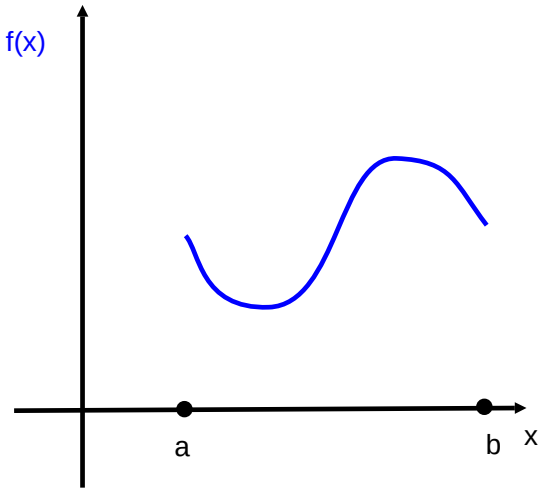
Polynominterpolation

Polynominterpolation

Sei $f : [a, b] \to \mathbb{R}$ eine Funktion.

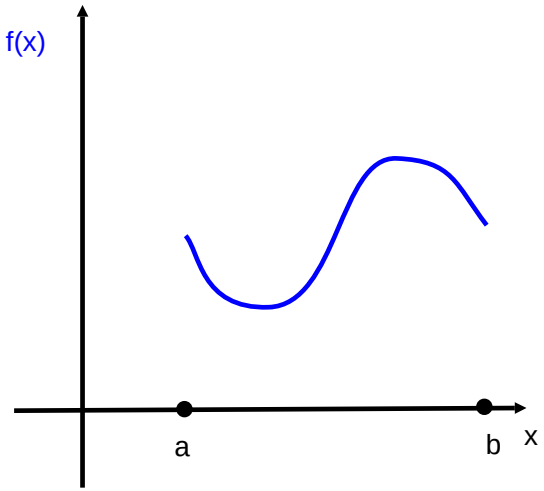
Polynominterpolation

Sei $f : [a, b] \longrightarrow \mathbb{R}$ eine Funktion.



Polynominterpolation

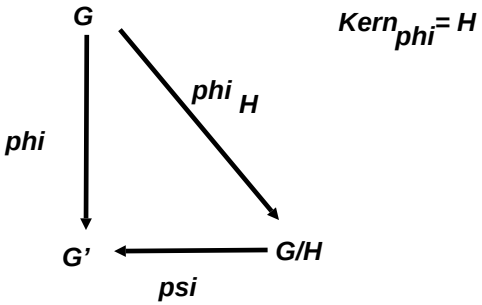
Sei $f : [a, b] \longrightarrow \mathbb{R}$ eine Funktion.



Unter Polynominterpolation versteht man die Lösung der Aufgabe, ein Polynom P zu finden,

Polynominterpolation

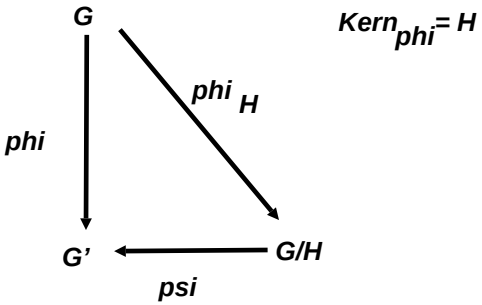
Sei $f : [a, b] \longrightarrow \mathbb{R}$ eine Funktion.



Unter Polynominterpolation versteht man die Lösung der Aufgabe, ein Polynom P zu finden, s.d. $\sup_{x\in[a,b]} |P(x) - f(x)|$ möglich klein ist.
Taylorentwicklung (Ana I) ist eine mögliche Polynominterpolation;

Polynominterpolation

Sei $f : [a, b] \to \mathbb{R}$ eine Funktion.



Unter Polynominterpolation versteht man die Lösung der Aufgabe, ein Polynom P zu finden, s.d. $\sup_{x\in[a,b]} |P(x) - f(x)|$ möglich klein ist.
Taylorentwicklung (Ana I) ist eine mögliche Polynominterpolation;
wir werden auch andere diskutieren

Ein Vorschlag:

Ein Vorschlag: Wähle $n+1$ verschiedenen Punkten $x_0, \ldots, x_n$ auf $[a, b]$

Ein Vorschlag: Wähle $n+1$ verschiedenen Punkten $x_0, \ldots, x_n$ auf $[a, b]$ und suche ein Polynom P,

Ein Vorschlag: Wähle $n+1$ verschiedenen Punkten $x_0, \ldots, x_n$ auf $[a, b]$ und suche ein Polynom P, $Grad(P) \leq n$,

Ein Vorschlag: Wähle $n+1$ verschiedenen Punkten $x_0, ..., x_n$ auf $[a, b]$ und suche ein Polynom P, $Grad(P) \leq n$, s.d. $P(x_i) = f(x_i)$.

Ein Vorschlag: Wähle $n+1$ verschiedenen Punkten $x_0, ..., x_n$ auf $[a, b]$ und suche ein Polynom P, $Grad(P) \leq n$, s.d. $P(x_i) = f(x_i)$.
Hausaufgabe 3 Blatt 9: Solches Polynom existiert und ist eindeutig.

Ein Vorschlag: Wähle $n+1$ verschiedenen Punkten $x_0, ..., x_n$ auf $[a, b]$ und suche ein Polynom P, $Grad(P) \leq n$, s.d. $P(x_i) = f(x_i)$.
Hausaufgabe 3 Blatt 9: Solches Polynom existiert und ist eindeutig. Tatsächlich, die Bediengung $P(x_i) = f(x_i)$

Ein Vorschlag: Wähle $n+1$ verschiedenen Punkten $x_0, ..., x_n$ auf $[a, b]$ und suche ein Polynom P, $Grad(P) \leq n$, s.d. $P(x_i) = f(x_i)$.
Hausaufgabe 3 Blatt 9: Solches Polynom existiert und ist eindeutig. Tatsächlich, die Bediengung $P(x_i) = f(x_i)$ ist das folgende System von Gleichungen

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n & = & f(x_0) \\ \\ \end{cases}$$

Ein Vorschlag: Wähle $n+1$ verschiedenen Punkten $x_0, ..., x_n$ auf $[a, b]$ und suche ein Polynom P, $Grad(P) \leq n$, s.d. $P(x_i) = f(x_i)$.
Hausaufgabe 3 Blatt 9: Solches Polynom existiert und ist eindeutig. Tatsächlich, die Bediengung $P(x_i) = f(x_i)$ ist das folgende System von Gleichungen

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n &= f(x_0) \\ & \vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n &= f(x_n) \end{cases}$$

Ein Vorschlag: Wähle $n+1$ verschiedenen Punkten $x_0, ..., x_n$ auf $[a, b]$ und suche ein Polynom P, $Grad(P) \leq n$, s.d. $P(x_i) = f(x_i)$.
Hausaufgabe 3 Blatt 9: Solches Polynom existiert und ist eindeutig. Tatsächlich, die Bediengung $P(x_i) = f(x_i)$ ist das folgende System von Gleichungen

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n &= f(x_0) \\ & \vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n &= f(x_n) \end{cases}$$

Das System besteht aus $n+1$ linearen Gleichungen

Ein Vorschlag: Wähle $n+1$ verschiedenen Punkten $x_0, ..., x_n$ auf $[a, b]$ und suche ein Polynom P, $Grad(P) \leq n$, s.d. $P(x_i) = f(x_i)$.
Hausaufgabe 3 Blatt 9: Solches Polynom existiert und ist eindeutig. Tatsächlich, die Bediengung $P(x_i) = f(x_i)$ ist das folgende System von Gleichungen

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n &= f(x_0) \\ & \vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n &= f(x_n) \end{cases}$$

Das System besteht aus $n+1$ linearen Gleichungen mit $n+1$ Unbekanneten $a_0, ..., a_n$.

Ein Vorschlag: Wähle $n+1$ verschiedenen Punkten $x_0, ..., x_n$ auf $[a, b]$ und suche ein Polynom P, $Grad(P) \leq n$, s.d. $P(x_i) = f(x_i)$.
Hausaufgabe 3 Blatt 9: Solches Polynom existiert und ist eindeutig. Tatsächlich, die Bediengung $P(x_i) = f(x_i)$ ist das folgende System von Gleichungen

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n &= f(x_0) \\ & \vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n &= f(x_n) \end{cases}$$

Das System besteht aus $n+1$ linearen Gleichungen mit $n+1$ Unbekanneten $a_0, ..., a_n$. Deren Matrixform ist

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ & & & & \end{pmatrix}$$

Ein Vorschlag: Wähle $n+1$ verschiedenen Punkten $x_0, ..., x_n$ auf $[a, b]$ und suche ein Polynom P, $Grad(P) \leq n$, s.d. $P(x_i) = f(x_i)$.
Hausaufgabe 3 Blatt 9: Solches Polynom existiert und ist eindeutig. Tatsächlich, die Bediengung $P(x_i) = f(x_i)$ ist das folgende System von Gleichungen

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n &= f(x_0) \\ & \vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n &= f(x_n) \end{cases}$$

Das System besteht aus $n+1$ linearen Gleichungen mit $n+1$ Unbekanneten $a_0, ..., a_n$. Deren Matrixform ist

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ & & & \vdots & \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix}$$

Ein Vorschlag: Wähle $n+1$ verschiedenen Punkten $x_0, ..., x_n$ auf $[a, b]$ und suche ein Polynom P, $Grad(P) \leq n$, s.d. $P(x_i) = f(x_i)$.
Hausaufgabe 3 Blatt 9: Solches Polynom existiert und ist eindeutig. Tatsächlich, die Bediengung $P(x_i) = f(x_i)$ ist das folgende System von Gleichungen

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n &= f(x_0) \\ & \vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n &= f(x_n) \end{cases}$$

Das System besteht aus $n+1$ linearen Gleichungen mit $n+1$ Unbekanneten $a_0, ..., a_n$. Deren Matrixform ist

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ & & & \vdots & \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix}$$

Ein Vorschlag: Wähle $n+1$ verschiedenen Punkten $x_0, ..., x_n$ auf $[a, b]$ und suche ein Polynom P, $Grad(P) \leq n$, s.d. $P(x_i) = f(x_i)$.
Hausaufgabe 3 Blatt 9: Solches Polynom existiert und ist eindeutig. Tatsächlich, die Bediengung $P(x_i) = f(x_i)$ ist das folgende System von Gleichungen

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n &= f(x_0) \\ & \vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n &= f(x_n) \end{cases}$$

Das System besteht aus $n+1$ linearen Gleichungen mit $n+1$ Unbekanneten $a_0, ..., a_n$. Deren Matrixform ist

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ & & & \vdots & \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix}$$

Ein Vorschlag: Wähle $n+1$ verschiedenen Punkten $x_0, ..., x_n$ auf $[a, b]$ und suche ein Polynom P, $Grad(P) \leq n$, s.d. $P(x_i) = f(x_i)$.
Hausaufgabe 3 Blatt 9: Solches Polynom existiert und ist eindeutig. Tatsächlich, die Bediengung $P(x_i) = f(x_i)$ ist das folgende System von Gleichungen

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n &= f(x_0) \\ & \vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n &= f(x_n) \end{cases}$$

Das System besteht aus $n+1$ linearen Gleichungen mit $n+1$ Unbekanneten $a_0, ..., a_n$. Deren Matrixform ist

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ & & & \vdots & \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f(x_0) \\ \vdots \\ f(x_n) \end{pmatrix}$$

Ein Vorschlag: Wähle $n+1$ verschiedenen Punkten $x_0, ..., x_n$ auf $[a, b]$ und suche ein Polynom P, $Grad(P) \leq n$, s.d. $P(x_i) = f(x_i)$.
Hausaufgabe 3 Blatt 9: Solches Polynom existiert und ist eindeutig. Tatsächlich, die Bediengung $P(x_i) = f(x_i)$ ist das folgende System von Gleichungen

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n &= f(x_0) \\ & \vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n &= f(x_n) \end{cases}$$

Das System besteht aus $n+1$ linearen Gleichungen mit $n+1$ Unbekanneten $a_0, ..., a_n$. Deren Matrixform ist

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ & & & \vdots & \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f(x_0) \\ \vdots \\ f(x_n) \end{pmatrix}.$$

Da die Koeffizientenmatrix nichtausgeartet ist

Ein Vorschlag: Wähle $n+1$ verschiedenen Punkten $x_0, ..., x_n$ auf $[a, b]$ und suche ein Polynom P, $Grad(P) \leq n$, s.d. $P(x_i) = f(x_i)$.
Hausaufgabe 3 Blatt 9: Solches Polynom existiert und ist eindeutig. Tatsächlich, die Bediengung $P(x_i) = f(x_i)$ ist das folgende System von Gleichungen

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n &= f(x_0) \\ & \vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n &= f(x_n) \end{cases}$$

Das System besteht aus $n+1$ linearen Gleichungen mit $n+1$ Unbekanneten $a_0, ..., a_n$. Deren Matrixform ist

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ & & & \vdots & \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f(x_0) \\ \vdots \\ f(x_n) \end{pmatrix}.$$

Da die Koeffizientenmatrix nichtausgeartet ist(Hausaufgabe 3a Blatt 9),

Ein Vorschlag: Wähle $n+1$ verschiedenen Punkten $x_0, ..., x_n$ auf $[a, b]$ und suche ein Polynom P, $Grad(P) \leq n$, s.d. $P(x_i) = f(x_i)$.
Hausaufgabe 3 Blatt 9: Solches Polynom existiert und ist eindeutig. Tatsächlich, die Bediengung $P(x_i) = f(x_i)$ ist das folgende System von Gleichungen

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n &= f(x_0) \\ & \vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n &= f(x_n) \end{cases}$$

Das System besteht aus $n+1$ linearen Gleichungen mit $n+1$ Unbekanneten $a_0, ..., a_n$. Deren Matrixform ist

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ & & & \vdots & \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f(x_0) \\ \vdots \\ f(x_n) \end{pmatrix}.$$

Da die Koeffizientenmatrix nichtausgeartet ist(Hausaufgabe 3a Blatt 9), existiert die Lösung

Ein Vorschlag: Wähle $n+1$ verschiedenen Punkten $x_0, ..., x_n$ auf $[a, b]$ und suche ein Polynom P, $Grad(P) \leq n$, s.d. $P(x_i) = f(x_i)$.
Hausaufgabe 3 Blatt 9: Solches Polynom existiert und ist eindeutig. Tatsächlich, die Bediengung $P(x_i) = f(x_i)$ ist das folgende System von Gleichungen

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n &= f(x_0) \\ & \vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n &= f(x_n) \end{cases}$$

Das System besteht aus $n+1$ linearen Gleichungen mit $n+1$ Unbekanneten $a_0, ..., a_n$. Deren Matrixform ist

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ & & & \vdots & \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f(x_0) \\ \vdots \\ f(x_n) \end{pmatrix}.$$

Da die Koeffizientenmatrix nichtausgeartet ist(Hausaufgabe 3a Blatt 9), existiert die Lösung und ist eindeutig.

Ein Vorschlag: Wähle $n+1$ verschiedenen Punkten $x_0, ..., x_n$ auf $[a, b]$ und suche ein Polynom P, $Grad(P) \leq n$, s.d. $P(x_i) = f(x_i)$.
Hausaufgabe 3 Blatt 9: Solches Polynom existiert und ist eindeutig. Tatsächlich, die Bediengung $P(x_i) = f(x_i)$ ist das folgende System von Gleichungen

$$\begin{cases} a_0 + a_1 x_0 + a_2 x_0^2 + \ldots + a_n x_0^n &= f(x_0) \\ & \vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_n x_n^n &= f(x_n) \end{cases}$$

Das System besteht aus $n+1$ linearen Gleichungen mit $n+1$ Unbekanneten $a_0, ..., a_n$. Deren Matrixform ist

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ & & & \vdots & \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f(x_0) \\ \vdots \\ f(x_n) \end{pmatrix}.$$

Da die Koeffizientenmatrix nichtausgeartet ist(Hausaufgabe 3a Blatt 9), existiert die Lösung und ist eindeutig.

Interpolations-Polynom von Lagrange (1736-1813)

Interpolations-Polynom von Lagrange (1736-1813)

$P_i :=$

Interpolations-Polynom von Lagrange (1736-1813)

$$P_i := \frac{\prod_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)} =$$

Interpolations-Polynom von Lagrange (1736-1813)

$$P_i := \frac{\prod_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)} = \frac{(x - x_0)(x - x_1)...(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)(x_i - x_1)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}$$

Interpolations-Polynom von Lagrange (1736-1813)

$$P_i := \frac{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)} = \frac{(x - x_0)(x - x_1)...(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)(x_i - x_1)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}$$

$\in \mathbb{R}[x]$

Interpolations-Polynom von Lagrange (1736-1813)

$$P_i := \frac{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)} = \frac{(x - x_0)(x - x_1)...(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)(x_i - x_1)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}$$

$\in \mathbb{R}[x]$ (des Grades n).

Interpolations-Polynom von Lagrange (1736-1813)

$$P_i := \frac{\prod_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)} = \frac{(x - x_0)(x - x_1)...(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)(x_i - x_1)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}$$

$\in \mathbb{R}[x]$ (des Grades n). Dessen Eigenschaft: $P(x_i) = 1$

Interpolations-Polynom von Lagrange (1736-1813)

$$P_i := \frac{\prod_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)} = \frac{(x - x_0)(x - x_1)...(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)(x_i - x_1)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}$$

$\in \mathbb{R}[x]$ (des Grades n). Dessen Eigenschaft: $P(x_i) = 1$ (weil oben und unten gleiche Ausdrücke stehen)

Interpolations-Polynom von Lagrange (1736-1813)

$$P_i := \frac{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)} = \frac{(x - x_0)(x - x_1)...(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)(x_i - x_1)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}$$

$\in \mathbb{R}[x]$ (des Grades n). Dessen Eigenschaft: $P(x_i) = 1$ (weil oben und unten gleiche Ausdrücke stehen)
$P(x_k) = 0$ für $k \neq i$ (weil oben ein Faktor $= 0$ ist).

Interpolations-Polynom von Lagrange (1736-1813)

$$P_i := \frac{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)} = \frac{(x - x_0)(x - x_1)...(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)(x_i - x_1)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}$$

$\in \mathbb{R}[x]$ (des Grades n). Dessen Eigenschaft: $P(x_i) = 1$ (weil oben und unten gleiche Ausdrücke stehen)
$P(x_k) = 0$ für $k \neq i$ (weil oben ein Faktor $= 0$ ist).
Dann ist $P := \sum_i f(x_i) P_i$ das gesuchte Polynom:

Interpolations-Polynom von Lagrange (1736-1813)

$$P_i := \frac{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)} = \frac{(x - x_0)(x - x_1)...(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)(x_i - x_1)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}$$

$\in \mathbb{R}[x]$ (des Grades n). Dessen Eigenschaft: $P(x_i) = 1$ (weil oben und unten gleiche Ausdrücke stehen)
$P(x_k) = 0$ für $k \neq i$ (weil oben ein Faktor $= 0$ ist).
Dann ist $P := \sum_i f(x_i)P_i$ das gesuchte Polynom:
$P(x_0) = \sum_i \ f(x_0)P_i(x_0) = f(x_0) + 0 + 0 + ... = f(x_0)$,

Interpolations-Polynom von Lagrange (1736-1813)

$$P_i := \frac{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)} = \frac{(x - x_0)(x - x_1)...(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)(x_i - x_1)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}$$

$\in \mathbb{R}[x]$ (des Grades n). Dessen Eigenschaft: $P(x_i) = 1$ (weil oben und unten gleiche Ausdrücke stehen)
$P(x_k) = 0$ für $k \neq i$ (weil oben ein Faktor $= 0$ ist).
Dann ist $P := \sum_i f(x_i)P_i$ das gesuchte Polynom:
$P(x_0) = \sum_i \ f(x_0)P_i(x_0) = f(x_0) + 0 + 0 + ... = f(x_0)$,
$P(x_1) = \sum_i \ f(x_1)P_i(x_1) = 0 + f(x_1) + 0 + 0 + ... = f(x_1)$,

Interpolations-Polynom von Lagrange (1736-1813)

$$P_i := \frac{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)} = \frac{(x - x_0)(x - x_1)...(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)(x_i - x_1)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}$$

$\in \mathbb{R}[x]$ (des Grades n). Dessen Eigenschaft: $P(x_i) = 1$ (weil oben und unten gleiche Ausdrücke stehen)
$P(x_k) = 0$ für $k \neq i$ (weil oben ein Faktor $= 0$ ist).
Dann ist $P := \sum_i f(x_i)P_i$ das gesuchte Polynom:
$P(x_0) = \sum_i \ f(x_0)P_i(x_0) = f(x_0) + 0 + 0 + ... = f(x_0)$,
$P(x_1) = \sum_i \ f(x_1)P_i(x_1) = 0 + f(x_1) + 0 + 0 + ... = f(x_1)$,
$P(x_j) = \sum_i \ f(x_j)P_i(x_j) =$

Interpolations-Polynom von Lagrange (1736-1813)

$$P_i := \frac{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)} = \frac{(x - x_0)(x - x_1)...(x - x_{i-1})(x - x_{i+1})...(x - x_n)}{(x_i - x_0)(x_i - x_1)...(x_i - x_{i-1})(x_i - x_{i+1})...(x_i - x_n)}$$

$\in \mathbb{R}[x]$ (des Grades n). Dessen Eigenschaft: $P(x_i) = 1$ (weil oben und unten gleiche Ausdrücke stehen)
$P(x_k) = 0$ für $k \neq i$ (weil oben ein Faktor $= 0$ ist).
Dann ist $P := \sum_i f(x_i)P_i$ das gesuchte Polynom:
$P(x_0) = \sum_i \ f(x_0)P_i(x_0) = f(x_0) + 0 + 0 + ... = f(x_0)$,
$P(x_1) = \sum_i \ f(x_1)P_i(x_1) = 0 + f(x_1) + 0 + 0 + ... = f(x_1)$,
$P(x_j) = \sum_i \ f(x_j)P_i(x_j) = \underbrace{0 + ... + 0}$

Interpolations-Polynom von Lagrange (1736-1813)

$$P_i := \frac{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)} = \frac{(x-x_0)(x-x_1)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)(x_i-x_1)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)}$$

$\in \mathbb{R}[x]$ (des Grades n). Dessen Eigenschaft: $P(x_i) = 1$ (weil oben und unten gleiche Ausdrücke stehen)
$P(x_k) = 0$ für $k \neq i$ (weil oben ein Faktor $= 0$ ist).
Dann ist $P := \sum_i f(x_i)P_i$ das gesuchte Polynom:
$P(x_0) = \sum_i \ f(x_0)P_i(x_0) = f(x_0) + 0 + 0 + ... = f(x_0)$,
$P(x_1) = \sum_i \ f(x_1)P_i(x_1) = 0 + f(x_1) + 0 + 0 + ... = f(x_1)$,
$P(x_j) = \sum_i \ f(x_j)P_i(x_j) = \underbrace{0 + ... + 0}_{j\text{Stuck}} +$

Interpolations-Polynom von Lagrange (1736-1813)

$$P_i := \frac{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)} = \frac{(x-x_0)(x-x_1)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)(x_i-x_1)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)}$$

$\in \mathbb{R}[x]$ (des Grades n). Dessen Eigenschaft: $P(x_i) = 1$ (weil oben und unten gleiche Ausdrücke stehen)
$P(x_k) = 0$ für $k \neq i$ (weil oben ein Faktor $= 0$ ist).
Dann ist $P := \sum_i f(x_i)P_i$ das gesuchte Polynom:
$P(x_0) = \sum_i \ f(x_0)P_i(x_0) = f(x_0) + 0 + 0 + ... = f(x_0)$,
$P(x_1) = \sum_i \ f(x_1)P_i(x_1) = 0 + f(x_1) + 0 + 0 + ... = f(x_1)$,
$P(x_j) = \sum_i \ f(x_j)P_i(x_j) = \underbrace{0 + ... + 0}_{j\text{Stuck}} + \underbrace{f(x_j)}_{j-\text{te Stelle}}$

Interpolations-Polynom von Lagrange (1736-1813)

$$P_i := \frac{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)} = \frac{(x-x_0)(x-x_1)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)(x_i-x_1)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)}$$

$\in \mathbb{R}[x]$ (des Grades n). Dessen Eigenschaft: $P(x_i) = 1$ (weil oben und unten gleiche Ausdrücke stehen)
$P(x_k) = 0$ für $k \neq i$ (weil oben ein Faktor $= 0$ ist).
Dann ist $P := \sum_i f(x_i)P_i$ das gesuchte Polynom:
$P(x_0) = \sum_i \ f(x_0)P_i(x_0) = f(x_0) + 0 + 0 + ... = f(x_0)$,
$P(x_1) = \sum_i \ f(x_1)P_i(x_1) = 0 + f(x_1) + 0 + 0 + ... = f(x_1)$,
$P(x_j) = \sum_i \ f(x_j)P_i(x_j) = \underbrace{0 + ... + 0}_{j\text{Stuck}} + \underbrace{f(x_j)}_{j-\text{te Stelle}} + 0 + ... + 0 =$

Interpolations-Polynom von Lagrange (1736-1813)

$$P_i := \frac{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)} = \frac{(x-x_0)(x-x_1)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)(x_i-x_1)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)}$$

$\in \mathbb{R}[x]$ (des Grades n). Dessen Eigenschaft: $P(x_i) = 1$ (weil oben und unten gleiche Ausdrücke stehen)
$P(x_k) = 0$ für $k \neq i$ (weil oben ein Faktor $= 0$ ist).
Dann ist $P := \sum_i f(x_i)P_i$ das gesuchte Polynom:
$P(x_0) = \sum_i \ f(x_0)P_i(x_0) = f(x_0) + 0 + 0 + ... = f(x_0)$,
$P(x_1) = \sum_i \ f(x_1)P_i(x_1) = 0 + f(x_1) + 0 + 0 + ... = f(x_1)$,
$P(x_j) = \sum_i \ f(x_j)P_i(x_j) = \underbrace{0 + ... + 0}_{j\text{Stuck}} + \underbrace{f(x_j)}_{j-\text{te Stelle}} + 0 + ... + 0 = f(x_j)$.

Interpolations-Polynom von Lagrange (1736-1813)

$$P_i := \frac{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)} = \frac{(x-x_0)(x-x_1)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)(x_i-x_1)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)}$$

$\in \mathbb{R}[x]$ (des Grades n). Dessen Eigenschaft: $P(x_i) = 1$ (weil oben und unten gleiche Ausdrücke stehen)
$P(x_k) = 0$ für $k \neq i$ (weil oben ein Faktor $= 0$ ist).
Dann ist $P := \sum_i f(x_i)P_i$ das gesuchte Polynom:
$P(x_0) = \sum_i \ f(x_0)P_i(x_0) = f(x_0) + 0 + 0 + ... = f(x_0)$,
$P(x_1) = \sum_i \ f(x_1)P_i(x_1) = 0 + f(x_1) + 0 + 0 + ... = f(x_1)$,
$P(x_j) = \sum_i \ f(x_j)P_i(x_j) = \underbrace{0 + ... + 0}_{j\text{Stuck}} + \underbrace{f(x_j)}_{j-\text{te Stelle}} + 0 + ... + 0 = f(x_j)$.

Interpolations-Polynom von Lagrange (1736-1813)

$$P_i := \frac{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)} = \frac{(x-x_0)(x-x_1)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)(x_i-x_1)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)}$$

$\in \mathbb{R}[x]$ (des Grades n). Dessen Eigenschaft: $P(x_i) = 1$ (weil oben und unten gleiche Ausdrücke stehen)
$P(x_k) = 0$ für $k \neq i$ (weil oben ein Faktor $= 0$ ist).
Dann ist $P := \sum_i f(x_i)P_i$ das gesuchte Polynom:
$P(x_0) = \sum_i \ f(x_0)P_i(x_0) = f(x_0) + 0 + 0 + ... = f(x_0)$,
$P(x_1) = \sum_i \ f(x_1)P_i(x_1) = 0 + f(x_1) + 0 + 0 + ... = f(x_1)$,
$P(x_j) = \sum_i \ f(x_j)P_i(x_j) = \underbrace{0 + ... + 0}_{j\text{Stuck}} + \underbrace{f(x_j)}_{j-\text{te Stelle}} + 0 + ... + 0 = f(x_j)$.

Bemerkung *Lagrange-Polynom*

Interpolations-Polynom von Lagrange (1736-1813)

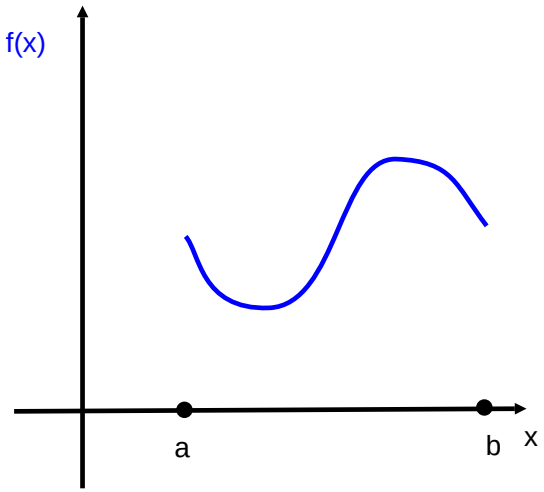
$$P_i := \frac{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod\limits_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)} = \frac{(x-x_0)(x-x_1)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)(x_i-x_1)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)}$$

$\in \mathbb{R}[x]$ (des Grades n). Dessen Eigenschaft: $P(x_i) = 1$ (weil oben und unten gleiche Ausdrücke stehen)
$P(x_k) = 0$ für $k \neq i$ (weil oben ein Faktor $= 0$ ist).
Dann ist $P := \sum_i f(x_i)P_i$ das gesuchte Polynom:
$P(x_0) = \sum_i \ f(x_0)P_i(x_0) = f(x_0) + 0 + 0 + ... = f(x_0)$,
$P(x_1) = \sum_i \ f(x_1)P_i(x_1) = 0 + f(x_1) + 0 + 0 + ... = f(x_1)$,
$P(x_j) = \sum_i \ f(x_j)P_i(x_j) = \underbrace{0 + ... + 0}_{j\text{Stuck}} + \underbrace{f(x_j)}_{j-\text{te Stelle}} + 0 + ... + 0 = f(x_j)$.

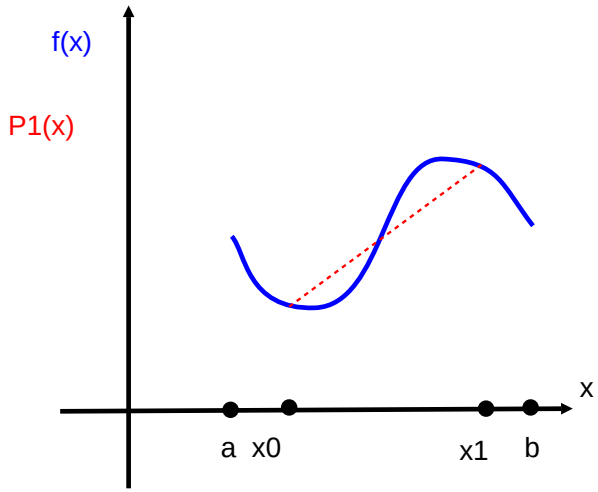
Bemerkung *Lagrange-Polynom kann man über einem beliebigen Körper definieren*

Bsp:

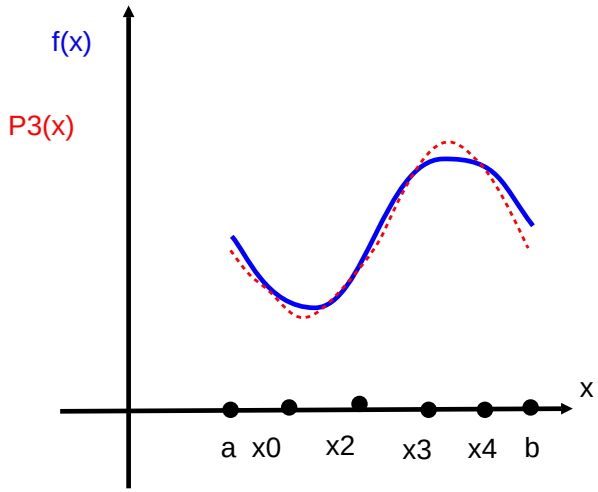
Bsp: approximieren mit Polynomen des Grades 1,3



Bsp: approximieren mit Polynomen des Grades 1,3



Bsp: approximieren mit Polynomen des Grades 1,3



Wie gut approximiert das Lagrange-Polynom?

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < \ldots < x_n \in [a,b]$

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < \ldots < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$.

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < \ldots < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < \ldots < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.
Frage

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < \ldots < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.
Frage Wie klein ist $P(y) - f(y)$?

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < \ldots < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.
Frage Wie klein ist $P(y) - f(y)$? Wie soll man die Punkte $x_0 < \ldots < x_n \in [a,b]$ auswählen,

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < \ldots < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.
Frage Wie klein ist $P(y) - f(y)$? Wie soll man die Punkte $x_0 < \ldots < x_n \in [a,b]$ auswählen, um $P(y) - f(y)$ möglich klein zu machen?

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < ... < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.
Frage Wie klein ist $P(y) - f(y)$? Wie soll man die Punkte $x_0 < ... < x_n \in [a,b]$ auswählen, um $P(y) - f(y)$ möglich klein zu machen?

Sei $y \neq x_0, ..., x_n$.

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < ... < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.
Frage Wie klein ist $P(y) - f(y)$? Wie soll man die Punkte $x_0 < ... < x_n \in [a,b]$ auswählen, um $P(y) - f(y)$ möglich klein zu machen?

Sei $y \neq x_0, ..., x_n$. Definiere

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < ... < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.

Frage Wie klein ist $P(y) - f(y)$? Wie soll man die Punkte $x_0 < ... < x_n \in [a,b]$ auswählen, um $P(y) - f(y)$ möglich klein zu machen?

Sei $y \neq x_0, ..., x_n$. Definiere $\omega(x) := \prod_{j=0}^{n} (x - x_j)$

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < ... < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.
Frage Wie klein ist $P(y) - f(y)$? Wie soll man die Punkte $x_0 < ... < x_n \in [a,b]$ auswählen, um $P(y) - f(y)$ möglich klein zu machen?
Sei $y \neq x_0, ..., x_n$. Definiere $\omega(x) := \prod_{j=0}^{n} (x - x_j)$ und

$\phi(x) := f(x) - P(x) - K\omega(x)$,

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < ... < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.
Frage Wie klein ist $P(y) - f(y)$? Wie soll man die Punkte $x_0 < ... < x_n \in [a,b]$ auswählen, um $P(y) - f(y)$ möglich klein zu machen?
Sei $y \neq x_0, ..., x_n$. Definiere $\omega(x) := \prod_{j=0}^{n} (x - x_j)$ und $\phi(x) := f(x) - P(x) - K\omega(x)$, wobei die Konstante K so gewählt ist,

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < ... < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.
Frage Wie klein ist $P(y) - f(y)$? Wie soll man die Punkte $x_0 < ... < x_n \in [a,b]$ auswählen, um $P(y) - f(y)$ möglich klein zu machen?
Sei $y \neq x_0, ..., x_n$. Definiere $\omega(x) := \prod_{j=0}^{n} (x - x_j)$ und $\phi(x) := f(x) - P(x) - K\omega(x)$, wobei die Konstante K so gewählt ist, dass $\phi(y) = 0$.

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < ... < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.
Frage Wie klein ist $P(y) - f(y)$? Wie soll man die Punkte $x_0 < ... < x_n \in [a,b]$ auswählen, um $P(y) - f(y)$ möglich klein zu machen?
Sei $y \neq x_0, ..., x_n$. Definiere $\omega(x) := \prod_{j=0}^{n} (x - x_j)$ und $\phi(x) := f(x) - P(x) - K\omega(x)$, wobei die Konstante K so gewählt ist, dass $\phi(y) = 0$. (Möglich, weil $\omega(x) \neq 0$ für $x \neq x_0, ..., x_n$ nach Lemma 6.

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < ... < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.
Frage Wie klein ist $P(y) - f(y)$? Wie soll man die Punkte $x_0 < ... < x_n \in [a,b]$ auswählen, um $P(y) - f(y)$ möglich klein zu machen?
Sei $y \neq x_0, ..., x_n$. Definiere $\omega(x) := \prod_{j=0}^{n} (x - x_j)$ und $\phi(x) := f(x) - P(x) - K\omega(x)$, wobei die Konstante K so gewählt ist, dass $\phi(y) = 0$. (Möglich, weil $\omega(x) \neq 0$ für $x \neq x_0, ..., x_n$ nach Lemma 6. Also, $K = \frac{f(y)-p(y)}{\omega(y)}$.)

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < ... < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.

Frage Wie klein ist $P(y) - f(y)$? Wie soll man die Punkte $x_0 < ... < x_n \in [a,b]$ auswählen, um $P(y) - f(y)$ möglich klein zu machen?

Sei $y \neq x_0, ..., x_n$. Definiere $\omega(x) := \prod_{j=0}^{n} (x - x_j)$ und $\phi(x) := f(x) - P(x) - K\omega(x)$, wobei die Konstante K so gewählt ist, dass $\phi(y) = 0$. (Möglich, weil $\omega(x) \neq 0$ für $x \neq x_0, ..., x_n$ nach Lemma 6. Also, $K = \frac{f(y)-p(y)}{\omega(y)}$.)

Nach Konstruktion ist

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < ... < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.

Frage Wie klein ist $P(y) - f(y)$? Wie soll man die Punkte $x_0 < ... < x_n \in [a,b]$ auswählen, um $P(y) - f(y)$ möglich klein zu machen?

Sei $y \neq x_0, ..., x_n$. Definiere $\omega(x) := \prod_{j=0}^{n} (x - x_j)$ und $\phi(x) := f(x) - P(x) - K\omega(x)$, wobei die Konstante K so gewählt ist, dass $\phi(y) = 0$. (Möglich, weil $\omega(x) \neq 0$ für $x \neq x_0, ..., x_n$ nach Lemma 6. Also, $K = \frac{f(y)-p(y)}{\omega(y)}$.)

Nach Konstruktion ist $\phi(x_i) = \underbrace{f(x_i) - P(x_i)}$

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < ... < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.

Frage Wie klein ist $P(y) - f(y)$? Wie soll man die Punkte $x_0 < ... < x_n \in [a,b]$ auswählen, um $P(y) - f(y)$ möglich klein zu machen?

Sei $y \neq x_0, ..., x_n$. Definiere $\omega(x) := \prod_{j=0}^{n} (x - x_j)$ und $\phi(x) := f(x) - P(x) - K\omega(x)$, wobei die Konstante K so gewählt ist, dass $\phi(y) = 0$. (Möglich, weil $\omega(x) \neq 0$ für $x \neq x_0, ..., x_n$ nach Lemma 6. Also, $K = \frac{f(y)-p(y)}{\omega(y)}$.)

Nach Konstruktion ist $\phi(x_i) = \underbrace{f(x_i) - P(x_i)}_{0} - \underbrace{K\omega(x_i)}$

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < ... < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.

Frage Wie klein ist $P(y) - f(y)$? Wie soll man die Punkte $x_0 < ... < x_n \in [a,b]$ auswählen, um $P(y) - f(y)$ möglich klein zu machen?

Sei $y \neq x_0, ..., x_n$. Definiere $\omega(x) := \prod_{j=0}^{n} (x - x_j)$ und $\phi(x) := f(x) - P(x) - K\omega(x)$, wobei die Konstante K so gewählt ist, dass $\phi(y) = 0$. (Möglich, weil $\omega(x) \neq 0$ für $x \neq x_0, ..., x_n$ nach Lemma 6. Also, $K = \frac{f(y)-p(y)}{\omega(y)}$.)

Nach Konstruktion ist $\phi(x_i) = \underbrace{f(x_i) - P(x_i)}_{0} - \underbrace{K\omega(x_i)}_{0}$

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < ... < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.

Frage Wie klein ist $P(y) - f(y)$? Wie soll man die Punkte $x_0 < ... < x_n \in [a,b]$ auswählen, um $P(y) - f(y)$ möglich klein zu machen?

Sei $y \neq x_0, ..., x_n$. Definiere $\omega(x) := \prod_{j=0}^{n} (x - x_j)$ und $\phi(x) := f(x) - P(x) - K\omega(x)$, wobei die Konstante K so gewählt ist, dass $\phi(y) = 0$. (Möglich, weil $\omega(x) \neq 0$ für $x \neq x_0, ..., x_n$ nach Lemma 6. Also, $K = \frac{f(y)-p(y)}{\omega(y)}$.)

Nach Konstruktion ist $\phi(x_i) = \underbrace{f(x_i) - P(x_i)}_{0} - \underbrace{K\omega(x_i)}_{0} = 0$

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < ... < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.

Frage Wie klein ist $P(y) - f(y)$? Wie soll man die Punkte $x_0 < ... < x_n \in [a,b]$ auswählen, um $P(y) - f(y)$ möglich klein zu machen?

Sei $y \neq x_0, ..., x_n$. Definiere $\omega(x) := \prod_{j=0}^{n} (x - x_j)$ und $\phi(x) := f(x) - P(x) - K\omega(x)$, wobei die Konstante K so gewählt ist, dass $\phi(y) = 0$. (Möglich, weil $\omega(x) \neq 0$ für $x \neq x_0, ..., x_n$ nach Lemma 6. Also, $K = \frac{f(y)-p(y)}{\omega(y)}$.)

Nach Konstruktion ist $\phi(x_i) = \underbrace{f(x_i) - P(x_i)}_{0} - \underbrace{K\omega(x_i)}_{0} = 0$ und

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < ... < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.

Frage Wie klein ist $P(y) - f(y)$? Wie soll man die Punkte $x_0 < ... < x_n \in [a,b]$ auswählen, um $P(y) - f(y)$ möglich klein zu machen?

Sei $y \neq x_0, ..., x_n$. Definiere $\omega(x) := \prod_{j=0}^{n} (x - x_j)$ und

$\phi(x) := f(x) - P(x) - K\omega(x)$, wobei die Konstante K so gewählt ist, dass $\phi(y) = 0$. (Möglich, weil $\omega(x) \neq 0$ für $x \neq x_0, ..., x_n$ nach Lemma 6. Also, $K = \frac{f(y)-p(y)}{\omega(y)}$.)

Nach Konstruktion ist $\phi(x_i) = \underbrace{f(x_i) - P(x_i)}_{0} - \underbrace{K\omega(x_i)}_{0} = 0$ und

$\phi(y) = 0$.

Da $f \in C^{n+1}([a,b])$ ist,

Wie gut approximiert das Lagrange-Polynom?

Betrachte $f \in C^{n+1}([a,b])$, $x_0 < ... < x_n \in [a,b]$ und das zugehörige Lagrange-Polynom P, $P(x_i) = f(x_i)$. Nehme ein $y \in [a,b]$.

Frage Wie klein ist $P(y) - f(y)$? Wie soll man die Punkte $x_0 < ... < x_n \in [a,b]$ auswählen, um $P(y) - f(y)$ möglich klein zu machen?

Sei $y \neq x_0, ..., x_n$. Definiere $\omega(x) := \prod_{j=0}^{n} (x - x_j)$ und $\phi(x) := f(x) - P(x) - K\omega(x)$, wobei die Konstante K so gewählt ist, dass $\phi(y) = 0$. (Möglich, weil $\omega(x) \neq 0$ für $x \neq x_0, ..., x_n$ nach Lemma 6. Also, $K = \frac{f(y)-p(y)}{\omega(y)}$.)

Nach Konstruktion ist $\phi(x_i) = \underbrace{f(x_i) - P(x_i)}_{0} - \underbrace{K\omega(x_i)}_{0} = 0$ und $\phi(y) = 0$.

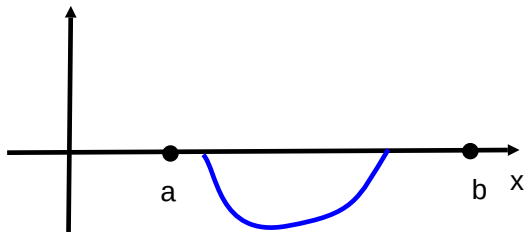
Da $f \in C^{n+1}([a,b])$ ist, ist $\phi \in C^{n+1}([a,b])$.

Weil ϕ mindestens $n+2$ Nulstellen hat,

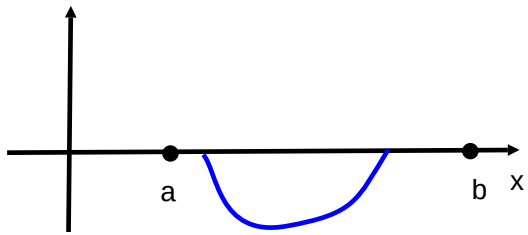
Weil ϕ mindestens $n+2$ Nulstellen hat, hat deren Ableitung ϕ' hat mindestens $n+1$ Nullstellen.

Weil ϕ mindestens $n+2$ Nulstellen hat, hat deren Ableitung ϕ' hat mindestens $n+1$ Nullstellen. (Auf jedem Intervall zwischen zwei Nullstellen gibt es eine Nullstelle von ϕ'.)

Weil ϕ mindestens $n+2$ Nulstellen hat, hat deren Ableitung ϕ' hat mindestens $n+1$ Nullstellen. (Auf jedem Intervall zwischen zwei Nullstellen gibt es eine Nullstelle von ϕ'.)

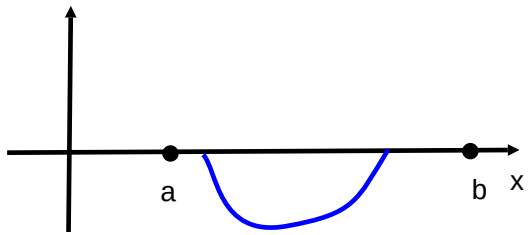


Weil ϕ mindestens $n+2$ Nulstellen hat, hat deren Ableitung ϕ' hat mindestens $n+1$ Nullstellen. (Auf jedem Intervall zwischen zwei Nullstellen gibt es eine Nullstelle von ϕ'.)



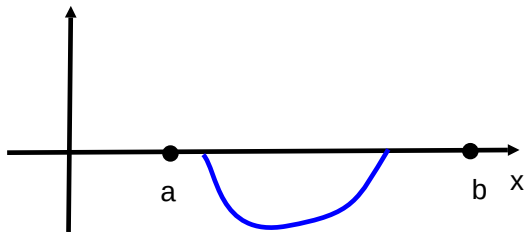
Ähnlich, die zweite Ableitung ϕ'' hat mindestens n Nullstellen

Weil ϕ mindestens $n+2$ Nulstellen hat, hat deren Ableitung ϕ' hat mindestens $n+1$ Nullstellen. (Auf jedem Intervall zwischen zwei Nullstellen gibt es eine Nullstelle von ϕ'.)



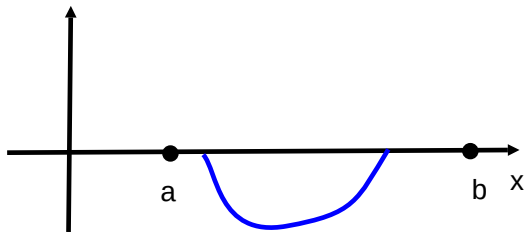
Ähnlich, die zweite Ableitung ϕ'' hat mindestens n Nullstellen U.S.W.

Weil ϕ mindestens $n+2$ Nulstellen hat, hat deren Ableitung ϕ' hat mindestens $n+1$ Nullstellen. (Auf jedem Intervall zwischen zwei Nullstellen gibt es eine Nullstelle von ϕ'.)



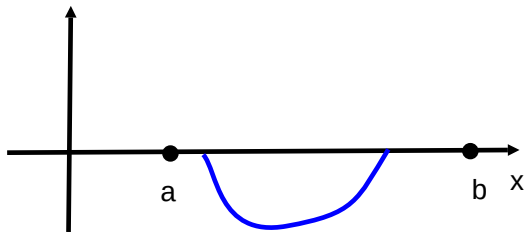
Ähnlich, die zweite Ableitung ϕ'' hat mindestens n Nullstellen U.S.W. Also, die $(n+1)$te Ableitung $\phi^{(n+1)}$ hat mindestens eine Nulstelle.

Weil ϕ mindestens $n+2$ Nulstellen hat, hat deren Ableitung ϕ' hat mindestens $n+1$ Nullstellen. (Auf jedem Intervall zwischen zwei Nullstellen gibt es eine Nullstelle von ϕ'.)



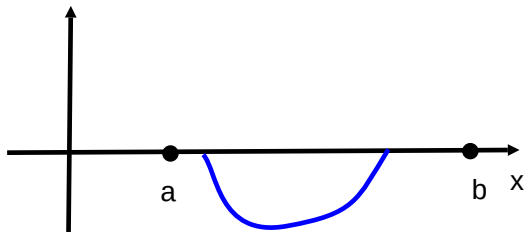
Ähnlich, die zweite Ableitung ϕ'' hat mindestens n Nullstellen U.S.W. Also, die $(n+1)$te Ableitung $\phi^{(n+1)}$ hat mindestens eine Nulstelle. Sei y_0 diese Nulstelle.

Weil ϕ mindestens $n+2$ Nulstellen hat, hat deren Ableitung ϕ' hat mindestens $n+1$ Nullstellen. (Auf jedem Intervall zwischen zwei Nullstellen gibt es eine Nullstelle von ϕ'.)



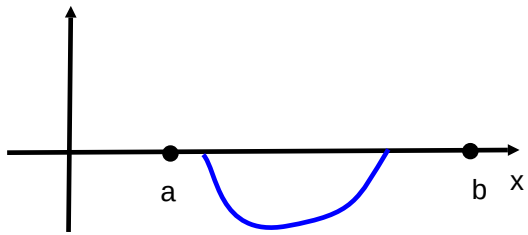
Ähnlich, die zweite Ableitung ϕ'' hat mindestens n Nullstellen U.S.W. Also, die $(n+1)$te Ableitung $\phi^{(n+1)}$ hat mindestens eine Nulstelle. Sei y_0 diese Nulstelle.
Da $\phi^{(n+1)} = f^{(n+1)} -$

Weil ϕ mindestens $n+2$ Nulstellen hat, hat deren Ableitung ϕ' hat mindestens $n+1$ Nullstellen. (Auf jedem Intervall zwischen zwei Nullstellen gibt es eine Nullstelle von ϕ'.)



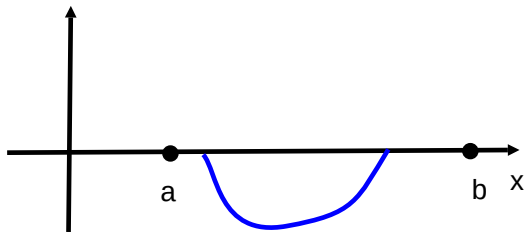
Ähnlich, die zweite Ableitung ϕ'' hat mindestens n Nullstellen U.S.W. Also, die $(n+1)$te Ableitung $\phi^{(n+1)}$ hat mindestens eine Nulstelle. Sei y_0 diese Nulstelle.
Da $\phi^{(n+1)} = f^{(n+1)} - \underbrace{P^{(n+1)}}$

Weil ϕ mindestens $n+2$ Nulstellen hat, hat deren Ableitung ϕ' hat mindestens $n+1$ Nullstellen. (Auf jedem Intervall zwischen zwei Nullstellen gibt es eine Nullstelle von ϕ'.)



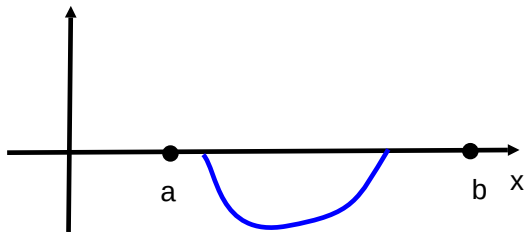
Ähnlich, die zweite Ableitung ϕ'' hat mindestens n Nullstellen U.S.W. Also, die $(n+1)$te Ableitung $\phi^{(n+1)}$ hat mindestens eine Nulstelle. Sei y_0 diese Nulstelle.
Da $\phi^{(n+1)} = f^{(n+1)} - \underbrace{P^{(n+1)}}_{0, \text{weil } P \text{ Grad n hat}}$

Weil ϕ mindestens $n+2$ Nulstellen hat, hat deren Ableitung ϕ' hat mindestens $n+1$ Nullstellen. (Auf jedem Intervall zwischen zwei Nullstellen gibt es eine Nullstelle von ϕ'.)



Ähnlich, die zweite Ableitung ϕ'' hat mindestens n Nullstellen U.S.W. Also, die $(n+1)$te Ableitung $\phi^{(n+1)}$ hat mindestens eine Nulstelle. Sei y_0 diese Nulstelle.
Da $\phi^{(n+1)} = f^{(n+1)} - \underbrace{P^{(n+1)}}_{0, \text{weil } P \text{ Grad n hat}} - K \underbrace{\omega^{(n+1)}}$

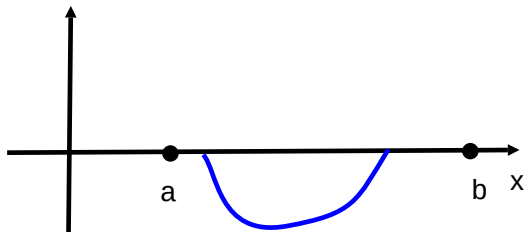
Weil ϕ mindestens $n+2$ Nulstellen hat, hat deren Ableitung ϕ' hat mindestens $n+1$ Nullstellen. (Auf jedem Intervall zwischen zwei Nullstellen gibt es eine Nullstelle von ϕ'.)



Ähnlich, die zweite Ableitung ϕ'' hat mindestens n Nullstellen U.S.W. Also, die $(n+1)$te Ableitung $\phi^{(n+1)}$ hat mindestens eine Nulstelle. Sei y_0 diese Nulstelle.
Da $\phi^{(n+1)} = f^{(n+1)} - \underbrace{P^{(n+1)}}_{0, \text{weil } P \text{ Grad n hat}} - K \underbrace{\omega^{(n+1)}}_{(n+1)! \text{weil } P \text{ Grad n+1 hat}}$.

Also,
$\phi^{(n+1)} = f^{(n+1)} - K(n+1)!$,

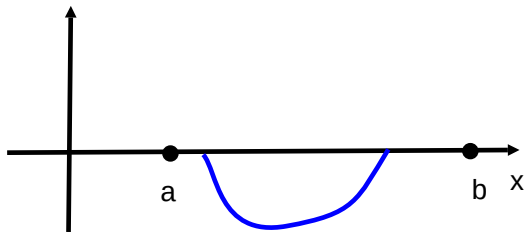
Weil ϕ mindestens $n+2$ Nulstellen hat, hat deren Ableitung ϕ' hat mindestens $n+1$ Nullstellen. (Auf jedem Intervall zwischen zwei Nullstellen gibt es eine Nullstelle von ϕ'.)



Ähnlich, die zweite Ableitung ϕ'' hat mindestens n Nullstellen U.S.W. Also, die $(n+1)$te Ableitung $\phi^{(n+1)}$ hat mindestens eine Nulstelle. Sei y_0 diese Nulstelle.
Da $\phi^{(n+1)} = f^{(n+1)} - \underbrace{P^{(n+1)}}_{0,\text{weil } P \text{ Grad n hat}} - K \underbrace{\omega^{(n+1)}}_{(n+1)!\text{weil } P \text{ Grad n+1 hat}}$.

Also,
$\phi^{(n+1)} = f^{(n+1)} - K(n+1)!$,und deswegen
$\phi^{(n)}(y_0) =$

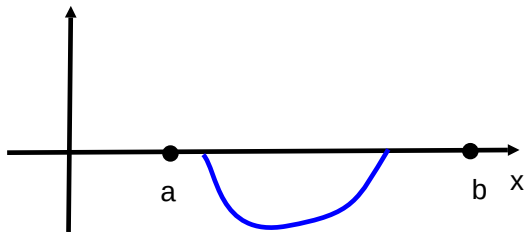
Weil ϕ mindestens $n+2$ Nulstellen hat, hat deren Ableitung ϕ' hat mindestens $n+1$ Nullstellen. (Auf jedem Intervall zwischen zwei Nullstellen gibt es eine Nullstelle von ϕ'.)



Ähnlich, die zweite Ableitung ϕ'' hat mindestens n Nullstellen U.S.W. Also, die $(n+1)$te Ableitung $\phi^{(n+1)}$ hat mindestens eine Nulstelle. Sei y_0 diese Nulstelle.
Da $\phi^{(n+1)} = f^{(n+1)} - \underbrace{P^{(n+1)}}_{0,\text{weil } P \text{ Grad n hat}} - K \underbrace{\omega^{(n+1)}}_{(n+1)!\text{weil } P \text{ Grad n+1 hat}}$.

Also,
$\phi^{(n+1)} = f^{(n+1)} - K(n+1)!$,und deswegen
$\phi^{(n)}(y_0) = f^{(n)}(y_0) - K(n+1)!$, und deswegen

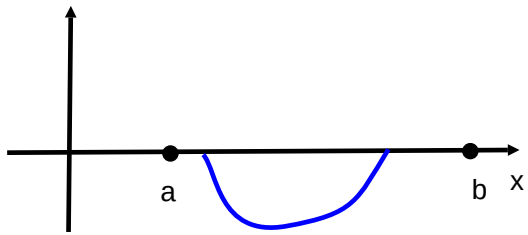
Weil ϕ mindestens $n+2$ Nulstellen hat, hat deren Ableitung ϕ' hat mindestens $n+1$ Nullstellen. (Auf jedem Intervall zwischen zwei Nullstellen gibt es eine Nullstelle von ϕ'.)



Ähnlich, die zweite Ableitung ϕ'' hat mindestens n Nullstellen U.S.W. Also, die $(n+1)$te Ableitung $\phi^{(n+1)}$ hat mindestens eine Nulstelle. Sei y_0 diese Nulstelle.
Da $\phi^{(n+1)} = f^{(n+1)} - \underbrace{P^{(n+1)}}_{0,\text{weil } P \text{ Grad n hat}} - K \underbrace{\omega^{(n+1)}}_{(n+1)!\text{weil } P \text{ Grad n+1 hat}}$.

Also,
$\phi^{(n+1)} = f^{(n+1)} - K(n+1)!$,und deswegen
$\phi^{(n)}(y_0) = f^{(n)}(y_0) - K(n+1)!$, und deswegen $K = \frac{f^{(n+1)}(y_0)}{(n+1)!}$.

Weil ϕ mindestens $n+2$ Nulstellen hat, hat deren Ableitung ϕ' hat mindestens $n+1$ Nullstellen. (Auf jedem Intervall zwischen zwei Nullstellen gibt es eine Nullstelle von ϕ'.)

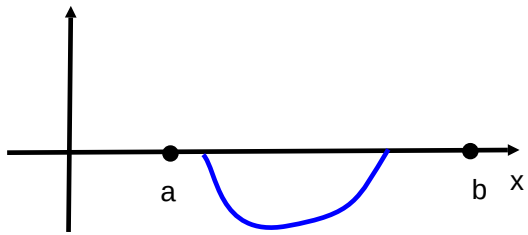


Ähnlich, die zweite Ableitung ϕ'' hat mindestens n Nullstellen U.S.W. Also, die $(n+1)$te Ableitung $\phi^{(n+1)}$ hat mindestens eine Nulstelle. Sei y_0 diese Nulstelle.
Da $\phi^{(n+1)} = f^{(n+1)} - \underbrace{P^{(n+1)}}_{0,\text{weil } P \text{ Grad n hat}} - K \underbrace{\omega^{(n+1)}}_{(n+1)!\text{weil } P \text{ Grad n+1 hat}}$.

Also,
$\phi^{(n+1)} = f^{(n+1)} - K(n+1)!$,und deswegen
$\phi^{(n)}(y_0) = f^{(n)}(y_0) - K(n+1)!$, und deswegen $K = \frac{f^{(n+1)}(y_0)}{(n+1)!}$.

Also, $f(y) - P(y) =$

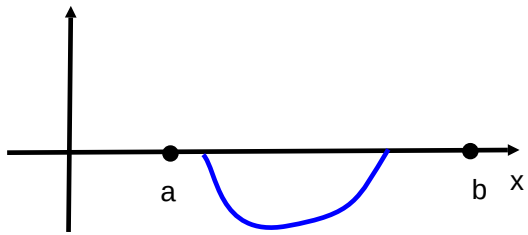
Weil ϕ mindestens $n+2$ Nulstellen hat, hat deren Ableitung ϕ' hat mindestens $n+1$ Nullstellen. (Auf jedem Intervall zwischen zwei Nullstellen gibt es eine Nullstelle von ϕ'.)



Ähnlich, die zweite Ableitung ϕ'' hat mindestens n Nullstellen U.S.W. Also, die $(n+1)$te Ableitung $\phi^{(n+1)}$ hat mindestens eine Nulstelle. Sei y_0 diese Nulstelle.
Da $\phi^{(n+1)} = f^{(n+1)} - \underbrace{P^{(n+1)}}_{0,\text{weil } P \text{ Grad n hat}} - K \underbrace{\omega^{(n+1)}}_{(n+1)!\text{weil } P \text{ Grad n+1 hat}}$.

Also,
$\phi^{(n+1)} = f^{(n+1)} - K(n+1)!$,und deswegen
$\phi^{(n)}(y_0) = f^{(n)}(y_0) - K(n+1)!$, und deswegen $K = \frac{f^{(n+1)}(y_0)}{(n+1)!}$.
Also, $f(y) - P(y) = \frac{f^{(n+1)}(y_0)\omega(y_0)}{(n+1)!}$.

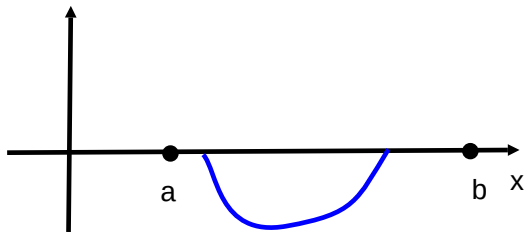
Weil ϕ mindestens $n+2$ Nulstellen hat, hat deren Ableitung ϕ' hat mindestens $n+1$ Nullstellen. (Auf jedem Intervall zwischen zwei Nullstellen gibt es eine Nullstelle von ϕ'.)



Ähnlich, die zweite Ableitung ϕ'' hat mindestens n Nullstellen U.S.W. Also, die $(n+1)$te Ableitung $\phi^{(n+1)}$ hat mindestens eine Nulstelle. Sei y_0 diese Nulstelle.
Da $\phi^{(n+1)} = f^{(n+1)} - \underbrace{P^{(n+1)}}_{0,\text{weil } P \text{ Grad n hat}} - K \underbrace{\omega^{(n+1)}}_{(n+1)!\text{weil } P \text{ Grad n+1 hat}}$.

Also,
$\phi^{(n+1)} = f^{(n+1)} - K(n+1)!$,und deswegen
$\phi^{(n)}(y_0) = f^{(n)}(y_0) - K(n+1)!$, und deswegen $K = \frac{f^{(n+1)}(y_0)}{(n+1)!}$.
Also, $f(y) - P(y) = \frac{f^{(n+1)}(y_0)\omega(y_0)}{(n+1)!}$. Da $(n+1)!$ sehr schnell wächst,

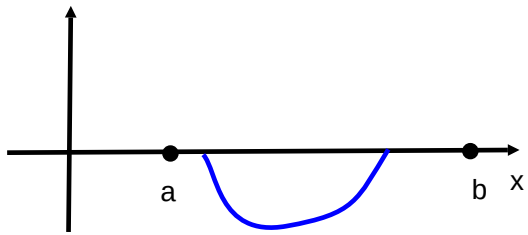
Weil ϕ mindestens $n+2$ Nulstellen hat, hat deren Ableitung ϕ' hat mindestens $n+1$ Nullstellen. (Auf jedem Intervall zwischen zwei Nullstellen gibt es eine Nullstelle von ϕ'.)



Ähnlich, die zweite Ableitung ϕ'' hat mindestens n Nullstellen U.S.W. Also, die $(n+1)$te Ableitung $\phi^{(n+1)}$ hat mindestens eine Nulstelle. Sei y_0 diese Nulstelle.
Da $\phi^{(n+1)} = f^{(n+1)} - \underbrace{P^{(n+1)}}_{0,\text{weil } P \text{ Grad n hat}} - K \underbrace{\omega^{(n+1)}}_{(n+1)!\text{weil } P \text{ Grad n+1 hat}}$.

Also,
$\phi^{(n+1)} = f^{(n+1)} - K(n+1)!$,und deswegen
$\phi^{(n)}(y_0) = f^{(n)}(y_0) - K(n+1)!$, und deswegen $K = \frac{f^{(n+1)}(y_0)}{(n+1)!}$.
Also, $f(y) - P(y) = \frac{f^{(n+1)}(y_0)\omega(y_0)}{(n+1)!}$. Da $(n+1)!$ sehr schnell wächst, approximiert P die Funktion f sehr gut,

Weil ϕ mindestens $n+2$ Nulstellen hat, hat deren Ableitung ϕ' hat mindestens $n+1$ Nullstellen. (Auf jedem Intervall zwischen zwei Nullstellen gibt es eine Nullstelle von ϕ'.)



Ähnlich, die zweite Ableitung ϕ'' hat mindestens n Nullstellen U.S.W. Also, die $(n+1)$te Ableitung $\phi^{(n+1)}$ hat mindestens eine Nulstelle. Sei y_0 diese Nulstelle.
Da $\phi^{(n+1)} = f^{(n+1)} - \underbrace{P^{(n+1)}}_{0,\text{weil } P \text{ Grad n hat}} - K \underbrace{\omega^{(n+1)}}_{(n+1)!\text{weil } P \text{ Grad n+1 hat}}$.

Also,
$\phi^{(n+1)} = f^{(n+1)} - K(n+1)!$,und deswegen
$\phi^{(n)}(y_0) = f^{(n)}(y_0) - K(n+1)!$, und deswegen $K = \frac{f^{(n+1)}(y_0)}{(n+1)!}$.
Also, $f(y) - P(y) = \frac{f^{(n+1)}(y_0)\omega(y_0)}{(n+1)!}$. Da $(n+1)!$ sehr schnell wächst, approximiert P die Funktion f sehr gut, falls n groß genug ist, und falls die höhere Ableitungen von f beschränkt sind.

Anwendungen von Lagrange-Polynomen

Anwendungen von Lagrange-Polynomen

Wiederholung

Anwendungen von Lagrange-Polynomen

Wiederholung *Sei f eine Funktion auf $[a, b]$*

Anwendungen von Lagrange-Polynomen

Wiederholung *Sei f eine Funktion auf $[a, b]$ und $x_0, ..., x_n \in [a, b]$ Punkte.*

Anwendungen von Lagrange-Polynomen

Wiederholung *Sei f eine Funktion auf $[a, b]$ und $x_0, ..., x_n \in [a, b]$ Punkte. Das zugehörige Lagrange-Polynom ist ein Polynom P*

Anwendungen von Lagrange-Polynomen

Wiederholung *Sei f eine Funktion auf $[a, b]$ und $x_0, ..., x_n \in [a, b]$ Punkte. Das zugehörige Lagrange-Polynom ist ein Polynom P des Grades n,*

Anwendungen von Lagrange-Polynomen

Wiederholung *Sei f eine Funktion auf $[a, b]$ und $x_0, ..., x_n \in [a, b]$ Punkte. Das zugehörige Lagrange-Polynom ist ein Polynom P des Grades n, s.d. $P(x_i) = f(x_i)$.*

Anwendungen von Lagrange-Polynomen

Wiederholung *Sei f eine Funktion auf $[a, b]$ und $x_0, ..., x_n \in [a, b]$ Punkte. Das zugehörige Lagrange-Polynom ist ein Polynom P des Grades n, s.d. $P(x_i) = f(x_i)$. Falls die Funktion nicht „wild“ ist,*

Anwendungen von Lagrange-Polynomen

Wiederholung *Sei f eine Funktion auf $[a, b]$ und $x_0, ..., x_n \in [a, b]$ Punkte. Das zugehörige Lagrange-Polynom ist ein Polynom P des Grades n, s.d. $P(x_i) = f(x_i)$. Falls die Funktion nicht „wild“ ist, approxiemiert das Lagrange-Polynom die Funktion ziemlich gut*

Anwendungen von Lagrange-Polynomen

Wiederholung *Sei f eine Funktion auf $[a, b]$ und $x_0, ..., x_n \in [a, b]$ Punkte. Das zugehörige Lagrange-Polynom ist ein Polynom P des Grades n, s.d. $P(x_i) = f(x_i)$. Falls die Funktion nicht „wild“ ist, approxiemiert das Lagrange-Polynom die Funktion ziemlich gut*

- Anwendung in Visualisierung

Anwendungen von Lagrange-Polynomen

Wiederholung *Sei f eine Funktion auf $[a, b]$ und $x_0, ..., x_n \in [a, b]$ Punkte. Das zugehörige Lagrange-Polynom ist ein Polynom P des Grades n, s.d. $P(x_i) = f(x_i)$. Falls die Funktion nicht „wild“ ist, approxiemiert das Lagrange-Polynom die Funktion ziemlich gut*

- Anwendung in Visualisierung
  Bei Visualisierung von komplexen Objekten

Anwendungen von Lagrange-Polynomen

Wiederholung *Sei f eine Funktion auf $[a, b]$ und $x_0, ..., x_n \in [a, b]$ Punkte. Das zugehörige Lagrange-Polynom ist ein Polynom P des Grades n, s.d. $P(x_i) = f(x_i)$. Falls die Funktion nicht „wild“ ist, approxiemiert das Lagrange-Polynom die Funktion ziemlich gut*

- Anwendung in Visualisierung
  Bei Visualisierung von komplexen Objekten (z.B. bei erstellen eines Trickfilms)

Anwendungen von Lagrange-Polynomen

Wiederholung *Sei f eine Funktion auf $[a, b]$ und $x_0, ..., x_n \in [a, b]$ Punkte. Das zugehörige Lagrange-Polynom ist ein Polynom P des Grades n, s.d. $P(x_i) = f(x_i)$. Falls die Funktion nicht „wild“ ist, approxiemiert das Lagrange-Polynom die Funktion ziemlich gut*

- Anwendung in Visualisierung
  Bei Visualisierung von komplexen Objekten (z.B. bei erstellen eines Trickfilms) berechnet man nur die Lage von endlich vielen Punkten

Anwendungen von Lagrange-Polynomen

Wiederholung *Sei f eine Funktion auf $[a, b]$ und $x_0, ..., x_n \in [a, b]$ Punkte. Das zugehörige Lagrange-Polynom ist ein Polynom P des Grades n, s.d. $P(x_i) = f(x_i)$. Falls die Funktion nicht „wild“ ist, approxiemiert das Lagrange-Polynom die Funktion ziemlich gut*

- Anwendung in Visualisierung
  Bei Visualisierung von komplexen Objekten (z.B. bei erstellen eines Trickfilms) berechnet man nur die Lage von endlich vielen Punkten (so gen. Noden).

Anwendungen von Lagrange-Polynomen

Wiederholung *Sei f eine Funktion auf $[a, b]$ und $x_0, ..., x_n \in [a, b]$ Punkte. Das zugehörige Lagrange-Polynom ist ein Polynom P des Grades n, s.d. $P(x_i) = f(x_i)$. Falls die Funktion nicht „wild“ ist, approxiemiert das Lagrange-Polynom die Funktion ziemlich gut*

- Anwendung in Visualisierung
  Bei Visualisierung von komplexen Objekten (z.B. bei erstellen eines Trickfilms) berechnet man nur die Lage von endlich vielen Punkten (so gen. Noden). Das Objekt zeichnet man dann u.a. mit Hilfe des Lagrange-Polynoms,

Anwendungen von Lagrange-Polynomen

Wiederholung *Sei f eine Funktion auf $[a, b]$ und $x_0, ..., x_n \in [a, b]$ Punkte. Das zugehörige Lagrange-Polynom ist ein Polynom P des Grades n, s.d. $P(x_i) = f(x_i)$. Falls die Funktion nicht „wild“ ist, approxiemiert das Lagrange-Polynom die Funktion ziemlich gut*

- Anwendung in Visualisierung
  Bei Visualisierung von komplexen Objekten (z.B. bei erstellen eines Trickfilms) berechnet man nur die Lage von endlich vielen Punkten (so gen. Noden). Das Objekt zeichnet man dann u.a. mit Hilfe des Lagrange-Polynoms, oder dessen Modifikationen.

Anwendungen von Lagrange-Polynomen

Wiederholung *Sei f eine Funktion auf $[a, b]$ und $x_0, ..., x_n \in [a, b]$ Punkte. Das zugehörige Lagrange-Polynom ist ein Polynom P des Grades n, s.d. $P(x_i) = f(x_i)$. Falls die Funktion nicht „wild“ ist, approxiemiert das Lagrange-Polynom die Funktion ziemlich gut*

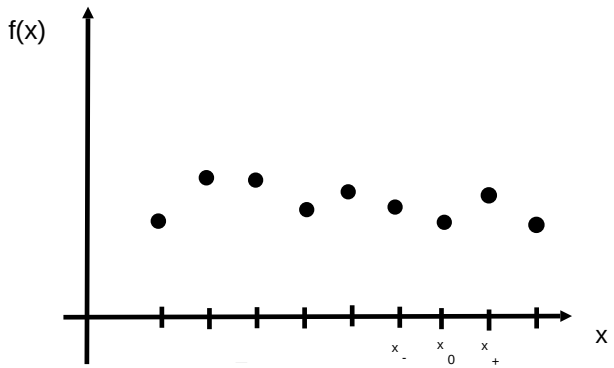
- Anwendung in Visualisierung
  Bei Visualisierung von komplexen Objekten (z.B. bei erstellen eines Trickfilms) berechnet man nur die Lage von endlich vielen Punkten (so gen. Noden). Das Objekt zeichnet man dann u.a. mit Hilfe des Lagrange-Polynoms, oder dessen Modifikationen.

Numerische Methoden, die Ableitung zu berechnen.

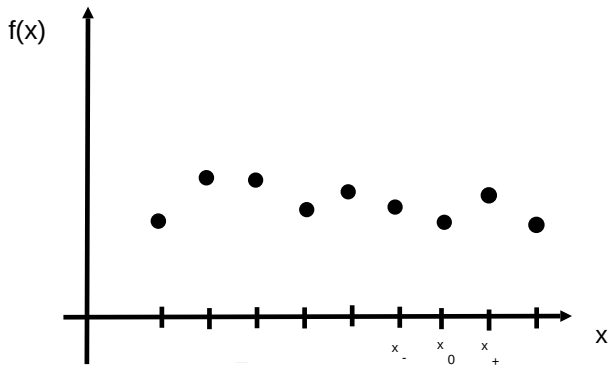
Numerische Methoden, die Ableitung zu berechnen.
Angenommen, wir haben die Werte einer Funktion f

Numerische Methoden, die Ableitung zu berechnen.
Angenommen, wir haben die Werte einer Funktion f in Punkten eines $\varepsilon-$Gitters

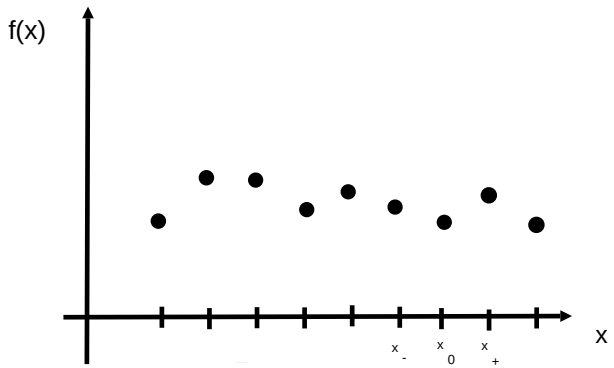
Numerische Methoden, die Ableitung zu berechnen.
Angenommen, wir haben die Werte einer Funktion f in Punkten eines $\varepsilon-$Gitters (d.h. in Punkten x, $x \pm \varepsilon, x \pm 2\varepsilon$, wobei ε klein ist),



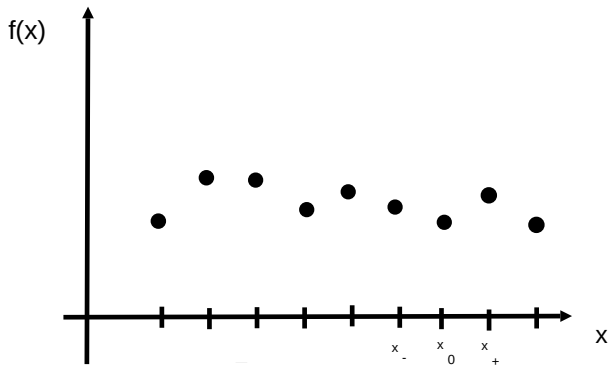
Numerische Methoden, die Ableitung zu berechnen.
Angenommen, wir haben die Werte einer Funktion f in Punkten eines $\varepsilon-$Gitters (d.h. in Punkten x, $x \pm \varepsilon, x \pm 2\varepsilon$, wobei ε klein ist), mit Hilfe eines numerischen oder technischen Verfahrens berechnet.



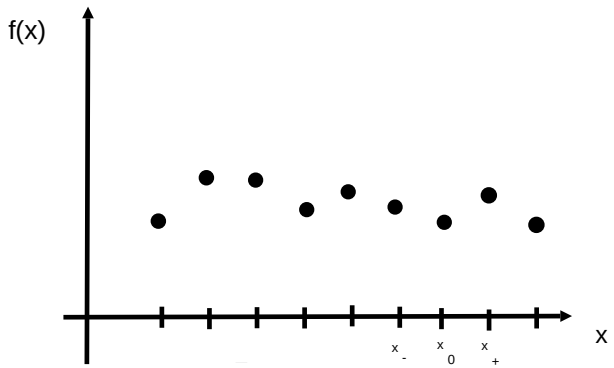
Numerische Methoden, die Ableitung zu berechnen.
Angenommen, wir haben die Werte einer Funktion f in Punkten eines $\varepsilon-$Gitters (d.h. in Punkten x, $x \pm \varepsilon, x \pm 2\varepsilon$, wobei ε klein ist), mit Hilfe eines numerischen oder technischen Verfahrens berechnet. Wie kann man deren $k-$Ableitung berechnen?



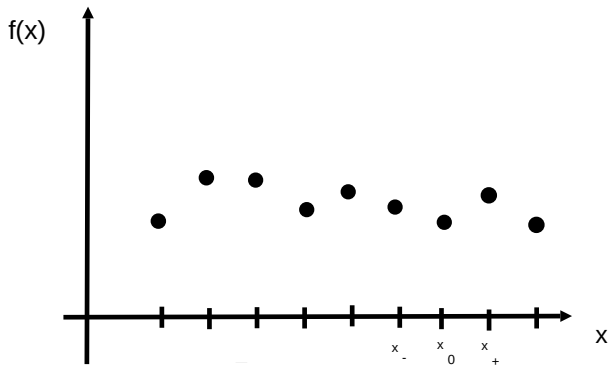
Numerische Methoden, die Ableitung zu berechnen.
Angenommen, wir haben die Werte einer Funktion f in Punkten eines $\varepsilon-$Gitters (d.h. in Punkten x, $x \pm \varepsilon, x \pm 2\varepsilon$, wobei ε klein ist), mit Hilfe eines numerischen oder technischen Verfahrens berechnet. Wie kann man deren $k-$Ableitung berechnen? Ein Vorschlag: Nehme $2n+1 > k$ nebeneinander liegenden Punkten des Gitters,



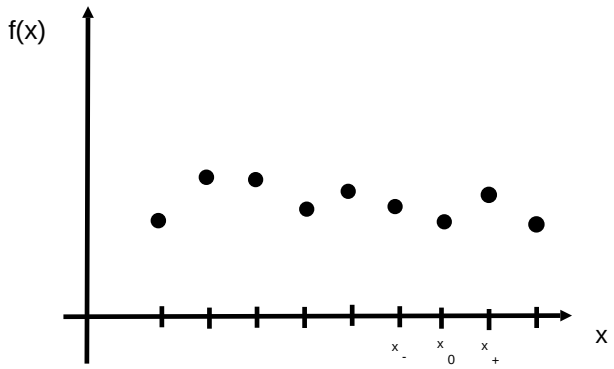
Numerische Methoden, die Ableitung zu berechnen.
Angenommen, wir haben die Werte einer Funktion f in Punkten eines $\varepsilon-$Gitters (d.h. in Punkten x, $x \pm \varepsilon, x \pm 2\varepsilon$, wobei ε klein ist), mit Hilfe eines numerischen oder technischen Verfahrens berechnet. Wie kann man deren $k-$Ableitung berechnen? Ein Vorschlag: Nehme $2n+1 > k$ nebeneinander liegenden Punkten des Gitters, z.B.
$x_0, x_{\pm 1} = x_0 \pm \varepsilon, ..., x_{\pm n} = x_0 \pm n\varepsilon,$



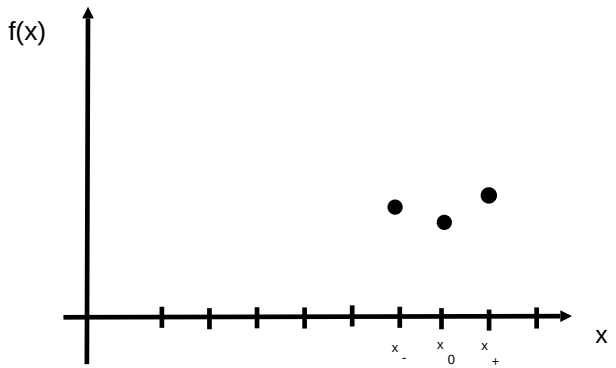
Numerische Methoden, die Ableitung zu berechnen.
Angenommen, wir haben die Werte einer Funktion f in Punkten eines $\varepsilon-$Gitters (d.h. in Punkten x, $x \pm \varepsilon, x \pm 2\varepsilon$, wobei ε klein ist), mit Hilfe eines numerischen oder technischen Verfahrens berechnet. Wie kann man deren $k-$Ableitung berechnen? Ein Vorschlag: Nehme $2n+1 > k$ nebeneinander liegenden Punkten des Gitters, z.B.
$x_0, x_{\pm 1} = x_0 \pm \varepsilon, ..., x_{\pm n} = x_0 \pm n\varepsilon$, konstruiere das Lagrange-Polynom s.d. $P(x_{\pm i}) = f(x_{\pm i})$, und berechne deren $k-$te Ableitung im x_0.



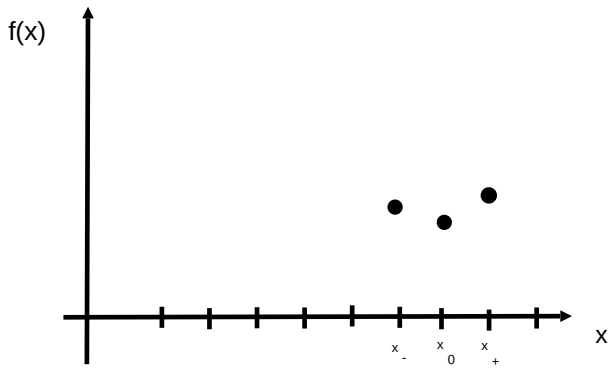
Numerische Methoden, die Ableitung zu berechnen.
Angenommen, wir haben die Werte einer Funktion f in Punkten eines $\varepsilon-$Gitters (d.h. in Punkten x, $x \pm \varepsilon, x \pm 2\varepsilon$, wobei ε klein ist), mit Hilfe eines numerischen oder technischen Verfahrens berechnet. Wie kann man deren $k-$Ableitung berechnen? Ein Vorschlag: Nehme $2n+1 > k$ nebeneinander liegenden Punkten des Gitters, z.B.
$x_0, x_{\pm 1} = x_0 \pm \varepsilon, ..., x_{\pm n} = x_0 \pm n\varepsilon$, konstruiere das Lagrange-Polynom s.d. $P(x_{\pm i}) = f(x_{\pm i})$, und berechne deren $k-$te Ableitung im x_0.
(Mann kann nur $k+1$ Punkte aus $\{x_{\pm n}, ..., x_0\}$ nehmen)



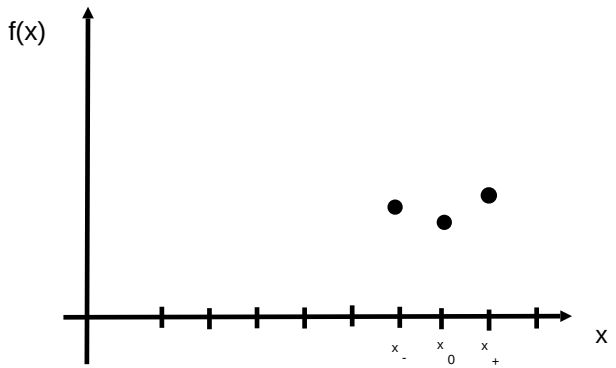
Numerische Methoden, die Ableitung zu berechnen.
Angenommen, wir haben die Werte einer Funktion f in Punkten eines $\varepsilon-$Gitters (d.h. in Punkten x, $x \pm \varepsilon, x \pm 2\varepsilon$, wobei ε klein ist), mit Hilfe eines numerischen oder technischen Verfahrens berechnet. Wie kann man deren $k-$Ableitung berechnen? Ein Vorschlag: Nehme $2n+1 > k$ nebeneinander liegenden Punkten des Gitters, z.B.
$x_0, x_{\pm 1} = x_0 \pm \varepsilon, ..., x_{\pm n} = x_0 \pm n\varepsilon$, konstruiere das Lagrange-Polynom s.d. $P(x_{\pm i}) = f(x_{\pm i})$, und berechne deren $k-$te Ableitung im x_0.
(Mann kann nur $k+1$ Punkte aus $\{x_{\pm n}, ..., x_0\}$ nehmen)
Z.B. Gegeben sind $f(x_0 - \varepsilon)$, $f(x_0 + \varepsilon)$.



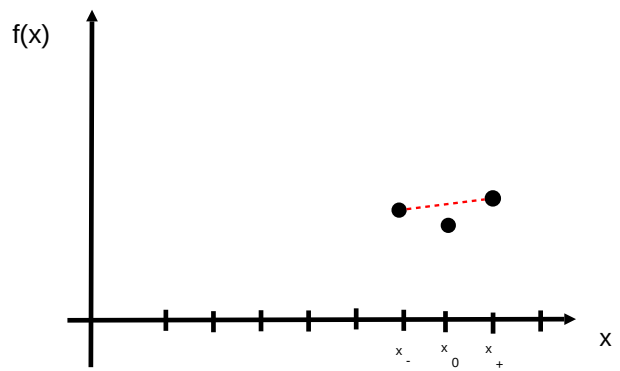
Numerische Methoden, die Ableitung zu berechnen.
Angenommen, wir haben die Werte einer Funktion f in Punkten eines $\varepsilon-$Gitters (d.h. in Punkten x, $x \pm \varepsilon, x \pm 2\varepsilon$, wobei ε klein ist), mit Hilfe eines numerischen oder technischen Verfahrens berechnet. Wie kann man deren $k-$Ableitung berechnen? Ein Vorschlag: Nehme $2n+1 > k$ nebeneinander liegenden Punkten des Gitters, z.B.
$x_0, x_{\pm 1} = x_0 \pm \varepsilon, ..., x_{\pm n} = x_0 \pm n\varepsilon$, konstruiere das Lagrange-Polynom s.d. $P(x_{\pm i}) = f(x_{\pm i})$, und berechne deren $k-$te Ableitung im x_0.
(Mann kann nur $k+1$ Punkte aus $\{x_{\pm n}, ..., x_0\}$ nehmen)
Z.B. Gegeben sind $f(x_0 - \varepsilon)$, $f(x_0 + \varepsilon)$. Zu berechnen (approximativ): $f'(x_0)$.



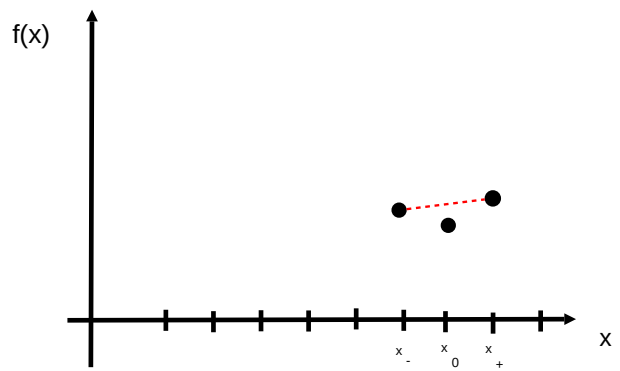
Numerische Methoden, die Ableitung zu berechnen.
Angenommen, wir haben die Werte einer Funktion f in Punkten eines $\varepsilon-$Gitters (d.h. in Punkten x, $x \pm \varepsilon, x \pm 2\varepsilon$, wobei ε klein ist), mit Hilfe eines numerischen oder technischen Verfahrens berechnet. Wie kann man deren $k-$Ableitung berechnen? Ein Vorschlag: Nehme $2n+1 > k$ nebeneinander liegenden Punkten des Gitters, z.B.
$x_0, x_{\pm 1} = x_0 \pm \varepsilon, ..., x_{\pm n} = x_0 \pm n\varepsilon$, konstruiere das Lagrange-Polynom s.d. $P(x_{\pm i}) = f(x_{\pm i})$, und berechne deren $k-$te Ableitung im x_0.
(Mann kann nur $k+1$ Punkte aus $\{x_{\pm n}, ..., x_0\}$ nehmen)
Z.B. Gegeben sind $f(x_0 - \varepsilon)$, $f(x_0 + \varepsilon)$. Zu berechnen (approximativ): $f'(x_0)$.
Lagrange-Polynom P mit $P(x_0 \pm \varepsilon) = f(x_0 \pm \varepsilon)$ ist



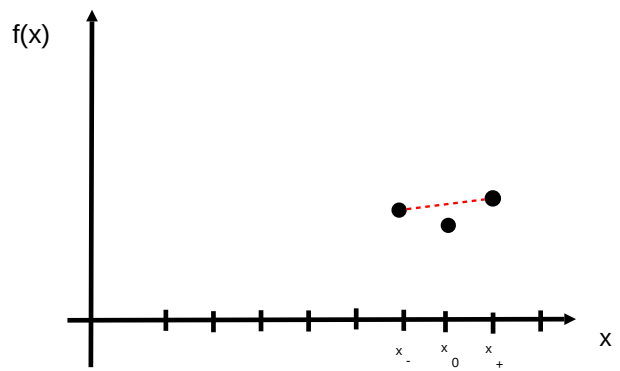
Numerische Methoden, die Ableitung zu berechnen.
Angenommen, wir haben die Werte einer Funktion f in Punkten eines $\varepsilon-$Gitters (d.h. in Punkten x, $x \pm \varepsilon, x \pm 2\varepsilon$, wobei ε klein ist), mit Hilfe eines numerischen oder technischen Verfahrens berechnet. Wie kann man deren $k-$Ableitung berechnen? Ein Vorschlag: Nehme $2n+1 > k$ nebeneinander liegenden Punkten des Gitters, z.B.
$x_0, x_{\pm 1} = x_0 \pm \varepsilon, ..., x_{\pm n} = x_0 \pm n\varepsilon$, konstruiere das Lagrange-Polynom s.d. $P(x_{\pm i}) = f(x_{\pm i})$, und berechne deren $k-$te Ableitung im x_0.
(Mann kann nur $k+1$ Punkte aus $\{x_{\pm n}, ..., x_0\}$ nehmen)
Z.B. Gegeben sind $f(x_0 - \varepsilon)$, $f(x_0 + \varepsilon)$. Zu berechnen (approximativ): $f'(x_0)$.
Lagrange-Polynom P mit $P(x_0 \pm \varepsilon) = f(x_0 \pm \varepsilon)$ ist
$\frac{x-(x_0+\varepsilon)}{(x_0-\varepsilon)-(x_0+\varepsilon)} f(x_0 - \varepsilon) +$



Numerische Methoden, die Ableitung zu berechnen.
Angenommen, wir haben die Werte einer Funktion f in Punkten eines $\varepsilon-$Gitters (d.h. in Punkten x, $x \pm \varepsilon, x \pm 2\varepsilon$, wobei ε klein ist), mit Hilfe eines numerischen oder technischen Verfahrens berechnet. Wie kann man deren $k-$Ableitung berechnen? Ein Vorschlag: Nehme $2n+1 > k$ nebeneinander liegenden Punkten des Gitters, z.B.
$x_0, x_{\pm 1} = x_0 \pm \varepsilon, ..., x_{\pm n} = x_0 \pm n\varepsilon$, konstruiere das Lagrange-Polynom s.d. $P(x_{\pm i}) = f(x_{\pm i})$, und berechne deren $k-$te Ableitung im x_0.
(Mann kann nur $k+1$ Punkte aus $\{x_{\pm n}, ..., x_0\}$ nehmen)
Z.B. Gegeben sind $f(x_0 - \varepsilon)$, $f(x_0 + \varepsilon)$. Zu berechnen (approximativ): $f'(x_0)$.
Lagrange-Polynom P mit $P(x_0 \pm \varepsilon) = f(x_0 \pm \varepsilon)$ ist
$\frac{x-(x_0+\varepsilon)}{(x_0-\varepsilon)-(x_0+\varepsilon)} f(x_0 - \varepsilon) + \frac{x-(x_0-\varepsilon)}{(x_0+\varepsilon)-(x_0-\varepsilon)} f(x_0 + \varepsilon) =$

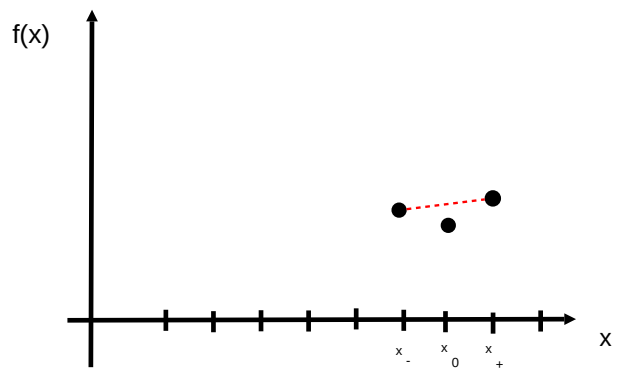


Numerische Methoden, die Ableitung zu berechnen.
Angenommen, wir haben die Werte einer Funktion f in Punkten eines $\varepsilon-$Gitters (d.h. in Punkten x, $x \pm \varepsilon, x \pm 2\varepsilon$, wobei ε klein ist), mit Hilfe eines numerischen oder technischen Verfahrens berechnet. Wie kann man deren $k-$Ableitung berechnen? Ein Vorschlag: Nehme $2n+1 > k$ nebeneinander liegenden Punkten des Gitters, z.B.
$x_0, x_{\pm 1} = x_0 \pm \varepsilon, ..., x_{\pm n} = x_0 \pm n\varepsilon$, konstruiere das Lagrange-Polynom s.d. $P(x_{\pm i}) = f(x_{\pm i})$, und berechne deren $k-$te Ableitung im x_0.
(Mann kann nur $k+1$ Punkte aus $\{x_{\pm n}, ..., x_0\}$ nehmen)
Z.B. Gegeben sind $f(x_0 - \varepsilon)$, $f(x_0 + \varepsilon)$. Zu berechnen (approximativ): $f'(x_0)$.
Lagrange-Polynom P mit $P(x_0 \pm \varepsilon) = f(x_0 \pm \varepsilon)$ ist
$\frac{x-(x_0+\varepsilon)}{(x_0-\varepsilon)-(x_0+\varepsilon)} f(x_0 - \varepsilon) + \frac{x-(x_0-\varepsilon)}{(x_0+\varepsilon)-(x_0-\varepsilon)} f(x_0 + \varepsilon) = \frac{f(x_0+\varepsilon)-f(x_0-\varepsilon)}{2\varepsilon} x +$



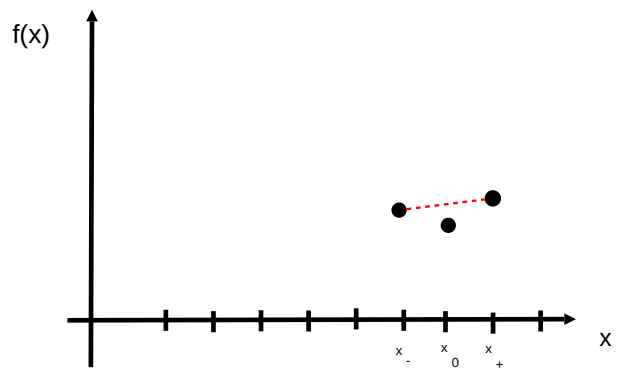
Numerische Methoden, die Ableitung zu berechnen.
Angenommen, wir haben die Werte einer Funktion f in Punkten eines $\varepsilon-$Gitters (d.h. in Punkten x, $x \pm \varepsilon, x \pm 2\varepsilon$, wobei ε klein ist), mit Hilfe eines numerischen oder technischen Verfahrens berechnet. Wie kann man deren $k-$Ableitung berechnen? Ein Vorschlag: Nehme $2n+1 > k$ nebeneinander liegenden Punkten des Gitters, z.B.
$x_0, x_{\pm 1} = x_0 \pm \varepsilon, ..., x_{\pm n} = x_0 \pm n\varepsilon$, konstruiere das Lagrange-Polynom s.d. $P(x_{\pm i}) = f(x_{\pm i})$, und berechne deren $k-$te Ableitung im x_0.
(Mann kann nur $k+1$ Punkte aus $\{x_{\pm n}, ..., x_0\}$ nehmen)
Z.B. Gegeben sind $f(x_0 - \varepsilon)$, $f(x_0 + \varepsilon)$. Zu berechnen (approximativ): $f'(x_0)$.
Lagrange-Polynom P mit $P(x_0 \pm \varepsilon) = f(x_0 \pm \varepsilon)$ ist

$$\frac{x-(x_0+\varepsilon)}{(x_0-\varepsilon)-(x_0+\varepsilon)} f(x_0-\varepsilon) + \frac{x-(x_0-\varepsilon)}{(x_0+\varepsilon)-(x_0-\varepsilon)} f(x_0+\varepsilon) = \frac{f(x_0+\varepsilon)-f(x_0-\varepsilon)}{2\varepsilon} x + \frac{(x_0-\varepsilon)f(x_0+\varepsilon)-(x_0+\varepsilon)f(x_0-\varepsilon)}{2\varepsilon}.$$

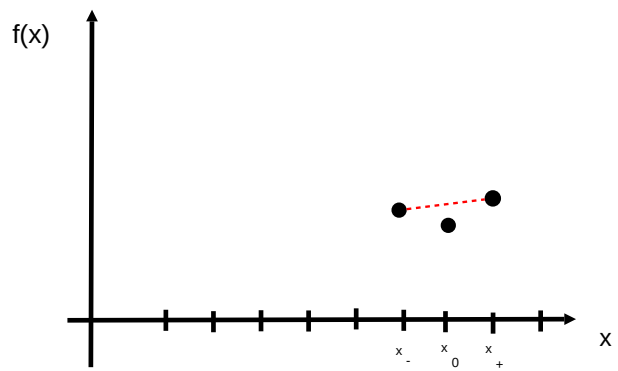


Numerische Methoden, die Ableitung zu berechnen.
Angenommen, wir haben die Werte einer Funktion f in Punkten eines $\varepsilon-$Gitters (d.h. in Punkten x, $x \pm \varepsilon, x \pm 2\varepsilon$, wobei ε klein ist), mit Hilfe eines numerischen oder technischen Verfahrens berechnet. Wie kann man deren $k-$Ableitung berechnen? Ein Vorschlag: Nehme $2n+1 > k$ nebeneinander liegenden Punkten des Gitters, z.B.
$x_0, x_{\pm 1} = x_0 \pm \varepsilon, ..., x_{\pm n} = x_0 \pm n\varepsilon$, konstruiere das Lagrange-Polynom s.d. $P(x_{\pm i}) = f(x_{\pm i})$, und berechne deren $k-$te Ableitung im x_0.
(Mann kann nur $k+1$ Punkte aus $\{x_{\pm n}, ..., x_0\}$ nehmen)
Z.B. Gegeben sind $f(x_0 - \varepsilon)$, $f(x_0 + \varepsilon)$. Zu berechnen (approximativ): $f'(x_0)$.
Lagrange-Polynom P mit $P(x_0 \pm \varepsilon) = f(x_0 \pm \varepsilon)$ ist

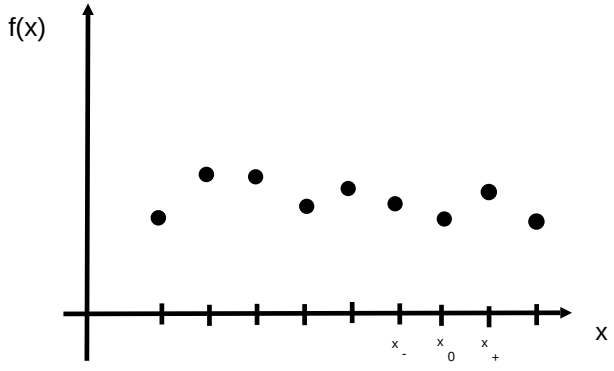
$$\frac{x-(x_0+\varepsilon)}{(x_0-\varepsilon)-(x_0+\varepsilon)} f(x_0-\varepsilon) + \frac{x-(x_0-\varepsilon)}{(x_0+\varepsilon)-(x_0-\varepsilon)} f(x_0+\varepsilon) = \frac{f(x_0+\varepsilon)-f(x_0-\varepsilon)}{2\varepsilon} x + \frac{(x_0-\varepsilon)f(x_0+\varepsilon)-(x_0+\varepsilon)f(x_0-\varepsilon)}{2\varepsilon}.$$



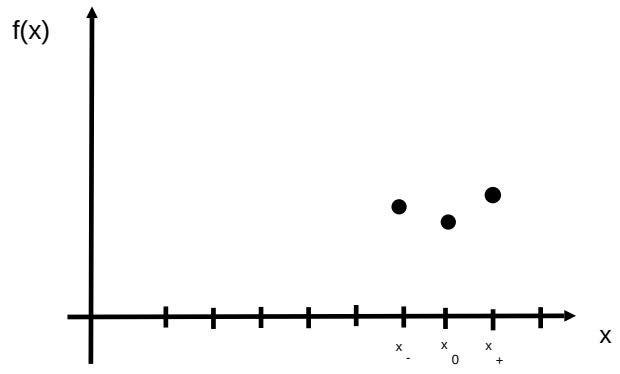
Numerische Methoden, die Ableitung zu berechnen.
Angenommen, wir haben die Werte einer Funktion f in Punkten eines $\varepsilon-$Gitters (d.h. in Punkten x, $x \pm \varepsilon, x \pm 2\varepsilon$, wobei ε klein ist), mit Hilfe eines numerischen oder technischen Verfahrens berechnet. Wie kann man deren $k-$Ableitung berechnen? Ein Vorschlag: Nehme $2n+1 > k$ nebeneinander liegenden Punkten des Gitters, z.B.
$x_0, x_{\pm 1} = x_0 \pm \varepsilon, ..., x_{\pm n} = x_0 \pm n\varepsilon$, konstruiere das Lagrange-Polynom s.d. $P(x_{\pm i}) = f(x_{\pm i})$, und berechne deren $k-$te Ableitung im x_0.
(Mann kann nur $k+1$ Punkte aus $\{x_{\pm n}, ..., x_0\}$ nehmen)
Z.B. Gegeben sind $f(x_0 - \varepsilon)$, $f(x_0 + \varepsilon)$. Zu berechnen (approximativ): $f'(x_0)$.
Lagrange-Polynom P mit $P(x_0 \pm \varepsilon) = f(x_0 \pm \varepsilon)$ ist
$\frac{x-(x_0+\varepsilon)}{(x_0-\varepsilon)-(x_0+\varepsilon)} f(x_0 - \varepsilon) + \frac{x-(x_0-\varepsilon)}{(x_0+\varepsilon)-(x_0-\varepsilon)} f(x_0 + \varepsilon) = \frac{f(x_0+\varepsilon)-f(x_0-\varepsilon)}{2\varepsilon} x + \frac{(x_0-\varepsilon)f(x_0+\varepsilon)-(x_0+\varepsilon)f(x_0-\varepsilon)}{2\varepsilon}$.
Dessen Ableitung ist $\frac{f(x+\varepsilon)-f(x-\varepsilon)}{2\varepsilon} \approx f'(x_0)$.



Dasselbe mit der zweiten Ableitung

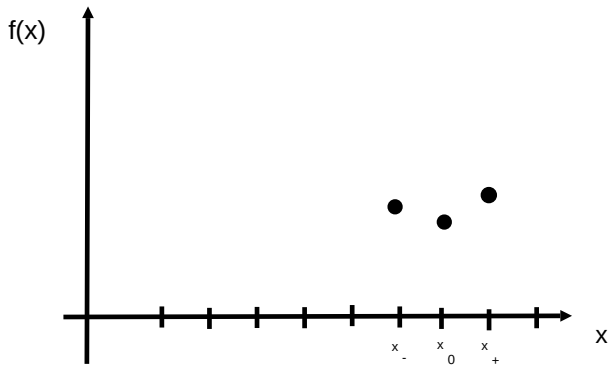


Dasselbe mit der zweiten Ableitung



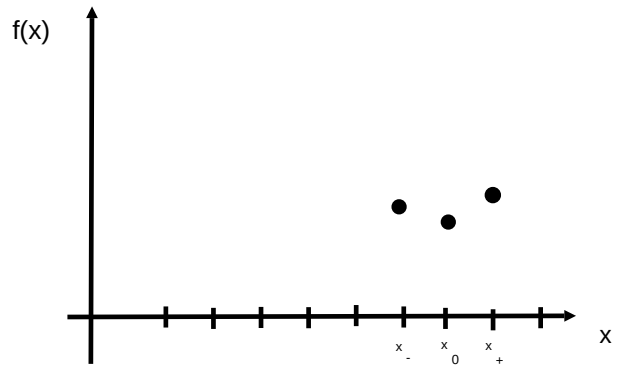
Bezeichne $x_+ := x_0 + \varepsilon$, $x_- := x_0 - \varepsilon$.

Dasselbe mit der zweiten Ableitung



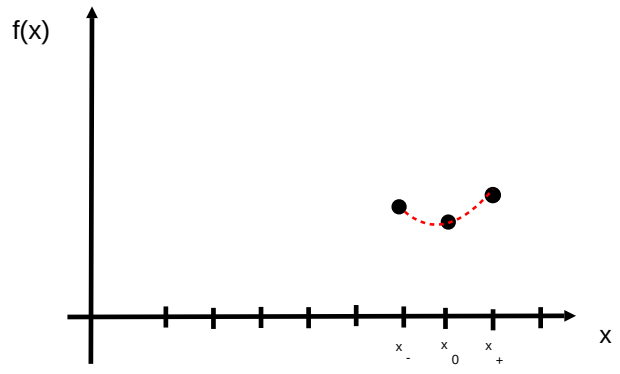
Bezeichne $x_+ := x_0 + \varepsilon$, $x_- := x_0 - \varepsilon$. Dann ist das Polynom 2tes Grades P

Dasselbe mit der zweiten Ableitung



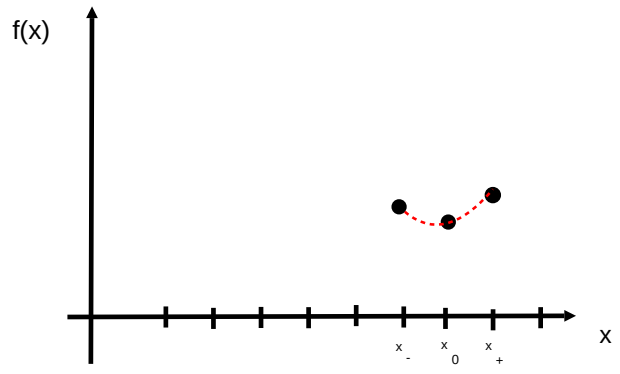
Bezeichne $x_+ := x_0 + \varepsilon$, $x_- := x_0 - \varepsilon$. Dann ist das Polynom 2tes Grades P mit $P(x_\pm) = f(x_\pm)$, $P(x_0) = f(x_0)$

Dasselbe mit der zweiten Ableitung



Bezeichne $x_+ := x_0 + \varepsilon$, $x_- := x_0 - \varepsilon$. Dann ist das Polynom 2tes Grades P mit $P(x_\pm) = f(x_\pm)$, $P(x_0) = f(x_0)$ gegeben durch

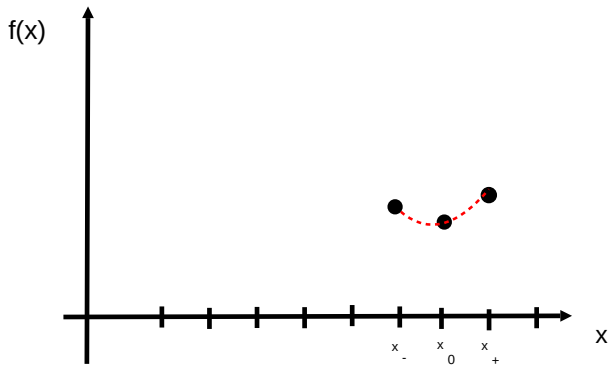
Dasselbe mit der zweiten Ableitung



Bezeichne $x_+ := x_0 + \varepsilon$, $x_- := x_0 - \varepsilon$. Dann ist das Polynom 2tes Grades P mit $P(x_\pm) = f(x_\pm)$, $P(x_0) = f(x_0)$ gegeben durch

$\frac{(x-x_0)(x-x_+)}{(x_- - x_0)(x_- - x_+)} f(x_-) +$

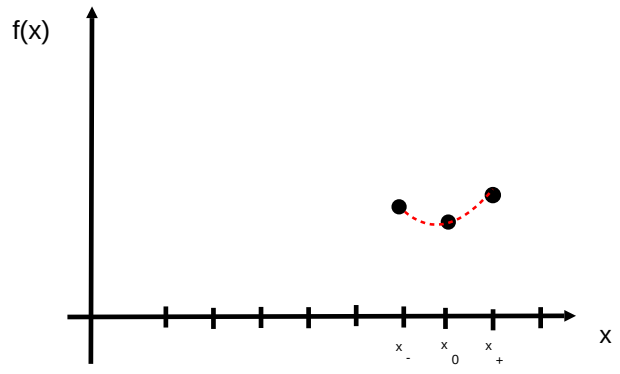
Dasselbe mit der zweiten Ableitung



Bezeichne $x_+ := x_0 + \varepsilon$, $x_- := x_0 - \varepsilon$. Dann ist das Polynom 2tes Grades P mit $P(x_\pm) = f(x_\pm)$, $P(x_0) = f(x_0)$ gegeben durch

$$\frac{(x-x_0)(x-x_+)}{(x_- - x_0)(x_- - x_+)} f(x_-) + \frac{(x-x_-)(x-x_+)}{(x_0 - x_-)(x_0 - x_+)} f(x_0) +$$

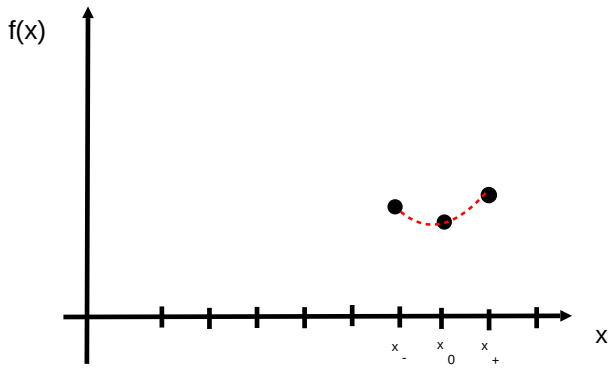
Dasselbe mit der zweiten Ableitung



Bezeichne $x_+ := x_0 + \varepsilon$, $x_- := x_0 - \varepsilon$. Dann ist das Polynom 2tes Grades P mit $P(x_\pm) = f(x_\pm)$, $P(x_0) = f(x_0)$ gegeben durch

$$\frac{(x-x_0)(x-x_+)}{(x_- -x_0)(x_- -x_+)} f(x_-) + \frac{(x-x_-)(x-x_+)}{(x_0-x_-)(x_0-x_+)} f(x_0) + \frac{(x-x_-)(x-x_0)}{(x_+-x_-)(x_+-x_0)} f(x_+)$$

Dasselbe mit der zweiten Ableitung

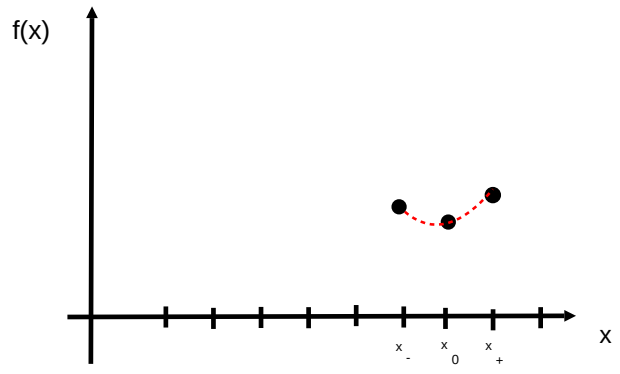


Bezeichne $x_+ := x_0 + \varepsilon$, $x_- := x_0 - \varepsilon$. Dann ist das Polynom 2tes Grades P mit $P(x_\pm) = f(x_\pm)$, $P(x_0) = f(x_0)$ gegeben durch

$$\frac{(x-x_0)(x-x_+)}{(x_- -x_0)(x_- -x_+)} f(x_-) + \frac{(x-x_-)(x-x_+)}{(x_0-x_-)(x_0-x_+)} f(x_0) + \frac{(x-x_-)(x-x_0)}{(x_+-x_-)(x_+-x_0)} f(x_+)$$

$$= x^2 \left[\frac{f(x_-)}{2\varepsilon^2} - \frac{f(x_0)}{\varepsilon^2} + \frac{f(x_+)}{2\varepsilon^2} \right] +$$

Dasselbe mit der zweiten Ableitung

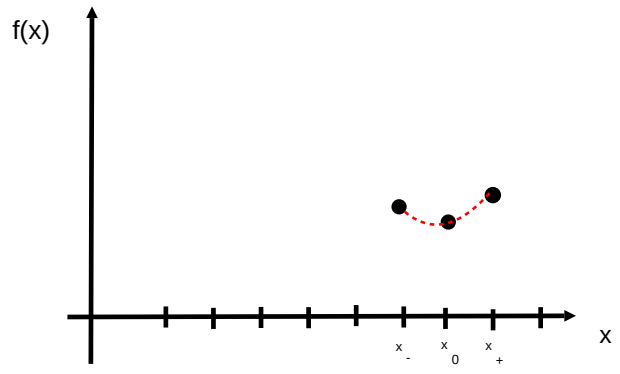


Bezeichne $x_+ := x_0 + \varepsilon$, $x_- := x_0 - \varepsilon$. Dann ist das Polynom 2tes Grades P mit $P(x_\pm) = f(x_\pm)$, $P(x_0) = f(x_0)$ gegeben durch

$$\frac{(x-x_0)(x-x_+)}{(x_- -x_0)(x_- -x_+)} f(x_-) + \frac{(x-x_-)(x-x_+)}{(x_0-x_-)(x_0-x_+)} f(x_0) + \frac{(x-x_-)(x-x_0)}{(x_+-x_-)(x_+-x_0)} f(x_+)$$

$$= x^2 \left[\frac{f(x_-)}{2\varepsilon^2} - \frac{f(x_0)}{\varepsilon^2} + \frac{f(x_+)}{2\varepsilon^2} \right] + \text{ Terme der kleineren Ordnung in } x$$

Dasselbe mit der zweiten Ableitung



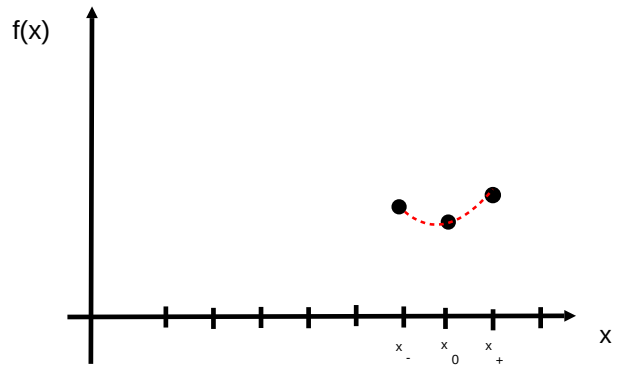
Bezeichne $x_+ := x_0 + \varepsilon$, $x_- := x_0 - \varepsilon$. Dann ist das Polynom 2tes Grades P mit $P(x_\pm) = f(x_\pm)$, $P(x_0) = f(x_0)$ gegeben durch

$$\frac{(x-x_0)(x-x_+)}{(x_- -x_0)(x_- -x_+)} f(x_-) + \frac{(x-x_-)(x-x_+)}{(x_0-x_-)(x_0-x_+)} f(x_0) + \frac{(x-x_-)(x-x_0)}{(x_+-x_-)(x_+-x_0)} f(x_+)$$

$$= x^2 \left[\frac{f(x_-)}{2\varepsilon^2} - \frac{f(x_0)}{\varepsilon^2} + \frac{f(x_+)}{2\varepsilon^2} \right] + \text{ Terme der kleineren Ordnung in } x$$

Dessen 2te Ableitung ist $\frac{f(x_-)+f(x_+)-2f(x_0)}{\varepsilon^2}$.

Dasselbe mit der zweiten Ableitung



Bezeichne $x_+ := x_0 + \varepsilon$, $x_- := x_0 - \varepsilon$. Dann ist das Polynom 2tes Grades P mit $P(x_\pm) = f(x_\pm)$, $P(x_0) = f(x_0)$ gegeben durch

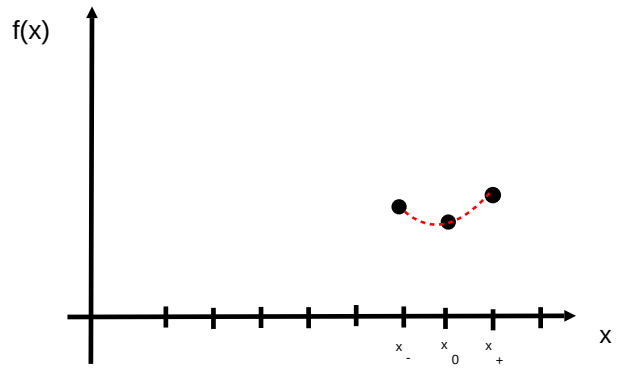
$$\frac{(x-x_0)(x-x_+)}{(x_- -x_0)(x_- -x_+)} f(x_-) + \frac{(x-x_-)(x-x_+)}{(x_0-x_-)(x_0-x_+)} f(x_0) + \frac{(x-x_-)(x-x_0)}{(x_+-x_-)(x_+-x_0)} f(x_+)$$

$$= x^2 \left[\frac{f(x_-)}{2\varepsilon^2} - \frac{f(x_0)}{\varepsilon^2} + \frac{f(x_+)}{2\varepsilon^2} \right] + \text{ Terme der kleineren Ordnung in } x$$

Dessen 2te Ableitung ist $\frac{f(x_-)+f(x_+)-2f(x_0)}{\varepsilon^2}$.

Also, $f''(x_0) \approx \frac{f(x_-)+f(x_+)-2f(x_0)}{\varepsilon^2}$.

Dasselbe mit der zweiten Ableitung



Bezeichne $x_+ := x_0 + \varepsilon$, $x_- := x_0 - \varepsilon$. Dann ist das Polynom 2tes Grades P mit $P(x_\pm) = f(x_\pm)$, $P(x_0) = f(x_0)$ gegeben durch

$$\frac{(x-x_0)(x-x_+)}{(x_- -x_0)(x_- -x_+)} f(x_-) + \frac{(x-x_-)(x-x_+)}{(x_0-x_-)(x_0-x_+)} f(x_0) + \frac{(x-x_-)(x-x_0)}{(x_+-x_-)(x_+-x_0)} f(x_+)$$

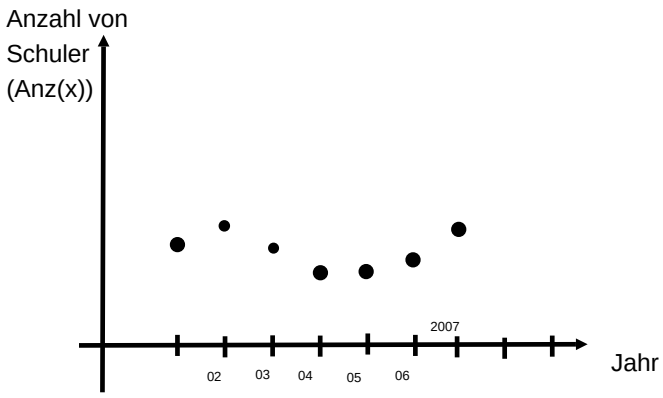
$$= x^2 \left[\frac{f(x_-)}{2\varepsilon^2} - \frac{f(x_0)}{\varepsilon^2} + \frac{f(x_+)}{2\varepsilon^2} \right] + \text{ Terme der kleineren Ordnung in } x$$

Dessen 2te Ableitung ist $\frac{f(x_-)+f(x_+)-2f(x_0)}{\varepsilon^2}$.

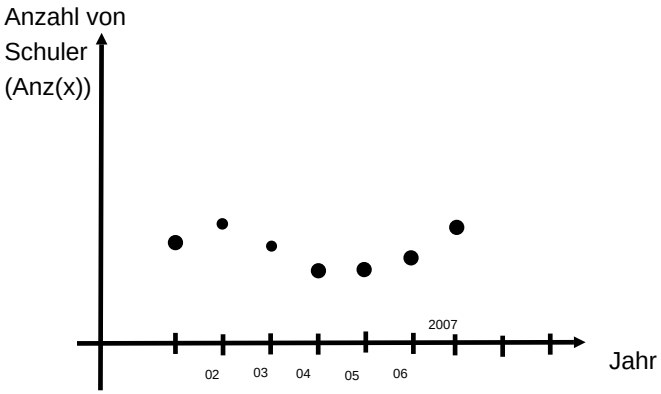
Also, $f''(x_0) \approx \frac{f(x_-)+f(x_+)-2f(x_0)}{\varepsilon^2}$. (Falls ε klein ist und die 3te Ableitung der Funktion nicht zu groß ist)

Zukunft vorhersagen.

Zukunft vorhersagen.

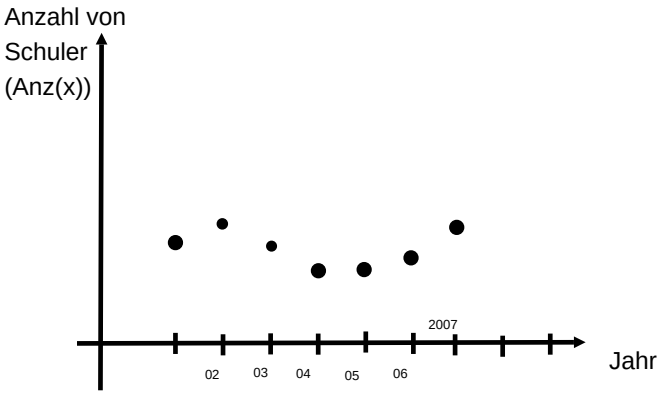


Zukunft vorhersagen.



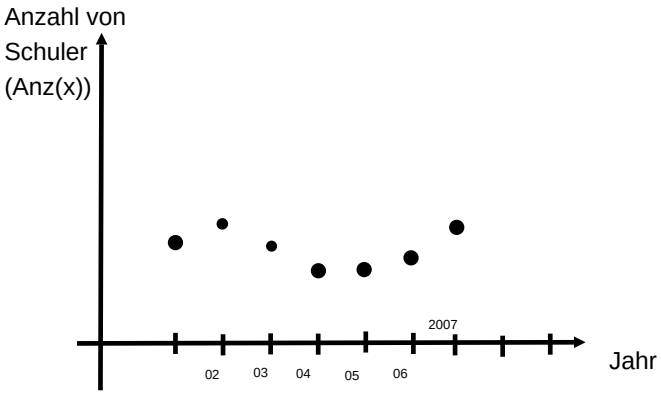
Was Passiert im Jahr 2007+1?

Zukunft vorhersagen.



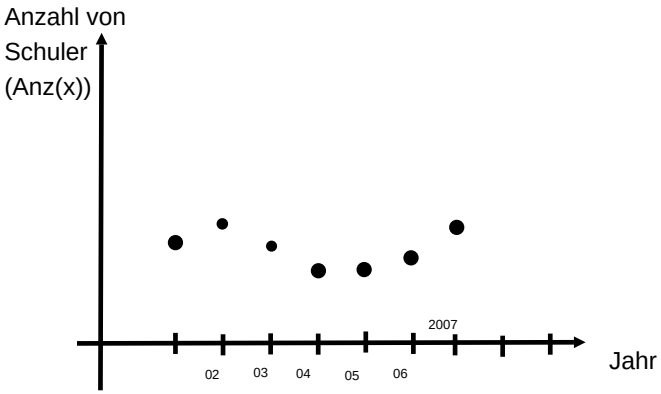
Was Passiert im Jahr 2007+1? 2007+2?

Zukunft vorhersagen.



Was Passiert im Jahr 2007+1? 2007+2?
Ein Vorschlag:

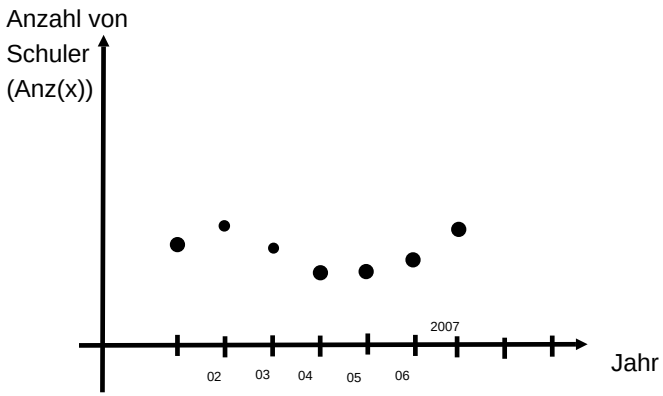
Zukunft vorhersagen.



Was Passiert im Jahr 2007+1? 2007+2?

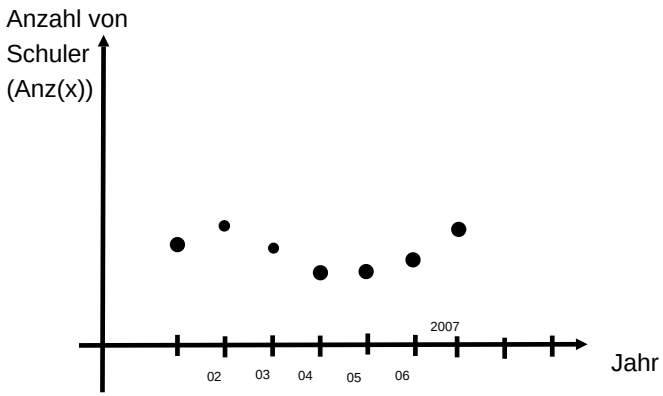
Ein Vorschlag: man nehme die Werte im Jahren $x_0 := 2007 - n$, ..., $x_{n-1} := 2006$, $x_n = 2007$

Zukunft vorhersagen.



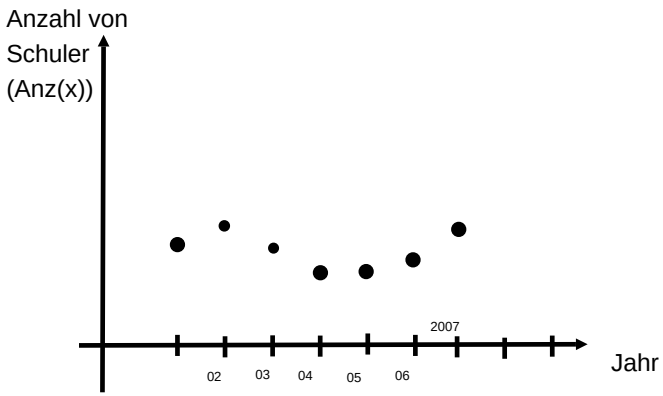
Was Passiert im Jahr 2007+1? 2007+2?
Ein Vorschlag: man nehme die Werte im Jahren $x_0 := 2007 - n$, ..., $x_{n-1} := 2006$, $x_n = 2007$ und betrachte das zugehörige Lagrange-Polynom P (mit $P(x_i) = Anz(x_i)$),

Zukunft vorhersagen.



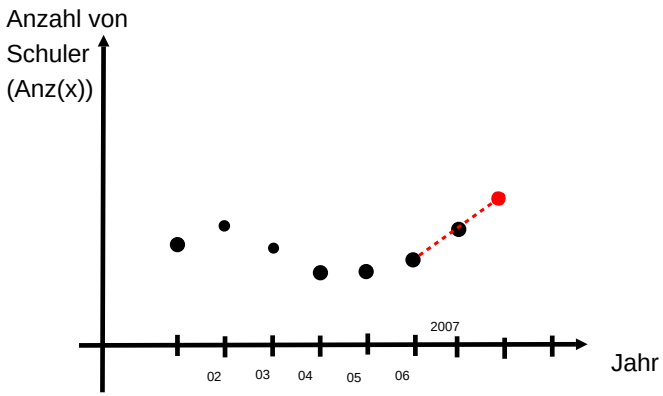
Was Passiert im Jahr 2007+1? 2007+2?
Ein Vorschlag: man nehme die Werte im Jahren $x_0 := 2007 - n$, ..., $x_{n-1} := 2006$, $x_n = 2007$ und betrachte das zugehörige Lagrange-Polynom P (mit $P(x_i) = Anz(x_i)$), und dann vorhersage $Anz(2008) = P(2008)$.

Zukunft vorhersagen.



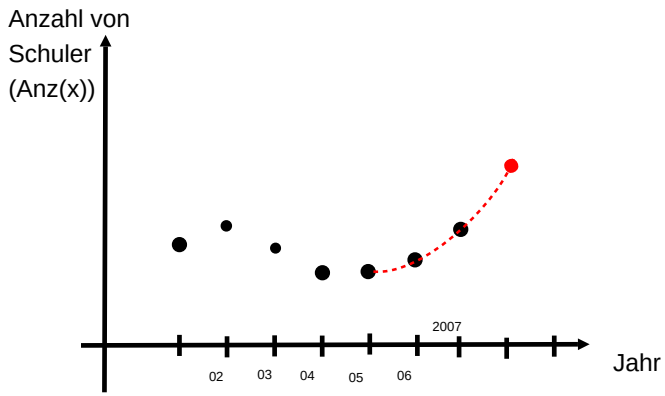
Was Passiert im Jahr 2007+1? 2007+2?
Ein Vorschlag: man nehme die Werte im Jahren $x_0 := 2007 - n$, ..., $x_{n-1} := 2006$, $x_n = 2007$ und betrachte das zugehörige Lagrange-Polynom P (mit $P(x_i) = Anz(x_i)$), und dann vorhersage $Anz(2008) = P(2008)$.
$n = 1$ lineare Interpolation

Zukunft vorhersagen.



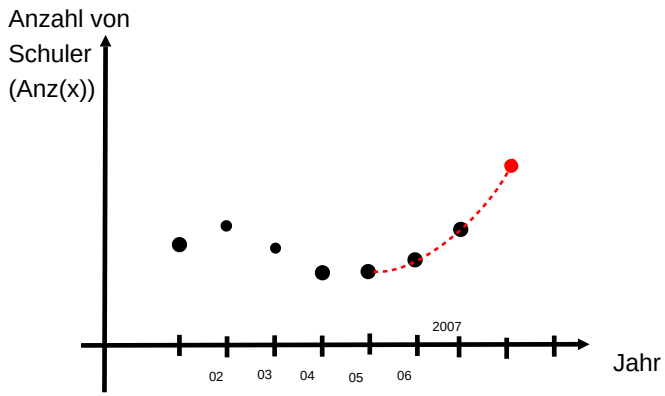
Was Passiert im Jahr 2007+1? 2007+2?
Ein Vorschlag: man nehme die Werte im Jahren $x_0 := 2007 - n$, ..., $x_{n-1} := 2006$, $x_n = 2007$ und betrachte das zugehörige Lagrange-Polynom P (mit $P(x_i) = Anz(x_i)$), und dann vorhersage $Anz(2008) = P(2008)$.
$n = 1$ lineare Interpolation

Zukunft vorhersagen.



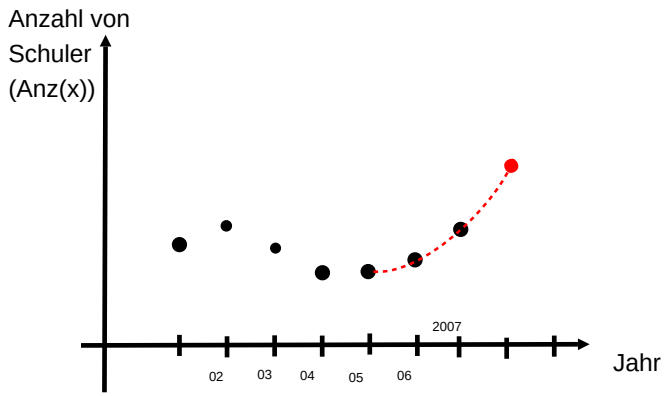
Was Passiert im Jahr 2007+1? 2007+2?
Ein Vorschlag: man nehme die Werte im Jahren $x_0 := 2007 - n$, ..., $x_{n-1} := 2006$, $x_n = 2007$ und betrachte das zugehörige Lagrange-Polynom P (mit $P(x_i) = Anz(x_i)$), und dann vorhersage $Anz(2008) = P(2008)$.
$n = 1$ lineare Interpolation
$n = 2$ Quadratische Interpolation

Zukunft vorhersagen.



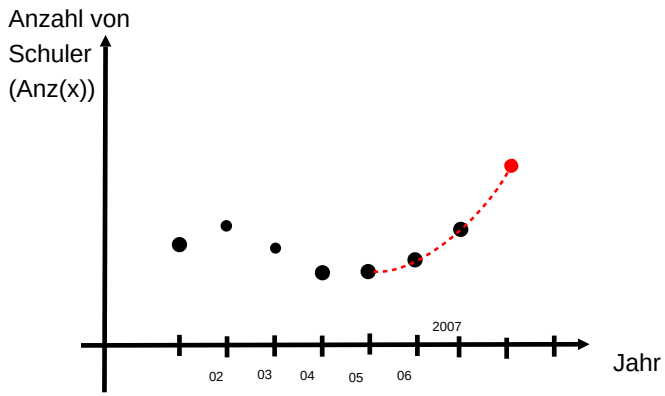
Was Passiert im Jahr 2007+1? 2007+2?
Ein Vorschlag: man nehme die Werte im Jahren $x_0 := 2007 - n$, ..., $x_{n-1} := 2006$, $x_n = 2007$ und betrachte das zugehörige Lagrange-Polynom P (mit $P(x_i) = Anz(x_i)$), und dann vorhersage $Anz(2008) = P(2008)$.
$n = 1$ lineare Interpolation
$n = 2$ Quadratische Interpolation
$n = 3$ Kubische Interpolation.

Zukunft vorhersagen.



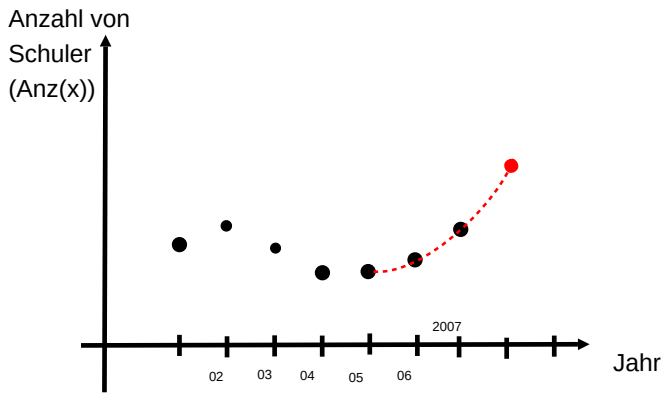
Was Passiert im Jahr 2007+1? 2007+2?
Ein Vorschlag: man nehme die Werte im Jahren $x_0 := 2007 - n$, ..., $x_{n-1} := 2006$, $x_n = 2007$ und betrachte das zugehörige Lagrange-Polynom P (mit $P(x_i) = Anz(x_i)$), und dann vorhersage $Anz(2008) = P(2008)$.
$n = 1$ lineare Interpolation
$n = 2$ Quadratische Interpolation
$n = 3$ Kubische Interpolation.
Welches n soll man nehmen?

Zukunft vorhersagen.



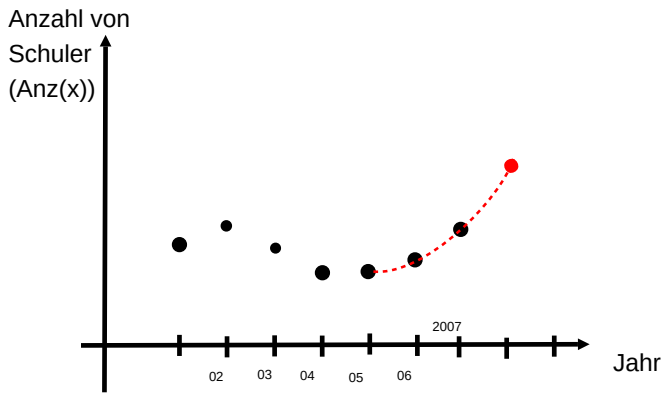
Was Passiert im Jahr 2007+1? 2007+2?
Ein Vorschlag: man nehme die Werte im Jahren $x_0 := 2007 - n$, ..., $x_{n-1} := 2006$, $x_n = 2007$ und betrachte das zugehörige Lagrange-Polynom P (mit $P(x_i) = Anz(x_i)$), und dann vorhersage $Anz(2008) = P(2008)$.
$n = 1$ lineare Interpolation
$n = 2$ Quadratische Interpolation
$n = 3$ Kubische Interpolation.
Welches n soll man nehmen? Das hängt von Aufgabe ab.

Zukunft vorhersagen.



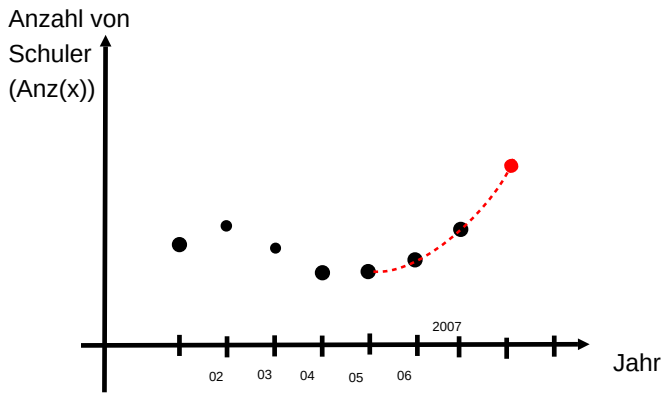
Was Passiert im Jahr 2007+1? 2007+2?
Ein Vorschlag: man nehme die Werte im Jahren $x_0 := 2007 - n$, ..., $x_{n-1} := 2006$, $x_n = 2007$ und betrachte das zugehörige Lagrange-Polynom P (mit $P(x_i) = Anz(x_i)$), und dann vorhersage $Anz(2008) = P(2008)$.
$n = 1$ lineare Interpolation
$n = 2$ Quadratische Interpolation
$n = 3$ Kubische Interpolation.
Welches n soll man nehmen? Das hängt von Aufgabe ab. Man kann verschiedene n nehmen,

Zukunft vorhersagen.



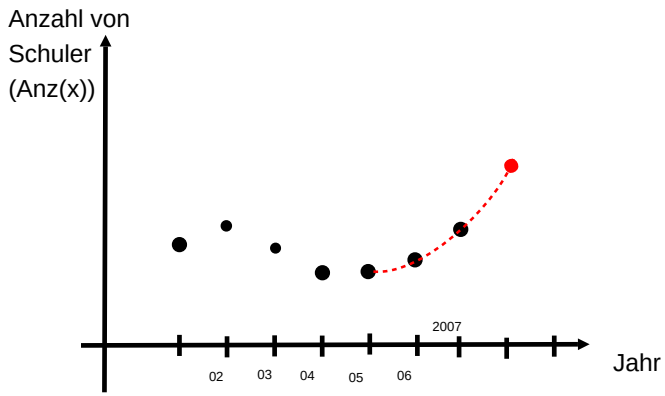
Was Passiert im Jahr 2007+1? 2007+2?
Ein Vorschlag: man nehme die Werte im Jahren $x_0 := 2007 - n$, ..., $x_{n-1} := 2006$, $x_n = 2007$ und betrachte das zugehörige Lagrange-Polynom P (mit $P(x_i) = Anz(x_i)$), und dann vorhersage $Anz(2008) = P(2008)$.
$n = 1$ lineare Interpolation
$n = 2$ Quadratische Interpolation
$n = 3$ Kubische Interpolation.
Welches n soll man nehmen? Das hängt von Aufgabe ab. Man kann verschiedene n nehmen, in der Vergangenheit testen,

Zukunft vorhersagen.



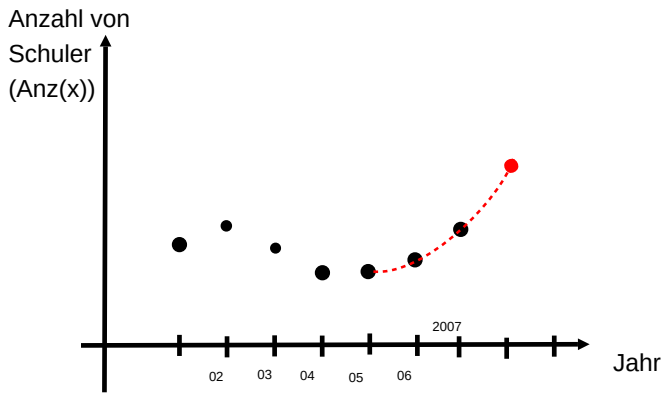
Was Passiert im Jahr 2007+1? 2007+2?
Ein Vorschlag: man nehme die Werte im Jahren $x_0 := 2007 - n$, ..., $x_{n-1} := 2006$, $x_n = 2007$ und betrachte das zugehörige Lagrange-Polynom P (mit $P(x_i) = Anz(x_i)$), und dann vorhersage $Anz(2008) = P(2008)$.
$n = 1$ lineare Interpolation
$n = 2$ Quadratische Interpolation
$n = 3$ Kubische Interpolation.
Welches n soll man nehmen? Das hängt von Aufgabe ab. Man kann verschiedene n nehmen, in der Vergangenheit testen, und das besseres übernehmen.

Zukunft vorhersagen.



Was Passiert im Jahr 2007+1? 2007+2?
Ein Vorschlag: man nehme die Werte im Jahren $x_0 := 2007 - n$, ..., $x_{n-1} := 2006$, $x_n = 2007$ und betrachte das zugehörige Lagrange-Polynom P (mit $P(x_i) = Anz(x_i)$), und dann vorhersage $Anz(2008) = P(2008)$.
$n = 1$ lineare Interpolation
$n = 2$ Quadratische Interpolation
$n = 3$ Kubische Interpolation.
Welches n soll man nehmen? Das hängt von Aufgabe ab. Man kann verschiedene n nehmen, in der Vergangenheit testen, und das besseres übernehmen. Auch wenn der Grad des Lagrange-Polynoms groß ist,

Zukunft vorhersagen.



Was Passiert im Jahr 2007+1? 2007+2?
Ein Vorschlag: man nehme die Werte im Jahren $x_0 := 2007 - n$, ..., $x_{n-1} := 2006$, $x_n = 2007$ und betrachte das zugehörige Lagrange-Polynom P (mit $P(x_i) = Anz(x_i)$), und dann vorhersage $Anz(2008) = P(2008)$.
$n = 1$ lineare Interpolation
$n = 2$ Quadratische Interpolation
$n = 3$ Kubische Interpolation.
Welches n soll man nehmen? Das hängt von Aufgabe ab. Man kann verschiedene n nehmen, in der Vergangenheit testen, und das besseres übernehmen. Auch wenn der Grad des Lagrange-Polynoms groß ist, haben die Werte $Anz(x_i)$ für die letzte x_i höhere Einfluss auf $P(2008)$ als die andere.

Codierungstheorie

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.

Frage

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem:

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n \alpha_{n-1} ... \alpha_0$

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n \alpha_{n-1} ... \alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n \alpha_{n-1} ... \alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp.

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n \alpha_{n-1} ... \alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 =$

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n \alpha_{n-1} ... \alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 =$

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n \alpha_{n-1} ... \alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n \alpha_{n-1} ... \alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n \alpha_{n-1} ... \alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung Binärsystem kommt mit zwei Zuständen aus,

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n \alpha_{n-1} ... \alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung Binärsystem kommt mit zwei Zuständen aus, nämlich '0' und '1'.

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n \alpha_{n-1} ... \alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung Binärsystem kommt mit zwei Zuständen aus, nämlich '0' und '1'. Im Computer oder bei Übertragung der Information durch z.B. Internet

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n \alpha_{n-1} ... \alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung Binärsystem kommt mit zwei Zuständen aus, nämlich '0' und '1'. Im Computer oder bei Übertragung der Information durch z.B. Internet werden diese beiden Zustände natürlich durch Strom und kein Strom repräsentiert.

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n \alpha_{n-1} ... \alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung Binärsystem kommt mit zwei Zuständen aus, nämlich '0' und '1'. Im Computer oder bei Übertragung der Information durch z.B. Internet werden diese beiden Zustände natürlich durch Strom und kein Strom repräsentiert. (oder durch Licht und kein Licht in einigen modernen Leitung,

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n \alpha_{n-1} ... \alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung Binärsystem kommt mit zwei Zuständen aus, nämlich '0' und '1'. Im Computer oder bei Übertragung der Information durch z.B. Internet werden diese beiden Zustände natürlich durch Strom und kein Strom repräsentiert. (oder durch Licht und kein Licht in einigen modernen Leitung, oder polarisiert/nicht polarisiert in DVD)

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n\alpha_{n-1}...\alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung Binärsystem kommt mit zwei Zuständen aus, nämlich '0' und '1'. Im Computer oder bei Übertragung der Information durch z.B. Internet werden diese beiden Zustände natürlich durch Strom und kein Strom repräsentiert. (oder durch Licht und kein Licht in einigen modernen Leitung, oder polarisiert/nicht polarisiert in DVD)
Wie werden die Daten übertragen bzw. gespeichert:

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n\alpha_{n-1}...\alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung Binärsystem kommt mit zwei Zuständen aus, nämlich '0' und '1'. Im Computer oder bei Übertragung der Information durch z.B. Internet werden diese beiden Zustände natürlich durch Strom und kein Strom repräsentiert. (oder durch Licht und kein Licht in einigen modernen Leitung, oder polarisiert/nicht polarisiert in DVD)
Wie werden die Daten übertragen bzw. gespeichert: verinfaches Bsp.

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n\alpha_{n-1}...\alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung Binärsystem kommt mit zwei Zuständen aus, nämlich '0' und '1'. Im Computer oder bei Übertragung der Information durch z.B. Internet werden diese beiden Zustände natürlich durch Strom und kein Strom repräsentiert. (oder durch Licht und kein Licht in einigen modernen Leitung, oder polarisiert/nicht polarisiert in DVD)

Wie werden die Daten übertragen bzw. gespeichert: verinfaches

Bsp. Man will ein Wort übertragen.

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n\alpha_{n-1}...\alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung Binärsystem kommt mit zwei Zuständen aus, nämlich '0' und '1'. Im Computer oder bei Übertragung der Information durch z.B. Internet werden diese beiden Zustände natürlich durch Strom und kein Strom repräsentiert. (oder durch Licht und kein Licht in einigen modernen Leitung, oder polarisiert/nicht polarisiert in DVD)
Wie werden die Daten übertragen bzw. gespeichert: verinfaches
Bsp. Man will ein Wort übertragen. Man codiert die Buchstaben eines Alphabet mit Hilfe von binäre Zahlen:

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n\alpha_{n-1}...\alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung Binärsystem kommt mit zwei Zuständen aus, nämlich '0' und '1'. Im Computer oder bei Übertragung der Information durch z.B. Internet werden diese beiden Zustände natürlich durch Strom und kein Strom repräsentiert. (oder durch Licht und kein Licht in einigen modernen Leitung, oder polarisiert/nicht polarisiert in DVD)
Wie werden die Daten übertragen bzw. gespeichert: verinfaches
Bsp. Man will ein Wort übertragen. Man codiert die Buchstaben eines Alphabet mit Hilfe von binäre Zahlen: $00001_2 = A$,

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n\alpha_{n-1}...\alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung Binärsystem kommt mit zwei Zuständen aus, nämlich '0' und '1'. Im Computer oder bei Übertragung der Information durch z.B. Internet werden diese beiden Zustände natürlich durch Strom und kein Strom repräsentiert. (oder durch Licht und kein Licht in einigen modernen Leitung, oder polarisiert/nicht polarisiert in DVD)
Wie werden die Daten übertragen bzw. gespeichert: verinfaches
Bsp. Man will ein Wort übertragen. Man codiert die Buchstaben eines Alphabet mit Hilfe von binäre Zahlen: $00001_2 = A$, $00010_2 = B$,

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n\alpha_{n-1}...\alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung Binärsystem kommt mit zwei Zuständen aus, nämlich '0' und '1'. Im Computer oder bei Übertragung der Information durch z.B. Internet werden diese beiden Zustände natürlich durch Strom und kein Strom repräsentiert. (oder durch Licht und kein Licht in einigen modernen Leitung, oder polarisiert/nicht polarisiert in DVD)
Wie werden die Daten übertragen bzw. gespeichert: verinfaches
Bsp. Man will ein Wort übertragen. Man codiert die Buchstaben eines Alphabet mit Hilfe von binäre Zahlen: $00001_2 = A$, $00010_2 = B$, $00011_2 = C$,

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n\alpha_{n-1}...\alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung Binärsystem kommt mit zwei Zuständen aus, nämlich '0' und '1'. Im Computer oder bei Übertragung der Information durch z.B. Internet werden diese beiden Zustände natürlich durch Strom und kein Strom repräsentiert. (oder durch Licht und kein Licht in einigen modernen Leitung, oder polarisiert/nicht polarisiert in DVD)
Wie werden die Daten übertragen bzw. gespeichert: verinfaches
Bsp. Man will ein Wort übertragen. Man codiert die Buchstaben eines Alphabet mit Hilfe von binäre Zahlen: $00001_2 = A$, $00010_2 = B$, $00011_2 = C$, usw. (Anzahl von fünfstelligen Zahlen ist $2^5 = 32$,

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n\alpha_{n-1}...\alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung Binärsystem kommt mit zwei Zuständen aus, nämlich '0' und '1'. Im Computer oder bei Übertragung der Information durch z.B. Internet werden diese beiden Zustände natürlich durch Strom und kein Strom repräsentiert. (oder durch Licht und kein Licht in einigen modernen Leitung, oder polarisiert/nicht polarisiert in DVD)
Wie werden die Daten übertragen bzw. gespeichert: verinfaches
Bsp. Man will ein Wort übertragen. Man codiert die Buchstaben eines Alphabet mit Hilfe von binäre Zahlen: $00001_2 = A$, $00010_2 = B$, $00011_2 = C$, usw. (Anzahl von fünfstelligen Zahlen ist $2^5 = 32$, reicht für Deutsche Buchstaben.

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n\alpha_{n-1}...\alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung Binärsystem kommt mit zwei Zuständen aus, nämlich '0' und '1'. Im Computer oder bei Übertragung der Information durch z.B. Internet werden diese beiden Zustände natürlich durch Strom und kein Strom repräsentiert. (oder durch Licht und kein Licht in einigen modernen Leitung, oder polarisiert/nicht polarisiert in DVD)
Wie werden die Daten übertragen bzw. gespeichert: verinfaches
Bsp. Man will ein Wort übertragen. Man codiert die Buchstaben eines Alphabet mit Hilfe von binäre Zahlen: $00001_2 = A$, $00010_2 = B$, $00011_2 = C$, usw. (Anzahl von fünfstelligen Zahlen ist $2^5 = 32$, reicht für Deutsche Buchstaben. In Wirklichkeit benutzt man das s.g. Bytes,

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n\alpha_{n-1}...\alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung Binärsystem kommt mit zwei Zuständen aus, nämlich '0' und '1'. Im Computer oder bei Übertragung der Information durch z.B. Internet werden diese beiden Zustände natürlich durch Strom und kein Strom repräsentiert. (oder durch Licht und kein Licht in einigen modernen Leitung, oder polarisiert/nicht polarisiert in DVD)
Wie werden die Daten übertragen bzw. gespeichert: verinfaches
Bsp. Man will ein Wort übertragen. Man codiert die Buchstaben eines Alphabet mit Hilfe von binäre Zahlen: $00001_2 = A$, $00010_2 = B$, $00011_2 = C$, usw. (Anzahl von fünfstelligen Zahlen ist $2^5 = 32$, reicht für Deutsche Buchstaben. In Wirklichkeit benutzt man das s.g. Bytes, d.h. binäre Zahlen aus 8 Ziffern.)

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n\alpha_{n-1}...\alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung Binärsystem kommt mit zwei Zuständen aus, nämlich '0' und '1'. Im Computer oder bei Übertragung der Information durch z.B. Internet werden diese beiden Zustände natürlich durch Strom und kein Strom repräsentiert. (oder durch Licht und kein Licht in einigen modernen Leitung, oder polarisiert/nicht polarisiert in DVD)
Wie werden die Daten übertragen bzw. gespeichert: verinfaches
Bsp. Man will ein Wort übertragen. Man codiert die Buchstaben eines Alphabet mit Hilfe von binäre Zahlen: $00001_2 = A$, $00010_2 = B$, $00011_2 = C$, usw. (Anzahl von fünfstelligen Zahlen ist $2^5 = 32$, reicht für Deutsche Buchstaben. In Wirklichkeit benutzt man das s.g. Bytes, d.h. binäre Zahlen aus 8 Ziffern.) Ersetzt man in einem beliebigen Wort die lateinische Buchstaben mit binären Zahlen,

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n\alpha_{n-1}...\alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung Binärsystem kommt mit zwei Zuständen aus, nämlich '0' und '1'. Im Computer oder bei Übertragung der Information durch z.B. Internet werden diese beiden Zustände natürlich durch Strom und kein Strom repräsentiert. (oder durch Licht und kein Licht in einigen modernen Leitung, oder polarisiert/nicht polarisiert in DVD)
Wie werden die Daten übertragen bzw. gespeichert: verinfaches
Bsp. Man will ein Wort übertragen. Man codiert die Buchstaben eines Alphabet mit Hilfe von binäre Zahlen: $00001_2 = A$, $00010_2 = B$, $00011_2 = C$, usw. (Anzahl von fünfstelligen Zahlen ist $2^5 = 32$, reicht für Deutsche Buchstaben. In Wirklichkeit benutzt man das s.g. Bytes, d.h. binäre Zahlen aus 8 Ziffern.) Ersetzt man in einem beliebigen Wort die lateinische Buchstaben mit binären Zahlen, und übertragt diese Zahlen.

Codierungstheorie

Ziel. Wir wollen Fehler, die bei der Übermittlung von Daten auftreten können, erkennen und korrigieren.
Frage Wie kann man die Daten Übermitteln?
Zählen im Binärsystem: Seien $\alpha_n, \alpha_{n-1}, ..., \alpha_0 \in \{0, 1\}$.
$\alpha_n\alpha_{n-1}...\alpha_0$ sei die Zahl $\alpha_n \cdot 2^n + \alpha_{n-1} \cdot 2^{n-1} \cdot ... + \alpha_0$.
Bsp. $110011_2 = 2^5 + 2^4 + 0 + 0 + 2 + 1 = 51_{10}$
Gründ für Einführung Binärsystem kommt mit zwei Zuständen aus, nämlich '0' und '1'. Im Computer oder bei Übertragung der Information durch z.B. Internet werden diese beiden Zustände natürlich durch Strom und kein Strom repräsentiert. (oder durch Licht und kein Licht in einigen modernen Leitung, oder polarisiert/nicht polarisiert in DVD)
Wie werden die Daten übertragen bzw. gespeichert: verinfaches
Bsp. Man will ein Wort übertragen. Man codiert die Buchstaben eines Alphabet mit Hilfe von binäre Zahlen: $00001_2 = A$, $00010_2 = B$, $00011_2 = C$, usw. (Anzahl von fünfstelligen Zahlen ist $2^5 = 32$, reicht für Deutsche Buchstaben. In Wirklichkeit benutzt man das s.g. Bytes, d.h. binäre Zahlen aus 8 Ziffern.) Ersetzt man in einem beliebigen Wort die lateinische Buchstaben mit binären Zahlen, und übertragt diese Zahlen.

Es treten leider die Fehlern bei Übertragung auf.

Es treten leider die Fehlern bei Übertragung auf.
Ziel:

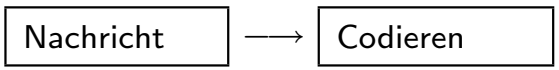
Es treten leider die Fehlern bei Übertragung auf.
Ziel: Einen Code zu finden,

Es treten leider die Fehlern bei Übertragung auf.
Ziel: Einen Code zu finden, das erlaubt, die Fehler zu entdecken und zu korriegieren.

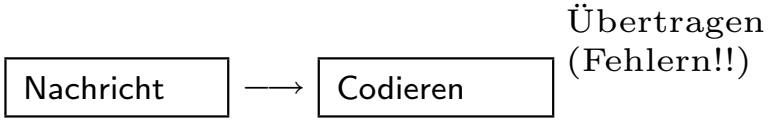
Es treten leider die Fehlern bei Übertragung auf.
Ziel: Einen Code zu finden, das erlaubt, die Fehler zu entdecken und zu korriegieren.

Nachricht

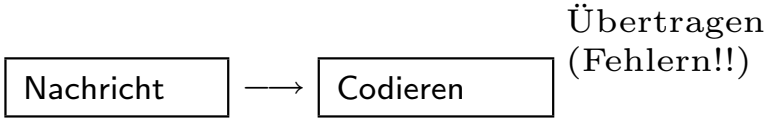
Es treten leider die Fehlern bei Übertragung auf.
Ziel: Einen Code zu finden, das erlaubt, die Fehler zu entdecken und zu korriegieren.



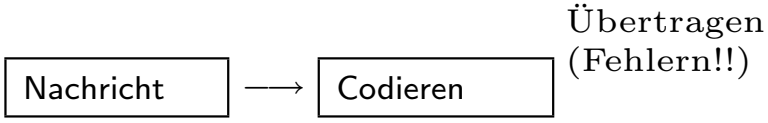
Es treten leider die Fehlern bei Übertragung auf.
Ziel: Einen Code zu finden, das erlaubt, die Fehler zu entdecken und zu korriegieren.



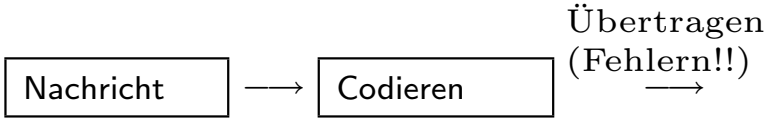
Es treten leider die Fehlern bei Übertragung auf.
Ziel: Einen Code zu finden, das erlaubt, die Fehler zu entdecken und zu korriegieren.



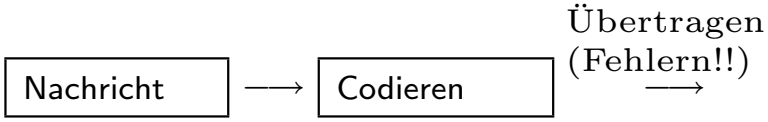
Es treten leider die Fehlern bei Übertragung auf.
Ziel: Einen Code zu finden, das erlaubt, die Fehler zu entdecken und zu korriegieren.



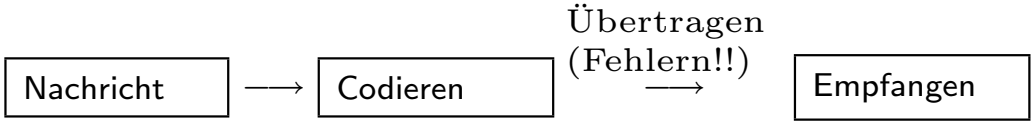
Es treten leider die Fehlern bei Übertragung auf.
Ziel: Einen Code zu finden, das erlaubt, die Fehler zu entdecken und zu korriegieren.



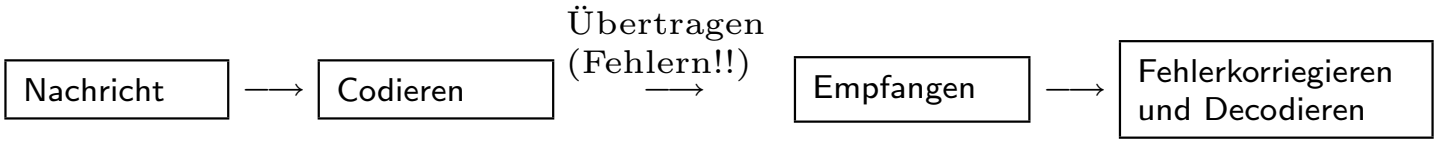
Es treten leider die Fehlern bei Übertragung auf.
Ziel: Einen Code zu finden, das erlaubt, die Fehler zu entdecken und zu korriegieren.



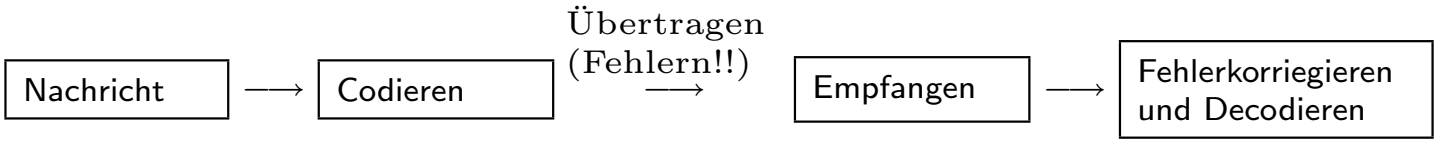
Es treten leider die Fehlern bei Übertragung auf.
Ziel: Einen Code zu finden, das erlaubt, die Fehler zu entdecken und zu korriegieren.



Es treten leider die Fehlern bei Übertragung auf.
Ziel: Einen Code zu finden, das erlaubt, die Fehler zu entdecken und zu korriegieren.

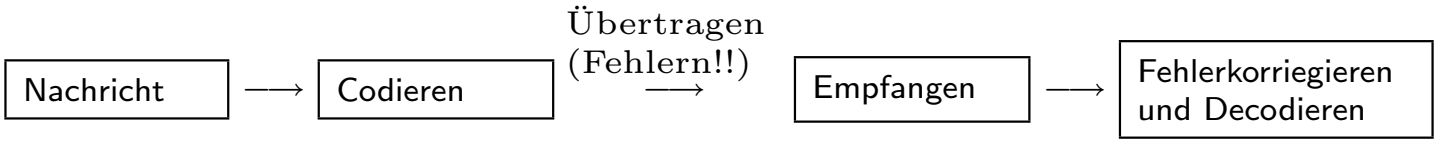


Es treten leider die Fehlern bei Übertragung auf.
Ziel: Einen Code zu finden, das erlaubt, die Fehler zu entdecken und zu korriegieren.



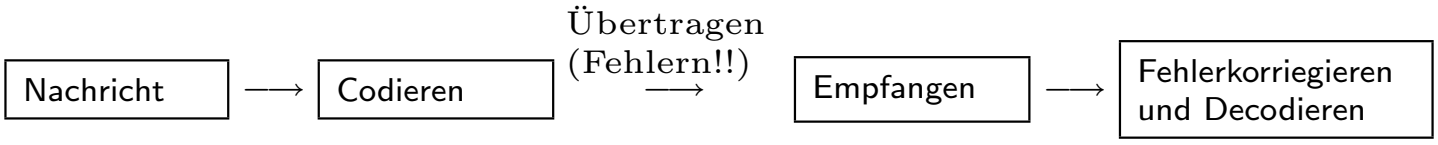
Ein naiver Weg.

Es treten leider die Fehlern bei Übertragung auf.
Ziel: Einen Code zu finden, das erlaubt, die Fehler zu entdecken und zu korriegieren.



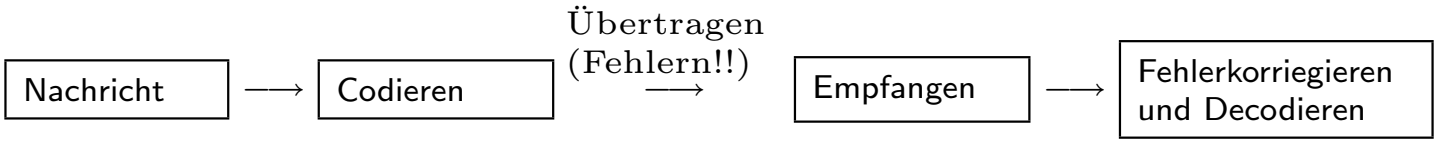
Ein naiver Weg. Wir werden jede Ziffer drei mal übertragen.

Es treten leider die Fehlern bei Übertragung auf.
Ziel: Einen Code zu finden, das erlaubt, die Fehler zu entdecken und zu korriegieren.



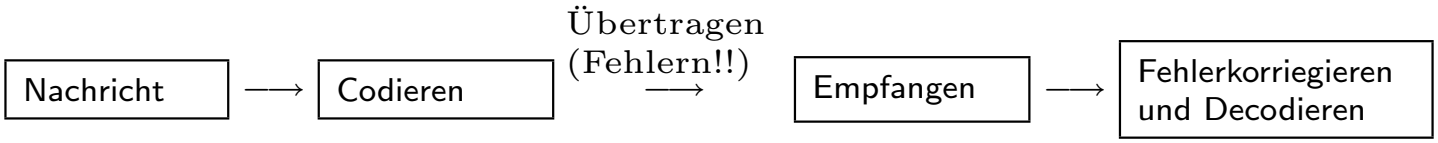
Ein naiver Weg. Wir werden jede Ziffer drei mal übertragen. Z.b. statt 10011 werden wir

Es treten leider die Fehlern bei Übertragung auf.
Ziel: Einen Code zu finden, das erlaubt, die Fehler zu entdecken und zu korriegieren.



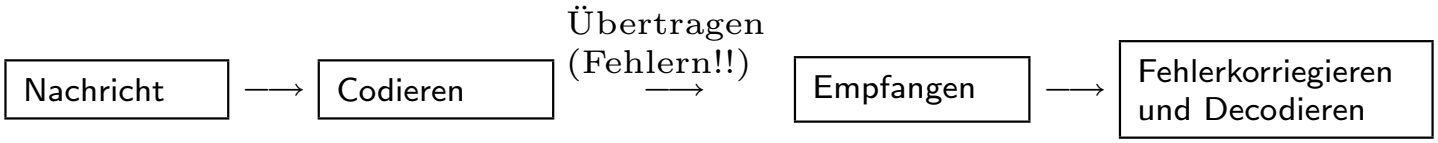
Ein naiver Weg. Wir werden jede Ziffer drei mal übertragen. Z.b. statt 10011 werden wir 111 000 000 111 111 senden.

Es treten leider die Fehlern bei Übertragung auf.
Ziel: Einen Code zu finden, das erlaubt, die Fehler zu entdecken und zu korriegieren.



Ein naiver Weg. Wir werden jede Ziffer drei mal übertragen. Z.b. statt 10011 werden wir 111 000 000 111 111 senden. Im Fall eines Fehlers werden wir „einfache Mehrheit“ benutzen.

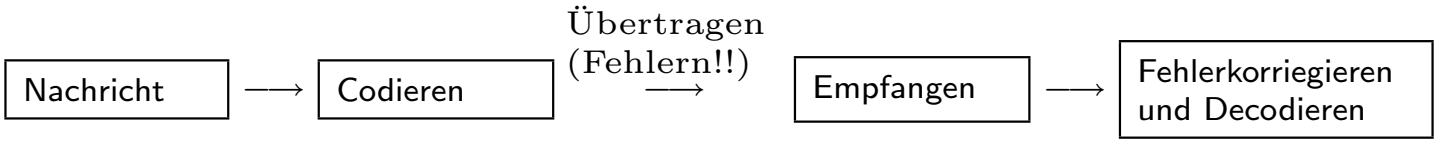
Es treten leider die Fehlern bei Übertragung auf.
Ziel: Einen Code zu finden, das erlaubt, die Fehler zu entdecken und zu korriegieren.



Ein naiver Weg. Wir werden jede Ziffer drei mal übertragen. Z.b. statt 10011 werden wir 111 000 000 111 111 senden. Im Fall eines Fehlers werden wir „einfache Mehrheit“ benutzen.

Lineare Algebra über Körper $\mathbb{Z}_2$ liefert ein effektives Methode, Fehler zu korriegieren.

Es treten leider die Fehlern bei Übertragung auf.
Ziel: Einen Code zu finden, das erlaubt, die Fehler zu entdecken und zu korriegieren.

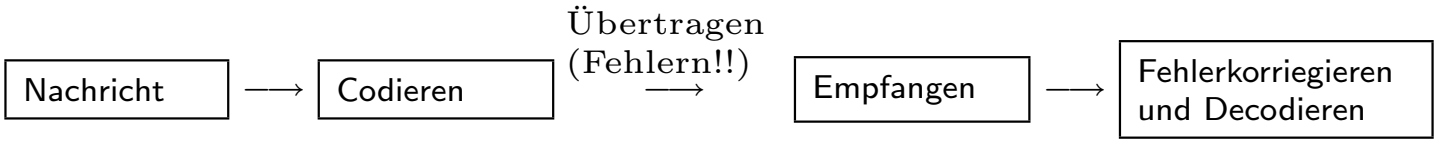


Ein naiver Weg. Wir werden jede Ziffer drei mal übertragen. Z.b. statt 10011 werden wir 111 000 000 111 111 senden. Im Fall eines Fehlers werden wir „einfache Mehrheit“ benutzen.

Lineare Algebra über Körper $\mathbb{Z}_2$ liefert ein effektives Methode, Fehler zu korriegieren.
Literatur.:

- MacWilliams, F.J., Sloane, N.J.A.: *The Theory of Error-Correcting Codes,* North- Holland, Amsterdam 1978.

Es treten leider die Fehlern bei Übertragung auf.
Ziel: Einen Code zu finden, das erlaubt, die Fehler zu entdecken und zu korriegieren.

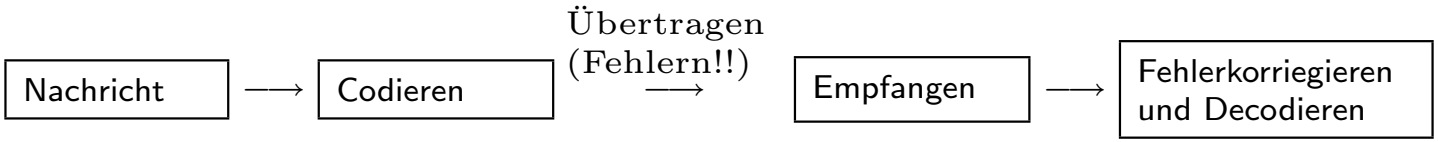


Ein naiver Weg. Wir werden jede Ziffer drei mal übertragen. Z.b. statt 10011 werden wir 111 000 000 111 111 senden. Im Fall eines Fehlers werden wir „einfache Mehrheit“ benutzen.

Lineare Algebra über Körper $\mathbb{Z}_2$ liefert ein effektives Methode, Fehler zu korriegieren.
Literatur.:

- MacWilliams, F.J., Sloane, N.J.A.: *The Theory of Error-Correcting Codes,* North- Holland, Amsterdam 1978.
- van Lint, J.H.:

Es treten leider die Fehlern bei Übertragung auf.
Ziel: Einen Code zu finden, das erlaubt, die Fehler zu entdecken und zu korriegieren.



Ein naiver Weg. Wir werden jede Ziffer drei mal übertragen. Z.b. statt 10011 werden wir 111 000 000 111 111 senden. Im Fall eines Fehlers werden wir „einfache Mehrheit“ benutzen.

Lineare Algebra über Körper $\mathbb{Z}_2$ liefert ein effektives Methode, Fehler zu korriegieren.
Literatur.:

- MacWilliams, F.J., Sloane, N.J.A.: *The Theory of Error-Correcting Codes,* North- Holland, Amsterdam 1978.
- van Lint, J.H.: *Introduction to Coding Theory (3. Aufl.)* , Springer 1999.

Betrachte $K = \mathbb{Z}_2$

Betrachte $K = \mathbb{Z}_2$ und $V = (\mathbb{Z}_2)^n$.

Def. 64

Betrachte $K = \mathbb{Z}_2$ und $V = (\mathbb{Z}_2)^n$.

Def. 64 *Die Abbildung* $d_H : V \times V \to \mathbb{N}$

Betrachte $K = \mathbb{Z}_2$ und $V = (\mathbb{Z}_2)^n$.

Def. 64 *Die Abbildung* $d_H : V \times V \to \mathbb{N}$ *definiert durch*

Betrachte $K = \mathbb{Z}_2$ und $V = (\mathbb{Z}_2)^n$.

Def. 64 *Die Abbildung* $d_H : V \times V \to \mathbb{N}$ *definiert durch*
$d_H(v, w) := \#\{j \in \{1, 2, ..., n\} \quad mit\ v_j \neq w_j\}$

Betrachte $K = \mathbb{Z}_2$ und $V = (\mathbb{Z}_2)^n$.

Def. 64 *Die Abbildung* $d_H : V \times V \to \mathbb{N}$ *definiert durch* $d_H(v, w) := \#\{j \in \{1, 2, ..., n\} \quad mit\ v_j \neq w_j\}$ *heißt Hemming-Abstand. (Das ist also die Anzahl der Komponenten, die bei einem Vektor 0, beim anderen aber 1 sind.)*

Lemma 45 *Der Hemming-Abstand hat für alle* $u, v, w \in V$ *die Eigenschaften:*

Betrachte $K = \mathbb{Z}_2$ und $V = (\mathbb{Z}_2)^n$.

Def. 64 *Die Abbildung* $d_H : V \times V \to \mathbb{N}$ *definiert durch* $d_H(v, w) := \#\{j \in \{1, 2, ..., n\} \quad mit\ v_j \neq w_j\}$ *heißt Hemming-Abstand. (Das ist also die Anzahl der Komponenten, die bei einem Vektor 0, beim anderen aber 1 sind.)*

Lemma 45 *Der Hemming-Abstand hat für alle* $u, v, w \in V$ *die Eigenschaften:*

(a) $d_H(v, w) \geq 0$ und $d_H(v, w) = 0 \iff v = w$

Betrachte $K = \mathbb{Z}_2$ und $V = (\mathbb{Z}_2)^n$.

Def. 64 *Die Abbildung* $d_H : V \times V \to \mathbb{N}$ *definiert durch* $d_H(v, w) := \#\{j \in \{1, 2, ..., n\} \quad mit\ v_j \neq w_j\}$ *heißt Hemming-Abstand. (Das ist also die Anzahl der Komponenten, die bei einem Vektor 0, beim anderen aber 1 sind.)*

Lemma 45 *Der Hemming-Abstand hat für alle* $u, v, w \in V$ *die Eigenschaften:*

(a) $d_H(v, w) \geq 0$ und $d_H(v, w) = 0 \iff v = w$

(b) $d_H(v, w) = d_H(w, v)$

Betrachte $K = \mathbb{Z}_2$ und $V = (\mathbb{Z}_2)^n$.

Def. 64 *Die Abbildung* $d_H : V \times V \to \mathbb{N}$ *definiert durch* $d_H(v, w) := \#\{j \in \{1, 2, ..., n\} \quad mit\ v_j \neq w_j\}$ *heißt Hemming-Abstand. (Das ist also die Anzahl der Komponenten, die bei einem Vektor 0, beim anderen aber 1 sind.)*

Lemma 45 *Der Hemming-Abstand hat für alle* $u, v, w \in V$ *die Eigenschaften:*

(a) $d_H(v, w) \geq 0$ und $d_H(v, w) = 0 \iff v = w$

(b) $d_H(v, w) = d_H(w, v)$

(c) $d_H(u, w) \leq d_H(u, v) + d_H(v, w)$

Betrachte $K = \mathbb{Z}_2$ und $V = (\mathbb{Z}_2)^n$.

Def. 64 *Die Abbildung* $d_H : V \times V \to \mathbb{N}$ *definiert durch* $d_H(v, w) := \#\{j \in \{1, 2, ..., n\} \quad mit\ v_j \neq w_j\}$ *heißt Hemming-Abstand. (Das ist also die Anzahl der Komponenten, die bei einem Vektor 0, beim anderen aber 1 sind.)*

Lemma 45 *Der Hemming-Abstand hat für alle* $u, v, w \in V$ *die Eigenschaften:*

(a) $d_H(v, w) \geq 0$ und $d_H(v, w) = 0 \iff v = w$

(b) $d_H(v, w) = d_H(w, v)$

(c) $d_H(u, w) \leq d_H(u, v) + d_H(v, w)$

(d) $d_H(v, w) = d_H(v + u, w + u)$

Betrachte $K = \mathbb{Z}_2$ und $V = (\mathbb{Z}_2)^n$.

Def. 64 *Die Abbildung* $d_H : V \times V \to \mathbb{N}$ *definiert durch* $d_H(v, w) := \#\{j \in \{1, 2, ..., n\} \quad mit\ v_j \neq w_j\}$ *heißt Hemming-Abstand. (Das ist also die Anzahl der Komponenten, die bei einem Vektor 0, beim anderen aber 1 sind.)*

Lemma 45 *Der Hemming-Abstand hat für alle* $u, v, w \in V$ *die Eigenschaften:*

(a) $d_H(v, w) \geq 0$ und $d_H(v, w) = 0 \iff v = w$

(b) $d_H(v, w) = d_H(w, v)$

(c) $d_H(u, w) \leq d_H(u, v) + d_H(v, w)$

(d) $d_H(v, w) = d_H(v + u, w + u)$

Beweis: (a), (b), (d) sind trivial.

Betrachte $K = \mathbb{Z}_2$ und $V = (\mathbb{Z}_2)^n$.

Def. 64 *Die Abbildung* $d_H : V \times V \to \mathbb{N}$ *definiert durch* $d_H(v, w) := \#\{j \in \{1, 2, ..., n\} \quad mit\ v_j \neq w_j\}$ *heißt Hemming-Abstand. (Das ist also die Anzahl der Komponenten, die bei einem Vektor 0, beim anderen aber 1 sind.)*

Lemma 45 *Der Hemming-Abstand hat für alle* $u, v, w \in V$ *die Eigenschaften:*

(a) $d_H(v, w) \geq 0$ und $d_H(v, w) = 0 \iff v = w$

(b) $d_H(v, w) = d_H(w, v)$

(c) $d_H(u, w) \leq d_H(u, v) + d_H(v, w)$

(d) $d_H(v, w) = d_H(v + u, w + u)$

Beweis: (a), (b), (d) sind trivial. (c) folgt aus der Beobachtung:

Betrachte $K = \mathbb{Z}_2$ und $V = (\mathbb{Z}_2)^n$.

Def. 64 *Die Abbildung* $d_H : V \times V \to \mathbb{N}$ *definiert durch* $d_H(v, w) := \#\{j \in \{1, 2, ..., n\} \quad mit\ v_j \neq w_j\}$ *heißt Hemming-Abstand. (Das ist also die Anzahl der Komponenten, die bei einem Vektor 0, beim anderen aber 1 sind.)*

Lemma 45 *Der Hemming-Abstand hat für alle* $u, v, w \in V$ *die Eigenschaften:*

(a) $d_H(v, w) \geq 0$ und $d_H(v, w) = 0 \iff v = w$

(b) $d_H(v, w) = d_H(w, v)$

(c) $d_H(u, w) \leq d_H(u, v) + d_H(v, w)$

(d) $d_H(v, w) = d_H(v + u, w + u)$

Beweis: (a), (b), (d) sind trivial. (c) folgt aus der Beobachtung: *Ist* $u_j \neq w_j$, *so ist wegen* $\mathbb{Z}_2 = \{0, 1\}$ *entweder* $u_j \neq v_j$ *oder* $w_j \neq v_j$.

Betrachte $K = \mathbb{Z}_2$ und $V = (\mathbb{Z}_2)^n$.

Def. 64 *Die Abbildung* $d_H : V \times V \to \mathbb{N}$ *definiert durch* $d_H(v, w) := \#\{j \in \{1, 2, ..., n\} \quad mit\ v_j \neq w_j\}$ *heißt Hemming-Abstand. (Das ist also die Anzahl der Komponenten, die bei einem Vektor 0, beim anderen aber 1 sind.)*

Lemma 45 *Der Hemming-Abstand hat für alle* $u, v, w \in V$ *die Eigenschaften:*

(a) $d_H(v, w) \geq 0$ und $d_H(v, w) = 0 \iff v = w$

(b) $d_H(v, w) = d_H(w, v)$

(c) $d_H(u, w) \leq d_H(u, v) + d_H(v, w)$

(d) $d_H(v, w) = d_H(v + u, w + u)$

Beweis: (a), (b), (d) sind trivial. (c) folgt aus der Beobachtung: *Ist* $u_j \neq w_j$, *so ist wegen* $\mathbb{Z}_2 = \{0, 1\}$ *entweder* $u_j \neq v_j$ *oder* $w_j \neq v_j$. □

Def. 65

Betrachte $K = \mathbb{Z}_2$ und $V = (\mathbb{Z}_2)^n$.

Def. 64 *Die Abbildung* $d_H : V \times V \to \mathbb{N}$ *definiert durch* $d_H(v, w) := \#\{j \in \{1, 2, ..., n\} \quad mit\ v_j \neq w_j\}$ *heißt Hemming-Abstand. (Das ist also die Anzahl der Komponenten, die bei einem Vektor 0, beim anderen aber 1 sind.)*

Lemma 45 *Der Hemming-Abstand hat für alle* $u, v, w \in V$ *die Eigenschaften:*

(a) $d_H(v, w) \geq 0$ und $d_H(v, w) = 0 \iff v = w$

(b) $d_H(v, w) = d_H(w, v)$

(c) $d_H(u, w) \leq d_H(u, v) + d_H(v, w)$

(d) $d_H(v, w) = d_H(v + u, w + u)$

Beweis: (a), (b), (d) sind trivial. (c) folgt aus der Beobachtung: *Ist* $u_j \neq w_j$, *so ist wegen* $\mathbb{Z}_2 = \{0, 1\}$ *entweder* $u_j \neq v_j$ *oder* $w_j \neq v_j$. □

Def. 65 *Sei* $\lambda \in \mathbb{N}$.

Betrachte $K = \mathbb{Z}_2$ und $V = (\mathbb{Z}_2)^n$.

Def. 64 *Die Abbildung* $d_H : V \times V \to \mathbb{N}$ *definiert durch* $d_H(v, w) := \#\{j \in \{1, 2, ..., n\} \quad mit\ v_j \neq w_j\}$ *heißt Hemming-Abstand. (Das ist also die Anzahl der Komponenten, die bei einem Vektor 0, beim anderen aber 1 sind.)*

Lemma 45 *Der Hemming-Abstand hat für alle* $u, v, w \in V$ *die Eigenschaften:*

(a) $d_H(v, w) \geq 0$ und $d_H(v, w) = 0 \iff v = w$

(b) $d_H(v, w) = d_H(w, v)$

(c) $d_H(u, w) \leq d_H(u, v) + d_H(v, w)$

(d) $d_H(v, w) = d_H(v + u, w + u)$

Beweis: (a), (b), (d) sind trivial. (c) folgt aus der Beobachtung: *Ist* $u_j \neq w_j$, *so ist wegen* $\mathbb{Z}_2 = \{0, 1\}$ *entweder* $u_j \neq v_j$ *oder* $w_j \neq v_j$. □

Def. 65 *Sei* $\lambda \in \mathbb{N}$. *Eine Teilmenge* $C \subseteq (\mathbb{Z}_2)^n$ *heißt* $\lambda-$*fehlerkorrigierender Code*,

Betrachte $K = \mathbb{Z}_2$ und $V = (\mathbb{Z}_2)^n$.

Def. 64 *Die Abbildung $d_H : V \times V \to \mathbb{N}$ definiert durch $d_H(v, w) := \#\{j \in \{1, 2, ..., n\} \quad mit\ v_j \neq w_j\}$ heißt Hemming-Abstand. (Das ist also die Anzahl der Komponenten, die bei einem Vektor 0, beim anderen aber 1 sind.)*

Lemma 45 *Der Hemming-Abstand hat für alle $u, v, w \in V$ die Eigenschaften:*

(a) $d_H(v, w) \geq 0$ und $d_H(v, w) = 0 \iff v = w$

(b) $d_H(v, w) = d_H(w, v)$

(c) $d_H(u, w) \leq d_H(u, v) + d_H(v, w)$

(d) $d_H(v, w) = d_H(v + u, w + u)$

Beweis: (a), (b), (d) sind trivial. (c) folgt aus der Beobachtung: *Ist $u_j \neq w_j$, so ist wegen $\mathbb{Z}_2 = \{0, 1\}$ entweder $u_j \neq v_j$ oder $w_j \neq v_j$.* □

Def. 65 *Sei $\lambda \in \mathbb{N}$. Eine Teilmenge $C \subseteq (\mathbb{Z}_2)^n$ heißt $\lambda-$fehlerkorrigierender Code, falls für $u, v \in C$ $u \neq v$ stets gilt: $d_H(u, v) \geq 2\lambda + 1$.*

Betrachte $K = \mathbb{Z}_2$ und $V = (\mathbb{Z}_2)^n$.

Def. 64 *Die Abbildung* $d_H : V \times V \to \mathbb{N}$ *definiert durch* $d_H(v, w) := \#\{j \in \{1, 2, ..., n\} \quad mit\ v_j \neq w_j\}$ *heißt Hemming-Abstand. (Das ist also die Anzahl der Komponenten, die bei einem Vektor 0, beim anderen aber 1 sind.)*

Lemma 45 *Der Hemming-Abstand hat für alle* $u, v, w \in V$ *die Eigenschaften:*

(a) $d_H(v, w) \geq 0$ und $d_H(v, w) = 0 \iff v = w$

(b) $d_H(v, w) = d_H(w, v)$

(c) $d_H(u, w) \leq d_H(u, v) + d_H(v, w)$

(d) $d_H(v, w) = d_H(v + u, w + u)$

Beweis: (a), (b), (d) sind trivial. (c) folgt aus der Beobachtung: *Ist* $u_j \neq w_j$, *so ist wegen* $\mathbb{Z}_2 = \{0, 1\}$ *entweder* $u_j \neq v_j$ *oder* $w_j \neq v_j$. □

Def. 65 *Sei* $\lambda \in \mathbb{N}$. *Eine Teilmenge* $C \subseteq (\mathbb{Z}_2)^n$ *heißt* $\lambda-$*fehlerkorrigierender Code, falls für* $u, v \in C$ $u \neq v$ *stets gilt:* $d_H(u, v) \geq 2\lambda + 1$.

Bsp.

Betrachte $K = \mathbb{Z}_2$ und $V = (\mathbb{Z}_2)^n$.

Def. 64 *Die Abbildung $d_H : V \times V \to \mathbb{N}$ definiert durch $d_H(v, w) := \#\{j \in \{1, 2, ..., n\} \quad mit\ v_j \neq w_j\}$ heißt Hemming-Abstand. (Das ist also die Anzahl der Komponenten, die bei einem Vektor 0, beim anderen aber 1 sind.)*

Lemma 45 *Der Hemming-Abstand hat für alle $u, v, w \in V$ die Eigenschaften:*

(a) $d_H(v, w) \geq 0$ und $d_H(v, w) = 0 \iff v = w$

(b) $d_H(v, w) = d_H(w, v)$

(c) $d_H(u, w) \leq d_H(u, v) + d_H(v, w)$

(d) $d_H(v, w) = d_H(v + u, w + u)$

Beweis: (a), (b), (d) sind trivial. (c) folgt aus der Beobachtung: *Ist $u_j \neq w_j$, so ist wegen $\mathbb{Z}_2 = \{0, 1\}$ entweder $u_j \neq v_j$ oder $w_j \neq v_j$.* □

Def. 65 *Sei $\lambda \in \mathbb{N}$. Eine Teilmenge $C \subseteq (\mathbb{Z}_2)^n$ heißt $\lambda-$fehlerkorrigierender Code, falls für $u, v \in C$ $u \neq v$ stets gilt: $d_H(u, v) \geq 2\lambda + 1$.*

Bsp. $n = 3$.

Betrachte $K = \mathbb{Z}_2$ und $V = (\mathbb{Z}_2)^n$.

Def. 64 *Die Abbildung* $d_H : V \times V \to \mathbb{N}$ *definiert durch* $d_H(v, w) := \#\{j \in \{1, 2, ..., n\} \quad mit\ v_j \neq w_j\}$ *heißt Hemming-Abstand. (Das ist also die Anzahl der Komponenten, die bei einem Vektor 0, beim anderen aber 1 sind.)*

Lemma 45 *Der Hemming-Abstand hat für alle* $u, v, w \in V$ *die Eigenschaften:*

(a) $d_H(v, w) \geq 0$ und $d_H(v, w) = 0 \iff v = w$

(b) $d_H(v, w) = d_H(w, v)$

(c) $d_H(u, w) \leq d_H(u, v) + d_H(v, w)$

(d) $d_H(v, w) = d_H(v + u, w + u)$

Beweis: (a), (b), (d) sind trivial. (c) folgt aus der Beobachtung: *Ist* $u_j \neq w_j$, *so ist wegen* $\mathbb{Z}_2 = \{0, 1\}$ *entweder* $u_j \neq v_j$ *oder* $w_j \neq v_j$. □

Def. 65 *Sei* $\lambda \in \mathbb{N}$. *Eine Teilmenge* $C \subseteq (\mathbb{Z}_2)^n$ *heißt* $\lambda-$*fehlerkorrigierender Code, falls für* $u, v \in C$ $u \neq v$ *stets gilt:* $d_H(u, v) \geq 2\lambda + 1$.

Bsp. $n = 3$. $C = \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right\}$

Betrachte $K = \mathbb{Z}_2$ und $V = (\mathbb{Z}_2)^n$.

Def. 64 *Die Abbildung* $d_H : V \times V \to \mathbb{N}$ *definiert durch* $d_H(v, w) := \#\{j \in \{1, 2, ..., n\} \quad mit\ v_j \neq w_j\}$ *heißt Hemming-Abstand. (Das ist also die Anzahl der Komponenten, die bei einem Vektor 0, beim anderen aber 1 sind.)*

Lemma 45 *Der Hemming-Abstand hat für alle* $u, v, w \in V$ *die Eigenschaften:*

(a) $d_H(v, w) \geq 0$ und $d_H(v, w) = 0 \iff v = w$

(b) $d_H(v, w) = d_H(w, v)$

(c) $d_H(u, w) \leq d_H(u, v) + d_H(v, w)$

(d) $d_H(v, w) = d_H(v + u, w + u)$

Beweis: (a), (b), (d) sind trivial. (c) folgt aus der Beobachtung: *Ist* $u_j \neq w_j$, *so ist wegen* $\mathbb{Z}_2 = \{0, 1\}$ *entweder* $u_j \neq v_j$ *oder* $w_j \neq v_j$. □

Def. 65 *Sei* $\lambda \in \mathbb{N}$. *Eine Teilmenge* $C \subseteq (\mathbb{Z}_2)^n$ *heißt* $\lambda-$*fehlerkorrigierender Code, falls für* $u, v \in C$ $u \neq v$ *stets gilt:* $d_H(u, v) \geq 2\lambda + 1$.

Bsp. $n = 3$. $C = \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right\}$ ist ein 1-fehlerkorrigierender Code.

Lemma 46

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code.*

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$*

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*

Beweis.

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*

Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$.

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*

Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*

Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt

$d_H(w_1, w_2)$

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*

Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt

$$d_H(w_1, w_2) \overset{\text{Lemma 13}}{\leq} d_H(w_1, v) + d_H(v, w_2)$$

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*

Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt

$$d_H(w_1, w_2) \overset{\text{Lemma 13}}{\leq} d_H(w_1, v) + d_H(v, w_2) \overset{\text{Voraussetzungen}}{\leq} 2\lambda.$$

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*

Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt $d_H(w_1, w_2) \overset{\text{Lemma 13}}{\leq} d_H(w_1, v) + d_H(v, w_2) \overset{\text{Voraussetzungen}}{\leq} 2\lambda$. Dann ist $w_1 = w_2$.

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*

Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt $d_H(w_1, w_2) \overset{\text{Lemma 13}}{\leq} d_H(w_1, v) + d_H(v, w_2) \overset{\text{Voraussetzungen}}{\leq} 2\lambda$. Dann ist $w_1 = w_2$. □

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*

Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt $d_H(w_1, w_2) \overset{\text{Lemma 13}}{\leq} d_H(w_1, v) + d_H(v, w_2) \overset{\text{Voraussetzungen}}{\leq} 2\lambda$. Dann ist $w_1 = w_2$. □

Hauptidea:

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*

Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt $d_H(w_1, w_2) \overset{\text{Lemma 13}}{\leq} d_H(w_1, v) + d_H(v, w_2) \overset{\text{Voraussetzungen}}{\leq} 2\lambda$. Dann ist $w_1 = w_2$. □

Hauptidea: Wähle λ-fehlerkorrigierenden Code C.

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*

Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt $d_H(w_1, w_2) \overset{\text{Lemma 13}}{\leq} d_H(w_1, v) + d_H(v, w_2) \overset{\text{Voraussetzungen}}{\leq} 2\lambda$. Dann ist $w_1 = w_2$. □

Hauptidea: Wähle λ-fehlerkorrigierenden Code C. Lasse nur Elemente aus C als Sendedaten zu.

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*

Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt $d_H(w_1, w_2) \overset{\text{Lemma 13}}{\leq} d_H(w_1, v) + d_H(v, w_2) \overset{\text{Voraussetzungen}}{\leq} 2\lambda$. Dann ist $w_1 = w_2$. □

Hauptidea: Wähle λ-fehlerkorrigierenden Code C. Lasse nur Elemente aus C als Sendedaten zu. Wenn bei der Übermittlung höchstens λ-viele Fehler auftreten,

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*

Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt $d_H(w_1, w_2) \overset{\text{Lemma 13}}{\leq} d_H(w_1, v) + d_H(v, w_2) \overset{\text{Voraussetzungen}}{\leq} 2\lambda$. Dann ist $w_1 = w_2$. □

Hauptidea: Wähle λ-fehlerkorrigierenden Code C. Lasse nur Elemente aus C als Sendedaten zu. Wenn bei der Übermittlung höchstens λ-viele Fehler auftreten, so können wir die gesendete Nachricht

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*

Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt $d_H(w_1, w_2) \overset{\text{Lemma 13}}{\leq} d_H(w_1, v) + d_H(v, w_2) \overset{\text{Voraussetzungen}}{\leq} 2\lambda$. Dann ist $w_1 = w_2$. □

Hauptidea: Wähle λ-fehlerkorrigierenden Code C. Lasse nur Elemente aus C als Sendedaten zu. Wenn bei der Übermittlung höchstens λ-viele Fehler auftreten, so können wir die gesendete Nachricht (ein Element aus C)

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*

Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt $d_H(w_1, w_2) \overset{\text{Lemma 13}}{\leq} d_H(w_1, v) + d_H(v, w_2) \overset{\text{Voraussetzungen}}{\leq} 2\lambda$. Dann ist $w_1 = w_2$. □

Hauptidea: Wähle λ-fehlerkorrigierenden Code C. Lasse nur Elemente aus C als Sendedaten zu. Wenn bei der Übermittlung höchstens λ-viele Fehler auftreten, so können wir die gesendete Nachricht (ein Element aus C) aus der empfangenen Nachricht

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*

Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt $d_H(w_1, w_2) \overset{\text{Lemma 13}}{\leq} d_H(w_1, v) + d_H(v, w_2) \overset{\text{Voraussetzungen}}{\leq} 2\lambda$. Dann ist $w_1 = w_2$. □

Hauptidea: Wähle λ-fehlerkorrigierenden Code C. Lasse nur Elemente aus C als Sendedaten zu. Wenn bei der Übermittlung höchstens λ-viele Fehler auftreten, so können wir die gesendete Nachricht (ein Element aus C) aus der empfangenen Nachricht (ein Element aus V)

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*

Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt $d_H(w_1, w_2) \overset{\text{Lemma 13}}{\leq} d_H(w_1, v) + d_H(v, w_2) \overset{\text{Voraussetzungen}}{\leq} 2\lambda$. Dann ist $w_1 = w_2$. □

Hauptidea: Wähle λ-fehlerkorrigierenden Code C. Lasse nur Elemente aus C als Sendedaten zu. Wenn bei der Übermittlung höchstens λ-viele Fehler auftreten, so können wir die gesendete Nachricht (ein Element aus C) aus der empfangenen Nachricht (ein Element aus V) rekonstruieren.

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*
Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt $d_H(w_1, w_2) \overset{\text{Lemma 13}}{\leq} d_H(w_1, v) + d_H(v, w_2) \overset{\text{Voraussetzungen}}{\leq} 2\lambda$. Dann ist $w_1 = w_2$. □

Hauptidea: Wähle λ-fehlerkorrigierenden Code C. Lasse nur Elemente aus C als Sendedaten zu. Wenn bei der Übermittlung höchstens λ-viele Fehler auftreten, so können wir die gesendete Nachricht (ein Element aus C) aus der empfangenen Nachricht (ein Element aus V) rekonstruieren.
Probleme.

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*
Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt $d_H(w_1, w_2) \overset{\text{Lemma 13}}{\leq} d_H(w_1, v) + d_H(v, w_2) \overset{\text{Voraussetzungen}}{\leq} 2\lambda$. Dann ist $w_1 = w_2$. □

Hauptidea: Wähle λ-fehlerkorrigierenden Code C. Lasse nur Elemente aus C als Sendedaten zu. Wenn bei der Übermittlung höchstens λ-viele Fehler auftreten, so können wir die gesendete Nachricht (ein Element aus C) aus der empfangenen Nachricht (ein Element aus V) rekonstruieren.
Probleme.

- Der Code C muss abgespeichert werden.

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*
Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt $d_H(w_1, w_2) \overset{\text{Lemma 13}}{\leq} d_H(w_1, v) + d_H(v, w_2) \overset{\text{Voraussetzungen}}{\leq} 2\lambda$. Dann ist $w_1 = w_2$. □

Hauptidea: Wähle λ-fehlerkorrigierenden Code C. Lasse nur Elemente aus C als Sendedaten zu. Wenn bei der Übermittlung höchstens λ-viele Fehler auftreten, so können wir die gesendete Nachricht (ein Element aus C) aus der empfangenen Nachricht (ein Element aus V) rekonstruieren.
Probleme.

- Der Code C muss abgespeichert werden. (Große Codes brauchen viel Speicherkapazität!)

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*
Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt $d_H(w_1, w_2) \overset{\text{Lemma 13}}{\leq} d_H(w_1, v) + d_H(v, w_2) \overset{\text{Voraussetzungen}}{\leq} 2\lambda$. Dann ist $w_1 = w_2$. □

Hauptidea: Wähle λ-fehlerkorrigierenden Code C. Lasse nur Elemente aus C als Sendedaten zu. Wenn bei der Übermittlung höchstens λ-viele Fehler auftreten, so können wir die gesendete Nachricht (ein Element aus C) aus der empfangenen Nachricht (ein Element aus V) rekonstruieren.
Probleme.

- Der Code C muss abgespeichert werden. (Große Codes brauchen viel Speicherkapazität!)
- Decodierung (d.h. ein $v \in C$

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*
Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt $d_H(w_1, w_2) \overset{\text{Lemma 13}}{\leq} d_H(w_1, v) + d_H(v, w_2) \overset{\text{Voraussetzungen}}{\leq} 2\lambda$. Dann ist $w_1 = w_2$. □

Hauptidea: Wähle λ-fehlerkorrigierenden Code C. Lasse nur Elemente aus C als Sendedaten zu. Wenn bei der Übermittlung höchstens λ-viele Fehler auftreten, so können wir die gesendete Nachricht (ein Element aus C) aus der empfangenen Nachricht (ein Element aus V) rekonstruieren.
Probleme.

- Der Code C muss abgespeichert werden. (Große Codes brauchen viel Speicherkapazität!)
- Decodierung (d.h. ein $v \in C$ wie im Lemma oben zu finden)

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*
Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt $d_H(w_1, w_2) \overset{\text{Lemma 13}}{\leq} d_H(w_1, v) + d_H(v, w_2) \overset{\text{Voraussetzungen}}{\leq} 2\lambda$. Dann ist $w_1 = w_2$. □

Hauptidea: Wähle λ-fehlerkorrigierenden Code C. Lasse nur Elemente aus C als Sendedaten zu. Wenn bei der Übermittlung höchstens λ-viele Fehler auftreten, so können wir die gesendete Nachricht (ein Element aus C) aus der empfangenen Nachricht (ein Element aus V) rekonstruieren.
Probleme.

- Der Code C muss abgespeichert werden. (Große Codes brauchen viel Speicherkapazität!)
- Decodierung (d.h. ein $v \in C$ wie im Lemma oben zu finden) erfordert viele Vergleiche der empfangenen Nachricht mit den Elementen aus C.

Lemma 46 *Sei $C \subseteq V$ ein λ-fehlerkorrigierender Code. Dann gibt es zu jedem $v \in V$ höchstens ein $w \in C$ mit $d_H(v, w) \leq \lambda$.*
Beweis. Seien $w_1, w_2 \in C$ mit $d_H(v, w_i) \leq \lambda$. Dann gilt $d_H(w_1, w_2) \overset{\text{Lemma 13}}{\leq} d_H(w_1, v) + d_H(v, w_2) \overset{\text{Voraussetzungen}}{\leq} 2\lambda$. Dann ist $w_1 = w_2$. □

Hauptidea: Wähle λ-fehlerkorrigierenden Code C. Lasse nur Elemente aus C als Sendedaten zu. Wenn bei der Übermittlung höchstens λ-viele Fehler auftreten, so können wir die gesendete Nachricht (ein Element aus C) aus der empfangenen Nachricht (ein Element aus V) rekonstruieren.
Probleme.

- Der Code C muss abgespeichert werden. (Große Codes brauchen viel Speicherkapazität!)
- Decodierung (d.h. ein $v \in C$ wie im Lemma oben zu finden) erfordert viele Vergleiche der empfangenen Nachricht mit den Elementen aus C.

Def. 66

Def. 66 *Ein λ-fehlerkorrigierender Code $C \in (\mathbb{Z}_2)^n$*

Def. 66 *Ein λ-fehlerkorrigierender Code $C \in (\mathbb{Z}_2)^n$ heißt linear, falls C zusätzlich ein Untervektorraum ist.*

Def. 66 *Ein λ-fehlerkorrigierender Code $C \in (\mathbb{Z}_2)^n$ heißt linear, falls C zusätzlich ein Untervektorraum ist.*
Um einen linearen Code festzulegen,

Def. 66 *Ein λ-fehlerkorrigierender Code $C \in (\mathbb{Z}_2)^n$ heißt linear, falls C zusätzlich ein Untervektorraum ist.*
Um einen linearen Code festzulegen, reicht es, eine Basis anzugeben.

Def. 66 *Ein λ-fehlerkorrigierender Code $C \in (\mathbb{Z}_2)^n$ heißt linear, falls C zusätzlich ein Untervektorraum ist.*
Um einen linearen Code festzulegen, reicht es, eine Basis anzugeben.
Wenn $dim(C) = k$, dann hat jede Basis k Elementen,

Def. 66 *Ein λ-fehlerkorrigierender Code $C \in (\mathbb{Z}_2)^n$ heißt linear, falls C zusätzlich ein Untervektorraum ist.*
Um einen linearen Code festzulegen, reicht es, eine Basis anzugeben.
Wenn $dim(C) = k$, dann hat jede Basis k Elementen, dagegen besteht C aus 2^k Elementen,

Def. 66 *Ein λ-fehlerkorrigierender Code $C \in (\mathbb{Z}_2)^n$ heißt linear, falls C zusätzlich ein Untervektorraum ist.*
Um einen linearen Code festzulegen, reicht es, eine Basis anzugeben.
Wenn $dim(C) = k$, dann hat jede Basis k Elementen, dagegen besteht C aus 2^k Elementen, was viel größer ist.

Def. 66 *Ein λ-fehlerkorrigierender Code $C \in (\mathbb{Z}_2)^n$ heißt linear, falls C zusätzlich ein Untervektorraum ist.*
Um einen linearen Code festzulegen, reicht es, eine Basis anzugeben. Wenn $dim(C) = k$, dann hat jede Basis k Elementen, dagegen besteht C aus 2^k Elementen, was viel größer ist. Also, benötige Speicherplatz ist klein.

Def. 66 *Ein λ-fehlerkorrigierender Code $C \in (\mathbb{Z}_2)^n$ heißt linear, falls C zusätzlich ein Untervektorraum ist.*
Um einen linearen Code festzulegen, reicht es, eine Basis anzugeben. Wenn $dim(C) = k$, dann hat jede Basis k Elementen, dagegen besteht C aus 2^k Elementen, was viel größer ist. Also, benötige Speicherplatz ist klein.
(Begründung: Ist $\{b_1, ..., b_k\}$ eine Basis von C,

Def. 66 *Ein λ-fehlerkorrigierender Code $C \in (\mathbb{Z}_2)^n$ heißt linear, falls C zusätzlich ein Untervektorraum ist.*
Um einen linearen Code festzulegen, reicht es, eine Basis anzugeben. Wenn $dim(C) = k$, dann hat jede Basis k Elementen, dagegen besteht C aus 2^k Elementen, was viel größer ist. Also, benötige Speicherplatz ist klein.
(Begründung: Ist $\{b_1, ..., b_k\}$ eine Basis von C, dann hat jedes Element aus C die Form $\lambda_1 b_1 + ... + \lambda_k b_k$,

Def. 66 *Ein λ-fehlerkorrigierender Code $C \in (\mathbb{Z}_2)^n$ heißt linear, falls C zusätzlich ein Untervektorraum ist.*
Um einen linearen Code festzulegen, reicht es, eine Basis anzugeben. Wenn $dim(C) = k$, dann hat jede Basis k Elementen, dagegen besteht C aus 2^k Elementen, was viel größer ist. Also, benötige Speicherplatz ist klein.
(Begründung: Ist $\{b_1, ..., b_k\}$ eine Basis von C, dann hat jedes Element aus C die Form $\lambda_1 b_1 + ... + \lambda_k b_k$, wobei $\lambda_i \in \mathbb{Z}_2$.

Def. 66 *Ein λ-fehlerkorrigierender Code $C \in (\mathbb{Z}_2)^n$ heißt linear, falls C zusätzlich ein Untervektorraum ist.*
Um einen linearen Code festzulegen, reicht es, eine Basis anzugeben. Wenn $dim(C) = k$, dann hat jede Basis k Elementen, dagegen besteht C aus 2^k Elementen, was viel größer ist. Also, benötige Speicherplatz ist klein.
(Begründung: Ist $\{b_1, ..., b_k\}$ eine Basis von C, dann hat jedes Element aus C die Form $\lambda_1 b_1 + ... + \lambda_k b_k$, wobei $\lambda_i \in \mathbb{Z}_2$. Also hat man für jedes λ_i genau zwei Auswahlmöglichkeiten

Def. 66 *Ein λ-fehlerkorrigierender Code $C \in (\mathbb{Z}_2)^n$ heißt linear, falls C zusätzlich ein Untervektorraum ist.*
Um einen linearen Code festzulegen, reicht es, eine Basis anzugeben. Wenn $dim(C) = k$, dann hat jede Basis k Elementen, dagegen besteht C aus 2^k Elementen, was viel größer ist. Also, benötige Speicherplatz ist klein.
(Begründung: Ist $\{b_1, ..., b_k\}$ eine Basis von C, dann hat jedes Element aus C die Form $\lambda_1 b_1 + ... + \lambda_k b_k$, wobei $\lambda_i \in \mathbb{Z}_2$. Also hat man für jedes λ_i genau zwei Auswahlmöglichkeiten und damit insgesamt 2^k mögliche Linearkombinationen.)

Codierung:

Codierung: Sei B die $(n \times k)$ matrix s.d. $Be_i = b_i$.

Codierung: Sei B die $(n \times k)$ matrix s.d. $Be_i = b_i$. Dann Codieren wir die binäre Zahl $\alpha_{k-1}..\alpha_0$

Codierung: Sei B die $(n \times k)$ matrix s.d. $Be_i = b_i$. Dann Codieren wir die binäre Zahl $\alpha_{k-1}..\alpha_0$ als $v = B\begin{pmatrix} \alpha_{k-1} \\ \vdots \\ \alpha_0 \end{pmatrix}$

Codierung: Sei B die $(n \times k)$ matrix s.d. $Be_i = b_i$. Dann Codieren wir die binäre Zahl $\alpha_{k-1}..\alpha_0$ als $v = B\begin{pmatrix} \alpha_{k-1} \\ \vdots \\ \alpha_0 \end{pmatrix} \in (\mathbb{Z}_2)^k$.

Codierung: Sei B die $(n \times k)$ matrix s.d. $Be_i = b_i$. Dann Codieren wir die binäre Zahl $\alpha_{k-1}..\alpha_0$ als $v = B\begin{pmatrix} \alpha_{k-1} \\ \vdots \\ \alpha_0 \end{pmatrix} \in (\mathbb{Z}_2)^k$.

Decodierung:

Codierung: Sei B die $(n \times k)$ matrix s.d. $Be_i = b_i$. Dann Codieren wir die binäre Zahl $\alpha_{k-1}..\alpha_0$ als $v = B\begin{pmatrix} \alpha_{k-1} \\ \vdots \\ \alpha_0 \end{pmatrix} \in (\mathbb{Z}_2)^k$.

Decodierung: Betrachte eine lineare Abbildung $d : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^k$,

Codierung: Sei B die $(n \times k)$ matrix s.d. $Be_i = b_i$. Dann Codieren wir die binäre Zahl $\alpha_{k-1}..\alpha_0$ als $v = B\begin{pmatrix} \alpha_{k-1} \\ \vdots \\ \alpha_0 \end{pmatrix} \in (\mathbb{Z}_2)^k$.

Decodierung: Betrachte eine lineare Abbildung $d : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^k$, die die Vektoren $b_1, ..., b_k$ in $e_1, ..., e_k$ überführt.

Codierung: Sei B die $(n \times k)$ matrix s.d. $Be_i = b_i$. Dann Codieren wir die binäre Zahl $\alpha_{k-1}..\alpha_0$ als $v = B\begin{pmatrix} \alpha_{k-1} \\ \vdots \\ \alpha_0 \end{pmatrix} \in (\mathbb{Z}_2)^k$.

Decodierung: Betrachte eine lineare Abbildung $d : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^k$, die die Vektoren $b_1, ..., b_k$ in $e_1, ..., e_k$ überführt.
Eine solche Abbildung existiert,

Codierung: Sei B die $(n \times k)$ matrix s.d. $Be_i = b_i$. Dann Codieren wir die binäre Zahl $\alpha_{k-1}..\alpha_0$ als $v = B\begin{pmatrix} \alpha_{k-1} \\ \vdots \\ \alpha_0 \end{pmatrix} \in (\mathbb{Z}_2)^k$.

Decodierung: Betrachte eine lineare Abbildung $d : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^k$, die die Vektoren $b_1, ..., b_k$ in $e_1, ..., e_k$ überführt.
Eine solche Abbildung existiert, denn: Ergänze eine Basis $\{b_1, ..., b_k\}$ von C zu einer Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$

Codierung: Sei B die $(n \times k)$ matrix s.d. $Be_i = b_i$. Dann Codieren wir die binäre Zahl $\alpha_{k-1}..\alpha_0$ als $v = B\begin{pmatrix} \alpha_{k-1} \\ \vdots \\ \alpha_0 \end{pmatrix} \in (\mathbb{Z}_2)^k$.

Decodierung: Betrachte eine lineare Abbildung $d : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^k$, die die Vektoren $b_1, ..., b_k$ in $e_1, ..., e_k$ überführt.
Eine solche Abbildung existiert, denn: Ergänze eine Basis $\{b_1, ..., b_k\}$ von C zu einer Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ von $(\mathbb{Z}_2)^n$. Dann leistet die

lineare Abbildung d definiert durch

Codierung: Sei B die $(n \times k)$ matrix s.d. $Be_i = b_i$. Dann Codieren wir die binäre Zahl $\alpha_{k-1}..\alpha_0$ als $v = B\begin{pmatrix} \alpha_{k-1} \\ \vdots \\ \alpha_0 \end{pmatrix} \in (\mathbb{Z}_2)^k$.

Decodierung: Betrachte eine lineare Abbildung $d : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^k$, die die Vektoren $b_1, ..., b_k$ in $e_1, ..., e_k$ überführt.
Eine solche Abbildung existiert, denn: Ergänze eine Basis $\{b_1, ..., b_k\}$ von C zu einer Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ von $(\mathbb{Z}_2)^n$. Dann leistet die lineare Abbildung d definiert durch $d(b_i) =$

Codierung: Sei B die $(n \times k)$ matrix s.d. $Be_i = b_i$. Dann Codieren wir die binäre Zahl $\alpha_{k-1}..\alpha_0$ als $v = B\begin{pmatrix} \alpha_{k-1} \\ \vdots \\ \alpha_0 \end{pmatrix} \in (\mathbb{Z}_2)^k$.

Decodierung: Betrachte eine lineare Abbildung $d : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^k$, die die Vektoren $b_1, ..., b_k$ in $e_1, ..., e_k$ überführt.
Eine solche Abbildung existiert, denn: Ergänze eine Basis $\{b_1, ..., b_k\}$ von C zu einer Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ von $(\mathbb{Z}_2)^n$. Dann leistet die lineare Abbildung d definiert durch $d(b_i) = \begin{cases} e_i & \text{falls } i \leq k \\ \end{cases}$

Codierung: Sei B die $(n \times k)$ matrix s.d. $Be_i = b_i$. Dann Codieren wir die binäre Zahl $\alpha_{k-1}..\alpha_0$ als $v = B\begin{pmatrix} \alpha_{k-1} \\ \vdots \\ \alpha_0 \end{pmatrix} \in (\mathbb{Z}_2)^k$.

Decodierung: Betrachte eine lineare Abbildung $d : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^k$, die die Vektoren $b_1, ..., b_k$ in $e_1, ..., e_k$ überführt.
Eine solche Abbildung existiert, denn: Ergänze eine Basis $\{b_1, ..., b_k\}$ von C zu einer Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ von $(\mathbb{Z}_2)^n$. Dann leistet die lineare Abbildung d definiert durch $d(b_i) = \begin{cases} e_i & \text{falls } i \leq k \\ \vec{0} & \text{falls } i > k \end{cases}$

Codierung: Sei B die $(n \times k)$ matrix s.d. $Be_i = b_i$. Dann Codieren wir die binäre Zahl $\alpha_{k-1}..\alpha_0$ als $v = B\begin{pmatrix} \alpha_{k-1} \\ \vdots \\ \alpha_0 \end{pmatrix} \in (\mathbb{Z}_2)^k$.

Decodierung: Betrachte eine lineare Abbildung $d : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^k$, die die Vektoren $b_1, ..., b_k$ in $e_1, ..., e_k$ überführt.
Eine solche Abbildung existiert, denn: Ergänze eine Basis $\{b_1, ..., b_k\}$ von C zu einer Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ von $(\mathbb{Z}_2)^n$. Dann leistet die lineare Abbildung d definiert durch $d(b_i) = \begin{cases} e_i & \text{falls } i \leq k \\ \vec{0} & \text{falls } i > k \end{cases}$ das Gewünschte.

Codierung: Sei B die $(n \times k)$ matrix s.d. $Be_i = b_i$. Dann Codieren wir die binäre Zahl $\alpha_{k-1}..\alpha_0$ als $v = B\begin{pmatrix} \alpha_{k-1} \\ \vdots \\ \alpha_0 \end{pmatrix} \in (\mathbb{Z}_2)^k$.

Decodierung: Betrachte eine lineare Abbildung $d : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^k$, die die Vektoren $b_1, ..., b_k$ in $e_1, ..., e_k$ überführt.
Eine solche Abbildung existiert, denn: Ergänze eine Basis $\{b_1, ..., b_k\}$ von C zu einer Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ von $(\mathbb{Z}_2)^n$. Dann leistet die lineare Abbildung d definiert durch $d(b_i) = \begin{cases} e_i & \text{falls } i \leq k \\ \vec{0} & \text{falls } i > k \end{cases}$ das Gewünschte. Sei $\tilde{B}$

Codierung: Sei B die $(n \times k)$ matrix s.d. $Be_i = b_i$. Dann Codieren wir die binäre Zahl $\alpha_{k-1}..\alpha_0$ als $v = B\begin{pmatrix} \alpha_{k-1} \\ \vdots \\ \alpha_0 \end{pmatrix} \in (\mathbb{Z}_2)^k$.

Decodierung: Betrachte eine lineare Abbildung $d : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^k$, die die Vektoren $b_1, ..., b_k$ in $e_1, ..., e_k$ überführt.
Eine solche Abbildung existiert, denn: Ergänze eine Basis $\{b_1, ..., b_k\}$ von C zu einer Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ von $(\mathbb{Z}_2)^n$. Dann leistet die lineare Abbildung d definiert durch $d(b_i) = \begin{cases} e_i & \text{falls } i \leq k \\ \vec{0} & \text{falls } i > k \end{cases}$ das Gewünschte. Sei $\tilde{B}$ die Matrix der Abbildung. Nach Konstruktion haben wir $\tilde{B}B = Id_k$,

Codierung: Sei B die $(n \times k)$ matrix s.d. $Be_i = b_i$. Dann Codieren wir die binäre Zahl $\alpha_{k-1}..\alpha_0$ als $v = B\begin{pmatrix} \alpha_{k-1} \\ \vdots \\ \alpha_0 \end{pmatrix} \in (\mathbb{Z}_2)^k$.

Decodierung: Betrachte eine lineare Abbildung $d : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^k$, die die Vektoren $b_1, ..., b_k$ in $e_1, ..., e_k$ überführt.
Eine solche Abbildung existiert, denn: Ergänze eine Basis $\{b_1, ..., b_k\}$ von C zu einer Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ von $(\mathbb{Z}_2)^n$. Dann leistet die lineare Abbildung d definiert durch $d(b_i) = \begin{cases} e_i & \text{falls } i \leq k \\ \vec{0} & \text{falls } i > k \end{cases}$ das Gewünschte. Sei $\tilde{B}$ die Matrix der Abbildung. Nach Konstruktion haben wir $\tilde{B}B = Id_k$, also, wenn wir eine codierte nachricht decodieren,

Codierung: Sei B die $(n \times k)$ matrix s.d. $Be_i = b_i$. Dann Codieren wir die binäre Zahl $\alpha_{k-1}..\alpha_0$ als $v = B\begin{pmatrix} \alpha_{k-1} \\ \vdots \\ \alpha_0 \end{pmatrix} \in (\mathbb{Z}_2)^k$.

Decodierung: Betrachte eine lineare Abbildung $d : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^k$, die die Vektoren $b_1, ..., b_k$ in $e_1, ..., e_k$ überführt.
Eine solche Abbildung existiert, denn: Ergänze eine Basis $\{b_1, ..., b_k\}$ von C zu einer Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ von $(\mathbb{Z}_2)^n$. Dann leistet die lineare Abbildung d definiert durch $d(b_i) = \begin{cases} e_i & \text{falls } i \leq k \\ \vec{0} & \text{falls } i > k \end{cases}$ das Gewünschte. Sei $\tilde{B}$ die Matrix der Abbildung. Nach Konstruktion haben wir $\tilde{B}B = Id_k$, also, wenn wir eine codierte nachricht decodieren, bekommen wir die ursprüngliche Nachricht zurück.

Codierung: Sei B die $(n \times k)$ matrix s.d. $Be_i = b_i$. Dann Codieren wir die binäre Zahl $\alpha_{k-1}..\alpha_0$ als $v = B\begin{pmatrix} \alpha_{k-1} \\ \vdots \\ \alpha_0 \end{pmatrix} \in (\mathbb{Z}_2)^k$.

Decodierung: Betrachte eine lineare Abbildung $d : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^k$, die die Vektoren $b_1, ..., b_k$ in $e_1, ..., e_k$ überführt.
Eine solche Abbildung existiert, denn: Ergänze eine Basis $\{b_1, ..., b_k\}$ von C zu einer Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ von $(\mathbb{Z}_2)^n$. Dann leistet die lineare Abbildung d definiert durch $d(b_i) = \begin{cases} e_i & \text{falls } i \leq k \\ \vec{0} & \text{falls } i > k \end{cases}$ das Gewünschte. Sei $\tilde{B}$ die Matrix der Abbildung. Nach Konstruktion haben wir $\tilde{B}B = Id_k$, also, wenn wir eine codierte nachricht decodieren, bekommen wir die ursprüngliche Nachricht zurück.

Fehlerkorrigierung:

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung
$f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung
$f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung
$f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n - k \times n)$ Matrix der Abbildung f

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung
$f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n - k \times n)$ Matrix der Abbildung f heißt die Kontrollmatrix des Codes.

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung
$f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n-k \times n)$ Matrix der Abbildung f heißt die Kontrollmatrix des Codes. Ist $f(v) = \vec{0}$, so $v \in C$.

Nenne $y \in (\mathbb{Z}_2)^{n-k}$ zulässig,

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n-k \times n)$ Matrix der Abbildung f heißt die Kontrollmatrix des Codes. Ist $f(v) = \vec{0}$, so $v \in C$.

Nenne $y \in (\mathbb{Z}_2)^{n-k}$ zulässig, falls ein x existiert mit $f(x) = y$ und $d_H(x, 0) \leq \lambda$.

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n-k \times n)$ Matrix der Abbildung f heißt die Kontrollmatrix des Codes. Ist $f(v) = \vec{0}$, so $v \in C$.

Nenne $y \in (\mathbb{Z}_2)^{n-k}$ zulässig, falls ein x existiert mit $f(x) = y$ und $d_H(x, 0) \leq \lambda$. Es kann höchstens ein solches x geben,

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n-k \times n)$ Matrix der Abbildung f heißt die Kontrollmatrix des Codes. Ist $f(v) = \vec{0}$, so $v \in C$.

Nenne $y \in (\mathbb{Z}_2)^{n-k}$ zulässig, falls ein x existiert mit $f(x) = y$ und $d_H(x, 0) \leq \lambda$. Es kann höchstens ein solches x geben, denn sind x, x' mit $d_H(x, 0) \leq \lambda$, $d_H(x', 0) \leq \lambda$,

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n-k \times n)$ Matrix der Abbildung f heißt die Kontrollmatrix des Codes. Ist $f(v) = \vec{0}$, so $v \in C$.

Nenne $y \in (\mathbb{Z}_2)^{n-k}$ zulässig, falls ein x existiert mit $f(x) = y$ und $d_H(x, 0) \leq \lambda$. Es kann höchstens ein solches x geben, denn sind x, x' mit $d_H(x, 0) \leq \lambda$, $d_H(x', 0) \leq \lambda$, dann ist $x - x' \in Kern_f = C$

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n-k \times n)$ Matrix der Abbildung f heißt die Kontrollmatrix des Codes. Ist $f(v) = \vec{0}$, so $v \in C$.

Nenne $y \in (\mathbb{Z}_2)^{n-k}$ zulässig, falls ein x existiert mit $f(x) = y$ und $d_H(x, 0) \leq \lambda$. Es kann höchstens ein solches x geben, denn sind x, x' mit $d_H(x, 0) \leq \lambda$, $d_H(x', 0) \leq \lambda$, dann ist $x - x' \in Kern_f = C$ und

$$d_H(x - x', \vec{0}) = d_H(x, x')$$

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n-k \times n)$ Matrix der Abbildung f heißt die Kontrollmatrix des Codes. Ist $f(v) = \vec{0}$, so $v \in C$.

Nenne $y \in (\mathbb{Z}_2)^{n-k}$ zulässig, falls ein x existiert mit $f(x) = y$ und $d_H(x, 0) \leq \lambda$. Es kann höchstens ein solches x geben, denn sind x, x' mit $d_H(x, 0) \leq \lambda$, $d_H(x', 0) \leq \lambda$, dann ist $x - x' \in Kern_f = C$ und

$$d_H(x - x', \vec{0}) = d_H(x, x') \overset{\text{Lemma 13}}{\leq}$$

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n-k \times n)$ Matrix der Abbildung f heißt die Kontrollmatrix des Codes. Ist $f(v) = \vec{0}$, so $v \in C$.

Nenne $y \in (\mathbb{Z}_2)^{n-k}$ zulässig, falls ein x existiert mit $f(x) = y$ und $d_H(x, 0) \leq \lambda$. Es kann höchstens ein solches x geben, denn sind x, x' mit $d_H(x, 0) \leq \lambda$, $d_H(x', 0) \leq \lambda$, dann ist $x - x' \in Kern_f = C$ und

$$d_H(x - x', \vec{0}) = d_H(x, x') \overset{\text{Lemma 13}}{\leq} d_H(x, 0) + d_H(0, x') \leq 2\lambda.$$

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n-k \times n)$ Matrix der Abbildung f heißt die Kontrollmatrix des Codes. Ist $f(v) = \vec{0}$, so $v \in C$.

Nenne $y \in (\mathbb{Z}_2)^{n-k}$ zulässig, falls ein x existiert mit $f(x) = y$ und $d_H(x, 0) \leq \lambda$. Es kann höchstens ein solches x geben, denn sind x, x' mit $d_H(x, 0) \leq \lambda$, $d_H(x', 0) \leq \lambda$, dann ist $x - x' \in Kern_f = C$ und $d_H(x - x', \vec{0}) = d_H(x, x') \overset{\text{Lemma 13}}{\leq} d_H(x, 0) + d_H(0, x') \leq 2\lambda$. Da C aber λ-fehlerkorrigierend ist,

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n-k \times n)$ Matrix der Abbildung f heißt die Kontrollmatrix des Codes. Ist $f(v) = \vec{0}$, so $v \in C$.

Nenne $y \in (\mathbb{Z}_2)^{n-k}$ zulässig, falls ein x existiert mit $f(x) = y$ und $d_H(x, 0) \leq \lambda$. Es kann höchstens ein solches x geben, denn sind x, x' mit $d_H(x, 0) \leq \lambda$, $d_H(x', 0) \leq \lambda$, dann ist $x - x' \in Kern_f = C$ und $d_H(x - x', \vec{0}) = d_H(x, x') \overset{\text{Lemma 13}}{\leq} d_H(x, 0) + d_H(0, x') \leq 2\lambda$. Da C aber λ-fehlerkorrigierend ist, folgt $x - x' = \vec{0}$,

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n-k \times n)$ Matrix der Abbildung f heißt die Kontrollmatrix des Codes. Ist $f(v) = \vec{0}$, so $v \in C$.

Nenne $y \in (\mathbb{Z}_2)^{n-k}$ zulässig, falls ein x existiert mit $f(x) = y$ und $d_H(x, 0) \leq \lambda$. Es kann höchstens ein solches x geben, denn sind x, x' mit $d_H(x, 0) \leq \lambda$, $d_H(x', 0) \leq \lambda$, dann ist $x - x' \in Kern_f = C$ und $d_H(x - x', \vec{0}) = d_H(x, x') \overset{\text{Lemma 13}}{\leq} d_H(x, 0) + d_H(0, x') \leq 2\lambda$. Da C aber λ-fehlerkorrigierend ist, folgt $x - x' = \vec{0}$, d.h. $x = x'$.

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n-k \times n)$ Matrix der Abbildung f heißt die Kontrollmatrix des Codes. Ist $f(v) = \vec{0}$, so $v \in C$.

Nenne $y \in (\mathbb{Z}_2)^{n-k}$ zulässig, falls ein x existiert mit $f(x) = y$ und $d_H(x, 0) \leq \lambda$. Es kann höchstens ein solches x geben, denn sind x, x' mit $d_H(x, 0) \leq \lambda$, $d_H(x', 0) \leq \lambda$, dann ist $x - x' \in Kern_f = C$ und $d_H(x - x', \vec{0}) = d_H(x, x') \overset{\text{Lemma 13}}{\leq} d_H(x, 0) + d_H(0, x') \leq 2\lambda$. Da C aber λ-fehlerkorrigierend ist, folgt $x - x' = \vec{0}$, d.h. $x = x'$.

Der Empfanger speichert in einer Liste alle zulässigen $y \in (\mathbb{Z}_2)^{n-k}$

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n-k \times n)$ Matrix der Abbildung f heißt die Kontrollmatrix des Codes. Ist $f(v) = \vec{0}$, so $v \in C$.

Nenne $y \in (\mathbb{Z}_2)^{n-k}$ zulässig, falls ein x existiert mit $f(x) = y$ und $d_H(x, 0) \leq \lambda$. Es kann höchstens ein solches x geben, denn sind x, x' mit $d_H(x, 0) \leq \lambda$, $d_H(x', 0) \leq \lambda$, dann ist $x - x' \in Kern_f = C$ und $d_H(x - x', \vec{0}) = d_H(x, x') \overset{\text{Lemma 13}}{\leq} d_H(x, 0) + d_H(0, x') \leq 2\lambda$. Da C aber λ-fehlerkorrigierend ist, folgt $x - x' = \vec{0}$, d.h. $x = x'$.

Der Empfanger speichert in einer Liste alle zulässigen $y \in (\mathbb{Z}_2)^{n-k}$ zusammen mit den zugehörigen x

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n-k \times n)$ Matrix der Abbildung f heißt die Kontrollmatrix des Codes. Ist $f(v) = \vec{0}$, so $v \in C$.

Nenne $y \in (\mathbb{Z}_2)^{n-k}$ zulässig, falls ein x existiert mit $f(x) = y$ und $d_H(x, 0) \leq \lambda$. Es kann höchstens ein solches x geben, denn sind x, x' mit $d_H(x, 0) \leq \lambda$, $d_H(x', 0) \leq \lambda$, dann ist $x - x' \in Kern_f = C$ und $d_H(x - x', \vec{0}) = d_H(x, x') \overset{\text{Lemma 13}}{\leq} d_H(x, 0) + d_H(0, x') \leq 2\lambda$. Da C aber λ-fehlerkorrigierend ist, folgt $x - x' = \vec{0}$, d.h. $x = x'$.

Der Empfanger speichert in einer Liste alle zulässigen $y \in (\mathbb{Z}_2)^{n-k}$ zusammen mit den zugehörigen x s.d. $f(x) = y$, für die $d_H(x, 0) \leq \lambda$ gilt.

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n-k \times n)$ Matrix der Abbildung f heißt die Kontrollmatrix des Codes. Ist $f(v) = \vec{0}$, so $v \in C$.

Nenne $y \in (\mathbb{Z}_2)^{n-k}$ zulässig, falls ein x existiert mit $f(x) = y$ und $d_H(x, 0) \leq \lambda$. Es kann höchstens ein solches x geben, denn sind x, x' mit $d_H(x, 0) \leq \lambda$, $d_H(x', 0) \leq \lambda$, dann ist $x - x' \in Kern_f = C$ und $d_H(x - x', \vec{0}) = d_H(x, x') \overset{\text{Lemma 13}}{\leq} d_H(x, 0) + d_H(0, x') \leq 2\lambda$. Da C aber λ-fehlerkorrigierend ist, folgt $x - x' = \vec{0}$, d.h. $x = x'$.

Der Empfanger speichert in einer Liste alle zulässigen $y \in (\mathbb{Z}_2)^{n-k}$ zusammen mit den zugehörigen x s.d. $f(x) = y$, für die $d_H(x, 0) \leq \lambda$ gilt.

Die Liste ist nicht gross.

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n-k \times n)$ Matrix der Abbildung f heißt die Kontrollmatrix des Codes. Ist $f(v) = \vec{0}$, so $v \in C$.

Nenne $y \in (\mathbb{Z}_2)^{n-k}$ zulässig, falls ein x existiert mit $f(x) = y$ und $d_H(x, 0) \leq \lambda$. Es kann höchstens ein solches x geben, denn sind x, x' mit $d_H(x, 0) \leq \lambda$, $d_H(x', 0) \leq \lambda$, dann ist $x - x' \in Kern_f = C$ und $d_H(x - x', \vec{0}) = d_H(x, x') \overset{\text{Lemma 13}}{\leq} d_H(x, 0) + d_H(0, x') \leq 2\lambda$. Da C aber λ-fehlerkorrigierend ist, folgt $x - x' = \vec{0}$, d.h. $x = x'$.

Der Empfanger speichert in einer Liste alle zulässigen $y \in (\mathbb{Z}_2)^{n-k}$ zusammen mit den zugehörigen x s.d. $f(x) = y$, für die $d_H(x, 0) \leq \lambda$ gilt.

Die Liste ist nicht gross.

Eine empfangene Nachricht $v \in (\mathbb{Z}_2)^n$,

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n-k \times n)$ Matrix der Abbildung f heißt die Kontrollmatrix des Codes. Ist $f(v) = \vec{0}$, so $v \in C$.

Nenne $y \in (\mathbb{Z}_2)^{n-k}$ zulässig, falls ein x existiert mit $f(x) = y$ und $d_H(x, 0) \leq \lambda$. Es kann höchstens ein solches x geben, denn sind x, x' mit $d_H(x, 0) \leq \lambda$, $d_H(x', 0) \leq \lambda$, dann ist $x - x' \in Kern_f = C$ und $d_H(x - x', \vec{0}) = d_H(x, x') \overset{\text{Lemma 13}}{\leq} d_H(x, 0) + d_H(0, x') \leq 2\lambda$. Da C aber λ-fehlerkorrigierend ist, folgt $x - x' = \vec{0}$, d.h. $x = x'$.

Der Empfanger speichert in einer Liste alle zulässigen $y \in (\mathbb{Z}_2)^{n-k}$ zusammen mit den zugehörigen x s.d. $f(x) = y$, für die $d_H(x, 0) \leq \lambda$ gilt.

Die Liste ist nicht gross.

Eine empfangene Nachricht $v \in (\mathbb{Z}_2)^n$, die möglicherweise verfälscht worden ist,

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n-k \times n)$ Matrix der Abbildung f heißt die Kontrollmatrix des Codes. Ist $f(v) = \vec{0}$, so $v \in C$.

Nenne $y \in (\mathbb{Z}_2)^{n-k}$ zulässig, falls ein x existiert mit $f(x) = y$ und $d_H(x, 0) \leq \lambda$. Es kann höchstens ein solches x geben, denn sind x, x' mit $d_H(x, 0) \leq \lambda$, $d_H(x', 0) \leq \lambda$, dann ist $x - x' \in Kern_f = C$ und $d_H(x - x', \vec{0}) = d_H(x, x') \overset{\text{Lemma 13}}{\leq} d_H(x, 0) + d_H(0, x') \leq 2\lambda$. Da C aber λ-fehlerkorrigierend ist, folgt $x - x' = \vec{0}$, d.h. $x = x'$.

Der Empfanger speichert in einer Liste alle zulässigen $y \in (\mathbb{Z}_2)^{n-k}$ zusammen mit den zugehörigen x s.d. $f(x) = y$, für die $d_H(x, 0) \leq \lambda$ gilt.

Die Liste ist nicht gross.

Eine empfangene Nachricht $v \in (\mathbb{Z}_2)^n$, die möglicherweise verfälscht worden ist, kann nun wie folgt decodiert werden.

Fehlerkorrigierung: Wähle eine lineare surjektive Abbildung $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^{n-k}$ mit $Kern_f = C$.
Z.B. betrachte die Basis $\{b_1, ..., b_k, b_{k+1}, ..., b_n\}$ wie oben, und definiere

$$f(b_i) = \begin{cases} \vec{0} & \text{falls } i \leq k \\ e_{i-k} & \text{falls } i > k \end{cases}$$

Die $(n-k \times n)$ Matrix der Abbildung f heißt die Kontrollmatrix des Codes. Ist $f(v) = \vec{0}$, so $v \in C$.

Nenne $y \in (\mathbb{Z}_2)^{n-k}$ zulässig, falls ein x existiert mit $f(x) = y$ und $d_H(x, 0) \leq \lambda$. Es kann höchstens ein solches x geben, denn sind x, x' mit $d_H(x, 0) \leq \lambda$, $d_H(x', 0) \leq \lambda$, dann ist $x - x' \in Kern_f = C$ und $d_H(x - x', \vec{0}) = d_H(x, x') \overset{\text{Lemma 13}}{\leq} d_H(x, 0) + d_H(0, x') \leq 2\lambda$. Da C aber λ-fehlerkorrigierend ist, folgt $x - x' = \vec{0}$, d.h. $x = x'$.

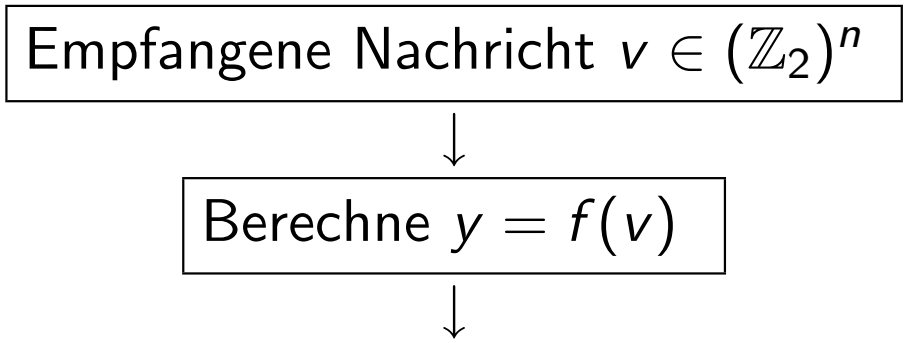
Der Empfanger speichert in einer Liste alle zulässigen $y \in (\mathbb{Z}_2)^{n-k}$ zusammen mit den zugehörigen x s.d. $f(x) = y$, für die $d_H(x, 0) \leq \lambda$ gilt.

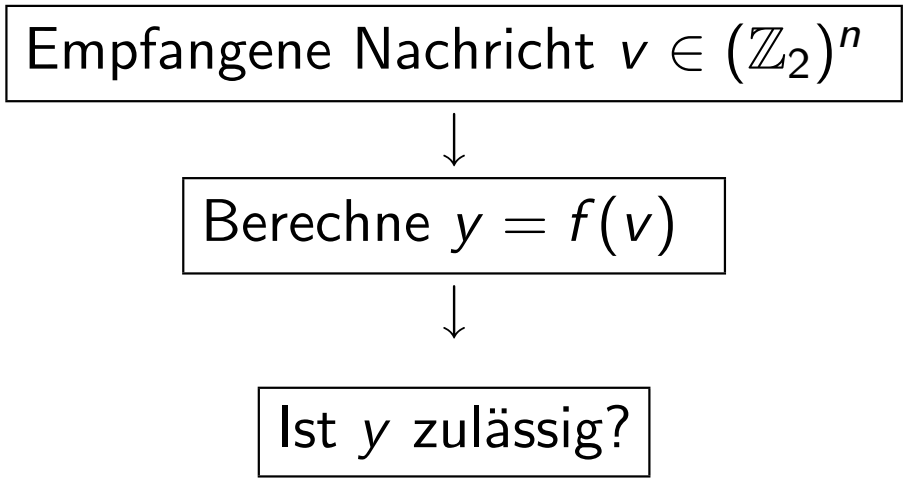
Die Liste ist nicht gross.

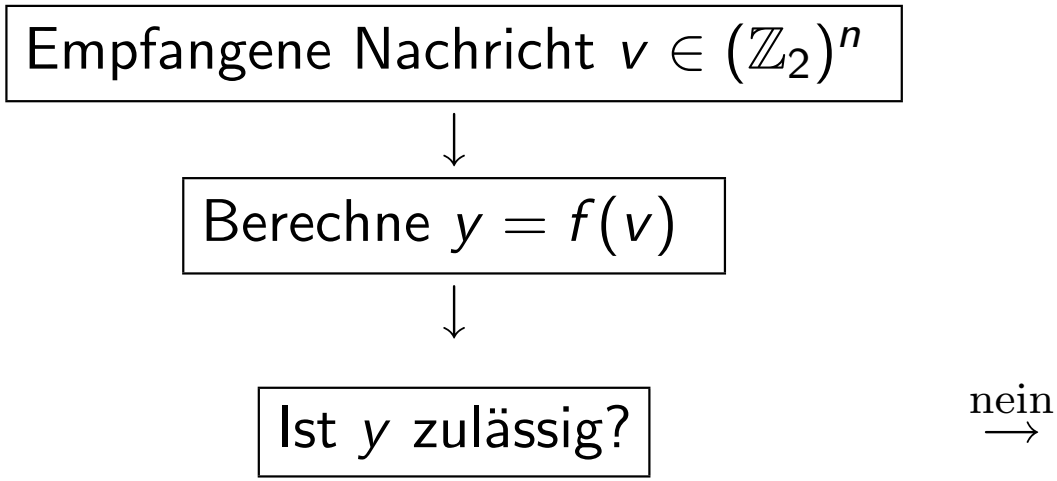
Eine empfangene Nachricht $v \in (\mathbb{Z}_2)^n$, die möglicherweise verfälscht worden ist, kann nun wie folgt decodiert werden.

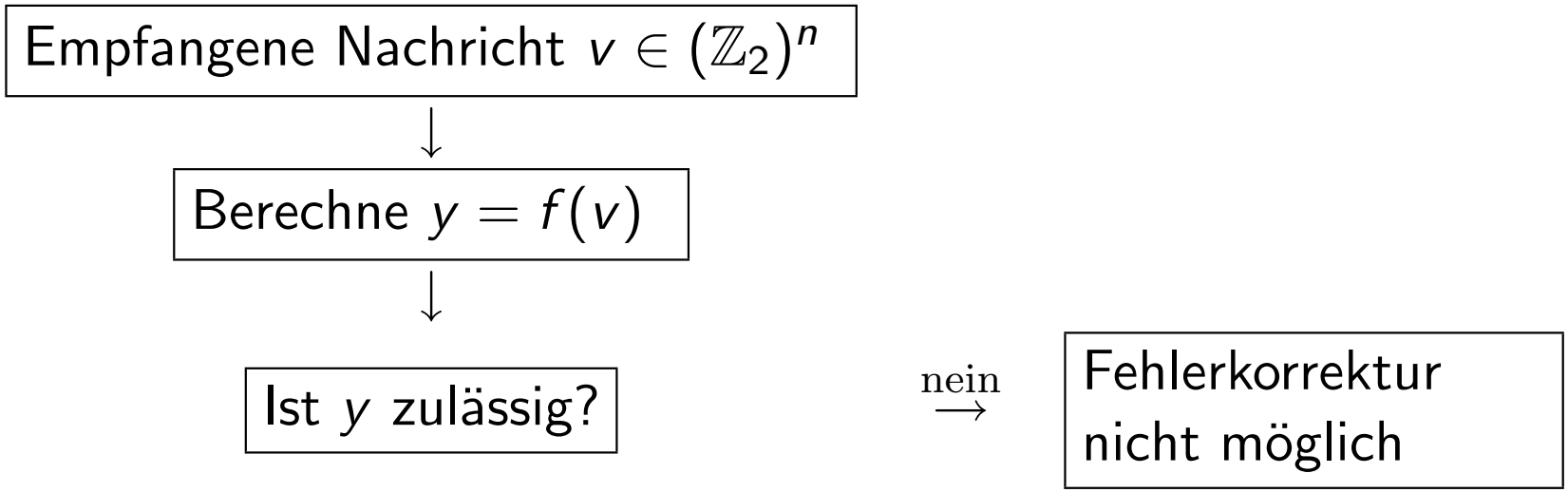
Empfangene Nachricht $v \in (\mathbb{Z}_2)^n$

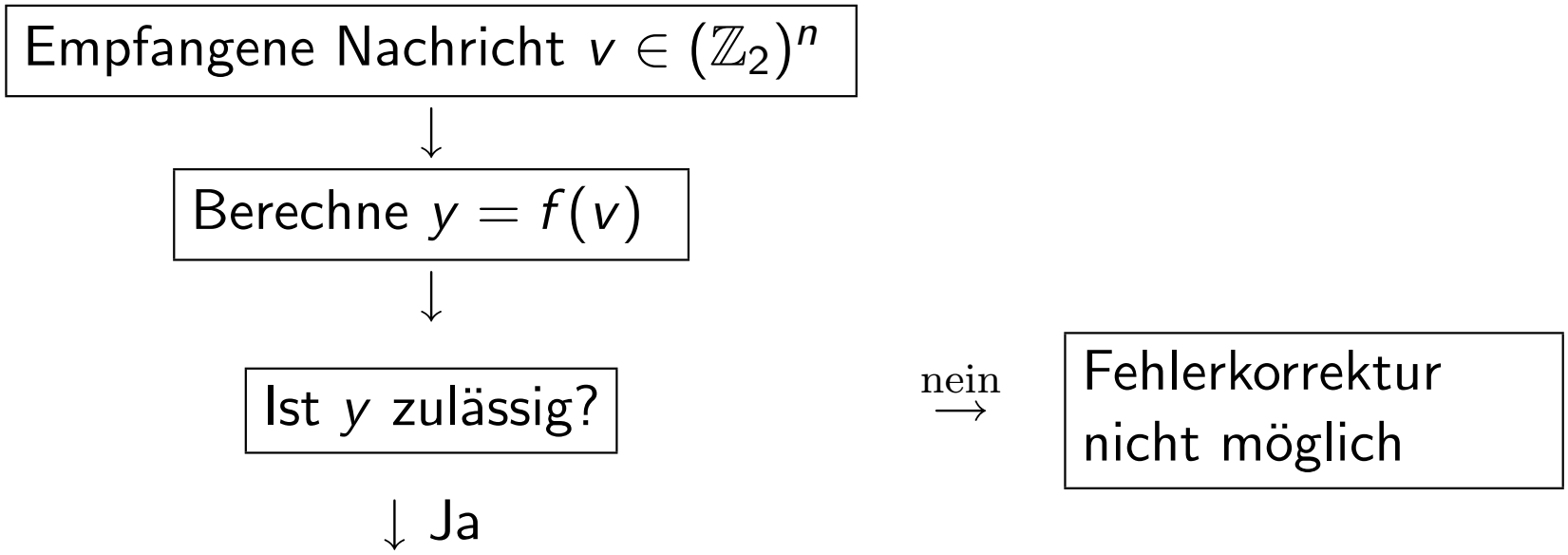
Empfangene Nachricht $v \in (\mathbb{Z}_2)^n$

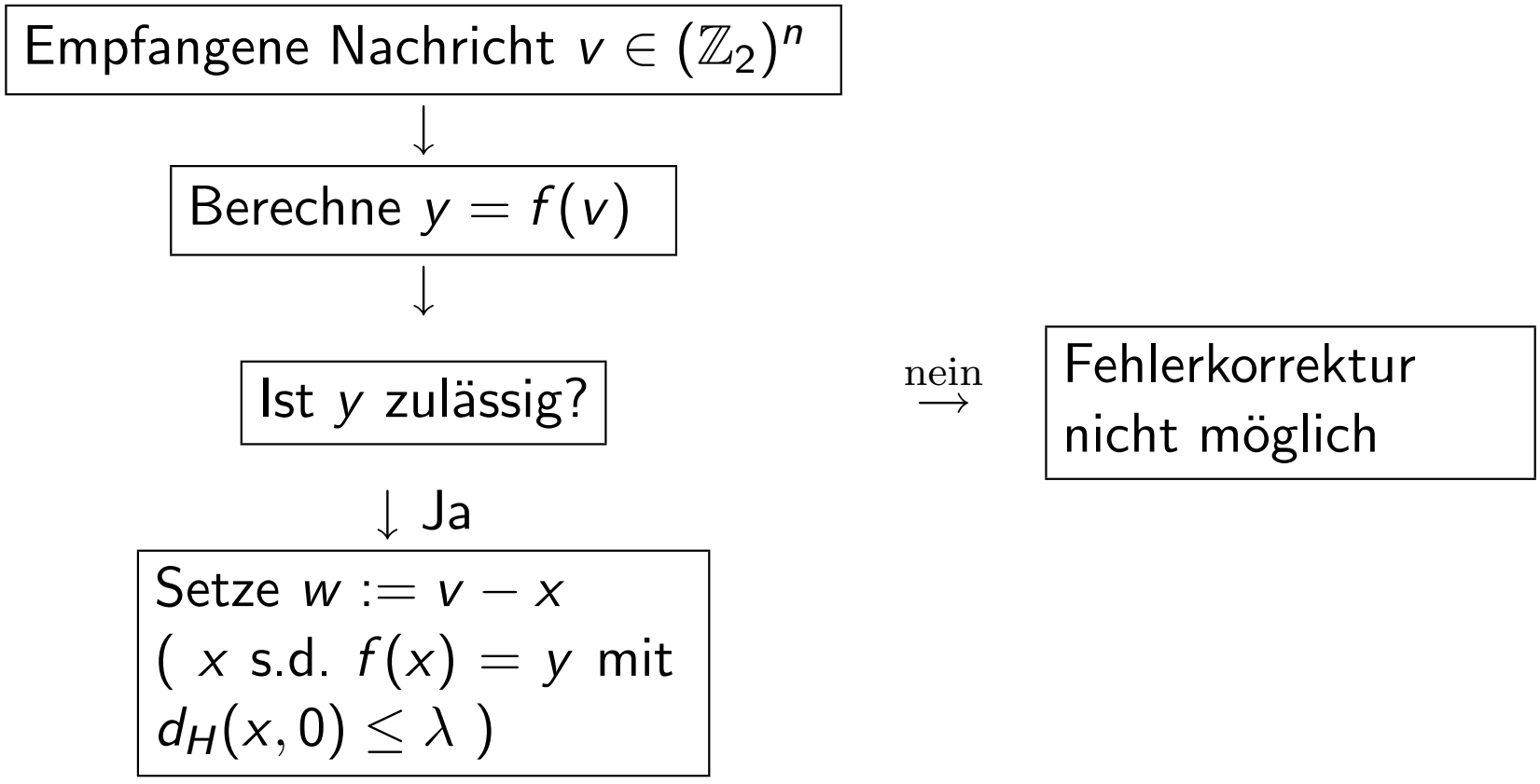

Empfangene Nachricht $v \in (\mathbb{Z}_2)^n$
Berechne $y = f(v)$

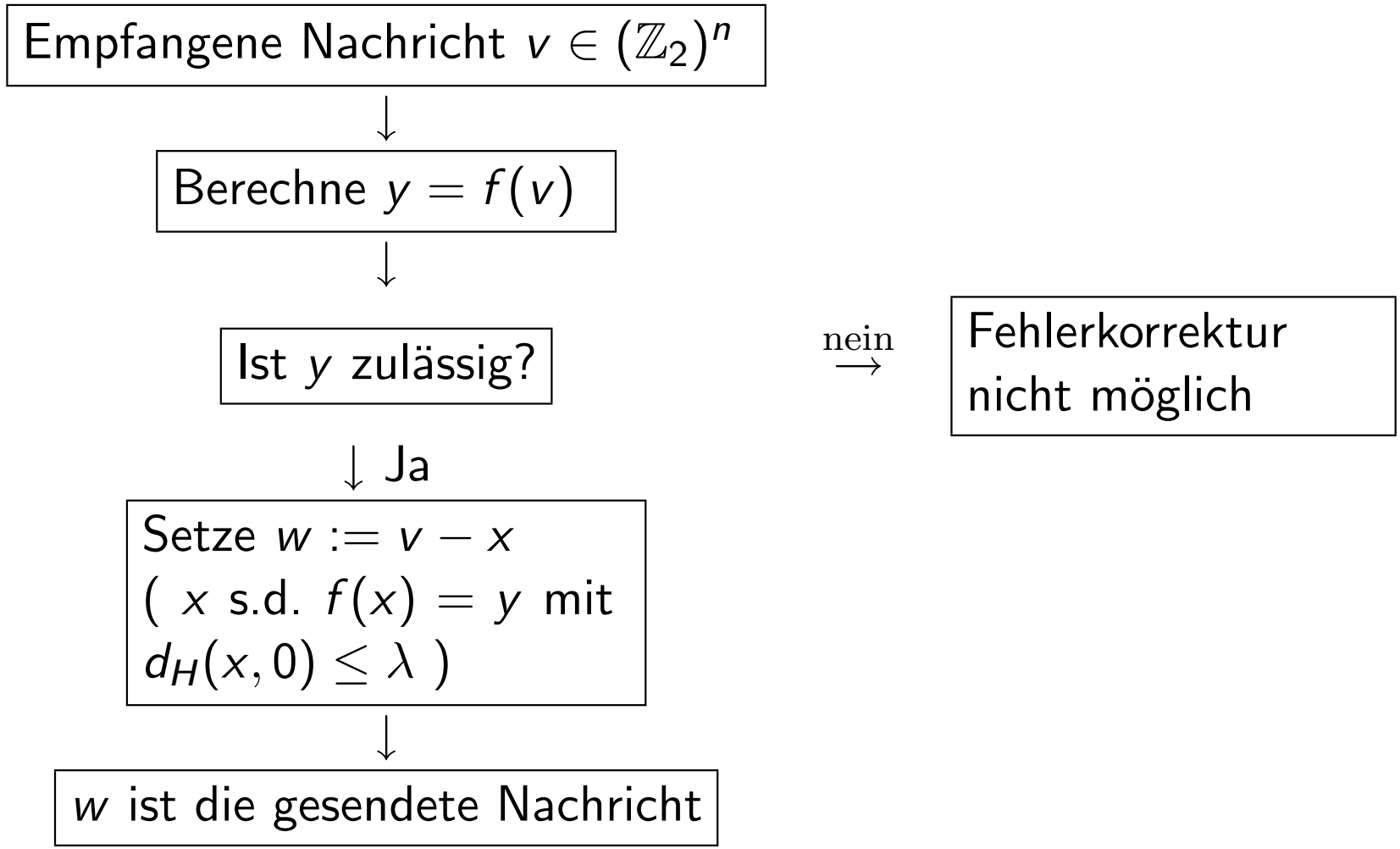

Empfangene Nachricht $v \in (\mathbb{Z}_2)^n$
Berechne $y = f(v)$
Ist y zulässig?


Empfangene Nachricht $v \in (\mathbb{Z}_2)^n$
Berechne $y = f(v)$
Ist y zulässig?
nein


Empfangene Nachricht $v \in (\mathbb{Z}_2)^n$
Berechne $y = f(v)$
Ist y zulässig?
nein
Fehlerkorrektur nicht möglich


Empfangene Nachricht $v \in (\mathbb{Z}_2)^n$
Berechne $y = f(v)$
Ist y zulässig?
nein
Fehlerkorrektur nicht möglich
Ja


Empfangene Nachricht $v \in (\mathbb{Z}_2)^n$
Berechne $y = f(v)$
Ist y zulässig?
nein
Fehlerkorrektur nicht möglich
Ja
Setze $w := v - x$
(x s.d. $f(x) = y$ mit $d_H(x, 0) \leq \lambda$)



Man kann $x = v - w$ als den „Fehlervektor“ ansehen.

Wie speichert man die Filmen auf DVD?

Wie speichert man die Filmen auf DVD?

1. Man übersetzt die Filmen in einer Folge aus 0 und 1 (Digitalisierung)

Wie speichert man die Filmen auf DVD?

1. Man übersetzt die Filmen in einer Folge aus 0 und 1 (Digitalisierung)
2. Man archiviert sie

Wie speichert man die Filmen auf DVD?

1. Man übersetzt die Filmen in einer Folge aus 0 und 1 (Digitalisierung)
2. Man archiviert sie (versucht diese Folge möglich klein zu machen, ohne information zu verlieren)

Wie speichert man die Filmen auf DVD?

1. Man übersetzt die Filmen in einer Folge aus 0 und 1 (Digitalisierung)
2. Man archiviert sie (versucht diese Folge möglich klein zu machen, ohne information zu verlieren)
3. Man codiert sie mit einem $\lambda-$fehlerkoriegierten Code,

Wie speichert man die Filmen auf DVD?

1. Man übersetzt die Filmen in einer Folge aus 0 und 1 (Digitalisierung)
2. Man archiviert sie (versucht diese Folge möglich klein zu machen, ohne information zu verlieren)
3. Man codiert sie mit einem $\lambda-$fehlerkoriegierten Code, λ ist sehr gross.

Wie speichert man die Filmen auf DVD?

1. Man übersetzt die Filmen in einer Folge aus 0 und 1 (Digitalisierung)
2. Man archiviert sie (versucht diese Folge möglich klein zu machen, ohne information zu verlieren)
3. Man codiert sie mit einem $\lambda-$fehlerkoriegierten Code, λ ist sehr gross.
4. Man speichert das Ergebnis auf *DVD*-Rolle.

Wie speichert man die Filmen auf DVD?

1. Man übersetzt die Filmen in einer Folge aus 0 und 1 (Digitalisierung)
2. Man archiviert sie (versucht diese Folge möglich klein zu machen, ohne information zu verlieren)
3. Man codiert sie mit einem $\lambda-$fehlerkoriegierten Code, λ ist sehr gross.
4. Man speichert das Ergebnis auf *DVD*-Rolle.
5. Um Film zu sehen,

Wie speichert man die Filmen auf DVD?

1. Man übersetzt die Filmen in einer Folge aus 0 und 1 (Digitalisierung)
2. Man archiviert sie (versucht diese Folge möglich klein zu machen, ohne information zu verlieren)
3. Man codiert sie mit einem $\lambda-$fehlerkoriegierten Code, λ ist sehr gross.
4. Man speichert das Ergebnis auf *DVD*-Rolle.
5. Um Film zu sehen, muss man zuerst decodieren, und dann dearchivieren.

Wie speichert man die Filmen auf DVD?

1. Man übersetzt die Filmen in einer Folge aus 0 und 1 (Digitalisierung)
2. Man archiviert sie (versucht diese Folge möglich klein zu machen, ohne information zu verlieren)
3. Man codiert sie mit einem $\lambda-$fehlerkoriegierten Code, λ ist sehr gross.
4. Man speichert das Ergebnis auf *DVD*-Rolle.
5. Um Film zu sehen, muss man zuerst decodieren, und dann dearchivieren.
6. Falls die Rolle kleine Kratzen hat,

Wie speichert man die Filmen auf DVD?

1. Man übersetzt die Filmen in einer Folge aus 0 und 1 (Digitalisierung)
2. Man archiviert sie (versucht diese Folge möglich klein zu machen, ohne information zu verlieren)
3. Man codiert sie mit einem $\lambda-$fehlerkoriegierten Code, λ ist sehr gross.
4. Man speichert das Ergebnis auf *DVD*-Rolle.
5. Um Film zu sehen, muss man zuerst decodieren, und dann dearchivieren.
6. Falls die Rolle kleine Kratzen hat, kann man den Film immer noch sehen, weil wir kleine Fehler korriegieren können.

Wie speichert man die Filmen auf DVD?

1. Man übersetzt die Filmen in einer Folge aus 0 und 1 (Digitalisierung)
2. Man archiviert sie (versucht diese Folge möglich klein zu machen, ohne information zu verlieren)
3. Man codiert sie mit einem $\lambda-$fehlerkoriegierten Code, λ ist sehr gross.
4. Man speichert das Ergebnis auf *DVD*-Rolle.
5. Um Film zu sehen, muss man zuerst decodieren, und dann dearchivieren.
6. Falls die Rolle kleine Kratzen hat, kann man den Film immer noch sehen, weil wir kleine Fehler korriegieren können.
7. Falls es mehrere kratzen gibt,

Wie speichert man die Filmen auf DVD?

1. Man übersetzt die Filmen in einer Folge aus 0 und 1 (Digitalisierung)
2. Man archiviert sie (versucht diese Folge möglich klein zu machen, ohne information zu verlieren)
3. Man codiert sie mit einem $\lambda-$fehlerkoriegierten Code, λ ist sehr gross.
4. Man speichert das Ergebnis auf *DVD*-Rolle.
5. Um Film zu sehen, muss man zuerst decodieren, und dann dearchivieren.
6. Falls die Rolle kleine Kratzen hat, kann man den Film immer noch sehen, weil wir kleine Fehler korriegieren können.
7. Falls es mehrere kratzen gibt, kann es zu Verzögerung kommen,

Wie speichert man die Filmen auf DVD?

1. Man übersetzt die Filmen in einer Folge aus 0 und 1 (Digitalisierung)
2. Man archiviert sie (versucht diese Folge möglich klein zu machen, ohne information zu verlieren)
3. Man codiert sie mit einem $\lambda-$fehlerkoriegierten Code, λ ist sehr gross.
4. Man speichert das Ergebnis auf *DVD*-Rolle.
5. Um Film zu sehen, muss man zuerst decodieren, und dann dearchivieren.
6. Falls die Rolle kleine Kratzen hat, kann man den Film immer noch sehen, weil wir kleine Fehler korriegieren können.
7. Falls es mehrere kratzen gibt, kann es zu Verzögerung kommen, weil man braucht mehr Zeit,

Wie speichert man die Filmen auf DVD?

1. Man übersetzt die Filmen in einer Folge aus 0 und 1 (Digitalisierung)
2. Man archiviert sie (versucht diese Folge möglich klein zu machen, ohne information zu verlieren)
3. Man codiert sie mit einem $\lambda-$fehlerkoriegierten Code, λ ist sehr gross.
4. Man speichert das Ergebnis auf *DVD*-Rolle.
5. Um Film zu sehen, muss man zuerst decodieren, und dann dearchivieren.
6. Falls die Rolle kleine Kratzen hat, kann man den Film immer noch sehen, weil wir kleine Fehler korriegieren können.
7. Falls es mehrere kratzen gibt, kann es zu Verzögerung kommen, weil man braucht mehr Zeit, um aller Fehler zu entdecken.

Wie speichert man die Filmen auf DVD?

1. Man übersetzt die Filmen in einer Folge aus 0 und 1 (Digitalisierung)
2. Man archiviert sie (versucht diese Folge möglich klein zu machen, ohne information zu verlieren)
3. Man codiert sie mit einem $\lambda-$fehlerkoriegierten Code, λ ist sehr gross.
4. Man speichert das Ergebnis auf *DVD*-Rolle.
5. Um Film zu sehen, muss man zuerst decodieren, und dann dearchivieren.
6. Falls die Rolle kleine Kratzen hat, kann man den Film immer noch sehen, weil wir kleine Fehler korriegieren können.
7. Falls es mehrere kratzen gibt, kann es zu Verzögerung kommen, weil man braucht mehr Zeit, um aller Fehler zu entdecken.