บทที่ 2

รหัสควบคุมความผิดพลาดของข้อมูล

2.1 พีชคณิตสำหรับรหัสควบคุมความผิดพลาดของข้อมูล

การหารหัสควบคุมความผิดพลาดของข้อมูลที่ดี มักจะใช้หลักของพีชคณิต ซึ่งรหัสที่สำคัญหลายๆ ชนิดมีโครงสร้างของโพลีโนเมียล ริง (Polynomial ring) และ กาโลอิส ฟิลด์ (Galois field) โดย ริง เป็นเซ็ตของสมาชิกทางคณิตศาสตร์ที่สามารถทำการบวก ลบ และคูณได้ และ ฟิลด์ เป็นเซ็ตของสมาชิกทางคณิตศาสตร์ที่สามารถทำการบวก ลบ คูณและหารได้ ซึ่งการบวก การลบ การคูณและหารนี้ ไม่ใช่การกระทำทางคณิตศาสตร์โดยทั่วไป เช่น ฟิลด์ที่มีสมาชิก 2 ตัว คือ 0 และ 1 และมีการกระทำทางบวกและคูณ โดย

$0 \oplus 0 = 0$ $\quad$ $0 \odot 0 = 0$

$0 \oplus 1 = 1$ $\quad$ $0 \odot 1 = 0$

$1 \oplus 0 = 1$ $\quad$ $1 \odot 0 = 0$

$1 \oplus 1 = 0$ $\quad$ $1 \odot 1 = 1$

เรียกว่า การบวกแบบมอดูโล 2 (Modulo-2 addition) และการคูณแบบมอดูโล 2 (Modulo-2 multiplication)

ซึ่ง $1 \oplus 1 = 0$ จะได้ว่า $-1 = 1$ และเมื่อ $1 \odot 1 = 1$ ก็จะได้ว่า $1^{-1} = 1$

โดยสามารถทำการลบและหารได้ ยกเว้นการหารด้วย 0 ซึ่งฟิลด์ที่มีสมาชิกเพียง 2 ตัว จะหมายถึง GF(2)

2.1.1 กรุ๊ป (Groups)

นิยามที่ 2.1.1.1 กรุ๊ป G เป็นเซ็ต (Set) ซึ่งมีการกระทำบนคู่ของสมาชิกในเซ็ตแสดงได้โดยสัญญลักษณ์ * ซึ่งสอดคล้องกับคุณสมบัติ 4 ข้อคือ

2.1.1.1.1 คุณสมบัติปิด (Closure)

สำหรับทุกๆ a,b ในเซ็ต $c = a*b$ จะยังคงอยู่ในเซ็ต

2.1.1.1.2 คุณสมบัติของการสลับที่ (Associativity) สำหรับทุกๆ a,b,c ในเซ็ต

a*(b*c) = (a*b)*c

2.1.1.1.3 คุณสมบัติของเอกลักษณ์ (Identity) จะมีสมาชิก e เรียกว่าสมาชิกเอกลักษณ์ (Identity element) ซึ่งสอดคล้องกับ

a*e = e*a = a สำหรับทุกๆ a ที่อยู่ในเซ็ต

2.1.1.1.4 คุณสมบัติการมีอินเวอร์ส (Inverse) ถ้า a อยู่ในเซ็ต ดังนั้นจะมีบางสมาชิก b ที่อยู่ในเซ็ต เรียกว่าอินเวอร์สของ a ซึ่ง

a*b = b*a = e

ถ้า G มีจำนวนสมาชิกจำกัด จะเรียกว่าไฟไนท์กรุ๊ป (Finite Group) และจำนวนของสมาชิกใน G เรียกว่าออเดอร์ (Order) ของ G ไฟไนท์กรุ๊ปบางกรุ๊ปซึ่งมีคุณสมบัติของการจัดกลุ่มโดย a*b = b*a จะเรียกว่าคอมมิวเททีฟกรุ๊ป (Commutative group) หรืออาบิเลียนกรุ๊ป(Abelian Group)

2.1.2 ริง เป็นเซ็ตซึ่งเป็นอาบิเลียน กรุ๊ป โดย

นิยามที่ 2.1.2.1 ริง R เป็นเซ็ตซึ่งมีการกระทำได้ 2 แบบ คือ การบวกและการคูณ โดย

2.1.2.1.1 R เป็นอาบิเลียนกรุ๊ป ภายใต้การบวก

2.1.2.1.2 คุณสมบัติปิด สำหรับ a,b ใดๆ ใน R ผลคูณของ a,b จะอยู่ใน R ด้วย

2.1.2.1.3 คุณสมบัติของการสลับที่ โดย a(bc) = (ab)c

2.1.2.1.4 คุณสมบัติการกระจาย โดย

a(b+c) = ab+ac

(b+c)a = ba+ca

ภายในอินทิเจอร์ริง (Integer ring) ใดๆ เลขจำนวนเต็มบวกที่มีค่ามากกว่าหรือเท่ากับ p ซึ่งจะหารได้ด้วย $\pm p$ หรือ ± 1 เท่านั้น เรียกว่าเป็นไพร์มอินทิเจอร์ (Prime integer)

นิยามที่ 2.1.2.2 ตัวหารร่วมมากของเลขจำนวนเต็ม r และ s จะได้จาก GCD(r,s) ซึ่งเป็นค่าจำนวนเต็มบวกที่มากที่สุดที่สามารถหาร r

และ s ได้ลงตัว

นิยามที่ 2.1.2.3 ตัวหารร่วมน้อยของเลขจำนวนเต็ม r และ s จะได้จาก LCM(r,s) ซึ่งเป็นค่าจำนวนเต็มบวกที่น้อยที่สุดที่ r และ s จะหารได้ลงตัว

เลขจำนวนเต็มใดๆ จะเรียกว่าเป็น รีเรทีฟลี่ไพร์ม (Relatively prime) ก็ต่อเมื่อ ค่า GCD เป็น 1

2.1.3 ฟิลด์

นิยามที่ 2.1.3.1 ฟิลด์ Fเป็นเซ็ตซึ่งมีการกระทำได้ 2 แบบ คือ การบวกและการคูณ โดย

2.1.3.1.1 เซ็ต จะเป็นอาบิเลียนกรุ๊ป ภายใต้การบวก

2.1.3.1.2 ฟิลด์มีคุณสมบัติปิด ภายใต้การคูณ และเซ็ตของสมาชิกที่ไม่เป็น 0 จะเป็นอาบิเลียน กรุ๊ป ภายใต้การคูณ

2.1.3.1.3 คุณสมบัติการกระจาย โดย

$$(a+b)c = ac+bc$$

สำหรับทุก a,b,c ที่อยู่ในฟิลด์

เอกลักษณ์ของการบวก คือ 0 ซึ่งมีอินเวอร์สการบวกของ a คือ -a

เอกลักษณ์ของการคูณ คือ 1 ซึ่งมีอินเวอร์สการคูณของ a คือ a^{-1}

ดังนั้นฟิลด์ซึ่งมีสมาชิกจำนวน q จะเรียกว่า ไฟไนท์ฟิลด์ หรือ กาไลอิสฟิลด์ ซึ่งเขียนได้ในรูปของ GF(q) (Coates 1982: 246) โดยมีคุณสมบัติดังนี้

2.1.3.1 คุณสมบัติของการสลับที่ คุณสมบัติของการกระจาย (Distributive) และการจัดกลุ่ม (Commutative)

2.1.3.2 สมาชิกสามารถมีการกระทำทางการบวก และการคูณได้ โดยผลลัพธ์ยังคงเป็นสมาชิกอยู่ใน GF(q)
เช่น 1 + 1 = 0 ใน GF(2)

2.1.3.3 สมาชิกในฟิลด์ประกอบด้วย สมาชิกที่เป็นเอกลักษณ์การบวกคือ 0 และเอกลักษณ์การคูณคือ 1
โดย a + (-a) = 0 และ a . 1 = a

2.1.3.4 สำหรับการลบและการหาร มีอินเวอร์สการบวกคือ $-a$ และอินเวอร์สการคูณคือ a^{-1}

โดย $a + (-a) = 0$ และ $a \cdot (a^{-1}) = 1$

ฟิลด์ที่เป็นกาโลอิสฟิลด์นั้น จึงเป็นฟิลด์ที่มีเพียงหนึ่งเดียว สำหรับแต่ละค่าของ q

2.1.4 เวคเตอร์สเปซ (Vector space)

นิยามที่ 2.1.4.1 ให้ F เป็นฟิลด์ ซึ่งสมาชิกของ F เรียกว่าสเกลาร์ (Scalar) เซ็ต V จะเรียกว่าเป็น เวคเตอร์สเปซ และสมาชิกของ V เรียกว่าเวคเตอร์ ซึ่งมีการกระทำได้ 2 แบบคือ การบวกแบบเวคเตอร์ (Vector addition) และ การคูณแบบสเกลาร์ (Scalar multiplication) โดยมีของการกระทำคือ

2.1.4.1.1 V เป็น อาบิเลียนกรุ๊ป ภายใต้การบวกแบบเวคเตอร์

2.1.4.1.2 มีคุณสมบัติ การกระจาย สำหรับเวคเตอร์ v_1, v_2 ใดๆ และสเกลาร์ c ใดๆ โดย

$$c(v_1+v_2) = cv_1+cv_2$$

2.1.4.1.3 มีคุณสมบัติ การกระจาย สำหรับเวคเตอร์ v ใดๆ และสเกลาร์ c_1, c_2 ใดๆ โดย

$$1v = v \text{ และ } (c_1+c_2)v = c_1v+c_2v$$

2.1.4.1.4 คุณสมบัติการสลับที่สำหรับเวคเตอร์ v และสเกลาร์ c_1, c_2 ใดๆ โดย

$$(c_1c_2)v = c_1(c_2v)$$

โดยสมาชิกของ v ที่เป็น 0 จะเรียกว่าจุดกำเนิด (Origin) ของ V และเขียนได้ในรูป 0

ให้ V เป็นเซ็ตของไพลีไนเมียล K ซึ่งมีสัมประสิทธิ์ เป็นสมาชิกอยู่ใน GF(q) ถ้าให้ $F = GF(q)$ ซึ่งในสเปซนี้เวคเตอร์จะเป็นไพลีไนเมียล โดยในเวคเตอร์สเปซ V ผลบวกของ U โดยที่

$U = a_1v_1 + a_2v_2 + \ldots + a_kv_k$ เมื่อ a_i เป็นสเกลาร์ใดๆ

จะเรียกว่าเป็นผลบวกเชิงเส้น (Linear combination) ของเวคเตอร์ $v_1, \ldots, v_k$ และเซ็ตของเวคเตอร์ $\{v_1, \ldots, v_k\}$ จะเรียกว่าขึ้นต่อกัน (Linearly dependent)

ถ้ามีเซ็ตของสเกลาร์ $\{a_1, \ldots, a_k\}$ ที่ไม่เป็น 0 โดยที่

$$a_1v_1 + a_2v_2 + \ldots + a_kv_k = 0$$

2.1.5 พีชคณิตเชิงเส้น (Linear algebra)

ในการนำพีชคณิตมาประยุกต์ใช้งานกับรหัสควบคุมความผิดพลาดของข้อมูลนั้น ส่วนใหญ่จะใช้คุณสมบัติของฟิลด์ ซึ่งแถวของเมตริกซ์ (Matrix) A ใดๆ ขนาด $n \times m$ บน GF(q) อาจเขียนได้ในรูปของเซ็ตของเวคเตอร์ใน $GF(q)^m$ ซึ่งมีความยาวรหัสเป็น m และในทำนองเดียวกัน คอลัมน์ของ A อาจเขียนได้เป็นเซ็ตของเวคเตอร์ใน $GF(q)^n$ ซึ่งมีความยาว n

2.1.6 รหัสแบบบล็อกเชิงเส้น (Linear block code)

2.1.6.1 โครงสร้างของรหัสแบบบล็อกเชิงเส้น

เซ็ตของ n ทูเบิล (Tuple) ของสมาชิก จาก GF(q) จะเป็นเวคเตอร์สเปซ เรียกว่า $GF(q)^n$ ซึ่งในที่นี้จะหมายถึง $GF(2)^n$ ซึ่งเป็นเวคเตอร์สเปซของทุกเวิร์ดของเลขฐานสอง ซึ่งมีความยาวรหัสเป็น n

นิยามที่ 2.1.6.1 รหัสเชิงเส้น เป็นสับสเปซ (Subspace) ของ $GF(q)^n$ โดยรหัสเชิงเส้นเป็นเซ็ตที่ไม่ว่างของ n ทูเบิลบน GF(q) เรียกว่าโคดเวิร์ด โดยผลบวกของ 2 โคดเวิร์ดใดๆ จะยังคงเป็นโคดเวิร์ด และผลคูณของโคดเวิร์ดใดๆ กับสมาชิกในฟิลด์ยังคงเป็นโคดเวิร์ดเสมอ

นิยามที่ 2.1.6.2 น้ำหนักของแฮมมิ่ง (Hamming weight) หรือ w(c) ของโคดเวิร์ด c คือ จำนวนของสมาชิกที่ไม่เป็น 0 ในโคดเวิร์ด ซึ่งน้ำหนักที่น้อยที่สุด w^* ของรหัสคือ น้ำหนักที่เล็กที่สุดของโคดเวิร์ดใดๆที่ไม่เป็น 0

ทฤษฎีที่ 2.1.6.3 สำหรับรหัสเชิงเส้น ดิสแทนส์ที่น้อยที่สุด d^* จะสอดคล้องกับ

$$d^* = \min w(c) = w^* .$$

ดังนั้น การหารหัสเชิงเส้น ที่สามารถแก้ความผิดพลาดจำนวน t บิตใดๆ ได้ จึงต้องหารหัสเชิงเส้น ซึ่งมีน้ำหนักที่น้อยที่สุดที่สอดคล้องกับอสมการ

$$w^* \geq 2t+1$$

และเนื่องจาก ค่าดิสแทนส์ที่น้อยที่สุด (Minimum distance) d^* ได้จากนิยามดังต่อไปนี้คือ

นิยามที่ 2.1.6.4 ค่าดิสแทนส์ที่น้อยที่สุด หรือ $d(x,y)$ ระหว่างลำดับของ 2 q-ary ของ x และ y ซึ่งมีความยาว n คือจำนวนของตำแหน่งที่แตกต่างกัน เช่น

$x = 1\ 0\ 1\ 0\ 1$

$y = 0\ 1\ 1\ 0\ 0$

ดังนั้น $d(10101,01100) = 3$

นิยามที่ 2.1.6.5 ให้ C มีค่าเป็น

$C = \{ c_i, i = 0,\ldots, M-1\}$ เป็นรหัส

ดังนั้น ค่าดิสแทนส์ที่น้อยที่สุดของ C คือ แฮมมิ่งดิสแทนส์ของคู่ของโคดเวิร์ด ที่มีค่าของแฮมมิ่งดิสแทนส์ที่น้อยที่สุด คือ

$d^* = \min d(c_i,c_j)$

ดังนั้น โคดเวิร์ดใดๆ ที่มีบิตแตกต่างกัน (Corresponding bits differ) ซึ่งได้จากการทำเอกคลูซีพออร์ (Exclusive OR) ของ 2 โคดเวิร์ดนั้น แล้วนับจำนวนบิตที่เป็น 1 ก็จะเป็นจำนวนตำแหน่งของบิตใน 2 โคดเวิร์ดที่แตกต่างกัน เรียกว่าดิสแทนส์ (Distance) d หรือ แฮมมิ่งดิสแทนส์ (Hamming Distance) ซึ่งค่าดิสแทนส์ที่น้อยที่สุดนั้น จะได้จาก 2 โคดเวิร์ดที่คล้ายคลึงกันมากที่สุด ซึ่งแสดงความสัมพันธ์ได้ดังสมการ

$d^* = \min d(c_i, c_j)$ โดย c_i เป็นโคดเวิร์ดที่ i.

และ $i \neq j$

2.1.6.2 รหัสแบบบล็อกเชิงเส้นในรูปของเมตริกซ์

เนื่องจาก รหัสเชิงเส้น C เป็นสับสเปซของ $GF(q)^n$ และเซ็ตของเบสิสเวคเตอร์ (Basis Vector) ใดๆ สำหรับสับสเปซ จะสามารถใช้เป็นแถวในเมตริกซ์ เพื่อให้ได้เมตริกซ์ G ขนาด $k \times n$ เรียกว่าเจเนเรเตอเมตริกซ์ (Generator matrix) ของรหัส โดยมีสเปซของแถวของ G เป็นรหัสเชิงเส้น C และโคดเวิร์ดใดๆ จะเป็นผลบวกเชิงเส้นของแถวของ G ดังนั้นเซ็ตของ q^k โคดเวิร์ด จะเรียกว่ารหัสเชิงเส้น (n,k) เพราะว่าแถวของ G เป็นอิสระต่อกัน และจำนวนของแถวเป็นจำนวน k จะเป็นมิติของสับสเปซ โดยมิติของทั้งสเปซ $GF(q)^n$ คือ n

ดังนั้น จะมี q^k โคดเวิร์ด และ q^k ที่แตกต่างกันของ k ทูเบิล บน GF(q) ซึ่งสามารถจับคู่บนเซ็ตของโคดเวิร์ดได้แบบ 1-1 ซึ่งคู่ของ k ทูเบิลและโคดเวิร์ดสามารถใช้สำหรับการเข้ารหัสดังสมการ

$$c = iG$$

โดย i เป็นเวิร์ดของข้อมูล ซึ่งมีสัญญลักษณ์ของข้อมูลจำนวน k หูเบิล ที่จะถูกเข้ารหัส และ c เป็นโคดเวิร์ดที่มี n หูเบิล

เช่น เจเนเรเตอเมตริกซ์

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

และ เวคเตอร์ของข้อมูล $i = [\ 0\ 1\ 1\ ]$

จะสามารถเข้ารหัสได้เป็นโคดเวิร์ด

$$c = [\ 0\ 1\ 1\ ] \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} = [\ 0\ 1\ 1\ 1\ 0\ ]$$

ให้ H เป็นเมตริกซ์ ซึ่งมีเบสิสเวคเตอร์เหล่านี้เป็นแถวอยู่ในเมตริกซ์ ดังนั้น n หูเบิลใน c จะเป็นโคดเวิร์ดก็ต่อเมื่อ c ออโธโกนอล (Orthogonal) กับทุกๆ แถวของเวคเตอร์ของ H นั่นคือ

$$CH^T = 0$$

ซึ่งเป็นวิธีที่จะทดสอบว่าเวิร์ดใดๆ เป็นโคดเวิร์ดหรือไม่ โดยเมตริกซ์ H เรียกว่า เมตริกซ์ตรวจสอบบิตเพิ่ม (Parity-check matrix) ของรหัส ซึ่งเป็นเมตริกซ์ขนาด $(n-k) \times n$ และเนื่องจาก $CH^T = 0$ แสดงว่า c เป็นแถวใดๆ ของ G เราจะได้ $GH^T = 0$ ด้วย ดังนั้นจะได้

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

ซึ่งเป็นทางเลือกหนึ่งของ H และจะสังเกตได้ว่าเมื่อมีทางเลือกของ G ได้มากกว่า 1 ก็จะมีทางเลือกของ H ได้หลายแบบเช่นกัน

ทฤษฎีที่ 2.1.6.2.1 รหัส C ประกอบด้วยโคดเวิร์ดที่ไม่เป็น 0 ซึ่งมีน้ำหนักของแฮมมิ่งเป็น w หรือน้อยกว่า ก็ต่อเมื่อ คอลัมน์

(Column) จำนวน w ของ H เป็นอิสระต่อกัน

ดังนั้น รหัสซึ่งมีน้ำหนักที่น้อยที่สุดจะมีค่ามากกว่า w ได้ ก็ต่อเมื่อ ทุกๆ เซ็ตของ w-1 คอลัมน์ของ H เป็นอิสระต่อกัน

การหารหัส (n,k) ที่สามารถแก้ความผิดพลาดจำนวน t ใดๆ ได้ ก็คือการหาเมตริกซ์ H ขนาด (n-k) x n ซึ่งทุกๆ เซ็ตของคอลัมน์จำนวน 2t จะมีอิสระไม่ขึ้นต่อกัน โดยทั่วไปรหัสเชิงเส้น 2 แบบ ที่เหมือนกันหมดนอกจากตำแหน่งของสมาชิก จะเรียกว่าเป็น รหัสอิควิวาเลนท์ (Equivalent code) โดยรหัสทั้ง 2 จะอิควิวาเลนท์กัน ก็ต่อเมื่อเมตริกซ์ G นั้นมีความสัมพันธ์กัน โดย

2.1.6.2.1 การสลับที่ของคอลัมน์ (Column permutation) และ

2.1.6.2.2 การกระทำทางแถว (Elementary row operation) โดยที่ G สามารถเขียนได้ในรูปของ

$$G = [\, I : P \,]$$

เมื่อ I เป็นเมตริกซ์เอกลักษณ์ขนาด k x k โดยมี P เป็นเมตริกซ์ขนาด k x (n-k) และทุกเจเนเรเตอเมตริกซ์ ที่สามารถเขียนให้อยู่ในรูปนี้ได้ โดยการกระทำทางแถว และการสลับที่ของคอลัมน์จะเรียกว่า รูปแบบซิสเต็มมาติก (Systematic) ของเจเนเรเตอเมตริกซ์

นิยามที่ 2.1.6.2.2 รหัสแบบซิสเต็มมาติก คือ โคดเวิร์ดที่เริ่มต้นด้วย บิตของข้อมูลที่ไม่มีการเปลี่ยนแปลง และตามด้วยบิตที่เหลือ ก็คือ บิตเพิ่ม

ทฤษฎีที่ 2.1.6.2.3 รหัสเชิงเส้นใดๆ จะอิควิวาเลนท์กับ รหัสเชิงเส้นแบบซิสเต็มมาติกเสมอ เช่น

ให้

$$G = \left[\begin{array}{ccc:cc} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{array}\right]$$

$$H = \left[\begin{array}{ccc:cc} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{array}\right]$$

ดังนั้นถ้า $i = [\ 0\ 1\ 1\ ]$

จะถูกเข้ารหัสแบบซิสเต็มมาติกได้เป็น

$$c = [\ 0\ 1\ 1\ 1\ 0\ ]$$

ซึ่งรหัสเชิงเส้นแบบซิสเต็มมาติก (n,k) ใดๆ ที่มีดิสแทนส์ที่น้อยที่สุด d^* จะสามารถเขียนได้เป็น(n,k,d^*) โดย

ทฤษฎีที่ 2.1.6.2.4 ขอบเขตของซิงเกิลตัน (Singleton bound)

ดิสแทนส์ที่น้อยที่สุด หรือน้ำหนักที่น้อยที่สุดของรหัสเชิงเส้น (n,k) ใดๆ จะสอดคล้องกับอสมการ

$$d^* \leq 1+n-k$$

นิยามที่ 2.1.6.2.5 รหัสใดๆ ที่มีดิสแทนส์ที่น้อยที่สุด สอดคล้องกับสมการ

$$d^* = 1+n-k$$

จะเรียกว่าเป็นรหัสที่มีดิสแทนส์มากที่สุด

การถอดรหัสโดยใช้ซินโดรม ของรูปแบบของความผิดพลาดโดยสำหรับเวิร์ดที่ได้รับใดๆ จะได้ซินโดรมของ v จากความสัมพันธ์

$$S = vH^T$$

โดยรหัสสามารถแสดงในรูปของพีชคณิต (Algebraic Code) (Stremler 1982: 157) สำหรับข้อมูลจำนวน m บิต ซึ่งสามารถเกิดโคดเวิร์ดในรูปแบบต่างๆ ได้ถึง 2^m แบบนั้น จะสามารถสืบหา และตรวจแก้ความผิดพลาดของข้อมูลให้ถูกต้องได้ โดยการเพิ่มบิตตรวจสอบจำนวน n บิตไปกับข้อมูล ซึ่งแสดงได้ดังนี้คือ

$$a_1\ a_2\ a_3\ \ldots\ a_m\ c_1\ c_2\ c_3\ \ldots\ c_n$$

โดย a_i เป็นบิตของข้อมูลในตำแหน่งที่ i

และ c_i เป็นบิตตรวจสอบในตำแหน่งที่ i

ซึ่งบิตตรวจสอบนี้จะสอดคล้องกับสมการ $[\ H\ ]\ T = 0$

โดย

$$T = [\ a_1\ a_2\ a_3\ \ldots\ a_m\ c_1\ c_2\ c_3\ \ldots\ c_n\ ]$$

$$H = \begin{bmatrix} h_{11} & h_{12} & \dots & h_{1m} & 1 & 0 & 0 & \dots & 0 \\ h_{21} & h_{22} & \dots & h_{2m} & 0 & 1 & 0 & \dots & 0 \\ . & . & \dots & . & . & . & . & \dots & . \\ h_{n1} & h_{n2} & \dots & h_{nm} & 0 & 0 & 0 & \dots & 1 \end{bmatrix}$$

ดังนั้นจะได้สมการ n สมการคือ

$$h_{11}\,a_1 + h_{12}\,a_2 + \dots + h_{1m}\,a_m + c_1 = 0 \quad \dots(1)$$

$$h_{21}\,a_1 + h_{22}\,a_2 + \dots + h_{2m}\,a_m + c_2 = 0 \quad \dots(2)$$

$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

$$h_{n1}\,a_1 + h_{n2}\,a_2 + \dots + h_{nm}\,a_m + c_n = 0 \quad \dots(n)$$

เมื่อตัวรับได้รับโคดเวิร์ด ก็สามารถตรวจสอบได้ว่าบิตใดที่เกิดความผิดพลาด จากการคูณ [H] กับโคดเวิร์ดที่ได้รับมา หากผลลัพธ์ไม่เป็น 0 ก็แสดงว่ามีความผิดพลาดเกิดขึ้น โดยความผิดพลาดของข้อมูลที่เกิดขึ้นสามารถแสดงได้ ในรูปของสมการ

$R = T + E$ โดย R เป็นโคดเวิร์ดที่ได้รับ

และ E เป็นความผิดพลาดที่เกิดขึ้น

และเนื่องจาก $[H]R = [H]T + [H]E$

$= [H]E$ ซึ่งเรียกว่า S หรือ ซินโดรม (Syndrome) ที่จะบอกตำแหน่งของบิตที่เกิดความผิดพลาด เพื่อให้สามารถตรวจแก้ความผิดพลาดของข้อมูลให้ถูกต้องได้

2.1.6.3 การดัดแปลงรหัสเชิงเส้น

เราสามารถเปลี่ยนแปลงรูปแบบของ รหัสเชิงเส้นให้เป็นรหัสแบบใหม่ โดยที่รหัสนั้นยังคงเป็นรหัสเชิงเส้นอยู่ โดยความยาวบลอก n สามารถเปลี่ยนแปลงได้จากการเปลี่ยนค่า k หรือ (n-k) บิต ซึ่งมีได้ 6 แบบคือ

2.1.6.3.1 การทำเอกซ์แพนดิ้ง (Expanding) ของรหัส เป็นการเพิ่มความยาวรหัส โดยการเพิ่มบิตตรวจสอบความผิดพลาด ซึ่งเป็นการเพิ่มมิติของเจเนเรเตอเมตริกซ์

2.1.6.3.2 การทำเลนท์เทนนิ่ง (Lengthening) ของรหัส เป็นการเพิ่มความยาวรหัสโดยการเพิ่มจำนวนบิตของข้อมูล

ซึ่งเป็นการเพิ่มมิติทั้ง 2 ของเจเนเรเตอเมตริกซ์

2.1.6.3.3 การทำพังเชอริ่ง (Puncturing) ของรหัส เป็นการลดความยาวรหัส โดยการลดบิตตรวจสอบความผิดพลาด ซึ่งเป็นการลดมิติของเจเนเรเตอเมตริกซ์

2.1.6.3.4 การทำชอทเทนนิ่ง (Shortening) ของรหัส เป็นการลดความยาวรหัส โดยการลดจำนวนบิตของข้อมูล ซึ่งเป็นการลดมิติทั้ง 2 ของเจเนเรเตอเมตริกซ์

2.1.6.3.5 การทำอากเมนติ้ง (Augmenting) ของรหัส เป็นการเพิ่มจำนวนบิตของข้อมูล โดยไม่เปลี่ยนแปลงความยาวรหัสคือ การเพิ่มมิติส่วนของข้อมูลในเจเนเรเตอเมตริกซ์

2.1.6.3.6 การทำเอกเพอเกติ้ง (Expurgating) ของรหัส เป็นการเพิ่มจำนวนบิตของข้อมูลโดยไม่เปลี่ยนแปลงความยาวรหัส คือ การลดมิติส่วนของข้อมูลในเจเนเรเตอเมตริกซ์

ซึ่งวิธีการเหล่านี้สามารถใช้เพื่อจัดรูปแบบของรหัสให้เหมาะสมกับงานที่จะนำไปประยุกต์ใช้

2.1.7 การคำนวณทางคณิตศาสตร์ของกาโลอิสฟิลด์

2.1.7.1 อัลกอริทึมสำหรับการหาร

ทฤษฎีที่ 2.1.7.1.1 อัลกอริทึม สำหรับการหาร สำหรับทุกๆ คู่ของเลขจำนวนเต็ม

c และ d โดย $d \neq 0$

จะมีได้เพียงค่า Q ซึ่งเป็นเลขจำนวนเต็มที่เป็นผลหาร (Quotient) และเศษเหลือเป็น s โดย

$$c = dQ + s \quad \text{เมื่อ } 0 \leq s \leq |d|$$

ซึ่งเศษเหลือสามารถเขียนได้ในรูปความสัมพันธ์ของ

$$s = R_d[c] = c(\text{mod } d)$$

และในทำนองเดียวกันจะได้อัลกอริทึม สำหรับการหารของโพลีโนเมียล c(x) โดย

$$c(x) = d(x)Q(x) + s(x)$$

โดยดีกรี (Degree) ของ s(x) จะน้อยกว่า ดีกรีของ d(x) และ

$$S(x) = R_{d(x)}[c(x)] = c(x)(\text{mod } d(x))$$

2.1.7.2 ไพลีโนเมียลรัง

ไพลีโนเมียลบนฟิลด์ GF(q) สามารถเขียนในรูปคณิตศาสตร์ได้เป็น

$$f(x) = f_{n-1}x^{n-1} + f_{n-2}x^{n-2} + \ldots + f_1x + f_0$$

โดย x เป็นตัวแปรใดๆ และสัมประสิทธิ์ $f_{n-1},\ldots,f_0$ เป็นสมาชิกของ GF(q) โดยไพลีโนเมียลที่เป็นศูนย์คือ $f(x) = 0$

โมนิคไพลีโนเมียล (Monic polynomial) คือไพลีโนเมียลที่มีสัมประสิทธิ์ $f_{n-1} = 1$

นิยามที่ 2.1.7.2.1 พริมิทีฟไพลีโนเมียล (Primitive polynomial) คือไพร์มไพลีโนเมียลบน GF(q) ซึ่งมีสมาชิกเป็น 0 โดยตัวอย่างของไพร์มไพลีโนเมียล แสดงได้ดังตารางที่ 2.1

ดีกรี	พริมิทีฟไพลีโนเมียล
2	x^2+x+1
3	x^3+x+1
4	x^4+x+1
5	x^5+x^2+1
6	x^6+x+1
7	x^7+x^3+1
8	$x^8+x^4+x^4+x^3+x^2+1$
9	x^9+x^4+1
10	$x^{10}+x^3+1$
11	$x^{11}+x^2+1$
12	$x^{12}+x^6+x^4+x+1$
13	$x^{13}+x^4+x^3+x+1$

ตารางที่ 2.1 แสดงไพร์มไพลีโนเมียลบน GF(2)

ดีกรี	พริมิทีฟโพลีโนเมียล
14	$x^{14}+x^{10}+x^{6}+x+1$
15	$x^{15}+x+1$
16	$x^{16}+x^{12}+x^{3}+x+1$
17	$x^{17}+x^{3}+1$
18	$x^{18}+x^{7}+1$
19	$x^{19}+x^{5}+x^{2}+x+1$
20	$x^{20}+x^{3}+1$
21	$x^{21}+x^{2}+1$
22	$x^{22}+x+1$
23	$x^{23}+x^{5}+1$
24	$x^{24}+x^{7}+x^{2}+x+1$
25	$x^{25}+x^{3}+1$
26	$x^{26}+x^{6}+x^{2}+x+1$
27	$x^{27}+x^{5}+x^{2}+x+1$
28	$x^{28}+x^{3}+1$

ตารางที่ 2.1 แสดงไพร์มโพลีโนเมียลบน GF(2) (ต่อ)

ซึ่งวิธีที่ทดสอบไพร์มทำได้โดยการลองผิดลองถูกของทุกแฟคเตอร์ (Factor) ที่เป็นไปได้

2.2 <u>ทฤษฎีรหัส</u>

ถ้าข้อมูลที่เราสนใจสามารถเขียนแทนได้ด้วยข้อมูลที่เป็นเลขฐานสอง ซึ่งก็คือลำดับ (Sequenec) ของ 0 และ 1 โดยข้อมูลเหล่านี้จะถูกส่งผ่านแชนแนล ซึ่งอาจเป็นต้นเหตุให้เกิดความผิดพลาดขึ้นได้นั้น การใช้รหัสก็คือการเพิ่มสัญลักษณ์พิเศษ (Extra symbol) ไปกับข้อมูล เพื่อให้ความผิดพลาดนั้นสามารถตรวจพบได้ และแก้ไข

ให้ถูกต้องได้ที่ตัวรับ โดยลำดับของข้อมูลจะถูกแทนด้วย ลำดับของสัญญลักษณ์ที่ยาวขึ้น เนื่องจากบิตเพิ่ม (Redundancy) โดยรหัสของเลขฐานสองขนาด M และมีความยาวบลอกเป็น n จะเป็นเซ็ตของเวิร์ดของเลขฐานสองจำนวน M ซึ่งมีความยาว n เรียกว่า โคดเวิร์ด ซึ่งโดยทั่วไป $M = 2^k$ สำหรับเลขจำนวนเต็ม k ใด ๆ และรหัสจะหมายถึงรหัส (n,k)ของเลขฐานสอง เช่น

$$\text{รหัส } C = \begin{bmatrix} 1\,0\,1\,0\,1 \\ 1\,0\,0\,1\,0 \\ 0\,1\,1\,1\,0 \\ 1\,1\,1\,1\,1 \end{bmatrix}$$

มีค่า M = 4 และ n = 5 ซึ่งเป็นรหัสที่แทนข้อมูลที่เป็นเลขฐานสองและมีความยาว 2 บิต โดยที่ 0 0 จะแทนด้วย 1 0 1 0 1

0 1 จะแทนด้วย 1 0 0 1 0

1 0 จะแทนด้วย 0 1 1 1 0

และ 1 1 จะแทนด้วย 1 1 1 1 1

ซึ่งถ้าตัวรับได้รับลำดับของ 5 บิต แบบใดแบบหนึ่งใน 4 แบบนี้ ก็จะสามารถหา 2 บิต ข้อมูลที่ถูกต้องได้ แต่ถ้าเกิดความผิดพลาดขึ้น ตัวรับก็จะได้รับเวิร์ดของ 5 บิตที่แตกต่างไปจาก 4 แบบนี้ ซึ่งสามารถหาเวิร์ดแบบใดแบบหนึ่งใน 4 แบบที่ใกล้เคียงกับเวิร์ดที่ตัวรับได้รับมามากที่สุดเช่นหากได้รับเวิร์ด 0 1 1 0 0 ก็สามารถสรุปได้ว่าข้อมูลที่ส่งมาคือ 0 1 1 1 0 หรือก็คือ 1 0 นั่นเอง ซึ่งรหัสดังกล่าวนี้ไม่ใช่รหัสที่ดีเนื่องจากไม่สามารถแก้ความผิดพลาดได้หลายรูปแบบของความผิดพลาดที่เกิดขึ้น โดยลักษณะของรหัสที่ดีจะมีโคดเวิร์ดที่แตกต่างจากโคดเวิร์ดอื่นๆ ให้มากที่สุดเท่าที่จะเป็นไปได้

นิยามที่ 2.2.1 รหัสแบบบลอกขนาด M ซึ่งแทนด้วยสัญญลักษณ์จำนวน q แบบ และเป็นเซ็ทของ M q-ary ที่มีลำดับความยาว n จะเรียกว่าโคดเวิร์ด และถ้า q = 2 จะเรียกสัญญลักษณ์นี้ว่า บิต ซึ่งโดยทั่วไป $M = 2^k$ สำหรับเลขจำนวนเต็ม k จะเรียกรหัสนี้ว่า รหัส (n,k)

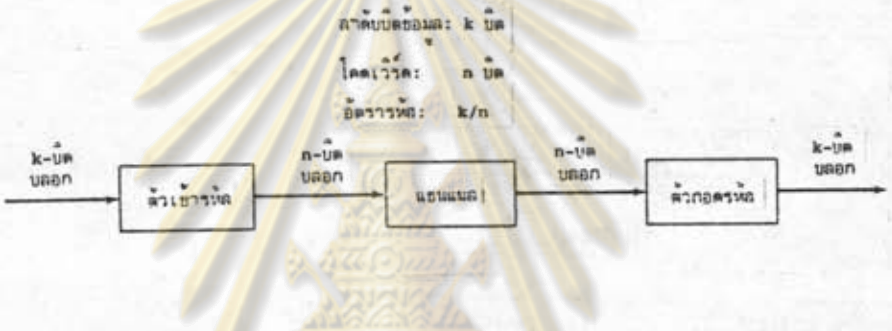
โดยในลำดับของเลขฐานสองจำนวน n จะมีโคดเวิร์ดได้ 2^k แบบ ดังนั้นรหัสจะสามารถมีจำนวนลำดับของเลขฐานสองที่แตกต่างกันได้ถึง $n2^k$ แบบ และมีวิธีที่จะเลือกแบบของเลขฐานสองเหล่านี้ได้ $2^{**}(n2^k)$ วิธี ดังนั้นจำนวนที่แตกต่างกัน ของรหัส (n,k) ก็คือ $2^{**}(n2^k)$ แบบ โดยรหัสสามารถแบ่งได้เป็น

2.2.1 รหัสแบบบลอก (Block Codes)

รหัสแบบบลอกสามารถแสดงได้ด้วย กลุ่มหรือบลอก (Block) ของข้อมูลจำนวน k บิต หรือ n โคดเวิร์ด (Codeword) ซึ่งส่งไปด้วยอัตรา (Rate) R รหัสแบบบลอกจึงแสดงได้ดังสมการ

$$R = k/n$$

และแสดงลักษณะของรหัสแบบบลอกได้ดังรูปที่ 2.1



รูปที่ 2.1 แสดงลักษณะของรหัสแบบบลอก

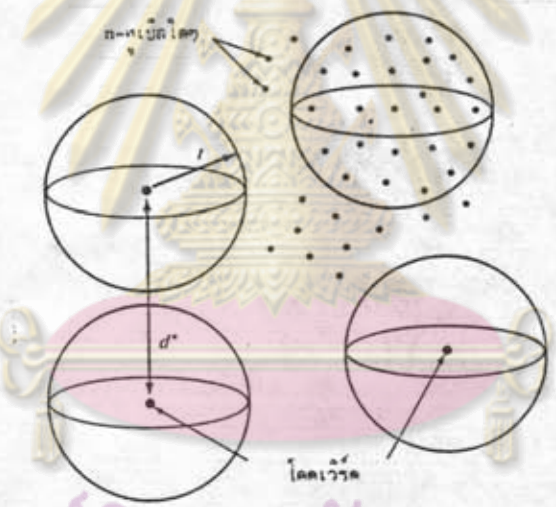
ประสิทธิภาพของรหัสแบบบลอกขึ้นอยู่กับตัวแปร 3 ตัว คือ ความยาวของรหัส (Block Length) n, ความยาวของข้อมูล (Information Length) k และ ค่าดิสแทนส์ที่น้อยที่สุด (Minimum distance) d^*

ดังนั้นรหัสแบบบลอก (n,k) ที่มีค่าดิสแทนส์ที่น้อยที่สุดเป็น d^* จึงสามารถแสดงได้ในรูปของ (n,k,d^*) ซึ่งค่า d^* มีความสัมพันธ์กับจำนวนบิตของข้อมูลที่ผิดพลาด โดยในกรณีที่มีการส่งผ่านโคดเวิร์ด แล้วเกิดความผิดพลาดขึ้น 1 บิต นั่นเวิร์ดที่ได้รับจะมีแฮมมิ่งดิสแทนส์เป็น 1 จากโคดเวิร์ดที่ส่งไป และถ้าดิสแทนส์จากเวิร์ดที่ได้รับเมื่อเทียบกับโคดเวิร์ดอื่นๆ มีค่ามากกว่า 1 แล้ว ตัวถอดรหัสจะสามารถแก้ความผิดพลาดนั้นได้โดยการเลือกโคดเวิร์ดที่ใกล้เคียงกับเวิร์ดที่ได้รับมากที่สุด

โดยทั่วไปถ้าเกิดความผิดพลาดขึ้นเป็นจำนวน t และค่าดิสแทนส์จากเวิร์ดที่ได้รับเมื่อเปรียบเทียบกับโคดเวิร์ดอื่นๆ มีค่ามากกว่า t แล้ว ตัวถอดรหัสจะสามารถแก้ความผิดพลาดนั้นได้ ดังอสมการ

$$d^* \geq 2t+1$$

อธิบายได้โดย ภายในสเปซ (Space) ของทุก n ทูเบิล เซ็ทของ n ทูเบิลจะถูกเลือกให้เป็นโคดเวิร์ด ซึ่งถ้า t เป็นเลขจำนวนเต็มที่มีค่ามากที่สุดที่สอดคล้องกับอสมการ $d^* \geq 2t+1$ แล้ว ส่วนที่ไม่คาบเกี่ยวกันของรัศมี t จะแสดงถึงแต่ละโคดเวิร์ด ซึ่งเวิร์ดที่ได้รับมาจะอยู่ในสเฟียร์ (Sphere) และจะถูกถอดรหัสได้เป็นโคดเวิร์ดที่จุดกึ่งกลาง (Center) ของสเฟียร์นั้น และถ้าเกิดความผิดพลาดน้อยกว่าหรือเท่ากับ t เวิร์ดที่ได้รับก็จะอยู่ในสเฟียร์ที่ถูกต้อง ซึ่งหมายถึงจะถอดรหัสได้ถูกต้องด้วย แต่ในกรณีที่เกิดความผิดพลาดมากกว่า t ซึ่งอาจอยู่ในสเฟียร์การถอดรหัสของโคดเวิร์ดอื่น ก็จะมีผลให้ถอดรหัสผิดได้ แสดงได้ดังรูปที่ 2.2



รูปที่ 2.2 แสดงเสฟียร์ของการถอดรหัส

โดยทั่วไปการถอดรหัสจะมี 2 แบบคือ

การถอดรหัสอย่างไม่สมบูรณ์ (Incomplete decoder) ซึ่งเป็นการถอดรหัส เฉพาะเวิร์ดที่ได้รับมาและอยู่ในสเฟียร์การถอดรหัสของโคดเวิร์ดเท่านั้น ดังนั้นเวิร์ดที่มีความผิดพลาดมากกว่า t ก็จะจัดให้เป็นเวิร์ดที่ไม่สามารถถอดรหัสได้ โดยรูปแบบของความผิดพลาดที่เกิดขึ้นจะไม่สามารถแก้ให้ถูกต้องได้ ซึ่งการถอดรหัสส่วนใหญ่จะใช้วิธีนี้

การถอดรหัสอย่างสมบูรณ์ (Complete decoder) เป็นการถอดรหัสของทุกโคดเวิร์ดที่ได้รับ มาเป็นโคดเวิร์ดที่ใกล้เคียงกับโคดเวิร์ดที่ได้รับมากที่สุด

ซึ่งหากมีความผิดพลาดเกิดขึ้นมากกว่า t แล้ว โดยทั่วไปจะถอดรหัสผิด แต่ก็มีบางโอกาสที่อาจได้ข้อมูลที่ถูก ซึ่งการถอดรหัสแบบนี้มักจะใช้เมื่อเห็นว่า วิธีเดาจากข้อมูลเป็นวิธีที่ดีกว่า ไม่ได้มีการประมาณการจากข้อมูลเลย

รหัสแบบบล็อกได้แก่

2.2.1.1 การใช้บิตเพิ่ม เพื่อตรวจสอบความผิดพลาดของข้อมูล (Parity-check codes) โดยข้อมูลจำนวน k บิต จะมีบิตที่ k+1 เพิ่มเข้าไปเพื่อให้จำนวนของบิตที่เป็น 1 ในโคดเวิร์ดเป็นเลขคู่หรือเลขคี่ ซึ่งจะเป็นรหัส (k+1,k) หรือ (n,n-1) และมีค่าดิสแทนส์ที่น้อยที่สุดเป็น 2 ดังนั้นจึงไม่สามารถแก้ไขความผิดพลาดใดๆ ที่เกิดขึ้นได้ จึงเหมาะสำหรับใช้ตรวจสอบความผิดพลาดของข้อมูลที่มีความผิดพลาดเพียง 1 บิตหรือเป็นจำนวนคี่ของบิตเท่านั้น

ในปัจจุบัน ระบบการสืบหาความผิดพลาดของข้อมูลมีอยู่หลายวิธี (Stalling 1985: 101) ซึ่งวิธีที่ง่ายที่สุดก็คือ การเพิ่มพาริตี้บิตเข้าไปที่ท้ายของข้อมูลที่ต้องการส่ง เพื่อที่ตัวรับจะสามารถทราบได้ว่า กลุ่มของข้อมูลที่ได้รับมานั้นถูกต้องหรือไม่ ซึ่งพาริตี้บิตมีหลายแบบคือ

2.2.1.1.1 การตรวจสอบในแนวดิ่ง (Vertical Redundancy Check) หรือ VRC (Stalling 1985: 104) เป็นการเพิ่ม 1 บิต สำหรับตรวจสอบในแต่ละตัวอักษรไปที่ท้ายของตัวอักษรนั้น ซึ่งอาจจะเป็นพาริตี้แบบคี่ (Odd Parity) หรือ พาริตี้แบบคู่ (Even Parity) เพื่อให้สามารถตรวจสอบจำนวนบิตที่มีค่าเป็น 1 ในตัวอักษรนั้นได้ โดยทั่วไปนิยมใช้พาริตี้แบบคี่มากกว่าพาริตี้แบบคู่ เนื่องจากจะต้องมีบิตที่เป็น 1 อยู่ในข้อมูลที่ส่งไปเสมอ ซึ่งสามารถแสดงในรูปของสมการทางคณิตศาสตร์ได้ดังนี้คือ

พาริตี้ในแนวแถว (Row Parity bit) แบบคู่

$$R_j = b_{1j} \oplus b_{2j} \oplus \ldots \oplus b_{nj}$$

เมื่อ R_j เป็นพาริตี้บิตของตัวอักษรตำแหน่งที่ j

b_{ij} เป็นบิตตำแหน่งที่ i ของตัวอักษรตำแหน่งที่ j

n เป็นจำนวนของบิตในตัวอักษรนั้น

$\oplus$ เป็นการบวกโดยไม่มีบิตทด (Carry bit) หรือการบวกแบบมอดูโล 2 (Modulo-2 Addition)

ในปัจจุบัน ความเร็ว (Speed) ที่ใช้ในการส่งผ่านข้อมูลเพิ่มมากขึ้น ซึ่งเมื่อความเร็วที่ใช้สูงกว่า 2400 บิตต่อวินาที ความผิดพลาดของข้อมูลที่ไม่สามารถสืบหาได้ จากการใช้การตรวจสอบในแนวดิ่ง จะเพิ่มขึ้นมาก และเนื่องจากการเกิดอิมพัลนอยส์ ซึ่งมักก่อให้เกิดความผิดพลาดของข้อมูลมากกว่า 1 บิต การใช้ VRC ซึ่งสามารถสืบหาข้อมูลที่ผิดพลาดได้เฉพาะจำนวนบิตข้อมูลที่ผิดพลาดได้เพียง 1 บิตหรือเป็น จำนวนคี่ (Odd Number) จึงไม่นิยมใช้นัก

2.2.1.1.2 การตรวจสอบในแนวนอน (Longitudinal Redundancy Check) หรือ LRC หรือ การตรวจสอบตัวอักษรแบบบลอก (Block Check Character) หรือ BCC (Stalling 1985: 105)

เป็นการปรับปรุงความสามารถ ในการสืบหาความผิดพลาดของข้อมูลให้เพิ่มขึ้น ซึ่งโดยทั่วไปจะใช้ร่วมกับ VRC เพื่อให้สามารถสืบหาข้อมูลที่ผิดพลาดเป็นจำนวนคู่ (Even Number) ได้ โดยเพิ่มพาริตี้บิตอีก 7 บิตเข้าไปที่ท้ายข้อมูล (ในกรณีที่ 1 ตัวอักษรแทนด้วยบิตจำนวน 7 บิต) เพื่อให้เป็นพาริตี้แบบคี่ หรือแบบคู่ในแนวแถวของบิตข้อมูลที่ต้องการส่ง ซึ่งสามารถแสดงในรูปของสมการทางคณิตศาสตร์ได้ดังนี้คือ

พาริตี้บิตสำหรับตรวจสอบตัวอักษร (Parity Check Character) แบบคู่

$$C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$$

เมื่อ C_i เป็นบิตตำแหน่งที่ i ของพาริตี้บิตสำหรับตรวจสอบอักษร

m เป็นจำนวนตัวอักษรที่ตรวจสอบ

การใช้ VRC/LRC นี้ จะใช้เมื่อมีการส่งผ่านข้อมูลด้วยความเร็วไม่เกิน 4800 บิตต่อวินาที ซึ่งหากใช้ความเร็วมากกว่านี้ โอกาสที่จะสืบหาความผิดพลาดของข้อมูลไม่พบ จะเพิ่มมากขึ้นจนไม่สามารถยอมรับได้ (โดยทั่วไปค่าที่ยอมรับได้ไม่ควรเกิน 2^{-n}) นอกจากนี้ การส่งผ่านข้อมูลในปัจจุบันนิยมส่งเป็นบิต (Bit Oriented) มากกว่าการส่งเป็นตัวอักษร (Character Oriented) การใช้พาริตี้แบบกลุ่ม (Block Parity) เพื่อสืบหาความผิดพลาดของข้อมูล จึงเป็นที่นิยมใช้มากกว่า ซึ่งพาริตี้แบบกลุ่ม ได้แก่

2.2.1.1.3 การตรวจสอบโดยสไปรัล (Spiral Redundancy Check) หรือ SRC (Sherman 1985: 148)

เป็นการใช้เทคนิค ในการสืบหาความผิดพลาดของข้อมูลเช่นเดียวกับ LRC แต่ตำแหน่งของบิตที่นำมาคำนวณหาพาริตี้บิตนั้น

อยู่ในแนวเส้นทแยงมุม แทนที่จะเป็นแนวนอน ซึ่งการทำ SRC นี้ แม้ว่าข้อมูลที่ต้องการส่งจะเป็นบิต แต่เมื่อครบ 8 บิตก็จะทำการคำนวณพาริตี้บิต ซึ่งอาจจะเป็นพาริตี้แบบคี่หรือแบบคู่ แล้วส่งพาริตี้บิตนั้นไปที่ท้ายข้อมูล และหากข้อมูลมีจำนวนบิตไม่ครบ 8 บิต ก็จะเพิ่ม (Pad Out) 8 บิตสุดท้ายให้ครบก่อนการคำนวณ โดยทั่วไปนิยมใช้ SRC กับการส่งผ่านข้อมูลที่เป็นตัวอักษรเดี่ยว (Single Character) การนำมาใช้งานจึงไม่กว้างขวางนัก

2.2.1.1.4 อินเทอลีฟวิ่ง (Interleaving) (Sherman 1985: 148)

เป็นการนำบิตแต่ละตำแหน่ง ของแต่ละตัวอักษรของข้อมูลมาไว้ด้วยกัน จนครบ 8 บิตของ 8 ตัวอักษรที่จะส่งไป ซึ่งในตัวอักษรที่ 9 ก็จะเริ่มการกระทำเช่นนี้ใหม่ และหากจำนวนตัวอักษรไม่ครบ 8 ก็จะทำการเพิ่มให้จนครบ ซึ่งเป็นวิธีที่ดีกว่าการเพิ่มพาริตี้โดยตรง (Straight BCC) แต่ก็ต้องจัดการกับข้อมูลมาก

2.2.1.2 การใช้บิตซ้ำ (Repetition codes) (Tranter 1983: 256)

เป็นรหัสแบบบลอกอย่างง่ายๆ โดยข้อมูล 1 บิต จะถูกกำหนดให้เป็นรหัสที่ซ้ำกัน n ครั้ง และโดยทั่วไป n จะเป็นเลขคี่ ดังนั้น จะมีบิตจำนวน n-1 บิต ที่เป็นบิตเพิ่มในแต่ละบิตของข้อมูล โดยมีอัตราของรหัสคือ 1/n ซึ่งอัตราของรหัสจะน้อยลง เมื่อค่า n มีขนาดใหญ่ แสดงได้ดังตารางที่ 2.2

อัตรา	d*	ความสามารถแก้ความผิดพลาดของข้อมูลต่อโคดเวิร์ด	โคดเวิร์ด
1/3	3	1	000 111
1/5	5	2	00000 11111
:	:	:	:
1/n	n	1/2(n-1)	00...0 11...1

ตารางที่ 2.2 แสดงลักษณะของการใช้บิตซ้ำ

จะเห็นได้ว่า ค่าดิสแทนส์ที่น้อยที่สุดคือ n และเมื่อค่า n มีค่ามากขึ้น การใช้บิตซ้ำจะมีประสิทธิภาพในการแก้ความผิดพลาด ของบิตที่ผิดพลาดได้มากขึ้นถึง 1/2(n-1) แต่ถ้าเปรียบเทียบกับอัตรารหัสของการใช้บิตซ้ำที่มีค่าน้อยมากแล้ว บิตซ้ำจึงไม่เป็นที่นิยมใช้

2.2.1.3 รหัสแบบบลอกเชิงเส้น (Linear Block Codes) เป็นรหัสที่มีโครงสร้างของรหัสที่ดี ซึ่งรหัสที่ดีส่วนใหญ่จะเป็นรหัสเชิงเส้น โดยตัวอย่างของรหัสแบบบลอกเชิงเส้นได้แก่

2.2.1.3.1 รหัสแฮมมิ่ง (Hamming Code) (Doll 1978: 274) เป็นรหัสที่ใช้ในการสืบหาและตรวจแก้ความผิดพลาดของข้อมูล โดยความสามารถในการสืบหาและแก้ความผิดพลาดของรหัสขึ้นอยู่กับค่าแฮมมิ่งดิสแทนส์ แสดงได้ในรูปความสัมพันธ์ของ

$$d = C + D + 1$$

โดย $D \geq C$

d เป็นค่าแฮมมิ่งดิสแทนส์ของรหัส

D เป็นจำนวนบิตที่เกิดความผิดพลาด ซึ่งสามารถสืบหาได้

C เป็นจำนวนบิตที่เกิดความผิดพลาด ซึ่งสามารถแก้ให้ถูกต้องได้

ซึ่งความสามารถในการตรวจแก้ความผิดพลาดของข้อมูลเป็นจำนวน 1 บิตนั้น แสดงได้ดังรูปความสัมพันธ์ของ

$$2^m \geq n + 1$$

โดย $n = m + k$

m เป็นจำนวนบิตเพิ่มซึ่งได้โดย การคำนวณจากจำนวนบิตน้อยที่สุดที่จำเป็นต้องมีใน n ตำแหน่งคือ $\log_2 n$ โดยการเพิ่มบิตเข้าไปเพื่อสืบหาความผิดพลาดของข้อมูล ซึ่งจะได้ว่า

$$m \geq \log_2 (n+1) \quad \text{หรือ} \quad 2^m \geq n+1$$

ดังนั้น จึงมีบิตในตำแหน่ง ที่เป็นเลขยกกำลังของสอง คือ 2, 4, 8, 16, ... และ 1 หรืออาจเป็นตำแหน่งใดๆ ในกลุ่มข้อมูล (Tomasi 1987: 70) เช่น รหัสแบบแฮมมิ่ง (17,12) ซึ่งมีตำแหน่งของบิตที่ 4, 8, 9, 13 และ 17 เป็นตำแหน่งของบิตที่เพิ่มเข้าไปเพื่อตรวจสอบความถูกต้องของข้อมูล โดยตัวรับจะมีตัวนับ (Counter) เริ่มต้นเป็น 0 แล้วตรวจสอบ กับบิตในตำแหน่งตรวจสอบเหล่านี้ ซึ่งจะทำการบวกค่าของตำแหน่งบิตตรวจสอบ ที่ผิดพลาดไว้ โดยผลบวกที่ได้ จะบอกตำแหน่งของข้อมูลบิตที่เกิดความผิดพลาดขึ้น แต่ถ้าตัวนับมีค่าเป็น 0 ก็แสดงว่าข้อมูลที่ได้รับมานั้นถูกต้อง ซึ่งการใช้รหัสแฮมมิ่งนี้ สามารถสืบหาความผิดพลาดของข้อมูลได้ 1 หรือ 2 บิตที่ผิดพลาด และสามารถตรวจแก้ความผิดพลาดของข้อมูลได้เพียง 1 บิตที่ผิดพลาดเท่านั้น เนื่องจากค่าดิสแทนส์น้อยที่สุดของรหัสแฮมมิ่งมีค่าเป็น 3 แสดงได้จากโคดเวิร์ด (n,k) ของรหัสแฮมมิ่ง ซึ่งสามารถสืบหาข้อมูลที่ผิดพลาดได้เป็นจำนวน 2k บิต และตรวจแก้ความผิดพลาดของข้อมูลได้เป็นจำนวน k บิต โดย $k = 1$ รหัสแฮมมิ่งจึงไม่สามารถตรวจแก้ความผิดพลาดของข้อมูล ที่ผิดพลาดเป็นช่วงได้

2.2.1.3.2 รหัสแบบไซคลิก (Cyclic Codes) (Blahut 1983: 93)

เป็นสับคลาส (Subclass) ของรหัสเชิงเส้น ซึ่งมีโครงสร้างที่ดี และเนื่องจากรหัสควบคุมความผิดพลาดที่นิยมใช้กันส่วนใหญ่ มักเป็นคลาสของรหัสแบบไซคลิก ซึ่งใช้หฤษฎีของกาโลอิส ฟิลด์ในการคำนวณ

เพื่อหารหัสที่ดี รหัสแบบไซคลิกจึงเป็นรหัสที่สำคัญ เนื่องจากมีฐานการเข้ารหัสและถอดรหัส ซึ่งมีอัลกอริทึมและการคำนวณที่มีประสิทธิภาพ

รหัสเชิงเส้นบน GF(q) สามารถอธิบายได้ในเทอม (Term) ของเมตริกซ์ ซึ่งมีสมาชิกจาก GF(q) เรียกว่าเมตริกซ์ตรวจสอบบิตเพิ่ม โดยเวคเตอร์ c บน GF(q) จะเป็นโคดเวิร์ดก็ต่อเมื่อ $cH^T = 0$ หรืออาจเขียนได้เป็น

$$c(x) = \sum_{i=0}^{n-1} C_i x^i$$

ซึ่งการคูณโคดเวิร์ดด้วย เมตริกซ์ตรวจสอบบิตเพิ่มก็คือ การหาค่าโพลีโนเมียล c(x) เมื่อ x = a ดังนั้นข้อบังคับสำหรับ c(x) ที่จะแทนโคดเวิร์ดก็คือ c(a) = 0 หรือหมายความว่า โพลีโนเมียลของเลขฐานสอง c(x) จะเป็นโคดเวิร์ดโพลีโนเมียล (Codeword polymomial) ก็ต่อเมื่อ a เป็น 0 ของ c(x)

การแปลงสมการของเมตริกซ์ ของรหัสเชิงเส้น ให้อยู่ในรูปสมการโพลีโนเมียลสำหรับรหัสเชิงเส้น เนื่องจาก สมการโพลีโนเมียลจะง่ายต่อการค้นหารหัสที่ดี ซึ่งรวมทั้งวิธีการเข้ารหัสและถอดรหัสด้วย

การอธิบาย รหัสแบบไซคลิกด้วยโพลีโนเมียล โดยรหัสเชิงเส้นบน GF(q) จะเรียกว่าเป็นรหัสแบบไซคลิก เมื่อ

$(C = C_o, C_1, \ldots, C_{n-1})$ อยู่ใน C แล้ว

$C' = (C_{n-1}, C_o, \ldots C_{n-2})$ ก็จะอยู่ใน C ด้วย ซึ่งโคดเวิร์ด C' จะได้จากการทำไซคลิกชิฟ (Cyclic shift) จากโคดเวิร์ด C โดยที่ทุกๆ รหัสเชิงเส้นบน GF(q) ซึ่งมีความยาว n จะเป็นสับสเปซของ $GF(q)^n$ และเนื่องจากรหัสแบบไซคลิกซึ่งเป็นสับสเปซหนึ่งใน $GF(q)^n$ นี้ มีคุณสมบัติของไซคลิก โดยทุกเวคเตอร์ใน $GF(q)^n$ สามารถแทนได้ด้วยโพลีโนเมียลในรูป x ดีกรีน้อยกว่าหรือเท่ากับ n-1 ซึ่งไซคลิกชิฟสามารถเขียนแทนได้ด้วยสมการ

$$xp(x) = R_{x^n-1}[xp(x)]$$

ดังนั้นถ้าโคดเวิร์ดสามารถแทนได้ด้วยโพลีโนเมียล และรหัสเป็นสับเซ็ทของ

$$GF(q)[x]/(x^n-1)$$

รหัสนั้นจะเป็นรหัสแบบไซคลิก ถ้า xc(x) ยังคงเป็นโคดเวิร์ดโพลีโนเมียล เมื่อ c(x) เป็นโคดเวิร์ดโพลีโนเมียล ดังนั้นเมื่อเลือกโคดเวิร์ดโพลีโนเมียลที่ไม่เป็น 0 ซึ่งมีดีกรี

น้อยที่สุดจาก C โดยดีกรีเป็น n-k แล้วคูณด้วยสมาชิกของฟิลด์ให้เป็นโพลีโนเมียลแล้ว จะต้องเป็นรหัสใน C เนื่องจากรหัสแบบไซคลิกเป็นรหัสเชิงเส้น ซึ่งโพลีโนเมียลที่ไม่เป็น 0 และมีดีกรีน้อยที่สุดเรียกว่าเป็น เจเนเรเตอโพลีโนเมียล (Generator polynomial) ของ C และเขียนแทนได้ด้วย g(x)

โดยรหัสแบบไซคลิกจะประกอบด้วย ผลคูณระหว่างเจเนเรเตอโพลีโนเมียล g(x) และโพลีโนเมียลดีกรี k-1 หรือน้อยกว่า คือ

$$c(x) = Q(x)g(x)$$

ทฤษฎีที่ 2.2.1.3.2.1 รหัสแบบไซคลิกซึ่งมีความยาวบลอกเป็น n จะมีเจเนเรเตอโพลีโนเมียล g(x) ก็ต่อเมื่อ g(x) หารได้ด้วย x^n-1

ดังนั้น ทุกๆ โพลีโนเมียล ที่หารได้ด้วย x^n-1 ก็สามารถใช้เป็นเจเนเรเตอโพลีโนเมียล เพื่อใช้กับรหัสแบบไซคลิกได้ สำหรับทุกรหัสแบบไซคลิก ซึ่งมีเจเนเรเตอโพลีโนเมียล g(x) โดย

$$x^n - 1 = g(x)h(x)$$

สำหรับบางโพลีโนเมียล h(x) หรือที่เรียกว่าโพลีโนเมียลตรวจสอบบิตเพิ่ม โดยทุกโคดเวิร์ดของ c(x) จะสอดคล้องกับสมการ

$$R_{x^n-1}[h(x)c(x)] = 0$$

ให้ c(x) แทนโคดเวิร์ดโพลีโนเมียลของรหัสแบบไซคลิกที่ถูกส่งไป ซึ่งสัมประสิทธิ์ของโพลีโนเมียล c(x) จะเป็นสัญลักษณ์ของเวคเตอร์ที่ส่งไป

ให้ v(x) แทนโคดเวิร์ดที่ได้รับ

และให้ $e(x) = v(x) - c(x)$

ซึ่งโพลีโนเมียล e(x) จะเรียกว่าเอเรอโพลีโนเมียล (Error polynomial) ซึ่งมีสัมประสิทธิ์ที่ไม่เป็น 0 ในตำแหน่งที่เกิดความผิดพลาดของลำดับของข้อมูล ซึ่งแทนด้วยโพลีโนเมียล i(x) ซึ่งมีดีกรี k-1 โดยเซ็ตของโพลีโนเมียลของข้อมูล (Information polynomial) อาจจับคู่กับเซ็ตของโคดเวิร์ดโพลีโนเมียลได้โดย

$$c(x) = i(x)g(x)$$

หรือเขียนได้ว่า

$$c(x) = x^{n-k}i(x) + t(x)$$

โดย t(x) จะต้องสอดคล้องกับสมการ

$$R_{g(x)}[c(x)] = 0$$

ซึ่ง $$R_{g(x)}[x^{n-k}i(x)] + R_{g(x)}[t(x)] = 0$$

และเนื่องจากดีกรีของ t(x) น้อยกว่า n-k ซึ่งเป็นดีกรีของ g(x)

ดังนั้น $$t(x) = -R_{g(x)}[x^{n-k}i(x)]$$

โดยการใช้กฎการจับคู่ 1-1 กำลังสูงสุดของสัมประสิทธิ์ซึ่งมีค่าเป็น k ของโพลีโนเมียล จะมีเพียงแบบเดียว ดังนั้นการเข้ารหัสแบบซิสเต็มมาติก หรือนันซิสเต็มมาติก (Non Systematic) จะให้ผลลัพธ์เป็นโคดเวิร์ดเดียวกัน แต่จะต่างกันที่ การจับกลุ่มระหว่าง i(x) และ c(x) จึงสามารถให้คำจำกัดความของซินโดรมโพลีโนเมียล (Syndrome polynomial) s(x) ซึ่งใช้ในการถอดรหัสแบบไซคลิก โดยซินโดรมโพลีโนเมียล จะเป็นเศษเหลือของ v(x) ภายใต้การหารของ g(x) คือ

$$s(x) = R_{g(x)}[v(x)]$$

และซินโดรมโพลีโนเมียลขึ้นอยู่กับ e(x) โดยไม่ขึ้นกับ c(x) หรือ i(x) เนื่องจาก

$$\begin{aligned} s(x) &= R_{g(x)}[v(x)] \\ &= R_{g(x)}[c(x)+e(x)] \\ &= R_{g(x)}[e(x)] \end{aligned}$$

หฤษฎีที่ 2.2.1.3.2.2 ให้ d^* เป็นดิสแทนส์ที่น้อยที่สุดของรหัสแบบไซคลิก C แล้ว ทุกๆ เอเรอโพลีโนเมียล ซึ่งมีน้ำหนักน้อยกว่า d^* จะมีซินโดรมโพลีโนเมียลเพียงแบบเดียว ซึ่งปัญหาของการแก้ความผิดพลาดของข้อมูลก็คือการหา e(x) ซึ่งมีสัมประสิทธิ์ที่ไม่เป็น 0 ที่น้อยที่สุด ที่สอดคล้องกับสมการ

$$s(x) = R_{g(x)}[e(x)]$$

ซึ่งสามารถใช้การสร้างตาราง ในกรณีที่จำนวนความผิดพลาดมีไม่มากนัก เรียกว่าตารางการคำนวณซินโดรม (Syndrome evaluator table) โดยตัวถอดรหัส จะหาโพลีโนเมียล e(x) ได้ โดยการคำนวณ s(x) จาก v(x) แล้วจึงหา s(x) จากตารางการคำนวณซินโดรมเพื่อให้ได้ e(x) แสดงได้ดังตารางที่ 2.3

e(x)	s(x)
1	$R_{g(x)}[1]$
x	$R_{g(x)}[x]$
x^2	$R_{g(x)}[x^2]$
:	:
1+x	$R_{g(x)}[1+x]$
$1+x^2$	$R_{g(x)}[1+x^2]$
:	:

ตารางที่ 2.3 แสดงตารางการคำนวณชินโดรม

ในการหาเจเนเตอโพลีโนเมียล ที่เป็นไปได้ สำหรับรหัสแบบไซคลิกซึ่งมีความยาวบลอกเป็น n นั้น ก็คือการหาตัวหารของ x^n-1 ซึ่ง x^n-1 สามารถเขียนได้ในเทอมของ ไพร์มแฟคเตอร์ (Prime Factor)

$$x^n-1 = f_1(x)f_2(x)...f_s(x)$$

เมื่อ s เป็นจำนวนของไพร์มแฟคเตอร์แล้ว สับเซ็ตใดๆ ของแฟคเตอร์เหล่านี้ สามารถคูณกันเพื่อให้ได้เจเนเรเตอโพลีโนเมียล g(x) ซึ่งถ้าไพร์มแฟคเตอร์ของ x^n-1 แตกต่างกัน ดังนั้นจะมีรหัสแบบไซคลิกความยาว n ที่แตกต่างกันได้ 2^s-2 แบบ โดยยกเว้นกรณีซึ่ง g(x) = 1 และ g(x) = (x^n-1)

เมื่อ g(x) เป็นเจเนเรเตอโพลีโนเมียลซึ่งต้องหาร x^n-1 ได้ ดังนั้น

$$g(x) = \prod f_i(x)$$

ซึ่งเป็นผลคูณบนสับเซ็ตของไพร์มโพลีโนเมียล ซึ่งรหัสแบบไซคลิกที่ได้จาก g(x) จะเป็นโพลีโนเมียลที่หารได้ด้วยแต่ละ $f_i(x)$

การหาไพร์มโพลีโนเมียล และเจเนเรเตอโพลีโนเมียล

นิยามที่ 2.2.1.3.2.3 ความยาว

บลอก n ซึ่งได้จากสมการ

$$n = q^m-1$$

เรียกว่าเป็น ความยาวบลอกเริ่มต้น (Primitive block length) สำหรับรหัสบน GF(q) ซึ่งรหัสแบบไซคลิกนี้ เรียกว่ารหัสเริ่มต้นของไซคลิก (Primitive cyclic code) โดย

$$x^{q^m-1}- 1 = f_1(x)...f_s(x)$$

ตัวอย่างเมื่อ n = 15 ซึ่งจะได้รหัสแบบไซคลิกที่มีความยาวบลอก 15 โดยการหาแฟคเตอร์ของ $x^{15}-1$ เพื่อให้ได้ไพร์มโพลีโนเมียล

$$x^{15}- 1 = (x+1)(x^2+x+1)(x^4+x+1)(x^4+x^3+1)(x^4+x^3+x^2+x+1)$$

ซึ่งแฟคเตอร์เหล่านี้ได้จาก การทดลองแบบลองผิดลองถูก (Trial and error) เพื่อให้ได้ไพร์มโพลีโนเมียล ซึ่งจะมี 2^5 หรือ 32 สับเซ็ตของไพร์มโพลีโนเมียลเหล่านี้ที่เป็นเจเนเรเตอโพลีโนเมียลของรหัสแบบไซคลิกที่มีความยาวบลอก 15 (ยกเว้นกรณี $g(x) = x^{15}-1$ เมื่อ k = 0 และ g(x) = 1 เมื่อ k = n) ดังนั้นจะมีรหัสแบบไซคลิกได้ 30 แบบ ซึ่งถ้าเลือก

$$g(x) = (x^4+x^3+1)(x^4+x^3+x^2+x+1)$$
$$= x^8+x^4+x^2+x+1$$

เพราะว่า g(x) มีดีกรี 8 ดังนั้น n-k = 8 ซึ่งจะได้ค่า k = 7 และเพราะว่า g(x) มีน้ำหนักเป็น 5 ดังนั้นดิสแทนซ์ที่น้อยที่สุดก็จะมีค่าน้อยกว่าหรือเท่ากับ 5 และรหัสแบบไซคลิก (15,7,5) นี้จะสามารถแก้ความผิดพลาดได้ 2 บิต

การเลือก g(x) สำหรับรหัสแบบไซคลิก จึงขึ้นอยู่กับว่า ต้องการรหัสลักษณะในรูปแบบใด และมีความสามารถแก้ความผิดพลาดได้เท่าไร

ในกรณีของรหัสแบบแบบแฮมมิ่ง ซึ่งมีความยาวบลอกเป็น

$$n = (q^m-1)/(q-1) \quad \text{ใน } GF(q)$$

จะมีคุณสมบัติเป็นไซคลิกถ้า m และ q-1 เป็นไพร์ม

รหัสแบบไซคลิก สำหรับแก้ความผิดพลาด 2 บิต จะแสดงได้โดย ให้ ความยาวบลอก n มีค่าเป็น 2^m-1 สำหรับบางค่าของ m และให้ a เป็นสมาชิกเริ่มต้นของ $GF(2^m)$

ถ้ารหัสซึ่งเป็นเลขฐานสองมี a และ a^3 เป็น 0 ของเจเนเตอไพลีโนเมียล รหัสจะสามารถแก้ความผิดพลาด 2 บิตได้เนื่องจากเมตริกซ์ตรวจสอบบิตเพิ่มซึ่งทำให้ $cH^T = 0$ หรือ

$$\sum_{i=0}^{n-1} c_i a^i = 0$$

สำหรับ GF(2) เลือกตำแหน่งของความผิดพลาดให้แทนด้วย a และ a^3 ให้ g(x) เป็นไพลีโนเมียลซึ่งมีดีกรีน้อยที่สุดซึ่งมี a และ a^3 เป็น 0 ใน $GF(2^m)$ โดยการถอดรหัสที่สามารถแก้ความผิดพลาด 1 บิตและ 2 บิตที่ผิดพลาดได้ รหัสนี้จะต้องมีดิสแทนส์ที่น้อยที่สุดอย่างน้อยเป็น 5 โดยเวิร์ดที่ได้รับจะเป็นไพลีโนเมียลดีกรี n-1 คือ

$v(x) = a(x)g(x) + e(x)$ ซึ่ง e(x) เป็นสัมประสิทธิ์ที่ไม่เป็น 0 ไม่เกิน 2 ตัว ซึ่ง

$e(x) = 0$ หรือ x^i หรือ $x^i+x^{i'}$ เมื่อเลขจำนวนเต็ม i และ i' เป็นตำแหน่งที่เกิดความผิดพลาด โดยใน $GF(2^m)$ สมาชิกของฟิลด์ a^i จะหมายถึงสมาชิกในตำแหน่งที่ i ซึ่งสมาชิกของฟิลด์เรียกว่าเลขที่ตำแหน่ง (Location number) กำหนดให้สมาชิกของฟิลด์ $x_1 = a^i$ และ $x_2 = a^{i'}$ ซึ่งเกิดความผิดพลาดในตำแหน่งของ x_1 และ x_2 และต้องมีเพียงแบบเดียว โดยกำลังของ a จะมีค่าได้สูงสุดคือความยาวบลอก n ดังนั้นถ้ามีความผิดพลาด 1 บิต ก็ให้ $x_2 = 0$ และถ้าไม่เกิดความผิดพลาดเลยก็คือ

$x_1 = x_2 = 0$

ให้ $s_1 = v(a)$ และ $s_3 = v(a^3)$ เป็นซินโดรมซึ่งคำนวณได้จาก v(x)
เนื่องจาก g(x) เป็น 0 ที่ a และ a^3
ดังนั้น $s_1 = e(a)$ และ $s_3 = e(a^3)$
สมมุติว่าเกิดความผิดพลาด 2 บิต ดังนั้น

$s_1 = a^i+a^{i'}$ และ $s_3 = a^{3i}+a^{3i'}$

หรือเขียนให้อยู่ในรูปของตัวแปร x_1 และ x_2 คือ

$s_1 = x_1+x_2$

$s_2 = x_1^3+x_2^3$

ถ้าสมการทั้ง 2 นี้ สามารถให้ค่า x_1 และ x_2 ที่มีเพียงแบบเดียว ก็แสดงว่าทั้ง 2 บิต ที่ผิดพลาดนั้น จะสามารถแก้ให้ถูกต้องได้ และรหัสจะมีดิสแทนส์น้อยที่สุดอย่างน้อยเป็น 5 โดยมีไพลีโนเมียล s(x) แสดงตำแหน่งความผิดพลาดที่เกิดขึ้น

รหัสแบบไซคลิก สำหรับแก้ความผิดพลาดที่ต่อเนื่อง โดยที่รหัสส่วนใหญ่มักออกแบบ เพื่อแก้ความผิดพลาดจำนวน t บิตใดๆ แต่เนื่องจากแชนแนลโดยทั่วไป มักจะเกิดความผิดพลาดแบบต่อเนื่อง ซึ่งความผิดพลาดในรูปแบบนี้ รหัสแบบไซคลิกสามารถแก้ได้เนื่องจากคุณสมบัติการเป็นไซคลิกของรหัส ซึ่งรหัสจะไม่เพียงแก้ไขความผิดพลาดที่ต่อเนื่องเท่านั้น แต่จะรวมทั้งความผิดพลาดต่อเนื่องที่เป็นไซคลิกด้วย

นิยามที่ 2.2.1.3.2.4 รูปแบบของความผิดพลาดต่อเนื่องที่เป็นไซคลิก มีความยาว t จะมีคุณสมบัติเป็นเวคเตอร์ ซึ่งมีสมาชิกไม่เป็น 0 และในสมาชิกจำนวน t จะมีตัวแรกและตัวสุดท้ายไม่เป็น 0 โดยรูปแบบของความผิดพลาดที่ต่อเนื่องอธิบายได้ดังสมการ

$$e(x) = x^i b(x) (\text{mod } x^n-1)$$

ซึ่ง b(x) เป็นโพลีโนเมียลดีกรีสูงสุดคือ t-1 และ b_o ไม่เป็น 0
ดังนั้น b(x) จะแสดงรูปแบบของความผิดพลาดที่ต่อเนื่อง และ x^i แสดงตำแหน่งแรกของความผิดพลาดที่ต่อเนื่องนั้น

รหัสแบบไซคลิก สำหรับแก้ความผิดพลาดที่ต่อเนื่องจะต้องมีซินโดรมโพลีโนเมียล s(x) ที่แตกต่างกัน โดย ความสัมพันธ์ระหว่าง s(x) และ e(x) คือ

$$s(x) = R_{g(x)}[e(x)]$$

โดย s(x) จะแตกต่างกันสำหรับแต่ละโพลีโนเมียล e(x) ซึ่งแทนรูปของแบบความผิดพลาดต่อเนื่องที่เป็นไซคลิกซึ่งมีความยาว t ดังนั้นรหัสนี้จะสามารถแก้ความผิดพลาดทุกความผิดพลาดที่ต่อเนื่องที่มีความยาว t ได้ เช่น เจเนเรเตอโพลีโนเมียล g(x) โดย

$$g(x) = x^6+x^3+x^2+x+1$$

สำหรับรหัสของเลขฐานสอง ที่มีความยาวบลอกเป็น 15 และสามารถแก้ความผิดพลาดที่ต่อเนื่องซึ่งมีความยาว 3 หรือน้อยกว่าได้

โดย
$$e(x) = x^i \qquad i=0,\ldots,14$$
$$e(x) = x^i(1+x) \quad (\text{mod } x^{15}-1) \quad i=0,\ldots,14$$
$$e(x) = x^i(1+x^2) \quad (\text{mod } x^{15}-1) \quad i=0,\ldots,14$$
$$e(x) = x^i(1+x+x^2) \quad (\text{mod } x^{15}-1) \quad i=0,\ldots,14$$

โดยที่แต่ละซินโดรมของแต่ละรูปแบบความผิดพลาดจะมี 56 แบบที่แตกต่างกัน ดังนั้นรหัสแบบไซคลิกที่ได้จาก g(x) จะสามารถแก้ความผิดพลาดที่ต่อเนื่องซึ่งมีความยาว 3 บิตได้

นิยามที่ 2.2.1.3.2.5 ขอบเขตของรีเจอร์ (Rieger bound) กำหนดไว้ว่า รหัสแบบบลอกเชิงเส้น ที่สามารถแก้ความผิดพลาดที่ต่อเนื่องซึ่งมีความยาว t หรือน้อยกว่าได้ จะต้องมีบิตเพิ่มเป็นจำนวนไม่น้อยกว่า 2t บิต แสดงได้ดังตารางที่ 2.4

เจเนเรเตอ ไพลีโนเมียล	พารามิเตอร์ (Parameter)	ความสามารถแก้ความผิดพลาดที่ต่อเนื่อง
$x^4+x^3+x^2+1$	(7,3)	2
$x^5+x^4+x^2+x+1$	(15,10)	2
$x^6+x^5+x^4+x^3+1$	(15,9)	3
$x^6+x^5+x^4+1$	(31,25)	2
$x^7+x^6+x^5+x^3+x^2+1$	(63,56)	2
$x^8+x^7+x^6+x^3+1$	(63,55)	3
$x^{12}+x^8+x^5+x^3+1$	(511,499)	4
$x^{13}+x^{10}+x^7+x^6+x^5+x^4+x^2+1$	(1023,1010)	4

ตารางที่ 2.4 แสดงความสามารถในการแก้ความผิดพลาดของรหัสแบบไซคลิก

การใช้เทคนิคของ อินเทอลีฟวิ่ง กับรหัสแบบไซคลิก จะทำให้ได้รหัสที่ยาวขึ้น จากรหัส (n,k) จะได้เป็นรหัส (jn,jk) จาก j โคดเวิร์ดใดๆ แล้วสลับโคดเวิร์ดเหล่านี้ โดยการสลับตำแหน่งของสัญญลักษณ์ ซึ่งถ้ารหัสแบบไซคลิกสามารถแก้ความผิดพลาดที่ต่อเนื่องที่มีความยาว t ใดๆ ได้แล้ว รหัสที่ได้จากการทำอินเทอลีฟ จะสามารถแก้ความผิดพลาดต่อเนื่องใดๆ ที่มีความยาว jt ได้ โดยเทคนิคของอินเทอลิฟวิ่งจะเป็นการสร้างรหัสแบบไซคลิก จากรหัสแบบไซคลิกอีกทีหนึ่ง แสดงได้โดย สมมุติให้ g(x) เป็นเจเนเรเตอไพลีโนเมียลของรหัส ดังนั้น $g(x^j)$ ก็ยังคงเป็นเจเนเรเตอไพลีโนเมียลของรหัสอินเทอลีฟด้วย โดย

ให้ $c_1(x) = i_1(x)g(x)$

$c_2(x) = i_2(x)g(x)$

: :

$c_j(x) = i_j(x)g(x)$

$c(x) = c_1(x^j)+xc_2(x^j)+...+x^{j-1}c_j(x^j)$

$c(x) = i_1(x^j)g(x^j)+xi_2(x^j)g(x^j)+...+x^{j-1}i_j(x^j)g(x^j)$

$=[i_1(x^j)+xi_2(x^j)+...+x^{j-1}i_j(x^j)]g(x^j)$

$=i(x)g(x^j)$

รหัสแบบไซคลิกจะถูกเข้ารหัสแบบนันซิสเต็มมาติก โดยการคูณโพลีโนเมียลข้อมูล $i(x)$ ด้วยโพลีโนเมียล $g(x)$ เพื่อให้ได้โคดเวิร์ด $c(x)$ ส่วนรหัสแบบซิสเต็มมาติกนั้นจะต้องให้บิตของข้อมูลอยู่ที่บิตตำแหน่งสูงสุดของโคดเวิร์ดเสียก่อน แล้วจึง คำนวณหาบิตเพิ่มเพื่อให้ได้โคดเวิร์ด โดยโคดเวิร์ดจะอยู่ในรูป

$$c(x) = x^{n-k}i(x)+t(x)$$

และ $$t(x) = -R_{g(x)}[x^{n-k}i(x)]$$

ดังนั้น $R_{g(x)}[c(x)]$ จะมีค่าเป็น 0 โดยภายหลังการหารบิตข้อมูลแล้ว เศษเหลือจะถูกส่งไปยังแซนแนล และเมื่อแซนแนลส่งโพลีโนเมียล $c(x)$ ไป ซึ่งจะถูกบวกด้วยเอเรอโพลีโนเมียล $e(x)$ ผลที่ได้จากแซนแนลก็คือ ลำดับของบิตที่ได้รับ ในรูปของโพลีโนเมียล

$$v(x) = c(x)+e(x)$$

และที่ตัวรับก็จะนำลำดับของบิตที่ได้รับ มาหารด้วย $g(x)$ โดยจะได้เศษเหลือเป็นซินโดรมโพลีโนเมียล ซึ่งจะใช้ในการหารูปแบบความผิดพลาดจากตารางการคำนวณซินโดรม และหลังจากแก้ความผิดพลาดของโคดเวิร์ดแล้ว จะได้ $c(x)$ แล้วจึงคำนวณหาบิตข้อมูลได้จากการหาร $c(x)$ ด้วย $g(x)$ ดังความสัมพันธ์จากสมการ

$$i(x) = R_{g(x)}[c(x)]$$

ซึ่งจะเป็นการสิ้นสุดการถอดรหัส

รหัสแบบไซคลิก ที่มีคุณสมบัติ เป็นรหัสซิสเต็มมาติกใดๆ สามารถทำให้ความยาวบล็อกของรหัสสั้นลงได้ จากรูปแบบของรหัส (n,k) ให้เป็นรหัส $(n-b,k-b)$ โดยการลดจำนวนบิตข้อมูลที่ใช้ลง เป็นจำนวน b บิต จากแต่ละโคดเวิร์ด โดยที่ b จะต้องน้อยกว่า k และบิตที่ไม่ใช้นั้นจะถูกทำให้สั้นลงโดยการเซ็ตค่าให้เป็น 0 และไม่ต้องส่งบิตนั้นไป แต่ตัวรับจะต้องเพิ่มค่านี้เข้าไป เพื่อ

ให้สามารถถอดรหัสได้ โดยถ้ารหัสแบบไซคลิกมีดิสแทนส์ที่น้อยที่สุดคือ d* แล้ว รหัสแบบไซคลิกที่ทำให้สั้นลงนี้ จะมีดิสแทนส์ที่น้อยที่สุดเป็น d* เหมือนเดิม หรืออาจจะมากกว่าได้ และในทำนองเดียวกัน ถ้ารหัสสามารถแก้ความผิดพลาดที่ต่อเนื่องซึ่งมีความยาว t ได้ รหัสที่ถูกทำให้สั้นลงนี้ ก็จะสามารถแก้ความผิดพลาดที่ต่อเนื่องกันซึ่งมีความยาว t หรือมากกว่าได้ โดยการใช้เทคนิคของการทำรหัสให้สั้นลงและอินเทอลีฟวิ่ง ก็จะได้รหัสที่สามารถแก้ความผิดพลาดที่ต่อเนื่องอย่างมีประสิทธิภาพเป็นจำนวนมาก แต่รหัสที่ทำให้สั้นลงนี้จะไม่เป็นรหัสแบบไซคลิกอีกต่อไป เนื่องจาก $R_x^{n'-1}[xc(x)]$ อาจจะไม่เป็นโคดเวิร์ดอีกต่อไป แม้ว่า c(x) จะเป็นโคดเวิร์ดก็ตาม

การตรวจสอบความถูกต้อง ของข้อมูล โดยใช้ไซคลิก (Cyclic Redundancy Check) หรือ CRC (Doll 1978: 265) เป็นเทคนิคที่นิยมใช้กันโดยทั่วไป สำหรับการสืบหาความผิดพลาดของข้อมูลที่เป็นบิต โดยใช้สมการทางคณิตศาสตร์ที่เป็นไพลีโนเมียล ซึ่งมีรูปแบบที่เหมาะสม สำหรับสืบหาความผิดพลาดของข้อมูลในลักษณะต่างๆ ที่ต้องการ ซึ่งโดยทั่วไปจะเป็นไพลีโนเมียลที่เป็นเลขจำนวนเฉพาะ (Prime Number) เพื่อนำไปหารข้อมูลที่ต้องการส่งผ่านไปนั้นให้ได้เศษเหลือที่เป็นหนึ่งเดียว (Unique) ซึ่งจะเป็นค่าที่ส่งไปพร้อมกับข้อมูล สำหรับตัวรับจะได้ทำการตรวจสอบความผิดพลาดของข้อมูลได้ โดยการคำนวณหาเศษเหลือ จากข้อมูลที่ได้รับมา ซึ่งหากมีค่าเท่ากัน ก็แสดงว่าข้อมูลที่ส่งมานั้นถูกต้อง หรืออาจจะบวกโดยไม่มีบิตหดของเศษเหลือนั้นกับข้อมูลที่ต้องการส่ง เมื่อตัวรับคำนวณแล้ว ไม่มีเศษเหลือก็แสดงว่าไม่มีการผิดพลาดของข้อมูล ซึ่งการใช้ CRC จะเลือกไพลีโนเมียลที่เหมาะสมจากเจเนเรเตอไพลีโนเมียล เพื่อนำไปคำนวณค่าสำหรับตรวจสอบความผิดพลาดของข้อมูลกับข้อมูลที่ต้องการส่ง ซึ่งอยู่ในรูปของเลขฐานสอง

$$a_{k-1}\ a_{k-2}\ \ldots\ a_1\ a_0$$

โดยสามารถจัดให้อยู่ในรูปของไพลีโนเมียลได้เป็น

$$M(x) = a_{k-1}\,x^{k-1} + a_{k-2}\,x^{k-2} + \ldots + a_1\,x + a_0$$

เจเนเรเตอไพลีโนเมียล จะนำไปทำการคำนวณกับบิตข้อมูล เพื่อหาเศษเหลือที่ต้องการ โดย คูณ M(x) ด้วย x^r เมื่อ r เป็นดีกรีของ g(x) และเพิ่ม 0 เข้าไปในตำแหน่งของบิตที่มีตำแหน่งต่ำ (Low Order) เพื่อให้ได้บิตเวิร์ด (Bit Word) เป็น n

โดย $n = r + k$

หาร $M(x)x^r$ ด้วย g(x) ซึ่งจะได้ Q(x) และเศษเหลือ คือ R(x)

อยู่ในรูปของ

$$Q(x) + R(x)/g(x)$$

การส่งผ่านข้อมูล จะส่งไปในรูปของ

$$T(x) = M(x)x^r + R(x)$$

ซึ่งหากตัวรับ ได้รับข้อมูลที่เกิดความผิดพลาด คือ $T(x)$ มีการเปลี่ยนแปลง โดย

$$T(x) = T(x) + E(x)$$

แล้ว จะสามารถสืบหาความผิดพลาดนั้นได้ ตราบใดที่ $T(x) + E(x)$ ไม่สามารถหารได้ด้วย $g(x)$ โดย (Martin 1970: 76) ในกรณีที่มีความผิดพลาดเพียงบิตเดียว คือ

$$E(x) = x^i$$

ซึ่ง i เป็นตำแหน่งบิตที่เกิดความผิดพลาด และ $i < n$ ดังนั้น ถ้า $g(x)$ มี 2 เทอม หรือมากกว่าก็จะไม่สามารถหาร $E(x)$ ได้ หนึ่งบิตที่ผิดพลาดก็จะถูกตรวจพบได้ ส่วนความผิดพลาดในกรณีอื่นๆ เช่น ความผิดพลาดเป็นจำนวน 2 บิต ซึ่งแสดงได้ในรูปสมการของ

$$E(x) = x^i + x^j \quad \text{เมื่อ } i \text{ และ } j \text{ น้อยกว่า } n$$
$$= x^i(1 + x^{j-i})$$

โดยความผิดพลาดนี้จะถูกตรวจพบได้ เมื่อ $g(x)$ ไม่สามารถหาร x^i และ $(1 + x^{j-i})$ ได้ ซึ่งถ้า $g(x)$ ประกอบด้วย 3 แฟคเตอร์ขึ้นไป ความผิดพลาดนี้ก็จะตรวจพบได้ ส่วนความผิดพลาดเป็นจำนวนคี่ของบิต ซึ่งจะไม่สามารถหารได้ด้วยเทอม $(x+1)$ เนื่องจาก $E(x)$ ซึ่งหารได้ด้วย $(x+1)$ จะสามารถแสดงได้ในรูปของสมการ

$$E(x) = (x+1)Q(x)$$

เมื่อแทนค่า $x = 1$ จะได้

$$E(x) = (1+1)Q(1)$$

ซึ่งก็คือ $E(1) = 0$ แสดงว่า $E(x)$ จะต้องประกอบด้วยจำนวนเทอมที่เป็นเลขคู่ ดังนั้น ถ้า $g(x)$ ประกอบด้วยแฟคเตอร์ $(x+1)$ แล้ว หากเกิดความผิดพลาดเป็นจำนวนคี่ของบิต ก็จะสามารถตรวจพบได้ ส่วนความผิดพลาดที่ต่อเนื่องหรือเป็นช่วง ซึ่งแสดงได้ในรูปสมการ

$$E(x) = x^i E_1(x) \quad \text{เมื่อ } i < n$$

ซึ่ง x^i จะไม่สามารถหารได้ด้วย $g(x)$ เนื่องจากเป็นเทอมเดียว และความผิดพลาดที่ต่อเนื่องนี้จะตรวจพบได้ ถ้า $E_1(x)$ ไม่สามารถหารได้ด้วย $g(x)$ โดยความยาวของบิตที่ผิดพลาดที่เป็นช่วงนี้จะต้องน้อยกว่าค่า $(r+1)$ ของ $g(x)$

ดังนั้น การเลือกเจเนเรเตอไพลีโนเมียลที่เหมาะสมกับลักษณะของความผิดพลาดที่เกิดขึ้น เพื่อให้สามารถสืบหาความผิดพลาดของข้อมูลในลักษณะต่างๆ ได้ครบถ้วนจึงนับว่าเป็นสิ่งจำเป็น สำหรับการใช้ CRC เพื่อสืบหาความผิดพลาดของข้อมูลที่เกิดขึ้นเป็นช่วงนั้น มีโอกาสที่ข้อมูลซึ่งผิดพลาดจะถูกยอมรับว่าถูกต้องเท่ากับ $2^{-(r-1)}$ สำหรับกรณีที่มีความผิดพลาดของข้อมูลเป็นช่วงที่ยาวกว่า r+1 ส่วนความผิดพลาดเป็นช่วงที่น้อยกว่านั้น มีโอกาสที่จะยอมรับข้อมูลซึ่งผิดพลาดเท่ากับ 2^{-r} ซึ่งถือว่าเป็นความผิดพลาดในช่วงที่ยอมรับได้

2.2.1.3.3 รหัสบอส-คลอด์ริ-ฮอคเควนเฮม (Bose-Chaudhouri-Hocquenghem Code) หรือ BCH (Blauhut 1983: 161) เป็นรหัสที่ใช้ตรวจแก้ความผิดพลาดของข้อมูล โดยใช้เทคนิคของ ยูคลิเดียน อัลกอริทึม (Eucildean Algorithm) เช่นเดียวกับ CRC คือใช้ฟังก์ชั่นไพลีโนเมียลที่เป็นจำนวนเฉพาะ โดยรหัสบอส-คลอด์ริ-ฮอคเควนเฮม จัดเป็นกลุ่มของรหัสที่สามารถแก้ความผิดพลาดของบิตได้มากกว่า 1 บิต ซึ่งเหมาะกับความผิดพลาดที่เกิดเป็นอิสระต่อกัน โดยการสร้างเจเนเรเตอไพลีโนเมียลของรหัส BCH บน GF(q) สำหรับรหัสที่แก้ความผิดพลาดได้เป็นจำนวน t บิต และมีความยาวบลอก q^m-1 ซึ่งรหัสที่มีรูปแบบนี้เรียกว่า รหัสเริ่มต้นของ BCH

นิยามที่ 2.2.1.3.3.1 รหัสแบบ BCH

เจเนเรเตอไพลีโนเมียลของรหัสแบบไซคลิก สามารถเขียนได้ในรูปของ

$$g(x) = LCM\ [f_1(x), f_2(x),.., f_r(x)]$$

โดย $f_1(x), f_2(x),..., f_r(x)$ เป็นไพลีโนเมียลที่เป็น 0 ของ g(x)

ให้ c(x) เป็นโคดเวิร์ดไพลีโนเมียล และ e(x) เป็นเอเรอไพลีโนเมียล โดยไพลีโนเมียลที่ตัวรับได้รับคือ

$$v(x) = c(x) + e(x)$$

โดยสัมประสิทธิ์ของไพลีโนเมียลจะอยู่ใน GF(q) ซึ่งสอดคล้องกับการเลือกไพร์มไพลีโนเมียล ดีกรี m และมีโครงสร้างช่อง $GF(q^m)$ การเลือก $f_j(x)$ ซึ่งเป็นไพลีโนเมียลของ a^j สำหรับ $j = 1,..,2t$ และ g(x) โดย

$$g(x) = LCM[f_1(x),..,f_{2t}(x)]$$

ซึ่งบางครั้งรหัสแบบ BCH จะสามารถออกแบบโดยวิธีดังกล่าว ให้สามารถแก้ความผิดพลาดได้มากกว่า t บิต โดยดีไซน์ดิสแทนส์ (Designed distance) ของรหัสหรือ d จะได้จากสมการ

$$d = 2t+1$$

และดิสแทนส์ที่น้อยที่สุดหรือ d^* อาจจะใหญ่กว่ารหัสโดยทั่วไป ซึ่งความสัมพันธ์ของค่าต่างๆ ของรหัสแบบ BCH แสดงได้ดังตารางที่ 2.5

ค่าเอกซ์โพเนนเชียล (Exponential)	ค่าโพลีโนเมียล	มินิมอลโพลีโนเมียล (Minimal polynomial)
0	0	-
a^o	1	-
a^1	z	x+1
a^2	z^2	x^4+x+1
a^3	z^3	x^4+x+1
a^4	z+1	$x^4+x^3+x^2+x+1$
a^5	z^2+z	x^4+x+1
a^6	z^3+z^2	x^2+x+1
a^7	z^3+z+1	$x^4+x^3+x^2+x+1$
a^8	z^2+1	x^4+x^3+1
a^9	z^3+z	x^4+x+1
a^{10}	z^2+z+1	$x^4+x^3+x^2+x+1$
a^{11}	z^3+z^2+z	x^4+x^3+1
a^{12}	z^3+z^2+z+1	$x^4+x^3+x^2+x+1$
a^{13}	z^3+z^2+1	x^4+x^3+1
a^{14}	z^3+1	x^4+x^3+1

ตารางที่ 2.5 แสดงความสัมพันธ์ของค่าต่างๆ ใน $GF(2^4)$

ดังนั้น เจเนเรเตอไพลีโนเมียลสำหรับรหัสแบบ BCH ที่สามารถแก้ความผิดพลาด 2 บิตได้ และมีความยาวบลอกเป็น 15 จะได้จาก

$$g(x) = LCM\ [f_1(x), f_2(x), f_3(x), f_4(x)]$$
$$= LCM\ [x^4+x+1, x^4+x+1, x^4+x^3+x^2+x+1, x^4+x+1]$$
$$= (x^4+x+1)(x^4+x^3+x^2+x+1)$$
$$= x^8+x^7+x^6+x^4+1$$

เนื่องจาก g(x) มีดีกรี 8 และ n-k = 8 ดังนั้น k = 7 และ g(x) สำหรับรหัส BCH(15,7) จึงเป็นรหัสที่สามารถแก้ความผิดพลาด 2 บิตได้ ซึ่งจะสังเกตได้ว่า รหัสแบบ BCH ออกแบบจาก n และ t และในทำนองเดียวกัน เราสามารถสร้างเจเนเรเตอไพลีโนเมียลสำหรับแก้ความผิดพลาดในรูปแบบอื่นๆ ได้ เช่น สำหรับ t = 3 จะได้

$$g(x) = LCM\ [f_1(x), f_2(x), f_3(x), f_4(x), f_5(x), f_6(x)]$$
$$= (x^4+x+1)(x^4+x^3+x^2+x+1)(x^2+x+1)$$
$$= x^{10}+x^8+x^5+x^4+x^2+x+1$$

ก็จะเป็น g(x) สำหรับรหัสแบบ BCH(15,5) ที่สามารถแก้ความผิดพลาดจำนวน 3 บิตที่ผิดพลาดได้

การถอดรหัสของ BCH เพื่อลดโอกาสที่จะถอดรหัสผิด ซึ่งอาจจะออกแบบการถอดรหัสให้สามารถแก้ความผิดพลาดได้เป็นจำนวน t บิต ซึ่งค่า 2t+1 จะต้องน้อยกว่า d^* โดยการนี้จะเกิดได้ก็ต่อเมื่อ ค่า d^* เป็นเลขคู่ ดังนั้นตัวถอดรหัสจะแน่ใจว่า สามารถสืบหารูปแบบของความผิดพลาด ที่ไม่สามารถแก้ให้ถูกต้องได้เสมอ ถ้า v เป็นจำนวนบิตของความผิดพลาดที่เกิดขึ้น โดย $t+v < d^*$

เนื่องจากต้องใช้ค่าอย่างน้อยเป็น d^*-t เพื่อที่จะเลื่อนโคดเวิร์ดใดๆ ไปยังสเฟียร์ของการถอดรหัสที่ไม่ถูกต้อง โดยเวิร์ดที่ไม่สามารถถอดรหัสได้ อาจจะถูกแฟลก (Flag) ว่ามีจำนวนความผิดพลาดมากกว่า t

2.2.1.3.4 รหัสแบบรีดโซโลมอน (Reed-Solomon Codes) (Blahut 1983: 174)

เป็นรหัสส่วนย่อย (Subset) ของรหัสแบบ BCH ซึ่งสามารถตรวจแก้ความผิดพลาดของข้อมูล จำนวน t บิตที่ผิดพลาด โดย

คลาส (Class) ของรหัสแบบรีดโซโลมอน เป็นสับเซ็ทที่สำคัญ และเป็นที่นิยมใช้ของรหัสแบบ BCH บน GF(q) โดยทั่วไป q จะมีค่ามากกว่า 2 ซึ่งรหัสแบบรีดโซโลมอนจะมีเจเนเรเตอไพลีโนเมียลเป็น

$$g(x) = (x-a)(x-a^2)\ldots(x-a^{2t})$$

ซึ่ง g(x) เป็นโพลีโนเมียลที่มีดีกรี 2t

ดังนั้น รหัสแบบรีด-โซโลมอน จะสอดคล้องกับสมการ

$$n - k = 2t$$

ดังตัวอย่างการหา g(x) สำหรับรหัส (15,11) ซึ่งมี t = 2 ของรหัสแบบรีดโซโลมอนบน GF(16) ดังนั้น

$$g(x) = (x-a)(x-a^2)(x-a^3)(x-a^4)$$

โดยใช้โพลีโนเมียลในรูป z ตามตารางที่ 2.5 จะได้

$$g(x) = x^4+(z^3+z^2+1)x^3+(z^3+z^2)x^2+z^3x+(z^2+z+1)$$
$$= x^4+a^{13}x^3+a^6x^2+a^3x+a^{10}$$

และเนื่องจาก g(x) มีดีกรี 4 ดังนั้น n-k = 4 ซึ่งจะได้ค่า k=11 และเป็นโพลีโนเมียลของลำดับข้อมูลจำนวน 11 บิต

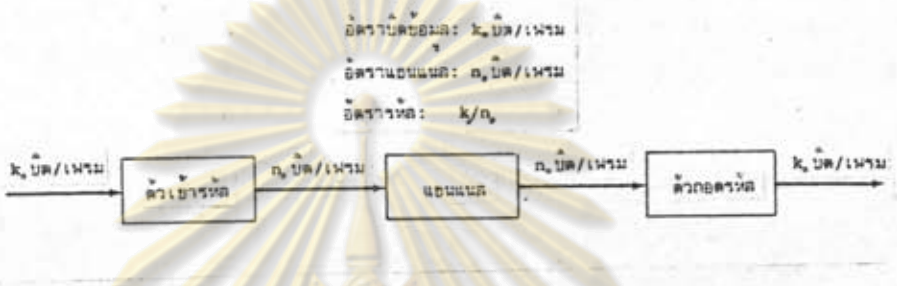
2.2.2 รหัสแบบทรี (Tree Codes) (Blahut 1983: 346)

รหัสแบบทรี สามารถอธิบายได้ด้วยลำดับของข้อมูลเริ่มต้นจะเป็น 0 โดยข้อมูลจะถูกแบ่งออกเป็นเซกเมนต์ (Segment) ซึ่งแต่ละเซกเมนต์จะมีจำนวน k_o บิต เรียกว่าเฟรมของข้อมูล (Information frames) โดยแต่ละเฟรมของข้อมูลอาจจะมีเพียง 1 บิตก็ได้ และโดยทั่วไปนิยมให้มีจำนวนบิตไม่มากนัก ซึ่งตัวเข้ารหัสจะสามารถเก็บเฟรมของข้อมูลไว้ได้เป็นจำนวน m เฟรม และระหว่างช่วงเวลา ของแต่ละเฟรมนั้น เฟรมของข้อมูลใหม่จะถูกชิฟ (Shift) เข้ามาในตัวเข้ารหัส โดยเฟรมของข้อมูลที่เก่าที่สุดจะถูกชิฟออกไปจากเฟรมการคำนวณรหัส ดังนั้นเฟรมของข้อมูล ในตัวเข้ารหัสจำนวน m เฟรมจะเป็นเฟรมที่ทันสมัยที่สุด นั่นก็คือ ข้อมูลจำนวน mk_o บิต จะถูกเข้ารหัส ได้เป็นเฟรมของโคดเวิร์ด (Codeword Frame) ซึ่งมีความยาว n_o บิต จากการคำนวณเฟรมของข้อมูลที่กำลังเข้ามา ซึ่งจะมี m เฟรมที่ทันสมัยที่สุด โดยเฟรมของโคดเวิร์ด จะถูกชิฟออกจากตัวเข้ารหัส ในขณะที่เฟรมของข้อมูลถัดไปจะถูกชิฟเข้ามา ดังนั้นแชนแนลจะได้โคดเวิร์ดซึ่งมีจำนวน n_o บิต สำหรับแต่ละ k_o บิตของข้อมูล รหัสแบบทรีจึงมีลำดับของข้อมูลเป็น k_o บิตต่อเวลา (Symbol per time) และลำดับที่

ต่อเนื่องกันของโคดเวิร์ดเป็น n_o บิตต่อช่วงเวลา (Symbol per Interval) แสดงได้ดังสมการ

$$R = k_o/n_o$$

และแสดงลักษณะของรหัสแบบทรีได้ดังรูปที่ 2.3



รูปที่ 2.3 แสดงลักษณะของรหัสแบบทรี

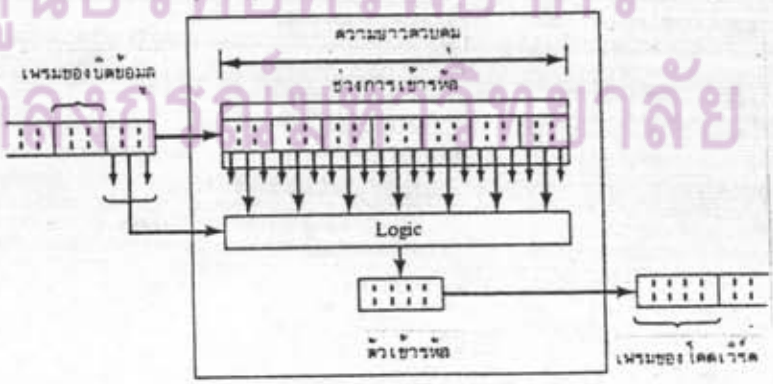
และเรียกว่ารหัสแบบทรี (n_o,k_o) ซึ่งมีอัตราเป็น $R = k_o/n_o$

ให้ $v = mk_o$ โดย v เป็นความยาวควบคุม (Constraint length)

และ $k = (m+1)k_o$ โดย k เรียกว่าความยาวเวิร์ด (Wordlength) ของรหัสคอนโวลูชันนัล ซึ่งจะได้ความยาวบลอก n จากสมการ

$$n = (m+1)n_o = k(n_o/k_o)$$

โดยความยาวบลอกจะเป็นความยาวของโคดเวิร์ดที่ได้จาก 1 เฟรมของข้อมูล แสดงได้ดังรูปที่ 2.4



รูปที่ 2.4 แสดงการเข้ารหัสของรหัสแบบทรี

จากรูป ถ้า $k_o = 3$, $n_o = 5$ จะได้ $v = 21$ และความยาวบลอก = 40 ซึ่งในทางปฏิบัติ รหัสแบบหรีจะใช้ k_o และ n_o เป็นเลขจำนวนเต็มน้อยๆ โดยรหัสจะมีประสิทธิภาพที่ดี ถ้า k_o มีค่าเป็น 1 แต่ทั้งนี้ก็ขึ้นอยู่กับอัตราของรหัสด้วย จึงไม่นิยมทำให้ค่า k_o เป็น 1 นัก

นิยามที่ 2.2.2.1 รหัสแบบหรีขนาด (n_o,k_o) จะเป็นการจับคู่จากเซ็ตของ ลำดับของสมาชิกใน GF(q) ไปที่ตัวมันเอง โดยรหัสแบบหรี จะมีความสัมพันธ์กับ

2.2.2.1.1 ความยาวควบคุมที่สิ้นสุด (Finite Constraint Length) โดยความยาวควบคุมของรหัสแบบหรี อาจจะสิ้นสุด (Finite) หรือไม่สิ้นสุด (Infinite) ก็ได้ แต่โดยทั่วไปรหัสแบบหรีจะมีคุณสมบัติของความยาวควบคุมที่สิ้นสุด โดยรหัสแบบหรี (n_o,k_o) ซึ่งมีความยาวควบคุมที่สิ้นสุดเป็น v และมีความยาวเวิร์ด $k = v + k_o$ และมีความยาวบลอก n จะเรียกว่าเป็น รหัสแบบหรีลิส (Trellis code) (n,k)

2.2.2.1.2 ความไม่เปลี่ยนแปลงทางเวลา (Time Invariance) คือช่วงเวลาของการเข้ารหัส ที่จะทำการชิฟแต่ละเฟรมเข้ามาในแต่ละครั้ง จะเท่ากันเสมอ

2.2.2.1.3 ความเป็นรหัสเชิงเส้น (Linearity) คือมีคุณสมบัติของการทำผลบวกเชิงเส้น (Linear Combination) ระหว่างลำดับของข้อมูลใดๆ ซึ่งจะมีโคดเวิร์ดที่มีคุณสมบัติของผลบวกเชิงเส้นเช่นเดียวกัน คือ ถ้า d_1 และ d_2 เป็น 2 ลำดับของข้อมูล โดยมีโคดเวิร์ดเป็น $G(d_1)$ และ $G(d_2)$ ดังนั้น ad_1+bd_2 จะมีโคดเวิร์ดเป็น

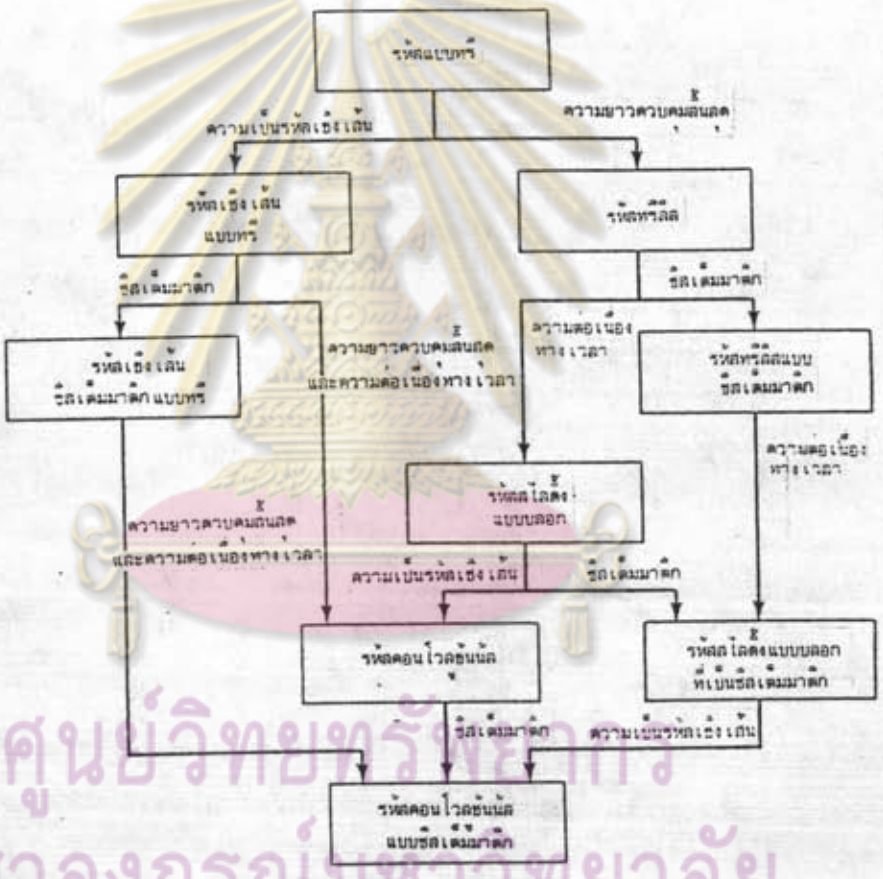
$$G(ad_1+bd_2) = aG(d_1) + bG(d_2)$$

2.2.2.1.4 ซิสเต็มมาติก โดยรหัสแบบหรี ซึ่งมีคุณสมบัติเป็นซิสเต็มมาติก จะมีเฟรมของข้อมูลคงที่ และไม่มีการสลับที่ภายใน k_o บิตแรก ของโคดเวิร์ดเฟรมแรก

นิยามที่ 2.2.2.2 รหัสแบบหรีขนาด (n_o,k_o) ซึ่งมีคุณสมบัติเป็นรหัสเชิงเส้น มีความไม่เปลี่ยนแปลงทางเวลา และมีความยาวเวิร์ดที่สิ้นสุด เป็น $k = (m+1)k_o$ จะเรียกว่าเป็น รหัสคอนโวลูชันนัล(n,k) และรหัสคอนโวลูชันนัล (n,k) ซึ่งมีคุณสมบัติซิสเต็มมาติกจะเรียกว่า รหัสซิสเต็มมาติกคอนโวลูชันนัล (Systematic Convolutional Code) (n,k) โดยรหัสแบบหรี(n_o,k_o) ก็คือ

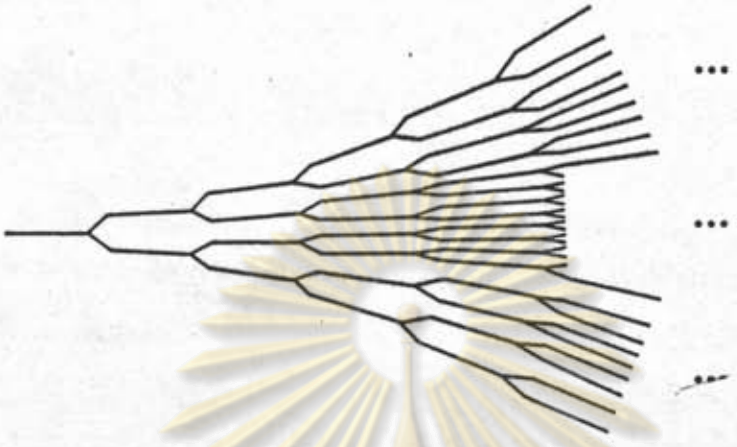
รหัสคอนโวลูชันนัล(n,k) ซึ่งมีค่า $k > k_o$

นิยามที่ 2.2.2.3 รหัสแบบทรี (n_o,k_o) ซึ่งมีความไม่เปลี่ยนแปลงทางเวลา และมีความยาวเวิร์ดที่สั้นสุดเป็น k จะเรียกว่าเป็นรหัสสไลดิ้งแบบบล็อก (Sliding Block Code) (n,k) ซึ่งรหัสสไลดิ้งแบบบล็อก ที่มีความเป็นรหัสเชิงเส้น ก็คือรหัสคอนโวลูชันนัลนั่นเอง โดยรหัสต่างๆ ที่มีรูปแบบเป็นทรี แสดงได้ดังรูปที่ 2.5



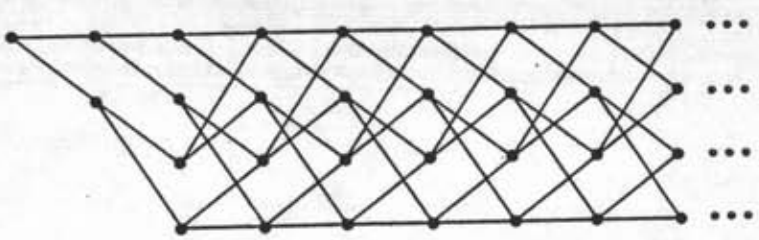
รูปที่ 2.5 แสดงลำดับชั้นของรหัสแบบทรี

รหัสแบบทรี (Coates 1982:245 ; Peebles 1987: 78) สามารถแทนได้ด้วยกราฟต้นไม้ (Graph tree) หรือ ทรีไดอะแกรม (Tree diagram) ซึ่งแสดงลำดับก่อนหน้าและลำดับปัจจุบันของรหัสแบบทรี โดยแสดงทุกๆ ลำดับที่เป็นไปได้ ซึ่งจำนวนของโหนด (Node) และสาขา (Branch) จะเพิ่มขึ้น โดยไม่มีข้อจำกัด ในรูปของต้นไม้ไปทางขวามือ แสดงได้ดังรูปที่ 2.6



รูปที่ 2.6 แสดงรูปกราฟต้นไม้

โดยแต่ละโหนดจะแทนสถานะซึ่งสัมพันธ์กับจำนวนรูปแบบของข้อมูล โดยมีค่าเริ่มต้นเป็น 0 แต่สำหรับรหัสซึ่งมีความยาวควบคุมที่ไม่สิ้นสุด หรือเพียงแต่มีความยาวควบคุมขนาดใหญ่ จะไม่เหมาะสมที่จะใช้กราฟต้นไม้นี้ ซึ่งกราฟที่นิยมใช้คือ กราฟทรีลิส (Trellis graph) หรือ ทรีลิสไดอะแกรม (Trellis diagram) โดยทรีลิสเป็นกราฟซึ่งมีโหนด อยู่ในแนวเส้นทแยงมุม และมีลำดับเช่นเดียวกับกราฟต้นไม้ โดยเป็นกราฟกึ่งไม่สิ้นสุด (Semi Infinite) ไปทางขวามือ แต่จำนวนของโหนดในแต่ละคอลัมน์จะสิ้นสุด ซึ่งโครงสร้างของสาขาที่เกี่ยวข้องกับแต่ละคอลัมน์ของโหนด ไปยังคอลัมน์ของโหนดทางขวามือจะเหมือนกันในแต่ละคอลัมน์ของโหนด โดยโหนดในแต่ละคอลัมน์ของทรีลิส จะแทนสถานะที่ q^{ν} ซึ่งอยู่ในระหว่างการเข้ารหัส โดยแต่ละสาขาจะชี้ไปพร้อมกับ 2 บิต ที่จะส่งไปยังแชนแนลในขณะที่มีการชิฟเปลี่ยนเป็นสถานะถัดไป ซึ่งกราฟเส้นบนจะแทนข้อมูลเข้ามา ที่เป็น 0 และเส้นล่างจะแทนข้อมูลเข้ามาที่เป็น 1 แสดงได้ดังรูปที่ 2.7



รูปที่ 2.7 แสดงรูปกราฟทรีลิส

นอกจากนี้ รหัสแบบทรีสามารถอธิบายได้ด้วย สเตทไดอะแกรม (State diagram) โดยเมื่ออยู่ในสถานะคงที่ (Steady-state) นั้น บิตข้อมูลที่ถูกชิฟเข้ามาใหม่ จะทำให้เกิดการเปลี่ยนแปลงสถานะ จากสถานะหนึ่งไปยังอีกสถานะหนึ่ง ซึ่งตัวเลขภายในวงกลมปิดจะหมายถึง สถานะปัจจุบันหรือสถานะก่อนหน้า และตัวเลขที่อยู่ตรงเส้นสูกศรบอกทิศทางแต่ละตัว จะแสดงบิตที่เข้ารหัสแล้ว เช่น สถานะเดิมคือ (00) แล้วเปลี่ยนสถานะเป็น (10) ก็จะได้ผลลัพธ์เป็น (11) ซึ่งสเตทไดอะแกรมแสดงได้ดัง รูปที่ 2.8

รูปที่ 2.8 แสดงสเตทไดอะแกรม

2.2.3.1 รหัสคอนโวลูชันนัลในรูปของโพลีโนเมียล

รหัสคอนโวลูชันนัลขนาด $((m+1)n_o, (m+1)k_o)$ บน GF(q) ซึ่งมีความยาวควบคุมเป็น $v = mk_o$ จะถูกเข้ารหัสได้เป็น n_o บิต โดยตัวเข้ารหัส สำหรับรหัสคอนโวลูชันนัล สามารถเขียนแทนได้ด้วยโพลีโนเมียล ซึ่งรหัสก็คือเซ็ตของโคดเวิร์ดที่ได้จากโพลีโนเมียล ซึ่งเป็นเจเนเรเตอโพลีโนเมียล ที่มีดีกรีสูงสุดคือ m ตรงกันข้ามกับรหัสแบบบล็อกซึ่งอธิบายได้ด้วย 1 เจเนเรเตอโพลีโนเมียล เพราะรหัสคอนโวลูชันนัลจะต้องใช้มากกว่า 1 เจเนเรเตอโพลีโนเมียล คือเป็นจำนวน k_on_o โพลีโนเมียล

ให้ $g_{ij}(x)$ สำหรับ $i = 1, \ldots, k_o$

และ $j = 1, \ldots, n_o$

เป็นเซ็ตของเจเนเรเตอโพลีโนเมียล ซึ่งสามารถจัดให้อยู่ในรูปเมตริกซ์ ของเจเนเรเตอโพลีโนเมียล ได้เป็น

$$G(x) = [g_{ij}(x)]$$

เช่น $G(x) = [1 \quad x^5+x^3+1]$

หรือ $G(x) = [x^2+x+1 \quad x^2+1]$

และเมื่อพิจารณาจาก เฟรมที่เข้ามาของข้อมูลจำนวน k_o บิต ซึ่งสามารถแทนได้ด้วยไพลีโนเมียล $d_i(x)$ สำหรับ $i = 1,\ldots, k_o$ หรือในเวคเตอร์ของไพลีโนเมียล

$$d(x) = [d_1(x), d_2(x),\ldots, d_{k_o}(x)]$$

และในทำนองเดียวกัน โคดเวิร์ดที่ได้จะแทนได้ด้วย n_o ดังนั้นโคดเวิร์ดไพลีโนเมียล $c_j(x)$ สำหรับ $j = 1,\ldots, n_o$ หรือในเวคเตอร์ของไพลีโนเมียล $c(x)$ โดย

$$c(x) = [c_j(x)]$$
$$= [c_1(x), c_2(x),\ldots, c_{n_o}(x)]$$

ซึ่งการเข้ารหัสจะสามารถอธิบายได้ด้วย การคูณของเมตริกซ์เวคเตอร์

$$c(x) = d(x)G(x)$$

หรือ $$c_j(x) = \sum_{i=1}^{k_o} d_i(x)g_{ij}(x)$$

และเมตริกซ์ของไพลีโนเมียลตรวจสอบบิตเพิ่ม $H(x)$ ซึ่งมีขนาดเป็น $(n_o-k_o) \times n_o$ จะสอดคล้องกับสมการ

$$G(x)\, H(x)^T = 0$$

โดยมีเวคเตอร์ของซินโดรมไพลีโนเมียล คือ

$$s(x) = v(x)H(x)^T$$

ซึ่งมีขนาดเป็น (n_o-k_o) และเป็นสมาชิกในเวคเตอร์ไพลีโนเมียล

การเข้ารหัสแบบซิสเต็มมาติก สำหรับรหัสแบบคอนโวลูชันนัล จะมีเมตริกซ์ของเจเนเรเตอไพลีโนเมียลอยู่ในรูป

$$G(x) = [I : \ P(x)]$$

โดย I เป็นเมตริกซ์เอกลักษณ์ขนาด $k_o \times k_o$

และ $P(x)$ เป็นเมตริกซ์ขนาด $k_o \times (n_o-k_o)$

สำหรับรหัสคอนโวลูชันนัลแบบซิสเต็มมาติกนั้น เมตริกซ์ของไพลีโนเมียลตรวจสอบบิตเพิ่มสามารถเขียนได้เป็น

$$H(x) = [-P(x)^T : I]$$

โดย I เป็นเมตริกซ์เอกลักษณ์ขนาด $(n_o-k_o) \times (n_o-k_o)$

ซึ่งจะได้ว่า

$$G(x)H(x)^T = 0$$

รหัสคอนโวลูชันนัลอาจจะเป็นรหัสแบบซิสเต็มมาติก หรือนันซิสเต็มมาติกก็ได้ โดยจะพิจารณาจากกรณีที่ $k_o = 1$ ดังนั้น

$$G(x) = [g_1(x)\ g_2(x)\ \ldots\ g_{n_o}(x)]$$

และ $c_j(x) = d(x)g_j(x)$ เมื่อ $j = 1, \ldots, n_o$

โดยสำหรับรหัสแบบซิสเต็มมาติกนั้น $g_1(x)$ จะมีค่าเป็น 1 เสมอ ซึ่งการจะหารหัสแบบคอนโวลูชันนัลที่ดีก็คือ การใช้กลุ่มของเจเนเรเตอไพลีโนเมียลที่เป็นไพร์ม

2.2.2.2 ความสามารถในการแก้ความผิดพลาดของรหัส

เมื่อรหัสคอนโวลูชันนัลถูกส่งผ่านแชนแนล และเกิดความผิดพลาดขึ้นนั้น ตัวถอดรหัสจะต้องจัดการกับโคดเวิร์ด โดยโคดเวิร์ดของรหัสคอนโวลูชันนัลนั้นอาจจะยาวมาก จนตัวถอดรหัสจะสามารถจำได้เพียง ส่วนหนึ่งในช่วงเวลาหนึ่งเท่านั้น และถึงแม้ว่าโคดเวิร์ดจะมีความยาวที่ไม่สิ้นสุด แต่การพิจารณาโคดเวิร์ดเพื่อทำการถอดรหัส จะทำในเซกเมนต์ของโคดเวิร์ดที่มีความยาวช่วงหนึ่งเท่านั้น

การถอดรหัสของรหัสคอนโวลูชันนัลนั้น ผลจากการถอดรหัสของเฟรมก่อนหน้า จะมีผลต่อการถอดรหัสในเฟรมถัดไปได้ โดยถ้า j เฟรมแรกสามารถแก้ความผิดพลาดได้ถูกต้องแล้ว การถอดรหัสในเฟรมที่ (j+1) ก็จะทำเช่นเดียวกันกับเฟรมที่ผ่านมา แต่หากโคดเวิร์ดในเฟรมแรก ไม่สามารถถอดรหัสให้ปราศจากบิตที่ผิดพลาดได้แล้ว ก็อาจจะทำให้โคดเวิร์ดในเฟรมถัดไปเกิดความผิดพลาดขึ้นได้ ซึ่งในกรณีที่ทำการถอดรหัสแล้ว ยังคงมีบิตที่ผิดพลาดอยู่ อาจจะชักนำให้โคดเวิร์ดเป็นจำนวนไม่สิ้นสุดเกิดความผิดพลาดขึ้นด้วยนั้น การถอดรหัสนั้นจะเรียกว่า เป็นการแพร่ขยายของบิตที่ผิดพลาด (Error propagation) ซึ่งจำนวนของบิตที่ตัวถอดรหัสสามารถเก็บไว้ได้จะเรียกว่าเป็น ช่วงกว้างของการถอดรหัส (Decoding window width) โดยประสิทธิภาพของการแก้ความผิดพลาด ที่มีช่วงกว้างของการถอดรหัสมากๆ จะดีกว่า แต่ก็มีขีดจำกัดที่ความกว้างนี้จะมีได้ถึงจุดหนึ่งเท่านั้น และช่วงกว้างของการถอดรหัสควรมีค่าอย่างน้อย เท่ากับความยาวบลอก n และส่วนใหญ่จะมากกว่า n

รหัสคอนโวลูชันนัล มีค่าดิสแทนส์ที่น้อยที่สุดได้หลายค่า ซึ่งได้จาก

นิยามที่ 2.2.2.2.1 ดิสแทนส์ที่น้อยที่สุด ตัวที่ 1 หรือ d_1^* ของรหัสคอนโวลูชันนัล จะเท่ากับค่าแฮมมิ่งดิสแทนส์ที่น้อยที่สุด ระหว่าง 2 โคดเวิร์ดเซกเมนต์ใดๆ ซึ่งมีความยาว l เฟรม ถ้า

$$l = m + 1$$

จะเรียกว่าเป็นดิสแทนส์ที่น้อยที่สุด แทนด้วย d^* และลำดับของ $d_1^*, d_2^*, d_3^*, \ldots$ จะเรียกว่าดิสแทนส์โปรไฟล์ (Distance profile) ของรหัสคอนโวลูชันนัล และเนื่องจากรหัสคอนโวลูชันนัลเป็นรหัสเชิงเส้น ดังนั้นดิสแทนส์ที่น้อยที่สุด ตัวที่ l จะเท่ากับน้ำหนักของโคดเวิร์ดเซกเมนต์ที่มีน้ำหนักน้อยที่สุด ซึ่งยาว l เฟรม โดยที่เฟรมแรกไม่เป็น 0 ดังนั้นถ้าเกิดความผิดพลาดจำนวน t บิต ซึ่งสอดคล้องกับอสมการ $2t+1 < d_1^*$ ใน l เฟรมแรก ซึ่งโคดเวิร์ดเฟรมแรกจะสามารถแก้ความผิดพลาดได้ โดยให้ $l = m+1$ และดิสแทนส์ที่น้อยที่สุด (d^*) ของรหัสเป็น d_{m+1}^* ดังนั้น t จะสอดคล้องกับอสมการ $2t+1 \leq d^*$ และรหัสจะสามารถแก้ความผิดพลาดของโคดเวิร์ดเฟรมแรกได้ ถ้า t บิตที่ผิดพลาดเกิดขึ้นในช่วงความยาวบลอกแรก ซึ่งรหัสนี้จะเรียกว่าเป็น รหัสคอนโวลูชันนัลที่สามารถแก้ความผิดพลาดได้เป็นจำนวน t บิต ที่ผิดพลาด

นิยามที่ 2.2.2.2.2 ฟรีดิสแทนส์ (Free distance) ของรหัสคอนโวลูชันนัล C จะได้จากสมการ

$$d_\infty = \max_l d_l \quad \text{โดย } d_{m+1} \leq d_{m+2} \leq \ldots \leq d_\infty$$

นิยามที่ 2.2.2.2.3 ฟรีเลนท์ (Free length) n_∞ ของรหัสคอนโวลูชันนัล จะเป็นความยาวของเซกเมนต์ที่ไม่เป็นศูนย์ของโคดเวิร์ด ของรหัสคอนโวลูชันนัลที่มีน้ำหนักน้อยที่สุด และน้ำหนักนั้นไม่เป็นศูนย์

ดังนั้น $d_1 = d_\infty$ ถ้า $l = n_\infty$ และ $d_1 < d_\infty$ ถ้า $l < n_\infty$

และโดยทั่วไปฟรีเลนท์จะมีค่ามากกว่าหรือเท่ากับความยาวบลอก

2.2.2.3 รหัสคอนโวลูชันนัลในรูปของเมตริกซ์

เนื่องจาก รหัสคอนโวลูชันนัล ประกอบด้วยโคดเวิร์ดที่เป็นจำนวนไม่สิ้นสุด แต่ด้วยความที่เป็นรหัสเชิงเส้น จึงสามารถแสดงได้ด้วยเจเนเรเตอเมตริกซ์ที่ไม่สิ้นสุด โดยเจเนเรเตอโพลีโนเมียล ซึ่งมีอินเด็กซ์(Index) i และ j อยู่ในรูปของความสัมพันธ์

$$g_{ij}(x) = \sum_l g_{ijl} x^l \quad \text{ซึ่ง } g_{ijl} \text{ จะอยู่ในรูปของเมตริกซ์}$$

และสำหรับแต่ละ l ให้ G_l เป็นเมตริกซ์ขนาด $k_o \times n_o$ โดย

$$G_1 = [g_{ij1}]$$

ดังนั้น เจเนเรเตอเมตริกซ์ของรหัสคอนโวลูชันนัล จึงสามารถแบ่งเป็นรหัสแบบบลอกขนาด n เป็น

$$G^{(n)} = \begin{bmatrix} G_o & G_1 & G_2 & \dots & G_m \\ 0 & G_o & G_1 & \dots & G_{m-1} \\ . & 0 & G_o & \dots & G_{m-2} \\ . & . & . & \dots & . \\ 0 & 0 & 0 & \dots & G_o \end{bmatrix}$$

โดยที่แต่ละ 0 เป็นเมตริกซ์ศูนย์ขนาด $k_o \times n_o$

ดังนั้น เจเนเรเตอเมตริกซ์สำหรับรหัสคอนโวลูชันนัล คือ

$$G = \begin{bmatrix} G_o & G_1 & G_2 & \dots & G_m & 0 & 0 & 0 & 0 & \dots \\ 0 & G_o & G_1 & \dots & G_{m-1} & G_m & 0 & 0 & 0 & \dots \\ 0 & 0 & G_o & \dots & G_{m-2} & G_{m-1} & G_m & 0 & 0 & \dots \\ . & . & . & \dots & . & . & . & . & . & \dots \end{bmatrix}$$

และสำหรับรหัสคอนโวลูชันนัลที่เป็นซิสเต็มมาติก 2 เมตริกซ์ข้างบน จะเขียนได้ดังนี้

$$G^{(n)} = \begin{bmatrix} I & P_o & 0 & P_1 & 0 & P_2 & & 0 & P_m \\ 0 & 0 & I & P_o & 0 & P_1 & \dots & 0 & P_{m-1} \\ 0 & 0 & 0 & 0 & I & P_0 & \dots & 0 & P_{m-2} \\ . & . & . & . & . & . & \dots & . & . \\ 0 & 0 & 0 & 0 & 0 & 0 & \dots & I & P_o \end{bmatrix}$$

และ

$$G = \begin{bmatrix} I & P_o & 0 & P_1 & 0 & P_2 & \dots & 0 & P_m & 0 & 0 & 0 & 0 \\ 0 & 0 & I & P_o & 0 & P_1 & \dots & 0 & P_{m-1} & 0 & P_m & 0 & 0 \\ 0 & 0 & 0 & 0 & I & P_o & \dots & 0 & P_{m-2} & 0 & P_{m-1} & 0 & P_m \\ . & . & . & . & . & . & \dots & . & . & . & . & . & . \end{bmatrix}$$

โดย I เป็นเมตริกซ์เอกลักษณ์ขนาด $k_o \times k_o$

0 เป็นเมตริกซ์ศูนย์ขนาด $k_o \times k_o$

และ $P_o, \ldots, P_m$ เป็นเมตริกซ์ขนาด $k_o \times (n_o - k_o)$

โดยแถวแรกจะเป็นการเข้ารหัสของเฟรมของข้อมูลแรก ไปยัง m เฟรมของโคดเวิร์ดแรก หรือเฟรมของข้อมูลแรกจะถูกเข้ารหัสเป็นโคดเวิร์ดเฟรมแรก โดยบลอกของ G

$$G^{(n_o)} = [I \quad P_o]$$

และในทำนองเดียวกัน 2 เฟรมแรกของข้อมูล จะถูกเข้ารหัสเป็น 2 โคดเวิร์ดเฟรมแรก โดย

$$G^{(2n_o)} = \begin{bmatrix} I & P_o & 0 & P_1 \\ 0 & 0 & I & P_o \end{bmatrix}$$

เมตริกซ์ตรวจสอบบิตเพิ่ม จึงเป็นเมตริกซ์ H ใดๆ ซึ่งสอดคล้องกับสมการ

$$G^{(ln_o)} H^{(ln_o)T} = 0 \quad \text{เมื่อ } l = 0,1,2,\ldots$$

โดยที่ตัวอย่างของรหัสคอนโวลูชันนัล ที่เป็นซิสเต็มมาติกของเลขฐานสอง ซึ่งมีค่า k_o เป็น 1 และค่า m = 1 แสดงได้ด้วยเมตริกซ์ขนาด 1x1 ของเมตริกซ์ $P_o = 1$ และ $P_1 = 1$ ดังนั้น

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & & & \\ & & 1 & 1 & 0 & 1 & \\ & & & & 1 & 1 & 0 & 1 \\ & & & & & \cdots & \\ & & & & & & \cdots \end{bmatrix}$$

$$H = \begin{bmatrix} 1 & 1 & & & & \\ 1 & 0 & 1 & 1 & & \\ & & 1 & 0 & 1 & 1 \\ & & & \cdots & & \\ & & & & \cdots & \end{bmatrix}$$

2.2.2.4 ตัวอย่างรหัสแบบคอนโวลูชันนัล

รหัสแบบคอนโวลูชันนัลส่วนใหญ่ ที่รู้จักกันและใช้อยู่ทุกวันนี้ ได้มาจากการหาโดยใช้คอมพิวเตอร์ ซึ่งในที่นี้จะกล่าวถึงคลาสโดยทั่วไปของรหัสคอนโวลูชันนัล ที่สามารถแก้ความผิดพลาด 1 บิตที่ผิดพลาดได้ โดยคลาส (Class) ของรหัสคอนโวลูชันนัลที่สามารถแก้ความผิดพลาด 1 บิตได้นั้น เรียกว่า รหัสวินเนอร์-แอช (Winer-Ash Codes) ซึ่งคล้ายคลึงกับคลาสของรหัสแบบแฮมมิ่ง จากเลขจำนวนเต็ม m ที่เป็นบวก จะมีรหัสวินเนอร์-แอช $((m+1)2^m, (m+1)(2^m-1))$ ซึ่งจะอธิบายได้ในเทอมของเมตริกซ์ตรวจสอบบิตเพิ่ม H ของรหัสแฮมมิ่ง $(2^m-1, 2^m-1-m)$ โดยจะมีเมตริกซ์ตรวจสอบบิตเพิ่มขนาด $m \times (2^m-1)$ ที่มี 2^m-1 คอลัมน์ ที่แตกต่างกัน และไม่เป็นศูนย์

ให้แถวของเมตริกซ์ แสดงเซ็ตของเมตริกซ์ ขนาด $1 \times (2^m-1)$ คือ $P_1^T, \ldots, P_m^T$ และให้แถวของเวคเตอร์ P_o^T ซึ่งมีสมาชิกจำนวน 2^m-1 เป็นศูนย์ทั้งหมด ดังนั้นเมตริกซ์ตรวจสอบบิตเพิ่มสำหรับรหัสวินเนอร์-แอชคือ

$$H = \begin{bmatrix}
P_o^T & 1 & 0 & 0 & 0 & 0\,0\,0 & \ldots \\
P_1^T & 0 & P_o^T & 1 & 0 & 0\,0\,0 & \ldots \\
P_2^T & 0 & P_1^T & 0 & P_o^T & 1\,0\,0 & \ldots \\
 & & & & P_1^T & : & \ldots \\
P_m^T & 0 & : & : & : & : & \ldots \\
0 & 0 & P_m^T & 0 & : & : & \ldots \\
: & : & : & : & : & : & \ldots
\end{bmatrix}$$

และ

$$H^{((m+1)2^m)} = \begin{bmatrix}
P_o^T & 1 & 0 & 0 & 0 & 0 & & \\
P_1^T & 0 & P_o^T & 1 & 0 & 0 & & \\
P_2^T & 0 & P_1^T & 0 & P_o^T & 1 & & \\
: & : & : & : & : & : & & \\
P_m^T & 0 & P_{m-1}^T & 0 & P_{m-2}^T & 0 & \ldots & P_o^T \quad 1
\end{bmatrix}$$

ซึ่ง 1 เป็นเมตริกซ์ $[1]_{1\times1}$ และ 0 เป็นเมตริกซ์ $[0]_{1\times1}$

และเนื่องจากรหัสวินเนอร์-แอชมีค่าดิสแทนส์ที่น้อยที่สุดคือ d* ซึ่งมีค่าเป็น 3 ดังนั้นจึงเป็นรหัสคอนโวลูชันนัลที่สามารถแก้ความผิดพลาด 1 บิตได้

2.2.2.5 อัลกอริทึมสำหรับถอดรหัสโดยใช้ซินโดรม สมมุติว่าตัวรับได้รับลำดับของบิต ซึ่งมีความยาวไม่สิ้นสุดเป็น v ซึ่งอยู่ในรูปสมการ

$$v = c + e$$

และเช่นเดียวกับรหัสแบบบล็อก เราสามารถคำนวณซินโดรมได้จากสมการ

$$s = vH^T$$
$$= eH^T$$

โดยซินโดรมก็จะมีความยาวไม่สิ้นสุดเช่นกัน ซึ่งตัวถอดรหัสจะไม่สนใจทั้งซินโดรม แต่จะทำในทำนองเดียวกันกับการเข้ารหัส คือจะไม่สนใจกับซินโดรมที่เก่าแล้ว ซึ่งตัวถอดรหัสจะมีตารางของเซกเมนต์ของซินโดรมเพื่อใช้ในการถอดรหัส และเมื่อตัวถอดรหัสพบเซกเมนต์ของซินโดรมที่อยู่ในตารางก็จะสามารถแก้ความผิดพลาด โดยสร้างเซกเมนต์ของโคดเวิร์ดใหม่ได้ ซึ่งหลังจากที่เฟรมแรกที่ผิดพลาด ถูกแก้ให้ถูกต้องแล้ว ซินโดรมจะต้องถูกเปลี่ยนแปลง เพื่อที่จะไม่ก่อให้เกิด การแก้ความผิดพลาดแบบผิดๆ (False Correction) ในเฟรมถัดไปได้

2.2.2.6 รหัสคอนโวลูชันนัล สำหรับแก้ความผิดพลาดที่ต่อเนื่อง

เนื่องจากความผิดพลาดเป็นช่วง ที่มีความยาว t คือลำดับใดๆ ของบิตจำนวน t โดยตัวแรกและตัวสุดท้ายของ t ไม่เป็น 0 ซึ่งสำหรับเวิร์ดที่มีความยาวไม่สิ้นสุดที่เป็นรหัสคอนโวลูชันนัลนั้น อาจเกิดความผิดพลาดขึ้นได้เป็นจำนวนมาก ซึ่งอาจเป็นกลุ่มของความผิดพลาดที่มีความยาวแตกต่างกันไป โดยรหัสคอนโวลูชันนัลซึ่งมีตัวถอดรหัสที่สามารถแก้ความผิดพลาดเป็นช่วง ซึ่งมีความยาว tใดๆ ได้ จะเรียกว่ารหัสนี้มีความสามารถแก้บิตที่ผิดพลาดได้ t บิต โดยรหัสคอนโวลูชันนัล (n,k) จะสามารถแก้ความผิดพลาดเป็นช่วงใดๆ ที่มีความยาวมากกว่า t ได้โดยใช้อินเทอลีฟวิ่ง ซึ่งจะได้เป็นรหัสคอนโวลูชันนัล (jn,jk) จากการเข้ารหัสจำนวน j ครั้ง แล้วสลับโคดเวิร์ดเหล่านี้เข้าด้วยกัน ซึ่งถ้ารหัสต้นแบบ (Original Code) สามารถแก้ความผิดพลาดได้เป็นจำนวน t ใดๆ แล้ว การใช้รหัสอินเทอลีฟด้วย จะทำให้สามารถแก้ความผิดพลาดเป็นช่วงยาว jt ได้ เช่น รหัสคอนโวลูชันนัลแบบซิสเต็มมาติก (14,7) ซึ่งมีความยาวควบคุมเป็น 6 และมีเจเนเรเตอโพลีโนเมียลเป็น

$$g_1(x) = 1$$

และ $$g_2(x) = x^6+x^5+x^2+1$$

จะสามารถแก้ 2 บิตที่ผิดพลาดใดๆ ได้ในช่วงของบิตข้อมูล 14 บิต และหากใช้การเข้ารหัสแบบอินเทอลีฟเป็นจำนวน 4 ครั้งของรหัส (14,7) แล้ว บิตที่ทำอินเทอลีฟเหล่านี้ ก็จะเป็นรหัส (56,28) ซึ่งจะสามารถแก้ความผิดพลาดเป็นช่วงใดๆ ได้ในช่วง 8 บิตที่ผิดพลาด โดยเทคนิคของอินเทอลีฟวิ่งนี้ จะสร้างรหัสคอนโวลูชันนัลจากรหัสคอนโวลูชันนัลอีกทีหนึ่ง โดยถ้า $g(x)$ เป็นเจเนเรเตอโพลีโนเมียลของรหัสต้นแบบ ดังนั้น $g(x^j)$ จะเป็นเจเนเรเตอโพลีโนเมียลของรหัสที่ทำอินเทอลีฟ โดยเจเนเรเตอโพลีโนเมียลของตัวอย่างข้างต้นก็จะกลายเป็น

$$g_1(x) = 1$$

และ $$g_2(x) = x^{24}+x^{20}+x^8+1$$

ซึ่งเป็นในทำนองเดียวกันกับการทำอินเทอลีฟวิ่งของรหัสแบบไซคลิก

การใช้อินเทอลีฟวิ่งกับรหัสคอนโวลูชันนัลที่สามารถแก้ความผิดพลาดในช่วงสั้นๆ ได้ จะทำให้ได้รหัสคอนโวลูชันนัลที่สามารถแก้ความผิดพลาดในช่วงที่ยาวขึ้นได้ โดยรหัสที่ทำอินเทอลีฟนี้จะไม่เพียงแก้ความผิดพลาดเป็นช่วงได้เท่านั้น แต่จะรวมถึงรูปแบบอื่นๆ ของความผิดพลาดแบบสุ่ม (Random error) ด้วย แต่หากต้องการแก้ความผิดพลาดที่เป็นช่วงเท่านั้น ก็ควรใช้รหัสที่เหมาะสมกับการแก้ความผิดพลาดเป็นช่วงเท่านั้น โดยรหัสไอวาแดร์ (Iwadare Code) เป็นคลาสของรหัสที่สามารถแก้ความผิดพลาดเป็นช่วงใดๆ ที่มีความยาว z หรือน้อยกว่าได้ โดย z เป็นดีไซน์พารามิเตอร์ (Design parameter)

นิยามที่ 2.2.2.6.1 ให้ z และ n_o เป็นจำนวนเต็มบวกใดๆ รหัสไอวาแดร์ จะเป็นรหัสคอนโวลูชันนัลแบบซิสเต็มมาติก ที่สามารถแก้ความผิดพลาดเป็นช่วงได้ ซึ่งมีเจเนเรเตอโพลีโนเมียลขนาด $(n_o-1) \times n_o$ คือ

$$G(x) = \begin{bmatrix} 1 & g_1(x) \\ 1 & g_2(x) \\ . & : \\ 1 & g_{(n_o-1)}(x) \end{bmatrix}$$

โดยสมาชิกในเมตริกซ์ $g_{ino}(x)$ จะเขียนได้สั้นๆ เป็น $g_i(x)$ โดย

$$g_i(x) = x^{(z+1)(2n_o-i)+i-3} + x^{(z+1)(n_o-i)-i}$$

เมื่อ $i = 1, \ldots, n_o-1$

ซึ่งดีกรีที่มากที่สุดของเจเนเรเตอโพลีโนเมียลคือ $g_1(x)$

โดยมีดีกรีเป็น $(z+1)(2n_o-1)-2$

ดังนั้นรหัสไอวาแดร์จึงเป็นรหัสคอนโวลูชันนัล $((m+1)n_o,(m+1)(n_o-1))$ โดย m เป็นจำนวนของเฟรมของข้อมูล ซึ่งสอดคล้องกับสมการ

$$m = (z+1)(2n_o-1)-2$$

และสามารถแก้ความผิดพลาดที่มีความยาว zn_o หรือน้อยกว่าได้ โดยเมตริกซ์ตรวจสอบบิตเพิ่มคือ

$$H(x) = [g_1(x)\ g_2(x)\ \ldots\ g_{(n_o-1)}(x)\ \ 1]$$

และเนื่องจาก $n_o - k_o = 1$ จึงมีเพียง 1 ซินโดรมโพลีโนเมียลคือ

$$S(x) = \sum_{i=1}^{n_o-1} g_i(x)e_i(x) + e_{n_o}(x)$$

$$= e_{n_o}(x) + \sum_{i=o}^{n_o-1} [x^{(z+1)(n_o-i)-1} + x^{(z+1)(2n_o-i)+i-3}]e_i(x)$$

โดยตัวถอดรหัสจะใช้โพลีโนเมียลนี้หาช่วงความผิดพลาดที่เกิดขึ้น สมมุติว่าช่วงของความผิดพลาดเกิดขึ้นที่เฟรมแรก และมีความยาว zn_o บิต ซึ่งถ้าความผิดพลาดนั้นไม่ได้เกิดที่บิตแรกของเฟรมแรก ก็อาจจะยาวมาถึงเฟรมที่ z+1 ได้ โดยตัวถอดรหัสจะต้องสามารถค้นหาบิตที่ผิดพลาดเป็นช่วงได้จากซินโดรม และจะต้องไม่มีบิตที่ผิดพลาดอื่นเพื่อไม่ให้ซินโดรมถูกรบกวน ในขณะที่ช่วงของบิตที่ผิดพลาดถูกแก้ให้ถูกต้อง คือต้องมีช่วงห่างที่พอเหมาะ ระหว่างความผิดพลาดเป็นช่วงที่เกิดขึ้นในแต่ละครั้ง

2.2.2.7 อัลกอริทึมสำหรับการถอดรหัส โดยใช้ไวเทอบิ (Viterbi decoding) (Blahut 1983: 377 ; Peebles 1987: 160)

ไวเทอบิเป็นอัลกอริทึมที่ดี สำหรับการถอดรหัสอย่างสมบูรณ์ของรหัสคอนโวลูชันนัล ซึ่งโอกาสที่จะเกิดการถอดรหัสแบบผิดๆ จะน้อยมาก ซึ่งอัลกอริทึมนี้ เหมาะกับรหัสซึ่งเป็นเลขฐานสองที่มีความยาวควบคุมสั้นๆ และโดยทั่วไปมีขอบเขตอยู่ที่ 7 ถึง 10 บิต ส่วนรหัสที่ไม่ใช่เลขฐานสองนั้น ไม่เหมาะสมที่จะใช้การถอดรหัส แบบไวเทอบิ เนื่องจากจะใช้ได้เฉพาะรหัสที่มีความยาวควบคุมที่สั้นมากๆ

เท่านั้น โดยเป็นการหาเส้นทางจาก กราฟต้นไม้ หรือกราฟทรีลิส ซึ่งใกล้เคียงกับโคดเวิร์ดที่ตัวรับได้รับมามากที่สุด ดังนั้น วิธีของการถอดรหัสโดยใช้ดิสแทนส์ที่น้อยที่สุด (Minimum distance Decoder) จึงเป็นการถอดรหัสที่ดี สำหรับรหัสคอนโวลูชันนัล โดยการเลือกช่วงกว้างของการถอดรหัส เป็น b สำหรับการถอดรหัส จากรหัสที่มีความยาวบลอกเป็น n โดยการคำนวณทุกโคดเวิร์ดที่มีความยาว b แล้วเปรียบเทียบกับเวิร์ดที่ตัวรับได้รับ เพื่อเลือกโคดเวิร์ดที่ใกล้เคียงกับเวิร์ดที่ตัวรับได้รับมามากที่สุด คือมีค่าดิสแทนส์ที่น้อยที่สุด ซึ่งเฟรมของข้อมูลแรกของโคดเวิร์ดที่คำนวณได้ จะได้จากการคัดเลือกโคดเวิร์ดที่เป็นเฟรมของข้อมูลแรกที่เข้ารหัสอยู่ ซึ่งจะถูกถอดรหัส หลังจากเปรียบเทียบกับเวิร์ดที่ได้รับ ดังนั้นบิตจำนวน n_0 ที่ได้รับใหม่นี้จะถูกชิฟไปยังตัวถอดรหัส แล้วเวิร์ดที่ได้รับมาจำนวน n_0 บิตเก่าก็จะทิ้งไป โดยทำการคำนวณเช่นเดิมอีก เพื่อหาเฟรมของข้อมูลถัดไป ซึ่งในทางทฤษฎีนั้น การถอดรหัสโดยใช้ดิสแทนส์ที่น้อยที่สุด นับว่าเป็นวิธีที่ดีที่สุดที่เป็นไปได้ โดยอัลกอริทึมของไวเทอบิ เป็นการถอดรหัสที่จัดการแบบซ้ำๆ กัน เฟรมต่อเฟรม โดยการไล่ตามกราฟทรีลิส เพื่อเลียนแบบการเข้ารหัส ซึ่งในช่วงของแต่ละเฟรมนั้น ตัวถอดรหัสจะไม่รู้ว่าได้ถอดรหัสไปยังโหนดใด แต่จากลำดับของข้อมูลที่ได้รับนั้น ตัวถอดรหัสจะพิจารณาจากเส้นทางที่คล้ายคลึงกันมากที่สุดจากทุกๆ โหนด ซึ่งเส้นทางที่มีค่าดิสแทนส์ที่น้อยที่สุดจะเรียกว่าเป็นเซอไวเวอร์ (Survivor) โดยทั่วไปจะมี 4 เซอไวเวอร์ในแต่ละระดับของสาขา และดิสแทนส์ระหว่างแต่ละเส้นทางกับลำดับของบิตที่ได้รับจากตัวส่ง ซึ่งดิสแทนส์นี้เรียกว่าดิสครีแพนซี (Discrepancy) ของเส้นทาง และในเฟรมถัดไป ตัวถอดรหัสก็จะพิจารณาเส้นทางที่คล้ายคลึงกันมากที่สุด ไปยังโหนดใหม่ของเฟรม เพื่อให้ได้โหนดใหม่ ซึ่งอาจจะหาเส้นทางไปยังโหนดใหม่ได้ โดยการขยายไปยังโหนดใหม่ โดยเริ่มจากโหนดเก่า แล้วพิจารณาความแตกต่างกันของแต่ละเส้นทางที่ขยายไป เพื่อเปรียบเทียบกับความแตกต่างของเส้นทางเก่า ซึ่งจะมีเส้นทางเช่นนี้ได้ q^v แบบ ซึ่งเป็นเส้นทางไปยังโหนดใหม่ โดยเส้นทางที่มีความแตกต่างกันน้อยที่สุดก็จะเป็นเส้นทางที่คล้ายคลึงกันมากที่สุดไปยังโหนดในเฟรมใหม่ ซึ่งการพิจารณาเส้นทางไปยังเซ็ตของโหนดในเฟรมที่ r ใดๆ ได้นั้น ตัวถอดรหัสจะต้องรู้โหนดที่คล้ายคลึงกันมากที่สุดของเฟรมแรกเสียก่อน

การสร้างการถอดรหัสแบบไวเทอบินั้น จะต้องเลือกช่วงกว้างของการถอดรหัสซึ่งมีความกว้าง b ซึ่งโดยทั่วไปจะมีขนาดเท่ากับความยาวบลอก และจะได้โคดเวิร์ดจากการสร้างแบบจำลองของการถอดรหัสในเฟรมที่ n โดยตัวถอดรหัสจะพิจารณาจากทุกๆ เส้นทางที่สอดคล้องกับสาขาก่อนหน้านี้ เพื่อให้สามารถถอด

รหัสเฟรมของข้อมูลได้แล้วจึงจะทิ้งสาขานั้น เพื่อไปจัดการกับเฟรมใหม่ของเวิร์ดที่ได้รับสำหรับการคำนวณครั้งถัดไป ซึ่งจะทำการถอดรหัสได้อย่างไม่สิ้นสุด และถ้าเลือก b ให้ยาวพอก็จะสามารถทราบจำนวนการตัดสินใจได้ว่าจะมีกี่ครั้งในแต่ละเฟรม ซึ่งบางครั้งเส้นทางที่ได้จากการถอดรหัสอาจผิดพลาด ก็อาจจะแฟลกไว้ ถ้าเป็นโคดเวิร์ดที่ไม่สามารถแก้ให้ถูกต้องได้ ซึ่งกรณีนี้จะเป็นการถอดรหัสอย่างไม่สมบูรณ์ และบางครั้งตัวถอดรหัสอาจจะตัดสินใจผิด ซึ่งจะสามารถแก้ไขได้ ถ้ารหัสนั้นเป็นรหัสที่ไม่ถูกทำลาย (Noncatastrophic Code)

การพิจารณา การถอดรหัสแบบไวเทอบิ เป็นช่วงกว้าง โดยการใช้ทรีลิสซึ่งเป็นช่วงที่มีความยาวจำกัด ถ้าพบว่ามีเส้นทางเดียวในเฟรมก็แสดงว่าการถอดรหัสนั้นจบสิ้นสมบูรณ์ แต่ถ้ามีมากกว่า 1 เส้นทางก็แสดงว่า สามารถสืบหาความผิดพลาดที่เกิดขึ้นได้ แต่ไม่สามารถแก้ให้ถูกต้องได้ ซึ่งอาจแฟลกไว้หรือเดาค่าที่อาจเป็นไปได้ ซึ่งการเพิ่มช่วงกว้างของการถอดรหัส จะไม่ทำให้สามารถแก้ความผิดพลาดนั้นได้ โดยรหัสที่จำเป็นต้องแก้ความผิดพลาดให้ได้นั้น จะต้องเป็นรหัสที่มีฟรีดิสแทนส์มาก ซึ่งการใช้การถอดรหัสแบบไวเทอบิจะเร็ว ตราบใดที่รหัสนั้นมีความยาวควบคุมสั้นๆ

2.2.2.8 อัลกอริทึมสำหรับการค้นหาแบบทรีลิส

ประสิทธิภาพของรหัสคอนโวลูชันนั้น จะปรับปรุงได้โดยการเพิ่มความยาวควบคุมของรหัส ซึ่งจะไม่เหมาะสมกับ การใช้การถอดรหัสแบบไวเทอบิ เช่น ที่ความยาวควบคุมของรหัสเป็น 10 ตัวถอดรหัสของรหัสที่เป็นเลขฐานสองจะต้องเก็บเส้นทางที่เป็นไปได้ไว้ถึง 1024 เส้นทาง ดังนั้นกรณีที่มีความยาวควบคุมยาวๆ จึงไม่เหมาะที่จะใช้ทรีลิส ซึ่งทุกๆ การค้นหาที่มีโอกาสเป็นไปได้สูง โดยใช้กราฟทรีลิสนั้นเรียกว่า การถอดรหัสแบบซีเควนเชียล (Sequential Decoding) หรือการถอดรหัสโดยใช้ความน่าจะเป็น (Probabilistic Decoding) เนื่องจากการใช้การถอดรหัสแบบทรีลิสนั้น ตัวถอดรหัสจะต้องหาเส้นทางผ่านกราฟทรีลิส ซึ่งมีความคล้ายคลึงกันกับเวิร์ดที่ได้รับมากที่สุด โดยใช้แฮมมิ่งดิสแทนส์ การถอดรหัสแบบซีเควนเชียล จึงเป็นการถอดรหัสที่มีประสิทธิภาพสูง เนื่องจากใช้การคำนวณน้อยและการเลือกเส้นทางที่ถูกต้องนั้น ทำโดยใช้หลักความน่าจะเป็นที่สูงที่สุด ซึ่งการเปรียบเทียบบิตของรหัสในแต่ละสาขาของกราฟ กับบิตที่ตัวรับได้รับมานั้น จะเป็นการทำแบบบิตต่อบิต

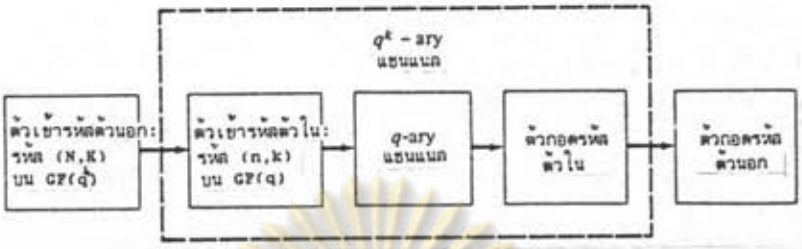
ดังนั้น แทนที่จะใช้อัลกอริทึมของไวเทอบิ ตัวถอดรหัสก็จะดูที่เฟรมแรกก่อน แล้วจึงตัดสินใจก่อนจะไปยังโหนดของทรีลิสในระดับแรกของเฟรมถัดไปก็จะเลือกสาขาที่ใกล้เคียงกับเฟรมที่ได้รับมามากที่สุด ซึ่งจะมีเพียงโหนดเดียว

ที่จะไปยังระดับถัดไป แต่ก็มีโอกาสที่ตัวถอดรหัสอาจจะถอดรหัสผิด แม้ว่าจะเป็นการตัดสินใจจากดิสแทนส์น้อยที่สุดก็ตาม ซึ่งถ้ามีความผิดพลาดเกิดขึ้น ก็มีโอกาสที่ตัวถอดรหัสจะเลือกสาขาผิดได้ ซึ่งก็จะไปพบกับความผิดพลาดอื่นๆ อีกได้ ซึ่งการใช้การถอดรหัสแบบซีเควนเชียล จะเก็บเส้นทาง 2 ถึง 3 เฟรมสุดท้ายไว้ เพื่อกรณีที่สามารถย้อนกลับมาหา (Trace) เส้นทางเก่าได้ ซึ่งประสิทธิภาพของตัวถอดรหัสนั้น ขึ้นอยู่กับ b ซึ่งเป็นความกว้างของการถอดรหัสในเฟรมของโคดเวิร์ด เมื่อตัวถอดรหัสพบเส้นทางจากเฟรมจำนวน b ไปยังทรีลิสแล้ว จึงจะตัดสินใจในเฟรมที่เก่าที่สุด เพื่อให้ได้บิตเอาท์พุท แล้วจึงชิพเฟรมใหม่เข้ามาในช่องการถอดรหัส

โดยทั่วไป การถอดรหัสแบบทรีลิสจะใช้งานได้ดี แต่ก็มีบางโอกาสที่อาจมีปัญหาบ้างที่หยุดชั่วขณะเพื่อกลับไปแก้จุดที่ผ่านมา แต่ก็พบได้น้อยมาก ซึ่งบางครั้งอาจมีการเก็บเฟรมเก่าๆ ไว้มาก จนบัฟเฟอร์ไม่สามารถเก็บไว้ได้ เรียกว่าเกิดโอเวอโฟลของบัฟเฟอร์ (Buffer overflow) ซึ่งเป็นข้อจำกัดหนึ่งของการถอดรหัสโดยใช้ทรีลิส

นอกจากนี้ ยังมีการจัดการรหัสโดยใช้ รหัสแบบผสม (Nested Codes) (Blahut 1983: 198 ; Deng 1987: 698) โดยรหัสแบบผสมเป็นการจัดการรหัสวิธีหนึ่ง เพื่อที่จะได้รหัสแบบบลอกที่มีความยาวบลอกเพิ่มขึ้นจากเดิม โดยการทำรหัสแบบผสมหรือที่เรียกว่ารหัสแบบคอนแคทาเนท (Concatenated Code) ซึ่งการเข้ารหัส และถอดรหัสจะต่อเนื่องกันไป โดยการทำรหัสแบบผสมนี้ จะเป็นการรวมรหัสขนาดเล็กให้เป็นรหัสที่ใหญ่กว่า จากการพิจารณาบลอกของสัญญลักษณ์จำนวน q-ary ซึ่งมีความยาว kK โดยบลอกจะถูกแบ่งเป็น K สับบลอก (Subblock) ของสัญญลักษณ์จำนวน k และแต่ละสับบลอกจะเป็นสมาชิกจาก q^k-ary

ลำดับของ K สับบลอกเหล่านี้ จะถูกเข้ารหัสได้เป็นรหัส (N,K) บน GF(q^k) แล้วแต่ละรหัสจำนวน N ของสัญญลักษณ์ q^k-ary จะถูกแปลงเป็นสัญญลักษณ์ k q-ary และเข้ารหัสอีกทีหนึ่ง ด้วย (n,k) q-ary โดยวิธีดังกล่าวนี้ รหัสแบบผสมจะเป็นลำดับของการเข้ารหัส 2 แบบที่แตกต่างกัน แสดงได้ดังรูปที่ 2.9



รูปที่ 2.9 แสดงลักษณะของรหัสแบบผสม

การใช้รหัสแบบผสมนั้น อาจเป็นการผสมระหว่าง รหัสแบบบลอกกับรหัสแบบบลอก หรือ รหัสแบบทรีกับรหัสแบบทรี หรือระหว่างรหัสแบบบลอกกับรหัสแบบทรี ซึ่งอาจเลือกใช้รหัสแบบบลอกกับรหัสคอนโวลูชันนัล เนื่องจากรหัสคอนโวลูชันนัลจะเหมาะสมกับ การแก้ไขความผิดพลาดที่เกิดขึ้นไม่บ่อยนัก แต่กรณีของความผิดพลาดที่เกิดในช่วงเวลาใกล้ๆ กันมากเกินไป อาจจะทำให้การถอดรหัสผิดได้ ซึ่งการใช้รหัสแบบบลอกร่วมด้วย จะช่วยแก้ความผิดพลาดเป็นช่วงเหล่านี้ได้ โดยเทคนิคของรหัสแบบผสมระหว่าง รหัสแบบบลอกกับรหัสคอนโวลูชันนัล จะทำให้ได้รหัสที่ทนต่อแชนแนลที่เกิดสัญญาณรบกวนแบบเกาส์เซียนได้ ซึ่งการเลือกใช้รหัสแบบผสมอาจใช้รหัสตัวใน (Inner code) เป็นรหัส ที่สามารถสืบหาและแก้ความผิดพลาดของข้อมูลได้ แล้วใช้รหัสตัวนอก (Outer code) เพื่อสืบหาความผิดพลาดของข้อมูลเท่านั้น ทั้งนี้ขึ้นอยู่กับความต้องการของผู้ใช้ ซึ่งการใช้รหัสแบบผสมจะเพิ่มความเชื่อถือได้ของข้อมูล และประสิทธิภาพของข้อมูลได้

2.3 ประสิทธิภาพของรหัสควบคุมความผิดพลาดของข้อมูล

การจะเลือกใช้รหัสรูปแบบใดนั้น นอกจากจะต้องรู้ว่า คุณสมบัติของรหัสแต่ละชนิดเป็นอย่างไรแล้ว ยังจำเป็นต้องรู้ว่ารหัสต่างๆ นั้นมีโอกาสที่จะถอดรหัสได้ข้อมูลที่ผิด (Decode Error) และการไม่สามารถถอดรหัสได้ (Decoding Failure) เป็นเท่าไรด้วย ซึ่งค่าเหล่านี้จะได้จาก

2.3.1 การกระจายของน้ำหนัก (Weight distributions)

สำหรับรหัสแบบบลอกเชิงเส้นแบบใดๆ จะมีดิสแทนส์ที่น้อยที่สุด คือ d^* ซึ่งอย่างน้อยหนึ่งโคดเวิร์ดจะต้องมีน้ำหนักเป็น d^* เสมอ โดยเราจะต้องรู้ว่า มีกี่โคดเวิร์ดที่มีน้ำหนักเป็น d^* และโคดเวิร์ดอื่นๆ มีน้ำหนักเป็นเท่าไร ซึ่งสำหรับรหัส

ขนาดเล็ก จะสามารถหาตารางของน้ำหนักได้โดย การค้นหาแบบเอกซ์ซอสทีฟ (Exhaustive search) ส่วนรหัสขนาดใหญ่นั้น จะต้องใช้เทคนิคของการวิเคราะห์ (Analytical technique) โดย

ให้ A_l เป็นจำนวนของโคดเวิร์ดซึ่งมีน้ำหนัก l ในรหัสเชิงเส้น (n,k) โดยจะมีเวคเตอร์ขนาด $(n+1)$ ซึ่งมีสมาชิกเป็น A_l สำหรับ $l = 0,\ldots, n$ ที่เรียกว่าเป็นการกระจายของน้ำหนักของรหัส ถ้าดิสแทนส์ที่น้อยที่สุดเป็น d^* ดังนั้น $A_0 = 1$, $A_1,\ldots, A_{d^*-1}$ จะเป็น 0 และ $A_{d^*} \neq 0$ ซึ่งการใช้การวิเคราะห์มาอธิบายจะใช้ได้กับรหัสแบบรีด-โซโลมอน หรือรหัสอื่นๆ ซึ่งเป็นรหัสที่มีดิสแทนส์มากที่สุด

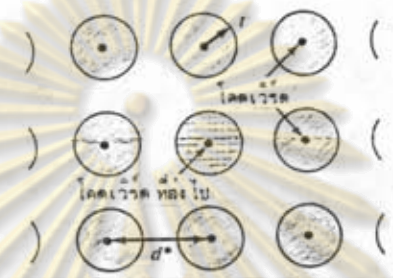
2.3.2 โอกาสของการถอดรหัสแล้วได้ข้อมูลที่ผิดพลาด และการไม่สามารถถอดรหัสได้

การถอดรหัสอย่างไม่สมบูรณ์ของรหัสที่สามารถแก้ความผิดพลาดได้เป็นจำนวน t นั้น จะสามารถแก้ความผิดพลาดทุกๆ รูปแบบที่มีน้ำหนัก t หรือน้อยกว่าได้ แต่จะไม่สามารถแก้ความผิดพลาดทุกรูปแบบที่มีน้ำหนักมากกว่า t ได้ ดังนั้นเมื่อเกิดความผิดพลาดของข้อมูลมากกว่า t ขึ้น ตัวถอดรหัสอาจจะได้ข้อมูลที่ไม่สามารถแก้ให้ถูกต้องได้ (Uncorrectable message) หรือเกิดเป็นการถอดรหัสแล้วได้ข้อมูลผิด ดังนั้นผลที่ได้จากการถอดรหัสอย่างไม่สมบูรณ์ ก็จะเป็นข้อมูลที่ถูกต้อง ส่วนข้อมูลที่ไม่ถูกต้อง (Erased message) หรือ การไม่สามารถถอดรหัสได้นั้น โดยทั่วไปการคำนวณโอกาสของเหตุการณ์ต่างๆ เหล่านี้จะไม่สามารถทำได้ แต่ก็มีบางกรณีที่อาจเขียนได้ในรูปของสมการได้ ในที่นี้จะศึกษากรณีของรหัสเชิงเส้นบนแชนแนล ที่ความผิดพลาดของข้อมูลเกิดเป็นอิสระแก่กัน และเป็นระบบ (Systemetrically) ซึ่งโอกาสเหล่านี้สามารถแสดงได้ในเทอมของการกระจายของน้ำหนัก $\{A_l\}$ ของรหัส และจะมีประโยชน์ในกรณีที่รู้การกระจายของน้ำหนัก ซึ่งจะสามารถหาโอกาสของการถอดรหัสแล้วได้ข้อมูลที่ผิด และการไม่สามารถถอดรหัสได้ โดยแชนแนลที่จะพิจารณาเป็น q-ary แชนแนล ซึ่งจะทำให้เกิดความผิดพลาดที่เป็นอิสระแก่กันด้วยโอกาส p และส่งข้อมูลที่ถูกต้องไปด้วยโอกาส $1-p$ ดังนั้นแต่ละความผิดพลาดของสัญญลักษณ์ $q-1$ ที่เกิดขึ้น จะเกิดขึ้นด้วยโอกาส $p/(q-1)$ และแต่ละรูปแบบที่เกิดความผิดพลาดเป็นจำนวน k จะมีโอกาสเกิดความผิดพลาดเป็น

$$p(k) = (p/(q-1))^k(1-p)^{n-k}$$

โดยพิจารณาเฉพาะการถอดรหัสอย่างไม่สมบูรณ์ ซึ่งตัวถอดรหัส จะถอดรหัสทุกๆ เวิร์ด

ที่ได้รับมาไปยังโคดเวิร์ดที่ใกล้ที่สุด ซึ่งอยู่ภายในดิสแทนส์ t เมื่อ t เป็นค่าคงที่ และสอดคล้องกับอสมการ $2t + 1 \leq d^*$ ซึ่งสามารถนำไปวิเคราะห์ในกรณีต่างๆ สำหรับรหัสเชิงเส้นได้ โดยการส่งเวิร์ดที่เป็น 0 (Zero-word) แล้วนำมาวิเคราะห์ ซึ่งทุกๆ โคดเวิร์ดที่มีรูปแบบเดียวกันก็จะมีโอกาสเช่นเดียวกัน แสดงได้ดังรูปที่ 2.10



รูปที่ 2.10 แสดงขอบเขต (Region) ของการถอดรหัส

ซึ่งเวิร์ดที่ตัวรับได้รับ จะตกอยู่ในขอบเขตเหล่านี้ โดยโอกาสที่จะถอดรหัสได้ถูกต้องคือ โอกาสที่เวิร์ดที่ได้รับอยู่ในขอบเขตที่เป็นตาราง (Crosshatched region) และโอกาสเกิดการถอดรหัสไม่ถูกต้องคือ โอกาสที่เวิร์ดที่ได้รับอยู่ในขอบเขตส่วนที่แรเงา (Shaded region) และโอกาสที่ไม่สามารถถอดรหัสได้คือ โอกาสที่เวิร์ดที่ได้รับอยู่ในขอบเขตสีขาว (White region) โดยผลรวมของโอกาสที่จะเกิดเหตุการณ์ทั้ง 3เท่ากับ 1 ดังนั้นต้องการเฉพาะ 2 สมการก็พอที่จะคำนวณหาค่าต่างๆ ได้

หฤษฎีที่ 2.3.2.1 การถอดรหัสที่สามารถแก้ความผิดพลาดได้ของส่วนที่เป็นรัศมี (Packing radius) ของรหัส จะมีโอกาสของการถอดรหัสได้ถูกต้องแสดงได้ดังสมการ

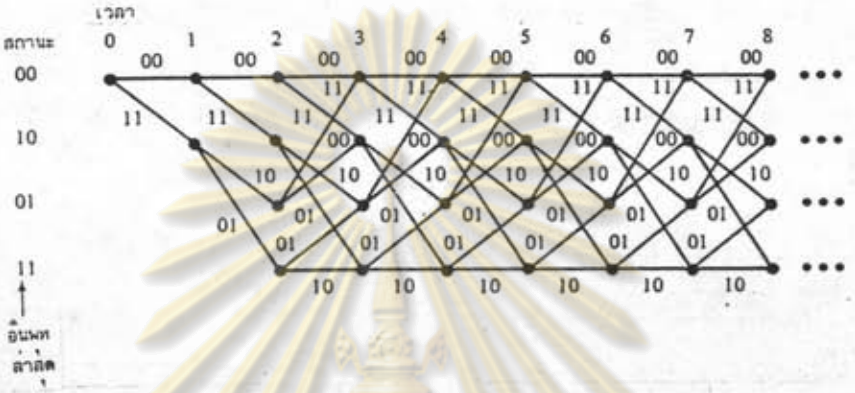
$$P_c = \sum_{v=0}^{t} \binom{n}{v} p^v(1-p)^{n-v}$$

โดยจะมี $\binom{n}{v}$ วิธีที่ความผิดพลาดจำนวน v จะถูกเลือก และแต่ละตัวจะมีโอกาสเท่ากับ $p^v(1-p)^{n-v}$

2.3.3 น้ำหนักการกระจายสำหรับรหัสคอนโวลูชันนัล

น้ำหนักของโคดเวิร์ดแบบคอนโวลูชันนัล ซึ่งมีความยาวไม่สิ้นสุด คือจำนวนของสมาชิกที่ไม่เป็น 0 และฟรีดิสแทนส์คือ น้ำหนักของโคดเวิร์ดที่มีน้ำหนักน้อยที่สุด โดยฟรีเลนท์คือ ความยาวของเซกเมนต์ของโคดเวิร์ดไปจนถึงเฟรมที่ไม่เป็น 0 ซึ่ง

เป็นเฟรมสุดท้ายของโคดเวิร์ดที่มีน้ำหนักน้อยที่สุด ถ้ารหัสนั้นอยู่ในรูปแบบง่ายพอที่จะแสดงในรูปของกราฟทรีลิสได้ ตัวแปรเหล่านี้ก็จะหาได้จากกราฟทรีลิส ตัวอย่างแสดงได้ดังรูปที่ 2.11



รูปที่ 2.11 แสดงกราฟทรีลิสของรหัสคอนโวลูชันนัล (6,3)

จากรูปมี 1 เส้นทาง (Path) ที่มีน้ำหนัก 5 และเป็นดิสแทนส์ที่น้อยที่สุดของรหัส โดยเส้นทางนี้จะใช้ 3 เฟรม และค่าฟรีเลนท์ของรหัสเป็น 6

2.3.4 ขอบเขตของดิสแทนส์ที่น้อยที่สุดสำหรับรหัสแบบบลอก

รหัสควบคุมความผิดพลาดของข้อมูลใดๆ ซึ่งมีความยาวบลอกเป็น n และมีอัตรา R ประสิทธิภาพของรหัสจะได้จากค่าดิสแทนส์น้อยที่สุด (d^*) เช่น ถ้ามีรหัส 2 แบบ ที่มีค่า n และ R เหมือนกัน เรามักจะเลือกรหัสที่มี d^* มากกว่า โดยถ้ามีรหัส C และต้องการรู้ว่า d^* จะใหญ่พอสำหรับรหัสใดๆ ที่มีความยาวบลอก n และมีอัตราเป็น R หรือไม่ โดยทั่วไปแล้ว ถ้าความยาวบลอกใหญ่มากจะหาไม่ได้ แต่ด้วยการวิเคราะห์แบบอะซิมโทติก (Asymptotic analysis) สำหรับค่าคงที่ q จะได้ว่า

$$d(n,R) = \max_C d^*(C)$$

ซึ่งค่ามากที่สุดก็คือทุกๆ รหัสที่มีความยาวบลอก n และอัตรา R โดยดิสแทนส์ที่น้อยที่สุดของรหัส C จะได้จากสมการ

$$d^*(C) = \min d(x,y) \quad \text{เมื่อ} \quad x,y \in c \text{ และ } x \neq o$$

โดยฟังก์ชัน $d(n,R)$ เป็นดิสแทนส์น้อยที่สุดที่ใหญ่ที่สุดของรหัสใดๆ บน GF(q) ซึ่งมีอัตรา R และความยาวบลอก n ยกเว้นสำหรับค่า n น้อยๆ

2.3.5 ขอบเขตของดิสแทนส์น้อยที่สุดสำหรับรหัสคอนโวลูชันนัล

ดิสแทนส์ที่น้อยที่สุดตัวที่ 1 หรือ d_1^*ของรหัสคอนโวลูชันนัล ที่มี

ความยาวบลอกสอดคล้องกับสมการ

$$n = (m+1)n_0$$

ซึ่งจัดว่าเป็น ค่าแฮมมิ่งดิสแทนส์ที่น้อยที่สุด ระหว่าง 2 เซกเมนต์ของโคดเวิร์ดใดๆ ที่แตกต่างกันในเฟรมเริ่มต้น โดยดิสแทนส์ที่น้อยที่สุด d^* ของรหัสคอนโวลูชันนัลจะขึ้นอยู่กับค่า d_{m+1}^* และค่าฟรีเลนท์จะได้จากสมการ

$$d = \max_1 d_1^*$$

โดยสามารถนำเซกเมนต์ขนาด N ใดๆ ของรหัสคอนโวลูชันนัลมาทำ เพื่อให้ได้รหัสแบบบลอก ซึ่งมีความยาว N แต่รหัสที่ได้นี้ จะไม่มีประสิทธิภาพดีกว่ารหัสแบบบลอก ซึ่งมีความยาว N ที่ดีที่สุดได้ โดยขอบเขตที่ได้นี้ สามารถเปรียบเทียบกับขอบเขตของรหัสแบบบลอกสำหรับค่าคงที่ q ซึ่งจะได้จากสมการ

$$d(n,R) = \max_C d^*(C)$$

โดยค่ามากที่สุดก็คือ ทุกรหัสคอนโวลูชันนัลที่มีความยาวบลอกเป็น n และมีอัตรา R โดย $d^*(C)$ เป็นดิสแทนส์ที่น้อยที่สุดของรหัส C และชั่งฟังก์ชั่น $d(n,R)$ เป็นดิสแทนส์ที่ใหญ่ที่สุดของรหัสคอนโวลูชันนัลใดๆ บน GF(q) ที่มีอัตรา R และมีความยาวบลอกเป็น n

2.3.6 ประสิทธิภาพของสัญญาณ สำหรับแชนแนลที่ถูกรบกวน (Noisy Channels)

รหัสควบคุมความผิดพลาดของข้อมูลจะให้ผลลัพธ์ที่ดีที่สุด ถ้ามีการออกแบบโมเดม (Modem) และรหัสที่ดี ซึ่งในส่วนนี้จะใช้บางส่วนของทฤษฎีข่าวสาร (Information theory) และทฤษฎีการสื่อสารข้อมูล (Communication theory) เพื่อให้ได้ รหัสควบคุมความผิดพลาดของข้อมูลที่ดี ซึ่งรหัสควบคุมความผิดพลาดของข้อมูลจะเป็นเครื่องมือ ช่วยให้การส่งสัญญาณสำหรับการสื่อสารข้อมูล ในแชนแนลที่ถูกรบกวน มีประสิทธิภาพมากขึ้น โดยอัลกอริทึมสำหรับการถอดรหัสสามารถปรับปรุงได้ จากการใช้ซอฟท์-ดีซีสชั่น (Soft-decision) ซึ่งโดยทั่วไปจะมีประสิทธิภาพดีกว่า การใช้ฮาร์ด-ดีซีสชั่น (Hard-decision) โดยฮาร์ด-ดีซีสชั่นเป็นการถอดรหัสโดยการวัดจากเมตริกซ์ ซึ่งก็คือการพิจารณาโดยใช้ค่าแฮมมิ่งดิสแทนส์ ส่วนซอฟท์-ดีซีสชั่นนั้น ได้จากการปรับปรุงเมตริกซ์ ให้เป็นลำดับของการสังเกตุการณ์อื่นๆ ด้วย เช่น การนำค่าอัตราของสัญญาณข้อมูลต่อสัญญาณที่เป็นสิ่งรบกวน (Signal to noise ratio) มาเป็นตัวแปรในการถอดรหัสด้วย แต่เนื่องจากการใช้ซอฟท์-ดีซีสชั่นมีความสลับซับซ้อนมากกว่า จึงมีประโยชน์เฉพาะรหัสเล็กๆ เท่านั้น

จากการพิจารณาแชนแนล ที่ถูกรบกวนด้วยสัญญาณเกาส์เซียน

ในช่วงสัญญาณยาว T นั้น โดยทั่วไปแชนแนลที่ส่งสัญญาณเป็นคลื่น จะรับอินพุทเป็นฟังก์ชั่น s(t) ใดๆ ที่ต่อเนื่องกัน สำหรับ t จาก 0 ถึง T โดย

$$E_m = \int_0^T s(t)^2 dt$$

ซึ่งอินทิกรัลของ $s(t)^2$ เรียกว่าพลังงานของข้อมูล (Message energy) หรือ E_m สำหรับคลื่น s(t)

ถ้า T มีค่าได้โดยไม่มีขีดจำกัด ดังนั้น กำลังเฉลี่ย s จะได้จากสมการ

$$S = \lim_{T \to \infty} (1/T) E_m$$

โดย $E_m(t)$ เป็นพลังงานของข้อมูลในช่วงความยาว T

ตามปรกติแชนแนลจะมีข้อจำกัด ในช่วงคลื่นที่ยอมรับได้ (Allowable waveform) จากช่วงกว้างของแชนแนล (Bandpass channel) ซึ่งจะส่งเฉพาะข้อมูลที่มอดูเลทแล้ว โดยสมการของคลื่นแสดงได้ดังสมการ

$$s_c(t) = a(t)\cos(2 f_o t + \theta(t))$$
$$= S_R(t)\cos 2 f_o t + s_I(t)\sin 2 f_o t$$

ซึ่ง a(t) และ θ(t) เป็นฟังก์ชั่นที่มีค่าจริง (Real-valued) เรียกว่าแอมปลิจูด และเฟสมอดูเลชั่น (Phase modulation) โดย $S_R(t)$ เรียกว่าส่วนจริง (Real) และ $S_I(t)$ เรียกว่าส่วนจินตภาพ (Imaginary) ของสมาชิกของมอดูเลชั่น โดยหลังจากเข้ารหัสแล้ว โคดเวิร์ด หรือลำดับของโคดเวิร์ด จะต้องจับคู่ไปยังสัญญาณมอดูเลชั่น (a(t),θ(t)) หรือ ($S_R(t)$,$S_I(t)$) ซึ่งการจับคู่นี้เรียกว่ามอดูเลชั่น และในทางกลับกันเรียกว่าดีมอดูเลชั่น (Demodulation)

แชนแนลที่มีช่วงกว้างของเกาส์เซียน ซึ่งมีอินพุท s(t) จะมีเอาท์พุทเป็น v(t) ซึ่งได้จากสมการ

$$v(t) = h(t) * (s(t) + n(t))$$

เมื่อ h(t) เป็นฟังก์ชั่นของช่วงกว้าง หรือที่เรียกว่า แรงตอบของอิมพัลส์ (Impulse respond) ของแชนแนล และ n(t) เป็นสัญญาณรบกวนที่เป็นฟังก์ชั่นตัวอย่าง จากกรณีของเกาส์เซียนซึ่งเป็นมาตราฐาน ที่นิยมใช้สำหรับพิจารณาคุณภาพของมอดูเลชั่น โดยสัญญาณรบกวนแบบเกาส์เซียนซึ่งเป็นมาตราฐานนั้น อาจจัดว่าเป็นไวท์นอยส์ได้ โดยสัญญาณรบกวนแบบไวท์นอยส์ จะมีความเข้มของสัญญาณเพียงด้านเดียวขนาด N_o วัตต์/เฮิร์ต (Watts/Hertz)

จากหฤษฎีการสื่อสารข้อมูล ข้อมูลที่เป็นดิจิตอล (Digital)

นั้น สัญญาณที่เข้าสู่แชนแนล และสัญญาณที่ออกมาจากแชนแนล ต้องพิจารณาเฉพาะส่วนที่ต่อเนื่อง (Discrete) เท่านั้น โดยมีเวลาเป็นค่าคงที่คือ $k\Delta t$ ของสัญญาณเข้า ซึ่งเป็นช่วงเวลาที่ต่อเนื่อง จะประกอบด้วย

$$S(k\Delta T) = S_R(k\Delta T) + jS_I(k\Delta T)$$

โดย $S_k = S_{Rk} + jS_{Ik}$ สำหรับ $k = 0,\pm1,\pm2,\ldots$

เมื่อ S_{RK} และ S_{Ik} เป็นจำนวนจริง

และเมื่อเลือก $h(t)$ เพื่อให้ $h(k\Delta t) = 0$ สำหรับ k ที่ไม่เป็น 0 และเท่ากับ 1 สำหรับ k ที่มีค่าเป็น 0 ดังนั้นสมการของ $v(t)$ ที่ได้จากตัวอย่างช่วงเวลาที่ต่อเนื่อง (Discrete Sample) คือ

$$v_k = s_k + n_k$$

$$= (s_{RK} + n_{RK}) + j(s_{Ik} + n_{Ik}) \quad \text{เมื่อ } k = 0,\pm1,\pm2,\ldots$$

โดย n_k เป็นตัวอย่างที่เวลา $k\Delta T$ ของสัญญาณรบกวนที่เพิ่มเข้าไปในแชนแนลแบบอนาลอก (Analog channel)

2.3.7 อัตราการเกิดความผิดพลาดของข้อมูล

ในระบบการสื่อสารข้อมูลจะมีการเข้ารหัส และมอดูเลทบิตของข้อมูลที่ตัวส่ง ซึ่งจะถูกดีมอดูเลทและถอดรหัสที่ตัวรับ โดยคุณภาพของระบบขึ้นอยู่กับโอกาสของการเกิดความผิดพลาดของข้อมูล หรือเรียกว่าอัตราการเกิดความผิดพลาดของข้อมูล (Bit error rate) หรือ BER ซึ่งการประมาณค่า BER ตั้งบนสมมุติฐานว่า เหตุการณ์การเกิดความผิดพลาดในแชนแนล เกิดเป็นอิสระต่อกัน ถึงแม้ว่าในความเป็นจริงแล้ว การสื่อสารข้อมูล และสื่อการเก็บข้อมูลต่างๆ เช่น สื่อสายโทรศัพท์ หรือ เทปแม่เหล็ก อาจจะมีรูปแบบของความผิดพลาด เป็นรูปแบบใดรูปแบบหนึ่ง ขึ้นอยู่กับชนิดของสิ่งรบกวน และลักษณะของแชนแนล เช่น สัญญาณรบกวนแบบเกาส์เซียน ซึ่งการพิจารณาเลือกรูปแบบของรหัส ให้เหมาะสมกับสัญญาณรบกวนแบบใดแบบหนึ่งแล้ว หากมีสัญญาณรบกวนแบบอื่นๆ เช่น อิมพัลนอยส์ หรือ อินเทอเฟียเรนส์ด้วย ก็จะทำให้รูปแบบของการเกิดความผิดพลาดของข้อมูลที่ได้เปลี่ยนไป จึงไม่มีรูปแบบของการคำนวณใดที่เหมาะสม เนื่องจากรูปแบบของความผิดพลาด ที่ได้จากสัญญาณรบกวนแบบหนึ่ง อาจจะไม่เหมาะกับสัญญาณรบกวน อีกแบบหนึ่งได้ ดังนั้น การใช้ BER จึงเป็น ค่าเฉลี่ยของรูปแบบความผิดพลาดของข้อมูล (Error distribution) (Knowles 1988: 767 ; Pursley 1987: 1 ; Yamada 1987: 21) ที่ได้จากสภาพความเป็นจริง

โดยพิจารณาจากแชนแนล ซึ่งสื่อสารข้อมูลด้วยรหัสเลขฐานสอง

จะมีลำดับของความผิดพลาดคือ

$$E = \{ e_1, e_2, \ldots \}$$

เมื่อ $e_i = 1$ ถ้าบิตในตำแหน่งที่ i เกิดความผิดพลาด

และ $e_i = 0$ ในกรณีอื่นๆ

การประมาณค่า BER นั้น P(e) จะได้จากเอาท์พุทบิตจำนวน n บิต โดย

$$P(e) = \sum_{i=1}^{n} e_i$$

และค่า BER ที่ใกล้เคียงกับค่าจริงคือ P(e) เมื่อค่า n เข้าใกล้ ∞

ซึ่งค่า P(e) เมื่อ n มีค่ามากนั้น การกระจายของโอกาสการเกิดความผิดพลาดของข้อมูลจะอยู่ในรูปการกระจายแบบนอร์มอล (Normal distribution) โดยที่โอกาสการเกิดความผิดพลาดของข้อมูล P(e) และ BER จะใช้แทนกันได้ แม้ว่าจะแตกต่างกันในทางความหมาย เนื่องจาก P(e) ได้จากทฤษฎีทางคณิตศาสตร์ โดยการคำนวณอัตราการเกิดความผิดพลาดของข้อมูล ส่วน BER จะมีความหมายในทางปฏิบัติของการบันทึกความผิดพลาดของข้อมูลที่เกิดขึ้นจริง (Mortimer 1987: 1113 ; Wang 1987: 1231)

โดยทั่วไป BER จะมีค่าประมาณ 5×10^{-3} บิต และส่วนใหญ่มักเกิดเป็นแบบสุ่ม (Random) ที่เป็นอิสระต่อกัน มากกว่าความผิดพลาดเป็นช่วง ซึ่งการพิจารณาประสิทธิภาพของรหัสแบบต่างๆ โดยการใช้ BER เป็นส่วนสำคัญ ในการพิจารณารหัสแบบต่างๆ นั้น จึงนับว่าเป็นวิธีที่ดี