Überblick zu Kapitel 2 von „Einführung in die Kanalcodierung"

Dieses Kapitel behandelt die **Reed–Solomon–Codes**, die Anfang der 1960er Jahre von **Irving Stoy Reed** und **Gustave Solomon** erfunden wurden. Im Gegensatz zu den binären Blockcodes basieren diese auf einem Galoisfeld GF(2^m) mit $m > 1$. Sie arbeiten also mit mehrstufigen Symbolen anstelle von Binärzeichen (Bit).

Im Einzelnen werden in diesem Kapitel behandelt:

- Die Grundlagen der *linearen Algebra*: Menge, Gruppe, Ring, Körper, endlicher Körper,
- die Definition von *Erweiterungskörpern* ⇒ GF(2^m) und die zugehörigen Operationen,
- die Bedeutung von *irreduziblen Polynomen* und *primitiven Elementen*,
- die *Beschreibungs– und Realisierungsmöglichkeiten* eines Reed–Solomon–Codes,
- Fehlerkorrektur eines solchen Codes beim *Auslöschungskanal* (BEC),
- die Decodierung mit Hilfe des *Error Locator Polynoms* ⇒ *Bounded Distance Decoding*,
- die *Blockfehlerwahrscheinlichkeit* der Reed–Solomon–Codes und typische *Anwendungen*.

Geeignete Literatur: [Böch15] – [Bos98] – [Bos99] – [CT06] – [Fri96] – [Gal68] – [JZ99] – [KM+08] – [Liv10] – [LN97] – [Mas69] – [Pet60] – [RS60] – [Sha48] – [SK+75]

Die grundlegende Theorie wird auf 44 Seiten dargelegt. Dazu beinhaltet dieses Kapitel noch 54 Grafiken, 16 Aufgaben und elf Zusatzaufgaben mit insgesamt 136 Teilaufgaben, sowie ein Lernvideo (LV) und drei Interaktionsmodule (IM):

- **Galoisfeld: Eigenschaften und Anwendungen** (LV zu Kap. 2.1 und 2.2 – Dauer: 39:00)
- **Komplementäre Gaußsche Fehlerfunktionen** (IM zu Kap. 2.6)
- **Symbolfehlerwahrscheinlichkeit von Digitalsystemen** (IM zu Kap. 2.6)
- **Ereigniswahrscheinlichkeiten der Binomialverteilung** (IM zu Kap. 2.6)

Definition eines Galoisfeldes

Bevor wir uns der Beschreibung der Reed–Solomon–Codes zuwenden können, benötigen wir einige algebraische Grundbegriffe. Beginnen wir mit den Eigenschaften eines Galoisfeldes GF(q), benannt nach dem Franzosen **Évariste Galois,** dessen Biografie für einen Mathematiker eher ungewöhnlich ist.

Definition: Ein **Galoisfeld** GF(q) ist ein **endlicher Körper** mit q Elementen z_0, z_1, ... , z_{q-1}, wenn alle acht nachfolgend aufgeführten Aussagen hinsichtlich der Addition („+") und der Multiplikation („·") zutreffen. Addition und Multiplikation sind hierbei modulo q zu verstehen.

(A) GF(q) ist in sich abgeschlossen ⇒ **Closure**:

$$\forall\, z_i \in \mathrm{GF}(q),\ z_j \in \mathrm{GF}(q):\quad (z_i + z_j) \in \mathrm{GF}(q),\ (z_i \cdot z_j) \in \mathrm{GF}(q)\,.$$

(B) Es gibt ein hinsichtlich der Addition neutrales Element N_A, das man oft auch als das *Nullelement* bezeichnet ⇒ **Identity for „+"**:

$$\exists\, z_j \in \mathrm{GF}(q):\quad z_i + z_j = z_i \quad \Rightarrow \quad z_j = N_\mathrm{A} = \text{"0"}\quad (\text{Nullelement})\,.$$

(C) Es gibt ein hinsichtlich der Multiplikation neutrales Element N_M, das oft auch als das *Einselement* bezeichnet wird ⇒ **Identity for „·"**:

$$\exists\, z_j \in \mathrm{GF}(q):\quad z_i \cdot z_j = z_i \quad \Rightarrow \quad z_j = N_\mathrm{M} = \text{"1"}\quad (\text{Einselement})\,.$$

(D) Für jedes z_i existiert eine additive Inverse ⇒ **Inverse for „+"**:

$$\forall\, z_i \in \mathrm{GF}(q),\ \exists\, \mathrm{Inv_A}(z_i) \in \mathrm{GF}(q):\quad z_i + \mathrm{Inv_A}(z_i) = N_\mathrm{A} = \text{"0"}$$
$$\Rightarrow\quad \text{kurz}: \mathrm{Inv_A}(z_i) = -z_i\,.$$

(E) Für jedes z_i mit Ausnahme des Nullelements existiert die multiplikative Inverse ⇒ **Inverse for „·"**:

$$\forall\, z_i \in \mathrm{GF}(q),\ z_i \neq N_\mathrm{A},\ \exists\, \mathrm{Inv_M}(z_i) \in \mathrm{GF}(q):\quad z_i \cdot \mathrm{Inv_M}(z_i) = N_\mathrm{M} = \text{"1"}$$
$$\Rightarrow\quad \text{kurz}: \mathrm{Inv_M}(z_i) = z_i^{-1}\,.$$

(F) Für Addition und Multiplikation gilt jeweils das Kommutativgesetz ⇒ **Commutative Law**:

$$\forall\, z_i \in \mathrm{GF}(q),\ z_j \in \mathrm{GF}(q):\quad z_i + z_j = z_j + z_i,\ z_i \cdot z_j = z_j \cdot z_i\,.$$

(G) Für Addition und Multiplikation gilt jeweils das Assoziativgesetz ⇒ **Associative Law**:

$$\forall\, z_i,\ z_j,\ z_k \in \mathrm{GF}(q):\quad (z_i + z_j) + z_k = z_i + (z_j + z_k),\ (z_i \cdot z_j) \cdot z_k = z_i \cdot (z_j \cdot z_k)\,.$$

(H) Für die Kombination „Addition – Multiplikation" gilt das Distributivgesetz ⇒ **Distributive Law**:

$$\forall\, z_i,\ z_j,\ z_k \in \mathrm{GF}(q):\quad (z_i + z_j) \cdot z_k = z_i \cdot z_k + z_j \cdot z_k\,.$$

Es sei nochmals darauf hingewiesen, dass Addition („+") und Multiplikation („·") modulo q zu verstehen sind. Hierbei bezeichnet die **Ordnung q** die Anzahl der Elemente des Galoisfeldes.

Beispiele und Eigenschaften von Galoisfeldern

Wir überprüfen zunächst, ob für die binäre Zahlenmenge $Z_2 = \{0, 1\}$ $\Rightarrow$ $q = 2$ (gültig für den einfachen Binärcode) die auf der letzten Seite genannten acht Kriterien tatsächlich erfüllt werden, so dass man tatsächlich von „GF(2)" sprechen kann. Sie sehen nachfolgend die Additions– und Multiplikationstabelle:

$$Z_2 = \{0, 1\} \quad \Rightarrow \quad \begin{array}{c|cc} + & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 0 \end{array} \qquad \begin{array}{c|cc} \cdot & 0 & 1 \\ \hline 0 & 0 & 0 \\ 1 & 0 & 1 \end{array} \quad \Rightarrow \quad \mathrm{GF}(2)\,.$$

Man erkennt aus dieser Darstellung:

- Jedes Element der Additions– und der Multiplikationstabelle von Z_2 ist wieder entweder $z_0 = 0$ oder $z_1 = 1$ $\Rightarrow$ das Kriterium **(A)** ist erfüllt.
- Die Zahlenmenge Z_2 enthält das Nullelement ($N_\mathrm{A} = z_0 = 0$) und das Einselement ($N_\mathrm{M} = z_1 = 1$) $\Rightarrow$ die Kriterien **(B)** und **(C)** sind ebenfalls erfüllt.
- Die additiven Inversen $\mathrm{Inv_A}(0) = 0$ und $\mathrm{Inv_A}(1) = (–1) \bmod 2 = 1$ existieren und gehören zu Z_2. Ebenso gibt es die multiplikative Inverse $\mathrm{Inv_M}(1) = 1$ $\Rightarrow$ die Kriterien **(D)** und **(E)** sind erfüllt.
- Die Gültigkeit des Kommutativgesetzes **(F)** in der Menge Z_2 erkennt man an der Symmetrie bezüglich der Tabellendiagonalen. Die Kriterien **(G)** und **(H)** werden hier ebenfalls erfüllt.

Die Zahlenmenge $Z_3 = \{0, 1, 2\}$ $\Rightarrow$ $q = 3$ erfüllt alle acht Kriterien und ist somit ein Galoisfeld GF(3):

$$Z_3 = \{0, 1, 2\} \quad \Rightarrow \quad \begin{array}{c|ccc} + & 0 & 1 & 2 \\ \hline 0 & 0 & 1 & 2 \\ 1 & 1 & 2 & 0 \\ 2 & 2 & 0 & 1 \end{array} \qquad \begin{array}{c|ccc} \cdot & 0 & 1 & 2 \\ \hline 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 2 \\ 2 & 0 & 2 & 1 \end{array} \quad \Rightarrow \quad \mathrm{GF}(3)\,.$$

Dagegen ist die Zahlenmenge $Z_4 = \{0, 1, 2, 3\}$ $\Rightarrow$ $q = 4$ kein Galoisfeld. Der Grund hierfür ist, dass es hier zum Element $z_2 = 2$ keine multiplikative Inverse gibt. Bei einem Galoisfeld müsste nämlich gelten: $2 \cdot \mathrm{Inv_M}(2) = 1$. In nachfolgender Multiplikationstabelle gibt es aber in der dritten Zeile und in der dritten Spalte (jeweils gültig für den Multiplikanden $z_2 = 2$) keinen Eintrag mit „1".

$$Z_4 = \{0, 1, 2, 3\} \quad \Rightarrow \quad \begin{array}{c|cccc} + & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 1 & 2 & 3 \\ 1 & 1 & 2 & 3 & 0 \\ 2 & 2 & 3 & 0 & 1 \\ 3 & 3 & 0 & 1 & 2 \end{array} \qquad \begin{array}{c|cccc} \cdot & 0 & 1 & 2 & 3 \\ \hline 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 2 & 3 \\ 2 & 0 & 2 & 0 & 2 \\ 3 & 0 & 3 & 2 & 1 \end{array} \quad \Rightarrow \quad \text{kein } \mathrm{GF}(4)\,.$$

Das Ergebnis dieser Seite soll hier ohne weiteren Beweis verallgemeinert werden:

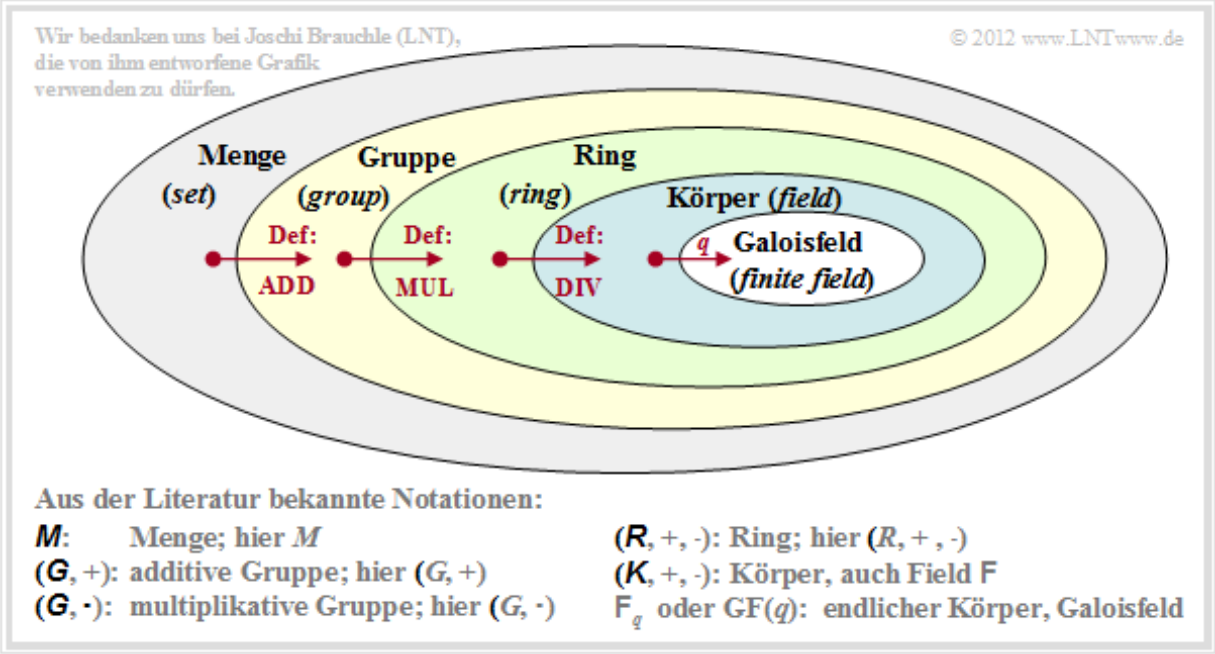
- Ein Galoisfeld GF(q) kann in der hier beschriebenen Weise als **Ring** von Integergrößen modulo q nur dann gebildet werden, wenn q eine Primzahl ist: $q = 2$, $q = 3$, $q = 5$, $q = 7$, $q = 11$, ...
- Kann man aber die Ordnung q in der Form $q = P^m$ mit einer Primzahl P und ganzzahligem m ausdrücken, so lässt sich das Galoisfeld GF(q) über einen **Erweiterungskörper** finden.

Gruppe, Ring, Körper – algebraische Grundbegriffe

Auf den beiden ersten Seiten sind bereits einige algebraische Grundbegriffe gefallen, ohne dass deren Bedeutungen genauer erläutert wurden. Dies soll nun in aller Kürze aus Sicht eines Nachrichtentechnikers nachgeholt werden, wobei wir uns vorwiegend auf die Darstellung in **[Fri96]** und **[KM+08]** beziehen. Zusammenfassend lässt sich sagen:

Definition: Ein **Galoisfeld** GF(q) ist ein Körper (englisch: *Field*) mit einer begrenzten Anzahl (q) an Elementen ⇒ *endlicher Körper*. Jeder Körper ist wieder ein Sonderfall eines Rings (gleichlautende englische Bezeichnung), der sich selbst wieder als Spezialfall einer abelschen Gruppe (englisch: *Abelian Group*) darstellen lässt.

Die folgende Grafik verdeutlicht schrittweise, wie sich aus einer Menge durch Definition von Addition, Multiplikation und Division innerhalb dieser Menge eine *Gruppe*, ein *Ring* und ein *Körper* ergibt. Weiter angegeben sind in der Grafik auch in der Literatur häufig verwendete Schriftzeichen für eine Menge, eine Gruppe, einen Ring, einen Körper und einen endlichen Körper. Aufgrund von Restriktionen durch den HTML–Zeichensatz benutzen wir in LNTwww die Zeichen M, G, R, K (bzw. F) sowie GF(q).



Auf den beiden nächsten Seiten werden die hier genannten algebraischen Begriffe noch etwas genauer behandelt. Zum Verstehen der Reed–Solomon–Codes sind diese Kenntnisse allerdings nicht unbedingt erforderlich. Sie könnten also auch direkt zum Kapitel **Erweiterungskörper** springen.

Algebraische Gruppe und Beispiele

Für die allgemeinen Definitionen von Gruppe (und später Ring) gehen wir von einer Menge mit unendlich vielen Elementen aus:

$$M = \{ z_1,\ z_2,\ z_3,\ \ldots \}.$$

Definition: Eine **algebraische Gruppe** $(G, +)$ ist eine (beliebige) Teilmenge $G \subset M$ zusammen mit einer zwischen allen Elementen definierten additiven Verknüpfung („+"), allerdings nur dann, wenn die folgenden Eigenschaften zwingend erfüllt sind:

- Für alle $z_i, z_j \in G$ gilt $(z_i + z_j) \in G \Rightarrow$ *Closure*–Kriterium für „+".
- Es gibt stets ein hinsichtlich der Addition neutrales Element $N_{\rm A} \in G$, so dass für alle $z_i \in G$ gilt: $z_i + N_{\rm A} = z_i$. Bei einer Zahlengruppe ist $N_{\rm A} = 0$.
- Für alle $z_i \in G$ gibt es zudem ein hinsichtlich der Addition inverses Element $\text{Inv}_{\rm A}(z_i) \in G$ mit der Eigenschaft $z_i + \text{Inv}_{\rm A}(z_i) = N_{\rm A}$. Bei einer Zahlengruppe ist $\text{Inv}_{\rm A}(z_i) = -z_i$.
- Für alle $z_i, z_j, z_k \in G$ gilt: $z_i + (z_j + z_k) = (z_i + z_j) + z_k \Rightarrow$ Assoziativgesetz für „+".

Wird zusätzlich für alle $z_i, z_j \in G$ das Kommutativgesetz $z_i + z_j = z_j + z_i$ erfüllt, so spricht man von einer kommutativen oder einer **abelschen Gruppe**, benannt nach dem norwegischen Mathematiker **Niels Hendrik Abel.**

Beispiele für eine algebraische Gruppe:

Die *Menge der rationalen Zahlen* ist definiert als die Menge aller Quotienten I/J mit ganzen Zahlen I und $J \neq 0$. Diese Menge ist eine Gruppe $(G, \text{„+"})$ hinsichtlich der Addition, da

- für alle $a \in G$ und $b \in G$ auch die Summe $a + b$ wieder zu G gehört,
- das Assozitativgesetz gilt,
- mit der 0 das neutrale Element der Addition in G enthalten ist, und
- es für jedes a die additive Inverse $b = -a$ existiert.

Da zudem das Kommutativgesetz erfüllt ist, handelt es sich um eine *abelsche Gruppe*.

Die *Menge der natürlichen Zahlen*, $\{0, 1, 2, ...\}$ ist hinsichtlich Addition keine algebraische Gruppe, da es für kein einziges Element z_i die additive Inverse $(-z_i)$ gibt.

Die *begrenzte natürliche Zahlenmenge* $\{0, 1, 2, \ldots, q-1\}$ erfüllt dagegen dann die Bedingungen an eine Gruppe $(G, +)$, wenn man die Addition modulo q definiert. Dagegen ist $\{1, 2, 3, \ldots, q\}$ keine Gruppe, da das neutrale Element der Addition („0") fehlt.

Algebraischer Ring und Beispiele

Entsprechend der **Übersichtsgrafik** kommt man von der Gruppe $(G, +)$ durch Definition einer zweiten Rechenoperation – der Multiplikation („·") – zum Ring $(R, +, \cdot)$. Man benötigt hierfür also neben einer Additionstabelle auch eine Multiplikationstabelle.

Definition: Ein **algebraischer Ring** ist eine Teilmenge $R \subset G \subset M$ zusammen mit zwei in dieser Menge definierten Rechenoperationen, der Addition („+") und der Multiplikation („·"). Dabei müssen die folgenden Eigenschaften erfüllt werden:

- Hinsichtlich der Addition ist Ring $(R, +, \cdot)$ eine **abelsche Gruppe** $(G, +)$.
- Zusätzlich gilt für alle $z_i, z_j \in R$ auch $(z_i \cdot z_j) \in R \Rightarrow$ **Closure–Kriterium für „·"**.
- Es gibt ein **hinsichtlich Multiplikation neutrales Element** $N_{\rm M} \in R$, so dass für alle $z_i \in R$ gilt: $z_i \cdot N_{\rm M} = z_i$. Bei einem Zahlenring gilt $N_{\rm M} = 1$.
- Für alle $z_i, z_j, z_k \in R$ gilt $z_i \cdot (z_j \cdot z_k) = (z_i \cdot z_j) \cdot z_k \Rightarrow$ **Assoziativgesetz für „·"**.
- Für alle $z_i, z_j, z_k \in R$ gilt $z_i \cdot (z_j + z_k) = z_i \cdot z_j + z_i \cdot z_k \Rightarrow$ **Distributivgesetz für „·"**.

Ein Ring $(R, +, \cdot)$ ist nicht notwendigerweise kommutativ. Gilt tatsächlich auch für alle Elemente $z_i, z_j \in R$ das Kommutativgesetz $z_i \cdot z_j = z_j \cdot z_i$ hinsichtlich der Multiplikation, so spricht man in der Fachliteratur von einem **kommutativen Ring**. Existiert für jedes Element $z_i \in R$ mit Ausnahme von $N_{\rm A}$ (neutrales Element der Addition, Nullelement) ein hinsichtlich der Multiplikation inverses Element $\text{Inv}_{\rm M}(z_i)$, so dass $z_i \cdot \text{Inv}_{\rm M}(z_i) = 1$ gilt, so liegt ein **Divisionsring** (englisch: *Division Ring*) vor.

Weiter sollen die folgenden Voraussetzungen gelten:

- Der Ring ist **nullteilerfrei** (englisch: *free of zero devisors*), wenn aus $z_i \cdot z_j = 0$ zwingend $z_i = 0$ oder $z_j = 0$ folgt. In der abstrakten Algebra ist ein Nullteiler eines Ringes ein vom Nullelement verschiedenes Element z_i, falls es ein Element $z_j \neq 0$ gibt, so dass das Produkt $z_i \cdot z_j = 0$ ist.
- Ein kommutativer nullteilerfreier Ring wird als **Integritätsring** oder **Integritätsbereich** (englisch: *Integral Domain*) bezeichnet.

Vergleicht man die Definitionen von **Gruppe,** Ring (oben auf dieser Seite), **Körper** und **Galoisfeld,** so erkennt man, dass ein Galoisfeld GF(q)

- ein endlicher Körper (englisch: *Field*) mit q Elementen ist,
- gleichzeitig als *Commutative Division Ring* aufgefasst werden kann, und
- einen Integritätsbereich (englisch: *Integral Domain*) beschreibt.

GF(2^2) – Beispiel eines Erweiterungskörpers (1)

Im **Kapitel 2.1** wurde bereits gezeigt, dass die endliche **Zahlenmenge {0, 1, 2, 3}** ⇒ $q = 4$ nicht die Eigenschaften eines Galoisfeldes GF(4) erfüllt. Vielmehr ergeben sich für die Addition modulo 4 und die Multiplikation modulo 4 folgende Tabellen:

Operationen modulo $q = 4$ ⇒

+	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

·	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

Für $z_i = 2$ gibt es keine multiplikative Inverse $\text{Inv}_\text{M}(z_i)$. Dies erkennt man daran, dass kein einziges Element $z_j \in \{0, 1, 2, 3\}$ die Bedingung $2 \cdot z_j = 1$ erfüllt. Geht man dagegen vom binären Galoisfeld GF(2) = {0, 1} aus und erweitert dieses entsprechend der Gleichung

$$\text{GF}(2^2) = \left\{k_0 + k_1 \cdot \alpha \mid k_0, k_1 \in \text{GF}(2) = \{0, 1\}\right\},$$

so ergibt sich die ebenfalls endliche **Menge {0, 1, α, 1 + α}** ⇒ die Ordnung ist weiterhin $q = 4$. Führt man die Rechenoperationen modulo $p(\alpha) = \alpha^2 + \alpha + 1$ durch, so erhält man das folgende Ergebnis:

modulo $p(\alpha)$ ⇒

+	0	1	α	$1+\alpha$
0	0	1	α	$1+\alpha$
1	1	0	$1+\alpha$	α
α	α	$1+\alpha$	0	1
$1+\alpha$	$1+\alpha$	α	1	0

·	0	1	α	$1+\alpha$
0	0	0	0	0
1	0	1	α	$1+\alpha$
α	0	α	$1+\alpha$	1
$1+\alpha$	0	$1+\alpha$	1	α

Hierzu ist anzumerken:

- Die neutralen Elemente der Addition bzw. Multiplikation sind weiterhin $N_\text{A} = 0$ und $N_\text{M} = 1$.
- Da bei Modulo–Ausführung kein Unterschied zwischen Addition und Subtraktion besteht, ist $\alpha + \alpha = \alpha - \alpha = 0$. Für alle z_i gilt somit: Die additive Inverse von z_i ist das Element z_i selbst.
- Die Einträge in der Multiplikationstabelle ergeben sich nach folgenden Berechnungen:

$$\begin{aligned}\left[\alpha \cdot (1+\alpha)\right] \bmod p(\alpha) &= (\alpha^2 + \alpha) \bmod (\alpha^2 + \alpha + 1) = 1,\\ \left[\alpha \cdot \alpha\right] \bmod p(\alpha) &= (\alpha^2) \bmod (\alpha^2 + \alpha + 1) = 1 + \alpha,\\ \left[(1+\alpha) \cdot (1+\alpha)\right] \bmod p(\alpha) &= (\alpha^2 + 1) \bmod (\alpha^2 + \alpha + 1) = \alpha.\end{aligned}$$

- Damit existieren für alle Elemente mit Ausnahme des Nullelements die multiplikativen Inversen:

$$\text{Inv}_\text{M}(1) = 1, \quad \text{Inv}_\text{M}(\alpha) = 1 + \alpha, \quad \text{Inv}_\text{M}(1 + \alpha) = \alpha.$$

Daraus folgt: Die Menge {0, 1, α, 1 + α} stellt zusammen mit den zwei Rechenoperationen *Addition* und *Multiplikation* modulo $p(\alpha) = \alpha^2 + \alpha + 1$ ein **Galoisfeld der Ordnung $q = 4$** dar. Dieses mit **GF(2^2)** = GF(4) bezeichnete Galoisfeld erfüllt alle in **Kapitel 2.1** genannten Eigenschaften.

Im Gegensatz zum Zahlenkörper GF(3) = {0, 1, 2} mit der Eigenschaft, dass $q = 3$ eine Primzahl ist, nennt man GF(2^2) einen **Erweiterungskörper** (englisch: *Extension Field*).

GF(2^2) – Beispiel eines Erweiterungskörpers (2)

Das Polynom $p(\alpha)$ und damit die Bestimmungsgleichung $p(\alpha) = 0$ darf nicht beliebig vorgegeben werden. Zum Beleg hierfür vergleichen wir die Verknüpfungstabellen für zwei verschiedene Polynome.

Polynom entsprechend der letzten Seite $\Rightarrow$ $\boldsymbol{p(\alpha) = \alpha^2 + \alpha + 1}$:

+	0	1	α	$1+\alpha$
0	0	1	α	$1+\alpha$
1	1	0	$1+\alpha$	α
α	α	$1+\alpha$	0	1
$1+\alpha$	$1+\alpha$	α	1	0

·	0	1	α	$1+\alpha$
0	0	0	0	0
1	0	1	α	$1+\alpha$
α	0	α	$1+\alpha$	1
$1+\alpha$	0	$1+\alpha$	1	α

.

Neuer (allerdings ungeeigneter) Ansatz $\Rightarrow$ $\boldsymbol{p(\alpha) = \alpha^2 + 1}$:

+	0	1	α	$1+\alpha$
0	0	1	α	$1+\alpha$
1	1	0	$1+\alpha$	α
α	α	$1+\alpha$	0	1
$1+\alpha$	$1+\alpha$	α	1	0

·	0	1	α	$1+\alpha$
0	0	0	0	0
1	0	1	α	$1+\alpha$
α	0	α	1	$1+\alpha$
$1+\alpha$	0	$1+\alpha$	$1+\alpha$	0

.

Die Additionstabelle ist in beiden Fällen identisch und auch die Multiplikationstabellen unterscheiden sich nur durch die vier Einträge in den beiden unteren Zeilen und den beiden hinteren Spalten:

- Aus $p(\alpha) = 0$ folgt nun für das Produkt $\alpha \cdot \alpha = 1$ und das Produkt $(1+\alpha) \cdot (1+\alpha)$ ergibt das Element 0. Das gemischte Produkt $\alpha \cdot (1+\alpha)$ ist nun $1+\alpha$.
- In der letzten Zeile der Multipliaktionstabelle und auch in der letzten Spalte steht nun keine „1” $\Rightarrow$ Hinsichtlich der Bedingung $p(\alpha) = \alpha^2 + 1 = 0$ existiert die multiplikative Inverse zu $1+\alpha$ nicht.
- Damit erfüllt aber die endliche Menge $\{0, 1, \alpha, 1+\alpha\}$ zusammen mit Rechenoperationen modulo $p(\alpha) = \alpha^2 + 1$ auch nicht die Voraussetzungen eines Erweiterungskörpers GF(2^2).

Fassen wir zusammen: Aus dem binären Galoisfeld GF(2) = {0, 1} lässt sich unter Zuhilfenahme eines Polynoms vom Grad $m = 2$ mit binären Koeffizienten,

$$p(x) = x^2 + k_1 \cdot x + k_0, \quad k_0,\ k_1 \in \{0, 1\},$$

ein Erweiterungskörper GF(2^2) formulieren. Im vorliegenden Fall gibt es nur **ein geeignetes Polynom** $p_1(x) = x^2 + x + 1$. Alle anderen möglichen Polynome vom Grad $m = 2$, nämlich

$$\begin{aligned} p_2(x) &= x^2 + 1 = (x+1) \cdot (x+1), \\ p_3(x) &= x^2 = x \cdot x, \\ p_4(x) &= x^2 + x = (x+1) \cdot x \end{aligned}$$

lassen sich faktorisieren und ergeben keinen Erweiterungskörper. Man nennt die Polynome $p_2(x)$, $p_3(x)$ und $p_4(x)$ **reduzibel**.

Anmerkung: Die Umbenennung der Funktionsvariablen α in x hat nur formale Bedeutung im Hinblick auf die folgenden Seiten.

GF(2^2) – Beispiel eines Erweiterungskörpers (3)

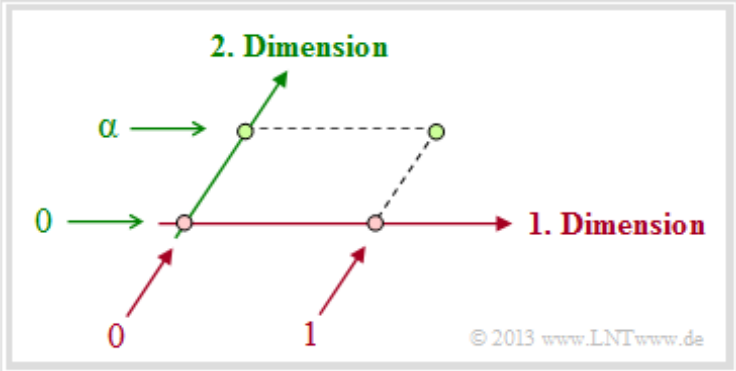
Wir betrachten weiterhin den Körper GF(2^2) = {0, 1, α, 1+α} entsprechend den beiden folgenden Operationstabellen, basierend auf der Nebenbedingung $\boldsymbol{p(\alpha) = \alpha^2 + \alpha + 1 = 0}$ (irreduzibles Ploynom):

+	0	1	α	1+α
0	0	1	α	1+α
1	1	0	1+α	α
α	α	1+α	0	1
1+α	1+α	α	1	0

·	0	1	α	1+α
0	0	0	0	0
1	0	1	α	1+α
α	0	α	1+α	1
1+α	0	1+α	1	α

Welche Bedeutung hat aber nun das neue Element α?

- Das Polynom $p(\alpha) = \alpha^2 + \alpha + 1$ hat keine Nullstelle in GF(2) = {0, 1}. Das bedeutet weiter, dass α weder 0 noch 1 sein kann.
- Wäre $\alpha = 0$ bzw. $\alpha = 1$, so wären zudem zwei der vier Mengenelemente {0, 1, α, 1+α} jeweils identisch. Entweder „0" und „α" sowie „1" und „1+α" oder „1" und „α" sowie „0" und „1+α".
- Vielmehr erhält der eindimensionale Körper GF(2) durch die Einführung des Elementes α eine zweite Dimension. Er wird also zum Galoisfeld GF(2^2) erweitert, wie die folgende Grafik zeigt.



- Das Element α hat ähnliche Bedeutung wie die imaginäre Einheit j, durch die man die Menge der reellen Zahlen unter der Nebenbedingung $j^2 + 1 = 0$ zur Menge der komplexen Zahlen erweitert.
- Aufgrund der Identität $\alpha^2 = 1 + \alpha$, die aus der Nebenbedingung $p(\alpha) = 0$ folgt, kann man in gleicher Weise GF(2^2) = {0, 1, α, α^2} schreiben, wobei nun folgende Operationstabellen gelten:

$\Rightarrow$

+	0	1	α	α^2
0	0	1	α	α^2
1	1	0	α^2	α
α	α	α^2	0	1
α^2	α^2	α	1	0

·	0	1	α	α^2
0	0	0	0	0
1	0	1	α	α^2
α	0	α	α^2	1
α^2	0	α^2	1	α

Polynome über einem endlichen Körper (1)

Ein Polynom in einem endlichen Körper GF(P), wobei P eine Primzahl angibt, hat folgende Form:

$$a(x) = \sum_{i=0}^{m} a_i \cdot x^i = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ \ldots \ + a_m \cdot x^m \, .$$

Anzumerken ist:

- Alle Koeffizienten sind Elemente des Körpers: $a_i \in$ GF(P).
- Ist der führende Koeffizient $a_m \neq 0$, so gibt m den Grad des Polynoms an.

Betrachten wir ein dazu zweites Polynom mit Grad M,

$$b(x) = \sum_{i=0}^{M} b_i \cdot x^i = b_0 + b_1 \cdot x + b_2 \cdot x^2 + \ \ldots \ + b_M \cdot x^M \, ,$$

so erhält man für die Summe (bzw. Differenz) und das Produkt jeweils in GF(P):

$$a(x) \pm b(x) \ = \sum_{i=0}^{\max\,(m,\,M)} (a_i \pm b_i) \cdot x^i \, ,$$

$$a(x) \cdot b(x) \ = \sum_{i=0}^{m+M} c_i \cdot x^i \, , \quad c_i = \sum_{j=0}^{i} a_j \cdot b_{i-j} \, .$$

Beispiel: Es gelte $a(x) = x^3 + x + 1$ und $b(x) = x^2 + x + 1$. Im binären Galoisfeld $\Rightarrow$ GF(2) ergibt sich nach den obigen Gleichungen für die Summe, die Differenz und das Produkt der beiden Polynome:

$$s(x) = a(x) + b(x) = x^3 + x^2 \, , \quad d(x) = a(x) - b(x) = x^3 + x^2 = s(x) \, ,$$

$$c(x) = a(x) \cdot b(x) = \sum_{i=0}^{3+2} c_i \cdot x^i \, , \quad c_i = \sum_{j=0}^{i} a_j \cdot b_{i-j} \, .$$

Mit $a_0 = a_1 = a_3 = b_0 = b_1 = b_2 = 1$ und $a_2 = a_4 = a_5 = b_3 = b_4 = b_5 = 0$ erhält man:

$$\begin{aligned}
c_0 &= a_0 \cdot b_0 = 1 \cdot 1 = 1 \, , \\
c_1 &= a_0 \cdot b_1 + a_1 \cdot b_0 = 1 \cdot 1 + 1 \cdot 1 = 0 \, , \\
c_2 &= a_0 \cdot b_2 + a_1 \cdot b_1 + a_2 \cdot b_0 = 1 \cdot 1 + 1 \cdot 1 + 0 \cdot 1 = 0 \, , \\
c_3 &= a_0 \cdot b_3 + a_1 \cdot b_2 + a_2 \cdot b_1 + a_3 \cdot b_0 = 1 \cdot 0 + 1 \cdot 1 + 0 \cdot 1 + 1 \cdot 1 = 0 \, , \\
c_4 &= a_0 \cdot b_4 + a_1 \cdot b_3 + \ldots + a_4 \cdot b_0 = 1 \cdot 0 + 1 \cdot 0 + 0 \cdot 1 + 1 \cdot 1 + 0 \cdot 1 = 1 \, , \\
c_5 &= a_0 \cdot b_5 + a_1 \cdot b_4 + \ldots + a_5 \cdot b_0 = 1 \cdot 0 + 1 \cdot 0 + 0 \cdot 0 + 1 \cdot 1 + 0 \cdot 1 + 0 \cdot 1 = 1 \\
\Rightarrow \quad c(x) &= x^5 + x^4 + 1 \, .
\end{aligned}$$

Im Galoisfeld GF(3) erhält man aufgrund der Modulo–3–Operationen andere Ergebnisse:

$$\begin{aligned}
s(x) &= (x^3 + x + 1) + (x^2 + x + 1) = x^3 + x^2 + 2x + 2 \, , \\
d(x) &= x^3 + x + 1) - (x^2 + x + 1) = x^3 + 2x^2 \, , \\
c(x) &= x^3 + x + 1) \cdot (x^2 + x + 1) = x^5 + x^4 + 2x^3 + 2x^2 + 2x + 1 \, .
\end{aligned}$$

Polynome über einem endlichen Körper (2)

Definition: Ein Polynom $a(x)$ bezeichnet man als **reduzibel** (englisch: *reducible*), wenn es als Produkt zweier Polynome $p(x)$ und $q(x)$ mit jeweils niedrigerem Grad dargestellt werden kann:

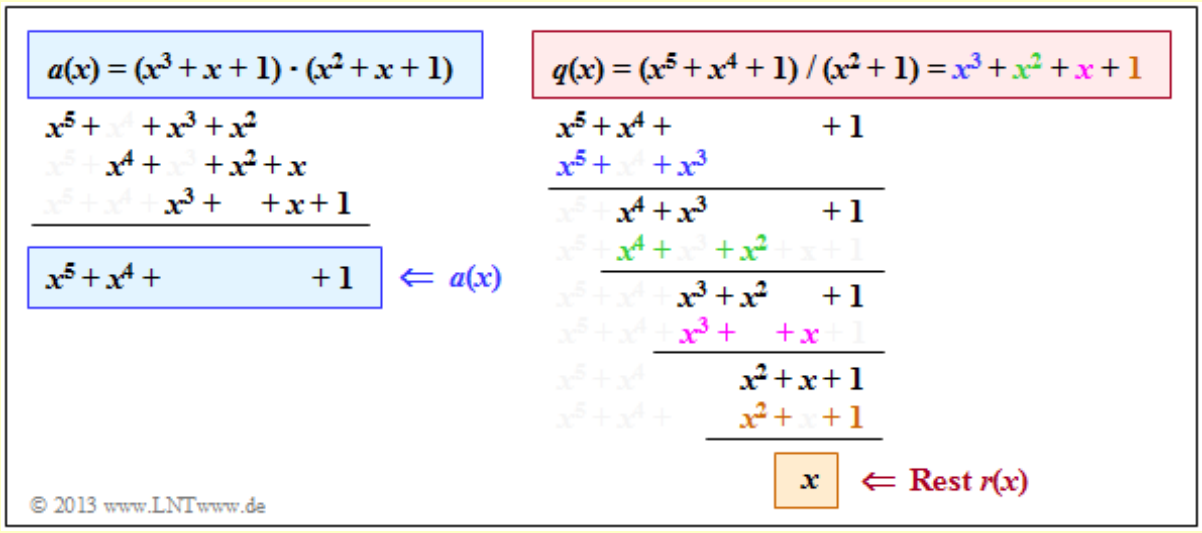
$$a(x) = p(x) \cdot q(x) \,.$$

Ist diese Faktorisierung nicht möglich, das heißt, wenn

$$a(x) = p(x) \cdot q(x) + r(x) \,, \quad r(x) \neq 0$$

gilt, so spricht man von einem **irreduziblen** (englisch: *irreducible* oder *prime*) Polynom.

Beispiel: Es gelte $b(x) = x^3 + x + 1$, $p_1(x) = x^2 + x + 1$ und $p_2(x) = x^2 + 1$. Die Grafik verdeutlicht links die Modulo–2–Multiplikation $a(x) = b(x) \cdot p_1(x)$. Das Ergebnis ist $a(x) = x^5 + x^4 + 1$.



Im rechten Teil der obigen Grafik ist die Modulo–2–Division $q(x) = a(x)/p_2(x)$ mit dem Ergebnis $q(x) = x^3 + x^2 + x + 1$ dargestellt. Es verbleibt der Rest $r(x) = x$. Allein nach dieser Rechnung könnte $a(x) = x^5 + x^4 + 1$ durchaus ein irreduzibles Polynom sein.

Der Nachweis, dass das Polynom $a(x) = x^5 + x^4 + 1$ tatsächlich irreduzibel ist, wäre allerdings erst dann erbracht, wenn $a(x)/p(x)$ für alle

$$p(x) = \sum_{i=0}^{m} a_i \cdot x^i = a_m \cdot x^m + a_{m-1} \cdot x^{m-1} + \ \ldots \ + a_2 \cdot x^2 + a_1 \cdot x + a_0$$

einen Rest $r(x) \neq 0$ liefert. Dies würde im vorliegenden Beispiel (nahezu) $2^5 = 32$ Divisionen erfordern.

Aufgrund unserer linken Berechnung können wir hier sofort erkennen, dass $a(x)$ mit Sicherheit kein irreduzibles Polynom ist, da zum Beispiel $a(x) = x^5 + x^4 + 1$ dividiert durch $p_1(x) = x^2 + x + 1$ das Polynom $b(x) = x^3 + x + 1$ ohne Rest ergibt.

Verallgemeinerte Definition eines Erweiterungskörpers

Wir gehen von folgenden Voraussetzungen aus:

- einem Galoisfeld GF(P), wobei P eine Primzahl angibt,
- einem irreduziblen Polynom $p(x)$ über GF(P) mit dem Grad m:

$$p(x) = a_m \cdot x^m + a_{m-1} \cdot x^{m-1} + \ \ldots \ + a_2 \cdot x^2 + a_1 \cdot x + a_0\,, \quad a_i \in \mathrm{G}(P)\,, \ a_m \neq 0\,.$$

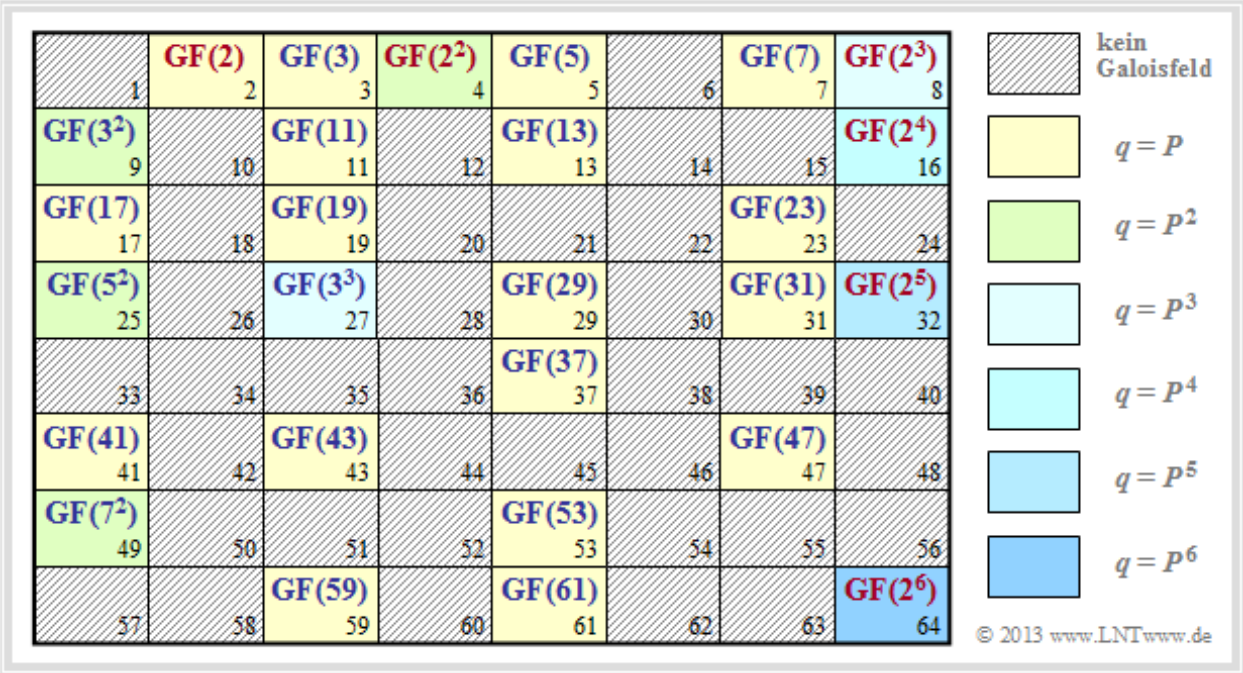
Mit den genannten Voraussetzungen gilt allgemein:

Definition: Es sei P eine Primzahl, m ganzzahlig, $p(x)$ ein irreduzibles Polynom vom Grad m und es gelte $p(\alpha) = 0$. Ein **Erweiterungskörper** lässt sich dann wie folgt beschreiben.

$$\mathrm{GF}(P^m) = \left\{ k_{m-1} \cdot \alpha^{m-1} + \ \ldots \ + k_1 \cdot \alpha + k_0 \,\middle|\, k_i \in \mathrm{GF}(P) = \{0, 1, \ \ldots, P-1\} \right\}$$

Die Addition und Multiplikation in diesem Erweiterungskörper entspricht dann der Polynomaddition und Polynommultiplikation modulo $p(\alpha)$.

Ein Galoisfeld GF(q) mit q Elementen lässt sich also immer dann angeben, wenn die Elementenanzahl in der Form $q = P^m$ geschrieben werden kann (P kennzeichnet eine Primzahl, m sei ganzzahlig).



Die Grafik zeigt, für welche q–Werte sich jeweils ein Galoisfeld konstruieren lässt. Für die schraffiert eingezeichneten q–Werte ist kein endlicher Körper angebbar. Weiter ist anzumerken:

- Die gelb hinterlegten Positionen $q = P \ \Rightarrow \ m = 1$ markieren Zahlenmengen $\{0, 1, \ \ldots \ , q-1\}$ mit Galoiseigenschaften, siehe **Kapitel 2.1.**
- Die anderen farblich hinterlegten Positionen markieren Erweiterungskörper mit $q = P^m$, $m \geq 2$. Für $q \leq 64$ basieren diese auf den Primzahlen 2, 3, 5 und 7.
- Mit roter Schrift hervorgehoben sind binäre Körper $\Rightarrow \ q = 2^m$, $m \geq 1$, die auf der nächsten Seite noch genauer betrachtet werden. Alle anderen Erweiterungskörper sind blau beschriftet.

Binäre Erweiterungskörper (1)

Im Folgenden betrachten wir binäre Erweiterungskörper mit

$$q = 2^m \ (m \geq 2) \quad \Rightarrow \quad q = 4, 8, 16, 32, 64, \ldots$$

Elementen. In der Tabelle sind für $2 \leq m \leq 6$ alle irreduziblen Polynome des Galoisfeldes GF(2) angegeben. Die Polynome in Spalte 2 und 3 sind nicht nur irreduzibel, sondern auch primitiv.

Grad m	irreduzible Polynome, gleichzeitig primitiv	irreduzibel & primitiv, reziprok zu Spalte 2	irreduzible Polynome, jedoch nicht primitiv
2	x^2+x+1		
3	x^3+x+1	x^3+x^2+1	
4	x^4+x+1	x^4+x^3+1	$x^4+x^3+x^2+x+1$
5	x^5+x^2+1 $x^5+x^3+x^2+x+1$ $x^5+x^4+x^2+x+1$	x^5+x^3+1 $x^5+x^4+x^3+x^2+1$ $x^5+x^4+x^3+x+1$	
6	x^6+x+1 $x^6+x^4+x^3+x+1$ $x^6+x^5+x^2+x+1$	x^6+x^5+1 $x^6+x^5+x^3+x^2+1$ $x^6+x^5+x^4+x+1$	x^6+x^3+1 $x^6+x^4+x^2+x+1$ $x^6+x^5+x^4+x^2+1$

Bevor wir uns der Definition eines primitiven Polynoms zuwenden, sollen zunächst die Besonderheiten primitiver Elemente am Beispiel von

$$\mathrm{GF}(q) = \{ z_0 = 0,\ z_1 = 1,\ \ldots, z_{q-1} \}$$

genannt werden. Das Element $z_i = \beta$ wird dann als primitiv bezeichnet,

- wenn die Potenz β^i modulo q zum ersten Mal für $i = q - 1$ das Ergebnis „1" liefert,
- so dass β^i für $1 \leq i \leq q - 1$ genau die Elemente $z_1, \ldots, z_{q-1}$ liefert, also alle Elemente von GF(q) mit Ausnahme des Nullelementes $z_0 = 0$.

Beispiel: Von der Zahlenmenge $Z_5 = \{0, 1, 2, 3, 4\}$ sind „2" und „3" wegen

$$2^1 = 2,\ 2^2 = 4,\ 2^3 = 8 \bmod 5 = 3,\ 2^4 = 16 \bmod 5 = 1,$$
$$3^1 = 3,\ 3^2 = 9 \bmod 5 = 4,\ 3^3 = 27 \bmod 5 = 2,\ 3^4 = 81 \bmod 5 = 1$$

primitive Elemente. Dagegen ist „4" kein primitives Element, weil bereits $4^2 = 1$ ist:

$$4^1 = 4,\ 4^2 = 16 \bmod 5 = 1,\ 4^3 = 64 \bmod 5 = 4,\ 4^4 = 256 \bmod 5 = 1.$$

Binäre Erweiterungskörper (2)

Definition: Ein irreduzibles Polynom bezeichnet man gleichzeitig als ein **primitives Polynom**, wenn die Wurzel α bezüglich des Polynoms $p(x)$ ein primitives Element von GF(q) ist. Dann gilt

$$\mathrm{GF}(q) = \{\, \alpha^{-\infty} = 0,\ \alpha^0 = 1,\ \alpha,\ \alpha^2,\ \ldots,\ \alpha^{q-2} \,\}.$$

Alle in Spalte 2 der **obigen Tabelle** angegebenen Polynome sind sowohl irreduzibel als auch primitiv. Ist $p_1(x)$ ein primitives Polynom, so ist auch das dazu reziproke Polynom

$$p_2(x) = x^m \cdot p_1(x^{-1})$$

primitiv. Alle Polynome in Spalte 3 sind reziprok zum Polynom in Spalte 2. Beispielsweise gilt für $m = 3$:

$$p_1(x) = x^3 + x + 1 \quad \Rightarrow \quad p_2(x) = x^3 \cdot [x^{-3} + x^{-1} + 1] = x^3 + x^2 + 1\,.$$

Die irreduziblen Polynome der Spalte 4 sind dagegen nicht primitiv; sie spielen nur eine untergeordnete Rolle zur Beschreibung von Fehlerkorrekturverfahren.

Hinweis: Primitive Polynome liefern auch die Grundlage für **Pseudo–Noise–Generatoren.**

Beispiel: Zur Verdeutlichung dieser Aussagen betrachten wir beispielhaft

- das Galoisfeld GF(2^3) = GF(8), sowie
- das Polynom $p(x) = x^3 + x + 1$.

Aus der Bedingung $p(\alpha) = 0$ erhält man in GF(2^3) weiter:

$$\alpha^3 + \alpha + 1 = 0 \quad \Rightarrow \quad \alpha^3 = \alpha + 1\,,$$

und damit für die Potenzen α^i der Wurzel für $i \ge 4$:

$$\begin{aligned}
\alpha^4 &= \alpha \cdot \alpha^3 = \alpha \cdot (\alpha + 1) = \alpha^2 + \alpha\,,\\
\alpha^5 &= \alpha^2 \cdot \alpha^3 = \alpha^2 \cdot (\alpha + 1) = \alpha^3 + \alpha^2 = \alpha^2 + \alpha + 1\,,\\
\alpha^6 &= \alpha^3 \cdot \alpha^3 = (\alpha + 1) \cdot (\alpha + 1) = \alpha^2 + \alpha + \alpha + 1 = \alpha^2 + 1\,,\\
\alpha^7 &= \alpha^4 \cdot \alpha^3 = (\alpha^2 + \alpha) \cdot (\alpha + 1) = \alpha^3 + \alpha^2 + \alpha^2 + \alpha = \alpha + 1 + \alpha = 1 = \alpha^0\,.
\end{aligned}$$

Binäre Erweiterungskörper (3)

Die Elemente z_0, z_1, ... , z_7 des Galoisfeldes GF(2^3) lassen sich entsprechend der nebenstehenden Tabelle wie folgt darstellen:

- als Potenzen von α (Exponentendarstellung),
- als Polynome der Form $k_2 \cdot \alpha^2 + k_1 \cdot \alpha + k_0$ mit binären Koeffizienten k_2, k_1, k_0 (jeweils 0 oder 1) ,
- als Vektoren der Koeffizienten (k_2, k_1, k_0).

	Potenzen von α	Polynome in α	Vektoren $k_2\,k_1\,k_0$
z_0	$\alpha^{-\infty}=0$	0	0 0 0
z_1	$\alpha^0=1$	1	0 0 1
z_2	α^1	α	0 1 0
z_3	α^2	α^2	1 0 0
z_4	α^3	$\alpha+1$	0 1 1
z_5	α^4	$\alpha^2+\alpha$	1 1 0
z_6	α^5	$\alpha^2+\alpha+1$	1 1 1
z_7	α^6	α^2+1	1 0 1

Für die Addition (oder Subtraktion) zweier Elemente eignen sich die Polynom– und die Vektordarstellung gleichermaßen, wobei die Komponenten modulo 2 zu addieren sind, zum Beispiel:

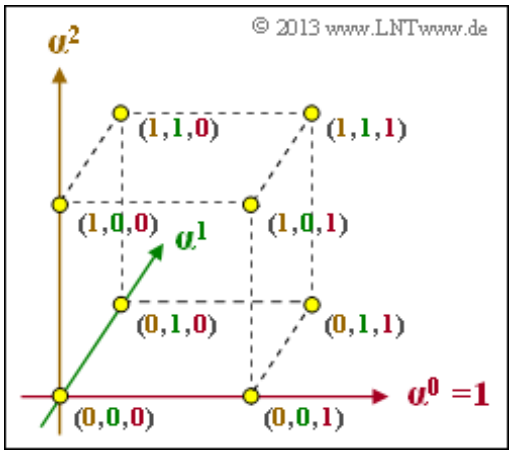
$$z_5 + z_7 = (\alpha^2 + \alpha) + (\alpha^2 + 1) = \alpha + 1 = \alpha^3 = z_4\,,$$
$$\text{oder } z_5 + z_7 = (110) + (101) = (011) = z_4\,,$$
$$z_1 + z_2 + z_3 = (001) + (010) + (100) = (111) = z_6\,.$$

Für Multiplikationen ist die Exponentendarstellung besser geeignet, wie die folgenden Beispiele zeigen:

$$z_3 \cdot z_4 = \alpha^2 \cdot \alpha^3 = \alpha^{2+3} = \alpha^5 = z_6\,,$$
$$z_0 \cdot z_5 = \alpha^{-\infty} \cdot \alpha^4 = \alpha^{-\infty} = z_0\,,$$
$$z_5 \cdot z_7 = \alpha^4 \cdot \alpha^6 = \alpha^{10} = \alpha^7 \cdot \alpha^3 = 1 \cdot \alpha^3 = z_4\,.$$

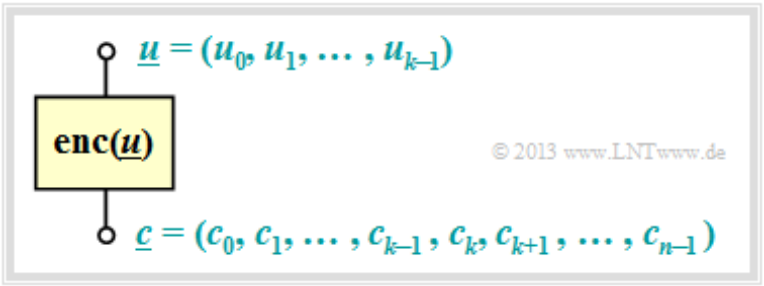
Man erkennt, dass sich hierbei die Exponenten modulo (q – 1) ergeben; im Beispiel modulo 7.

Die Grafik zeigt den endlichen Erweiterungskörper GF(2^3) in einer 3D–Darstellung, wobei die Achsen mit $\alpha^0 = 1$, α^1 und α^2 bezeichnet sind. Die $2^3 = 8$ Punkte im dreidimensionalen Raum sind mit den Koeffizientenvektoren beschriftet, wobei die Zuordnung der einzelnen Koeffizienten k_2, k_1, k_0 zu den Achsen farblich deutlich gemacht ist.



Konstruktion von Reed–Solomon–Codes (1)

Ein **Reed–Solomon–Code** – im Folgenden manchmal auch verkürzt als RS–Code bezeichnet – ist ein **linearer Blockcode,** der einem Informationsblock $\underline{u}$ mit k Symbolen ein entsprechendes Codewort $\underline{c}$ mit $n > k$ Symbolen zuordnet. Diese noch heute vielfach eingesetzten Codes wurden bereits Anfang der 1960er Jahre von **Irving Stoy Reed** und **Gustave Solomon** erfunden.



In **Kapitel 1** wurde der Informationsblock mit $\underline{u} = (u_1, \ \text{...} \ , u_k)$ und das Codewort mit $\underline{x} = (x_1, \ \text{...} \ , x_n)$ bezeichnet. Die Nomenklaturänderung gemäß obiger Grafik wurde vorgenommen, um Verwechslungen mit dem Argument von Polynomen auszuschließen und die Beschreibung der RS–Codes zu vereinfachen.

Alle in **Kapitel 1.4** genannten Eigenschaften linearer zyklischer Blockcodes gelten auch für einen Reed–Solomon–Code. Zusätzlich gilt:

- Konstruktion und Decodierung von RS–Codes basieren auf der Arithmetik eines Galoisfeldes GF(q), wobei wir uns hier auf binäre Erweiterungskörper mit $q = 2^m$ Elementen beschränken:

$$\mathrm{GF}(2^m) = \left\{0, \ \alpha^0, \ \alpha^1, \ \alpha^2, \ \text{...} \ , \alpha^{q-2}\right\}.$$

- Prinzipiell unterschiedlich zum ersten Kapitel ist, dass die Koeffizienten $u_0, \ u_1, \ \text{...} \ , \ u_{k-1}$ nun nicht mehr einzelne Informationsbits (0 oder 1) angeben, sondern ebenfalls Elemente aus GF(2^m) sind. Jedes der m Symbole steht vielmehr für m Bit.

- Bei den Reed–Solomon–Codes ist der Parameter n (Codelänge) gleich der Anzahl der Elemente des Galoisfeldes ohne das Nullwort: $n = q - 1$. Wir verwenden hierzu folgende Nomenklatur:

$$\mathrm{GF}(2^m) \setminus \{0\} = \left\{\alpha^0, \ \alpha^1, \ \alpha^2, \ \text{...} \ , \alpha^{n-1}\right\}.$$

- Die k Koeffizienten $u_i \in \mathrm{GF}(2^m)$ des Informationsblocks $\underline{u}$ ($0 \le i < k$) kann man formal auch durch ein Polynom $u(x)$ ausdrücken. Der Grad des Polynoms ist dabei k–1:

$$u(x) = \sum_{i=0}^{k-1} u_i \cdot x^i = u_0 + u_1 \cdot x + u_2 \cdot x^2 + \ \text{...} \ + u_{k-1} \cdot x^{k-1}, \quad u_i \in \mathrm{GF}(2^m).$$

- Die n Symbole $c_0, \ \text{...} \ , c_{n-1}$ des zugehörigen Codewortes $\underline{c} = (c_0, c_1, \ \text{...} \ , c_{n-1})$ ergeben sich mit diesem Polynom $u(x)$ zu

$$c_0 = u(\alpha^0), \quad c_1 = u(\alpha^1), \ \text{...} \ , \quad c_{n-1} = u(\alpha^{n-1}).$$

- Meist werden die Codesymbole $c_i \in \mathrm{GF}(2^m)$ vor der Übertragung wieder in Binärform ⇒ GF(2) gebracht, wobei dann jedes Symbol durch m Bit dargestellt wird.

Konstruktion von Reed–Solomon–Codes (2)

Fassen wir die Aussagen der letzten Seite kurz zusammen:

Definition: Ein (n, k)–Reed–Solomon–Code für das Galoisfeld GF(2^m) wird festgelegt durch

- die $n = 2^m–1$ Elemente von GF(2^m)\{0} = $\{\alpha^0, \alpha^1, \ ... \ , \alpha^{n-1}\}$, wobei α ein **primitives Element** von GF(2^m) bezeichnet,
- ein an den Informationsblockt $\underline{u}$ angepasstes **Polynom** vom Grad $k–1$ der Form

$$u(x) = \sum_{i=0}^{k-1} u_i \cdot x^i = u_0 + u_1 \cdot x + u_2 \cdot x^2 + \ ... \ + u_{k-1} \cdot x^{k-1}, \quad u_i \in \mathrm{GF}(2^m).$$

Damit lässt sich der **(n, k)–Reed–Solomon–Code** beschreiben als

$$C_{\rm RS} = \left\{ \underline{c} = \big(u(\alpha^0), \, u(\alpha^1), \, ... \, , u(\alpha^{n-1})\big) \;\middle|\; u(x) = \sum_{i=0}^{k-1} u_i \cdot x^i, \;\; u_i \in \mathrm{GF}(2^m) \right\}.$$

Die bisherigen Angaben sollen nun an einem einfachen Beispiel verdeutlicht werden.

Beispiel: Wir gehen von den folgenden Codeparametern aus:

$$k = 2, \;\; n = 3 \quad \Rightarrow \quad \underline{u} = (u_0, u_1), \;\; \underline{c} = (c_0, c_1, c_2),$$
$$q = n + 1 = 4 \quad \Rightarrow \quad \mathrm{GF}(q) = \mathrm{GF}(2^m) \quad \Rightarrow \quad m = 2.$$

Ausgehend von der Bedingungsgleichung $p(\alpha) = \alpha^2 + \alpha + 1 = 0 \;\Rightarrow\; \alpha^2 = \alpha + 1$ erhält man folgende Zuordnungen zwischen der Exponenten–, der Polynom– und der Koeffizientendarstellung von GF(2^2):

Exponentendarstellung	**0**	α^0	α^1	α^2
Polynomdarstellung	**0**	**1**	α	$\alpha+1$
Koeffizientendarstellung	**00**	**01**	**10**	**11**

Der Koeffizientenvektor wird durch das Polynom $u(x) = u_0 + u_i \cdot x$ ausgedrückt. Der Polynomgrad ist $k – 1 = 1$. Für $u_0 = \alpha^1$ und $u_1 = \alpha^2$ erhält man beispielsweise das Polynom $u(x) = \alpha + \alpha^2 \cdot x$ und damit

$$c_0 = u(x = \alpha^0) = u(x = 1) = \alpha + \alpha^2 \cdot 1 = \alpha + (\alpha + 1) = 1,$$
$$c_1 = u(x = \alpha^1) = \alpha + \alpha^2 \cdot \alpha = \alpha + \alpha^3 = \alpha + \alpha^0 = \alpha + 1 = \alpha^2,$$
$$c_2 = u(x = \alpha^2) = \alpha + \alpha^2 \cdot \alpha^2 = \alpha + \alpha^4 = \alpha + \alpha^1 = 0.$$

Daraus ergeben sich folgende Zuordnungen auf Symbol– bzw. Bitebene:

$$\underline{u} = (\alpha^1, \alpha^2) \quad \leftrightarrow \quad \underline{c} = (1, \alpha^2, 0),$$
$$\underline{u}_{\rm bin} = (1, 0, 1, 1) \quad \leftrightarrow \quad \underline{c}_{\rm bin} = (0, 1, 1, 1, 0, 0).$$

Konstruktion von Reed–Solomon–Codes (3)

Die folgende Grafik zeigt die Codetabelle dieses RSC $(3, 2, 2)_4$ genannten Reed–Solomon–Codes. Die Bezeichnung bezieht sich auf die Parameter $n = 3$, $k = 2$, $d_{\rm min} = 2$ und $q = 4$. In den Spalten 1 bis 3 erkennt man den Zusammenhang $\underline{u} \rightarrow u(x) \rightarrow \underline{c}$, in den beiden letzten die Codiervorschrift $\underline{u}_{\rm bin} \leftrightarrow \underline{c}_{\rm bin}$.

Zur Verdeutlichung nochmals der Eintrag für (α^0, α^2):

$$u(x) = u_0 + u_1 \cdot x = \alpha^0 + \alpha^2 \cdot x.$$

Daraus ergeben sich folgende Codesymbole:

$$\begin{aligned} c_0 &= u(x = \alpha^0) = 1 + \alpha^2 \cdot 1 = \\ &= 1 + (1 + \alpha) = \alpha\,, \\ c_1 &= u(x = \alpha^1) = 1 + \alpha^2 \cdot \alpha = \\ &= 1 + (1 + \alpha) \cdot \alpha = 1 + \alpha + \alpha^2 = 0\,, \\ c_2 &= u(x = \alpha^2) = 1 + \alpha^2 \cdot \alpha^2 = \\ &= 1 + \alpha = \alpha^2\,. \end{aligned}$$

$\underline{u}$	$u(x)$	$\underline{c}$	$\underline{u}_{\rm bin}$	$\underline{c}_{\rm bin}$
$0\,,0$	0	$0\,,0\,,0$	0000	000000
$0, \alpha^0$	x	$\alpha^0,\alpha^1,\alpha^2$	0001	011011
$0, \alpha^1$	αx	$\alpha^1,\alpha^2,\alpha^0$	0010	101101
$0, \alpha^2$	$\alpha^2 x$	$\alpha^2,\alpha^0\,\alpha^1$	0011	110110
$\alpha^0, 0$	1	$\alpha^0,\alpha^0,\alpha^0$	0100	010101
α^0, α^0	$1+x$	$0\,,\alpha^2,\alpha^1$	0101	001110
α^0, α^1	$1+\alpha x$	$\alpha^2,\alpha^1,\,0$	0110	111000
α^0, α^2	$1+\alpha^2 x$	$\alpha^1,\,0\,,\alpha^2$	0111	100011
$\alpha^1,\,0$	α	$\alpha^1,\alpha^1,\alpha^1$	1000	101010
α^1, α^0	$\alpha+x$	$\alpha^2,\,0\,,\alpha^0$	1001	110001
α^1, α^1	$\alpha+\alpha x$	$0\,,\alpha^0,\alpha^2$	1010	000111
α^1, α^2	$\alpha+\alpha^2 x$	$\alpha^0,\alpha^2,\,0$	1011	011100
$\alpha^2,\,0$	α^2	$\alpha^2,\alpha^2,\alpha^2$	1100	111111
α^2, α^0	α^2+x	$\alpha^1,\alpha^0,\,0$	1101	100100
α^2, α^1	$\alpha^2+\alpha x$	$\alpha^0,\,0\,,\alpha^1$	1110	010010
α^2, α^2	$\alpha^2+\alpha^2 x$	$0\,,\alpha^1,\alpha^0$	1111	001001

Hinweis: Aus der Elementenmenge $\{0, \alpha^0 = 1, \alpha^1, \alpha^2\}$ sollte nicht geschlossen werden, dass für diesen Code die **3D–Darstellung** mit den Achsen $\alpha^0 = 1$, α^1 und α^2 zutrifft. Aus der Koeffizientendarstellung geht vielmehr eindeutig hervor, dass GF(2^2) ein zweidimensionaler Code ist, wobei die Achsen der **2D–Darstellung** mit $\alpha^0 = 1$ und $\alpha^1 = \alpha$ zu beschriften sind.

Generatormatrix und Prüfmatrix (1)

Da es sich beim Reed–Solomon–Code um einen linearen Blockcode handelt, ist der Zusammenhang zwischen Informationswort $\underline{u}$ und Codewort $\underline{c}$ durch die **Generatormatrix G** gegeben:

$$\underline{c} = \underline{u} \cdot \mathbf{G}.$$

Wie bei jedem linearen (n, k)–Blockcode besteht die Generatormatrix aus k Zeilen und n Spalten. Im Gegensatz zum **Kapitel 1.4** sind nun aber die Elemente der Generatormatrix nicht mehr binär (0 oder 1), sondern entstammen dem Galoisfeld $\mathrm{GF}(2^m) \setminus \{0\}$.

Beispiel: Wir betrachten wie auf der letzten Seite wieder den RSC $(3, 2, 2)_4$, dessen Generatormatrix folgende Form hat:

$$\mathbf{G} = \begin{pmatrix} g_{00} & g_{01} & g_{02} \\ g_{10} & g_{11} & g_{12} \end{pmatrix}, \quad g_{ij} \in \mathrm{GF}(2^2) \setminus \{0\} = \{\alpha^0, \alpha^1, \alpha^2\}.$$

Daneben gilt:

- Die erste Zeile von **G** gibt das Codewort für das Informationswort $\underline{u}_1 = (1, 0)$ an bzw. für die Polynomfunktion $u_1(x) = 1$. Damit erhält man die Matrixelemente der ersten Zeile zu

$$g_{00} = u_1(\alpha^0) = 1, \quad g_{01} = u_1(\alpha^1) = 1, \quad g_{02} = u_1(\alpha^2) = 1.$$

- Die zweite Zeile ist gleich dem Codewort für das Informationswort $\underline{u}_2 = (0, 1) \Rightarrow u_2(x) = x$. Die Matrixelemente der zweiten Zeile lauten somit:

$$g_{10} = u_2(\alpha^0) = \alpha^0 = 1, \quad g_{11} = u_2(\alpha^1) = \alpha, \quad g_{12} = u_2(\alpha^2) = \alpha^2.$$

$$\Rightarrow \quad \mathbf{G} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & \alpha & \alpha^2 \end{pmatrix}.$$

Für das Informationswort $\underline{u} = (u_0, u_1)$ mit den Symbolen $u_0, u_1 \in \{0, \alpha^0, \alpha^1, \alpha^2\}$ erhält man unter Berücksichtigung der beiden Gleichungen $\alpha^2 = \alpha + 1$ sowie $\alpha^3 = \alpha^0 = 1$ wiederum die Codetabelle des RSC $(3, 2, 2)_4$ auf Symbolebene.

$\underline{u}$	$\underline{c}$	$\underline{u}$	$\underline{c}$	$\underline{u}$	$\underline{c}$	$\underline{u}$	$\underline{c}$
$0, 0$	$0, 0, 0$	$\alpha^0, 0$	$\alpha^0, \alpha^0, \alpha^0$	$\alpha^1, 0$	$\alpha^1, \alpha^1, \alpha^1$	$\alpha^2, 0$	$\alpha^2, \alpha^2, \alpha^2$
$0, \alpha^0$	$\alpha^0, \alpha^1, \alpha^2$	α^0, α^0	$0, \alpha^2, \alpha^1$	α^1, α^0	$\alpha^2, 0, \alpha^0$	α^2, α^0	$\alpha^1, \alpha^0, 0$
$0, \alpha^1$	$\alpha^1, \alpha^2, \alpha^0$	α^0, α^1	$\alpha^2, \alpha^1, 0$	α^1, α^1	$0, \alpha^0, \alpha^2$	α^2, α^1	$\alpha^0, 0, \alpha^1$
$0, \alpha^2$	$\alpha^2, \alpha^0\ \alpha^1$	α^0, α^2	$\alpha^1, 0, \alpha^2$	α^1, α^2	$\alpha^0, \alpha^2, 0$	α^2, α^2	$0, \alpha^1, \alpha^0$

Man erhält natürlich mit der Generatormatrix genau die gleiche Codetabelle $\underline{u} \leftrightarrow \underline{c}$ wie nach der Berechnung über die Funktion $u(x)$. Die entsprechende **Codetabelle auf Bitebene** ergibt sich wieder, wenn man die Elemente nicht in Exponentendarstellung angibt, sondern in Koeffizientenform:

$$(0,\ \alpha^0,\ \alpha^1,\ \alpha^2) \quad \Leftrightarrow \quad (00,\ 01,\ 10,\ 11).$$

Generatormatrix und Prüfmatrix (2)

Wir verallgemeinern nun das Ergebnis der letzten Seite für einen Reed–Solomon–Code mit

- der Dimension k (Symbolanzahl pro Informationsblock),
- der Codewortlänge n (Symbolanzahl pro Codewort).

Die **Generatormatrix G** (mit k Zeilen und n Spalten) und die **Prüfmatrix H** ($n - k$ Zeilen, n Spalten) müssen folgende Gleichung erfüllen:

$$\mathbf{G} \cdot \mathbf{H}^{\rm T} = \mathbf{0}.$$

Hierbei bezeichnet **0** eine Nullmatrix (alle Elemente gleich 0) mit k Zeilen und $n - k$ Spalten.

Beispiel: Wir betrachten den RSC $(7, 3, 5)_8$ $\Rightarrow$ Codeparameter $n = 7$, $k = 3$, basierend auf dem Galoisfeld GF($2^3 = 8$) mit der Nebenbedingung $\alpha^3 = \alpha + 1$. Beachten Sie hinsichtlich der Bezeichnung:

- Der dritte Parameter der für Blockcodes üblichen Nomenklatur nennt die freie Distanz $d_{\rm min} = 5$.
- Anders als bei den in **Kapitel 1.3** behandelten binären Codes (SPC, RC, HC) wird bei den Reed–Solomon–Codes noch der Hinweis q zum Galoisfeld hinzugefügt (hier: $q = 8$).

Alle Elemente der Generatormatrix und der Prüfmatrix

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^1 & \alpha^3 & \alpha^5 \end{pmatrix}, \quad \mathbf{H} = \begin{pmatrix} 1 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^1 & \alpha^3 & \alpha^5 \\ 1 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha^1 & \alpha^4 \\ 1 & \alpha^4 & \alpha^1 & \alpha^5 & \alpha^2 & \alpha^6 & \alpha^3 \end{pmatrix}$$

entstammen dem Galoisfeld GF(2^3) \ {0} = {α^0, α^1, α^2, α^3, α^4, α^5, α^6}. Für das Matrixprodukt gilt:

$$\mathbf{G} \cdot \mathbf{H}^{\rm T} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^1 & \alpha^3 & \alpha^5 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 \\ \alpha^2 & \alpha^4 & \alpha^6 & \alpha^1 \\ \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 \\ \alpha^4 & \alpha^1 & \alpha^5 & \alpha^2 \\ \alpha^5 & \alpha^3 & \alpha^1 & \alpha^6 \\ \alpha^6 & \alpha^5 & \alpha^4 & \alpha^3 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Dies soll hier nur für zwei Elemente nachgewiesen werden:

- Erste Zeile, erste Spalte:

$$1 \cdot \left[1 + \alpha^1 + \alpha^2 + \alpha^3 + \alpha^4 + \alpha^5 + \alpha^6\right] =$$
$$= 1 + \alpha + \alpha^2 + (\alpha + 1) + (\alpha^2 + \alpha) + (\alpha^2 + \alpha + 1) + (\alpha^2 + 1) = 0,$$

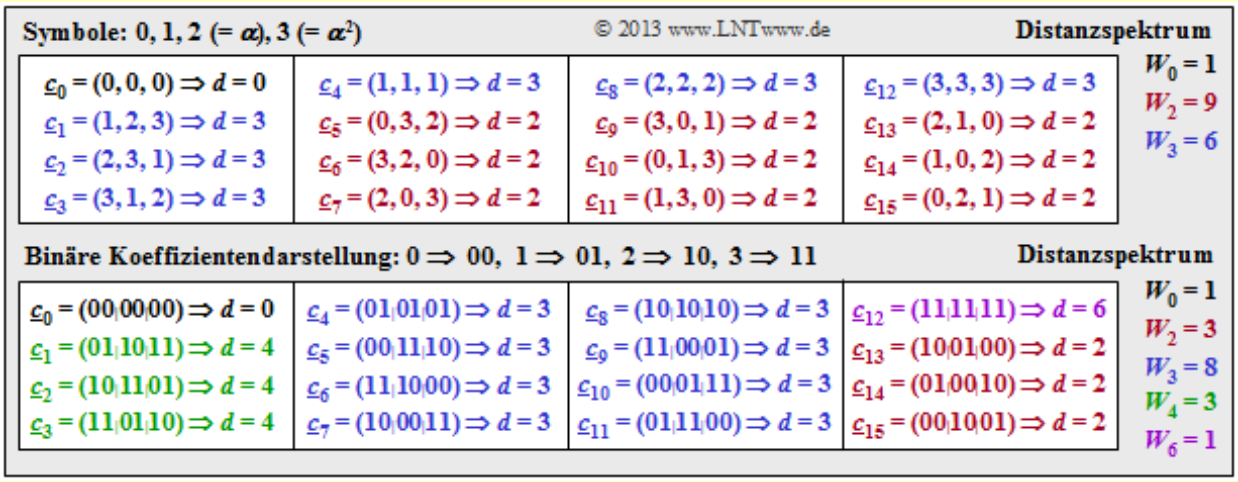
- Letzte Zeile, letzte Spalte:

$$1 \cdot 1 + \alpha^2 \cdot \alpha^4 + \alpha^4 \cdot \alpha^1 + \alpha^6 \cdot \alpha^5 + \alpha^1 \cdot \alpha^2 + \alpha^3 \cdot \alpha^6 + \alpha^5 \cdot \alpha^3 =$$
$$= 1 + \alpha^6 + \alpha^5 + \alpha^{11} + \alpha^3 + \alpha^9 + \alpha^8 =$$
$$= 1 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha^1 = 0.$$

Singleton–Schranke und minimale Distanz

Eine wichtige Kenngröße eines jeden Blockcodes ist die **minimale Distanz** zwischen zwei beliebigen Codeworten $\underline{c}_i$ und $\underline{c}_j$. Die Reed–Solomon–Codes gehören zur Klasse der *linearen* und *zyklischen* Codes. Bei diesen kann man vom Nullwort $\underline{c}_0 = (0\ 0\ \ldots\ 0)$ als Bezugsgröße ausgehen. Aus der Anzahl der Nullen in den anderen Codeworten $\underline{c}_j \neq \underline{c}_0$ lässt sich das **Distanzspektrum** $\{W_i\}$ angeben.

Beispiel: Die Grafik zeigt die Bestimmung des Distanzspektrums für den RSC $(3, 2, 2)_4$. Gegenüber den bisherigen Grafiken sind die Symbole mit 0, 1, 2, 3 anstelle von 0, α^0, α^1, α^2 bezeichnet. Die Distanz d zwischen $\underline{c}_j$ und dem Nullwort $\underline{c}_0$ ist identisch dem Hamming–Gewicht $w_H(\underline{c}_j)$.



Neun der Codeworte unterscheiden sich vom Nullwort in zwei Symbolen und sechs Codeworte in drei Symbolen: $W_2 = 9$, $W_3 = 6$. Es gibt kein einziges Codewort mit nur einer Null. Das heißt: Die minimale Distanz beträgt hier $d_{min} = 2$.

Aus der zweiten Tabelle erkennt man, dass auch für die Binärdarstellung $d_{min} = 2$ gilt.

Dieses empirische Ergebnis soll nun ohne Beweis verallgemeinert werden:

- Die *minimale Distanz* eines jeden (n, k)–Reed–Solomon–Codes beträgt $\boldsymbol{d_{min} = n - k + 1}$. Damit lassen sich $e = d_{min} - 1 = n - k$ Symbolfehler erkennen.
- Bei *fehlerkorrigierenden Codes* wählt man meist ein d_{min} ungeradzahlig $\Rightarrow$ $n - k$ geradzahlig. Bei RS–Codes können dann bis zu $t = (n - k)/2$ Symbolfehler korrigiert werden.
- Die **Singleton–Schranke** besagt, dass für alle linearen Codes $d_{min} \le n - k + 1$ gilt. RS–Codes erreichen die Schranke mit Gleichheit; sie sind **MDS–Codes** (*Maximum Distance Separable*).
- Das **Distanzspektrum** setzt sich zusammen aus $W_0 = 1$ sowie weiteren Gewichtsfaktoren W_i mit $d \le i \le n$, wobei in der folgenden Gleichung d_{min} mit d abgekürzt ist:

$$W_i = \binom{n}{i} \cdot \sum_{j=0}^{i-d} (-1)^j \cdot \binom{i}{j} \cdot \left[q^{i-j-d+1} - 1\right].$$

Codebezeichnung und Coderate

Die übliche Bezeichnung für die Reed–Solomon–Codes ist **RSC $(n, k, d_{\rm min})_q$** mit

- der Länge n des Codes (Symbolanzahl eines Codewortes),
- der Dimension k des Codes (Symbolanzahl eines Informationswortes),
- der minimalen Distanz $d_{\rm min} = n - k + 1$, maximal entsprechend der Singleton–Schranke, und
- der Größe $q = 2^m$ des Galoisfeldes $\Rightarrow$ GF(q).

Alle Elemente u_i des Informationswortes $\underline{u} = (u_0, \text{...}, u_i, \text{...}, u_{k-1})$ und alle Elemente c_i des Codewortes $\underline{c} = (c_0, \text{...}, c_i, \text{...}, c_{n-1})$ sind nicht binäre Symbole und entstammen dem Galoisfeld GF(q).

Für die Realisierung werden diese Symbole stets auch binär dargestellt $\Rightarrow$ u_i, $x_i \in$ GF(2), und man kommt zum äquivalenten binären Code **RSC $(n_{\rm bin}, k_{\rm bin}, d_{\rm min})_2$** mit

- $n_{\rm bin} = n \cdot m$ Bit eines Codewortes,
- $k_{\rm bin} = k \cdot m$ Bit eines Informationswortes.

Die Coderate wird durch diese Maßnahme nicht verändert:

$$R = \frac{k}{n} = \frac{k_{\rm bin}}{n_{\rm bin}} = \frac{k \cdot m}{n \cdot m} = \frac{k}{n}.$$

Ebenso ändert sich durch den Übergang von GF(q) auf GF(2) nichts an der minimalen Distanz $d_{\rm min}$.

Beispiel: Beim RSC $(3, 2, 2)_4$ ist die Coderate $R = k/n = 2/3$ und die minimale Distanz $d_{\rm min} = 2$. Für das **Distanzspektrum** $\{W_i\}$ und die **Gewichtsfunktion** $W(X)$ gilt nach **Kapitel 1.6** (siehe Tabelle):

$$W_0 = 1, \quad W_2 = 9, \quad W_3 = 6 \quad \Rightarrow \quad W(X) = 1 + 9 \cdot X^2 + 6 \cdot X^3.$$

Symbole: 0, 1, 2 (= α), 3 (= α^2) — © 2013 www.LNTwww.de — **Distanzspektrum**

$\underline{c}_0 = (0,0,0) \Rightarrow d = 0$	$\underline{c}_4 = (1,1,1) \Rightarrow d = 3$	$\underline{c}_8 = (2,2,2) \Rightarrow d = 3$	$\underline{c}_{12} = (3,3,3) \Rightarrow d = 3$	$W_0 = 1$
$\underline{c}_1 = (1,2,3) \Rightarrow d = 3$	$\underline{c}_5 = (0,3,2) \Rightarrow d = 2$	$\underline{c}_9 = (3,0,1) \Rightarrow d = 2$	$\underline{c}_{13} = (2,1,0) \Rightarrow d = 2$	$W_2 = 9$
$\underline{c}_2 = (2,3,1) \Rightarrow d = 3$	$\underline{c}_6 = (3,2,0) \Rightarrow d = 2$	$\underline{c}_{10} = (0,1,3) \Rightarrow d = 2$	$\underline{c}_{14} = (1,0,2) \Rightarrow d = 2$	$W_3 = 6$
$\underline{c}_3 = (3,1,2) \Rightarrow d = 3$	$\underline{c}_7 = (2,0,3) \Rightarrow d = 2$	$\underline{c}_{11} = (1,3,0) \Rightarrow d = 2$	$\underline{c}_{15} = (0,2,1) \Rightarrow d = 2$	

Binäre Koeffizientendarstellung: 0 $\Rightarrow$ 00, 1 $\Rightarrow$ 01, 2 $\Rightarrow$ 10, 3 $\Rightarrow$ 11 — **Distanzspektrum**

$\underline{c}_0 = (00,00,00) \Rightarrow d = 0$	$\underline{c}_4 = (01,01,01) \Rightarrow d = 3$	$\underline{c}_8 = (10,10,10) \Rightarrow d = 3$	$\underline{c}_{12} = (11,11,11) \Rightarrow d = 6$	$W_0 = 1$
$\underline{c}_1 = (01,10,11) \Rightarrow d = 4$	$\underline{c}_5 = (00,11,10) \Rightarrow d = 3$	$\underline{c}_9 = (11,00,01) \Rightarrow d = 3$	$\underline{c}_{13} = (10,01,00) \Rightarrow d = 2$	$W_2 = 3$
$\underline{c}_2 = (10,11,01) \Rightarrow d = 4$	$\underline{c}_6 = (11,10,00) \Rightarrow d = 3$	$\underline{c}_{10} = (00,01,11) \Rightarrow d = 3$	$\underline{c}_{14} = (01,00,10) \Rightarrow d = 2$	$W_3 = 8$
$\underline{c}_3 = (11,01,10) \Rightarrow d = 4$	$\underline{c}_7 = (10,00,11) \Rightarrow d = 3$	$\underline{c}_{11} = (01,11,00) \Rightarrow d = 3$	$\underline{c}_{15} = (00,10,01) \Rightarrow d = 2$	$W_4 = 3$
				$W_6 = 1$

Die binäre Repräsentation des Codes RSC $(3, 2, 2)_4$ führt zum RSC $(6, 4, 2)_2$, ebenfalls mit der Rate $R = 4/6 = 2/3$ und der Minimaldistanz $d_{\rm min} = 2$. Es ergibt sich allerdings ein anderes Distanzspektrum:

$$W_0 = 1, \quad W_2 = 3, \quad W_3 = 8, \quad W_4 = 3, \quad W_6 = 1$$
$$\Rightarrow \quad W(X) = 1 + 3 \cdot X^2 + 8 \cdot X^3 + 4 \cdot X^4 + X^6.$$

Bedeutung der Reed–Solomon–Codes

Anhand des hier oft beispielhaft betrachteten RSC $(3, 2, 2)_4$ konnten wir viele Eigenschaften der Reed–Solomon–Codes in überschaubarem Rahmen kennenlernen. Praxisrelevant ist dieser Code nicht, da wegen $d_{\rm min} = 2$ kein einziger Fehler korrigiert und auch nur ein einziger Fehler erkannt werden kann. Schon der nächstgrößere Code RSC $(7, 3, 5)_8$, der bis zu $t = 2$ Fehler korrigieren kann, weist bereits eine Codetabelle mit $8^3 = 512$ Einträgen auf und ist zu Demonstrationszwecken weniger gut geeignet.

In der Praxis werden meist größere RS–Codes eingesetzt, zum Beispiel der RSC $(255, 223, 33)_{256}$ mit den folgenden Eigenschaften:

- Der Code basiert auf dem Galoisfeld GF(2^8). Jedes Symbol entspricht somit einem Byte. Die Coderate ist mit $R = 0.8745$ relativ groß.
- Trotz dieser großen Coderate (geringen Redundanz) können mit diesem Code bis zu $e = 32$ Fehler innerhalb eines Blocks aus 255 Symbolen erkannt und $t = 16$ Fehler korrigiert werden.
- Die Codetabelle würde allerdings $2^{8 \cdot 223} = 2^{1784} \approx 10^{537}$ Einträge aufweisen und wird deshalb wahrscheinlich auch von niemanden tatsächlich erstellt.

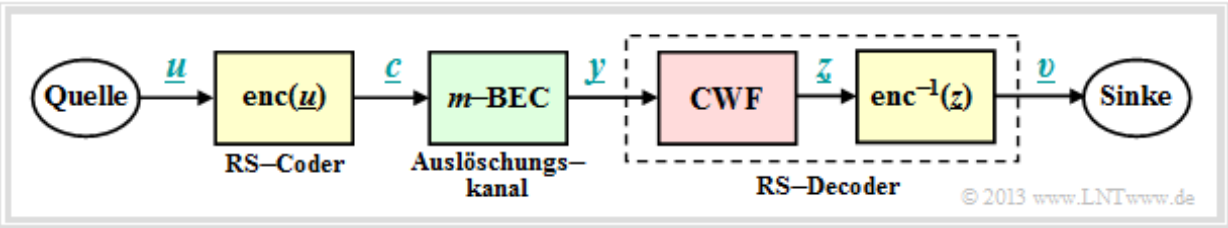
Der große Vorteil der Reed–Solomon–Codes (und einer ganzen Reihe davon abgeleiteter weiterer Codes) ist zum einen, dass sie analytisch geschlossen konstruiert werden können, zum anderen ihre große Flexibilität hinsichtlich der Codeparameter. Meist geht man wie folgt vor:

- Man gibt die Korrekturfähigkeit in Form des Parameters t vor. Daraus ergibt sich die minimale Distanz $d_{\rm min} = 2t + 1$ und die Differenz $n - k = 2t$ entsprechend der Singleton–Schranke. Einen besseren Wert gibt es nicht.
- Ein weiterer Entwurfsparameter ist die Coderate $R = k/n$, wobei die Codewortlänge $n = 2^m - 1$ nicht völlig frei wählbar ist. Durch Erweiterung, Verkürzung und Punktierung – siehe **Aufgabe Z1.9** – kann die Vielzahl an möglichen Codes weiter vergrößert werden.
- Bei Reed–Solomon–Codes ist die Gewichtsverteilung exakt bekannt und es ist eine Anpassung an die Fehlerstruktur des Kanals möglich. Sie sind insbesondere für Bündelfehlerkanäle gut geeignet, die bei mobilen Funksystemen aufgrund von temporären Abschattungen häufig vorliegen.
- Im Falle statistisch unabhängiger Fehler sind BCH–Codes (von Bose–Chaudhuri–Hocquenghem) besser geeignet. Diese sind eng verwandt mit den RS–Codes, allerdings erfüllen sie nicht immer das Singleton–Kriterium. Eine ausführliche Beschreibung finden Sie in **[Fri96].**
- Die Decodierung nach dem BDD–Prinzip (*Bounded Distance Decoding*) kann rechentechnisch sehr einfach erfolgen, zum Beispiel mit dem **Berlekamp–Massey–Algorithmus.** Zudem kann im Decoder ohne wesentlichen Mehraufwand auch *Soft–Decision–Information* verarbeitet werden.

Blockschaltbild und Voraussetzungen zu Kapitel 2.4

Im **Kapitel 1.5** wurde für die binären Blockcodes gezeigt, welche Berechnungen der Decoder ausführen muss, um aus einem unvollständigen Empfangswort $\underline{y}$ das gesendete Codewort $\underline{x}$ bestmöglich decodieren zu können. Zugrunde gelegt war dabei das **BEC–Kanalmodell** (*Binary Erasure Channel*), das ein unsicheres Bit als *Erasure* E („Auslöschung") markiert.

Im Gegensatz zu **BSC** (*Binary Symmetric Channel*) und **AWGN** (*Additive White Gaussian Noise*) wurden hier Bitfehler ($y_i \neq x_i$) ausgeschlossen. Jedes Bit eines Empfangswortes stimmt also mit dem entsprechenden Bit des Codewortes überein ($y_i = x_i$) oder ist bereits als Auslöschung markiert (y_i = E).



Die Grafik zeigt das Blockschaltbild, das sich von dem **Modell** in Kapitel 1.5 geringfügig unterscheidet:

- Da Reed–Solomon–Codes lineare Blockcodes sind, stehen Informationswort $\underline{u}$ und Codewort $\underline{c}$ über die Generatormatrix **G** und die folgende Gleichung in Zusammenhang:

$$\underline{c} = \text{enc}(\underline{u}) = \underline{u} \cdot \mathbf{G} \quad \text{mit} \quad \underline{u} = (u_0, u_1, \ldots, u_i, \ldots, u_{k-1}), \quad \underline{c} = (c_0, c_1, \ldots, c_i, \ldots, c_{n-1}).$$

- Für die einzelnen Symbole von Informations– und Codewort gilt bei Reed–Solomon–Codierung:

$$u_i \in \text{GF}(q), \quad c_i \in \text{GF}(q) \quad \text{mit} \quad q = n + 1 = 2^m \quad \Rightarrow \quad n = 2^m - 1.$$

  Jedes Codesymbol c_i wird somit mit $m \geq 2$ Binärsymbolen (Bit) dargestellt. Zum Vergleich: Für die binären Blockcodes gilt $q = 2$, $m = 1$ und die Codewortlänge n ist frei wählbar.

- Bei Codierung auf Symbolebene muss das BEC–Modell zum m–BEC–Modell erweitert werden. Mit der Wahrscheinlichkeit $\lambda_m \approx m \cdot \lambda$ wird ein Codesymbol c_i ausgelöscht (y_i = E) und es gilt $\Pr(y_i = c_i) = 1 - \lambda_m$. Näheres zur Umrechnung der beiden Modelle finden Sie in **Aufgabe Z2.11.**

Im Folgenden beschäftigen wir uns ausschließlich mit dem Block *Codewortfinder* (CWF), der aus dem Empfangsvektor $\underline{y}$ den Vektor $\underline{z} \in C_{\text{RS}}$ gewinnt:

- Falls die Anzahl e der Auslöschungen in Vektor $\underline{y}$ hinreichend klein ist, lässt sich das gesamte Codewort mit Sicherheit ($\underline{z} = \underline{c}$) finden.
- Sind zuviele Symbole des Empfangswortes $\underline{y}$ ausgelöscht, meldet der Decoder, dass dieses Wort nicht decodierbar ist. Eventuell wird dann die Codesequenz noch einmal gesendet.

Beim Auslöschungskanal (m–BEC) ist also im Gegensatz zum m–BSC, der im Kapitel 2.5 Anwendung findet, eine Fehlentscheidung ($\underline{z} \neq \underline{c}$) ausgeschlossen ⇒ Blockfehlerwahrscheinlichkeit $\Pr(\underline{z} \neq \underline{c}) = 0$ ⇒ $\Pr(\underline{v} \neq \underline{u}) = 0$. Das rekonstruierte Informationswort ergibt sich gemäß dem Blockschaltbild (gelbe Hinterlegung) zu $\underline{v} = \text{enc}^{-1}(\underline{z})$. Mit der Generatormatrix **G** kann hierfür auch geschrieben werden:

$$\underline{c} = \underline{u} \cdot \mathbf{G} \quad \Rightarrow \quad \underline{z} = \underline{v} \cdot \mathbf{G} \quad \Rightarrow \quad \underline{v} = \underline{z} \cdot \mathbf{G}^{\text{T}}.$$

Vorgehensweise am Beispiel des RSC $(7, 3, 5)_8$

Um die RS–Decodierung beim Auslöschungskanal so einfach wie möglich darstellen zu können, gehen wir von einer konkreten Aufgabenstellung aus:

- Verwendet wird ein Reed–Solomon–Code mit den Parametern $n = 7$, $k = 3$ und $q = 2^3 = 8$. Allgemein gilt für das Informationswort $\underline{u}$, das Codewort $\underline{c}$ und die Prüfmatrix $\mathbf{H}$:

$$\underline{u} = (u_0, u_1, u_2), \ \underline{c} = (c_0, c_1, c_2, c_3, c_4, c_5, c_6), \ \ u_i, c_i \in \mathrm{GF}(2^3) = \{0, 1, \alpha, \alpha^2, ..., \alpha^6\},$$

$$\mathbf{H} = \begin{pmatrix} 1 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^1 & \alpha^3 & \alpha^5 \\ 1 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha^1 & \alpha^4 \\ 1 & \alpha^4 & \alpha^1 & \alpha^5 & \alpha^2 & \alpha^6 & \alpha^3 \end{pmatrix}.$$

- Der Empfangsvektor wird mit $\underline{y} = (\alpha^1, 1, \mathrm{E}, \mathrm{E}, \alpha^2, \mathrm{E}, \alpha^5)$ vorgegeben. Da der Auslöschungskanal keine Fehler produziert, sind dem Decoder vier der Codesymbole bekannt:

$$c_0 = \alpha^1, \ \ c_1 = 1, \ \ c_4 = \alpha^2, \ \ c_6 = \alpha^5.$$

- Es ist offensichtlich, dass der Block „Codewortfinder" – im Blockschaltbild mit CWF bezeichnet – einen Vektor der Form $\underline{z} = (c_0, c_1, z_2, z_3, c_4, z_5, c_6)$ liefern soll mit $z_2, z_3, z_5 \in \mathrm{GF}(2^3)$.

- Da das vom Decoder gefundene Codewort $\underline{z}$ aber auch ein gültiges Reed–Solomon–Codewort sein soll $\Rightarrow$ $\underline{z} \in C_{\rm RS}$, muss entsprechend den Ausführungen in **Kapitel 2.3** gelten:

$$\mathbf{H} \cdot \underline{z}^{\rm T} = \underline{0}^{\rm T} \ \ \Rightarrow \ \ \begin{pmatrix} 1 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^1 & \alpha^3 & \alpha^5 \\ 1 & \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 & \alpha^1 & \alpha^4 \\ 1 & \alpha^4 & \alpha^1 & \alpha^5 & \alpha^2 & \alpha^6 & \alpha^3 \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ z_2 \\ z_3 \\ c_4 \\ z_5 \\ c_6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

- Daraus ergeben sich vier Gleichungen für die Unbekannten z_2, z_3, z_5. Bei eindeutiger Lösung – und nur bei einer solchen – ist die Decodierung erfolgreich und man kann dann mit Sicherheit sagen, dass tatsächlich $\underline{c} = \underline{z}$ gesendet wurde.

Die Beschreibung wird auf der nächsten Seite fortgesetzt.

Lösung der Matrixgleichungen am Beispiel des RSC $(7, 3, 5)_8$

Gefunden werden muss also das zulässige Codewort $\underline{z}$, das die Bestimmungsgleichung $\mathbf{H} \cdot \underline{z}^{\rm T} = \underline{0}^{\rm T}$ erfüllt. Zweckmäßigerweise spalten wir dazu den Vektor $\underline{z}$ in zwei Teilvektoren auf, nämlich in

- den Vektor $\underline{z}_{\rm E} = (z_2, z_3, z_5)$ der ausgelöschten Symbole (Index E für *Erasures*),
- den Vektor $\underline{z}_{\rm K} = (c_0, c_1, c_4, c_6)$ der bekannten Symbole (Index K für *Korrekt*).

Mit den zugehörigen Teilmatrizen (jeweils mit $n - k = 4$ Zeilen)

$$\mathbf{H}_{\rm E} = \begin{pmatrix} \alpha^2 & \alpha^3 & \alpha^5 \\ \alpha^4 & \alpha^6 & \alpha^3 \\ \alpha^6 & \alpha^2 & \alpha^1 \\ \alpha^1 & \alpha^5 & \alpha^6 \end{pmatrix}, \quad \mathbf{H}_{\rm K} \begin{pmatrix} 1 & \alpha^1 & \alpha^4 & \alpha^6 \\ 1 & \alpha^2 & \alpha^1 & \alpha^5 \\ 1 & \alpha^3 & \alpha^5 & \alpha^4 \\ 1 & \alpha^4 & \alpha^2 & \alpha^3 \end{pmatrix}$$

lautet somit die Bestimmungsgleichung:

$$\mathbf{H}_{\rm E} \cdot \underline{z}_{\rm E}^{\rm T} + \mathbf{H}_{\rm K} \cdot \underline{z}_{\rm K}^{\rm T} = \underline{0}^{\rm T} \quad \Rightarrow \quad \mathbf{H}_{\rm E} \cdot \underline{z}_{\rm E}^{\rm T} = -\mathbf{H}_{\rm K} \cdot \underline{z}_{\rm K}^{\rm T}.$$

Da für alle Elemente $z_i \in {\rm GF}(2^m)$ die **additive Inverse** ${\rm Inv_A}(z_i)$ $(= -z_i) = z_i$ ist, gilt in gleicher Weise

$$\mathbf{H}_{\rm E} \cdot \underline{z}_{\rm E}^{\rm T} = \mathbf{H}_{\rm K} \cdot \underline{z}_{\rm K}^{\rm T} = \begin{pmatrix} 1 & \alpha^1 & \alpha^4 & \alpha^6 \\ 1 & \alpha^2 & \alpha^1 & \alpha^5 \\ 1 & \alpha^3 & \alpha^5 & \alpha^4 \\ 1 & \alpha^4 & \alpha^2 & \alpha^3 \end{pmatrix} \cdot \begin{pmatrix} \alpha^1 \\ 1 \\ \alpha^2 \\ \alpha^6 \end{pmatrix} = \ \ldots \ = \begin{pmatrix} \alpha^3 \\ \alpha^4 \\ \alpha^2 \\ 0 \end{pmatrix}.$$

Die rechte Gleichungsseite ergibt für das betrachtete Beispiel $\Rightarrow$ $\underline{z}_{\rm K} = (c_0, c_1, c_4, c_6)$ und basiert auf dem Polynom $p(x) = x^3 + x + 1$, das zu folgenden Potenzen in α führt:

$$\begin{aligned} &\alpha^3 = \alpha + 1, \quad \alpha^4 = \alpha^2 + \alpha, \quad \alpha^5 = \alpha^2 + \alpha + 1, \quad \alpha^6 = \alpha^2 + 1, \\ &\alpha^7 = 1, \qquad \alpha^8 = \alpha^1, \qquad \alpha^9 = \alpha^2, \qquad \alpha^{10} = \alpha^3 = \alpha + 1, \ \ldots \end{aligned}$$

Damit lautet die Matrizengleichung zur Bestimmung des gesuchten Vektors $\underline{z}_{\rm E}$:

$$\begin{pmatrix} \alpha^2 & \alpha^3 & \alpha^5 \\ \alpha^4 & \alpha^6 & \alpha^3 \\ \alpha^6 & \alpha^2 & \alpha^1 \\ \alpha^1 & \alpha^5 & \alpha^6 \end{pmatrix} \cdot \begin{pmatrix} z_2 \\ z_3 \\ z_5 \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} \alpha^3 \\ \alpha^4 \\ \alpha^2 \\ 0 \end{pmatrix}.$$

Löst man diese Matrizengleichung (am einfachsten per Programm), so erhält man

$$z_2 = \alpha^2, \quad z_3 = \alpha^1, \quad z_5 = \alpha^5 \quad \Rightarrow \quad \underline{z} = \left(\alpha^1, 1, \alpha^2, \alpha^1, \alpha^2, \alpha^5, \alpha^5\right).$$

Das Ergebnis ist richtig, wie die folgenden Kontrollrechnungen zeigen:

$$\begin{aligned} \alpha^2 \cdot \alpha^2 + \alpha^3 \cdot \alpha^1 + \alpha^5 \cdot \alpha^5 &= \alpha^4 + \alpha^4 + \alpha^{10} = \alpha^{10} = \alpha^3, \\ \alpha^4 \cdot \alpha^2 + \alpha^6 \cdot \alpha^1 + \alpha^3 \cdot \alpha^5 &= (\alpha^2 + 1) + (1) + (\alpha) = \alpha^2 + \alpha = \alpha^4, \\ \alpha^6 \cdot \alpha^2 + \alpha^2 \cdot \alpha^1 + \alpha^1 \cdot \alpha^5 &= (\alpha) + (\alpha + 1) + (\alpha^2 + 1) = \alpha^2, \\ \alpha^1 \cdot \alpha^2 + \alpha^5 \cdot \alpha^1 + \alpha^6 \cdot \alpha^5 &= (\alpha + 1) + (\alpha^2 + 1) + (\alpha^2 + \alpha) = 0. \end{aligned}$$

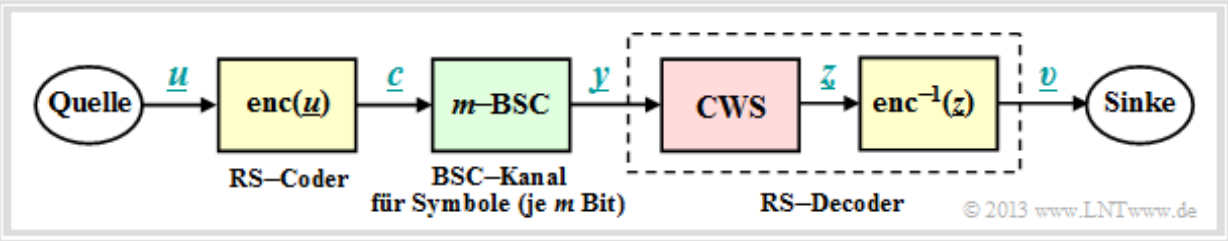
Das zugehörige Informationswort erhält man mit der **Generatormatrix G** zu $\underline{v} = \underline{z} \cdot \mathbf{G}^{\rm T} = (\alpha^1, 1, \alpha^3)$.

Blockschaltbild und Voraussetzungen zu Kapitel 2.5

Wie im **Kapitel 2.4** betrachten wir ein Übertragungssystem mit Reed–Solomon–Codierung, das durch die beiden Codeparameter $n = 2^m - 1$ und k gekennzeichnet ist. Mit der Generatormatrix $\mathbf{G}$ lautet der Zusammenhang zwischen dem Informationswort $\underline{u}$ und dem Codewort $\underline{c}$:

$$\underline{c} = \text{enc}(\underline{u}) = \underline{u} \cdot \mathbf{G} \quad \text{mit} \quad \underline{u} = (u_0, u_1, \ldots, u_i, \ldots, u_{k-1}), \quad \underline{c} = (c_0, c_1, \ldots, c_i, \ldots, c_{n-1}).$$

Sowohl die Informationssymbole u_i als auch die Codesymbole c_i entstammen dem Körper GF(q) mit $q = n + 1 = 2^m$, und sind somit durch m Binärsymbole (Bit) darstellbar.



Ein Vergleich dieses Blockschaltbildes mit dem entsprechenden **Modell zu Kapitel 2.4** zeigt:

- Der wesentliche Unterschied liegt im verwendeten diskreten Kanalmodell (grün hinterlegt). Anstelle des Auslöschungskanals („m–BEC") wird nun der m–BSC betrachtet. Für jedes einzelne Bit des Codesymbols c_i wird der **Binary Symmetric Channel** (BSC) angewandt. Ist auch nur ein Bit innerhalb des Codesymbols verfälscht, so ist $y_i \neq c_i$.
- Im Kapitel 2.4 sind unsichere Bit bereits durch Auslöschungen E (*Erasures*) markiert. Aufgabe des *Codewortfinders* (CWF) ist es deshalb, aus dem verstümmelten Empfangswort $\underline{y}$ das Decodierergebnis $\underline{z}$ zu rekonstruieren. Ist die Anzahl e der Auslöschungen kleiner als die minimale Distanz d_{min}, so gelingt dies und man erhält $\underline{z} = \underline{c}$. Andernfalls meldet der CWF, dass er das aktuelle Empfangswort $\underline{y}$ nicht decodieren kann. Eine Fehlentscheidung ($\underline{z} \neq \underline{c}$) ist ausgeschlossen.
- In diesem Kapitel wird nun der erste Decoderblock als *Codewortschätzer* (CWS) bezeichnet. Die Namensgebung soll deutlich machen, dass aufgrund des m–BSC–Modells Fehlentscheidungen ($\underline{z} \neq \underline{c}$) unvermeidlich sind, nämlich dann, wenn durch mehrere Symbolfehler das Empfangswort $\underline{y}$ zu einem gültigen Codewort verfälscht wurde.

Aufgabe des Decoders ist es, seinen Ausgangsvektor $\underline{v}$ so zu bestimmen, dass er „möglichst gut" mit dem Informationswort $\underline{u}$ übereinstimmt. Oder etwas genauer formuliert:

$$\text{Pr(Blockfehler)} = \text{Pr}(\underline{v} \neq \underline{u}) \stackrel{!}{=} \text{Minimum}.$$

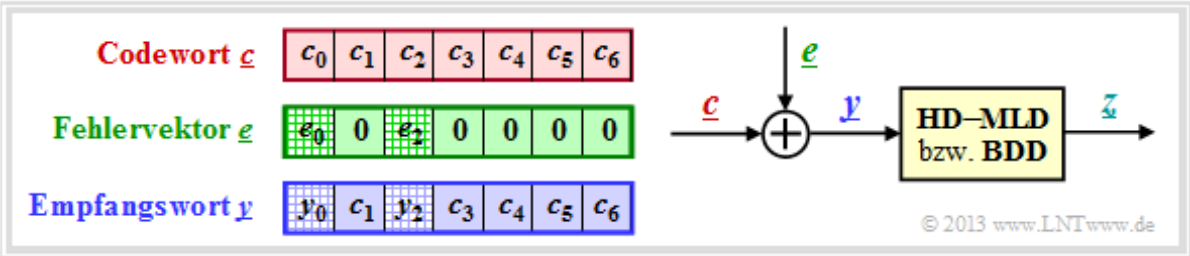
Aufgrund des deterministischen Mappings $\underline{c} = \text{enc}(\underline{u})$ und $\underline{v} = \text{enc}^{-1}(\underline{z})$ gilt in gleicher Weise:

$$\text{Pr(Blockfehler)} = \text{Pr}(\underline{z} \neq \underline{c}) \stackrel{!}{=} \text{Minimum}.$$

Deshalb werden im Folgenden die zwei gelb hinterlegten Blöcke nicht weiter betrachtet. Im Mittelpunkt der Betrachtungen steht vielmehr der rot hinterlegte *Codewortschätzer* (CWS).

Mögliche Codewortschätzer: HD–MLD bzw. BDD (1)

Die rechte Skizze der nachfolgenden Grafik verdeutlicht nochmals die Aufgabenstellung, wobei hier das Kanalmodell „*m*–BSC" durch den **additiven Fehlervektor** $\underline{e} = \underline{y} - \underline{c}$ ersetzt ist. Die linke Skizze verdeutlicht den Zusammenhang zwischen diesen drei Vektoren.



Diese Aufgabenstellung soll durch ein Beispiel verdeutlicht werden.

Beispiel: Alle nachfolgend genannten Symbole sind Elemente von $GF(2^3) = \{0, 1, \alpha^1, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\}$. Zur Umrechnung zwischen der Koeffizientendarstellung (mit der Reihenfolge k_2, k_1, k_0) und der Exponentendarstellung (als Potenzen des primitiven Elements α) kann die nebenstehende Tabelle verwendet werden.

Potenzen von α	Polynome in α	Vektoren $k_2\,k_1\,k_0$
$\alpha^{-\infty}=0$	0	0 0 0
$\alpha^0=1$	1	0 0 1
α^1	α	0 1 0
α^2	α^2	1 0 0
α^3	$\alpha+1$	0 1 1
α^4	$\alpha^2+\alpha$	1 1 0
α^5	$\alpha^2+\alpha+1$	1 1 1
α^6	α^2+1	1 0 1

Codewort und Empfangswort lauten in Koeffizientendarstellung:

$$\underline{c} = \big((010), (001), (100), (010), (100), (111), (111)\big),$$
$$\underline{y} = \big((011), (001), (000), (010), (100), (111), (111)\big).$$

Damit ergibt sich für den Fehlervektor $\underline{e} = \underline{y} - \underline{c}$:

$$\underline{e} = \big((001), (000), (100), (000), (000), (000), (000)\big).$$

Umgewandelt in die Exponentendarstellung erhält man:

$$\underline{c} = \big(\alpha^1, 1, \alpha^2, \alpha^1, \alpha^2, \alpha^5, \alpha^5\big),$$
$$\underline{y} = \big(\alpha^3, 1, 0, \alpha^1, \alpha^2, \alpha^5, \alpha^5\big) \quad \Rightarrow \quad \underline{e} = \big(1, 0, \alpha^2, 0, 0, 0, 0\big).$$

Aufgabe des Codewortschätzers (CWS) ist es, das zu $\underline{y}$ wahrscheinlichste Codewort $\underline{c}_i$ zu finden und sein Ergebnis $\underline{z} = \underline{c}_i$ an das nachfolgende Mapping weiterzugeben. Es gibt verschiedene Möglichkeiten:

- *Hard Decision Maximum Likelihood Decoding* (HD–MLD),
- *Bounded Distance Decoding* (BDD),
- Decodierung über die halbe Mindestdistanz.

Diese Decodierprinzipien werden auf der nächsten Seite ausführlicher behandelt.

Mögliche Codewortschätzer: HD–MLD bzw. BDD (2)

Aufgabe des Codewortschätzers (CWS) ist es, das zu $\underline{y}$ wahrscheinlichste Codewort $\underline{c}_i$ zu finden und sein Ergebnis $\underline{z} = \underline{c}_i$ weiterzugeben. Hierfür gibt es mehrere Möglichkeiten:

Hard Decision Maximum Likelihood Decoding (HD–MLD):

Man wählt von allen möglichen Reed–Solomon–Codeworten $\underline{c}_i$ (hiervon gibt es insgesamt q^k) dasjenige mit der geringsten **Hamming–Distanz** zum Empfangswort $\underline{y}$ aus. Somit lautet das Ergebnis:

$$\underline{z} = \arg \min_{\underline{x}_i \in \mathcal{C}_{\rm RS}} d_{\rm H}(\underline{y}, \underline{c}_i).$$

Die Entscheidung passiert hier auf der maximalen Rückschlusswahrscheinlichkeit $\Pr(\underline{c}_i \mid \underline{y})$ und führt zum bestmöglichen Ergebnis. Näheres siehe **Kapitel 1.2.** Es wird stets entschieden, selbst wenn die Anzahl r der Symbolfehler größer ist als die Korrekturfähigkeit t des Codes. In einem solchen Fall ist allerdings das Decodierergebnis sehr unsicher.

Es sei nochmals erwähnt, dass bei ML–Decodierung immer entschieden wird. Ein Decodierversagen ist ausgeschlossen. Aber natürlich gibt es auch falsche Entscheidungen.

Bounded Distance Decoding (BDD):

Falls die Anzahl r der Symbolfehler im Empfangswort $\underline{y}$ nicht größer ist als die Korrekturfähigkeit $t = \lfloor (d_{\rm min} - 1)/2 \rfloor$ des Codes, kann man die r Symbolfehler vollständig korrigieren. Der Fall $r > t$ führt zu einem Abbruch des Decodiervorgangs ohne Ergebnis. Anders ausgedrückt: Es werden nur diejenigen Empfangsworte zum Kugelmittelpunkt decodiert, die in einer Kugel um diesen mit Radius t liegen. Andere werden als undecodierbar markiert, zum Beispiel als *Erasure*.

Decodierung über die halbe Mindestdistanz:

Hier wird auch im Fall $r > t$ versucht, das Codewort zu decodieren. Im Gegensatz zu HD–MLD, das ebenfalls über die halbe Mindestdistanz hinaus decodiert, ist hier aber ein Decodierversagen nicht per se ausgeschlossen.

Für den Rest dieses Kapitels beschäftigen wir uns ausschließlich mit *Bounded Distance Decoding*. Der Grund hierfür ist die enorme Komplexität der *Maximum Likelihood Detection* proportional zu q^{n-k}.

Vorgehensweise beim „Bounded Distance Decoding"

Im Folgenden werden die einzelnen Schritte des BDD–Algorithmuses kurz und rezeptartig beschrieben. Auf den nächsten Seiten werden dann die einzelnen Punkte genauer behandelt und die Vorgehensweise an typischen Beispielen verdeutlicht.

(A) Berechne das Syndrom $\underline{s} = \underline{y} \cdot \mathbf{H}^{\rm T}$:

- Ergibt sich aus dem Empfangswort $\underline{y}$ und der Prüfmatrix $\mathbf{H}$ des Codes das Syndrom $\underline{s} = \underline{0}$, so setze den BDD–Ausgang $\underline{z} = \underline{y}$ und beende den Decodiervorgang für dieses Empfangswort.
- Andernfalls setze den Parameter $r = 1$ und mache mit Schritt (B) weiter.

(B) Bestimme die tatsächliche Symbolfehleranzahl r:

- Erstelle und überprüfe die Gleichungen $\underline{\Lambda}_l \cdot \underline{s}^{\rm T} = 0$ für $l = 1, ..., 2t–r$ unter der Annahme, dass das Empfangswort genau r Symbolfehler beinhaltet.
- $\underline{\Lambda}_l$ bezeichnet die verallgemeinerten ELP–Koeffizientenvektoren und t die Korrekturfähigkeit des Codes. Für die Reed–Solomon–Codes gilt einheitlich $t = \lfloor (n – k)/2 \rfloor$.
- Gibt es eine eindeutige Lösung, dann mache mit Schritt (C) weiter. Im Empfangsvektor $\underline{y}$ sind dann tatsächlich genau r Symbole verfälscht und im Fehlervektor $\underline{e}$ gibt es r Einträge ungleich 0.
- Andernfalls erhöhe r um 1. Falls $r \le t$, dann wiederhole Schritt (B) von Beginn an: Das bisher angenommene r war offensichtlich zu klein. Deshalb nun ein neuer Versuch mit größerem r.
- Ist das neue r größer als die Korrekturfähigkeit t des Codes, so kann das aktuelle Empfangswort nicht decodiert werden. Beende den Decodierversuch mit einer Fehlermeldung.

(C) Lokalisiere die r Fehlerpositionen:

- Erstelle das *Error Locator Polynom* $\Lambda(x)$ und finde dessen r Nullstellen in $\mathrm{GF}(q) \setminus \{0\}$.
- Ein Fehler an der Stelle i liegt immer dann vor, wenn $\Lambda(\alpha^i) = 0$ ist.

(D) Bestimme die r Fehlerwerte $e_i \neq 0$:

- Bekannt sind die r Fehlerstellen. Ersetzt man im Empfangsvektor $\underline{y}$ die falschen Symbole durch Auslöschungen $\Rightarrow$ $y_i = \mathrm{E}$, falls $e_i \neq 0$, so findet man das Ergebnis $\underline{z}$ entsprechend Kapitel 2.4.
- Eine Alternative: Aus der Gleichung $\underline{e} \cdot \mathbf{H}^{\rm T} = \underline{s}$ kommt man unter Ausnutzung der fehlerfreien Stellen ($e_i = 0$) zu einem linearen Gleichungssystem für die fehlerhaften Symbole ($e_i \neq 0$).

Schritt (A): Auswertung des Syndroms beim BDD

Wie in **Kapitel 1.5** gezeigt, kann zur Decodierung eines linearen Codes das Syndrom $\underline{s}$ herangezogen werden. Mit dem Empfangswort $\underline{y}$ gleich Codewort $\underline{c}$ plus Fehlervektor $\underline{e}$ gilt für dieses:

$$\underline{s} = \underline{y} \cdot \mathbf{H}^{\rm T} = \underline{c} \cdot \mathbf{H}^{\rm T} + \underline{e} \cdot \mathbf{H}^{\rm T}.$$

Da stets $\underline{c} \cdot \mathbf{H}^{\rm T} = \underline{0}$ gilt, folgt aus $\underline{s} = \underline{0}$ auch $\underline{e} \cdot \mathbf{H}^{\rm T} = \underline{0}$. Das heißt:

- Mit sehr großer Wahrscheinlichkeit kann aus $\underline{s} = \underline{0}$ auch auf $\underline{e} = \underline{0}$ und damit auch auf das richtige Decodierergebnis $\underline{z} = \underline{y}$ geschlossen werden. Der Decodiervorgang wäre damit abgeschlossen.
- Es gibt aber auch Fehlermuster $\underline{e} \neq \underline{0}$, die zum Syndrom $\underline{s} = \underline{0}$ führen. Solche Muster beinhalten sicher mehr als t Symbolfehler, so dass auch hier der Abbruch des Decodiervorgangs sinnvoll ist. Alle nachfolgenden Berechnungen würden auch nicht zum Erfolg führen.

Beispiel A: Diesem und den folgenden Beispielen auf den nächsten Seiten liegt stets der Reed–Solomon–Code $(7, 3, 5)_8$ zugrunde, so dass die hier angegebenen Umrechnungen in GF(2^3) genutzt werden können. Das Empfangswort lautet:

$$\underline{y} = (\alpha^3, 1, 0, \alpha^1, \alpha^2, \alpha^5, \alpha^5).$$

Potenzen von α	Polynome in α	Vektoren $k_2\, k_1\, k_0$
$\alpha^{-\infty}=0$	0	0 0 0
$\alpha^0=1$	1	0 0 1
α^1	α	0 1 0
α^2	α^2	1 0 0
α^3	$\alpha+1$	0 1 1
α^4	$\alpha^2+\alpha$	1 1 0
α^5	$\alpha^2+\alpha+1$	1 1 1
α^6	α^2+1	1 0 1

Mit der **Prüfmatrix H** ergibt sich für das Syndrom:

$$\underline{s} = \underline{y} \cdot \mathbf{H}^{\rm T} = (\alpha^3, 1, 0, \alpha^1, \alpha^2, \alpha^5, \alpha^5) \cdot \begin{pmatrix} 1 & 1 & 1 & 1 \\ \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 \\ \alpha^2 & \alpha^4 & \alpha^6 & \alpha^1 \\ \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 \\ \alpha^4 & \alpha^1 & \alpha^5 & \alpha^2 \\ \alpha^5 & \alpha^3 & \alpha^1 & \alpha^6 \\ \alpha^6 & \alpha^5 & \alpha^4 & \alpha^3 \end{pmatrix} =$$

$$= (\alpha^3, \alpha^3, \alpha^3, \alpha^3) + (\alpha^1, \alpha^2, \alpha^3, \alpha^4) + (0, 0, 0, 0) + (\alpha^4, 1, \alpha^3, \alpha^6) +$$
$$+ (\alpha^6, \alpha^3, 1, \alpha^4) + (\alpha^3, \alpha^1, \alpha^6, \alpha^4) + (\alpha^4, \alpha^3, \alpha^2, \alpha^1) = \ldots = (\alpha^5, \alpha^2, \alpha^3, \alpha^1).$$

Das Empfangswort wurde also verfälscht. Andernfalls hätte sich $\underline{s} = \underline{0} = (0, 0, 0, 0)$ ergeben müssen.

Die Beschreibung des Decodiervorgangs beim RSC $(7, 3, 5)_8$ wird im **Beispiel B** fortgesetzt.

Error Locator Polynom – Definition und Eigenschaften (1)

Nach der **Syndromberechnung** mit dem Ergebnis $\underline{s} \neq 0$ wissen wir,

- dass das Empfangswort $\underline{y}$ nicht mit dem Codewort $\underline{c}$ übereinstimmt, bzw.
- dass der Fehlervektor $\underline{e} = (e_0, e_1, \ldots, e_{n-1})$ auch Elemente ungleich 0 beinhaltet.

Wir wissen allerdings nicht, wie viele Symbole verfälscht wurden ($0 < r \leq n$) und wir können auch nicht die Positionen der Fehlerstellen ($e_i \neq 0$) im Fehlervektor $\underline{e}$ nennen.

Einen Lösungsansatz für diese Aufgabe bietet das *Error Locator Polynom*, das von W. W. Peterson eingeführt wurde. Siehe **[Pet60].** Im Deutschen ist hierfür auch der Begriff *Schlüsselgleichung* üblich.

Definition: Es sei bekannt, dass genau r Elemente des Fehlervektors $\underline{e}$ ungleich 0 sind, erkennbar am Hamming–Gewicht $w_{\rm H}(\underline{e}) = r$. Ebenfalls bekannt sei die Menge $I_{\rm FP}$ der Fehlerpositionen:

$$I_{\rm FP} = \{i \mid e_i \neq 0,\ 0 \leq i < n\}.$$

Dann gilt für das *Error Locator Polynom* (ELP):

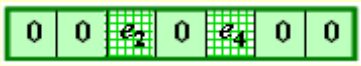
$$\Lambda(x) = x \cdot \prod_{i \in I_{\rm FP}} (x - \alpha^i) = x \cdot \left[\lambda_0 + \lambda_1 \cdot x + \ldots + \lambda_{r-1} \cdot x^{r-1} + x^r\right].$$

Vom *Error Locator Polynom* wissen wir aufgrund der Definition:

- Wegen des Faktors x vor dem Produktzeichen ist $\Lambda(x = 0) = 0$.
- Weitere r Nullstellen ergeben sich für $x = \alpha^i$ mit $i \in I_{\rm FP}$, das heißt, für alle Fehlerpositionen.
- Dagegen ergibt das *Error Locator Polynom* für $i \notin I_{\rm FP} \Rightarrow e_i = 0$ keine Nullstelle: $\Lambda(x = \alpha^i) \neq 0$.

Wir suchen also die r nichttrivialen Nullstellen von $\Lambda(x)$ mit dem Argument $x \in {\rm GF}(q) \setminus \{0\}$. Gelingt uns dies, so kennen wir die r Fehlerpositionen, jedoch noch nicht die tatsächlichen Fehlerwerte $e_i \in {\rm GF}(q)$.

Beispiel: Es gelte $n = 7 \Rightarrow q = 8$, $r = 2$ und $I_{\rm FP} = \{2, 4\}$:



Damit erhält man für das *Error Locator Poynom* aus GF(2^3):

$$\begin{aligned}\Lambda(x) &= x \cdot (x - \alpha^2) \cdot (x - \alpha^4) = \\ &= x \cdot (x + \alpha^2) \cdot (x + \alpha^4) = \\ &= x \cdot \left[x^2 + (\alpha^2 + \alpha^4) \cdot x + \alpha^6\right] = \\ &= x \cdot \left[\alpha^6 + \alpha \cdot x + x^2\right].\end{aligned}$$

Potenzen von α	Polynome in α	Vektoren $k_2\, k_1\, k_0$
$\alpha^{-\infty} = 0$	0	0 0 0
$\alpha^0 = 1$	1	0 0 1
α^1	α	0 1 0
α^2	α^2	1 0 0
α^3	$\alpha+1$	0 1 1
α^4	$\alpha^2+\alpha$	1 1 0
α^5	$\alpha^2+\alpha+1$	1 1 1
α^6	α^2+1	1 0 1

Die beiden Nullstellen (außer bei $x = 0$) ergeben sich hier natürlich für $x = \alpha^2$ und $x = \alpha^4$, wie die folgende Kontrollrechnung zeigt:

$$\begin{aligned}\Lambda(x = \alpha^2) &= x \cdot \left[\alpha^6 + \alpha \cdot \alpha^2 + (\alpha^2)^2\right] = x \cdot \left[\alpha^6 + \alpha^3 + \alpha^4\right] = 0, \\ \Lambda(x = \alpha^4) &= x \cdot \left[\alpha^6 + \alpha \cdot \alpha^4 + (\alpha^4)^2\right] = x \cdot \left[\alpha^6 + \alpha^5 + \alpha\right] = 0.\end{aligned}$$

Error Locator Polynom – Definition und Eigenschaften (2)

Für die weitere Herleitung gehen wir stets vom **RSC (7, 3, 5)$_8$** mit den folgenden Parameterwerten aus: $n = 7$, $k = 3$, $d_{\rm min} = 5 \Rightarrow t = (d_{\rm min} - 1)/2 = 2$. Die Anzahl der Symbolfehler sei $\boldsymbol{r = 2 = t}$.

Damit lautet das zu lösende Gleichungssystem mit den Hilfsgrößen $A_i = \Lambda(\alpha^i)$:

$$\begin{aligned}
A_0 = \Lambda(\alpha^0) &= \alpha^0 \cdot \left[\lambda_0 + \lambda_1 \cdot (\alpha^0)^1 + (\alpha^0)^2\right] = \lambda_0 \cdot 1 + \lambda_1 \cdot 1 + 1\,,\\
A_1 = \Lambda(\alpha^1) &= \alpha^1 \cdot \left[\lambda_0 + \lambda_1 \cdot (\alpha^1)^1 + (\alpha^1)^2\right] = \lambda_0 \cdot \alpha^1 + \lambda_1 \cdot \alpha^2 + \alpha^3\,,\\
&\cdots\\
A_6 = \Lambda(\alpha^6) &= \alpha^6 \cdot \left[\lambda_0 + \lambda_1 \cdot (\alpha^6)^1 + (\alpha^6)^2\right] = \lambda_0 \cdot \alpha^6 + \lambda_1 \cdot \alpha^{12} + \alpha^{18}\,.
\end{aligned}$$

In Vektorform lautet dieses Gleichungssystem mit dem Hilfsvektor $\underline{A} = (A_0, A_1, A_2, A_3, A_4, A_5, A_6)$:

$$\underline{A}^{\rm T} = \begin{pmatrix} A_0\\ A_1\\ A_2\\ A_3\\ A_4\\ A_5\\ A_6 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1\\ \alpha^1 & \alpha^2 & \alpha^3\\ \alpha^2 & \alpha^4 & \alpha^6\\ \alpha^3 & \alpha^6 & \alpha^9\\ \alpha^4 & \alpha^8 & \alpha^{12}\\ \alpha^5 & \alpha^{10} & \alpha^{15}\\ \alpha^6 & \alpha^{12} & \alpha^{18} \end{pmatrix} \cdot \begin{pmatrix} \lambda_0\\ \lambda_1\\ 1 \end{pmatrix}.$$

Wir erweitern nun den ELP–Koeffizientenvektor $\underline{\Lambda}$ durch Anhängen von Nullen auf die Länge $n - k$. Im betrachteten Beispiel erhält man somit $\underline{\Lambda} = (\lambda_0, \lambda_1, 1, 0)$ und folgende Vektorgleichung:

$$\underline{A}^{\rm T} = \begin{pmatrix} 1 & 1 & 1 & 1\\ \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4\\ \alpha^2 & \alpha^4 & \alpha^6 & \alpha^8\\ \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12}\\ \alpha^4 & \alpha^8 & \alpha^{12} & \alpha^{16}\\ \alpha^5 & \alpha^{10} & \alpha^{15} & \alpha^{20}\\ \alpha^6 & \alpha^{12} & \alpha^{18} & \alpha^{24} \end{pmatrix} \cdot \begin{pmatrix} \lambda_0\\ \lambda_1\\ 1\\ 0 \end{pmatrix}.$$

Aus der 7 × 3–Matrix wurde nun eine 7 × 4–Matrix. Ddie vierte Spalte kann eigentlich beliebig gefüllt werden, da alle Elemente mit Nullen multipliziert werden. Durch die hier gewählte Ergänzung erhält man die transponierte **Prüfmatrix** des RSC (7, 3, 5)$_8$, und man kann für die letzte Vektorgleichung schreiben:

$$\underline{A}^{\rm T} = \mathbf{H}^{\rm T} \cdot \underline{\Lambda}^{\rm T} \quad \Rightarrow \quad \underline{A} = \underline{\Lambda} \cdot \mathbf{H}.$$

Da aber für die Fehlerstellen ($e_i \neq 0$) stets $A_i = \Lambda(\alpha^i) = 0$ gilt, ist das Produkt $A_i \cdot e_i$ immer 0 und man erhält als Bestimmungsgleichung für die Nullstellen des *Error Locator Polynoms*:

$$\underline{A}^{\rm T} \cdot \underline{e}^{\rm T} = 0 \quad \Rightarrow \quad \underline{\Lambda} \cdot \mathbf{H} \cdot \underline{e}^{\rm T} = 0 \quad \Rightarrow \quad \underline{\Lambda} \cdot \underline{s}^{\rm T} = 0\,.$$

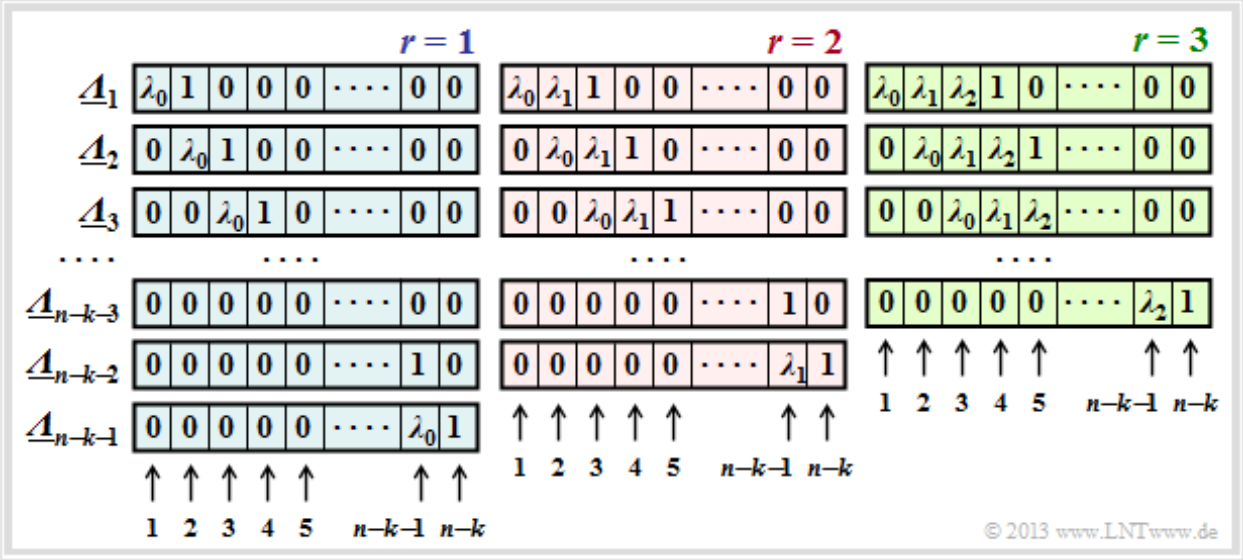
Daraus folgt das wichtige **Zwischenergebnis**: Die nichttrivialen Nullstellen (ungleich 0) λ_0, λ_1, ... des *Error Locator Polynoms* $\Lambda(x)$ müssen stets der Vektorgleichung $\underline{\Lambda} \cdot \underline{s}^{\rm T} = 0$ genügen, wobei $\underline{\Lambda}$ den ELP–Koeffizientenvektor bezeichnet und $\underline{s} = \underline{y} \cdot \mathbf{H}^{\rm T}$ das **Syndrom** angibt.

Schritt (B): Aufstellen/Auswerten des ELP–Koeffizientenvektors (1)

Bevor wir das Zwischenergebnis $\underline{\Lambda} \cdot \underline{s}^{\rm T} = 0$ an einem Beispiel verdeutlichen können, müssen noch einige Verallgemeinerungen vorgenommen werden. Der Grund hierfür ist:

- Die Gleichung $\underline{\Lambda} \cdot \underline{s}^{\rm T} = 0$ liefert nur eine einzige Bestimmungsgleichung. Damit kann das Problem für $r = 1$ gelöst werden, wenn man sicher ist, dass tatsächlich nur ein Symbol verfälscht wurde.
- Ist man sich dessen nicht sicher, führt aber die Berechnung trotzdem für $r = 1$ durch, so braucht man noch eine zweite Gleichung (oder auch mehrere), um die Annahme zu verifizieren.

Die Eigenschaft des **Error Locator Polynoms,** dass $\Lambda(\alpha^i)$ nur für $e_i \neq 0$ (i–tes Symbol verfälscht) gleich Null ist, bleibt erhalten, wenn man $\Lambda(x)$ mit beliebigen Potenzen von x multipliziert. Jede Multiplikation mit x bedeutet für den ELP–Koeffizientenvektor eine Verschiebung um eine Stelle nach rechts.



Mit der **verallgemeinerten Definition** (wobei $\Lambda_1(x)$ dem bisherigen $\Lambda(x)$ entspricht),

$$\Lambda_l(x) = x^l \cdot \prod_{i \in I_{\rm FP}} (x - \alpha^i) = x^l \cdot \left[\lambda_0 + \lambda_1 \cdot x + \ldots + \lambda_{r-1} \cdot x^{r-1} + x^r\right],$$ ergeben sich durch sukzessive Verschiebung gegenüber $\underline{\Lambda}$ die ELP–Koeffizientenvektoren $\underline{\Lambda}_l$. Die Grafik zeigt die Belegung unter der Annahme von $1 \le r \le 3$ Fehlerstellen im Vektor $\underline{e}$. Man erkennt:

- Die Länge aller $\underline{\Lambda}_l$ ist stets $n - k$. Jeder Vektor beinhaltet jeweils r Koeffizienten $\lambda_i (0 \le i < r)$ und eine Eins. Der Rest eines jeden Vektors ist mit Nullen aufgefüllt.
- Für jedes r gibt es genau $n–k–r$ Koeffizientenvektoren $\underline{\Lambda}_l$, wobei sich $\underline{\Lambda}_l$ aus $\underline{\Lambda}_{l-1}$ stets durch Rechtsverschiebung um eine Position ergibt. Der Vektor $\underline{\Lambda}_{n-k-r}$ endet immer mit einer 1.
- Das Gleichungssystem $\underline{\Lambda}_l \cdot \underline{s}^{\rm T} = 0$ führt deshalb zu $n - k - r$ Gleichungen. Der gewählte Ansatz für r ist nur dann richtig, wenn alle Gleichungen zu den gleichen Ergebnissen für $\lambda_0, \ldots, \lambda_{r-1}$ führen.
- Ist dies nicht der Fall, so muss man r erhöhen und damit ein neues Gleichungssystem bearbeiten, und zwar solange, bis sich aus allen Gleichungen für das aktuelle r eine eindeutige Lösung ergibt.
- Ist schließlich r größer als die Korrekturfähigkeit t des Codes, so kann die Berechnung beendet werden. Das anstehende Empfangswort $\underline{y}$ ist dann nicht decodierbar.

Schritt (B): Aufstellen/Auswerten des ELP–Koeffizientenvektors (2)

Hier nochmals die *verallgemeinerte Definition* für das **Error Locator Polynom** (ELP):

$$\Lambda_l(x) = x^l \cdot \prod_{i \in I_{\rm FP}} (x - \alpha^i) = x^l \cdot \left[\lambda_0 + \lambda_1 \cdot x + \ldots + \lambda_{r-1} \cdot x^{r-1} + x^r\right].$$

Dieses lässt sich am einfachsten mit den **verschobenen ELP–Koeffizientenvektoren** $\underline{\Lambda}_l$ auswerten, wie im folgenden Beispiel gezeigt wird. Hierbei beziehen wir uns auf die **Grafik auf der letzen Seite,** die $\underline{\Lambda}_l$–Belegung unter der Annahme $r = 1$, $r = 2$ oder $r = 3$ Fehler im Fehlervektor $\underline{e}$ zeigt.

Beispiel B: Es gelten weiterhin die im **Beispiel A** genannten Voraussetzungen. Dort wurde aufgrund des Syndroms $\underline{s} = (\alpha^5, \alpha^2, \alpha^3, \alpha) \neq 0$ auch nachgewiesen, dass der Empfangsvektor $\underline{y}$ verfälscht wurde ⇒ Fehlervektor $\underline{e} \neq \underline{0}$. Nicht bekannt ist allerdings die tatsächliche Symbolfehleranzahl r.

Unter der Annahme eines einzigen falschen Symbols ($r = 1$) erhält man folgendes Gleichungssystem (hier in Matrixform geschrieben):

$$\left(\boldsymbol{\Lambda}_l\right) \cdot \underline{s}^{\rm T} = \begin{pmatrix} \lambda_0 & 1 & 0 & 0 \\ 0 & \lambda_0 & 1 & 0 \\ 0 & 0 & \lambda_0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \alpha^5 \\ \alpha^2 \\ \alpha^3 \\ \alpha \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\Rightarrow \quad \alpha^5 \cdot \lambda_0 + \alpha^2 = 0 \quad \Rightarrow \quad \lambda_0 = \alpha^{2-5} = \alpha^{-3} = \alpha^4,$$
$$\alpha^2 \cdot \lambda_0 + \alpha^3 = 0 \quad \Rightarrow \quad \lambda_0 = \alpha^{3-2} = \alpha,$$
$$\alpha^3 \cdot \lambda_0 + \alpha^1 = 0 \quad \Rightarrow \quad \lambda_0 = \alpha^{1-3} = \alpha^{-2} = \alpha^5.$$

Potenzen von α	Polynome in α	Vektoren $k_2\, k_1\, k_0$
$\alpha^{-\infty}=0$	0	0 0 0
$\alpha^0=1$	1	0 0 1
α^1	α	0 1 0
α^2	α^2	1 0 0
α^3	$\alpha+1$	0 1 1
α^4	$\alpha^2+\alpha$	1 1 0
α^5	$\alpha^2+\alpha+1$	1 1 1
α^6	α^2+1	1 0 1

Diese drei Gleichungen liefern drei unterschiedliche Lösungen für λ_0, was nicht zielführend ist.

Deshalb stellen wir nun ein weiteres Gleichungssystem auf, und zwar unter der Annahme $r = 2$:

$$\left(\boldsymbol{\Lambda}_l\right) \cdot \underline{s}^{\rm T} = \begin{pmatrix} \lambda_0 & \lambda_1 & 1 & 0 \\ 0 & \lambda_0 & \lambda_1 & 1 \end{pmatrix} \cdot \begin{pmatrix} \alpha^5 \\ \alpha^2 \\ \alpha^3 \\ \alpha \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\Rightarrow \quad \alpha^5 \cdot \lambda_0 + \alpha^2 \cdot \lambda_1 + \alpha^3 = 0,$$
$$\alpha^2 \cdot \lambda_0 + \alpha^3 \cdot \lambda_1 + \alpha^1 = 0.$$

Dieses Gleichungssystem ist nun eindeutig lösbar. Man erhält $\lambda_0 = \alpha^2$ und $\lambda_1 = \alpha^6$. Das bedeutet: Die Annahme, dass tatsächlich $r = 2$ Positionen des Empfangsvektors $\underline{y}$ verfälscht wurden, ist richtig.

Man weiß aber noch nicht, welche Positionen verfälscht wurden. Soviel vorneweg; Es sind nicht die Positionen 2 und 6, sondern die Symbolpositionen 0 und 2, wie im folgenden **Beispiel C** gezeigt wird.

Schritt (C): Lokalisierung der Fehlerstellen

Nach Abarbeitung von Schritt (B) sind bekannt:

- die Anzahl r der Fehlerstellen $e_i \neq 0$ im Vektor $\underline{e} = (e_0, \ldots, e_i, \ldots, e_{n-1})$,
- die Koeffizienten $\lambda_0, \ldots, \lambda_{r-1}$ des *Error Locator Polynoms*.

Zu bestimmen ist nun noch die Menge der Fehlerpositionen:

$$I_{\rm FP} = \{i \mid e_i \neq 0,\ 0 \le i < n\}.$$

Hierzu gibt es zwei Möglichkeiten:

- die so genannte *Chien–Suche*, in dem man durch Einsetzen der möglichen Codesymbole außer dem Nullsymbol $\Rightarrow \alpha^i$ $(0 \le i < n)$ in das *Error Locator Polynom* dessen Nullstellen er- mittelt,
- die Auswertung der Gleichung $\underline{A} = (A_0, \ldots, A_i, \ldots, A_{n-1}) = \underline{\Lambda} \cdot \mathbf{H}$ mit der Abkürzung $A_i = \Lambda(\alpha^i)$.

Beide Verfahren werden im folgenden Beispiel angewendet.

Beispiel C: In **Beispiel B** wurde entsprechend den in **Beispiel A** genannten Randbedingungen ermittelt, dass $r = 2$ Symbolfehler vorliegen und die ELP–Koeffizienten $\lambda_0 = \alpha^2$ und $\lambda_1 = \alpha^6$ lauten. Damit ergibt sich das *Error Locator Polynom*:

$$\Lambda(x) = x \cdot \left[\lambda_0 + \lambda_1 \cdot x + x^2\right] = x \cdot \left[\alpha^2 + \alpha^6 \cdot x + x^2\right].$$

Entsprechend der Chien–Suche erhält man

$$\begin{aligned}
\Lambda(\alpha^0) &= \alpha^0 \cdot \left[\alpha^2 + \alpha^6 \cdot 1 + 1\right] = \\
&= \alpha^2 + (\alpha^2 + 1) + 1 = 0 \quad \Rightarrow \quad \mathbf{Nullstelle}, \\
\Lambda(\alpha^1) &= \alpha^1 \cdot \left[\alpha^2 + \alpha^6 \cdot \alpha^1 + \alpha^2\right] = \\
&= \alpha^1 \quad \Rightarrow \quad \text{Keine Nullstelle}, \\
\Lambda(\alpha^2) &= \alpha^2 \cdot \left[\alpha^2 + \alpha^6 \cdot \alpha^2 + \alpha^4\right] = \alpha^4 + \alpha^{10} + \alpha^6 = \\
&= (\alpha^2 + \alpha) + (\alpha + 1) + (\alpha^2 + 1) = \\
&= 0 \quad \Rightarrow \quad \mathbf{Nullstelle}.
\end{aligned}$$

Potenzen von α	Polynome in α	Vektoren $k_2\, k_1\, k_0$
$\alpha^{-\infty}=0$	0	0 0 0
$\alpha^0=1$	1	0 0 1
α^1	α	0 1 0
α^2	α^2	1 0 0
α^3	$\alpha+1$	0 1 1
α^4	$\alpha^2+\alpha$	1 1 0
α^5	$\alpha^2+\alpha+1$	1 1 1
α^6	α^2+1	1 0 1

Damit sind die beiden Fehlerpositionen mit $i = 0$ und $i = 2$ gefunden und der Fehlervektor lautet: $\underline{e} = (e_0, 0, e_2, 0, 0, 0, 0)$.

Die Vektorgleichung $\underline{A} = \underline{\Lambda} \cdot \mathbf{H}$ liefert das gleiche Ergebnis in kompakterer Form:

$$\begin{aligned}
\underline{A} &= \underline{\Lambda} \cdot \mathbf{H} = (\alpha^2, \alpha^6, 1, 0) \cdot \begin{pmatrix} 1 & \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 \\ 1 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^1 & \alpha^3 & \alpha^5 \\ 1 & \alpha^3 & \alpha^6 & \alpha^3 & \alpha^5 & \alpha^1 & \alpha^4 \\ 1 & \alpha^4 & \alpha^1 & \alpha^5 & \alpha^2 & \alpha^6 & \alpha^3 \end{pmatrix} = \\
&= (\alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, 1, \alpha^1) + (\alpha^6, \alpha^1, \alpha^3, \alpha^5, 1, \alpha^2, \alpha^4) + \\
&+ (1, \alpha^3, \alpha^6, \alpha^3, \alpha^5, \alpha^1, \alpha^4) = (0, \alpha^1, 0, \alpha^3, \alpha^3, \alpha^5, \alpha^1) \quad \Rightarrow \quad A_0 = A_2 = 0.
\end{aligned}$$

Fortsetzung im **Beispiel D** auf der nächsten Seite.

Schritt (D): Abschließende Fehlerkorrektur

Im letzten Schritt müssen nun nur noch die r Symbolfehler korrigiert werden, deren Positionen nach Beendigung von Schritt (D) bekannt sind:

- Markiert man die Fehlerpositionen im Empfangswort $\underline{y}$ als Auslöschungen E (*Erasures*), so kann das zugehörige Codewort $\underline{z}$ entsprechend der Beschreibung im **Kapitel 2.4** gefunden werden.
- Eine zweite Möglichkeit bietet die Bestimmung des Fehlervektors $\underline{e}$ aus der Gleichung $\underline{e} \cdot \mathbf{H}^{\rm T} = \underline{s}$ und die Korrektur entsprechend $\underline{z} = \underline{y} - \underline{e}$. Diese liegt dem folgenden Beispiel zugrunde.

Beispiel D: In **Beispiel A** wurde das Empfangswort mit $\underline{y} = (\alpha^3, 1, 0, \alpha^1, \alpha^2, \alpha^5, \alpha^5)$ vorgegeben und daraus das Syndrom $\underline{s} = (\alpha^5, \alpha^2, \alpha^3, \alpha)$ ermittelt. Nach den Berechnungen im **Beispiel C** lautet der Fehlervektor $\underline{e} = (e_0, 0, e_2, 0, 0, 0, 0)$. Alle diese Angaben gelten für den RSC $(7, 3, 5)_8$.

Aus $\underline{e} \cdot \mathbf{H}^{\rm T} = \underline{s}$ erhält man nun folgende Bestimmungsgleichungen für die Fehlerwerte e_0 und e_2:

$$\underline{e} \cdot \mathbf{H}^{\rm T} = \begin{pmatrix} e_0 & 0 & e_2 & 0 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 1 & 1 & 1 \\ \alpha^1 & \alpha^2 & \alpha^3 & \alpha^4 \\ \alpha^2 & \alpha^4 & \alpha^6 & \alpha^1 \\ \alpha^3 & \alpha^6 & \alpha^2 & \alpha^5 \\ \alpha^4 & \alpha^1 & \alpha^5 & \alpha^2 \\ \alpha^5 & \alpha^3 & \alpha^1 & \alpha^6 \\ \alpha^6 & \alpha^5 & \alpha^4 & \alpha^3 \end{pmatrix} \stackrel{!}{=} \underline{s} = \begin{pmatrix} \alpha^5 & \alpha^2 & \alpha^3 & \alpha^1 \end{pmatrix}$$

$$\Rightarrow \quad e_0 \cdot (1,1,1,1) + e_2 \cdot (\alpha^2, \alpha^4, \alpha^6, \alpha^1) \stackrel{!}{=} (\alpha^5, \alpha^2, \alpha^3, \alpha^1)$$

$$\Rightarrow \quad e_0 + e_2 \cdot \alpha^2 = \alpha^5, \;\; e_0 + e_2 \cdot \alpha^4 = \alpha^2, \;\; e_0 + e_2 \cdot \alpha^6 = \alpha^3, \;\; e_0 + e_2 \cdot \alpha^1 = \alpha^1.$$

Alle diese Gleichungen führen zum Ergebnis $e_0 = 1$, $e_2 = \alpha^2$. Damit lautet das korrigierte Codewort:

$$\underline{y} = (\alpha^3, 1, 0, \alpha^1, \alpha^2, \alpha^5, \alpha^5),$$
$$\underline{e} = (1, 0, \alpha^2, 0, 0, 0, 0),$$
$$\Rightarrow \quad \underline{z} = \underline{y} - \underline{e} = \underline{y} + \underline{e} = (\alpha^1, 1, \alpha^2, \alpha^1, \alpha^2, \alpha^5, \alpha^5).$$

Potenzen von α	Polynome in α	Vektoren $k_2\, k_1\, k_0$
$\alpha^{-\infty}=0$	0	0 0 0
$\alpha^0=1$	1	0 0 1
α^1	α	0 1 0
α^2	α^2	1 0 0
α^3	$\alpha+1$	0 1 1
α^4	$\alpha^2+\alpha$	1 1 0
α^5	$\alpha^2+\alpha+1$	1 1 1
α^6	α^2+1	1 0 1

Schnelle Reed–Solomon–Decodierung

Die Klasse der Reed–Solomon–Codes wurde bereits im Jahre 1960 durch die Veröffentlichung **[RS60]** eingeführt. Ihre effiziente Decodierung war jedoch erst ein bis zwei Jahrzehnte später möglich.

Auf den letzten Seiten haben wir den so genannten Petersen–Algorithmus inklusive der Chien–Suche am Beispiel des RSC $(7, 3, 5)_8$ demonstriert, der bis zu $t = 2$ Fehler korrigieren kann. Im Mittelpunkt des Decodiervorgangs stand dabei das Aufstellen und Lösen der Schlüsselgleichung (englisch: *Error Locator Polynom*), wobei die Nullstellen eines Grad–2–Polynoms in GF(7) gefunden werden mussten. Sie konnten erkennen, dass diese algebraische Decodierung mit großem Aufwand verbunden ist.

Bei den in Praxis eingesetzten Codes mit großer Codewortlänge n und hoher Korrekturfähigkeit t würde der Decodieraufwand explodieren, wenn nicht schnellere Decodieralgorithmen gefunden worden wären. So müssen beim Reed–Solomon–Code $(255, 223, 33)_{256}$, der schon früh im ESA/NASA–Standard zur Satellitenübertragung genannt wurde, zur Decodierng eines einzigen Codewortes die bis zu $t = 16$ Nullstellen im GF(255) gefunden werden, und das auch noch in Echtzeit.

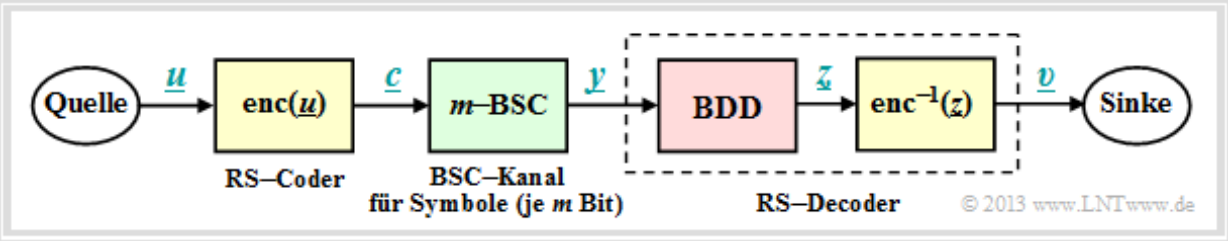
Ab Ende der 1960er Jahre haben sich viele Wissenschaftler um schnellere Decodieralgorithmen für Reed–Solomon–Codes bemüht:

- Beim **Berlekamp–Massey–Algorithmus** (BMA) wird die Schlüsselgleichung $\underline{\Lambda} \cdot \underline{s}^{\rm T} = 0$ als rückgekoppeltes Schieberegister dargestellt, siehe zum Beispiel **[Mas69], [Fri96]** und **[Bos98].** Das Problem wird damit auf die Synthese eines autoregressiven Filters zurückgeführt. Dieser Algorithmus arbeitet wesentlich schneller als der (leichter durchschaubare) Petersen–Algorithmus.
- Etwas später wurde in **[SK+75]** ein Decodierverfahren vorgeschlagen, das auf dem **Euklidischen Algorithmus** basiert. Dieser liefert den größten gemeinsamen Teiler zweier ganzer Zahlen, was zur Decodierung genutzt wird. Der Euklidische Algorithmus ist vergleichbar schnell wie der BMA. Genauere Informationen finden Sie wieder in **[Bos98]** und **[Fri96].**
- Weitere effiziente Decodiernethoden von Reed–Solomon–Codes arbeiten im **Frequenzbereich** unter Verwendung der **Diskreten Fouriertransformation** (DFT) im Körper GF(n).

Die Grundzüge der Reed–Solomon–Fehlerkorrektur wurden bereits in den 1960er Jahren entwickelt. Aber bis in die heutige Zeit (2013) ist die (möglichst schnelle) algebraische Decodierung dieser Codes ein hochaktuelles Forschungsgebiet.

Blockfehlerwahrscheinlichkeit für RSC und BDD

Zur Fehlerwahrscheinlichkeitsberechnung gehen wir vom gleichen Blockschaltbild wie im **Kapitel 2.5** aus, wobei wir uns hier für den Codewortschätzer ($\underline{y} \rightarrow \underline{z}$) auf **Bounded Distance Decoding** (BDD) beschränken. Für *Maximum Likelihood Decoding* sind die Ergebnisse geringfügig besser.



Die Blockfehlerwahrscheinlichkeit sei wie folgt definiert:

$$\Pr(\text{Blockfehler}) = \Pr(\underline{v} \neq \underline{u}) = \Pr(\underline{z} \neq \underline{c}) = \Pr(f > t)\,.$$

Aufgrund der BDD–Annahme ergibt sich das gleiche einfache Ergebnis wie für die binären Blockcodes, nämlich die Wahrscheinlichkeit, dass die Anzahl f der Fehler im Block (Empfangswort) größer ist als die Korrekturfähigkeit t des Codes. Da für die Zufallsgröße f (Fehleranzahl) eine **Binominalverteilung** im Bereich $0 \le f \le n$ vorliegt, erhält man:

$$\Pr(\text{Blockfehler}) = \sum_{f=t+1}^{n} \binom{n}{f} \cdot \varepsilon_{\rm S}{}^{f} \cdot (1-\varepsilon_{\rm S})^{n-f}\,.$$

Während aber im **Kapitel 1** stets $c_i \in \text{GF}(2)$ gegolten hat und damit die f Übertragungsfehler jeweils Bitfehler waren, versteht man bei Reed–Solomon–Codierung unter einem Übertragungsfehler ($y_i \neq c_i$) wegen $c_i \in \text{GF}(2^m)$ bzw. $y_i \in \text{GF}(2^m)$ einen *Symbolfehler*. Damit ergeben sich folgende Unterschiede:

- Das diskrete Kanalmodell zur Beschreibung der binären Blockcodes ist der **Binary Symmetric Channel** (BSC). Jedes Bit c_i eines Codewortes wird mit der Wahrscheinlichkeit ε verfälscht ($y_i \neq c_i$) und mit der Wahrscheinlichkeit $1 - \varepsilon$ richtig übertragen ($y_i = c_i$).
- Bei Reed–Solomon–Codierung muss das BSC–Modell durch das m–BSC–Kanalmodell ersetzt werden. Ein Symbol c_i wird mit der Wahrscheinlichkeit $\varepsilon_{\rm S}$ in ein anderes Symbol y_i verfälscht (egal, in welches) und kommt mit der Wahrscheinlichkeit $1 - \varepsilon_{\rm S}$ unverfälscht beim Empfänger an.

Beispiel: Wir gehen vom BSC–Parameter $\varepsilon = 0.1$ aus und betrachten Reed–Solomon–Codesymbole $c_i \in \text{GF}(2^8)$ $\Rightarrow$ $m = 8$, $q = 256$, $n = 255$. Für eine Symbolverfälschung ($y_i \neq c_i$) genügt bereits ein falsches Bit. Oder anders ausgedrückt: Soll $y_i = c_i$ gelten, so müssen alle $m = 8$ Bit des Codesymbols richtig übertragen werden:

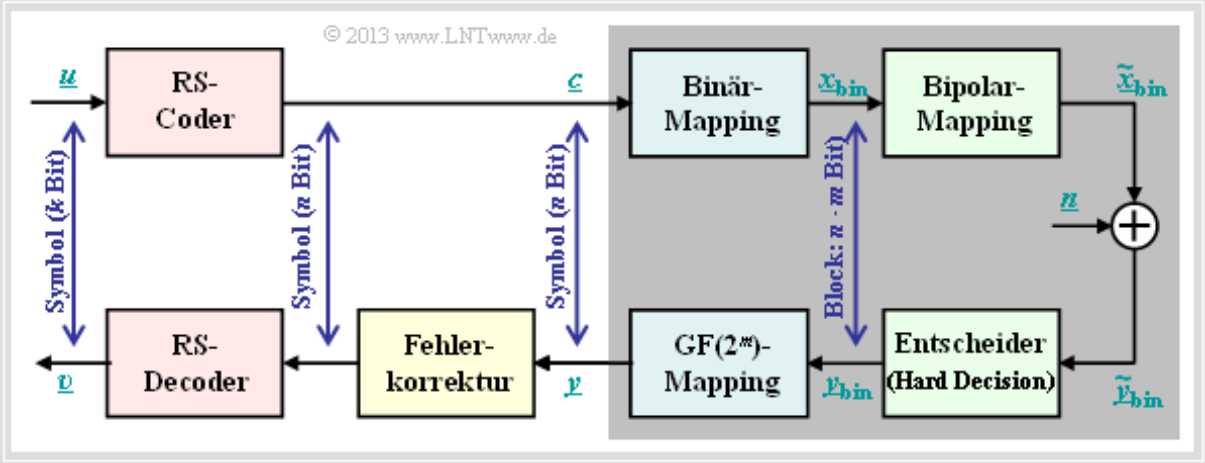
$$1 - \varepsilon_{\rm S} = (1-\varepsilon)^m = 0.9^8 \approx 0.43\,.$$

Damit ergibt sich für das 8–BSC–Modell die Verfälschungswahrscheinlichkeit $\varepsilon_{\rm S} \approx 0.57$. Mit der Annahme, dass die Verfälschung von $c_i = \beta$ in jedes andere Symbol $y_i = \gamma \neq \beta$ gleichwahrscheinlich ist, erhält man $\Pr(y_i = \gamma \mid c_i = \beta) = 0.57/255 \approx 0.223\%$.

Anwendung der Reed–Solomon–Codierung bei binären Kanälen (1)

Die Voraussetzungen für die folgende Berechnung der Blockfehlerwahrscheinlichkeit eines Systems mit Reed–Solomon–Codierung und Umsetzung auf Binärsymbole sind in der Grafik zusammenge- fasst:

- Angenommen wird eine (n, k)–Reed–Solomon–Codierung mit Symbolen aus GF(2^m). Je kleiner die Coderate $R = k/n$ ist, um so weniger Information kann bei fester Datenrate übertragen werden.
- Jedes Symbol wird durch m Bit binär dargestellt ⇒ *Binär–Mapping*. Ein Block (Codewort $\underline{c}$) besteht somit aus n Symbolen bzw. aus $n \cdot m$ Binärzeichen (Bit), die in $\underline{c}_{bin}$ zusammengefasst sind.
- Vorausgesetzt wird außerdem der **AWGN–Kanal,** gekennzeichnet durch den Parameter E_B/N_0. Entsprechend diesem Kanalmodell geschieht die Übertragung bipolar: „0" ↔ „+1", „1" ↔ „–1".
- Der Empfänger trifft harte Entscheidungen (*Hard Decision*) auf Bitebene. Vor der Decodierung inclusive der Fehlerkorrektur werden die Binärsymbole wieder auf GF(2^m)–Symbole umgesetzt.



Die auf der letzten Seite angegebene Gleichung (gültig für *Bounded Distance Decoding*),

$$\Pr(\text{Blockfehler}) = \sum_{f=t+1}^{n} \binom{n}{f} \cdot \varepsilon_S{}^f \cdot (1-\varepsilon_S)^{n-f},$$

basiert auf dem m–BSC–Kanal. Ausgehend vom AWGN–Kanal kommt man

- mit dem **komplementären Gaußschen Fehlerintegral** Q(x) zum BSC–Parameter

$$\varepsilon = \text{Q}\big(\sqrt{2 \cdot E_S/N_0}\big) = \text{Q}\big(\sqrt{2 \cdot R \cdot E_B/N_0}\big),$$

- und daraus zur Verfälschungswahrscheinlichkeit ε_S auf Symbolebene:

$$\varepsilon_S = 1 - (1-\varepsilon)^m.$$

Beispiel: Für $R = k/n = 239/255 = 0.9373$, $10 \cdot \lg E_B/N_0 = 7$ dB ⇒ $E_B/N_0 \approx 5$ und $n = 2^8 - 1$ ⇒ $m = 8$ erhält man für den Parameter ε_S des 8–BSC–Modells:

$$\varepsilon = \text{Q}\big(\sqrt{2 \cdot 0.9373 \cdot 5}\big) = \text{Q}(3.06) \approx 1.1 \cdot 10^{-3}$$

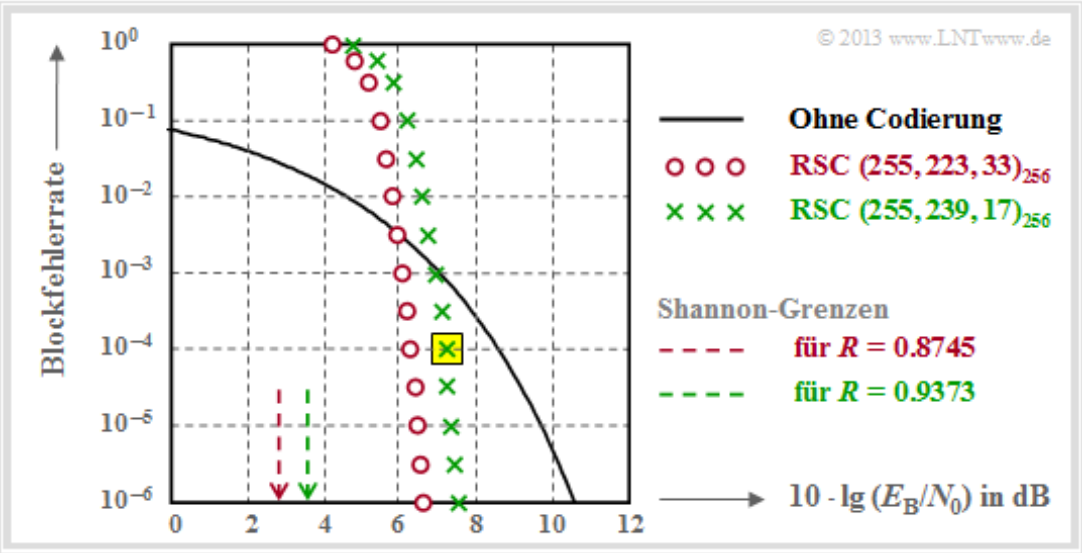
$$\Rightarrow \quad \varepsilon_S = 1 - (1 - 1.1 \cdot 10^{-3})^8 = 1 - 0.9989^8 = 1 - 0.9912 \approx 0.88 \cdot 10^{-2}.$$

Jedes einzelne Symbol wird also mit mehr als 99–prozentiger Wahrscheinlichkeit fehlerfrei übertragen.

Anwendung der Reed–Solomon–Codierung bei binären Kanälen (2)

Die folgende Grafik zeigt die in **[Liv10]** angegebenen Blockfehlerwahrscheinlichkeiten in Abhängigkeit des AWGN–Quotienten $10 \cdot \lg (E_B/N_0)$. Dargestellt sind die berechneten Kurvenverläufe $\Pr(\underline{v} \neq \underline{u})$ für zwei verschiedene Reed–Solomon–Codes entsprechend den *Deep Space Standards* nach CCSDS (*Consultative Committee for Space Data Systems*), nämlich

- der RSC $(255, 239,17)_{256}$, der bis zu $t = 8$ Fehler korrigieren kann,
- der RSC $(255, 223, 33)_{256}$ mit höherer Korrekturfähigkeit ($t = 16$) aufgrund kleinerer Coderate.



Wir analysieren den in der Grafik gelb hinterlegten Punkt, gültig für

- den RSC $(255, 239, 17)_{256}$, und
- $10 \cdot \lg E_B/N_0 = 7.1$ dB $\Rightarrow \varepsilon_S = 0.007$.

Die dazugehörige Blockfehlerwahrscheinlichkeit ergibt sich entsprechend der Grafik zu

$$\Pr(\text{Blockfehler}) = \sum_{f=9}^{255} \binom{255}{f} \cdot \varepsilon_S^{\,f} \cdot (1-\varepsilon_S)^{255-f} \approx 10^{-4}.$$

Dominant ist hierbei der erste Term (für $f = 9$), der bereits etwa 80% ausmacht:

$$\binom{255}{9} \approx 1.1 \cdot 10^{16}, \;\; \varepsilon_S^9 \approx 4 \cdot 10^{-20}, \;\; (1-\varepsilon_S)^{246} \approx 0.18 \;\; \Rightarrow \;\; \Pr(f=9) \approx 8 \cdot 10^{-5}.$$

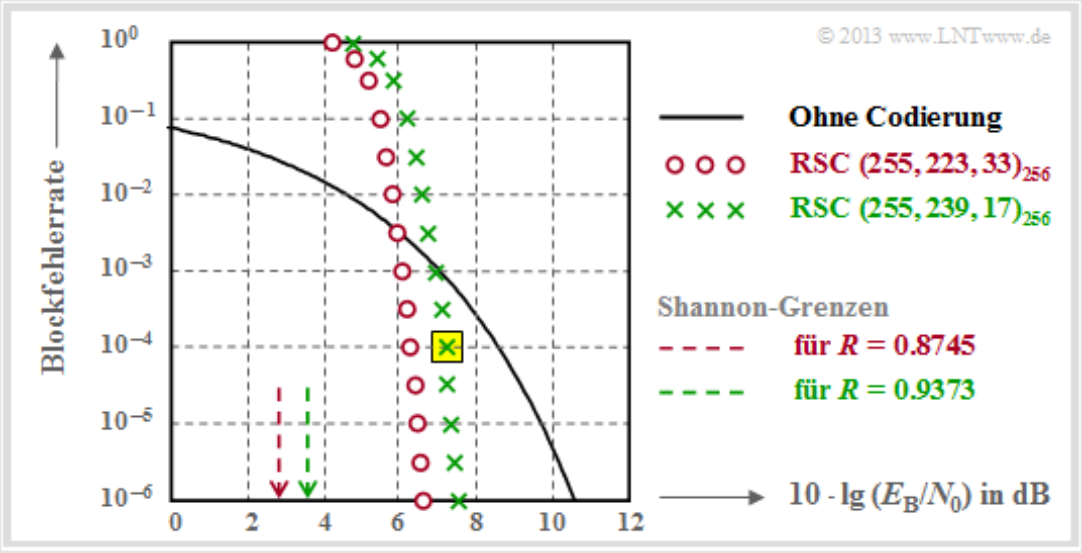
Dies soll als Beleg dafür gelten, dass man die Summation schon nach wenigen Termen abbrechen darf.

Die hier nur angedeutete Berechnung sollen Sie in der **Aufgabe A2.15** für den RSC $(7, 3, 5)_8$ – also für etwas übersichtlichere Parameter – vollständig durchführen.

Anwendung der Reed–Solomon–Codierung bei binären Kanälen (3)

Die in der Grafik dargestellten Ergebnisse kann man wie folgt zusammenfassen:

- Für kleines E_B/N_0 (des AWGN–Kanals) sind die Reed–Solomon–Codes völlig ungeeignet. Beide Codes liegen für $10 \cdot \lg E_B/N_0 < 6$ dB über der Vergleichskurve für uncodierte Übertragung.
- Die Berechung für den RSC $(255, 223, 33)_{256}$ unterscheidet sich von obiger Rechnung nur in der unteren Summationsgrenze ($f_{min} = 17$) und (wegen $R = 0.8745$) durch ein etwas größeres ε_S.
- Dieser ($t = 16$)–Code ist für BER = 10^{-6} auch nur weniger als 1 dB besser als der durch grüne Kreuze gekennzeichnete Code mit $t = 8$. Die Ergebnisse beider Codes sind eher enttäuschend.



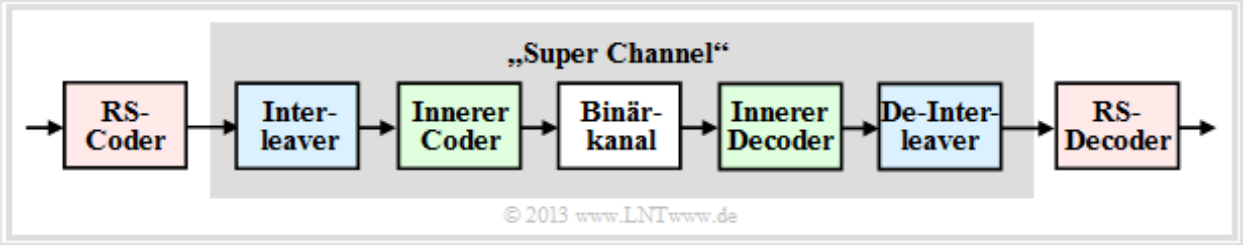
Allgemein gilt:

- Reed–Solomon–Codes sind beim gedächtnislosen Binärkanal (AWGN–Kanal) nicht sehr gut. Beide Codes liegen mehr als 4 dB von der informationstheoretischen **Shannon–Grenze** entfernt.
- Reed–Solomon–Codes sind dagegen sehr wirkungsvoll bei so genannten **Bündelfehlerkanälen.** Sie werden deshalb vorwiegend bei Fadingkanälen, Speichersysteme, usw. eingesetzt.

Hinweis: Die Reed–Solomon–Codes sind nicht perfekt. Welche Konsequenzen sich daraus ergeben, wird in der **Aufgabe A2.16** behandelt.

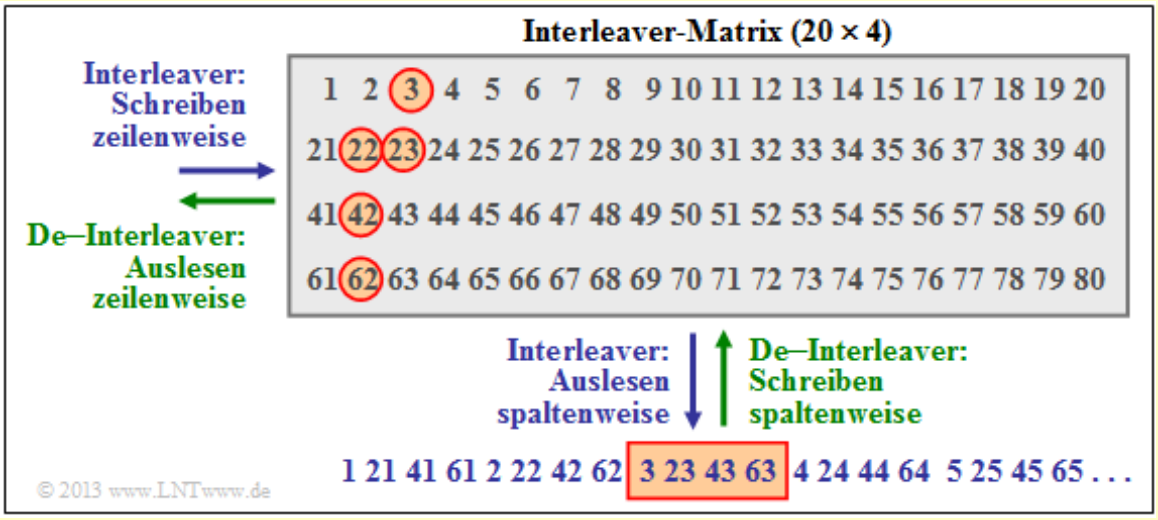
Typische Anwendungen mit Reed–Solomon–Codierung (1)

Reed–Solomon–Codierung wird häufig entsprechend der Grafik zusammen mit einem *inneren Code* in kaskadierter Form angewandt. Der innere Code ist fast immer ein Binärcode und in der Satelliten– und Mobilkommunikation oft ein **Faltungscode.** Will man nur die Reed–Solomon–Codierung/Decodierung untersuchen, so ersetzt man in einer Simulation die grau hinterlegten Komponenten durch einen einzigen Block, den man *Super Channel* nennt.



Besonders effizient ist ein solches **verkettetes** (englisch: *concatenated*) **Codiersystem**, wenn zwischen den beiden Codierern ein Interleaver geschaltet ist, um Bündelfehler weiter zu entspreizen.

Beispiel: Die Grafik zeigt beispielhaft einen solchen Interleaver, wobei wir uns auf eine 20 × 4 Matrix beschränken. In der Praxis sind diese Matrizen deutlich größer.



Der Interleaver wird zeilenweise beschrieben und spaltenweise ausgelesen (blaue Beschriftung). Beim De–Interleaver (grüne Beschriftung) ist die Reihenfolge genau umgekehrt.

- Die Reed–Solomon–Symbole werden also nicht fortlaufend an den inneren Coder weitergeleitet, sondern entsprechend der angegebenen Reihenfolge als Symbol 1, 21, 41, 61, 2, 22, usw. .
- Auf dem Kanal wird ein zusammenhängender Bereich (hier die Symbole 22, 42, 62, 3, 23 ⇒ rot umrandetes Rechteck) zerstört, z. B. durch einen Kratzer auf dem Kanal „Speichermedium”.
- Nach dem De–Interleaving ist die Symbolreihenfolge wieder 1, 2, 3, ... Man erkennt an den rot umrandeten Kreisen, dass nun dieses Fehlerbündel weitgehend „aufgebrochen” wurde.

Typische Anwendungen mit Reed–Solomon–Codierung (2)

Eine sehr weit verbreitete Anwendung von Reed–Solomon–Codierung – und zudem die kommerziell erfolgreichste – ist die **Compact Disc** (CD), deren Fehlerkorrekturmechanismus bereits im **Kapitel 1.1** dieses Buches beschrieben wurde. Hier ist der innere Code ebenfalls ein Reed–Solomon–Code, und das verkette Codiersystem lässt sich wie folgt beschreiben:

- Beide Kanäle des Stereo–Audiosignals werden mit je 44.1 kHz abgetastet und jeder einzelne Abtastwert wird mit 32 Bit (4 Byte) digital dargestellt.
- Die Gruppierung von 6 Samples ergibt einen Rahmen (192 Bit) und damit 24 Codesymbole aus dem Galoisfeld GF(2^8). Jedes Codesymbol entspricht somit genau einem Byte.
- Der erste Reed–Solomon–Code mit der Rate $R_1 = 24/28$ liefert 28 Byte, die einem Interleaver der Größe 28 · 109 Byte zugeführt werden. Das Auslesen erfolgt (kompliziert) diagonal.
- Der Interleaver verteilt zusammenhängende Bytes großräumig über die gesamte Disk. Dadurch werden so genannte Bursts „aufgelöst”, die zum Beispiel durch Kratzer auf der CD herrühren.
- Zusammen mit dem zweiten Reed–Solomon–Code (Rate $R_2 = 28/32$) ergibt sich eine Gesamtrate von $R = (24/28) \cdot (28/32) = 3/4$. Beide Codes können jeweils $t = 2$ Symbolfehler korrigieren.
- Beide RS–Komponentencodes (28, 24, 5) und (32, 28, 5) basieren jeweils auf dem Galoisfeld GF(2^8), was eigentlich eine Codelänge $n = 255$ bedeuten würde.
- Die hier benötigten kürzeren Komponentencodes ergeben sich aus aus dem RSC $(255, 251, 5)_{256}$ durch *Shortening*. Man sieht, dass dadurch die minimale Distanz $d_{min} = 5$ nicht verändert wird.
- Mit der anschließenden **Eight–to–Fourteen–Modulation** und weiterer Kontrollsymbole kommt man schließlich zur endgültigen Coderate 192/588 ≈ 0.326 der CD–Daten- komprimierung.

Auf der Seite **Geschlitzte CD** kann man sich anhand eines Audio–Beispiels von der Korrekturfähigkeit dieser *cross–interleaved Reed–Solomon–Codierung* überzeugen, aber auch deren Grenzen erkennen.