmal im Vorfaktor $1/n$ oder sind vertauscht:
z.B. DFT und IDFT beide unitär mit Faktor $1/\sqrt{n}$,
oder einmal Faktor 1 und einmal 1/n

5.2. Die Fast Fourier-Transformation

5.2.1. Rekursive Formulierung der IDFT:

Die IDFT besteht aus n Summen mit n Summanden, die aus Produkten mit n-ten Einheitswurzeln bestehen.

Idee:
Zerlege die Summen geschickt in zwei Teilsummen halber Länge, aus denen sich die ursprünglichen Summen leicht gewinnen lassen.

Dazu notwendig: $n=2m$ gerade! Oder $m=n/2$.

Aufteilung in Summanden zu geradem und ungeradem Index.

Für j=0,1,...,m-1   erhält man die erste Hälfte der Komponenten

$$v_j = \sum_{k=0}^{n-1} c_k \cdot \exp\left(\frac{2\pi ijk}{n}\right) =$$

$$= \sum_{k=0}^{n/2-1} c_{2k} \cdot \exp\left(\frac{2\pi ij2k}{n}\right) + \sum_{k=0}^{n/2-1} c_{2k+1} \cdot \exp\left(\frac{2\pi ij(2k+1)}{n}\right)$$

$$= \sum_{k=0}^{m-1} c_{2k} \cdot \exp\left(\frac{2\pi ijk}{m}\right) + \omega^j \cdot \sum_{k=0}^{m-1} c_{2k+1} \cdot \exp\left(\frac{2\pi ijk}{m}\right), \; j = 0,1,\ldots, m-1$$

j-te Komponente → ω^j .

Daher erhält man also die erste Hälfte der Komponenten von v aus zwei IDFT's halber Länge m, einmal angewendet auf den Vektor der geradzahligen Indizes von c, und einmal auf den Vektor der ungeradzahligen Indizes von c.

Entsprechend muss noch die zweite Hälfte von v bestimmt werden:

$$v_{m+j} =$$

$$= \sum_{k=0}^{m-1} c_{2k} \cdot \exp\left(\frac{2i\pi(j+m)k}{m}\right) + \omega^{j+m} \cdot \sum_{k=0}^{m-1} c_{2k+1} \cdot \exp\left(\frac{2i\pi(j+m)k}{m}\right)$$

$$= \sum_{k=0}^{m-1} c_{2k} \cdot \exp\left(\frac{2ijk\pi}{m}\right) - \omega^{j} \cdot \sum_{k=0}^{m-1} c_{2k+1} \cdot \exp\left(\frac{2ijk\pi}{m}\right)$$

da $\omega^m = \exp(2im\pi / n) = \exp(i\pi) = \cos(\pi) + i \cdot \sin(\pi) = -1$

und $\exp\left(\frac{2imk\pi}{m}\right) = \exp(2ik\pi) = 1^k = 1$

Daher ergibt sich die zweite Hälfte des Vektors v aus denselben Fouriertransformierten halber Länge wie vorher, diesmal aber aus der Differenz.

Damit lässt sich also

$$\begin{pmatrix} v_0 & \cdots & v_{n-1} \end{pmatrix} = IDFT\begin{pmatrix} c_0 & \cdots & c_{n-1} \end{pmatrix}$$

zurückführen auf

$$\begin{pmatrix} v_0^{(g)} & \cdots & v_{m-1}^{(g)} \end{pmatrix} = IDFT\begin{pmatrix} c_0 & c_2 & \cdots & c_{n-2} \end{pmatrix}$$

und

$$\begin{pmatrix} v_0^{(u)} & \cdots & v_{m-1}^{(u)} \end{pmatrix} = IDFT\begin{pmatrix} c_1 & c_3 & \cdots & c_{n-1} \end{pmatrix}$$

mit der Kombination für j=0,1,...,m-1:

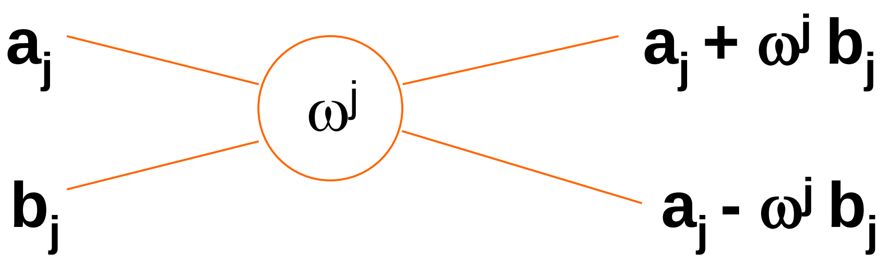
$$v_j = v_j^{(g)} + \omega^j \cdot v_j^{(u)} \qquad und \qquad v_{j+m} = v_j^{(g)} - \omega^j \cdot v_j^{(u)}$$

Gerade/ungerade in c → erste Hälfte/zweite Hälfte in v
Entsprechend wird die IDFT der halb so langen Vektoren wieder auf jeweils zwei IDFT (viertelter Länge) zurückgeführt.

Anstelle einer IDFT Länge n sind also nun vier IDFT’s der Länge n/4 zu berechnen.
Allgemein dazu notwendig: $n=2^p$ ist Zweierpotenz;
Dann kann dieses Verfahren rekursiv durchgeführt werden, bis man *eine IDFT der Länge n* ausgedrückt hat durch

n IDFT’s der Länge 1.

Grundidee:
Rekursive Aufteilung des aktuellen Vektors in geraden und ungeraden Anteil; falls die Ergebnisse der beiden IDFT’s halber Länge vorliegen, werden die beiden Teile zum Ergebnis doppelter Länge kombiniert mittels



Rekursives Programm für IFFT:

FUNCTION$(v_0,\ldots,v_{n-1})$ = IDFT$(c_0,\ldots,c_{n-1},n)$
IF n==1
  $v_0 = c_0$;
ELSE
  m=n/2 ;
  $(g_0,\ldots, g_{m-1})$ = IDFT$(c_0, c_2,\ldots, c_{n-2},m)$;
  $(u_0,\ldots, u_{m-1})$ = IDFT$(c_1, c_3,\ldots, c_{n-1},m)$;
  $\omega = \exp(2i\pi/n)$;
  FOR j=0:m-1
    $v_j \quad = g_j + \omega^j u_j$;
    $v_{j+m} = g_j - \omega^j u_j$;
  END
END

MATLAB: my_IDFT.m