

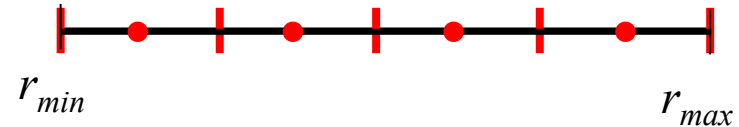
Algorithmik kontinuierlicher Systeme

Diskretisierung und Quantisierung (Teil 2)



- Diskretisierung
 - ▶ Faltung
 - ▶ Fourier Transformation
 - ▶ Abtasttheorem
 - ▶ Aliasing / Anti-Aliasing
- Quantisierung
 - ▶ Diskrete Approximation reeller Werte
 - ▶ Gleitpunktzahlen, Maschinenzahlen
 - ▶ Beispiel: Quantisierung von Farbwerten (*median cut*)
 - ▶ Vektor-Quantisierung

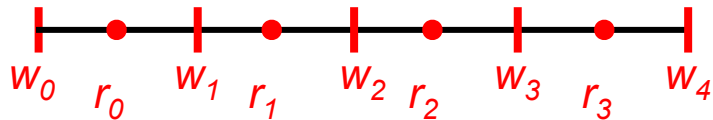
- Zu approximieren $r \in \mathbb{R}, r_{min} \leq r \leq r_{max}$
- Wertebereich $W := [r_{min}, r_{max}]$
- Zerlegung in disjunkte, gleich große Teilintervalle
- L Teilintervalle $[w_j, w_{j+1}]$ der Länge $d = (r_{max} - r_{min})/L$
- Repräsentant $r_j \in [w_j, w_{j+1}]$, z.B: $r_j = (w_j + w_{j+1})/2$
- **Quantisierung:** $Q[r] \approx r_j$ falls $r \in [w_j, w_{j+1}]$
- Beobachtung: nicht optimal sofern die Punkte nicht gleichverteilt sind.



- Nahezu gleichverteilte Punkte:

~~XXXXXXXXXXXX XXXX XXXX XXXX XXXX~~

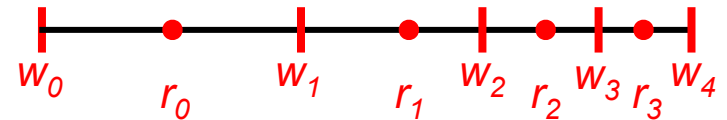
→ uniforme Quantisierung



- Teile Intervall in äquidistante Teilintervalle und nimm jeweils Mittelpunkt als Repräsentanten

- Ungleich verteilte Punkte:

→ nicht uniforme Quantisierung



- Unterschiedliche Strategien
 - Median Cut
 - Least square minimizer
 - Maschinenzahlen

- Definition:

Die Menge der **normalisierten t -stelligen Gleitpunktzahlen** zur Basis B sind definiert als

$$FP_{B,t} = \{ \pm M \cdot B^E : M, E \text{ ganze Zahlen,} \\ E \text{ beliebig, } B^{t-1} \leq M < B^t \text{ oder } M = 0 \}$$

- B heißt Basis (eine ganze Zahl > 1)
- t ist die Anzahl der Stellen (ganze Zahl > 0)
- $M \geq B^{t-1}$ bedeutet, dass die erste Stelle $\neq 0$ sein muss

M ist die Mantisse,
 E ist der Exponent

- Beispiel:

Basis $B = 10$, Mantissenlänge $t = 2$

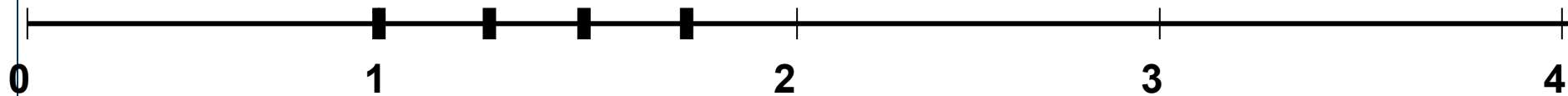
$$B^{t-1} \leq M < B^t : 10, 11, 12, \dots, 19, 20, 21, \dots, 91, 92, \dots, 99$$

- $E = 0$: $10, 11, 12, \dots, 19, 20, 21, \dots, 91, 92, \dots, 99$
- $E = 1$: $100, 110, 120, \dots, 190, 200, 210, \dots, 910, 920, \dots, 990$
- $E = -1$: $1.0, 1.1, 1.2, \dots, 1.9, 2.0, 2.1, \dots, 9.1, 9.2, \dots, 9.9$

- Beispiel:

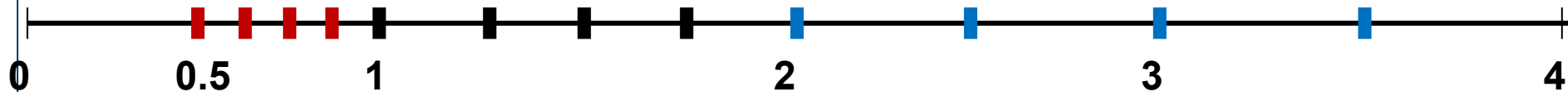
Basis $B = 2$, Mantissenlänge $t = 3$

- $E = -2 : 1.00, 1.01, 1.10, 1.11$ (binär)
 $1.0, 1.25, 1.5, 1.75$ (dezimal)



- $E = -3 : 0.100, 0.101, 0.110, 0.111$ (binär)

- $E = -1 : 10.0, 10.1, 11.0, 11.1$ (binär)



- Logarithmische Verteilung!

- Definition:

Die Menge der **Maschinenzahlen** ist definiert als

$$MFP_{B,t,\alpha,\beta} = \{ g \in FP_{B,t} : \alpha \leq E \leq \beta \}$$

- Hier ist jetzt auch der Exponent eingeschränkt.
 - ▶ B, t, α, β sind durch die Implementierung (also den Hersteller des Computers) festgelegt. Sie werden nie explizit gespeichert.
 - ▶ Für jede Maschinenzahl $\pm M \cdot B^E$ wird Vorzeichen und die Zahlen M (Mantisse) und E (Exponent) gespeichert.

- Im IEEE-754 Format (von 1985, heute fast universell)

(Details zB http://de.wikipedia.org/wiki/IEEE_754)

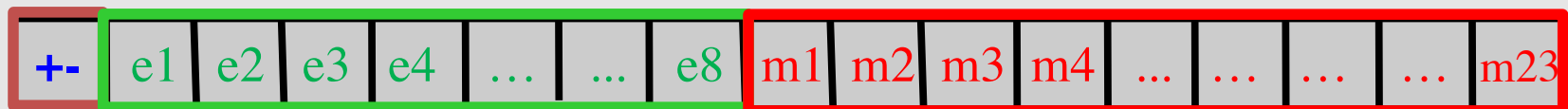
- ▶ $B=2$
- ▶ Single precision (4 Bytes, 32 Bit): $t=24$ (**nur 23 Bits!**)

normalisierte Mantisse
$$m = 1 + \frac{m_1}{2} + \frac{m_2}{4} + \frac{m_3}{8} + \dots + \frac{m_{23}}{2^{23}}$$

Single Precision

Exponent

Mantisse



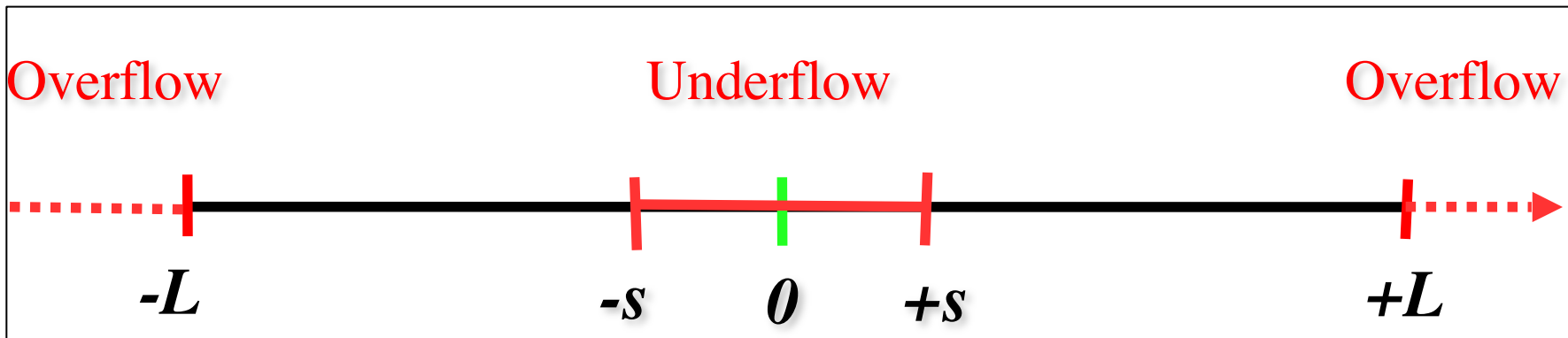
- ▶ Double precision (8 Bytes, 64 Bit): $t=53$ (**nur 52 Bits!**)

- Kleinste und größte positive Zahl:

$$E_{\max} = 254 - 127 = 127, m_{\max} = 1.111...1: L \approx 2 \cdot 2^{127} \approx 3.40 \cdot 10^{38}$$

$$E_{\min} = 1 - 127 = -126, m_{\min} = 1.000...0: s \approx 1 \cdot 2^{-126} \approx 1.18 \cdot 10^{-38}$$

	t	Alpha	Beta	Resolution	s	L
Single	24	-149	104	1,19E-007	1,18E-038	3,40E+038
Double	53	-1074	971	2,22E-016	2,23E-308	2 ⁺¹⁰²⁴
Extend.	64	-16445	16320	1,08E-019	2 ⁻¹⁶³⁸²	2 ⁺¹⁶³⁸⁴



- Einfache Genauigkeit (single) entspricht etwa 7-8 Dezimalstellen, bei doppelter Genauigkeit (double) sind etwa 15-16 Dezimalstellen gespeichert.
- Sonderfälle (Exponent $E=11111111$ und $E = 00000000$)
 - ▶ Die Zahl 0
 - ▶ NaN (not a number): unbestimmter Wert, implementiert als „quiet“ (vererbt sich stillschweigend) oder „signaling“ (löst Alarm aus)
 - ▶ Exponentenüberlauf: $+\infty$ (Absolutwert der Zahl $> L$)
 - ▶ Exponentenunterlauf: $+0$ (Absolutwert der Zahl $< s$)
 - ▶ ...
 - ▶ (siehe auch: Webseite zu IEEE-Standard)

- Gleitpunktzahlen sind die Repräsentanten
- Quantisierung erfolgt durch Runden
 - Aufrunden: nächst größere Gleitpunktzahl
 - Abrunden: nächst kleinere Gleitpunktzahl
 - Exaktes/kaufmännisches Runden: nächst gelegene Gleitpunktzahl
 Was tun bei Mehrdeutigkeit?

- Achtung: Es entstehen Rundungsfehler

- **Absoluter Fehler** kann sehr groß werden
- **Relativer Fehler** ist beschränkt durch die Auflösung:

$$\rho = \frac{1}{B^{t-1}} = \frac{1}{2^{23}} \approx 1.19 \cdot 10^{-7}$$

- (das ist der maximale **relative** Abstand zweier benachbarter Zahlen Gleitpunktzahlen)

- Alle guten Implementierungen (z.B. gem IEEE-Standard) der Gleitpunktarithmetik arbeiten so, dass das Ergebnis einer Gleitpunktoperation $\dot{*}$ zu folgendem Prozess gleichwertig ist:
 - Berechne das exakte Resultat
 - Runde das Resultat zur nächsten Maschinenzahl (entsprechend eines der Rundungsmodi)
- Die Auflösung $\rho = \frac{1}{B^{t-1}}$ (rel. Abstand zweier Gleitpunktzahlen, Maschinengenauigkeit) ist eine Schranke für den relativen Rundungsfehler
- Es gilt dann die starke Hypothese: Es gibt ein $\tilde{\varepsilon} = O(\rho)$ so dass

$$a \dot{*} b = (a * b) \cdot (1 + \varepsilon) \quad \text{mit} \quad |\varepsilon| = |\varepsilon(*, a, b)| \leq \tilde{\varepsilon}$$

- Gleitpunktarithmetik ist fehlerbehaftet
- Zusätzlich: Über- / Unterlauf (kleineres Problem)
- Hinsichtlich des relativen Fehlers sind
 - * und / gutartig;
 - +, - gefährlich, weil Auslöschung auftreten kann
- Wohlbekannten Rechengesetze gelten nur bedingt:
 - kommutativ OK,
 - assoziativ, distributiv hingegen **nicht!**
- Beispiel (Rechnung mit 6 Dezimalstellen)

$$(1 + 1\,000\,000) - 1\,000\,000 = 1\,000\,000 - 1\,000\,000 = 0$$

$$1 + (1\,000\,000 - 1\,000\,000) = 1 - 0 = 1$$
- Frage: Wie pflanzen sich (Rundungs-)Fehler fort?

- Die Anfälligkeit gegenüber Störungen (Fehlern in Eingabedaten, Rundungsfehler,) eines numerischen Verfahrens wird durch zwei Konzepte beschrieben:
 - ▶ **Kondition**
 - ▶ **Stabilität**
- **Kondition ist eine Eigenschaft des Problems !**
- **Stabilität ist eine Eigenschaft des Algorithmus (des Lösungsverfahrens)!**
- Formal: ideal: Eingabe $x \rightarrow$ Ergebnis $F(x)$
 real: Eingabe $\tilde{x} \rightarrow$ Ergebnis $\tilde{F}(\tilde{x})$

- Formal: ideal: Eingabe $x \rightarrow$ Ergebnis $F(x)$
real: Eingabe $\tilde{x} \rightarrow$ Ergebnis $\tilde{F}(\tilde{x})$
- Ein Problem ist **gut konditioniert**, wenn (bei exakter Rechnung) Fehler in den Eingangsdaten nicht verstärkt

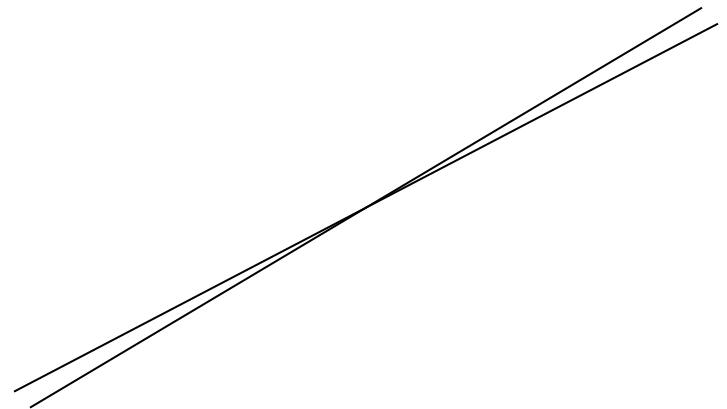
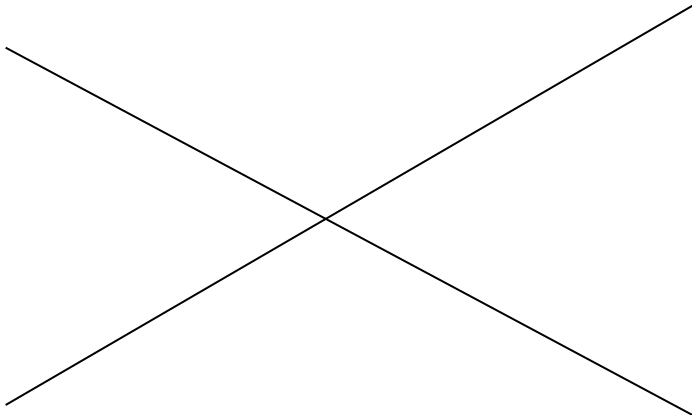
werden:
$$\frac{|F(\tilde{x}) - F(x)|}{|F(x)|} \approx \frac{|\tilde{x} - x|}{|x|}$$

- Ein Algorithmus ist (vorwärts-) **stabil**, wenn die Rechenfehler nicht

akkumulieren:
$$\frac{|\tilde{F}(y) - F(y)|}{|F(y)|} \ll 1$$

Berechnung des Schnittpunktes zweier Geraden

- Geraden „fast orthogonal“: Gute Kondition, d.h. leichtes „Wackeln“ an den Geraden verändert den Schnittpunkt kaum
- Geraden „fast parallel“: Schlechte Kondition, denn auch ein geringes „Wackeln“ kann den Schnittpunkt drastisch verändern.



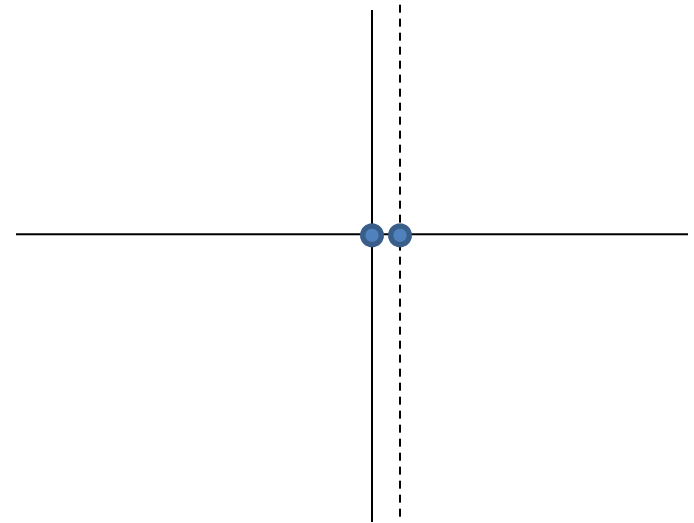
- Bei schlecht konditionierten Problemen ist die einfache Definition der Vorwärts-Stabilität nicht sachgemäß:
- Eingabewerte x , exaktes Ergebnis sei $y = F(x)$
- Das tatsächlich berechnete Ergebnis y' heißt **akzeptabel** (wenn es als exaktes Ergebnis zu nur leicht gestörten Eingabewerten verstanden werden kann,
also $y' = F(x')$ mit $\|x' - x\| \leq \varepsilon$)
- Ein Algorithmus, der akzeptable Ergebnisse liefert heißt rückwärts-stabil
- Bei dieser Definition müssen die Ergebnisse nicht unbedingt sehr genau sein.
- Bei einem schlecht konditionierten Problem können auch hundsmiserabel falsche Werte „akzeptabel“ sein
- Außer den Rundungsfehlern müssen noch andere Verfahrensfehler mit berücksichtigt werden:
 - Diskretisierungsfehler
 - Iterationsfehler

- **Gute Kondition:**
 - ▶ Kleine Eingabestörungen führen zu kleinen Änderungen der Ergebnisse, Eingabestörungen sind unkritisch
 - ▶ Es lohnt sich einen guten Algorithmus zu entwickeln (der die gute Kondition des Problems ausnützt)

- **Schlechte Kondition**
 - ▶ Selbst kleine Eingabestörungen können zu großen Störungen der Ergebnisse führen, Lösung reagiert empfindlich auf kleine Störungen der Eingabe
 - ▶ Daran kann auch ein guter Algorithmus nichts ändern!
 - ▶ Aus Perspektive des Anwenders:
 - ★ Schwierige Probleme, die im Extremfall gar nicht gelöst werden können
 - ★ Oft ein Zeichen, dass die Aufgabenstellung falsch oder unsinnig ist (dies kritisch hinterfragen!)

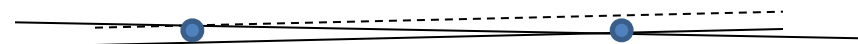
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \epsilon \\ 1 \end{bmatrix}$$

$$x = \epsilon, y = 1$$



$$\begin{bmatrix} 0.00000001 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 + \epsilon \\ 1 \end{bmatrix}$$

$$x = 10000000\epsilon, y = 1$$



- In beiden Fällen hat das ungestörte Problem (d.h. $\varepsilon = 0$) die exakte Lösung $x = 0$, $y = 1$
- Im ersten Fall hat das gestörte Problem die Lösung $x = \varepsilon$, $y = 1$, d.h. eine kleine Störung der Daten bewirkt eine kleine Störung des Ergebnisses, das Problem ist **gut konditioniert**
- Im zweiten Fall hat das gestörte Problem die Lösung $x = 10000000\varepsilon$, $y = 1$, d.h. eine kleine Störung der Daten bewirkt eine enorme Störung des Ergebnisses, das Problem ist **schlecht konditioniert**.

- Die arithmetischen Grundoperationen sind stabil
- Aber:
Die Hintereinanderausführung stabiler Operationen ist nicht automatisch stabil
(sonst wäre ja alles stabil, was aus den Grundoperationen zusammengesetzt wird)

- Rechnung mit 3 Nachkommastellen
- Problemstellung:
 - ★ Eingabe Werte a, b
 - ★ Gesucht Nullstellen des Polynoms: $(x - a)(x - b) = 0$
- Gegeben (Input): $a = 1/100 = 0.01$, $b = -100$

Algorithmus 1:

- Ergebnis: (Trivialerweise) sind die Nullstellen a und b
- egal mit wie vielen Stellen
- Ergebnis so genau wie die Eingabe
- Resultat „akzeptabel“
- „Algorithmus“ stabil

• Algorithmus 2

- ▶ Multipliziere das Polynom aus: $p(x) := x^2 - (a+b)x + ab = 0$
- ▶ Setze a und b ein: $x^2 + 99.99x - 1 = 0$, wird gerundet zu $x^2 + 100x - 1 = 0$
- ▶ Wende Formel für Nullstellen an (mit 3-stelliger Genauigkeit)

$$\begin{aligned}
 x_{1,2} &= \frac{-100 \pm \sqrt{100^2 - 4}}{2} \\
 &= -50 \pm \frac{\sqrt{10000 - 4}}{2} \\
 &\approx -50 \pm \sqrt{10000}/2 \\
 &= \{0; -100\}
 \end{aligned}$$

- ▶ Man erhält $x_1 = 0 \neq 0.01$, also 100% Fehler
- ▶ Das ist im Sinne unserer Definition kein akzeptables Resultat
- ▶ Für $x = 0$ gilt $p(x) = -1$, also überhaupt keine Nullstelle.

- **Fazit:**
Sogar „Selbstverständlichkeiten“

z.B. die wohlbekannte Lösungsformel $x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

für die quadratische Gleichung $ax^2 + bx + c = 0$

kann (für manche Eingabewerte) zu einem instabilen Algorithmus führen!

Aufgabe: Löse lineares Gleichungssystem $Ax = b$

- mit LR-Zerlegung **ohne** Pivot : evt. instabil
- mit LR-Zerlegung mit Pivot : stabil
- mit QR-Zerlegung : stabil

- Spezialfall:
Eingabe: reelle Zahl $x \rightarrow$ Ausgabe reelle Zahl $y = F(x)$

- **Kondition κ :**

Um welchen Faktor verstärkt sich der relative Fehler (bei exakter Rechnung).

$$\frac{|\tilde{y} - y|}{|y|} \leq \kappa \cdot \frac{|\tilde{x} - x|}{|x|}$$

$$(x \approx \tilde{x}, y = F(x), \tilde{y} = F(\tilde{x}))$$

- Gilt auch allgemein für $F : \mathbb{R}^m \rightarrow \mathbb{R}^n$. Man muss nur den Betrag $|\cdot|$ durch eine Norm $\|\cdot\|$ ersetzen (z.B. Euklidische Norm)

$$\frac{\|\tilde{y} - y\|}{\|y\|} \leq \kappa \cdot \frac{\|\tilde{x} - x\|}{\|x\|}$$

- Ein Problem wird oft in mehrere Teilprobleme zerlegt
→ mehrer Lösungsschritte
- Man kann die Gesamtkondition eines solchen zusammengesetzten Problems $F(x) = G(H(x)) = G \circ H(x)$ per Kettenregel auf die Kondition der Teilprobleme zurückführen:

$$\text{cond}(F, x) = \text{cond}(G \circ H, x) \leq \text{cond}(G, H(x)) \cdot \text{cond}(H, x)$$

- **Vorsicht:** Dabei kann „ \leq “ durch aus „ \ll “ sein!

Das heißt, das Gesamtproblem ist zwar gut konditioniert, aber die Teilprobleme sind schlecht konditioniert!

- Problem: Löse $Ax = b$ mit (invertierbarer) $n \times n$ –Matrix A und rechter Seite b .
- input: rechte Seite $b \rightarrow$ output: Lösung x
- Die Kondition dieses Problems ist durch die **Konditionszahl der Matrix A** gegeben:

$$\kappa(A) = \frac{\max \{ \|Ay\| : \|y\| = 1 \}}{\min \{ \|Ay\| : \|y\| = 1 \}}$$

- Falls $Ax = b$ und $A\tilde{x} = \tilde{b}$ gilt dann
$$\frac{\|\tilde{x} - x\|}{\|x\|} \leq \kappa(A) \cdot \frac{\|\tilde{b} - b\|}{\|b\|}$$

• Konditionszahl: Beispiel

‣ Es gilt $\kappa(AB) \leq \kappa(A) \cdot \kappa(B)$ u.U. aber „ \ll “ !

‣ Beispiel:

$$M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ -99 & -200 & 50 & 40 \\ 900 & 50 & -50 & -80 \\ 100 & -10 & 90 & 150 \end{bmatrix} \text{ hat } \kappa(M) \approx 1176.1$$

‣ LR-Zerlegung **ohne** Pivotisierung:

$$M = L_1 R_1 : \kappa(L_1) \approx 1.108 \cdot 10^8, \kappa(R_1) \approx 7.188 \cdot 10^5$$

‣ LR-Zerlegung **mit** Pivotisierung:

$$M = L_2 R_2 : \kappa(L_2) \approx 1.1833, \kappa(R_2) \approx 1161.5$$

‣ QR-Zerlegung:

$$M = QR_3 : \kappa(Q) = 1, \kappa(R_3) = \kappa(M)$$

LR:
ohne Pivot
instabil (siehe
aber Spezial-
Literatur)

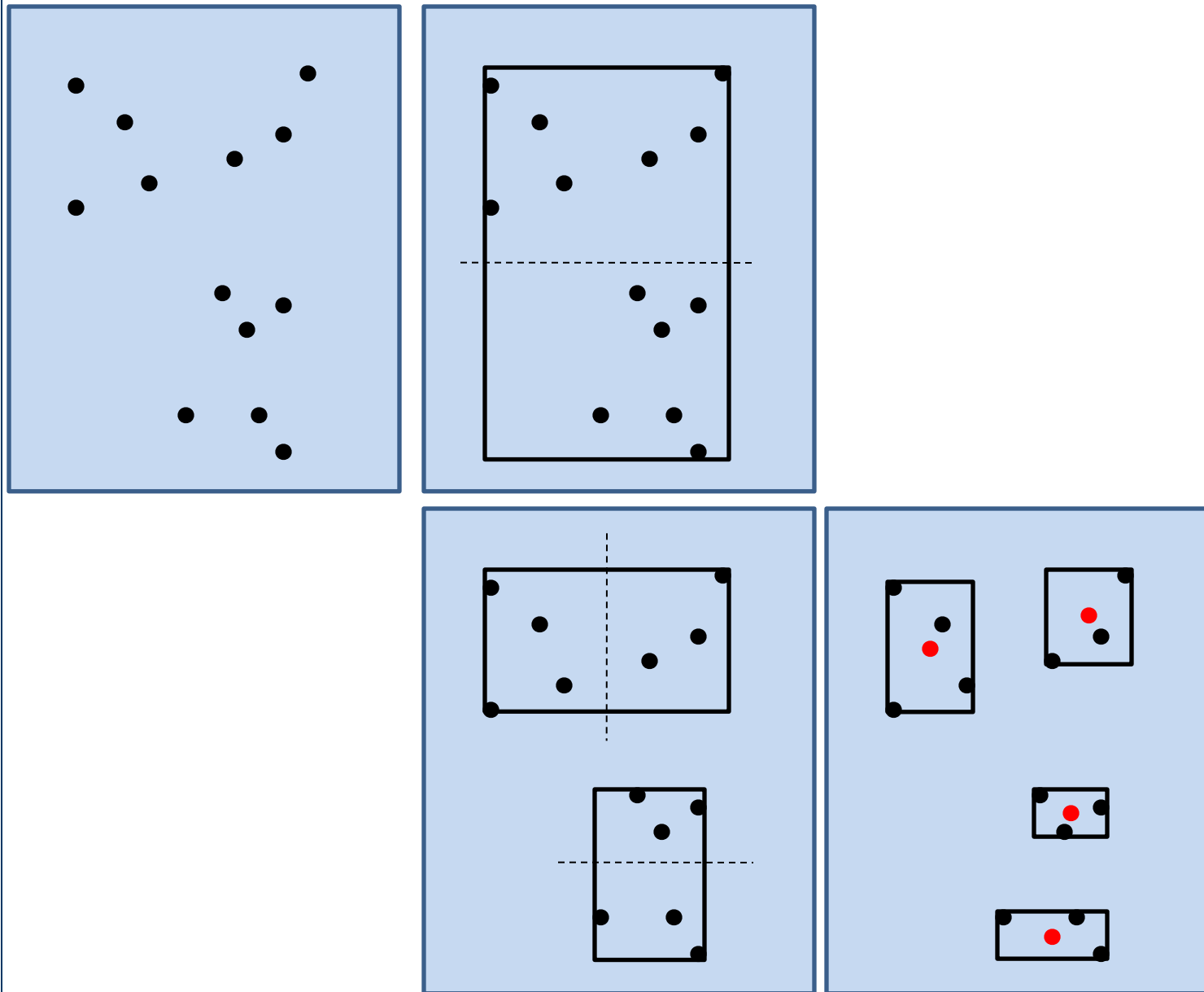
mit Pivot
stabil

QR stabil

- Ungenauigkeit der Eingabewerte
 - Diskretisierungs-/Quantisierungsfehler
 - Rundungsfehler
 - Iterationsfehler
 - Linearisierungsfehler
 - Modellierungsfehler
 - ...
-
- Ist es da nicht überraschend, dass man überhaupt irgend etwas zuverlässig berechnen kann?

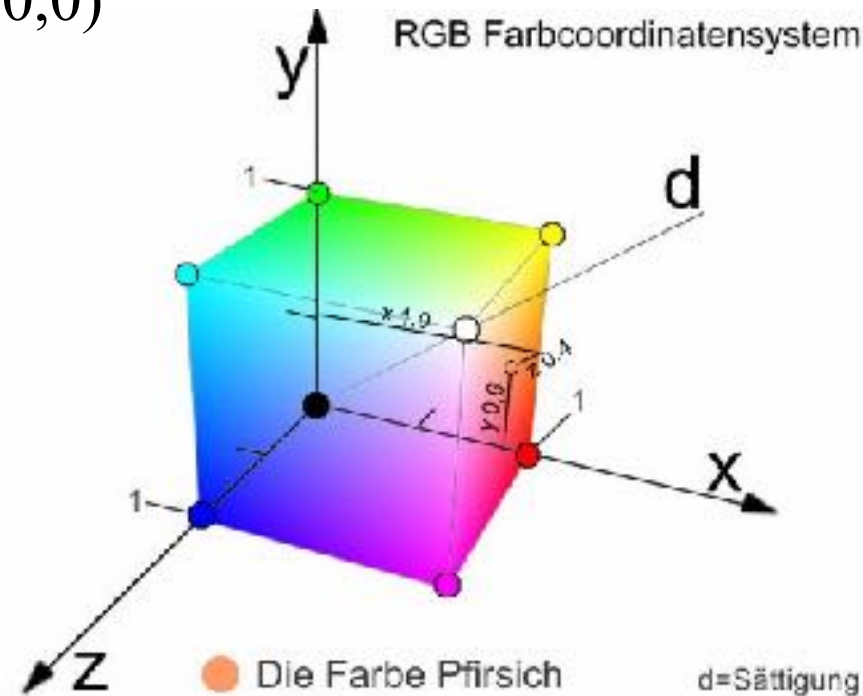
- Diskretisierungsfehler, Rundungsfehler
- Gleitpunktarithmetik
 - Auflösung bzw. Maschinengenauigkeit
 - Unterschiede zw. „Punkt- und Strich-Rechnung“
 - Rechengesetze gelten nur eingeschränkt
- Rundungsfehler-Fortplanzung
- Kondition und Stabilität
 - Probleme sind gut oder schlecht konditioniert
 - Algorithmen sind stabil oder instabil
- Kondition linearer Gleichungssysteme, Konditionszahl

- Der **median-cut Algorithmus** zur Quantisierung einer Punktwolke (= aller im Bild vorkommenden RGB-Werte)
- Teilschritte:
 1. Finde best-fittende bounding box (→ Rechteck bzw. Quader)
 2. Teile Quader an längster Kante so dass beide Teile gleich viele Punkte enthalten (→ *Median*)
 3. 1. und 2. Schritt für jeden Halbquader rekursiv wiederholen
- Rekursiver Algorithmus; STOP falls gewünschte Anzahl erreicht ist (z.B. 256 nach 8 Schritten)
- Wahl der Repräsentanten
 - Mittelpunkt des Quaders
 - Mittelwert der enthaltenen Punkte

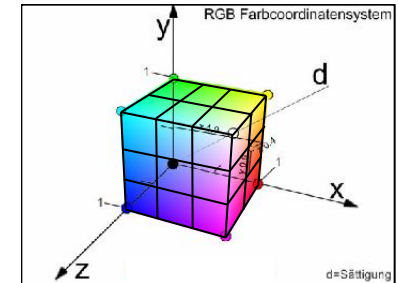


- Der RGB-Würfel. (**R**ot-**G**rün-**B**lau)
- Farbwert $(r,g,b) \in [0,1]^3$ bzw. diskretisiert: $(r,g,b) \in [0..255]^3$
 r = Rot, g = Grün, b = Blau (16 Mio Farbwerte!)
- Farbwerte werden aus drei Grundfarben (**additiv**) gemischt:
 weiß = $(1,1,1)$, schwarz = $(0,0,0)$

- $(1,0,0)$ $(0,1,0)$ $(0,0,1)$
 rot grün blau
- $(0,1,1)$ $(1,0,1)$ $(1,1,0)$
 cyan magenta gelb



- Der RGB-Würfel. Farbwert $(r,g,b) \in [0,1]^3$



24bpp
bit per pixel

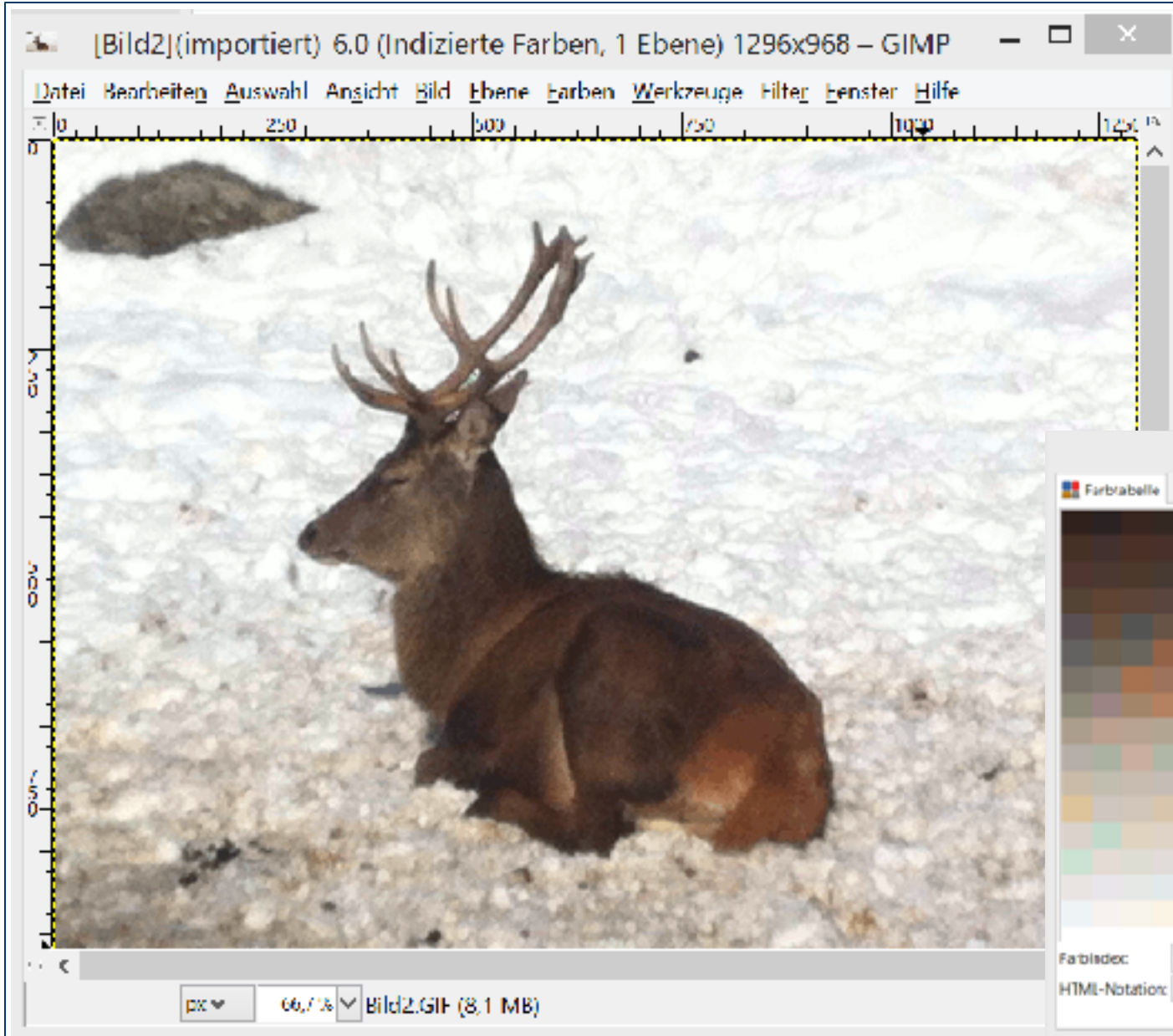


8bpp

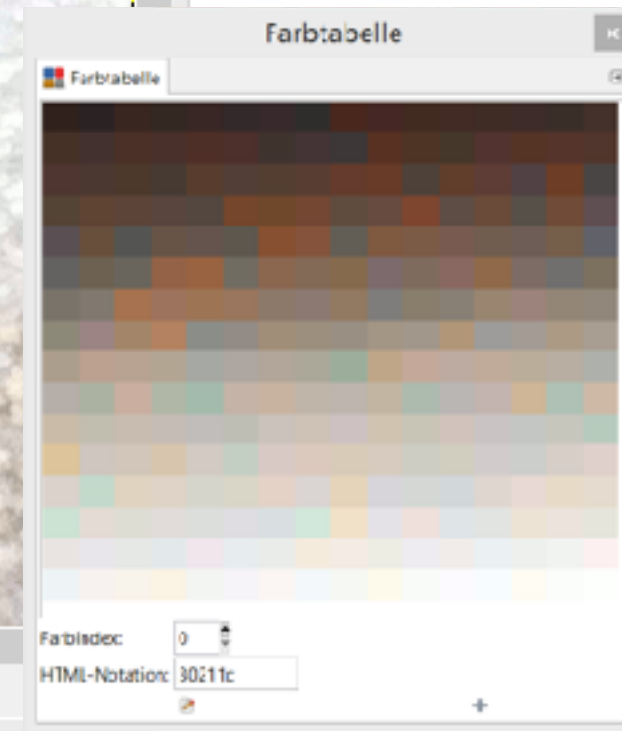
- Rot : 3 bit
- Grün: 3 bit
- Blau: 2 bit



4bpp



484 kB



- Alternative bei nicht gleichverteilten Werten w_j nicht äquidistant und Repräsentant $r_j \in [w_j, w_{j+1}]$
- Annahme: die Werte sind gemäß einer Dichtefunktion $p(r)$ verteilt

$$P(r \leq R) = \int_{-\infty}^R p(\rho) d\rho$$

- Ist r Realisierung einer Zufallsvariablen mit Dichte $p(\rho)$, die die Wahrscheinlichkeits-Verteilung der Punkte beschreibt

$$p(\rho) = 0 \text{ für } \rho < r_{\min} \text{ oder } \rho > r_{\max}$$

der mittlere quadratische Fehler ist gegeben durch

$$\varepsilon = \sum_{j=0}^{L-1} \int_{w_j}^{w_{j+1}} (\rho - r_j)^2 p(\rho) d\rho$$

$$\varepsilon = \sum_{j=0}^{L-1} \int_{w_j}^{w_{j+1}} (\rho - r_j)^2 p(\rho) d\rho$$

Theorem: Obiges Funktional wird minimiert falls

1. $w_j = \frac{r_{j-1} + r_j}{2}$ (nearest neighbor) und
2. $r_j = \frac{\int_{w_j}^{w_{j+1}} \rho p(\rho) d\rho}{\int_{w_j}^{w_{j+1}} p(\rho) d\rho}$ (Schwerpunkts-Bedingung)

- Mehrdimensionaler Fall: Zu approximieren
 $\mathbf{r} = [r_1, \dots, r_n] \in \mathbb{R}^n, \quad w_{i,\min} \leq r_i \leq w_{i,\max}, i \in [1, \dots, n]$
- Beispiel 2D: Wertebereich

$$W := [w_{1,\min}, w_{1,\max}] \times [w_{2,\min}, w_{2,\max}]$$
- Quantisierung in Rechtecke i.A. nicht optimal!
- L Vektoren \mathbf{r}_i als Repräsentanten (Codewörter) für disjunkte Teilgebiete W_i
- Quantisierungsfehler ($p(\mathbf{r})$ ist r -dim. Dichte)

$$\varepsilon = \sum_{i=1}^L \int_{W_i} \|\mathbf{r} - \mathbf{r}_i\|^2 p(\mathbf{r}) d\mathbf{r}$$

Satz:

1. Das obige Fehlerfunktional wird für gegebene Repräsentanten \mathbf{r}_i , $i = 1, \dots, L$ minimiert wenn für

$$\mathbf{r} \in W_i \text{ gilt } \|\mathbf{r} - \mathbf{r}_i\| \leq \|\mathbf{r} - \mathbf{r}_j\| \quad \forall j \neq i$$

2. Ist eine Zerlegung von W in disjunkte W_i gegeben, dann gilt für die Repräsentanten

$$\mathbf{r}_i = \frac{\int_{W_i} \mathbf{r} p(\mathbf{r}) d\mathbf{r}}{\int_{W_i} p(\mathbf{r}) d\mathbf{r}}$$

Die Berechnung der optimalen Vektorquantisierung erfolgt iterativ!

- Gegeben:

- ▶ Menge von M n -dim. Trainingsvektoren $T = \{\mathbf{x}_1, \dots, \mathbf{x}_M\}$
- ▶ Menge von L n -dim. Codevektoren (Repräsentanten) $C = \{\mathbf{r}_1, \dots, \mathbf{r}_L\}$
- ▶ Zu den Codevektoren gehörige Partition $P = \{P_1, \dots, P_L\}$ des \mathbb{R}^n
- ▶ Funktion $Q: \mathbb{R}^n \rightarrow \mathbb{R}^n$, die jedem Trainingsvektor einen Codevektor zuordnet:
 Falls ein Trainingsvektor \mathbf{x}_i in P_j liegt, wird er durch $Q(\mathbf{x}_i) = \mathbf{r}_j$ approximiert.

- Ziel: Minimiere den Fehler

$$\varepsilon = \frac{1}{Mn} \sum_{i=1}^M \|\mathbf{x}_i - Q(\mathbf{x}_i)\|^2$$

- Optimalitätskriterien

- ▶ *Nearest Neighbor Condition* (Voronoi-Region)

$$P_i = \{\mathbf{x} : \|\mathbf{x} - \mathbf{r}_i\|^2 \leq \|\mathbf{x} - \mathbf{r}_k\|^2 \quad \forall k = 1, 2, \dots, L\}$$

(Schnitt von Halbräumen zu Mittelsenkrechten)

- ▶ Schwerpunkts-Bedingung (*Centroid Condition*)

$$\mathbf{r}_i = \frac{1}{|M_i|} \sum_{m \in M_i} \mathbf{x}_m, \quad i = 1, 2, \dots, L$$

- ▶ wobei $M_i = \{m : \mathbf{x}_m \in P_i\}$

- ▶ Startlösung ($P_1 = \mathbb{R}^n$, $M_1 = \{1, 2, \dots, M\}$)

$$\mathbf{r}_1^* = \frac{1}{M} \sum_{i=1}^M \mathbf{x}_i \quad \varepsilon^* = \frac{1}{Mn} \sum_{i=1}^M \|\mathbf{x}_i - \mathbf{r}_1^*\|^2$$

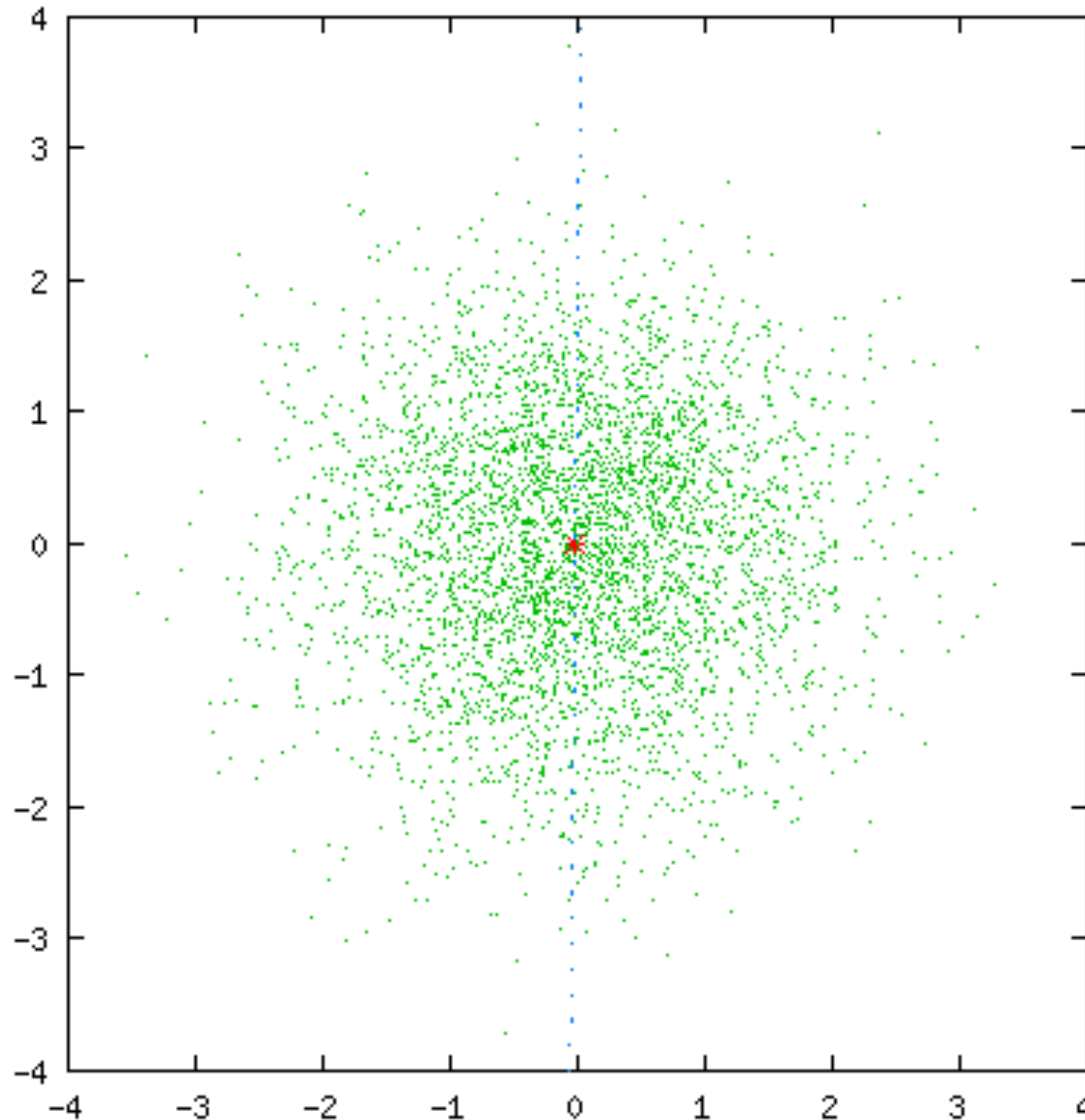
Lloyds-Algorithmus - *k-means clustering*

1. Berechne Startlösung ($L=1$, $i=1$): $P_1 = \mathbb{R}^n$, r_1 Schwerpunkt
2. Splitting der Codebook-Vektoren

$$r_i^{(0)} = (1 + \beta) r_i *$$

$$r_{i+L}^{(0)} = (1 - \beta) r_i *$$

3. Zuordnung der Samples zu Codebook-Vektoren
(d.h. Bestimmung der P_i und M_i)
4. Berechnung der neuen Codebook-Vektoren
(d.h. Bestimmung der r_i)
5. solange Abstand alte-neue Codebook-Vektoren größer als Schranke, gehe zu 3.
6. falls gewünschte Anzahl der Codebook-Vektoren noch nicht erreicht gehe zu 2.



- Uniforme vs nicht uniforme Quantisierung
- Gleitpunktzahlen bzw. Maschinenzahlen
 - IEEE Standard
 - Rundung
- Median Cut Algorithmus
- Least square minimizer
- Vektorquantisierung
 - Lloyd Algorithmus (*k-means clustering*)