การเข้ารหัสควบคุมความผิดพลาดโดยใช้คอนโวลูชันแนลโค๊ด

ERROR CONTROL CODING USING CONVOLUTIONAL CODE

โดย

นายสงคราม สุนทราญู

นางสาวอังสุมารินทร์ สุพรรณรัตน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

การเข้ารหัสควบคุมความผิดพลาดโดยใช้คอนโวลูชันแนลโค๊ด

ERROR CONTROL CODING USING CONVOLUTIONAL CODE

โดย

นายสงคราม สุนทราญู 40010804

นางสาวอังสุมารินทร์ สุพรรณรัตน์ 40010983

อาจารย์ที่ปรึกษา

ดร.สุทธิชัย นพนาคีพงษ์

อ.กฤษณ์ วงศ์รุจิระ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **การเข้ารหัสควบคุมความผิดโดยใช้คอนโวลูชันแนลโค๊ด**

ERROR CONTROL CODING USING CONVOLUTIONAL CODE

ผู้จัดทำ

1. นายสงคราม สุนทราญู 40010804
2. นางสาวอังสุมารินทร์ สุพรรณรัตน์ 40010983

อาจารย์ที่ปรึกษา

(ดร.สุทธิชัย นพนาคีพงษ์)

อาจารย์ที่ปรึกษา

(อ. กฤษณ์ วงศ์รุจิระ)

การเข้ารหัสควบคุมความผิดโดยใช้คอนโวลูชันแนลโค๊ด

Error Control Coding Using Convolutional Code

โดย นายสงคราม สุนทราญู 40010804
นางสาวอังสุมารินทร์ สุพรรณรัตน์ 40010983

อาจารย์ที่ปรึกษา ดร.สุทธิชัย นพนาคีพงษ์
อ. กฤษณ์ วงศ์รุจิระ

บทคัดย่อ

ในระยะเวลาหลายปีที่ผ่านมานี้ ได้มีความต้องการเพิ่มขึ้นเพื่อประสิทธิภาพและความน่าเชื่อถือในการส่งข้อมูลดิจิตอล ในการส่งข้อมูลดิจิตอลนั้นจะต้องส่งผ่านตัวกลางจากต้นทางไปยังปลายทางซึ่งสิ่งที่สำคัญที่ต้องพิจารณาคือ การควบคุมความผิดพลาดจากสภาพแวดล้อมที่จะเข้ามารบกวนสัญญาณ ดังนั้นปริญญานิพนธ์ฉบับนี้จะนำการใช้คอนโวลูชันแนลโค๊ด เพื่อลดความน่าจะเป็นของการเกิดความผิดพลาดในการส่งข้อมูล เนื่องจากการส่งสัญญาณรบกวน โดยปริญญานิพนธ์นี้จะนำซอฟแวร์มาประยุกต์ใช้งานซึ่งสามารถแบ่งออกได้เป็น 2 ส่วน คือ ส่วนจำลองเหตุการณ์ และส่วนประยุกต์ใช้งานจริง

ข้อมูลดิจิตอลจากคอมพิวเตอร์จะถูกแบ่งออกเป็นแพ็คเก็ตและทำการเข้ารหัสการประสาน แล้วถูกส่งออกทางพอร์ตอนุกรมเพื่อส่งไปยังคอมพิวเตอร์ด้านรับ ผ่านช่องสัญญาณที่มีการรบกวนบนสายส่งทองแดง คอมพิวเตอร์ทางด้านรับก็จะนำสัญญาณที่รับได้มาถอดรหัสการประสาน ซึ่งประสิทธิภาพของการเข้ารหัสนั้นจะแสดงด้วยกราฟโดยเปรียบเทียบกับข้อมูลที่ไม่ได้ทำการเข้ารหัส

ABSTRACT

In recent years , there has been an increasing demand for efficient and reliable digital data transmission , The transmission of digital information must be transmitted from source to destination via medium . A major concern is the control of error from noisy environment. Therefore this thesis will use convolutional code to minimize the error transmitting probability due to noise. This thesis will use software in application by divided in 2 major parts, Simulation and Practical Application part.

Digital data from computer will be separated in packets and encoded with convolutional code, then will be transmitted from serial port through noisy channel via copper wire. The receiving computer will receive data and decode them. The efficiency of coding will be depicted by graph compares with uncoded data.

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

หน้า

สารบัญรูปภาพ (ต่อ)

หน้า

สารบัญตาราง

บทที่ 1

บทนำ

ในปัจจุบันความต้องการสำหรับระบบการสื่อสารและจัดเก็บข้อมูลดิจิตอลที่มีประสิทธิภาพและเชื่อถือได้นั้นได้เพิ่มขึ้นอย่างรวดเร็ว ความต้องการเหล่านี้ได้ถูกกระตุ้นจากการเกิดขึ้นของโครงข่ายแลกเปลี่ยนข้อมูลข่าวสารที่มีขนาดใหญ่และความเร็วในการส่งข้อมูลที่สูงขึ้น โดยสิ่งสำคัญที่นักออกแบบระบบจะต้องพิจารณาถึงคือ การควบคุมความผิดพลาดในการรับส่งข้อมูลเพื่อที่เราจะสามารถนำหรือสร้างข้อมูลกลับคืนมาได้อย่างน่าเชื่อถือ

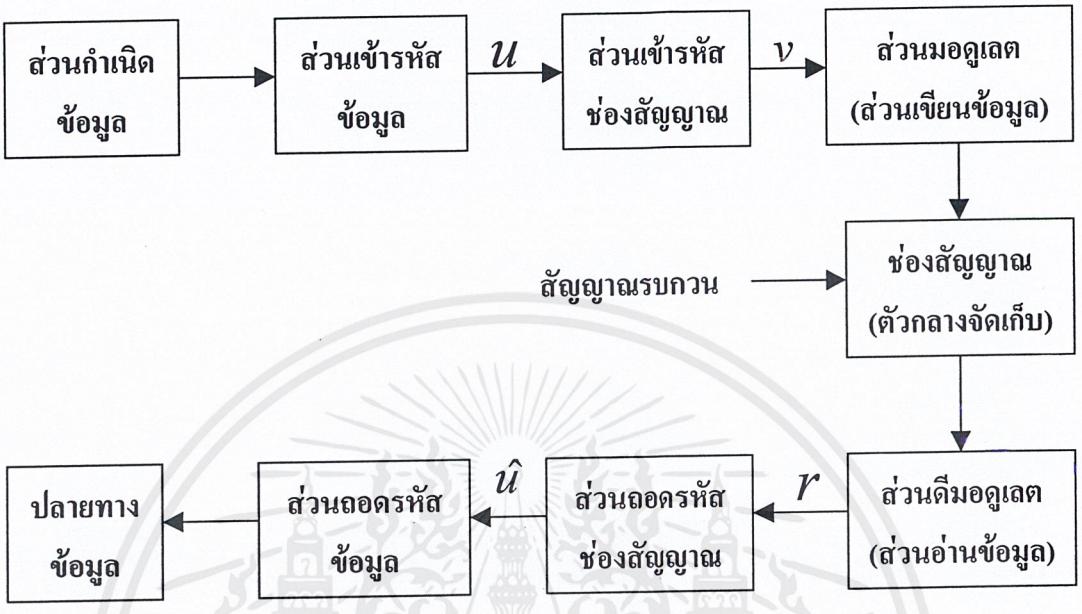
ในปี ค.ศ. 1948 นาย Claude Shannon ได้นำเสนอทฤษฎีซึ่งเป็นรากฐานสำคัญของทฤษฎีข่าวสาร (Information Theory) โดยทฤษฎีของ Shannon นั้นได้แสดงถึงคุณลักษณะของช่องสัญญาณ (Channel) โดยค่าของระดับสัญญาณรบกวน, แบนวิดท์ และระดับกำลังงานของสัญญาณ ซึ่งถูกใช้ในการกำหนดหาค่าตัวแปร C หรือความจุของช่องสัญญาณ (Channel Capacity) โดยเป็นตัวกำหนดขอบเขตสูงสุดของอัตราการส่งสัญญาณข้อมูลผ่านช่องสัญญาณ ซึ่งยังคงรักษาระดับความน่าเชื่อถือที่ทางด้านรับสามารถรับได้ ดังนั้นตราบใดที่เรายังส่งข้อมูลด้วยอัตราที่ต่ำกว่า C แล้ว ความน่าจะเป็นของความผิดพลาดในการรับส่งข้อมูลนั้นสามารถทำให้มีค่าน้อยลงตามต้องการได้ โดยการใช้สัญญาณเข้ารหัสที่มีความยาวของสัญญาณที่เหมาะสม หลังจากการนำเสนอของ Shannon ก็ทำให้เกิดการวิจัยอย่างกว้างขวางในการคิดสร้างรหัสที่ใช้ในการควบคุมความผิดพลาด ซึ่งในปัจจุบันเราสามารถแบ่งรหัสออกเป็น 2 ประเภทคือ ลิเนียร์บล็อคโค๊ด (Linear block codes) และ การเข้ารหัสประสาน (Convolutional codes)

สำหรับการออกแบบระบบนั้น ในการที่จะลดความน่าจะเป็นของความผิดพลาดในการส่งข้อมูลผ่านช่องสัญญาณนั้น สามารถกระทำได้ 2 วิธีคือ การเพิ่มค่ากำลังงานที่ใช้ในการส่งสัญญาณหรือการเข้ารหัสควบคุมความผิดพลาด แต่สำหรับวิธีการเพิ่มค่ากำลังงานนั้นจำเป็นต้องปรับเปลี่ยนอุปกรณ์ขนาดและน้ำหนักของอุปกรณ์จะเพิ่มขึ้นนั่นหมายถึงจะทำให้ต้องเสียค่าใช้จ่ายที่สูงขึ้นด้วย ส่วนการใช้การเข้ารหัสควบคุมความผิดพลาดนั้นจะทำให้ค่าแบนวิดท์ของระบบเพิ่มขึ้น ดังนั้นนักออกแบบระบบจำเป็นต้องคำนึงถึงทรัพยากรที่มีอยู่

บทที่ 2

ทฤษฎีและหลักการ

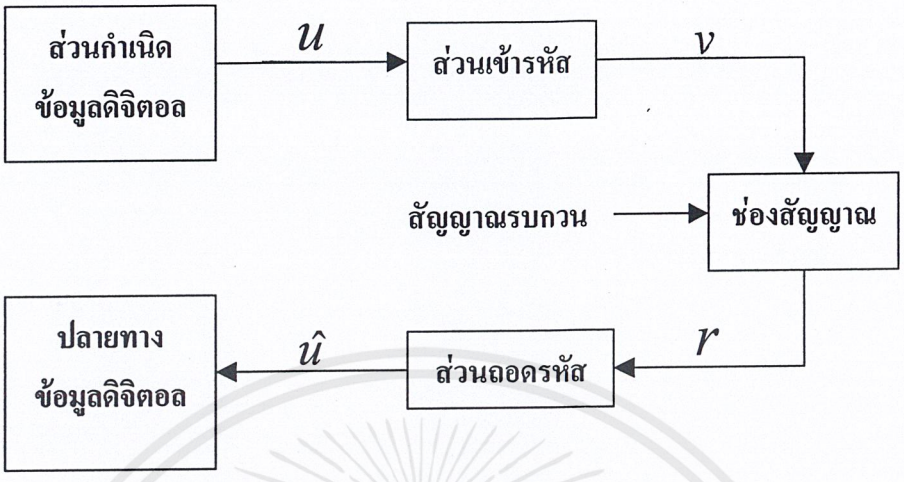
2.1 โครงสร้างของระบบการสื่อสารดิจิตอลพื้นฐาน



รูปที่ 2.1 แสดงบล็อคไดอะแกรมของการส่ง-รับข้อมูลหรือการจัดเก็บข้อมูลโดยทั่วไป

ในระบบการสื่อสารโดยทั่วไปสามารถแสดงได้ดังรูปที่ 2.1 โดยแหล่งกำเนิดข้อมูล (Information Source) จะให้กำเนิดข้อมูลข่าวสาร เช่น เสียงพูดของคน ซึ่งจะถูกทำการเข้ารหัสข้อมูล (Source Encoding) ทำให้ได้ลำดับสัญญาณดิสครีต u ซึ่งเรียกว่า **ลำดับข้อมูล** (Information Sequence) โดยลำดับดังกล่าวนี้จะถูกส่งเข้าไปยังตัวเข้ารหัสช่องสื่อสาร (Channel Encoder) เพื่อเพิ่มประสิทธิภาพให้แก่ระบบซึ่งจะได้ลำดับสัญญาณ v หรือเรียกว่า **คำรหัส** (Code Word) ออกมา แต่เนื่องจากสัญญาณดีสครีตที่ได้นี้ไม่เหมาะสมต่อการส่งผ่านช่องสัญญาณสื่อสารจึงจำเป็นต้องทำการมอดูเลตเพื่อแปลงลำดับสัญญาณดีสครีตให้อยู่ในรูปแบบคลื่น (Waveform) ที่เหมาะสมต่อการส่งผ่านช่องสัญญาณ (Channel) ไปยังด้านรับ โดยระหว่างทางของช่องสัญญาณนั้นจะประกอบด้วยสัญญาณรบกวนต่างๆ ที่เข้ามามีอิทธิพลต่อสัญญาณที่เราทำการส่ง โดยทางด้านรับนั้นก็จะประกอบด้วยส่วนต่างๆซึ่งทำหน้าที่ย้อนกลับจากที่กล่าวไว้ข้างต้น โดยจากรูปที่ 2.1 นั้นลำดับสัญญาณ r คือค่าของสัญญาณที่ได้จากการตัดสินทางด้านเครื่องรับโดยตัวดีมอดูเลเตอร์ (Demodulator) ส่วนลำดับสัญญาณ $\bar{u}$ คือค่าของสัญญาณที่ได้หลังจากผ่านการถอดรหัสช่องสัญญาณแล้ว ซึ่งในทางอุดมคตินั้นค่าของลำดับสัญญาณ u และ $\bar{u}$ ควรจะมีค่าเหมือนกัน กล่าวคือ ไม่เกิดความผิดพลาดในการรับส่งข้อมูลเลย แต่เนื่องจากอิทธิพลของสัญญาณรบกวนจึงอาจมีผลทำให้เกิดความผิดพลาดขึ้น เพราะฉะนั้นการเข้ารหัสควบคุมความผิดพลาดจึงเป็นการลดค่าความน่าจะเป็นของความผิดพลาด (Probability of error) ให้น้อยที่สุด

ในกรณีที่แหล่งกำเนิดข้อมูลของเรานั้นอยู่ในรูปของสัญญาณดิจิตอลอยู่แล้ว เราสามารถดัดแปลงรูปแบบโครงสร้างของระบบได้ดังรูปที่ 2.2

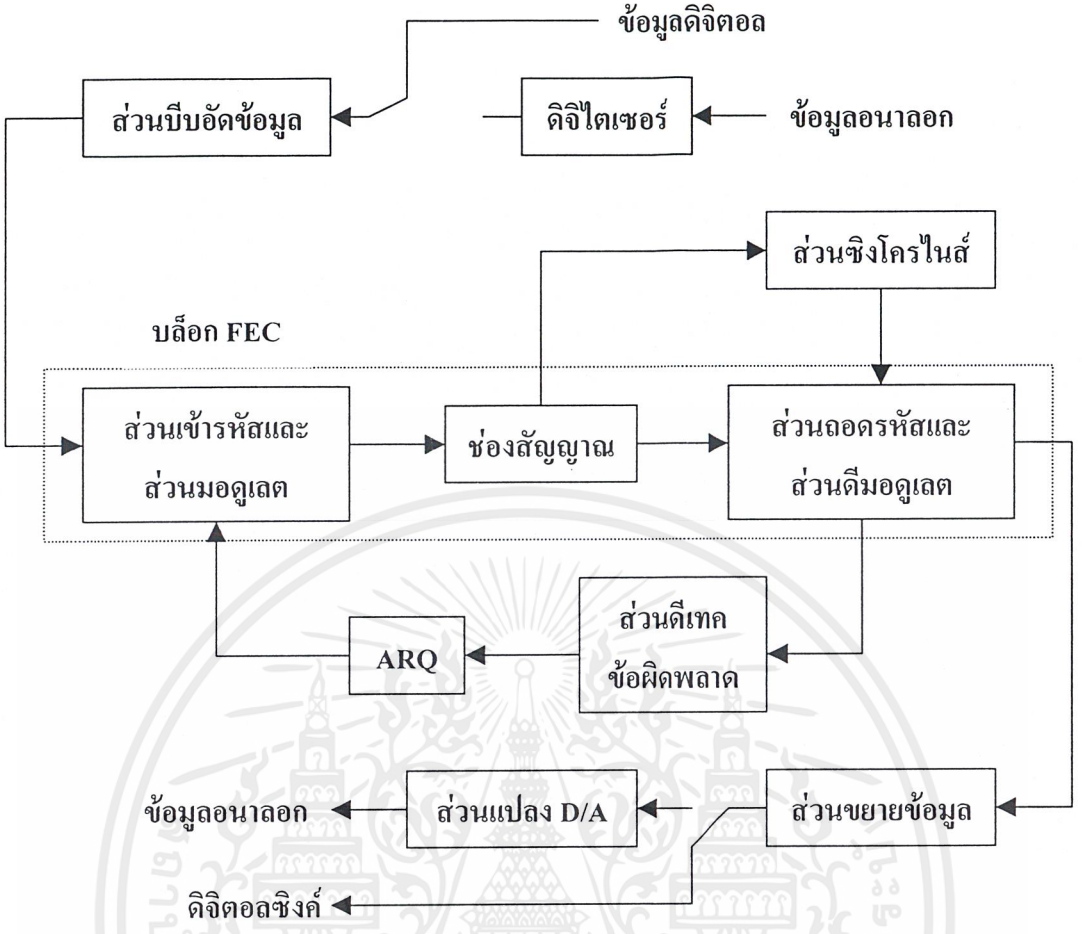


รูปที่ 2.2 แสดงแบบจำลองของระบบการเข้ารหัส

2.2 รูปแบบของการควบคุมความผิดพลาด

สำหรับระบบที่แสดงในรูปที่ 2.1 และ 2.2 นั่นเป็นระบบทิศทางเดียว (one – way system) ซึ่งการส่งสัญญาณนั้นถูกจำกัดเพียงแค่ทิศทางเดียวจากเครื่องส่งไปยังเครื่องรับ เช่น ในกรณีที่ระยะห่างของการติดต่อในระยะทางไกล เช่น ในการสื่อสารในอวกาศซึ่งระบบการควบคุมความผิดพลาดแบบนี้เรียกว่า “Forward Error Correcting ” (FEC) โดยที่รหัสที่ทำการใส่เพิ่มเข้าไปนั้นมีความสามารถในการแก้ไขความผิดพลาดที่เกิดขึ้นที่ด้านรับได้ด้วยตัวเอง

สำหรับการสื่อสารข้อมูล 2 ทิศทางนั้น ในการควบคุมความผิดพลาดยังสามารถใช้เทคนิคที่เรียกว่า “Automatic Repeat Request ”(ARQ) ซึ่งหากทางด้านรับตรวจจับความผิดพลาดที่เกิดขึ้นได้ในการรับข้อมูล เครื่องรับจะส่งสัญญาณร้องขอไปยังด้านส่งเพื่อให้ส่งข้อมูลนั้นซ้ำอีกครั้ง โดยเทคนิคแบบ ARQ นั้นมีความซับซ้อนของอุปกรณ์น้อยกว่าแบบ FEC โดยโครงสร้างระบบแสดงดังรูปที่ 2.3



รูปที่ 2.3 แสดงบล็อคของ FEC(Forward Error Control)

2.3 ตัวแปรสุ่มและฟังก์ชันการแจกแจง

2.3.1 ตัวแปรสุ่ม (Random Variable)

ในการศึกษาการทดลองนั้นผลลัพธ์ของการทดลองอาจอยู่ในรูปแบบของตัวเลขหรือไม่ใช่ตัวเลข การที่จะเปลี่ยนผลลัพธ์ของการทดลองให้เป็นตัวเลขทั้งหมดนั้น เราจะต้องใช้ตัวแปรสุ่ม (Random - Variable) โดยทั่วไปตัวแปรสุ่มนิยมเขียนด้วยตัวอักษรใหญ่ ดังเช่นเรามีตัวแปรสุ่ม X ที่มีค่า x_i โดย i คือค่าดัชนี ซึ่งบอกให้รู้ถึงลำดับของผลลัพธ์ในการทดลองนั้นและเราสามารถเขียนค่าความน่าจะเป็นของตัวแปรสุ่ม X ที่มีค่า x_i แทนด้วยสัญลักษณ์ $P_x(x_i)$

ตัวแปรสุ่มจะเป็นชนิดดีสครีตเมื่อ x_i มีค่าแยกกันอย่างชัดเจนและเงื่อนไขสำหรับตัวแปรชนิดดีสครีตที่เกี่ยวกับโอกาสการเกิดของมัน ก็คือ

$$\sum_i P_x(x_i) = 1 \qquad (2.1)$$

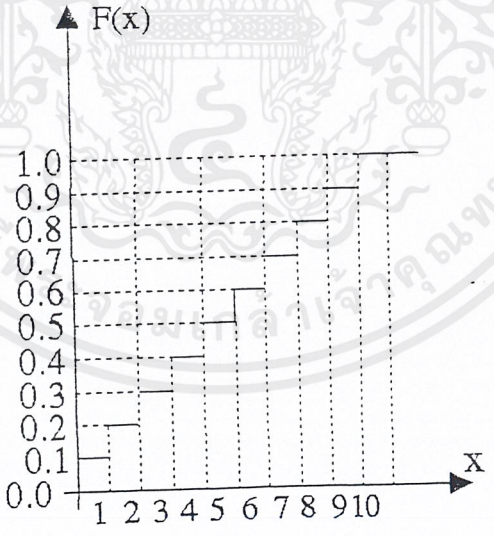
ตัวแปรสุ่มชนิดดีสครีตจะมีจำนวนที่จำกัด ในกรณีที่ตัวแปรสุ่มมีจำนวนไม่จำกัดและค่าของมันต่อเนื่องอยู่ในช่วงของตัวแปรที่แน่นอน ตัวแปรสุ่มชนิดนี้จะถูกเรียกว่าเป็นตัวแปรสุ่มชนิดต่อเนื่อง (Continuous Random Variable)

2.3.2 ฟังก์ชันการแจกแจง (Distribution Function)

จากการศึกษาตัวแปรสุ่มที่ผ่านมาจะเห็นได้ว่าค่าตัวเลขที่ตัวแปรสุ่มจะได้รับนั้นเป็นไปได้หลายอย่างและค่าต่างๆเหล่านี้อาจจะมีการแจกแจงเป็นค่าดีสครีตหรือค่าต่อเนื่องขึ้นอยู่กับธรรมชาติของปัญหา โดยฟังก์ชันการแจกแจงความน่าจะเป็นของตัวแปรสุ่มสามารถแสดงได้ดังนี้ ถ้าให้ X เป็นตัวแปรสุ่มและ x เป็นค่าจริงใดๆใน $(-\infty, \infty)$ ฟังก์ชันการแจกแจงของตัวแปรสุ่ม X ก็คือความน่าจะเป็นของเหตุการณ์ที่ตัวแปรสุ่มมีค่าน้อยกว่าหรือเท่ากับ x ซึ่งเขียนแทนด้วยสัญลักษณ์ F_x

$$F_x(x) = P(X \leq x) \tag{2.2}$$

ซึ่ง $F_x(x)$ นี้ถูกเรียกว่าฟังก์ชันการแจกแจงสะสม (Cumulative Distribution Function) ของ X หรือเรียกย่อว่า CDF ดังแสดงในรูปที่ 2.4



รูปที่ 2.4 รูปแสดงฟังก์ชันการแจกแจงสะสม

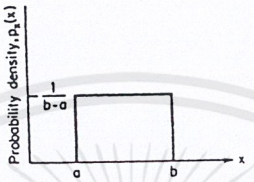
โดยค่าอนุพันธ์ของ $F_x(x)$ เมื่อเทียบกับ x นี้จะนิยมแทนด้วย $p_x(x)$ ซึ่งมีชื่อเรียกว่า ฟังก์ชันความหนาแน่นของความน่าจะเป็น (Probability Density Function) ของตัวแปรสุ่ม X หรือเรียกย่อว่า PDF

2.3.2.1 การแจกแจงแบบยูนิฟอร์ม (Uniform Distribution)

ตัวแปรสุ่ม X มีค่าความหนาแน่นของความน่าจะเป็นที่เท่าๆกันในช่วง (a,b) ซึ่งแสดงได้ในรูปที่ 2.5 และฟังก์ชันความหนาแน่นของความน่าจะเป็น เขียนได้ดังนี้

$$p_x(x) = \frac{1}{b-a} \qquad \text{เมื่อ } a \le x \le b \tag{2.3}$$

$$= 0 \qquad \text{สำหรับกรณีอื่น}$$



รูปที่ 2.5 แสดงรูปการแจกแจงแบบยูนิฟอร์ม

2.3.2.2 การแจกแจงแบบเกาส์ (Gaussian Distribution)

การแจกแจงแบบนี้เรียกอีกชื่อได้ว่า การแจกแจงแบบปกติ (Normal Density Function) โดยค่า PDF สามารถเขียนได้ดังนี้

$$p_x(x) = \frac{1}{(\sqrt{2\pi\sigma})\exp[-(x-m)^2/2\sigma^2]} \tag{2.4}$$

โดยค่ามัธยฐาน (Mean) m กำหนดได้เป็น

$$m = \int_{-\infty}^{\infty} x\, p_x(x)\, dx = E(x) \tag{2.5}$$

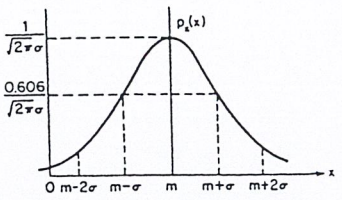
และค่ามัธยฐานกำลังสอง คือ

$$E(x^2) = \int_{-\infty}^{\infty} x^2\, p_x(x)\, dx \tag{2.6}$$

ส่วนค่าเบี่ยงเบนมาตรฐาน (Standard Deviation) σ หาได้จาก

$$\sigma = [E(x^2) - m^2]^{1/2} \tag{2.7}$$

และค่า σ^2 เรียกว่า ค่าความแปรปรวน (Variance)

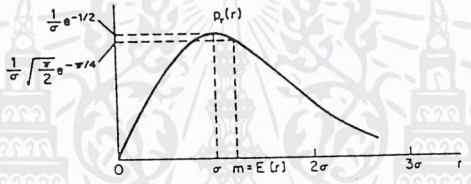


รูปที่ 2.6 แสดงรูปการแจกแจงแบบเกาส์

2.3.2.3 การแแจกแจงแบบเรย์ลี (Rayleigh Distribution)

การแจกแจงลักษณะนี้ปกติพบบ่อยในการแสดงถึงค่าการเฟดดิ้ง (Fading) ของสัญญาณวิทยุเคลื่อนที่ในลักษณะ Short – term ซึ่งค่า PDF เขียนได้ดังนี้

$$p_r(r) = (r/\sigma^2)\exp\left\{-r^2/2\sigma^2\right\} \quad \text{สำหรับ } r \geq 0 \tag{2.8}$$



รูปที่ 2.7 แสดงรูปการแจกแจงแบบเรย์ลี

2.3.2.4 การแจกแจงแบบทวินาม (Binomial Distribution)

การแจกแจงแบบนี้มักพบบ่อยในการใช้หาค่าความน่าจะเป็นของเหตุการณ์ เช่น การหาอัตราความผิดพลาดของสัญญาณ เช่น ให้การทดลองมีผลลัพธ์ที่เป็นไปได้อยู่ 2 ค่า คือ 0 และ 1 โดยค่าความน่าจะเป็น คือ $P(0) = p$ และ $P(1) = q = 1-p$ มีการทดลองกระทำซ้ำ m ครั้งและปรากฏค่า “0” จำนวน n ครั้ง ดังนั้นการแจกแจงแบบทวินามคือ

$$P_m(n) = \binom{m}{n} p^n q^{m-n} \tag{2.9}$$

เรากล่าวว่าสมการที่ 2.9 คือ ค่าความน่าจะเป็นที่จะตรวจวัดความผิดพลาด n ครั้ง ใน m ครั้ง โดยถ้าหาก “0” แสดงถึงค่าผิดพลาดและ “1” แสดงถึงค่าถูกต้อง

2.3.2.5 การแจกแจงแบบปัวส์ซอง (Poisson Distribution)

การแจกแจงแบบนี้ใช้แสดงการแจกแจงตามธรรมชาติที่เกิดในลักษณะพัลส์สั้นๆ ที่ไม่ได้คาดหมายหรือ spike ซึ่งปรากฏในสัญญาณที่เข้ามา ในการแจกแจงแบบทวินามดังสมการที่ 2.9 นั้น ถ้าหากจำนวนค่าของ m มีค่ามากและ p มีค่าน้อย สมการจะไม่สามารถแก้ได้ แต่อย่างไรก็ตาม

ถ้าค่าของ m และ p ยังคงมีค่าจำกัด (Finite) ผลลัพธ์ของสมการที่ 2.9 สามารถประมาณโดยการแจกแจงแบบปัวส์ซองได้คือ

$$p(n) = \frac{e^{-\overline{n}}(\overline{n})^2}{n!} \qquad (2.10)$$

โดยค่า $\overline{n} = E(n) = mp$, $\sigma^2 = \overline{nq} \cong \overline{n}$

ยกตัวอย่างเช่น ถ้าค่าสุ่ม N จุด อยู่ในช่วง (0 , T) แล้ว ค่าความน่าจะเป็นของการมีค่า n จุด ใน ΔT ในช่วงเวลา (0 ,T) คือ

$$p(n) = \frac{e^{-\mu\Delta T}(\mu\Delta T)^n}{n!} \qquad (2.11)$$

โดย $\mu = N/T$

2.3.3 ฮีสโทแกรม

ในการวิเคราะห์ข้อมูลของการทดลอง จะต้องทำการประเมินการวัดจำนวนมาก ดังนั้น ปัญหาแรกก็คือ การออกแบบวิธีการบันทึกและแสดงค่าตัวเลขที่วัดได้ สมมุติว่าเราทำการประเมินค่าที่วัดได้จำนวน 10 ค่า (หน่วยเป็น เซนติเมตร) ดังสมการ

26, 24, 26, 28, 23, 24, 25, 24, 26, 25 (2.12)

อันดับแรกทำการจัด สมการ 2.12 เป็นดังนี้

23, 24, 24, 24, 25, 25, 26, 26, 26, 28 (2.13)

นำข้อมูลที่ได้ไปเขียนลงในตารางที่ 2.1

ตารางที่ 2.1 แสดงค่าตัวอย่างกับจำนวนครั้งในการพบ

ค่า X_k	23	24	25	26	27	28
จำนวนครั้งที่พบ n_k	1	3	2	3	0	1

จากตารางจะเห็นได้ว่า X_k เมื่อ k = 1,2,3, ... จะมีค่าต่างกันโดย x1= 23, x2= 24 , x3= 25 , ... ตามลำดับและ n_k เมื่ k= 1,2,3,.... จะแสดงจำนวนครั้งที่เกิดโดย n_1= 1, n_2 = 3,... ตามลำดับ จากตารางที่ 2.1 เราสามารถหาค่า Mean ($\overline{x}$) ได้จาก

$$\bar{x} = \frac{\sum_i x_i}{N}$$

$$= \frac{23 + 24 + 24 + 24 + 25 + \ldots + 28}{10}$$

หรือ

$$\bar{x} = \frac{23 + (24 * 3) + (25 * 2) + \ldots + 28}{10}$$

เขียนในรูปทั่วไปได้ดังนี้

$$\bar{x} = \frac{\sum_k x_k n_k}{N} \qquad (2.14)$$

จากตัวอย่างสมการเป็นการบวกค่าที่วัดได้ทั้งหมด แต่ในสมการ 2.14 เป็นการบวกค่าที่ต่างกัน โดยแต่ละค่าจะคูณด้วยจำนวนครั้งที่เกิด ซึ่งสมการ 2.14 จะใช้ได้ดีกว่าเมื่อมีการวัดในปริมาณที่มาก การบวกแบบสมการ 2.14 เรียกว่า " การบวกเพิ่มน้ำหนัก " (Weighted Sum) ซึ่งค่า x_k แต่ละค่าจะถูกเพิ่มน้ำหนัก (Weighted) ด้วยจำนวนครั้งที่เกิด (n_k) ซึ่งถ้ารวมค่า n_k ในแต่ละค่าจะเขียนได้ดังนี้คือ

$$\sum_k n_k = N \qquad (2.15)$$

จากแนวความคิดที่กล่าวข้างต้นแทนที่จะบอกว่าที่ $x = 24$ เกิดขึ้น 3 ครั้งนั้น ก็สามารถบอกได้ว่าที่ $x = 24$ จะได้รับ 3/10 ของค่าวัดทั้งหมดซึ่งเขียนในรูปเศษส่วนดังสมการ

$$F_k = \frac{n_k}{N} \qquad (2.16)$$

ซึ่งเศษส่วนของการวัดจำนวน N ครั้ง จะทำให้ได้ x_k ซึ่งจากสมการ 2.15 สามารถเขียนใหม่ได้ดังนี้คือ

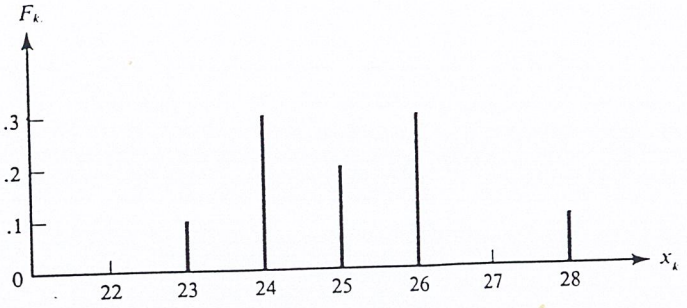
$$\bar{x} = \sum_k x_k F_k \qquad (2.17)$$

และจากสมการที่ 2.16 จะได้ว่า

$$\sum_k F_k = 1 \qquad (2.18)$$

จากสมการผลบวกจะมีค่าเท่ากับ 1 ซึ่งเรียกว่า " Normalized " และความสัมพันธ์ในสมการที่ 2.18 จะเรียกว่า " Normalization Condition "

การกระจายของการวัดสามารถแสดงฮีสโทแกรม (Histogram) ได้ดังรูป 2.8 โดยในแกนตั้งเป็นค่า F_k และแกนนอนเป็นค่า x_k



รูปที่ 2.8 แสดงค่าฮีสโทแกรมที่วัดค่า x 10 ค่า

ฮีสโทแกรมในรูปที่ 2.8 จะเรียกว่า " Bar Histogram " ซึ่งการกระจายของข้อมูลแสดงได้โดยความสูงในแกนตั้ง ฮีสโทแกรมแบบนี้จะเหมาะกับค่า x_k ที่เป็นเลขจำนวนเต็ม อย่างไรก็ตาม การวัดส่วนใหญ่ไม่ได้ให้ผลตัวเลขที่เหมาะสมเนื่องจากปริมาณนั้นมีค่าความเป็นไปได้ของการต่อเนื่อง ยกตังอย่างดังสมการที่ 2.19

26.4, 23.9, 25.1, 24.6, 22.7, 23.8, 25.1, 23.9, 25.3, 25.4 (2.19)

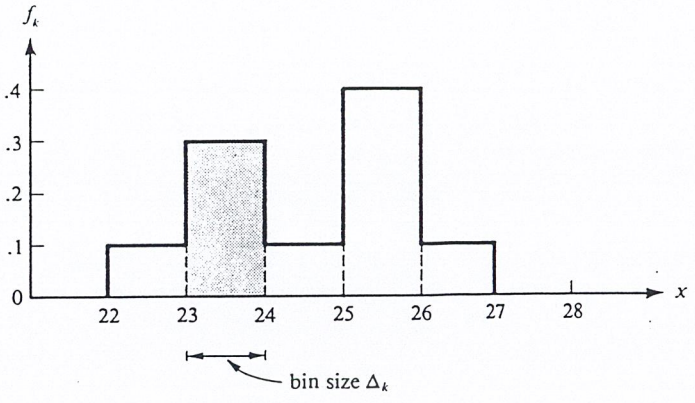
บาร์ฮีสโทแกรม (Bar Histogram) ของค่า 10 ค่านี้จะมีความสูงเท่ากัน จากค่าที่วัดได้ในสมการที่ 2.19 หากมีการจัดค่าเป็นช่วงหรือถัง (Bins) และนับว่าค่านั้นๆตกลงไปในถัง (Bins) จำนวนเท่าใด ยกตัวอย่าง หากจัดค่า x เป็นช่วงๆ จะได้ดังตารางที่ 2.2

ตารางที่ 2.2 แสดงค่า Bin Histrogram และจำนวนครั้งในการพบ

ถัง (Bins)	22-23	23-24	24-25	25-26	26-27	27-28
ข้อมูลในถัง (Observation in Bin)	1	3	1	4	1	0

จากตารางที่ 2.2 เมื่อนำไปพล็อตฮีสโทแกรม จะเรียกว่า " Bin Histogram " แสดงดังรูปที่ 2.2 ในการพล็อตการกระจายของการวัดที่ตกในแต่ละถังแสดงโดยพื้นที่สี่เหลี่ยมที่แรเงา จากรูปพื้นที่ที่แรเงามีช่วง $x = 23$ ถึง $x = 24$ มีพื้นที่ $0.3 * 1 = 0.3$ นั้นคือมีค่า 3/10 ของการวัดทั้งหมดที่ตกในช่วงนี้ จากรูปค่าความกว้างในแต่ละถังจะมีค่า Δk ความสูงเทากับ f_k ดังนั้นมีขนาดพื้นที่เท่ากับ $f_k\Delta k$ ซึ่ง

$f_k\Delta k$ = การกระจายการวัดในถังที่ k



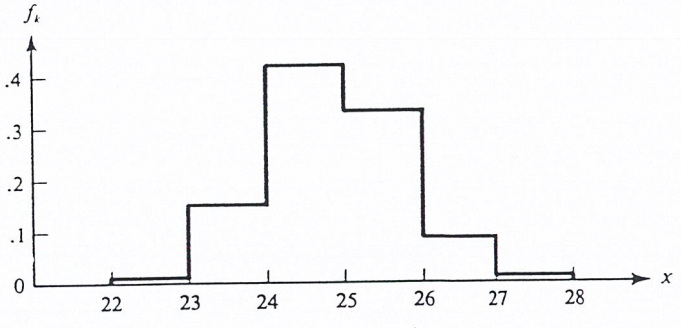
รูปที่ 2.9 แสดงค่า Bin Histrogram

หรือพูดอีกแง่ก็คือใน Bin Histogram มีพื้นที่ $f_k\Delta k$ ของสี่เหลี่ยมที่ k ซึ่งมีลักษณะคล้ายกับความสูง F_k ของบาร์ที่ k ใน Bar Histogram

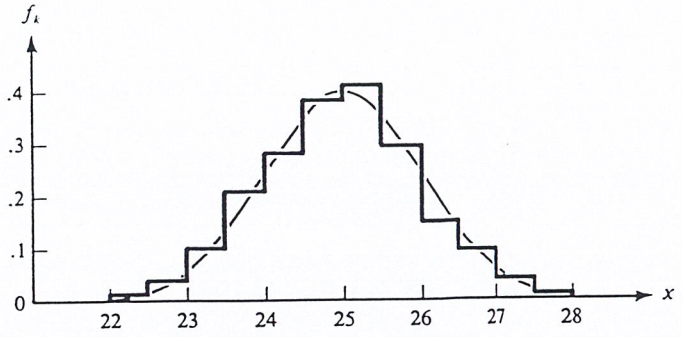
ในการเลือกค่าความกว้าง Δk หาก Δk กว้างเกินไป ค่าที่อ่านได้ทั้งหมดก็ตกลงในถังเดียว และฮีสโทแกรมก็จะได้แท่งสี่เหลี่ยมแท่งเดียว แต่ถ้า Δk แคบเกินไป เราก็จะได้แท่งสี่เหลี่ยมที่แคบมาก และมีความสูงเกือบเท่ากัน จึงควรเลือกค่าให้เหมาะสม ดังนั้นหากจำนวนที่วัดได้ (N) มีน้อยก็ควรจะใช้ค่าที่กว้าง แต่ถ้าจำนวนที่วัด (N)มีน้อยก็ควรจะใช้ค่าที่กว้าง แต่ถ้าจำนวนที่วัด (N) มีค่ามากขึ้นก็เลือกใช้ค่าที่แคบ

2.3.4 การจำกัดการกระจาย

ในการประเมินส่วนใหญ่ จำนวนการวัดที่เพิ่มจะทำให้ฮีสโทแกรมทีรูปร่างที่แน่ชัดมากขึ้นซึ่งแสดงได้ดังรูปที่ 2.10 และ 2.11 ซึ่งมีจำนวนการวัด 100 และ 1000 ค่า ตามลำดับ และเป็นข้อมูลเดียวกับรูปที่ 2.9 จากรูปที่ 2.10 การวัด 100 ค่า ฮีสโทแกรมที่ได้จะมีค่าสูงสุดช่วงเดียวซึ่งจากรูปจะสมมาตร ส่วนรูปที่ 2.10 การวัด 1000 ค่า ฮีสโทแกรมที่ได้จะค่อนข้างเรียบและสม่ำเสมอ จากภาพทั้ง 3 นี้จะแสดงถึงคุณสมบัติที่สำคัญสูงสุดของการวัด คือ ยิ่งจำนวนของการวัดมีค่าเข้าใกล้อินฟินิตี้แล้ว เคริฟที่ได้จะต่อเนื่อง (Continuous Curve) ซึ่งเคริฟต่อเนื่องนี้เรียกว่า " การจำกัดการกระจาย " (Limiting Distribution) ดังนั้นการวัดดังรูปที่ 2.9 ถึงรูปที่ 2.11 การจัดการกระจาย (Limiting Distribution) จะได้เคริฟที่มีรูปร่างคล้ายระฆังและสมมาตร

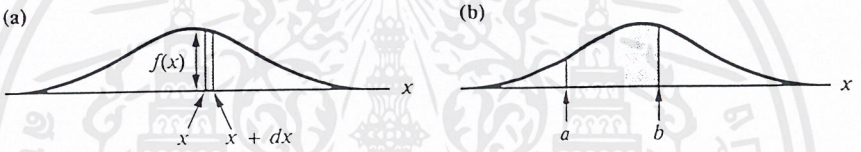


รูปที่ 2.10 แสดงค่าฮีสโทแกรมที่ทำการวัดค่า 100 ค่า



รูปที่ 2.11 แสดงค่าฮีสโทแกรมที่ทำการวัดค่า 1000 ค่า

เนื่องจากว่าการจำกัดการกระจายเป็นการสร้างขึ้นตามทฤษฎีซึ่งไม่สามารถทำการวัดด้วยตนเองได้ ยิ่งเราทำการวัดมากขึ้นเท่าไหร่ ฮีสโทแกรมที่ได้ก็จะยิ่งเข้าใกล้การจำกัดการกระจาย อย่างไรก็ตามได้มีหลักฐานที่ว่าการวัดทั้งหมดที่มีการจำกัดการกระจายซึ่งถ้าฮีสโทแกรมยิ่งเข้าใกล้กันมากเท่าใดก็ต้องทำการวัดมากขึ้น



รูปที่ 2.12 แสดงค่าการจำกัดการกระจาย

(a) แสดงการวัดจำนวนมากซึ่งค่าจะตอยู่ระหว่าง x และ x+dx

(b) แสดงค่าที่ตกอยู่ระหว่าง x=a และ x=b ดังพื้นที่ที่แรเงา

การจำกัดการกระจาย เช่นเคริฟในรูปที่ 2.11 ซึ่งเรียกว่า f(x) ความหมายของฟังก์ชันแสดงดังรูปที่ 2.12 ยิ่งเราทำการวัดมากขึ้นเท่าใดค่าฮีสโทแกรมจะไม่สามารถจำแนกได้ ดังนั้นการกระจายของการวัดซึ่งตกในช่วงห่างที่เล็กจาก x ถึง x+dx เท่ากับค่าพื้นที่ f(x)dx ดังรูปที่ 212 (a)

$$f(x)dx = \text{การกระจายการวัดซึ่งอยู่ระหว่าง } x \text{ ถึง } x+dx \qquad (2.20)$$

โดยทั่วไปแล้วการกระจายของการวัดจะตกระหว่าง 2 ค่าคือ a และ b ซึ่งค่าพื้นที่รวมของกราฟจะอยู่ระหว่าง x= a และ x= b (รูป 2.12 b) พื้นที่นี้เกิดจากการอินทิเกรตแบบจำกัดขอบเขตของฟังก์ชัน f(x) ดังสมการ

$$\int_a^b f(x)dx = \text{การกระจายการวัดที่มีค่าระหว่าง } x = a \text{ และ } x = b \qquad (2.21)$$

จากสมการทั้งสอง สมการที่ 2.21 จะเป็นการวัดค่าตัวเลขที่มีขนาดใหญ่มากและมีประโยชน์มาก ส่วนค่าของ f(x)dx เป็นค่าของความน่าจะเป็นของการวัดค่าๆหนึ่งของ x ซึ่งมีค่าอยู่ระหว่าง x และ x+dx ซึ่งจะได้ว่า

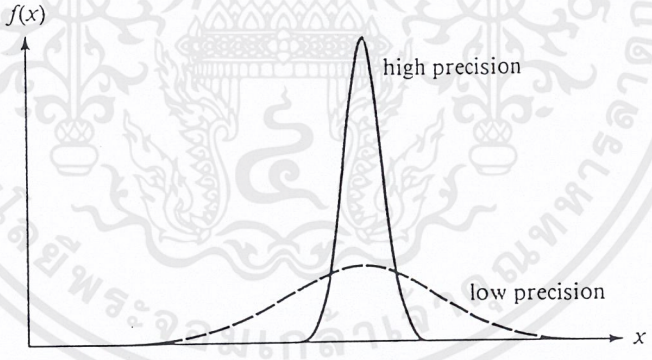
f(x)dx = ความน่าจะเป็นที่การวัดค่าๆหนึ่งจะให้คำตอบอยู่ระหว่าง x และ x+dx (2.22)

และในแง่เดียวกันอินทิกรัลของ $\int_a^b f(x)dx$ เป็นค่าความน่าจะเป็นที่การวัดค่าๆหนึ่งจะตกอยู่ระหว่าง x= a และx= b สรุปก็คือ ถ้าเรารู้การจำกัดการกระจาย f(x) สำหรับวัดค่าปริมาณของ x แล้ว ค่าความน่าจะเป็นของคำตอบจะอยู่ในช่วง $a \leq x \leq b$

จากค่าความน่าจะเป็นของคำตอบรวมมีค่าในช่วง $(-\infty, \infty)$ แล้ว f(x) จะได้ดังสมการ

$$\int_{-\infty}^{\infty} f(x)dx = 1 \qquad (2.23)$$

จากสมการที่ 2.18 และสมการที่ 2.21 จะกล่าวได้ว่า ถูกนอมอไลท์ (to be normalized) จากการวัดอยู่ภายใต้การพิจารณาที่แน่นอนแล้วค่าทั้งหมดจะมีค่าใกล้เคียงกับค่าจริงของ x ดังนั้น ฮีสโทแกรมของผลและการจำกัดการกระจายจะมียอดที่แคบ (Narrow Peaked) ดังที่ Solid Curve ในรูปที่ 2.13 หากการวัดนี้มีความแน่นอนต่ำแล้วค่าที่ได้จะแผ่กระจายและให้ค่าที่กว้างและต่ำดัง Dashed Curve ในรูปที่ 2.13



รูปที่ 2.13 แสดงค่าการกระจายจำกัด 2 ค่าซึ่งแสดงการวัดที่มีความถูกต้องสูง และการวัดที่มีความถูกต้องต่ำ

การจำกัดการกระจาย f(x) จะสามารถนำไปหาค่า Mean x โดยจะหาได้หลังจากการวัดค่าหลายค่าจากสมการที่ 2.17 ค่า Mean หาได้ดังสมการคือ

$$\bar{x} = \sum_k x_k F_k \qquad (2.24)$$

แต่ในกรณีนี้ค่าที่วัดได้มีจำนวนมาก ซึ่งหากหาค่า F_k แล้วจะได้ว่า $F_k = f(x_k)dx_k$ ดังนั้น จะจัดรูปสมการที่ 2.24 ได้ดังนี้คือ

แต่ในกรณีนี้ค่าที่วัดได้มีจำนวนมาก ซึ่งหากหาค่า F_k แล้วจะได้ว่า $F_k = f(x_k)dx_k$ ดังนั้น จะจัดรูปสมการที่ 2.24 ได้ดังนี้คือ

$$\bar{x} = \int_{-\infty}^{\infty} xf(x)dx \qquad (2.25)$$

สมการนี้ใช้ในการหาค่า Mean เมื่อมีการทดลองจำนวนมากและทำให้สามารถหาค่าเบี่ยงเบนมาตรฐาน (Standard Deviation) ได้ดังสมการคือ

$$\sigma_x^2 = \int_{-\infty}^{\infty} (x-\bar{x})^2 f(x)dx \qquad (2.26)$$

2.4 สัญญาณรบกวน

ในธรรมชาตินั้นมีปริมาณทางกายภาพหลายอย่างที่มีการเปลี่ยนแปลงอยู่ตลอดเวลา โดยหากฎเกณฑ์ที่แน่นอนไม่ได้ ในเรื่องของสัญญาณไฟฟ้าก็เช่นกัน สัญญาณบางชนิดเกิดการเปลี่ยนแปลงอยู่ตลอดเวลาตามธรรมชาติและอาจจะมารบกวนสัญญาณข่าวสารที่ต้องการให้เกิดความผิดเพี้ยน ถ้าหากเป็นสัญญาณที่เราไม่พึงปรารถนาแล้ว เราจัดมันว่าเป็นสัญญาณรบกวนทั้งสิ้น โดยการรบกวนของสัญญาณจะถูกกล่าวว่าเป็นแบบ Additive หรือ Multiplication ก็ขึ้นอยู่กับลักษณะที่มันเข้าไปกระทำกับสัญญาณข่าวสาร ซึ่งการรบกวนแบบ Additive นั้นจะมีลักษณะแค่เพียงไปทับ (Superimposed) กับสัญญาณข่าวสาร ส่วนสัญญาณรบกวนแบบ Multiplication นั้น สามารถมองได้ว่าเป็นการมอดูเลตทางแอมปลิจูดของสัญญาณข่าวสารโดยสัญญาณรบกวน

2.4.1 สัญญาณรบกวนแบบ Additive

เราสามารถแบ่งชนิดของสัญญาณรบกวนแบบ Addition ที่เครื่องรับได้ดังนี้

2.4.1.1 สัญญาณรบกวนเนื่องจากอุณหภูมิ (Thermal Noise)

โดยปรกติเมื่ออะตอมของสสารได้รับพลังงานความร้อน อิเล็กตรอนจะถูกทำให้เคลื่อนไหวอย่างไร้กฎเกณฑ์ มากหรือน้อยขึ้นอยู่กับพลังงานที่ได้รับ การเคลื่อนไหวของอิเล็กตรอนทำให้เกิดกระแสหรือศักดาไฟฟ้าขึ้นในสสารนั้น ปรากฏการณ์เช่นนี้ ทำให้เกิดสัญญาณรบกวนขึ้นมีชื่อเรียกว่า " สัญญาณรบกวนเนื่องจากอุณหภูมิ " ซึ่งเมื่ออาศัยหลักการกลศาสตร์ควอนตัม (Quantum Mechanic) จะรู้ว่าค่าความหนาแน่นกำลังเชิงสเปกตรัมหรือ PSD ของสัญญาณรบกวนเนื่องจากอุณหภูมิ $Sn_T(\omega)$ นี้มีค่าขึ้นกับความถี่ ω และอุณหภูมิ T แต่จะไม่ขึ้นกับค่าความต้านทาน R ของสสารนั้นเลย กล่าวคือ

$$Sn_T(\omega) = h|\omega|\left[1 + 1/\Pi\left\{e^{h|\omega|/(2\pi KT)} - 1\right\}\right] \qquad (2.27)$$

โดย h คือ ค่าคงที่ของแพลงค์ (Palnk's constant) = 6.2×10^{-34} จูล – วินาที

ที่อุณหภูมิห้องปกติ (≈ 300 K) และคิดที่ความถี่ $f < 1000$ GHz ค่า $h|\omega| / 2\pi KT$ จะมีค่า 0.024 จึงทำให้ประมาณได้ว่า $e^{h|\omega|/2\pi KT} \approx \frac{1+h|\omega|}{2\pi KT}$ ดังนั้นสมการที่ (2.27) จะประมาณมีค่าดังนี้ คือ

$$Sn_T(\omega) \approx h|\omega| \{ 1+ 2KT/ h|\omega| \}$$
$$\approx 2KT \quad W/Hz$$

จะเห็นได้ว่า PSD ของสัญญาณรบกวนเนื่องจากอุณหภูมินั้นมีค่าประมาณคงที่ ในย่านความถี่ต่ำกว่า 1000 GHz ที่อุณหภูมิปกติ จึงถือได้ว่าสัญญาณรบกวนเนื่องจากอุณหภูมินั้นเป็นตัวอย่างของสัญญาณรบกวนขาว (White noise) ที่มีย่านความถี่จำกัด

2.4.1.2 สัญญาณรบกวนจากบรรยากาศ (Atmospheric noise)

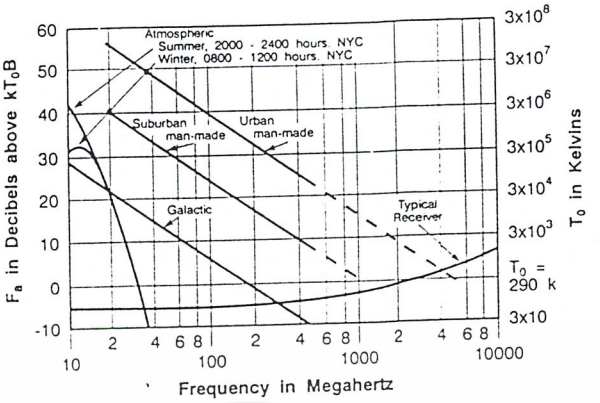
การปลดปล่อยประจุที่เกิดจากฟ้าผ่าหรือฟ้าแลบนั้นคือ แหล่งกำเนิดสำคัญของสัญญาณรบกวนชนิดนี้และระดับของมันจะเปลี่ยนแปลงตามความถี่ของสัญญาณ ลักษณะภูมิประเทศ ฤดู เวลา และสภาพอากาศ จากการสังเกตเราพบว่าค่าของสัญญาณรบกวนชนิดนี้จะมีระดับลดลง ณ บริเวณตำแหน่งของละติจูดที่เพิ่มขึ้น

2.4.1.3 สัญญาณรบกวนจากอวกาศ (Galactic noise)

สัญญาณรบกวนทุกชนิดที่มีแหล่งกำเนิดจากนอกโลก เราถือว่าเป็นสัญญาณรบกวนจากอวกาศทั้งสิ้นโดยแหล่งกำเนิดที่สำคัญของสัญญาณรบกวนชนิดนี้มาจากดวงอาทิตย์ สัญญาณรบกวนชนิดนี้มีอิทธิพลต่อสัญญาณที่มีความถี่ในย่าน 15 MHz –100 GHz โดยบรรยากาศไอโอโนสเฟียร์ (Ionosphere) จะแสดงหน้าที่เป็นตัวกรองความถี่สูงผ่าน และบรรยากาศ (Atmosphere) จะแสดงหน้าที่เป็นตัวกรองความถี่ต่ำผ่าน

2.4.1.4 สัญญาณรบกวนจากมนุษย์ (Man – made noise)

สัญญาณรบกวนที่เกิดจากการกระทำของมนุษย์นั้นมีสาเหตุมาจาก มอเตอร์ไฟฟ้า , สายไฟ , เครื่องจักร และอื่นๆ โดยจะมีในบริเวณตัวเมืองมากกว่าชนบท ซึ่งบางครั้งค่ากำลังงานของสัญญาณรบกวนในสองบริเวณนี้อาจแตกต่างกันถึง 16 dB ระดับของสัญญาณรบกวนชนิดต่างๆ แสดงได้ดังรูปที่ 2 .14

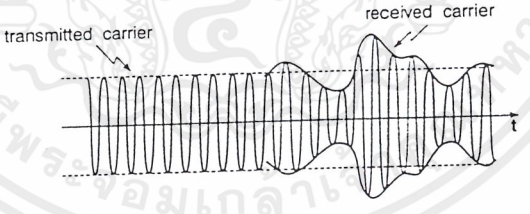


รูปที่ 2.14 แสดงระดับของสัญญาณรบกวนชนิดต่างๆ

โดยค่ากำลังงานของสัญญาณที่เครื่องรับ Fa แสดงในหน่วยของเดซิเบลเหนือสัญญาณรบกวนเนื่องจากอุณหภูมิ

2.4.2 สัญญาณรบกวนแบบ Multiplicative

คลื่นวิทยุเมื่อส่งออกไปในอากาศแล้ว สัญญาณที่ส่งออกไปจะถูกมอดูเลตทางแอมปลิจูดจากสัญญาณรบกวนแบบสุ่มดังรูปที่ 2.15 ซึ่งเกิดจากการสะท้อนของคลื่นหลายทิศทางที่มารวมกัน ทำให้เกิดปรากฏการณ์ที่เรียกว่า "การจางหายของสัญญาณ "หรือเฟดดิ้ง(Fading)



รูปที่ 2.15 แสดงการมอดูเลตทางแอมพลิจูดจากสัญญาณรบกวน

ซึ่งเราสามารถแบ่งชนิดของเฟดดิ้ง ออกได้เป็น 2 แบบคือ Long – term fading และ Short – term fading โดย Long – term fading นั้นจะมีลักษณะของฟังก์ชั่นความหนาแน่นของความน่าจะเป็น (PDF) เป็นแบบปกติ (Normal distribition) ส่วน Short – term fading นั้นสามารถแยกประเภทออกเป็นอีก 2 กรณี คือ

1) กรณีที่พิจารณาสัญญาณรวมที่เครื่องรับที่เกิดจากการสะท้อนของคลื่นสะท้อน (Indirect path) เท่านั้น ซึ่งจะมีค่า PDF ในลักษณะของการแจกแจงแบบเรย์ลี (Rayleigh distribution) [9]

2) กรณีที่พิจารณาสัญญาณรวมที่เครื่องรับที่เกิดจากทั้งคลื่นตรงและคลื่นสะท้อน ซึ่งจะมีค่า PDF ในลักษณะของการแจกแจกแบบไรเชียน (Riciean distribution) ซึ่งเขียนได้ดังนี้ [9]

$$p(r) = (r/\sigma^2)\exp\left\{\frac{-(r^2+a^2)}{2\sigma^2}\right\} I_o(ar/\sigma^2) \qquad \text{เมื่อ } r \geq 0 \qquad (2.28)$$

$$= 0 \qquad \text{สำหรับกรณีอื่น}$$

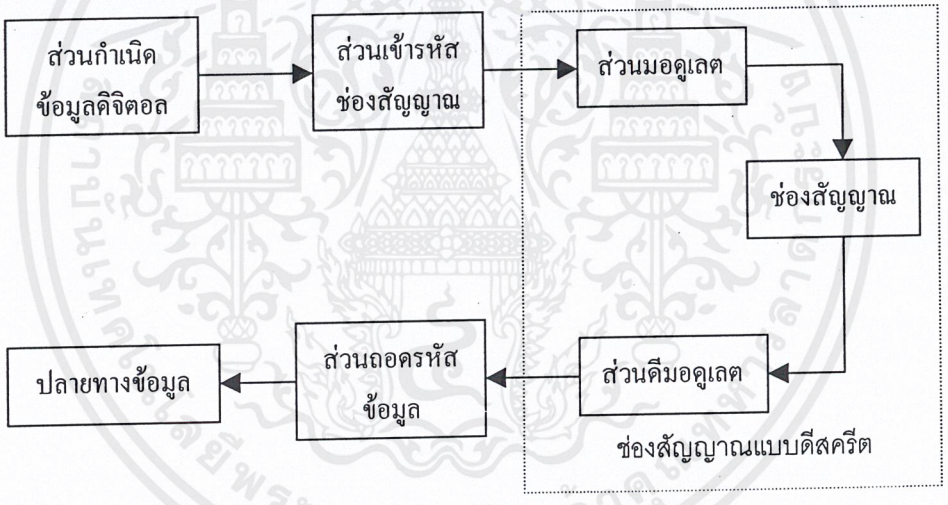
โดย a คือ ค่าแอมปลิจูดของคลื่นตรง

$I_o(.)$ คือค่าการแปลงฟังก์ชันเบสเสล (Bessel Function) ลำดับศูนย์

2.5 แบบจำลองโครงสร้างช่องสัญญาณ

2.5.1 ช่องสัญญาณแบบดิสครีตทางเวลา (The Discrete – Time Channel)

โดยปกติแล้วเราจะกำหนดแบบจำลองช่องสัญญาณ (Channel model) ให้รวมด้วยตัวมอดูเลเตอร์, ดีมอดูเลเตอร์ และส่วนประกอบทั้งหมดที่อยู่ระหว่างการส่งสัญญาณและตัวกลาง ซึ่งในที่นี้เราจะแสดงด้วยกรอบของเส้นประ ดังในรูปที่ 2.16 โดยถูกกำหนดให้ประกอบด้วยกลุ่มของอินพุตที่เข้าตัวมอดูเลเตอร์, กลุ่มของเอาท์พุทจากดีมอดูเลเตอร์ และค่าทางสถิติที่เกี่ยวข้องกับความเป็นไปได้ของเอาท์พุทต่อความเป็นไปได้ของแต่ละอินพุต



รูปที่ 2.16 แสดงส่วนประกอบของช่องสัญญาณดีสครีต

ทั้งหมดที่กล่าวมานี้ถูกเรียกว่า " Discrete – time channel model " หรือเรียกอย่างง่ายๆ ว่า " Discrete channel " และต่อจากนั้นเราจะพิจารณาถึงรูปแบบต่างๆ ของแบบจำลองโครงสร้างช่องสัญญาณที่ถูกเรียกว่า " Discrete memoryless channel " (DMC) ซึ่งถูกกำหนดโดยกลุ่มเอ็มแอรี (M – ary) ของคลื่นอินพุท $\{x_i\}$ และกลุ่มดิวแอรี(Q – ary) ของค่าเอาท์พุท $\{y_j\}$ และกลุ่มของค่า transition probabilities ซึ่งสามารถเขียนได้เป็น

$$P(y = y_j \mid x = x_i) = P(y_j \mid x_i) \qquad (2.29)$$

โดย $0 \leq i \leq M-1$ และ $0 \leq j \leq Q-1$

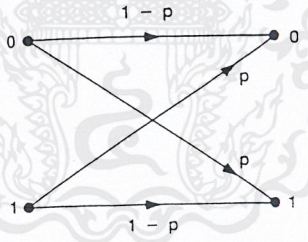
คำอธิบายของช่องสัญญาณที่เป็นแบบไร้ความจำ (Memoryless) พอจะกล่าวได้ว่า คือการที่ค่าเอาท์พุตที่ได้ทางด้านรับ ณ เวลาใดๆ นั้น ขึ้นอยู่กับค่าอินพุต ณ เวลานั้นเท่านั้น โดยไม่เกี่ยวเนื่องกับค่าใดๆก่อนหน้านั้น ซึ่งแบบจำลองช่องสัญญาณลักษณะนี้จะไม่สามารถใช้ประยุกต์ได้กับกรณีที่ช่องสัญญาณได้รับผลกระทบจาก Atmosphere impulse noise หรือ Intersymbol Interference

2.5.2 ช่องสัญญาณแบบ BSC (The Binary Symmetric Channel)

คราวนี้เราจะมาพิจารณาตัวอย่างสำคัญของแบบจำลอง DMC ให้ลองสมมุติพิจารณาว่าเราใช้การมอดูเลตแบบไบนารี่ที่ใช้การดีมอดูเลตแบบ Hard – Decision ทางด้านรับ ซึ่งเราให้ค่าอินพุต x ที่มอดูเลตมีค่าเป็น " 0 " หรือ " 1 " และค่าเอาท์พุตที่ทางดีมอดูเลตก็มีค่าเป็น " 0 " หรือ " 1 " เช่นเดียวกัน และหากกำหนดให้ x ถูกรับอย่างผิดพลาดด้วยค่าความน่าจะเป็น p หรือรับอย่างถูกต้องด้วยค่าความน่าจะเป็น 1- p ซึ่งสามารถเขียนได้เป็น

$$P(y=1 \mid x=0) = P(y=0 \mid x=1) = p \qquad (2.30)$$
$$P(y=0 \mid x=0) = P(y=1 \mid x=1) = 1-p$$

โดยที่สามารถเขียนเป็นไดอะแกรมได้ดังรูปที่ 2.17

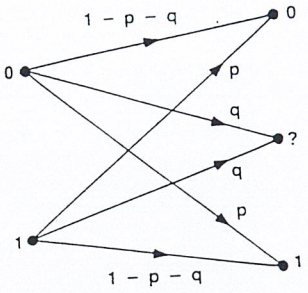


รูปที่ 2.17 แสดงแบบจำลองช่องสัญญาณแบบ BSC

แบบจำลองโครงสร้างแบบนี้เรียกว่า "Binary Symmetric Channel " (BSC)

2.5.3 ช่องสัญญาณแบบ BSEC (The Binary Symmetric Erasource Channel)

อีกกรณีพิเศษของแบบจำลอง DMC คือ ช่องสัญญาณแบบ Binary – Input , Ternary – Output ซึ่งถูกเรียกว่า "Binary Symmetric Erasure Channel " (BSEC) แสดงดังรูปที่ 2.18 ซึ่งมีลักษณะเป็น Symmetric Transition จากอินพุทค่าใดค่าหนึ่งไปยังเอาท์พุตที่แทนด้วยสัญลักษณ์ บอกถึงความคลุมเคลือ โดยเกิดจากการที่ค่าสัญญาณทางด้านรับของดีมอดูเลเตอร์มีค่าอ่อนมากจนไม่สามารถระบุความแน่ชัดได้แน่นอนและค่าบิทเอาท์พุตเหล่านี้ (?) จะถูกลบทิ้งเมื่อออกจากดีมอดูเลเตอร์ ซึ่งดีมอดูเลเตอร์ที่ให้ค่าเอาท์พุท 3 ค่านี้คือ ตัวอย่างแบบง่ายของการดีมอดูเลทแบบ Soft - Decision



รูปที่ 2.18 แสดงแบบจำลองช่องสัญญาณแบบ BSEC

2.5.4 ช่องสัญญาณแบบ AWGN (The Addition White Gaussion Noise Channel)

ชนิดของช่องสัญญาณ DMC ที่มีความสำคัญในทางปฏิบัติ คือ ช่องสัญญาณที่ให้ค่าสัญญาณเอาท์พุตที่เกิดจากสัญญาณอินพุตรวมกับสัญญาณรบกวนเกาส์แบบแถบกว้าง (Broadband Gaussian Noise) ซึ่งโดยทั่วไปเราจะเรียกว่า "White Gaussian Noise " โดยเป็นสัญญาณที่มีค่าเฉลี่ยเป็นศูนย์และมีการแจกแจงของค่ากำลังเท่ากันตลอดย่านความถี่ $-\infty \leq f \leq \infty$ ด้วยระดับ No /2 W/Hz หรือบนแกนความถี่ ข้างเดียว เท่ากับ No W/Hz

AWGN นั้นสามารถเขียนอธิบายในรูปแบบอย่างง่ายในเทอมของสัญญาณอินพุต x และสัญญาณเอาท์พุต y ดังนี้

$$y = x + n_G \tag{2.31}$$

โดย n_G คือค่าตัวแปรสุ่มแบบเกาส์ที่มีค่าเฉลี่ยเป็นศูนย์ และค่าความแปรปรวน σ^2

แบบจำลองโครงสร้างช่องสื่อสารแบบ AWGN นี้เป็นแบบจำลองที่ถูกต้องที่ใช้ในระบบสื่อสารต่างๆ เช่น ดาวเทียมและการสื่อสารในอวกาศ โดยค่าที่มีอิทธิพลเด่นในการจำกัดประสิทธิภาพของการสื่อสาร ก็คือ Thermal Noise หรือ Galactic Noise

2.6 ทฤษฎีข้อมูลและความจุของช่องสัญญาณ

2.6.1 การวัดปริมาณข้อมูล

ในทฤษฎีข้อมูล (Information Theory) นั้นวิธีธรรมดาที่สุดที่ใช้กำหนดปริมาณของข้อมูลที่บรรจุในสถานะของข่าวสาร (Message State) หรือในสัญลักษณ์ (Symbol) หรือในรหัส(Code) ต่างๆนั้นได้ใช้วิธีกำหนดโดยการคิดค่าล็อก (Logarithm) ของความเป็นไปได้ (Probability) หรือ ค่าล็อกของโอกาสของการเกิดสถานะของข่าวสารหรือสัญลักษณ์นั้นๆ กล่าวคือ ถ้าสัญลักษณ์ S มีโอกาสที่จะเกิดขึ้นเท่ากับ P แล้ว ปริมาณของข้อมูล (Information) I_i ที่มีอยู่ในสัญลักษณ์ S นั้น จะกำหนดได้โดย

$$I_i = - \mathrm{Log}_a (P_i) \tag{2.32}$$

โดยค่าหน่วยของ I ขึ้นอยู่กับการคิดฐานของล็อกที่ใช้ในสมการที่ 2.32

ถ้า a = e (e = 2.71828...) "I" ก็จะมีหน่วยเป็นนิท (nit : natural information unit)

ถ้า a = 2 "I" ก็จะมีหน่วยเป็นบิต (bit : binary information unit)

ถ้าสัญลักษณ์ที่ใช้อยู่ทั้งหมดมี n ตัว ปริมาณของข้อมูลที่มีเฉลี่ยอยู่ในสัญลักษณ์แต่ละตัว สามารถหาได้จาก

$$I_{av} = -\sum_{i=1}^{n} P_i I_i$$

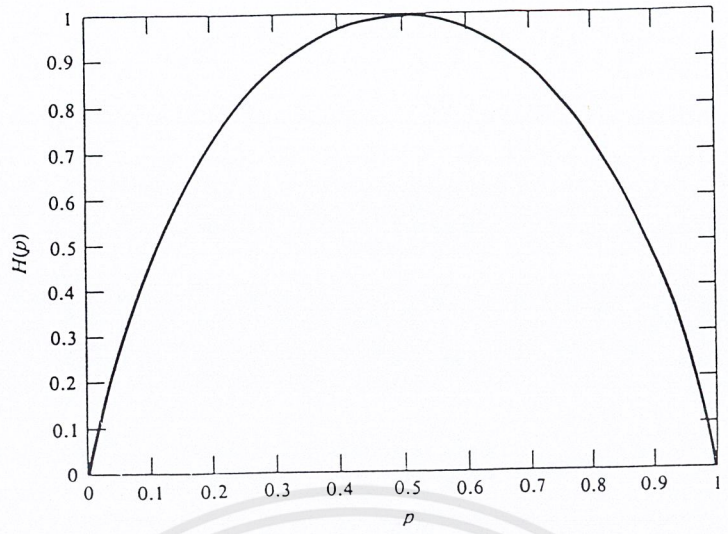
$$I_{av} = -\sum_{i=1}^{n} P_i \log_a (P_i) \qquad (2.33)$$

ค่าปริมาณข้อมูลเฉลี่ยมีชื่อเฉพาะเรียกว่า "เอนโทรปี" (Entropy) นิยมเขียนแทนด้วยตัว H และนอกจากนี้ค่าเอนโทรปียังสามารถใช้บอกถึงความไม่คงที่เฉลี่ยเบื้องต้น (Prior Uncertainty) ของข้อมูลในแต่ละสัญลักษณ์ (Symbol) ที่กำเนิดโดยแหล่งกำเนิด (Source) โดยแนวความคิดของความไม่คงที่นี้จะมีประโยชน์เมื่อเราพิจารณาถึงการวัดค่าของข้อมูลที่ใช้ส่งผ่านช่องสัญญาณ

สำหรับการสื่อสารระบบดิจิตอลระบบฐานสองนั้น สถานะของสัญญาณที่ใช้มีเพียง 2 สถานะเท่านั้น ที่เกิดขึ้นในช่วงเวลาที่กำหนดคือ สถานะ "1" และสถานะ "0" หากเราใช้สมการที่ (2.33) เพื่อพิจารณาค่าฟังก์ชันเอนโทรปีสำหรับข้อมูลไบนารี่แล้วจะได้ว่า

$$H = - p \log p - (1- p) \log (1- p) = H(p) \qquad (2.34)$$

โดยค่า p คือค่าความเป็นไปได้ของโอกาสที่จะเกิดขึ้นในแต่ละสัญลักษณ์ และเมื่อนำค่าฟังก์ชันในสมการที่ 2.34 มาวาดกราฟดังรูปที่ 2.19 เราจะพบว่าค่าเอนโทรปีจะมีค่ามากที่สุดเมื่อค่า p = 0.5 ดังนั้นค่าปริมาณข้อมูลของสัญญาณดิจิตอลที่เกิดขึ้น โดยเฉลี่ยในแต่ละช่วงเวลา (Time Slot) นั้นมีค่าเท่ากับ 1 บิทต่อสัญลักษณ์



รูปที่ 2.19 กราฟแสดงความสัมพันธ์ของค่าเอนโทรปีกับค่าความเป็นไปได้

ในกรณีของฟังก์ชันเอนโทรปีสำหรับสัญญาณ (Source) ที่มีค่าต่อเนื่อง ยกตัวอย่างเช่น ค่าแรงดันไฟฟ้า ซึ่งมีค่าฟังก์ชันความหนาแน่นของความน่าจะเป็น (PDF) เป็น p(x) เราสามารถคำนวณหาเอนโทรปีได้จาก

$$H = -\int_{-\infty}^{\infty} p(x)\log[p(x)]dx \tag{2.35}$$

ยกตัวอย่าง ทำการพิจารณาแหล่งกำเนิดสัญญาณข้อมูลต่อเนื่องที่ให้ค่าเอาท์พุต X และมี PDF เท่ากับ p(x) เราต้องการหาค่าฟังก์ชัน p(x) ที่ทำให้ฟังก์ชันเอนโทปีมีค่ามากที่สุด โดยกำหนดค่าความแปรปรวนของ p(x) ที่ σ^2 ซึ่งเท่ากับเป็นการกำหนดค่ากำลังงานเฉลี่ยของสัญญาณแรงดันไฟฟ้า x ต้องการ Maximize the integral

$$H = -\int_{-\infty}^{\infty} p(x)\log[p(x)]dx$$

โดยที่

$$\int_{-\infty}^{\infty} p(x)dx = 1 \tag{2.36}$$

และ

$$\int_{-\infty}^{\infty} x^2 p(x)dx = \sigma^2 \tag{2.37}$$

เราสามารถประยุกต์ใช้วิธีของ Lagrange Multipliers ซึ่งเราจะจัดรูป(Form) ของอินทิกรัลได้เป็น

$$\int_{-\infty}^{\infty} p(x)\{-\log[p(x)] + L + Mx^2\}dx \tag{2.38}$$

โดยวิธีการจะไม่ขอกล่าวรายละเอียดไว้ ณ ที่นี้ ซึ่งผลลัพธ์ที่ได้คือ

$$p_x(x) = \{\frac{1}{\sqrt{2\pi\sigma}}\} \exp[\frac{-x^2}{2\sigma^2}] \tag{2.39}$$

จะเห็นได้ว่าเป็น Zero – Mean Gaussian Density Function เพราะฉะนั้นเมื่อพิจารณาถึงค่ากำลังงานแล้ว แหล่งกำเนิดข้อมูลต่อเนื่อง (Continuous Information Source) จะมีค่าเอนโทรปีมากที่สุด คือ แหล่งกำเนิดแบบเกาส์ (Gaussian Source) โดยค่าเอนโทนปีของมันคือ

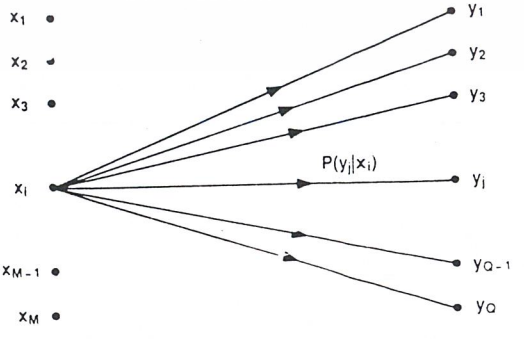
$$\begin{aligned} H &= \log\left[(\sqrt{2\pi e})\sigma\right] \\ &= \tfrac{1}{2}\log(2\pi e\sigma^2) \end{aligned} \tag{2.40}$$

โดยมีหน่วยเป็น bits/ sample เราจะย้อนมาพูดถึงสมการที่ (2.40) นี้กันภายหลัง

2.6.2 การส่งข้อมูลผ่านช่องสัญญาณ

จากที่เคยกล่าวไว้แล้วข้างต้นว่า เอนโทรปี H ของข้อมูลสามารถใช้วัดความไม่คงที่แน่นอนเบื้องต้น (Prior Uncertainty) เกี่ยวกับข้อมูลที่กำเนิดโดยแหล่งกำเนิดข้อมูลได้ ในการสื่อสารตามความจริงนั้นไม่ได้เป็นการสื่อสารแบบปราศจากความผิดพลาด (Error - Free) ดังนั้นจึงควรมีการพิจารณาถึงการใช้ฟังก์ชันที่เหมาะสมในการวัดค่าความไม่แน่นอนภายหลัง (Posteriori Uncertainty) ของข้อมูลหลังจากการรับข้อมูลผ่านช่องสื่อสารที่มีสัญญาณรบกวน (Noisy Channel)

โดยเราพิจารณาช่องสื่อสารดีสครีตแบบไร้ความจำ (Discrete Memoryless Channel) ซึ่งกำหนดคุณลักษณะโดยกลุ่มของอินพุต $x_1, x_2, \ldots, x_m$ มีค่า Priori Probablity $\{P(x_i)\}$ และกลุ่มของเอาท์พุต $y_1, y_2, \ldots, y_Q$ และกลุ่มของค่า Transition Probability $\{P(y_j | x_i)\}$ หรือความน่าจะเป็นของสัญญาณทางด้านรับ y_j เมื่อกำหนดสัญลักษณ์ x_i ที่ถูกส่งโดยรูปแบบของช่องสื่อสัญญาณนี้สามารถแสดงได้ดังรูปที่ 2.20



รูปที่ 2.20 แสดงแบบจำลองช่องสัญญาณดีสครีตแบบไร้ความจำ ที่มี M อินพุท Q เอาท์พุท

จากการใช้กฎของเบเยสสำหรับตัวแปรสุ่มเราสามารถเขียนค่า Posteriori Probability ของ x_i เมื่อกำหนดค่า y_j ที่ด้านรับ ได้ดังนี้

$$P(y = x_i | y_j) = \frac{P(x_i, y_i)}{P(y_j)} = \frac{P(y_j | x_i)P(x_i)}{P(y_j)} \quad (2.41)$$

จากที่ทราบว่าค่า Priori Uncertainty ของ x_i สามารถวัดได้จาก $-\log P(x_i)$ ดังนั้นเรากำหนดค่า Posteriori Uncertainty ของ x_i โดยค่าทางด้านรับเท่ากับ y_j เป็น $-\log P(x_i | y_j)$ โดยความแตกต่างของทั้งสองค่านี้ถูกเรียกว่า "Mutaul Information"

$$I(X;Y) = \log[P(y_j|x_i)/P(y_j)] \quad (2.42)$$

ตอนนี้เราคำนวณค่าอัตราทั้งหมดของการส่งข้อมูลผ่านช่องสัญญาณโดยการเฉลี่ยค่า $I(x_i ; y_j)$ ซึ่งเรียกว่า "Average Mutaul Information" กำหนดโดย

$$I(X;Y) = \sum_{i=1}^{M}\sum_{j=1}^{Q} P(x_i, y_j)\log\{P(y_j|x_i)/P(y_j)\} \quad (2.43)$$

ซึ่งมีหน่วยเป็น Bits/Symbol เมื่อให้ค่าฐานของล็อกเป็น 2 เพราะค่าความน่าจะเป็นของสัญลักษณ์เอาท์พุท $P(y_j)$ คือ

$$I(X;Y) = \sum_{i=1}^{M} P(x_i)P(y_j|x_i) \quad (2.44)$$

จะเห็นได้ว่า I(x;y) นั้นสามารถกำหนดโดยตรงจากค่าความน่าจะเป็นของการเกิด $P(x_i)$ และกลุ่มของค่า Transition Probability $P(y_j | x_i)$ ซึ่งค่า Average Mutaul Information นี้คืออัตราเฉลี่ยของการส่งข้อมูลผ่านช่องสื่อสาร โดยกำหนดการกระจายของ Source Symbols และ Channel - Transition Probabilities ถ้าเราให้ค่า Transition Probability เป็นการกำหนดช่องสัญญาณอัตราการส่งข้อมูลผ่านช่องสัญญาณจะขึ้นอยู่กับค่าความน่าจะเป็นของการเกิด Input Symbols ความจุ (Capacity) ของช่องสัญญาณถูกกำหนดที่ค่าสูงสุดของ I(x;y) นั่นคือ

$$C = \max_{P(X_i)} I(X;Y) \quad (2.45)$$

2.6.2.1 ความจุของช่องสัญญาณแบบ BSC

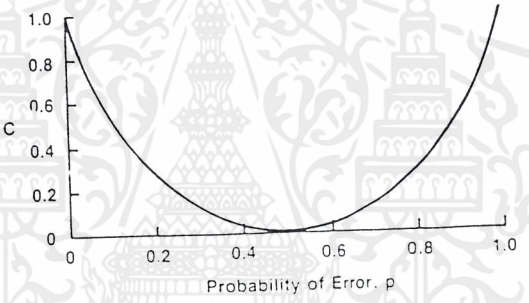
เราเคยกล่าวถึงช่องสัญญาณแบบ BSC แล้วในหัวข้อ 2.5.2 ถ้าเรากำหนดให้สัญลักษณ์อินพุต $x = 0,1$ และสัญลักษณ์เอาท์พุต $y = 0,1$ โดยค่า Transition Probability $P(y_j \mid x_i)$ เขียนได้ดังนี้

$$P\langle 0|0\rangle = P\langle 1|1\rangle = 1 - p$$
$$P\langle 1|0\rangle = P\langle 0|1 = p\rangle$$

อัตราการส่งข้อมูลของช่องสัญญาณแบบนี้จะค่ามากที่สุดเมื่อสัญลักษณ์อินพุตมี $P(0) = P(1) = 0.5$ ซึ่งจะได้ว่า

$$C = 1 + p\log p + (1-p)\log(1-p)$$

โดยเราแสดงค่าที่พล็อตกราฟได้ดังรูปที่ 2.21



รูปที่ 2.21 แสดงความจุของช่องสัญญาณแบบ BSC โดยหน่วยของ C คือ จำนวนบิทต่อสัญลักษณ์ที่ส่ง

สังเกตุว่าสำหรับค่า $p = 0$ ซึ่งตรงกับการส่งข้อมูลปราศจากความผิดพลาดนั้นค่า C มีค่าเท่ากับ 1 Bit/Transmitted Symbol ซึ่งก็คือค่าของเอนโทรปีของแหล่งกำเนิดที่ $P(0) = P(1) = 0.5$ และสำหรับค่า $p = 0.5$ นั้น แต่ละค่าของ y ที่รับได้นั้นมีค่าความน่าเป็นไปได้ที่เหมือนกันโดยไม่พิจารณาว่าค่าของ x ที่ส่งจะเป็นอะไร ดังนั้น $C = 0$ และช่องสัญญาณทำให้การส่งข้อมูลไปยังผู้รับล้มเหลวและเมื่อสังเกตดูจะเห็นว่ากราฟดังรูปที่ 2.21 นั้นสมมาตรที่ $p = 0.5$ ดังนั้นเราพิจารณาเพียงค่าของ p ในช่วง $0 \le p \le 0.5$ เรายังสามารถเขียนรูปแบบสมการที่ 2.43 ได้ในรูปแบบดังนี้

$$I(X;Y) = H(X) - H\langle X|Y\rangle \tag{2.46}$$

หรือ

$$I(X;Y) = H(Y) - H\langle Y|X\rangle$$

โดย

$$H(Y) = -\sum_{i=1}^{Q} P(y_i)\log P(y_j) \tag{2.47}$$

และ

$$I(X;Y) = -\sum_{i=1}^{M}\sum_{j=1}^{Q} P(x_i, y_j)\log\{P\langle y_j \,|\, x_i\rangle\} \tag{2.48}$$

ดังนั้น

$$\begin{aligned} I(X;Y) &= H(Y) - H\langle Y \,|\, X\rangle \\ &= H(Y) - \sum p(x)H(Y|X = x) \\ &= H(Y) - \sum p((x)H(p) \\ &\le 1 - H(p) \end{aligned}$$

ได้อสมการสุดท้ายเพราะ Y คือ ค่าตัวแปรสุ่มไบนารี่ อสมการจะมีค่าเท่ากับเมื่ออินพุทมีการกระจายแบบยูนิฟอร์ม ดังนั้นค่าความจุข้อมูลของช่องสัญญาณแบบ BSC ด้วยพารามิเตอร์ p คือ

$$C = 1 - H(p) \quad \text{bits}$$

2.6.2.2 ความจุของช่องสัญญาณแบบ Continuous AWGN

จากหัวข้อที่รู้จักกันดี Sampling Theory ซึ่งมีอัตราไนควิสท์ 2W โดย W คือแบนวิดท์ของช่องสัญญาณ ถ้ากำหนดให้คลื่นที่ส่งมีรูปแบบคลื่นเป็น x(t) และ Additive White Gaussian Noise เป็น n(t) ดังนั้นสัญญาณที่เครื่องรับ คือ y(t) = x(t) + n(t) และเราสามารถกำหนดค่ากำลังงานเฉลี่ยของสัญญาณที่รับได้หรือตัวอย่างสัญญาณรบกวน เป็น $\overline{x^2} = S$ และ $\overline{n^2} = N$ ตามลำดับ

หากเราสมมุติว่าค่ากำลังงานเฉลี่ยเหล่านี้ถูกทำให้คงที่ จากหัวข้อที่ 2.6.2 เราจะทำการดัดแปลงโดยแทนค่า $P(x_i)$ และ $P(y_j)$ ด้วย PDF p(x) และ p(y) ตามลำดับ และแทน $P(y_j \mid x_i)$ ด้วย p(x|y) โดยค่า p(x|y) นั้นจะขึ้นอยู่กับ y และ x โดยค่าผลต่าง y – x และสำหรับกรณีของ Gaussian Noise เราจะได้ว่า

$$p\langle y|x\rangle = \frac{1}{(\sqrt{2\pi N})\exp[-(y-x)^2/2N]}$$

โดยค่า N คือค่ากำลังเฉลี่ยหรือความแปรปรวนของตัวอย่างสัญญาณรบกวน

สัญญาณรบกวนแบบบวก (Addition) เป็นคุณสมบัติของช่องสัญญาณต่อเนื่อง (Continuous-Channel) ซึ่งเกี่ยวข้องสอดคล้องโดยตรงกับคุณสมบัติของอินพุทที่เป็นยูนิฟอร์ม ซึ่งถูกกำหนดให้

$$C = \max_{P(X_i)}[H(Y)] - H\langle Y \,|\, X\rangle \tag{2.49}$$

โดยค่าเอนโทรปีเอาท์พุต H(y) คิดจากลำดับของตัวอย่างสัญญาณที่รับได้ y(t) และค่าของเอนโทรปีอย่างมีเงื่อนไข H(Y|X) ขึ้นอยู่กับการกระจายทางสถิติของตัวอย่างสัญญาณรบกวนเท่านั้น นั่นคือ ตัวอย่าง (Sample) ของ n(t) ดังนั้นเราสามารถเขียนสมการที่ (2.49) ใหม่ได้ดังนี้

$$C = \max_{P(X_i)}[H(y)] - H(n) \tag{2.50}$$

ดังนั้นตอนนี้เราสามารถกำหนดความจุของ Continuous – Waveform AWGN Channel ได้เท่ากับค่าสูงสุดของเอนโทรปีของสัญญาณที่รับได้ลบกับค่าเอนโทรปีของสัญญาณรบกวนค่าสูงสุดของสมการที่ (2.50) สามารถทำได้โดยเลือกรูปแบบคลื่นที่ส่ง x(t) ให้เหมาะสม

เริ่มแรกเราพิจารณา H(n) จากตัวอย่างสมการที่ (2.40) ซึ่งค่าเอนโทรปีของตัวอย่างจากการกระจายแบบเกาส์ คือ $(1/2)\log(2\pi e\sigma^2)$ โดย σ^2 คือค่าความแปรปรวน (variance) ของการกระจาย ดังนั้นเรากำหนดให้เอนโทรปีของตัวอย่างของสัญญาณรบกวนแบบเกาส์คือ $(1/2)\log(2\pi eN)$ ซึ่ง N คือค่าความแปรปรวน หรือค่ากำลังงานเฉลี่ยของตัวอย่างสัญญาณรบกวน

เอนโทรปีของลำดับของ Noise Samples ที่ 2WT ในระหว่างช่วงเวลาเท่ากับ T วินาที คือ

$$H(n) = WT\log(2\pi eN)$$

ตอนนี้เราพิจารณา H(y) จากสมการที่ (2.40) เรากำหนดให้ตัวแปรสุ่มมีค่าความแปรปรวนที่คงที่และค่าเอนโทรปีของตัวแปรสุ่มจะมีค่ามากที่สุดเมื่อให้ PDF เป็นฟังก์ชันแบบเกาส์ ดังนั้นถ้าเรากำหนดให้ค่าสัญญาณ y(t) มีค่ากำลังงานเฉลี่ย S + N จะได้ว่าเอนโทรปีของสัญญาณที่รับได้ H(y) จะมีค่ามากที่สุดเมื่อให้ y(t) มีรูปแบบทางสถิติของ Gaussian Noise ซึ่งแต่ละ Sample มีค่าความแปรปรวน S + N จะได้ลำดับของ 2WT Samples ดังนี้

$$\max_{P(X_i)} H(y) = WT\log[2\pi e(S+N)]$$

ดังนั้นค่าสูงสุดของอัตราการส่งข้อมูล คือ

$$I_{\max} = WT\log\{\frac{(S+N)}{N}\}$$

ซึ่งถูกวัดในหน่วย Bits Transmitted Per T Seconds นั่นคือ

$$\mathbf{C = W\log\left(S+N/N\right) = W\log\left(1+S/N\right)} \quad \text{บิท / วินาที} \tag{2.51}$$

นี่คือสูตรความจุของ Shannon สำหรับ Band- Limited Continuous AWGN Channel

2.6.3 ทฤษฎีของการเข้ารหัสช่องสัญญาณ

ทฤษฎีของการเข้ารหัสช่องสัญญาณนั้นกล่าวว่าทุกช่องสื่อสาร C และที่ค่าอัตราการส่งข้อมูลใดๆ $R < C$ แล้วจะมีรหัสความยาวบล็อก (Block Length) n และอัตรา R ซึ่งมีความน่าจะเป็นของการถอดรหัสผิดพลาด P(E) ถูกกำหนดโดย

$$P(E) \le 2^{-nE_b(R)} \tag{2.52}$$

โดยค่าของ $E_b(R)$ คือค่าบวกของฟังก์ชัน R สำหรับ $R < C$ และเป็นการพิจารณาโดยคุณลักษณะของช่องสัญญาณเท่านั้น สำหรับการนำสมการที่ 2.52 ไปใช้นั้น คือ ที่อัตราการส่งข้อมูลใดๆ น้อยกว่า C แล้ว ค่าความน่าจะเป็นของความผิดพลาดสามารถทำให้น้อยลงตามต้องการได้โดยการเพิ่มความยาวรหัส Block Length n ในขณะที่ยังคงรักษาอัตราเข้ารหัส (Code Rate) ให้คงที่ โดยเงื่อนไขดังกล่าวสามารถเขียนได้เช่นเดียวกันสำหรับการเข้ารหัสการประสาน (Convolutional Codes) ซึ่ง n ถูกแทนด้วย k (Code Constraint Length)

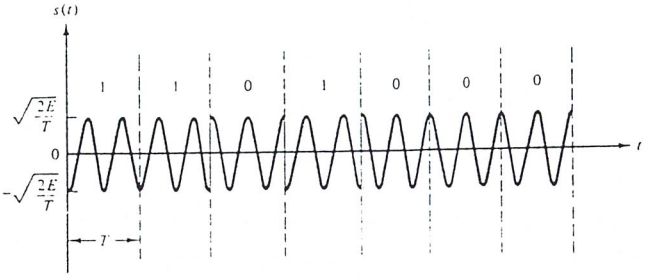
2.6.4 การมอดูเลทและดีมอดูเลท

ตัวมอดูเลเตอร์ในระบบสื่อสารนั้นจะต้องเลือกรูปแบบคลื่น (Waveform) ของช่วงเวลา T วินาที ซึ่งเหมาะสมกับการส่งข้อมูล สำหรับสัญลักษณ์เอาท์พุทแต่ละตัวที่ออกมาจากตัวเข้ารหัส (Encoder) ในกรณีของรหัสไบนารี่นั้น ตัวมอดูเลเตอร์จะต้องกำเนิดคลื่นหนึ่งใน 2 รูปแบบคือ $S_o(t)$ สำหรับรหัส “0” หรือ $S_1(t)$ สำหรับรหัส “1” ออกมา ทางเลือกหนึ่งของรูปแบบสัญญาณคือ

$$S_o(t) = [\sqrt{\frac{2E}{T}}]\sin(2\pi f_o t + \pi/2) \quad , \quad 0 \le t \le T$$

$$S_1(t) = [\sqrt{\frac{2E}{T}}]\sin(2\pi f_o t - \pi/2) \quad , \quad 0 \le t \le T \tag{2.53}$$

โดย $f_o = 1/T$ และ E คือพลังงานของแต่ละสัญญาณ รูปแบบนี้เรียกว่า “ Binary – Phase – Shift – Keyed (BPSK) Modulator ” เพราะคลื่นสัญญาณรูปไซน์ที่ถูกส่งออกไปนั้นจะมีค่าเฟสอยู่ 2 ค่าคือ $+\pi/2$ และ $-\pi/2$ ขึ้นอยู่กับค่ารหัสเอาท์พุตที่ได้จากตัวเข้ารหัสโดยรูปแบบคลื่นดังกล่าวที่สัมพันธ์กับรหัส (Code Word) $\mathbf{v} = (1\,1\,0\,1\,0\,0\,0)$ แสดงดังรูปที่ 2.22



รูปที่ 2.22 แสดงรูปสัญญาณที่ถูกมอดูเลตแบบ BPSK ซึ่งสอดคล้องกับคำรหัส
v = (1 1 0 1 0 0 0)

รูปแบบการรบกวนของสัญญาณรบกวนทั่วไปในระบบสื่อสารคือ Addition White Gaussian Noise (AWGN) ตัวดีมอดูเลเตอร์จะต้องให้สัญญาณเอาท์พุทที่สอดคล้องกับสัญญาณที่รับได้ในแต่ละช่วงเวลา T วินาที สำหรับการมอดูเลชั่นแบบ BPSK ด้วย Coherent Detection ค่าเอาท์พุตตัวอย่าง (Sample Output) จะเป็นเลขจำนวนจริง

$$S = \int_0^T r(t)[\sqrt{\frac{2E}{T}}]\sin(2\pi f_o t + \pi/2)dt \qquad (2.54)$$

โดย r(t) คือสัญญาณที่รับได้

สำหรับลำดับเอาท์พุตที่ไม่มีการจัดระดับ (Unquantized) ของดีมอดูเลเตอร์ สามารถผ่านได้โดยตรงเพื่อการถอดรหัสต่อไป โดยในกรณีนี้ตัวถอดรหัสช่องสัญญาณ (Channel Decoder) จะต้องสามารถจัดการกับสัญญาณอนาล็อกได้ โดยมันจะต้องเป็น Analog Decoder แต่โดยทั่วไปแล้วในการถอดรหัสนั้นจะต้องทำการจัดระดับค่าสัญญาณต่อเนื่อง S ออกเป็นจำนวนที่แน่นอน Q โดยในกรณีนี้ตัวถอดรหัสช่องสัญญาณจะได้รับสัญญาณดีสครีตอินพุต ดังนั้นมันจึงต้องเป็น Digital Decoder

เมื่อเราใช้การมอดูเลทแบบ BPSK บนช่องสัญญาณแบบ AWGN และใช้ Optimum Coherent Detection เราสามารถหาค่าความน่าจะเป็นของความผิดพลาดของการส่งข้อมูล โดยไม่มีการเข้ารหัสช่องสัญญาณได้จาก

$$p = 1/2 erfc(\sqrt{\frac{E_b}{N_o}}) \qquad (2.55)$$

โดย erfc(x) คือ Complementary Error Function ซึ่งกำหนดโดย

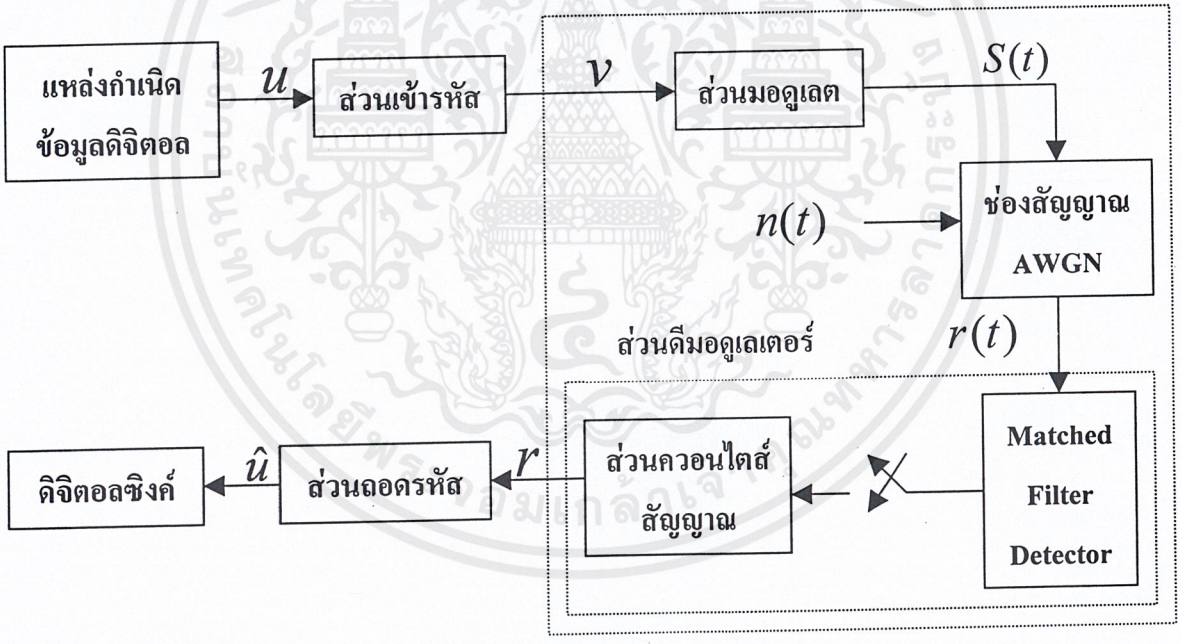
$$erfc(x) = (\frac{1}{\sqrt{\pi}})\int_x^\infty e^{-t^2}dt$$

และค่า E_b/N_o คือค่า Signal – To – Noise Ratio (SNR) ต่อบิทและสำหรับกรณีช่องสัญญาณแบบ BSC นั้นเราสามารถหาค่า Transition Probability (p) ได้จาก

$$p = erfc(\sqrt{\frac{2E_b}{N_o}}) \qquad (2.56)$$

เมื่อใช้การเข้ารหัสไบนารี่ ตัวมอดูเลเตอร์นั้นมีค่าอินพุตเพียงแค่ 2 ค่าคือ "0" หรือ "1" (M = 2) และเมื่อตัวมอดูเลเตอร์ก็มีค่าเอาท์พุตที่จัดระดับเพียงแค่ 2 ระดับ (Q = 2) ตัวถอดรหัสก็จะมีค่าอินพุตเพียงแค่ 2 ค่าเท่านั้น กรณีแบบนี้เราเรียกตัวมอดูเลเตอร์ว่าทำ Hard Decisions แต่ถ้าหากค่า Q > 2 หรือค่าเอาท์พุตจากดีมอดูเลเตอร์ไม่มีการจัดระดับเลย เราเรียกว่าทำ Soft Decisions โดยในกรณีนี้ตัวถอดรหัสจะต้องยอมรับค่าอินพุทที่มีหลายระดับ ซึ่งการทำ Soft Decisions นั้นจะให้ค่าประสิทธิภาพที่ดีกว่าการทำ Hard Decisions

2.6.5 MLD (Maximum Likelihood Decoding)



รูปที่ 2.23 ระบบของการ ส่ง-รับ ข้อมูลผ่านช่องสัญญาณเกาส์ขาวแบบบวก

จากรูปที่ 2.23 ในระบบของบล็อกโค๊ดนั้นค่าเอาท์พุตจากแหล่งกำเนิดข้อมูล u จะแทนข้อความ k- bit และค่าเอาท์พุตจากตัวเข้ารหัส v จะแทน n – Symbol Code Word ค่าเอาท์พุตจากดีมอดูเลเตอร์ r จะถูกส่งไปยังตัวถอดรหัสเพื่อประเมินค่าเอาท์พุต $\hat{u}$ ซึ่งแทนการประเมินค่า k – bit สำหรับระบบคอนโว ลูชั่นโค๊ดนั้น u จะแทนลำดับของข้อมูล kL บิท และ v จะแทนคำรหัสซึ่งประกอบด้วย

N= nL + nm = n(L+m) สัญลักษณ์ โดย kL คือความยาวของข้อมูล และ N คือความยาวของคำรหัส ส่วน nm นั้นจะถูกเพิ่มเข้าไป หลังจากชุดสุดท้ายของข้อมูลได้เข้าไปในตัวเข้ารหัส โดยค่า m นี้คือ หน่วยของหน่วยความจำในตัวเข้ารหัส ซึ่งเราจะกล่าวถึงภายหลัง [1]

ตัวถอดรหัสนั้นจะต้องทำการประมาณค่า $\hat{u}$ ของลำดับสัญญาณ u ซึ่งอาศัยลำดับสัญญาณ r ที่รับได้ ตัวถอดรหัสสามารถให้ค่าประเมิน $\hat{v}$ ของคำรหัส v โดยที่ $\hat{u} = u$ ถ้า $\hat{v} = v$ โดยกฎของการถอดรหัส คือ กลวิธีที่จะเลือกการประเมินคำรหัส $\hat{v}$ สำหรับในแต่ละลำดับสัญญาณ r ที่รับได้ ถ้าคำรหัส v ถูกส่ง ความผิดพลาดของการถอดรหัสจะเกิดขึ้นเมื่อ $\hat{v} \neq v$ โดยกำหนด r คือสัญญาณที่รับได้โดยค่าความน่าจะเป็นอย่างมีเงื่อนไขของความผิดพลาดของตัวถอดรหัส คือ

$$P\langle E|r\rangle = P\langle \hat{v} \neq v|r\rangle \tag{2.57}$$

และค่า Error Probability of The Decoder ถูกกำหนดโดย

$$P(E) = \sum_r P\langle E|r\rangle P(r) \tag{2.58}$$

ในการที่จะทำให้ค่า $P\langle \hat{v} \neq v|r\rangle$ มีค่าน้อยที่สุด ก็เสมือนว่าต้องทำให้ค่า $P\langle \hat{v} = v|r\rangle$ มีค่ามากที่สุด

$$P\langle v|r\rangle = \frac{P\langle r|v\rangle P(v)}{P(r)} \tag{2.59}$$

นั่นคือ $\hat{v}$ จะถูกเลือกโดยความที่มีลักษณะใกล้เคียงกับคำรหัสที่รับได้ (r) มากที่สุด ถ้าลำดับข้อมูลทั้งหมดและคำรหัสทั้งหมดมีความเป็นไปได้ที่เกิดขึ้นเท่าๆ กัน ซึ่ง P(v) จะมีค่าเหมือนกันสำหรับทุกๆค่า v ดังนั้นในการที่จะให้สมการ (2.59) มีค่ามากที่สุด จะเหมือนกับการทำให้ P(r|v) มีค่ามากที่สุด สำหรับ DMC เราจะได้ว่า

$$P\langle r|v\rangle = \prod_i P\langle r_i|v_i\rangle \tag{2.60}$$

ตัวถอดรหัสจะทำการเลือกค่าประเมินของมันเพื่อที่จะทำให้สมการ (2.45) มีค่ามากที่สุด ซึ่งเรียกว่า "Maximum Likelihood Decoding" (MLD) หากเรา take log เข้าไปทั้งสองด้านของสมการ (2.60) จะได้ว่า

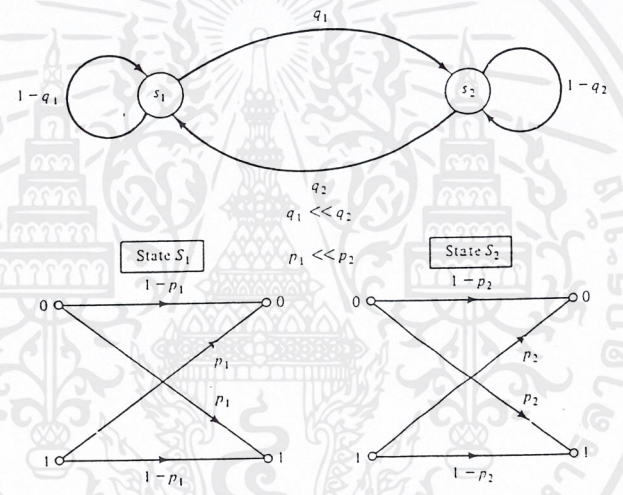
$$\log P\langle r|v\rangle = \sum_i \log P\langle r_i|v_i\rangle \tag{2.61}$$

โดย MLD สำหรับ DMC นั้นจะทำการเลือก $\hat{v}$ ในฐานะคำรหัส v ซึ่งจะทำให้ค่าผลบวกใน (2.61) มีค่ามากที่สุด

2.6.6 ชนิดของความผิดพลาดในช่องสัญญาณ

สำหรับช่องสัญญาณแบบไร้ความจำ (Memoryless Channels) นั้น สัญญาณรบกวนมีผลกระทบต่อสัญญาณที่ส่งอย่างอิสระ ยกตัวอย่างเช่น พิจารณา BSC ซึ่ง Transition Diagram สามารถแสดงได้ดังรูปที่ 2.17 แต่ละสัญญาณที่ส่งนั้นมีความน่าจะเป็น p ในการรับอย่างไม่ถูกต้องและมีความน่าจะเป็น 1- p ในการรับที่ถูกต้อง ซึ่งเป็นอิสระต่อสัญญาณที่ส่งอื่นๆ เพราะฉะนั้นความผิดพลาดในการส่งสัญญาณจะเกิดขึ้นอย่างสุ่มที่สัญญาณรับและช่องสัญญาณไร้ความจำแบบนี้เรียกว่า "Random – Error Channel "

สำหรับช่องสัญญาณที่มีความจำสัญญาณรบกวนนั้นไม่ได้เป็นอิสระจากการส่งสัญญาณ โดยรูปแบบของช่องสัญญาณที่มีความจำนี้สามารถแสดงได้ดังรูปที่ 2.24



รูปที่ 2.24 แสดงแบบจำลองช่องสัญญาณที่มีความจำ

ช่องสัญญาณนี้ประกอบด้วย 2 สเตท "Good State" ซึ่งความผิดพลาดในการส่งสัญญาณเกิดขึ้นไม่บ่อย , $p_1 \approx 0$ และ "Bad State" ซึ่งความผิดพลาดในการส่งสัญญาณนั้นเกิดขึ้นสูง , $p_2 \approx 0.5$ ช่องสัญญาณโดยส่วนมากจะอยู่ใน Good State แต่ในบางโอกาสมันก็อาจเลื่อนไปอยู่ใน Bad State เนื่องจากการเปลี่ยนแปลงคุณสมบัติของช่องสัญญาณในการส่งสัญญาณ ซึ่งทำให้ผลที่ตามมา คือ กลุ่ม (Clusters หรือ Bursts) ของความผิดพลาดในการส่งสัญญาณเนื่องจากความน่าจะเป็นที่สูงของการเปลี่ยนสเตทไปสู่ Bad State และช่องสัญญาณที่มีความจำแบบนี้เรียกว่า "Burst – Error Channels " และในบางช่องสัญญาณที่ประกอบไปด้วย 2 ลักษณะคือ Random และ Burst Errors เราเรียกว่า " Compound Channels "

2.7 ทฤษฎีเบื้องต้นสำหรับการเข้ารหัสช่องสัญญาณ

2.7.1 น้ำหนักแฮมมิ่ง (Hamming Weight) และระยะแฮมมิ่ง (Hamming Distance)

น้ำหนักแฮมมิ่ง(Hamming Weight) : ค่าน้ำหนักแฮมมิ่งหรือน้ำหนักของเวกเตอร์ c ถูกกำหนดโดยจำนวนที่ไม่เป็นศูนย์ (Non - Zero) ของค่าในเวกเตอร์ ซึ่งจำนวนนี้จะมีช่วงจากค่าน้อยที่สุดคือศูนย์ไปจนถึงความยาว n ของเวกเตอร์ c

$$W_t(c) = \sum_{j=0}^{n=1} W_t(c_j) \qquad \text{โดย} \quad W_t(c_j) = 0 \ , \ c_j = 0$$

$$= 1 \ , \ c_j \neq 0 \tag{2.62}$$

ระยะแฮมมิ่ง(Hamming Distance) : ค่าระยะแฮมมิ่งของระหว่าง 2 เวกเตอร์ a และ c ถูกกำหนดโดย จำนวนของค่าในเวกเตอร์ a และ c ที่ต่างกัน

$$dist(a,c) = \sum_{j=0}^{n=1} W_t(a_j + c_j) \qquad \text{โดย} \quad W_t(c_j) = 0, c_j = 0$$

$$= 1, c_j \neq 0 \tag{2.63}$$

dist (a,c) = wt (a +c) เป็นการบวกโดยใช้การดำเนินการแบบ modulo 2

2.7.2 ระยะความแตกต่างต่ำสุด (Minimum Distance) และการแก้ไขความผิดพลาด (Error Correcting)

ค่าระยะความแตกต่างต่ำสุด (d) ของรหัส c คือค่าระยะต่ำสุดระหว่าง 2 รหัสที่แตกต่างกัน

$$d = \min\{dist(a,c)\} \qquad \text{โดย} \quad a,c \in C \quad \text{และ} \quad a \neq c \tag{2.64}$$

สำหรับลิเนียร์โค๊ดนั้นค่าระยะความแตกต่างต่ำสุด (Minimum Distance) จะมีค่าเท่ากับค่าน้ำหนักต่ำสุด (Minimum weigth)

$$d = \min\{W_t(a+c)\} \qquad \text{โดย} \quad a,c \in C \quad \text{และ} \quad a \neq c$$

$$= \min\{W_t(c)\} \qquad \text{โดย} \quad c \neq c \quad \text{และ} \quad c \neq 0 \tag{2.65}$$

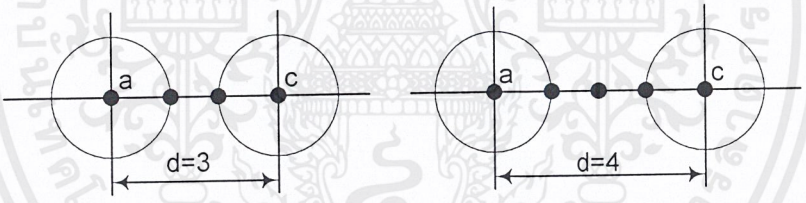
ซึ่งคุณสมบัติของสมการที่ (2.65) นั้นมีความสำคัญมากในการออกแบบรหัสต่างๆ ค่าระยะความแตกต่างต่ำสุด (Minimum Distance) ระหว่างคำรหัสนั้นจะสามารถบอกถึงความสามารถของรหัสในการตรวจหาความผิดและการแก้ไขความผิดพลาดได้ ในการคำนวณหาค่าระยะความแตกต่างต่ำสุดโดยใช้ค่าน้ำหนักต่ำสุดนั้นสามารถทำได้ง่ายกว่าการคำนวณจากค่าระยะความแตกต่างต่ำสุดโดยตรง และจากความรู้เรื่องค่าระยะความแตกต่างต่ำสุดนี้ ทำให้เกิดคำถามขึ้นว่าจำนวนความผิดพลาดเท่าไร

ที่สามารถจะแก้ไขได้ ลองพิจารณาจากรูปที่ 2.25 จากรูปทางซ้ายมือนั้น เราสมมุติว่ารหัสมีค่าระยะความแตกต่างต่ำสุด (Minimum Distance) d = 3 ซึ่งเท่ากับ Hamming Distance ระหว่างเวกเตอร์ a กับ c ถ้าเราล้อมรอบแต่ละคำรหัสด้วยทรงกลมของระยะแฮมมิ่ง 1 เราสังเกตได้ว่าทรงกลมนั้นจะไม่เหลื่อมหรือทับกัน ดังนั้นถ้าแต่ละเวกเตอร์มีค่าระยะแฮมมิ่ง $\leq$ 1 จากคำรหัสใดๆแล้ว เราสามารถที่จะได้คำรหัสโดยไม่ซ้ำกัน สำหรับรูปทางขวามือนั้นค่าระยะความแตกต่างต่ำสุดเป็น d = 4 ถ้าเวกเตอร์ที่รับได้มีค่าระยะเท่ากับ 2 แล้ว จากรูปจะสังเกตเห็นว่าไม่สามารถจะทำให้คำรหัสที่ไม่ซ้ำกันดังนั้น เวกเตอร์ a กับ c สามารถทำการแก้ไขเวกเตอร์ได้ด้วยค่าระยะแฮมมิ่ง $\leq$ 1 เท่านั้น สรุปคือ สิ่งที่ค่าระยะแฮมมิ่งระหว่างคำรหัสมีค่ามากเท่าไหร่ก็จะยิ่งเป็นการยากที่จะเปลี่ยนคำรหัสหนึ่งไปสู่อีกคำรหัสหนึ่ง ดังนั้นหากรหัสที่มีค่าระยะความแตกต่างต่ำสุด (Minimum Distance) มากก็จะเป็นการดี และเราจะได้ความสัมพันธ์คือถ้าหากคำรหัสมีค่าระยะความแตกต่างต่ำสุด d_{min} และความสามารถในการแก้ไขความผิดพลาดได้มากที่สุด t_c ตำแหน่งหรือความสามารถสูงสุดในการตรวจหาความผิด t_D ตำแหน่ง ดังนี้

$$d_{min} \geq 2\,t_c + 1 \tag{2.66}$$

และ

$$d_{min} \geq t_D + 1 \tag{2.67}$$



รูปที่ 2.25 แสดงระยะแฮมมิ่ง

2.8 รหัสลิเนียร์บล็อค (Linear Block Codes)

เป็นรหัสที่ประกอบด้วยเวกเตอร์ที่มีขนาดความยาวจำกัดซึ่งจะถูกเรียกว่า " คำรหัส " (Code Words) ความยาวของคำรหัสคือ จำนวนตัวประกอบทั้งหมดที่มีอยู่ในเวกเตอร์ ซึ่งจะมีค่าเท่ากับ n โดยตัวประกอบแต่ละตัวจะเรียกว่า " บิต " ซึ่งคำรหัสนี้จะเลือกจากจำนวนตัวเลขทั้งหมด q หมายเลข ถ้าหากตัวประกอบในเวกเตอร์ประกอบด้วยค่าเพียง 2 ค่า คือ 1 และ 0 รหัสนี้เรียกว่า "รหัสไบนารี่ " หาก $q > 2$ รหัสนี้จะถูกเรียกว่า "นอนไบนารี่ "

ในการเข้ารหัสของข้อมูลซึ่งมีขนาดความยาวเท่ากับ k บิต หลังจากผ่านการเข้ารหัสจะได้ความยาวของคำรหัสที่มีค่าเท่ากับ n บิต ซึ่งขนาด $n-k$ บิต เป็นบิตที่เพิ่มรวมเข้าไปกับข้อมูลจะถูกเรียกว่า "พาริตีบิท "หรือ "บิตตรวจสอบ " โดยรหัสที่ได้จะถูกเรียกว่าเป็น รหัส (n,k) ซึ่งอัตราส่วนของจำนวนบิตของข้อมูลกับความยาวของคำรหัส $k/n = Rc$ เรียกว่า "อัตราการเข้ารหัส "(Code Rate)

โดยทั่วไป คำรหัสของรหัสที่ใช้บิตตรวจสอบสถานะคู่ (Parity Check Bit) จะสามารถเขียนได้ในรูปแบบต่อไปนี้ คือ

$$a_1,a_2,a_3,\ldots,a_k,c_1,c_2,c_3,\ldots,c_r \tag{2.68}$$

โดยในที่นี้ a_i คือบิตที่ i ของคำรหัสข่าวสาร (Message Code Word) และ c_j คือบิตตรวจสถานะคู่ตัวที่ j บิตตรวจสอบ(check bit)จะถูกเลือกจากสมการเชิงเส้นจำนวน r สมการ โดยที่ $r = n - k$ ซึ่งจะได้สมการในรูปแบบต่อไปนี้ คือ

$$\begin{aligned}
0 &= h_{11}a_1 \oplus h_{12}a_2 \oplus \ldots\ldots\ldots\ldots \oplus h_{1k}a_k \oplus c_1 \\
0 &= h_{21}a_1 \oplus h_{22}a_2 \oplus \ldots\ldots\ldots\ldots \oplus h_{2k}a_k \oplus c_2 \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
0 &= h_{r1}a_1 \oplus h_{r2}a_2 \oplus \ldots\ldots\ldots\ldots \oplus h_{rk}a_k \oplus c_r
\end{aligned} \tag{2.69}$$

ซึ่งสมการ (2.69) นี้ สามารถเขียนในรูปของเมตริกซ์ ได้คือ

$$[H][T] = 0 \tag{2.70}$$

โดย [T] คือคอลัมเมตริกซ์ขนาด n x 1 ที่ใช้แทนคำรหัส

$$T = \begin{bmatrix} a1 \\ a2 \\ . \\ . \\ . \\ ak \\ c1 \\ . \\ . \\ . \\ cr \end{bmatrix} \tag{2.71}$$

และ [H] คือ เมตริกซ์ตรวจความเป็นคู่ขนาด r x n

$$[H] = \begin{bmatrix} h_{11}\ h_{12} \ldots h_{1k}\ 1\ 0 \ldots 0 \\ h_{12}\ h_{22} \ldots h_{2k}\ 0\ 1 \ldots 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ h_{r1}\ h_{r2} \ldots h_{rk}\ 0\ 0 \ldots 1 \end{bmatrix} \quad (2.72)$$

เมื่อให้เมตริกซ์ของคำรหัสที่รับได้ คือ [R] ซึ่งอาจจะเป็นค่าเดียวกับ [T] ได้ เมื่อการรับไม่มีความผิดพลาด เราสามารถทำการตรวจรหัสที่รับมานั้นว่าผิดพลาดได้หรือไม่ โดยการตรวจค่าผลคูณ [H][R] ถ้าผลคูณนั้นเป็น " 0 " แสดงว่า [R] คือคำรหัสที่ต้องการ แต่ถ้า $[H][R] \neq 0$ ก็จะแสดงว่าเกิดความผิดพลาดในคำรหัส [R] ที่รับได้นั้น

โดยกำหนดให้ $$[S] = [H]\,[R] \quad (2.73)$$

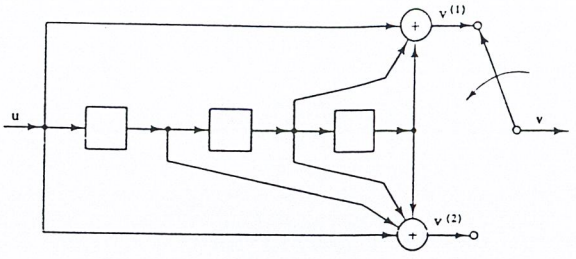
ซึ่งเมตริกซ์ [S] นี้เรียกว่า เมตริกซ์แสดงอาการ (Syndrome Matrix)

2.9 รหัสการประสาน (convolution codes)

จากที่กล่าวในตอนต้น การเข้ารหัสช่องสัญญาณนั้นแบ่งเป็น 2 ชนิด คือ แบบลิเนียร์บล็อคและการเข้ารหัสการประสาน โดยรหัสลิเนียร์บล็อค ตัวแปรที่สำคัญคือ จำนวนบิตของข้อมูลอินพุต k และจำนวนของบิตคำรหัส n โดยอัตราส่วนของตัวแปรทั้งสองเรียกว่า อัตราการเข้ารหัส ในส่วนของรหัสการประสานจะอธิบายอยู่ในรูปของตัวแปร 3 ตัวคือ n , k และ M โดยอัตราส่วนของ k/n ยังคงเรียกว่า "อัตราการเข้ารหัส " ส่วนตัวแปร M จะเรียกว่า "ความยาวคอนสเตร็นท์ " (Constraint Length) ซึ่งหมายถึง จำนวนของขั้นตอนกระบวนการในการเข้ารหัสของชิพรีจิสเตอร์ สิ่งที่แตกต่างกันระหว่างรหัสลิเนียร์ บล็อคและรหัสการประสานที่สำคัญ คือ การเข้ารหัสของรหัสการประสานจะมีหน่วยความจำโดยที่จำนวน n บิตเอาท์พุตของรหัสการประสาน ไม่เพียงขึ้นอยู่กับอินพุท k เท่านั้น แต่ยังขึ้นอยู่กับค่าอินพุทก่อนหน้านั้น M – 1 บิต

2.9.1 หลักการเข้ารหัสการประสาน

พิจารณารูปที่ 2.20 แสดงตัวเข้ารหัสไบนารี่ (2, 1 , 3) ซึ่งประกอบด้วยรีจิสเตอร์ 3 สเตทหรือ m = 3 และค่า n = 2



รูปที่ 2.26 แสดงโครงสร้างตัวเข้ารหัสการประสานแบบ (2,1,3)

ให้ลำดับข่าวสาร (Information sequence) $\mathbf{u} = (u_0, u_1, u_2, \ldots)$ เข้าไปยังตัวเข้ารหัสทีละบิต เนื่องจากตัวเข้ารหัสนี้เป็นระบบลิเนียร์ (Linear System) ดังนั้นลำดับเอาท์พุต (Output Sequence) $V^{(1)} = (v_0^{(1)}, v_1^{(1)}, v_2^{(1)}, \ldots.)$ และ $V^{(2)} = (v_0^{(2)}, v_1^{(2)}, v_2^{(2)}, \ldots.)$ สามารถได้มาจากการทำคอนโวลูชั่นลำดับอินพุต $\mathbf{u}$ กับค่าผลตอบสนองอิมพัลส์ (Impulse Responses) โดยผลตอบสนองอิมพัลส์นี้หาได้จากการป้อนลำดับอินพุต $\mathbf{u} = (1\ 0\ 0\ \ldots)$ และสังเกตลำดับเอาท์พุตทั้งสองที่ออกมา ซึ่งจะได้ $g^{(1)} = (g_0^{(1)}, g_1^{(1)}, \ldots.., g_m^{(1)})$ และ $g^{(2)} = (g_0^{(2)}, g_1^{(2)}, \ldots.., g_m^{(2)})$ โดยสำหรับรูปที่ 2.26 นั้น จะได้ว่า

$$g^{(1)} = (1\ 0\ 1\ 1)$$
$$g^{(2)} = (1\ 1\ 1\ 1)$$

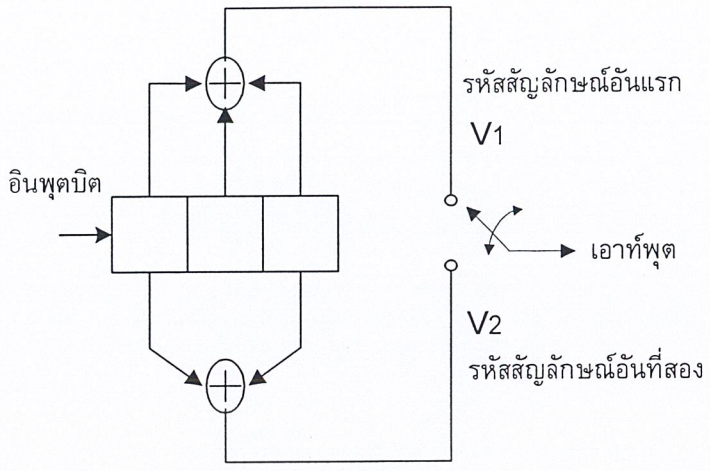
ซึ่งผลตอบสนองอิมพัลส์ $g^{(1)}$ และ $g^{(2)}$ นี้เรียกว่า Generator Sequences ของรหัสและสมการของการเข้ารหัส สามารถเขียนได้เป็น

$$v^{(1)} = u * g^{(1)}$$
$$v^{(2)} = u * g^{(2)} \qquad (2.74)$$

โดย * เป็นการดำเนินการทำดีสครีตคอนโวลูชั่นและการดำเนินการทั้งหมดเป็น modulo – 2 และหลังจากลำดับ $v^{(1)}$ และ $v^{(2)}$ ผ่านตัวมัลติเพลกซ์แล้วเราจะได้คำรหัสคือ

$$V = (v_0^{(1)} v_0^{(2)}, v_1^{(1)} v_1^{(2)}, v_2^{(1)} v_2^{(2)}, \ldots.)$$

ในที่นี้จะทำการพิจารณาที่อินพุตบิต ถูกชิพเข้าไปทีละ 1 บิต และเอาท์พุตที่ได้แต่ละข่าวสารจะมีความยาว n บิต ดังนั้นอัตราการเข้ารหัสมีค่า 1 / n โดย n คือ รหัสสัญลักษณ์ (Code Symbol) ที่เกิดณ เวลา t_i ซึ่งประกอบด้วยคำสาขา (branch word) ที่ i $Vi = v_{1i}, v_{2i}, v_{ni}$ โดย v_{ji} $(j = 1,2,\ldots,n)$ คือ รหัสสัญลักษณ์ที่ j ของคำสาขาที่ i



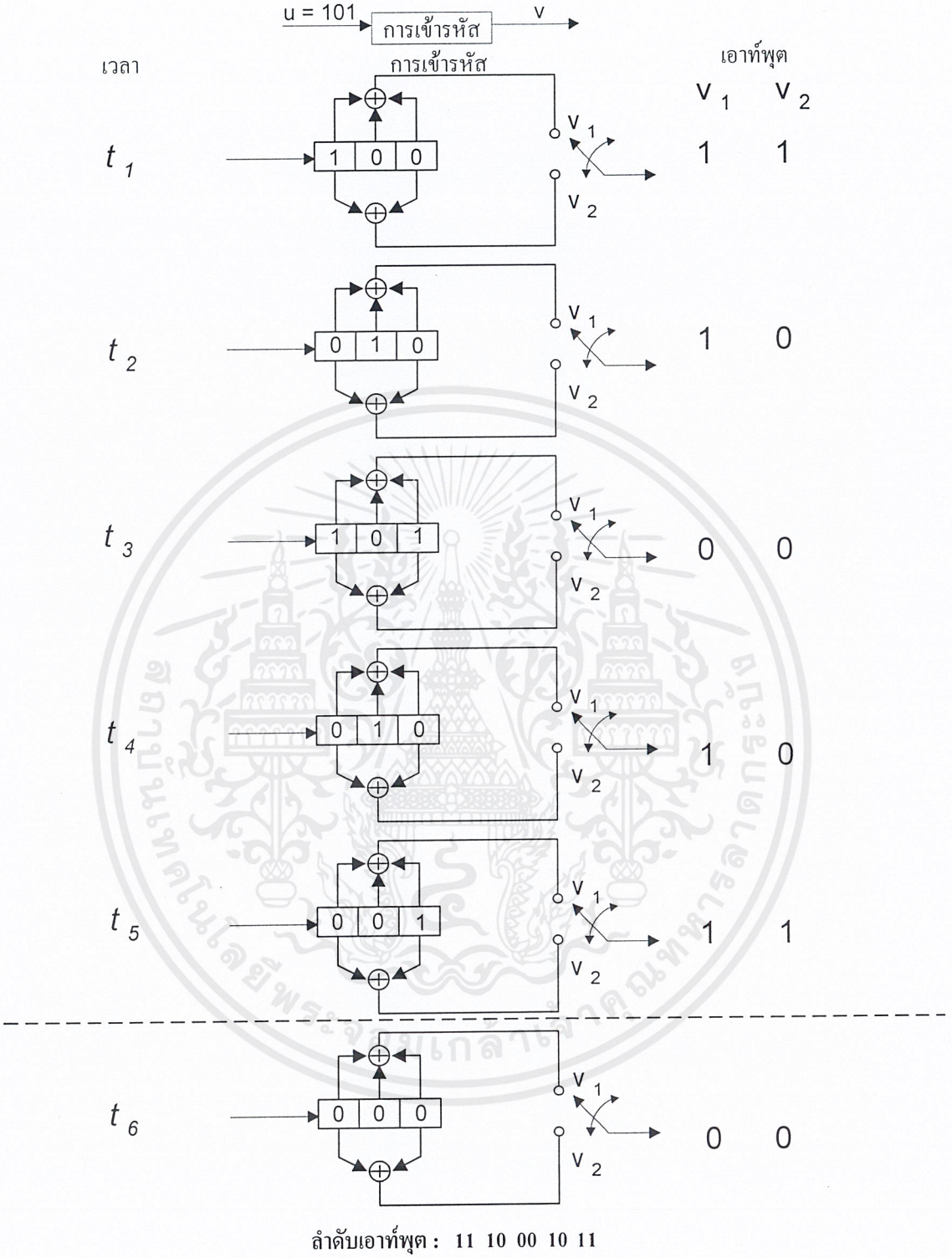
รูปที่ 2.27 แสดงการเข้ารหัสการประสานด้วยอัตราการเข้ารหัส $\frac{1}{2}$, ความยาวคอนสเตร็นท์ M=3

รูปที่ 2.27 แสดงถึงแบบจำลองของการเข้ารหัสการประสาน โดยมีค่าคอนสเตร็นท์ M = 3 เป็นรหัส (1,2) คือ อินพุต 1 บิต ให้เอาท์พุต 2 บิต โดยแต่ละช่วงเวลาอินพุตแต่ละบิตจะถูกชิพเข้าไปในรีจิสเตอร์ ซึ่งเอาท์พุตได้จากสวิตช์สุ่มเลือกจากวงจรบวกมอดูโล 2 แต่ละตัว (เช่น ในตอนแรกเลือกจากวงจรบวกตัวบน แล้วจึงเลือกจากวงจรบวกตัวล่าง) จะสังเกตเห็นว่านอกจากอินพุตแล้ว การเชื่อมต่อระหว่างตำแหน่งของรีจิสเตอร์กับวงจรบวกก็มีผลกับรหัสเอาท์พุตที่ได้ โดยสามารถที่จะแทนการเชื่อมต่อระหว่างชิพรีจิสเตอร์กับวงจรบวกด้วยเวกเตอร์ โดยเวกเตอร์ดังกล่าวจะมีขนาดเท่ากับค่าคอนสเตร็นท์ โดยจะมีค่าเป็น 1 หากมีการเชื่อมต่อ และจะมีค่าเป็น 0 เมื่อไม่มีการเชื่อมต่อโดยจากรูป 2.27 สามารถเขียนการเชื่อมต่อเป็นเวกเตอร์ g_1 สำหรับการเชื่อมต่อด้านบนและ g_2 สำหรับการเชื่อมต่อด้านล่าง

$$g_1 = 1\ 1\ 1$$

$$g_2 = 1\ 0\ 1$$

หากสมมุติว่าข้อมูลอินพุต u = 1 0 1 ถูกทำการเข้ารหัสการประสาน โดยตัวเข้ารหัสแสดงได้ดังรูปที่ 2.27 โดยอินพุตบิต 3 บิต ถูกชิพเข้าไปในเวลา t_1, t_2 และ t_3 ซึ่งผลที่ได้ดังรูปที่ 2.28 ส่วนเวลา t_4 และ t_5 เป็นการชิพบิต 0 เข้าไปซึ่งเรียกว่า Zero – Padding เพื่อให้บิตข้อมูลบิตสุดท้ายถูกชิพไปจนถึงรีจิสเตอร์ตัวสุดท้าย ส่วนเวลา t_6 เป็นการทำให้รีจิสเตอร์มีค่ากลับเป็น 0 ทั้งหมด โดยเอาท์พุตที่ได้จะมีค่าเท่ากับ 1 1 1 0 0 0 1 0 1 1 โดยสัญลักษณ์ที่อยู่ซ้ายมือสุดจะถูกส่งออกไปก่อน



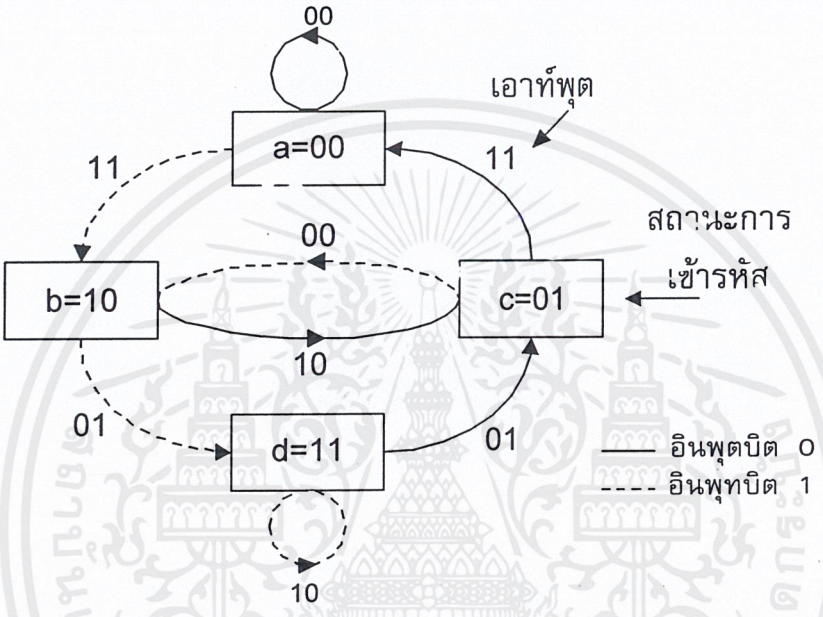
รูปที่ 2.28 ขั้นตอนการเข้ารหัสการประสาน อตราการเข้ารหัส ½

ความยาวคอนสเตร็นท์ M = 3

ในการอธิบายการเข้ารหัสประสานโดยทั่วไปสามารถอธิบายได้ 3 วิธี คือ แผนภาพต้นไม้ แผนภาพ Trellis และแผนภาพสเต็จ ซึ่งจะกล่าวในหัวข้อต่อไป

2.9.2 แผนภาพสเต็จ (State Diagram)

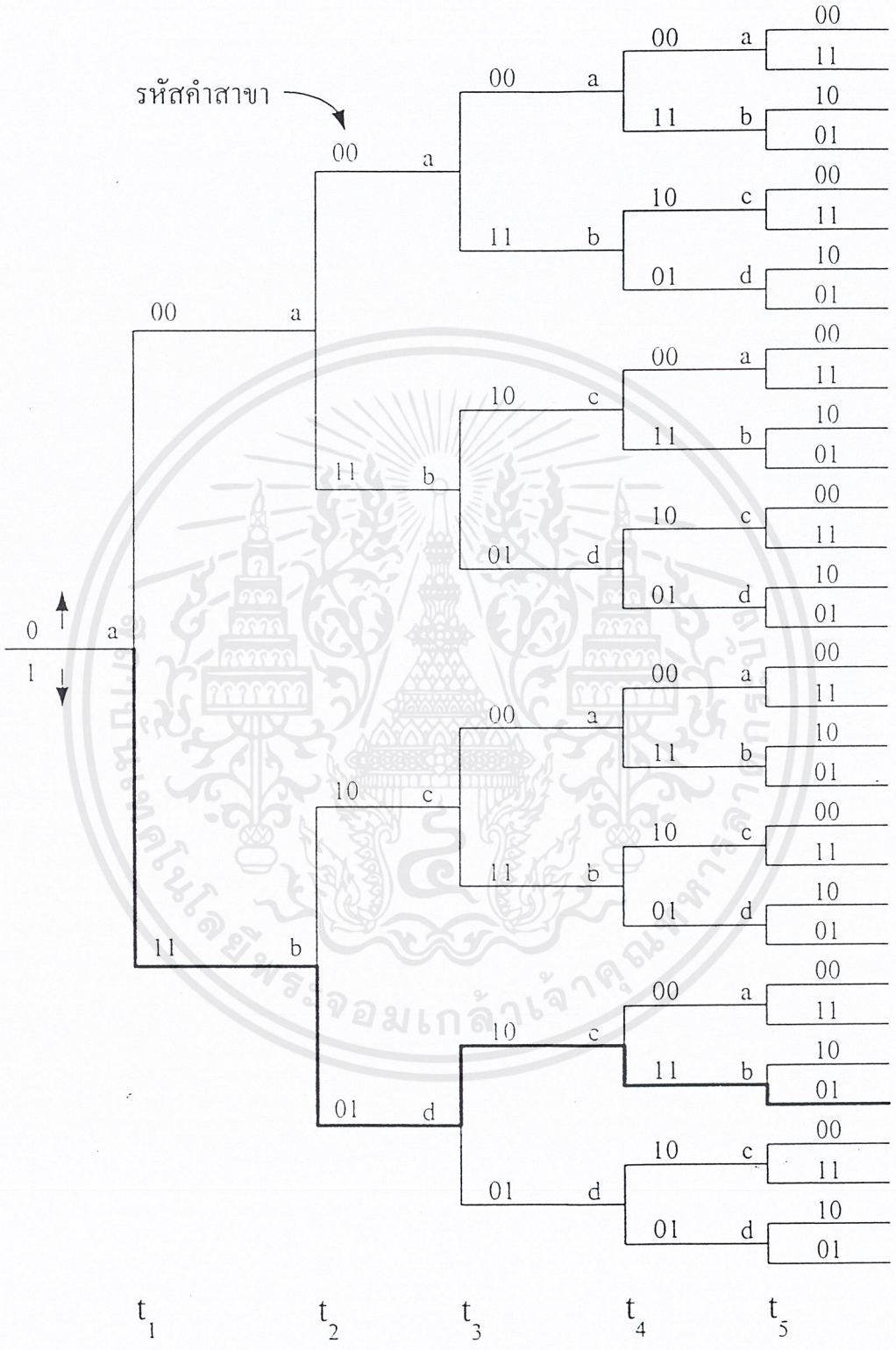
ในการเข้ารหัสการประสานจะเกิดสถานะของรีจิสเตอร์ซึ่งมีผลกับสัญญาณเอาท์พุต 2^{M-1} สถานะ โดยเมื่อนำสถานะดังกล่าวมาเขียนเป็นแผนภาพบล็อค และเชื่อมต่อแต่ละสถานะเข้าด้วยกันจะได้แผนภาพสเต็จ โดยเส้นทางที่เชื่อมต่อระหว่างสถานะต่างๆจะหมายถึง เอาท์พุตของคำสาขาซึ่งเป็นผลมาจากการเปลี่ยนจากสถานะหนึ่งไปเป็นอีกสถานะหนึ่ง ซึ่งสามารถแสดงได้ดังรูป 2.29 ซึ่งสถานะในรีจิสเตอร์ประกอบด้วย a = 00 , b = 10 , c = 01 และ d = 11 ในการเปลี่ยนสถานะนั้นสามารถทำได้จากอินพุตที่เข้ามา ในกรณีที่เป็นเส้นปะ หมายถึง อินพุต 0 ส่วนที่เป็นเส้นทึบ หมายถึง อินพุต 1



รูปที่ 2..29 แผนภาพสเต็จของการเข้ารหัสการประสานด้วยอัตราการเข้ารหัส ½ และ M=3

2.9.3 แผนภาพต้นไม้ (Tree diagram)

ถึงแม้ว่าแผนภาพสเต็จจะอธิบายถึงการเข้าการประสานได้ดี แต่เนื่องจากแผนภาพสเต็จไม่สามารถอธิบายสถานะในแต่ละเวลาที่เปลี่ยนไป ดังนั้นจึงมีการนำเสนอแผนภาพต้นไม้ซึ่งแผนภาพต้นไม้ แสดงดังรูป 2.30 โดยเมื่ออินพุตมีค่าเป็น 0 สาขาของคำจะอยู่ที่สาขาทางขวามือด้านบนแต่เมื่ออินพุทมีค่าเป็น 1 สาขาของคำจะอยู่ที่สาขาทางขวามือด้านล่าง โดยสมมติว่าในสภาวะเริ่มต้นมีค่าเป็น 0 หมด เมื่ออินพุทมีค่าเป็น 0 เข้ามา เอาท์พุตคำสาขาที่ได้มีค่าเป็น 00 แต่หากอินพุตมีค่าเป็น 1 เอาท์พุตคำสาขาจะมีค่าเป็น 11 จากรูป 2.30 แสดงอินพุตบิตมีค่า 1 1 0 1 1 โดยเอาท์พุตที่ได้แสดงได้ดังเส้นทึบ ซึ่งทำให้ลำดับรหัสคำมีค่าเป็น 1 1 0 1 0 1 0 0 0 1

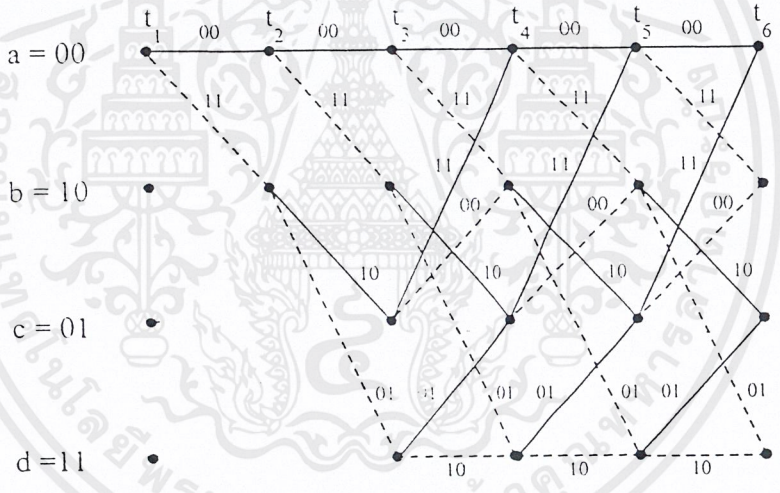


รูปที่ 2.30 แสดงแผนภาพต้นไม้ของการเข้ารหัสการประสานด้วยอัตราการเข้ารหัส 1/2 และ M = 3

อย่างไรก็ตามปัญหาที่เกิดขึ้นของแผนภาพต้นไม้คือ แผนภาพต้นไม้ไม่สามารถอธิบายลำดับที่มีความยาวมาก โดยจำนวนของสาขาจะมีค่าเพิ่มขึ้นด้วยฟังก์ชัน 2^n โดย n คือจำนวนบิตของลำดับอินพุต

2.9.4 แผนภาพ Trellis (Trellis Diagram)

จากรูปที่ 2.30 จะพบว่าเมื่อเวลา t_4 โครงสร้างจะซ้ำแบบเดิมนั้นคือ โครงสร้างจะซ้ำแบบเดิมหลังจากสาขาที่ 3 (โดยทั่วไปเป็นหลังจากสาขาที่ M โดย M เป็นความยาวคอนสเตร็นท์)โดยแต่ละโหนดของแผนภาพต้นไม้จะสอดคล้องกับสถานะต่างๆ 4 สถานะในชิฟรีจิสเตอร์ คือ a = 00 , b = 10 ,c = 01 และ d = 11 โดยที่เวลา t_1 สาขาแรกจะทำให้เกิดโหนด 2 โหนดคือ a และ b และแต่ละสาขาต่อไปก็จะเกิดโหนด 2 โหนดเช่นเดียวกันโดยที่สาขาที่ 2 ณ เวลา t_2 ทำให้เกิดโหนด a,b,c และ d และสาขา t_3 จะมีทั้งหมด 8 โหนดโดยเป็นโหนด a,b,c และ d อย่างละ 2 โหนด จะเห็นได้ว่าสาขาที่เกิดขึ้นจะมีสถานะซ้ำแบบเดิมโดยครึ่งบนและครึ่งล่างของแผนภาพต้นไม้จะมีค่าเหมือนกันดังนั้นเมื่อนำมาเขียนแผนภาพใหม่ซึ่งแสดงได้ดังรูปที่ 2.31 จะเรียกแผนภาพนี้ว่า “แผนภาพ Trellis”



รูปที่ 2.31 แสดงแผนภาพ Trellis ของการเข้ารหัสการประสานด้วยอัตราการเข้ารหัส 1/2 และ M = 3

เช่นเดียวกันกับแผนภาพสเต็จ เส้นทึบหมายถึงเอาท์พุตที่เกิดจากอินพุตบิตที่มีค่า 0 และเส้นประหมายถึง เอาท์พุตที่เกิดจากอินพุตบิตที่มีค่า 1 โดย Trellis จะมีโหนดทั้งหมด 2^{M-1} โหนด แต่ละโหนดของ Trellis หมายถึง สถานะต่างๆ เช่น a = 00, b = 10 , c = 01 และ d =11 จากรูปที่ 2.31 แผนภาพ Trellis จะมีโครงสร้างที่ซ้ำแบบเดิมหลังจากผ่านไป 3 สาขานั้นคือที่เวลา t_4 เอาท์พุตในแต่ละคำสาขาคือ การเปลี่ยนสถานะจากสถานะหนึ่งเป็นอีกสถานะหนึ่งซึ่งแสดงได้ตามสาขาของ Trellis ตามรูป 2.31

2.9.5 เมตริกอูคลิดีน (Euclidean Metric)

สิ่งสำคัญสำหรับการถอดรหัสคือการเลือกเมตริกซ์(Metric) เมตริกซ์คือการวัดทั่วไปสำหรับค่าระยะห่างระหว่างกลุ่มสำหรับการถอดรหัสนั้น กลุ่มในที่นี่คือ สัญญาณที่รับได้ (Received Signal) โดยค่าเมตริกซ์ของการถอดรหัสจะถูกใช้อ้างอิงสำหรับการบอกค่าระยะห่าง (Distance) ระหว่างคำรหัสและเวกเตอร์ใดๆ จากสัญญาณที่รับได้ซึ่งเราได้เคยกล่าวถึงแฮมมิ่ง (Hamming Matric) มาแล้วในตอนต้น แต่ยังมีอีกเมตริกซ์ที่จะใช้ในกรณีการถอดรหัสเป็นแบบ Soft Decision คือ เมตริกซ์อูคลิดีน (Euclidean Metric) เมตริกซ์อูคลิดีนถูกกำหนดสำหรับ (x,y) ∈ R โดย

$$d_E(x,y) = \sqrt{(x-y)^2} \tag{2.75}$$

เมตริกซ์อูคลิดีนนี้จะถูกใช้เมื่อค่าแต่ละสัญลักษณ์ (symbol) ของคำรหัสสามารถถูกแทนด้วยสัญญาณ(signal)

2.9.6 คุณสมบัติของระยะห่างของรหัสการประสาน

ประสิทธิภาพของรหัสการประสานนั้นขึ้นอยู่กับวิธีในการถอดรหัสและคุณสมบัติของค่าระยะห่างของรหัส การวัดค่าระยะห่างที่สำคัญของรหัสการประสานคือ ค่าระยะห่างอิสระ (Free Distnace) d_{free} กำหนดโดย

$$d_{free} = \min\{d(v',v'') : u' \neq u''\} \tag{2.76}$$

โดยค่า v' และ v'' คือคำรหัสที่สอดคล้องกับลำดับข้อมูล u' และ u'' ตามลำดับ (ในกรณีที่มีความยาวที่แตกต่างกัน เราจะทำการเติม 0 เข้าไปเพื่อให้มีความยาวเท่ากัน) ดังนั้นค่า d_{free} คือค่าระยะความแตกต่างต่ำสุด (Minimum Distance) ระหว่างสองคำรหัสใดๆ เพราะว่ารหัสการประสานเป็นรหัสที่เป็นเชิงเส้น ดังนั้น

$$\begin{aligned} d_{free} &= \min\{W(v'+v'') : u' \neq u''\} \\ &= \min\{W(v) : u \neq 0\} \\ &= \min\{W(uG) : u \neq 0\} \end{aligned} \tag{2.77}$$

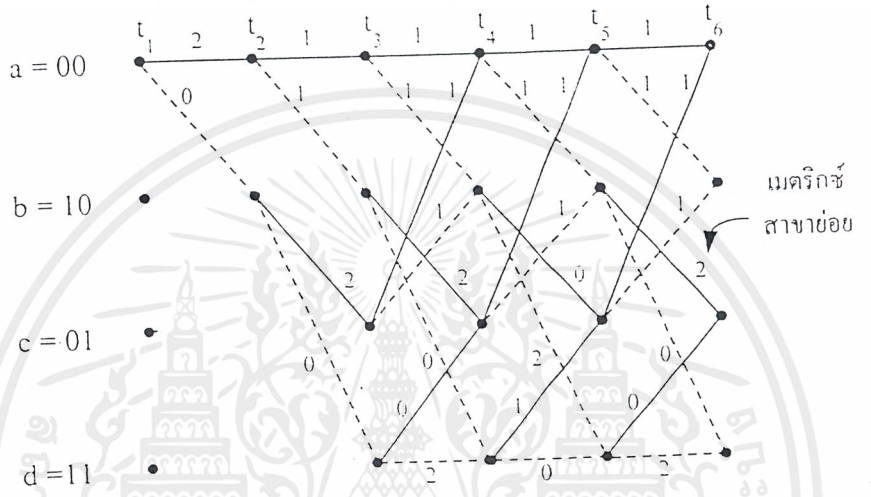
ดังนั้น d_{free} คือค่าน้ำหนักต่ำสุดของคำรหัสของความยาวใดๆที่ได้จากข้อมูลที่ไม่เป็น 0

2.9.7 การถอดรหัสการประสานด้วยวิธี Viterbi

การถอดรหัสด้วยวิธี Viterbi จะทำการเปรียบเทียบเส้นทางที่แตกต่างกัน 2 เส้นทางที่ให้สถานะของรีจิสเตอร์เดียวกัน โดยเส้นทางที่ดีที่สุดจะถูกเลือกและจะถูกเรียกว่า เส้นทางที่เหลืออยู่ (Surving Path) จากรูปที่ 2.32 สามารถจะนำมาเขียนการเข้า ถอดรหัสแผนภาพ Trellis ได้ดังรูปที่ 2.32

โดยหมายเลขที่เขียนบนสาขา Trellis ในแต่ละเวลา t_i จะเป็นระยะห่างแฮมมิ่งระหว่างรหัสสัญลักษณ์ที่รับได้กับคำสาขาที่ได้จากการเข้ารหัส Trellis โดยรูปที่ 2.32 u เป็นลำดับบิตอินพุต v เป็นลำดับคำรหัสที่ส่งจากด้านส่งและ z แทนลำดับที่รับได้

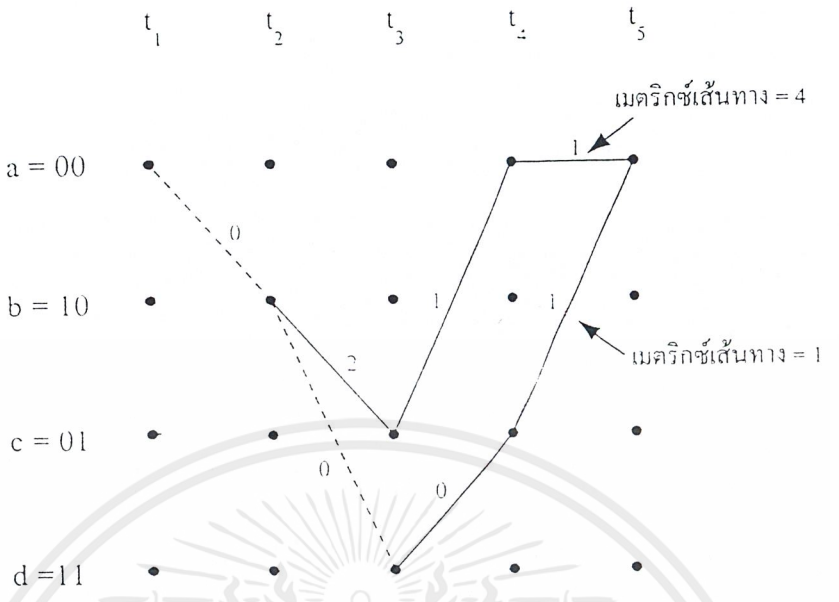
ข้อมูลอินพุท	u :	1	1	0	1	1	. . .
ลำดับคำรหัสที่ส่ง	V:	11	01	01	00	01	. . .
ลำดับที่รับได้	Z :	11	01	01	10	01	. . .



รูปที่ 2.32 การถอดรหัสโดยใช้แผนภาพ Trellis ของการเข้ารหัสการประสาน ด้วยอัตราการเข้ารหัส ½ และ M = 3

จากรูปที่ 2 .32 ลำดับที่รับได้ Z ณ.เวลา t_1 มีค่าเป็น 11 เมื่อเปรียบกับรูปที่ 2.31 ซึ่งเปลี่ยนจากสถานะ $00 \rightarrow 00$ จะได้เอาท์พุตคำสาขา 00 ซึ่งทำให้มีความแตกต่างระยะแฮมมิ่งเท่ากับ 2 และการเปลี่ยนจากสถานะ $00 \rightarrow 10$ ณ. เวลา t_1 ได้เอาท์พุตคำสาขา 11 ซึ่งจะสอดคล้องกับรหัสที่รับได้ นั่นคือ มีค่าความแตกต่างระยะแฮมมิ่งเท่ากับ 0 ซึ่งจะทำการเปรียบเทียบอย่างนี้เรื่อยไปในแต่ละช่วงเวลา t_i ในกระบวนการถอดรหัสโดยใช้เมตริกซ์ของความแตกต่างระยะแฮมมิ่งตลอดเส้นทาง Trellis นี้ เพื่อหาเส้นทางที่น่าจะเป็นไปได้มากที่สุด หรือเส้นทางที่ให้ค่าความแตกต่างของระยะแฮมมิ่งต่ำสุด

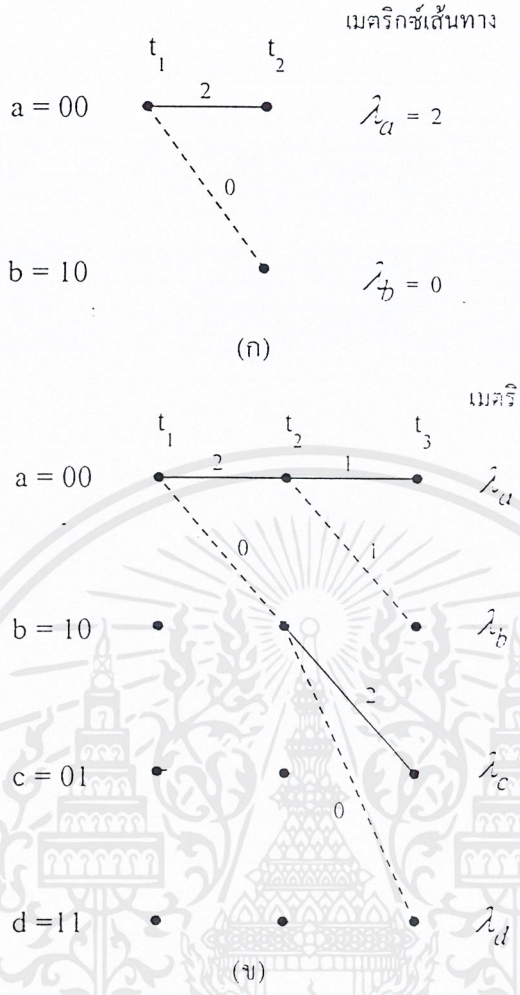
พื้นฐานการถอดรหัสด้วยวิธี Viterbi คือ เมื่อมีเส้นทางสองเส้นทางของ Trellis ที่มาพบกันที่สถานะเดียวกันจะมีเส้นทางหนึ่งจะไม่ถูกนำมาพิจารณาอีก ในรูปที่ 2 .33 แสดงสองเส้นทางที่มาพบกันที่เวลา t_5 ณ. สถานะ 00 ซึ่งจะต้องทำการพิจารณาเมตริกซ์ความแตกต่างของระยะแฮมมิ่งโดยรวมระหว่างสองเส้นทาง ซึ่งจะเป็นผลรวมของเมตริกซ์ความแตกต่างของระยะแฮมมิ่งในแต่ละช่วงเวลา t_i จะพบว่าจากรูปที่ 2.33 ในเส้นทางบนจะให้ผลรวมของเมตริกซ์ความแตกต่างของระยะแฮมมิ่งเท่ากับ 4 และเส้นทางล่างได้ผลรวมของเมตริกซ์ความแตกต่างของระยะแฮมมิ่งเท่ากับ 1 แสดงว่าเส้นทางบนจะไม่ถูกนำมาพิจารณาอีก เนื่องจากให้ค่าเมตริกซ์ความแตกต่างของระยะแฮมมิ่งมากกว่าเส้นทางด้านล่าง



รูปที่ 2 .33 แสดงเมตริกซ์เส้นทางสองเส้นทางที่มาพบกัน

ในกระบวนการถอดรหัสด้วยวิธีไวเทอร์บี(Viterbi) จะทำการคำนวณเมตริกซ์ระหว่างสองเส้นทางที่มาถึงสถานะเดียวกัน จากนั้นจะทำการกำจัดเส้นทางหนึ่งทิ้งไป โดยในการคำนวณดังกล่าวจะทำในแต่ละโหนด ณ. เวลา t_i จากนั้นจึงจะถอดรหัส ณ. เวลาถัดไปซึ่งจะทำกระบวนการดังกล่าวซ้ำอีก ซึ่งรูปที่ 2 .34 แสดงขั้นตอนการถอดรหัสด้วยวิธี ไวเทอร์บี(Viterbi) โดยสมมุติให้ข้อมูลอินพุตเป็น u คำรหัสเป็น V และลำดับที่รับได้เป็น Z เช่นเดียวกับรูปที่ 2 .32 เมื่อเวลา t_1 ด้านรับรับได้ 11 จากรูปที่ 2.34ก มีการเปลี่ยนจากสถานะ $00 \rightarrow 00$ และ $00 \rightarrow 10$ ซึ่งจะมีเมตริกซ์สาขาเท่ากับ 2 และ 0 ตามลำดับ ต่อมาที่เวลา t_2 ในแต่ละสถานะจะมีสาขาอย่างละ 2 สาขาซึ่งแสดงได้ดังรูปที่ 2.34ข โดยผลรวมของเมตริกซ์ของแต่ละสาขามีค่าเป็น λ_a , λ_b, λ_c และ λ_d ที่เวลา t_3 ในแต่ละสถานะยังคงมีอีก 2 สาขา ซึ่งแสดงได้ดังรูป 2 .34ค เป็นผลทำให้แต่ละโหนดจะมีเส้นทางที่มาถึง 2 เส้นทาง จากที่กล่าวไว้ในตอนต้นเส้นทางที่ให้ค่าผลเมตริกซ์เส้นทางมากที่สุดจะถูกกำจัดออกไป เส้นทางที่เหลืออยู่แสดงได้ดังรูป 2.34ง ซึ่งเมื่อถึงกระบวนการนี้จะพบว่าที่เวลาระหว่าง t_1 และ t_2 ตัวถอดรหัสจะเลือกเส้นทาง $00 \rightarrow 10$ เนื่องจากการเปลี่ยนสถานะดังกล่าวเกิดจากอินพุตที่มีค่าบิตเป็น 1 ดังนั้นตัวถอดรหัสจึงถอดรหัสได้บิตแรก คือ 1 จะพบว่าการถอดรหัสบิตแรกนั้นจะทำได้เมื่อเวลาผ่านไปช่วงเวลาหนึ่ง

ในแต่ละกระบวนการถอดรหัสแต่ละโหนดจะมีเส้นทางที่มาถึง 2 เส้นทาง ซึ่งรูปที่ 3.34จ แสดงเส้นทางที่มาถึงโหนดเมื่อเวลา t_5 ซึ่งจะมีเส้นทางหนึ่งที่ถูกกำจัด เส้นทางที่เหลือแสดงได้ดังรูป 3.34ฉ จะเห็นได้ว่ายังไม่สามารถที่จะถอดรหัสบิตที่สองได้ ซึ่งมีค่าเป็น 1 ซึ่งจะทำการถอดรหัสในลักษณะเช่นนี้ไปจนกว่าจะถอดรหัสได้ครบ



รูปที่ 2.34 การเลือกเส้นทางที่เหลืออยู่ (ก) เส้นทางที่เหลือเมื่อเวลา t_2

(ข) เส้นทางที่เหลือเมื่อเวลา t_3

(ค) การเปรียบเทียบเมตริกซ์เมื่อเวลา t_4

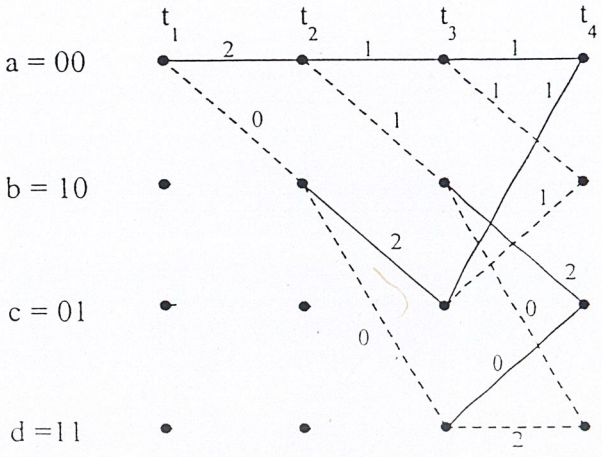
(ง) เส้นทางที่เหลือเมื่อเวลา t_4

(จ) การเปรียบเทียบเมตริกซ์เมื่อเวลา t_5

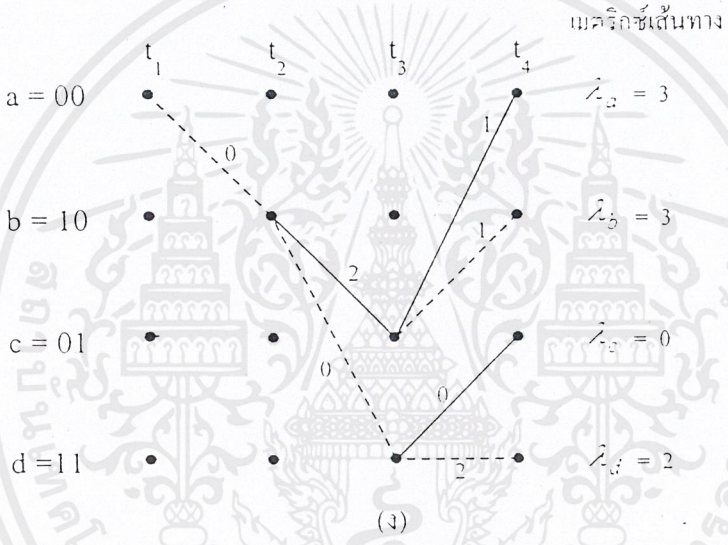
(ฉ) เส้นทางที่เหลือเมื่อเวลา t_5

(ช) การเปรียบเทียบเมตริกซ์เมื่อเวลา t_6

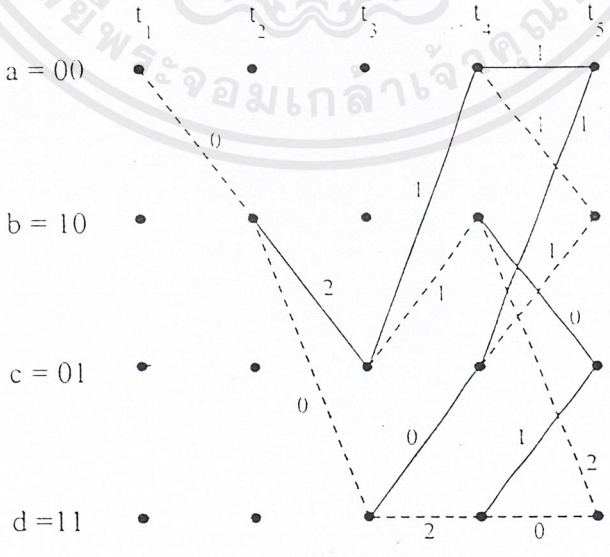
(ซ) เส้นทางที่เหลือเมื่อเวลา t_6



(ค)

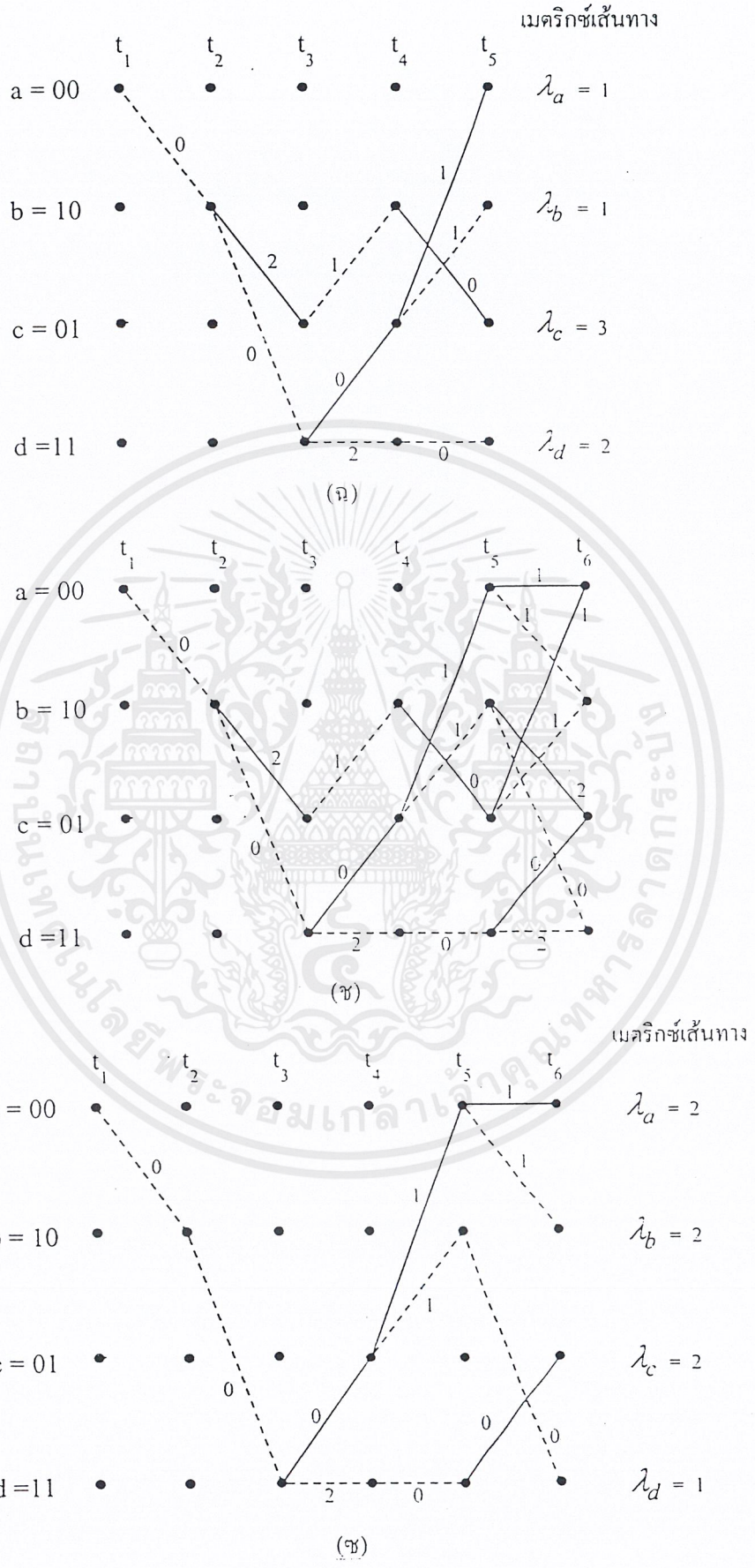


(ง)



(จ)

รูปที่ 2.34 (ต่อ)

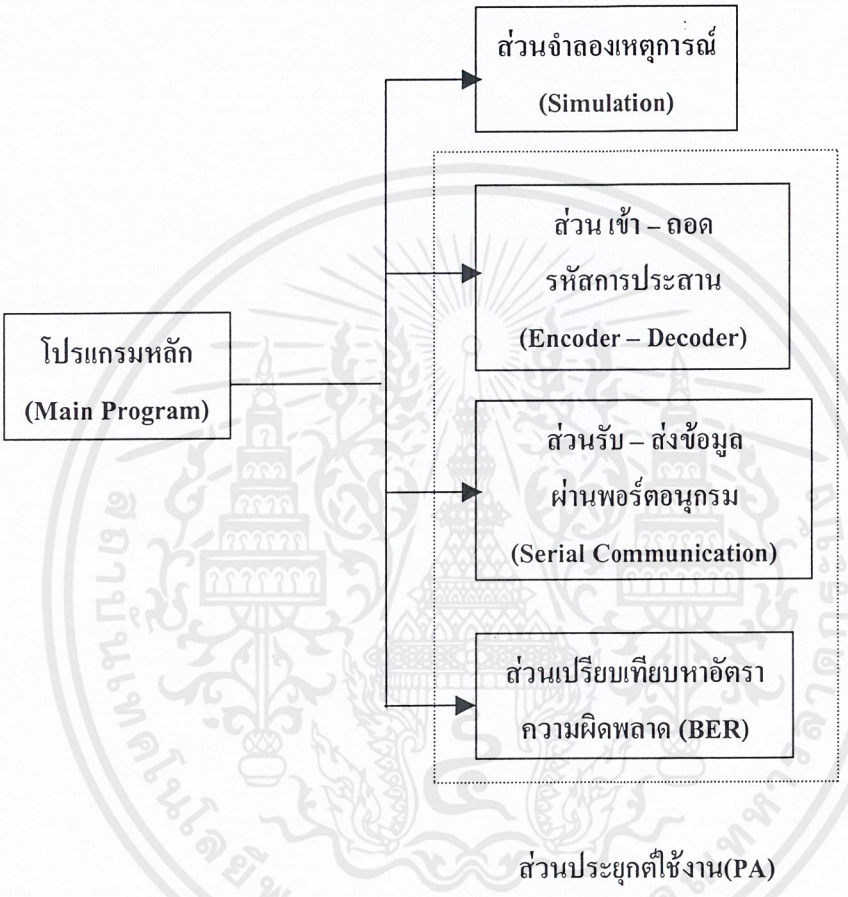


รูปที่ 2.34 (ต่อ)

บทที่ 3
การคำนวณและการสร้าง

3.1 โครงสร้างของโปรแกรม

ส่วนประกอบโครงสร้างของตัวโปรแกรม สามารถแบ่งออกได้เป็น 4 ส่วนใหญ่ๆ ดังแสดงในรูปที่ 3.1



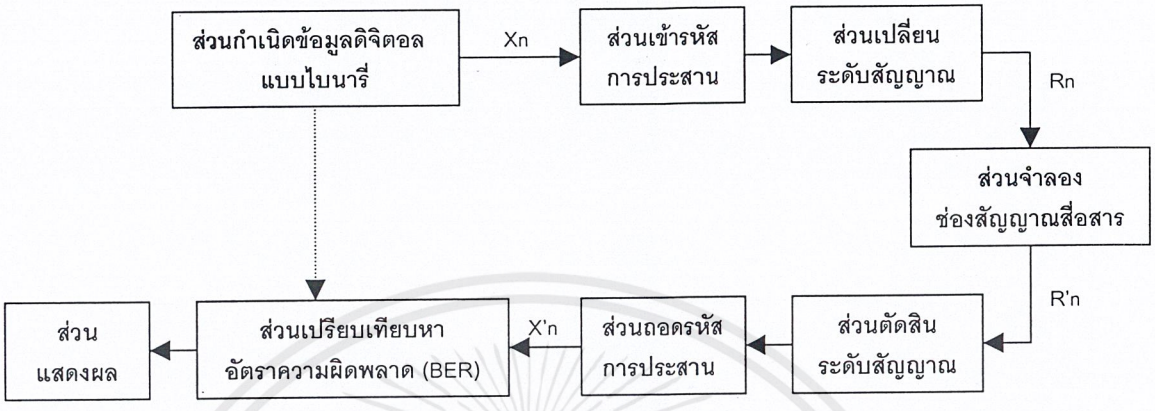
รูปที่ 3.1 แสดงโครงสร้างของโปรแกรม

โปรแกรมนี้เขียนด้วยภาษา Visual C++ เวอร์ชั่น 6.0 โดยการแสดงผลและวิเคราะห์บางส่วนได้ใช้โปรแกรม MATLAB ช่วยเหลือ จากรูปที่ 3.1 จะเห็นว่าเราพิจารณาการใช้งานของตัวโปรแกรมออกเป็น 2ส่วนคือ

1. ส่วนจำลองเหตุการณ์ (Simulation)
2. ส่วนประยุกต์ใช้งาน (PA : Practical Application)

3.2 ส่วนจำลองเหตุการณ์ (Simulation)

3.2.1 บล็อคไดอะแกรม (Block Diagram)



รูปที่ 3.2 แสดงบล็อคไดอะแกรมของส่วนจำลองเหตุการณ์ (Simulation)

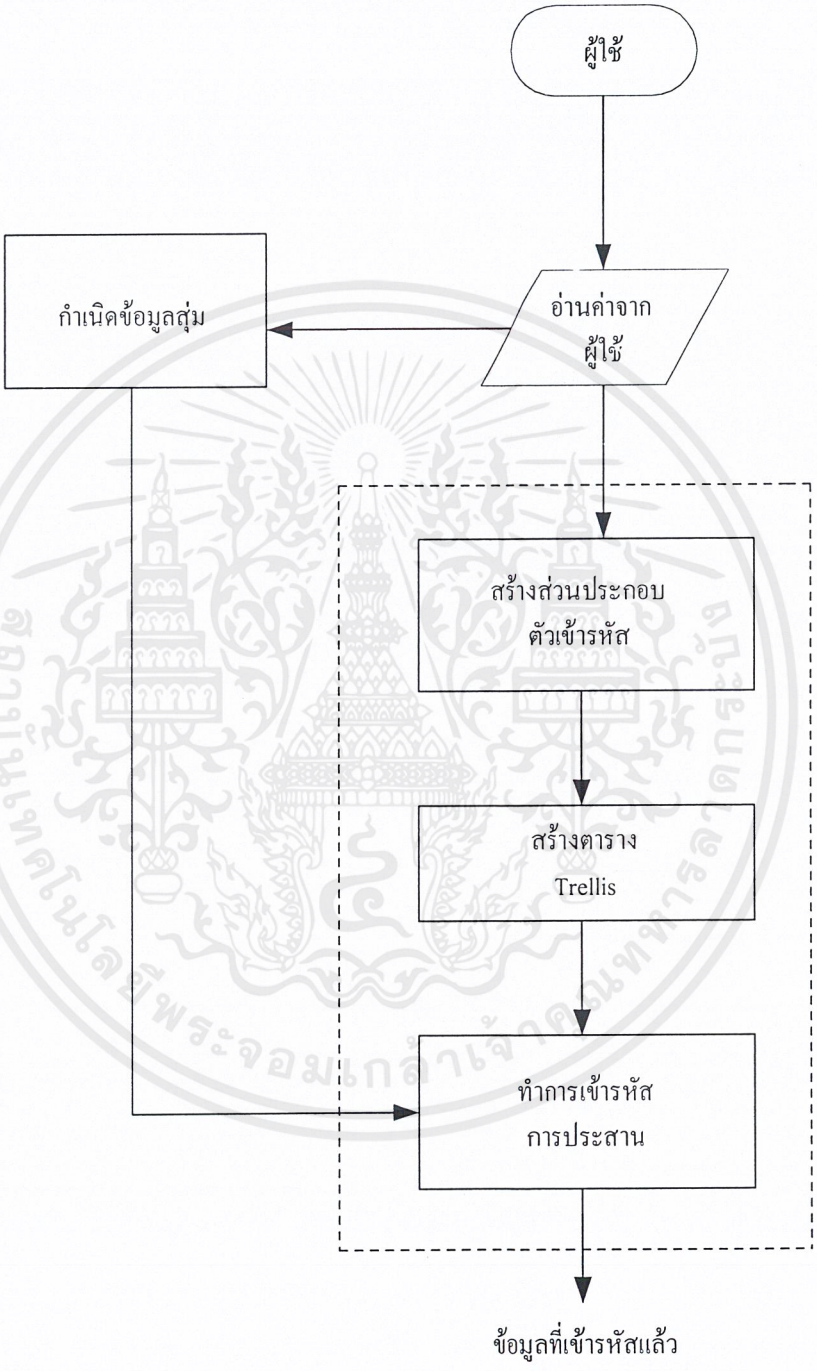
ส่วนประกอบของระบบพื้นฐาน คือ ตัวส่ง (Transmitter), ช่องสัญญาณสื่อสาร (Communication Channel) และตัวรับ (Receiver) โดยจากรูปที่ 3.2 สัญญาณ Rn คือสัญญาณที่ได้หลังจากทำการเข้ารหัสและแปลงระดับสัญญาณเพื่อส่งผ่านไปยังช่องสัญญาณสื่อสาร โดยจุดประสงค์ของการเข้ารหัสก็เพื่อให้ลำดับข่าวสาร (Message Sequence) มีระดับความน่าเชื่อถือมากขึ้นด้วยการปรับปรุงค่าอัตราความผิดพลาด (Bit Error -Rate) ที่มีต่อช่องสัญญาณรบกวน (Noisy Channel) โดยเมื่อสัญญาณ $R'n$ ผ่านการตัดสินระดับสัญญาณแล้วส่งผ่านไปยังตัวถอดรหัสจนได้สัญญาณ $X'n$ แล้ว ค่าสัญญาณเอาท์พุท $X'n$ ที่ได้นี้อาจมีค่าความผิดพลาดอยู่บ้างขึ้นกับสภาพของช่องสัญญาณ (Channel) และประสิทธิภาพของการเข้ารหัส

3.2.2 ส่วนกำเนิดข้อมูลดิติตอลแบบไบนารี่

ส่วนแรกจะเป็นการกำเนิดข้อมูลสุ่มขึ้นมาก่อน โดยข้อมูลที่ได้จะมีลักษณะแบบไบนารี่ คือ มีค่าลอจิก “0” หรือ “1” ซึ่งจำนวนข้อมูลที่ใช้ทั้งหมดในการจำลองเหตุการณ์นี้จะเท่ากับ 2 ล้านบิต

3.2.3 ตัวเข้ารหัสการประสาน (Convolutional Encoder)

สำหรับขั้นตอนในการเข้ารหัสการประสานนั้น จะแสดงได้ดังรูปที่ 3.2



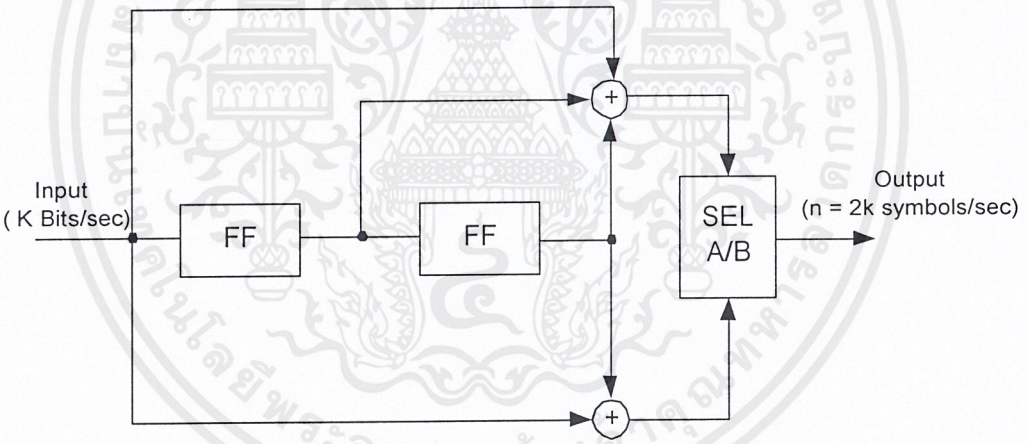
รูปที่ 3.3 แสดงโฟรชาร์ต(Flow Chart) ของโปรแกรมในส่วนเข้ารหัสการประสาน

โดยส่วนประกอบที่สำคัญของรูปที่ 3.3 นั้น คือส่วนของการสร้างองค์ประกอบของตัวเข้ารหัส และการสร้างตาราง Trellis Diagram ซึ่งจะแสดงรายละเอียดดังต่อไปนี้

<u>ส่วนสร้างองค์ประกอบของตัวเข้ารหัส</u> สำหรับในการจำลองเหตุการณ์ (Simulation) ครั้งนี้นั้น เราใช้อัตราการเข้ารหัส (Code Rate) ที่ 1/n โดย n = 2,3,4,... แล้วกำหนดการแปรค่าของจำนวนรีจีสเตอร์ (m) ซึ่งมีค่าตั้งแต่ m = 2 ถึง m = 5 หรือถ้าเราจะพูดในเทอมของความยาวคอนสเตนท์ (Constraint Length) จะมีค่าตั้งแต่ K = 3 ถึง K = 6 ซึ่งในแต่ละค่าของจำนวนรีจีสเตอร์นั้นจะมีค่า Generator Sequence ที่แตกต่างกันไปด้วย ทำให้ได้รูปแบบโครงสร้างของตัวเข้ารหัสที่แตกต่างกัน ซึ่งค่า Generator Sequence นี้โดยทั่วไปจะแสดงในรูปของเลขฐานแปด (Octal Number) จากรูปที่ 3.3 ผู้ใช้จะเป็นคนกำหนดค่าอัตราเข้ารหัส (Code Rate), ความยาวคอนสเตนท์, รูปแบบของการตัดสินระดับสัญญาณ และลักษณะของช่องสัญญาณสื่อสาร โดยรายละเอียดของโครงสร้างตัวเข้ารหัสที่อัตราเข้ารหัส ½ มีดังนี้

1. ที่ค่า m = 2, K = 3 จะใช้

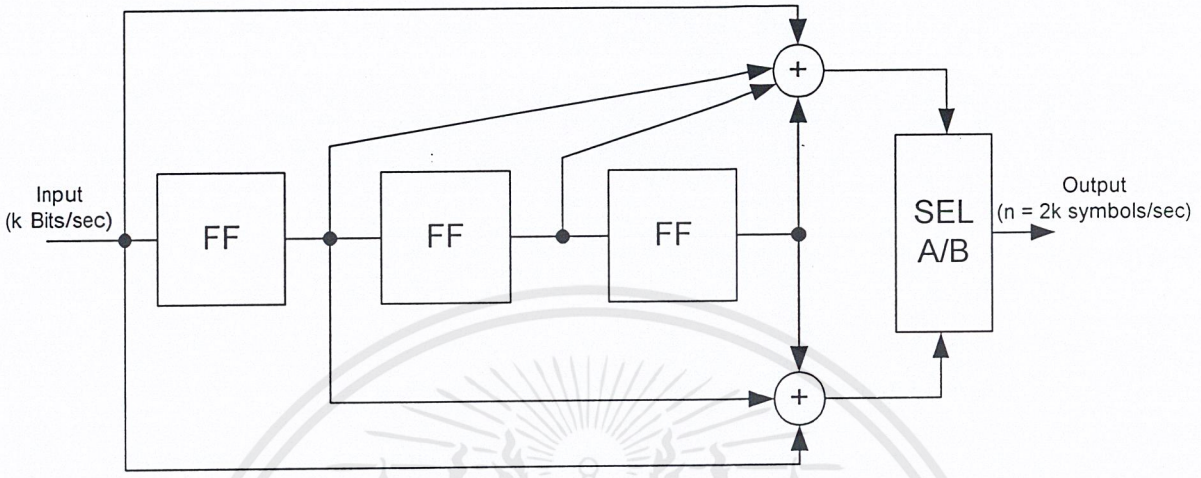
Generator Sequence $G(0) = 7_8 = 7_{10} = 111_2$ ซึ่งให้ค่า $d_{free} = 5$ $G(1) = 5_8 = 5_{10} = 101_2$



รูปที่ 3.4 แสดงโครงสร้างของส่วนเข้ารหัสการประสานเมื่อใช้จำนวนรีจีสเตอร์เท่ากับ 2

2. ที่ค่า m = 3 , K = 4 จะใช้

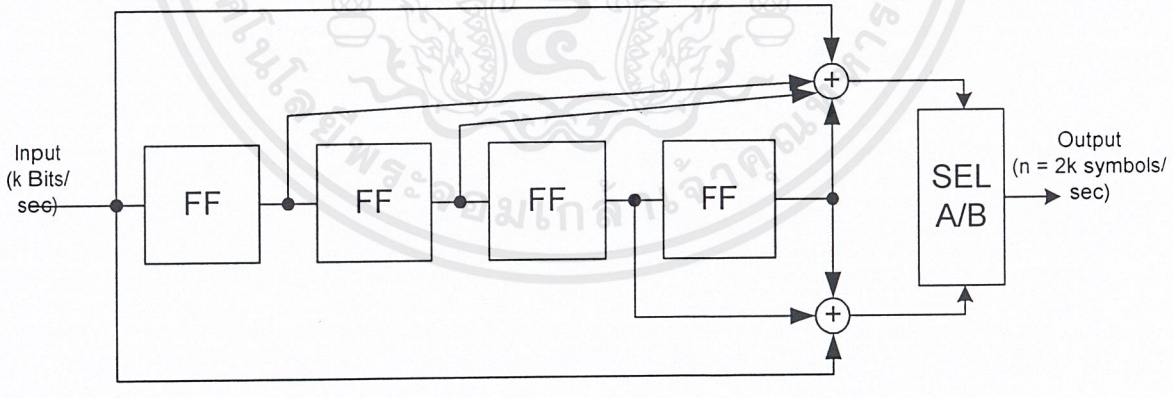
Generator Sequence $G(0) = 17_8 = 15_{10} = 1111_2$ ซึ่งให้ค่า $d_{free} = 6$ $G(1) = 15_8 = 13_{10} = 1101_2$



รูปที่ 3.5 แสดงโครงสร้างของส่วนเข้ารหัสการประสานเมื่อใช้
จำนวนรีจีสเตอร์เท่ากับ 3

3.ที่ค่า m = 4 , K = 5 จะใช้

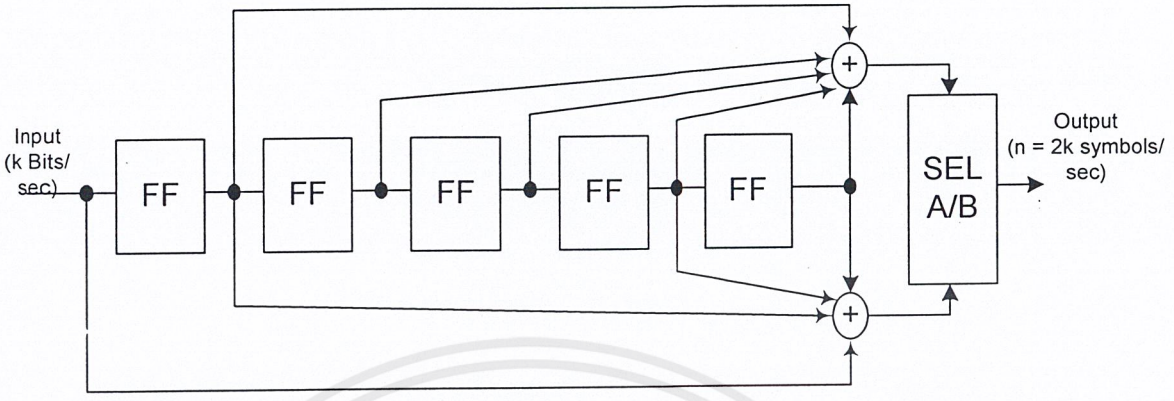
Generator Sequence $G(0) = 35_8 = 29_{10} = 11101_2$ ซึ่งให้ค่า $d_{free} = 7$ $G(1) = 23_8 = 19_{10} = 10011_2$



รูปที่ 3.6 แสดงโครงสร้างของส่วนเข้ารหัสการประสานเมื่อใช้
จำนวนรีจีสเตอร์เท่ากับ 4

4.ที่ค่า m = 5 , K = 6 จะใช้

Generator Sequence $G(0) = 59_8 = 47_{10} = 101111_2$ ซึ่งให้ค่า $d_{free} = 8$ $G(1) = 65_8 = 53_{10} = 110101_2$

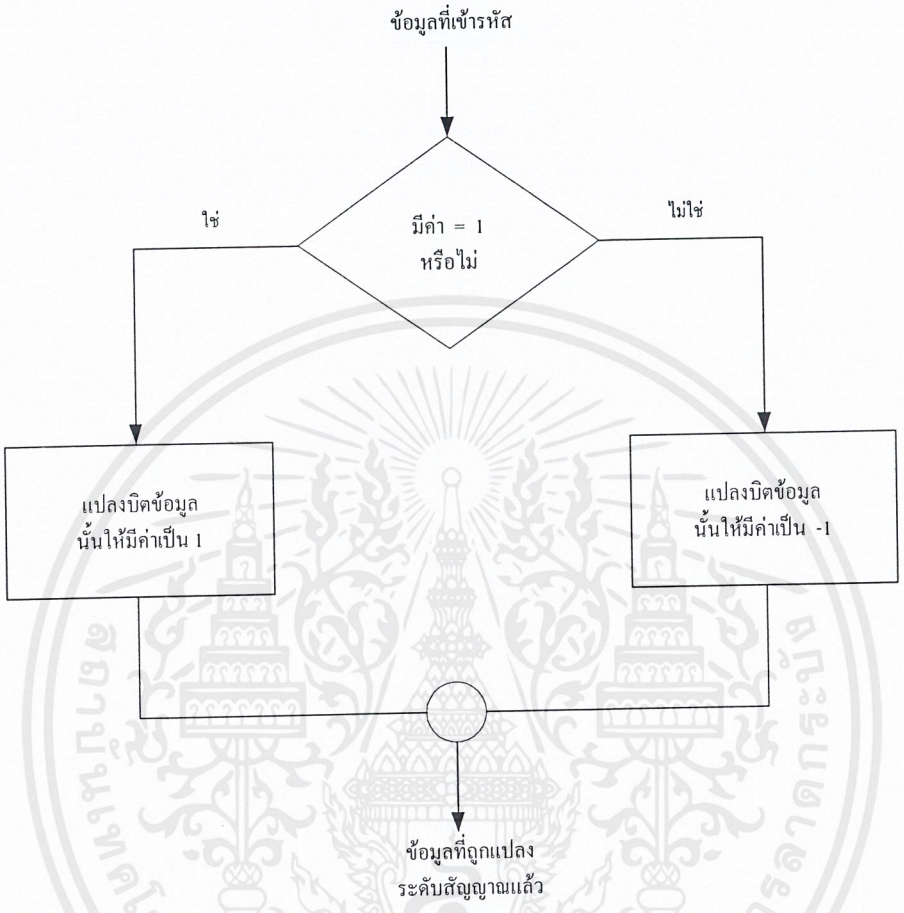


รูปที่ 3.7 แสดงโครงสร้างของส่วนเข้ารหัสการประสานเมื่อใช้
จำนวนรีจีสเตอร์เท่ากับ 5

ในทางปฏิบัตินั้นค่า K จะจำกัดที่ K = 9 เนื่องจากยิ่งมีค่าจำนวนรีจีสเตอร์หรือความยาว - คอนสเตรนท์มากขึ้นนั้น นั่นหมายถึงจำนวนหน่วยความจำที่ต้องการมากขึ้นด้วย โดยเฉพาะอย่างยิ่งทางด้านเครื่องรับซึ่งจะต้องจดจำค่าสเตท ค่าเมตริกซ์ของเส้นทาง จากรูปที่ 3.3 เมื่อผู้ใช้กำหนดค่าพารามิเตอร์ต่างๆ แล้ว ตัวโปรแกรมจะทำการกำเนิดข้อมูลสุ่มขึ้นมาเพื่อใช้เป็นอินพุทในการเข้ารหัส โดยในการเข้ารหัสแต่ละครั้งเราจะทำการแบ่งข้อมูลออกเป็นแพ็คเก็ต (Packet) ซึ่งจำนวนบิตข้อมูลทั้งหมดที่ใช้ในการจำลองผลนั้นเท่ากับ 2 ล้านบิต แต่เนื่องจากยิ่งข้อมูลในการเข้ารหัสแต่ละแพ็คเก็ตนั้นมีค่ามาก หน่วยความจำที่ต้องการในการจองพื้นที่ของโปรแกรมก็ยิ่งต้องการมากขึ้น ดังนั้นเราจึงกำหนดจำนวนบิตข้อมูลที่ใช้ในการเข้ารหัสแต่ละแพ็คเก็ตให้มีค่าไม่เท่ากันในแต่ละโครงสร้างของตัวเข้ารหัสหรือจำนวนของรีจีสเตอร์ เพื่อให้เหมาะสมและสอดล้องกับพื้นที่ของหน่วยความจำที่มีจำกัดแต่ยังคงใช้จำนวนบิตข้อมูลทั้งหมดเท่ากับ 2 ล้านบิต ซึ่งมีค่ามากเพียงพอในการคำนวณหาค่าความน่าจะเป็นของความผิดพลาดที่พอเชื่อถือได้ หลังจากสร้างส่วนประกอบตัวเข้ารหัสแล้ว โปรแกรมจะทำการสร้างข้อมูลของตาราง Trellis เพื่อใช้ในการเข้ารหัสต่อไป

3.2.4 ส่วนเปลี่ยนระดับสัญญาณ

สำหรับส่วนนี้ จะทำการเปลี่ยนระดับสัญญาณข้อมูลซึ่งบิต 0 จะให้มีค่าเป็น -1 และบิต 1 จะให้มีค่าเป็น 1 ดังรูปที่ 3.8



รูปที่ 3.8 แสดงโฟรชาร์ต(Flow Chart) ของโปรแกรมในส่วนมอดูเลต

3.2.5 ส่วนจำลองช่องสัญญาณสื่อสาร

สำหรับช่องสัญญาณนั้นเราแบ่งการจำลองออกเป็น 2 ลักษณะคือ ช่องสัญญาณแบบ BSC และช่องสัญญาณแบบ AWGN

- ช่องสัญญาณแบบ Binary Symmetric Channel (BSC)

ซึ่งช่องสัญญาณนี้จะถูกกำหนดด้วยพารามิเตอร์ Pe (Crossover Probability Of Error) ดังรูปที่ 2.17 โดย

$$Pe = Q\sqrt{\frac{2E_s}{N_o}} \tag{3.1}$$

ซึ่งค่า Es/No เป็นค่าอัตราส่วนธรรมดาไม่ใช่หน่วย dB และ

$$\frac{E_s}{N_o} = \frac{E_b}{N_o} + 10\log_{10}(\frac{k}{n}) \qquad \text{(หน่วย dB)} \tag{3.2}$$

โดยค่าอัตราส่วน k/n คือ อัตราการเข้ารหัส เช่น เมื่อเราใช้อัตราเข้ารหัสที่ ½ จะได้

Es/No = Eb/No – 3.01 dB

โปรแกรมจะกำหนดค่าของ Eb/No แล้วทำการคำนวณหา Pe หลังจกเมื่อรู้ค่า Pe โปรแกรมจะกำเนิดสุ่มค่าด้วยฟังก์ชัน rand() และเทียบกับค่า Pe ถ้าหากค่าที่ได้จากการสุ่มมีค่าน้อยกว่า Pe แล้วสัญญาณที่ส่งจะถูกกลับค่าจาก 1 เป็น -1 หรือจากค่า -1 เป็น 1

- ช่องสัญญาณแบบ Additive White Guassian Noise (AWGN)

สำหรับช่องสัญญาณสื่อสารนี้มีความสำคัญมากเพราะช่องสัญญาณ ในทางปฏิบัติส่วนใหญ่จะมีลักษณะแบบนี้ส่วนการจำลองนั้นเราจะให้โปรแกรมกำเนิดตัวแปรสุ่มขึ้นมา โดยการกระจายของตัวแปรสุ่มนี้จะต้องมีลักษณะรูปร่างแบบระฆังคว่ำหรือมีการแจกแจงแบบเกาส์เซียน และตัวแปรสุ่มที่ได้นี้จะมีค่าเบี่ยงเบนมาตรฐาน σ ที่สัมพันธ์กับค่าของ Es/No ดังนี้

$$\sigma = \sqrt{\frac{1}{2(E_S/N_O)}} \qquad (3..3)$$

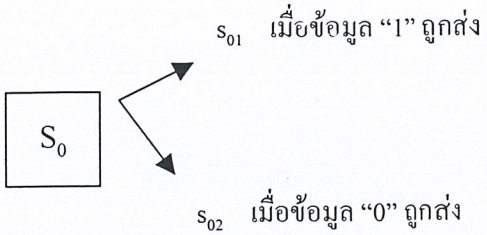
โดยค่าอัตราส่วน Es/No เป็นค่าอัตราส่วนธรรมดา ไม่ใช่หน่วย dB ตัวโปรแกรมจะกำหนดค่า Eb/No ขึ้นมา ซึ่งสามารถหา Es/No ได้จากความสัมพันธ์ในสมการที่ (3.2) ดังนั้นค่าเบี่ยงเบนมาตรฐานของตัวแปรสุ่มจะเปลี่ยนแปลงไปตามค่าอัตราส่วน Es/No หรือ Eb/No ซึ่งค่าตัวแปรสุ่มนี้จะถูกนำไปบวกหรือรวมกับขนาดของสัญญาณที่ส่งผ่านช่องสัญญาณสื่อสารมา นั่นคือ ระดับของสัญญาณจะมีอยู่ 2 ระดับคือ +1 และ -1 จะรวมกับตัวแปรสุ่มที่มีค่าเบี่ยงเบนมาตรฐานเท่ากับ σ

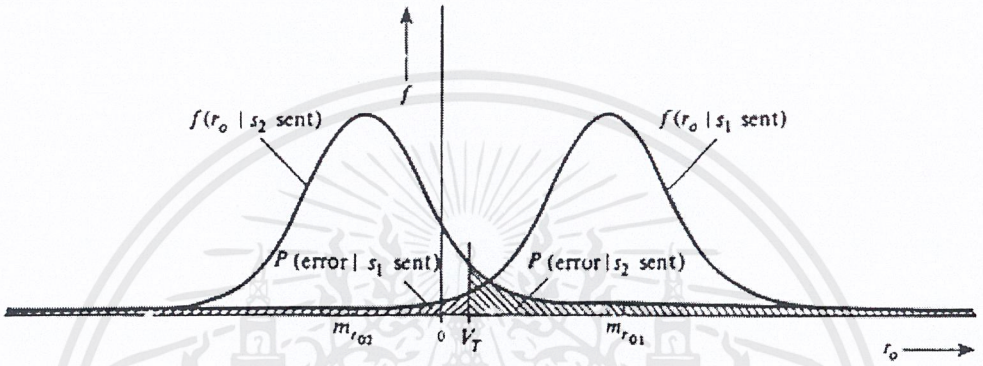
3.2.6 ส่วนตัดสินระดับสัญญาณ

ในกรณีของการตัดสินระดับสัญญาณนั้นค่าสัญญาณข้อมูลที่รับได้หลังจากผ่านช่องสัญญาณ (Channel) จะถูกนำมาเปรียบเทียบกับระดับอ้างอิง (Threshold) กล่าวคือ ในกรณีของการตัดสินใจแบบ Hard Decision นั้น สัญญาณที่รับได้จะถูกเปรียบเทียบกับระดับอ้างอิงค่าหนึ่ง ซึ่งหากข้อมูลที่เข้ามามีค่ามากกว่าระดับอ้างอิง จะถูกตัดสินให้บิตข้อมูลนั้นมีค่าเป็น 1 และจะถูกตัดสินให้มีค่าเป็น 0 ในทางตรงกันข้าม

แต่ในกรณีของการตัดสินใจแบบ Soft Decision นั้น สัญญาณที่เข้ามาจะไม่ถูกทำการเปรียบเทียบจัดระดับ โดยจะถูกส่งต่อไปยังส่วนถอดรหัสเลยซึ่งเสมือนกับค่าอนาล็อกของตัวมันถูกส่งต่อไปโดยไม่มีการจัดระดับ ในกรณีของช่องสัญญาณสื่อสารแบบเกาส์เซียนนั้น สำหรับค่าระดับอ้างอิง เราจะต้องเลือกค่าให้เหมาะสม ซึ่งเราพิจารณาได้ดังนี้

หากเราให้สัญญาณที่รับได้ทางด้านรับคือ $r_0 = s_0 + n_0$ โดยค่า n_0 คือ ตัวแปรสุ่มแบบเกาส์เซียนที่มีค่าเฉลี่ยอยู่ที่ศูนย์ และ s_0 นั้นเป็นค่าคงที่ ซึ่งขึ้นอยู่กับลักษณะของสัญญาณที่ส่ง นั่นคือ





รูปที่ 3.9 แสดงค่าความน่าจะเป็นของความผิดพลาด

ดังนั้นสัญญาณ r_0 ที่ได้ คือ ตัวแปรสุ่มแบบเกาส์เซียนที่มีค่าเฉลี่ยอยู่ที่ s_{01} หรือ s_{02} ดังแสดงในรูปที่ 3.9 โดยค่าเฉลี่ย(Mean) ของ r_0 คือ $m_{r01} = s_{01}$ เมื่อข้อมูล “1” ถูกส่ง และค่าเฉลี่ย(Mean) ของ r_0 คือ $m_{r02} = s_{02}$ เมื่อข้อมูล “0” ถูกส่ง ดังนั้นค่าพีดีเอฟแบบมีเงื่อนไข คือ

$$f\left\langle r_o \middle| s_1 \right\rangle = \left(\frac{1}{\sqrt{2\pi\sigma_0}}\right) e^{-(r_o - s_{01})^2/(2\sigma_0^2)} \qquad (3.4)$$

และ

$$f\left\langle r_o \middle| s_2 \right\rangle = \left(\frac{1}{\sqrt{2\pi\sigma_0}}\right) e^{-(r_o - s_{02})^2/(2\sigma_0^2)} \qquad (3.5)$$

$\sigma_0^2 = \overline{n_0^2}$ คือค่ากำลังงานเฉลี่ยของสัญญาณรบกวนทางด้านเอาท์พุทของเครื่องรับและหากข้อมูล “0” และข้อมูล “1” จากทางด้านส่ง มีโอกาสเกิดขึ้นเท่าๆกัน (Equally likely source statistics) จากรูป 3.9 จะเห็นว่าเมื่อ $r_0 > V_T$ แสดงว่าข้อมูล ”1” ถูกส่ง และเมื่อ $r_0 < V_T$ แสดงว่าข้อมูล ”0” ถูกส่ง โดยที่ V_T คือ ค่าระดับอ้างอิง(Threshold) ความผิดพลาดทางด้านรับนี้จะเกิดขึ้นเมื่อ $r_0 < V_T$ ถ้าข้อมูล “1” ถูกส่ง หรือบริเวณพื้นที่แรเงาด้านซ้ายมือ

$$P\left\langle error \middle| s_1\ sent \right\rangle = \int_{-\infty}^{V_T} f\left\langle r_o \middle| s_1 \right\rangle dr_o \qquad (3.6)$$

และ เมื่อ $r_0 > V_T$ ถ้าข้อมูล “0” ถูกส่ง หรือบริเวณพื้นที่แรเงาด้านขวามือ

$$P\langle error|s_{2sent}\rangle = \int_{-\infty}^{V_T} f\langle r_o|s_2\rangle dr_o \tag{3.7}$$

เพราะฉะนั้นค่า BER คือ

$$P_e = P\langle error|s_{1sent}\rangle P(s_{1sent}) + P\langle error|s_{2sent}\rangle P(s_{2sent}) \tag{3.8}$$

และจากความสัมพันธ์ของค่าความน่าจะเป็น

$$P(E) = \sum_{i=1}^{2} P(E, s_i) = \sum_{i=1}^{2} P\langle E|s_i\rangle P(s_i) \tag{3.9}$$

จะได้

$$P_e = P(s_{1sent}) \int_{-\infty}^{V_T} f\langle r_o|s_1\rangle dr_o + P(s_{2sent}) \int_{-\infty}^{V_T} f\langle r_o|s_2\rangle dr_o \tag{3.10}$$

และเมื่อแทน (3.4), (3.5) ลงในสมการ (3.10) เราจะได้ว่า

$$P_e = \tfrac{1}{2} \int_{-\infty}^{V_T} \left(\frac{1}{\sqrt{2\pi}\sigma_0}\right) e^{-(r_0 - s_{01})^2/2\sigma_0^2} dr_o + \tfrac{1}{2} \int_{V_T}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_0} e^{-(r_0 - s_{02})^2/(2\sigma_0^2)} dr_o \tag{3.11}$$

ซึ่งสามารถลดรูปได้เป็น

$$P_e = \tfrac{1}{2} \int_{(V_T - s_{01})/\sigma_0}^{\infty} \left(\frac{1}{\sqrt{2\pi}}\right) e^{-(\lambda^2/2)} d\lambda + \tfrac{1}{2} \int_{(V_T - s_{02})/\sigma_0}^{\infty} \left(\frac{1}{\sqrt{2\pi}}\right) e^{-(\lambda^2/2)} d\lambda$$

ซึ่งค่า $\lambda = -(r_0 - s_{01}) / (\sigma_0)$ ในส่วนอินทิเกรตพจน์แรก และ $\lambda = -(r_0 - s_{02}) / (\sigma_0)$ ในส่วนอินทิเกรตพจน์หลังหรือ

$$P_e = \tfrac{1}{2} Q\left(\frac{-V_T + s_{01}}{\sigma_0}\right) + \tfrac{1}{2} Q\left(\frac{V_T - s_{02}}{\sigma_0}\right) \tag{3.12}$$

โดยการเลือกค่าที่ V_T เหมาะสม เราสามารถทำให้ค่าความน่าจะเป็นในการเกิดความผิดพลาด มีค่าน้อยที่สุดได้ ซึ่งการที่จะหา V_T นี้ เราจะต้องทำให้ $dP_e / dV_T = 0$ โดยใช้กฎของ Leibniz กับสมการที่(3.11)

จะได้ว่า

$$\frac{dP_e}{dV_T} = \frac{1}{2}\left(\frac{1}{\sqrt{2\pi}\sigma_0}\right)e^{-(V_T-s_{01})^2/(2\sigma_0^2)} - \frac{1}{2}\left(\frac{1}{\sqrt{2\pi}\sigma_0}\right)e^{-(V_T-s_{02})^2/(2\sigma_0^2)} = 0$$

หรือ $$e^{-(V_T-s_{01})^2/(2\sigma_0^2)} = e^{-(V_T-s_{02})^2/(2\sigma_0^2)}$$

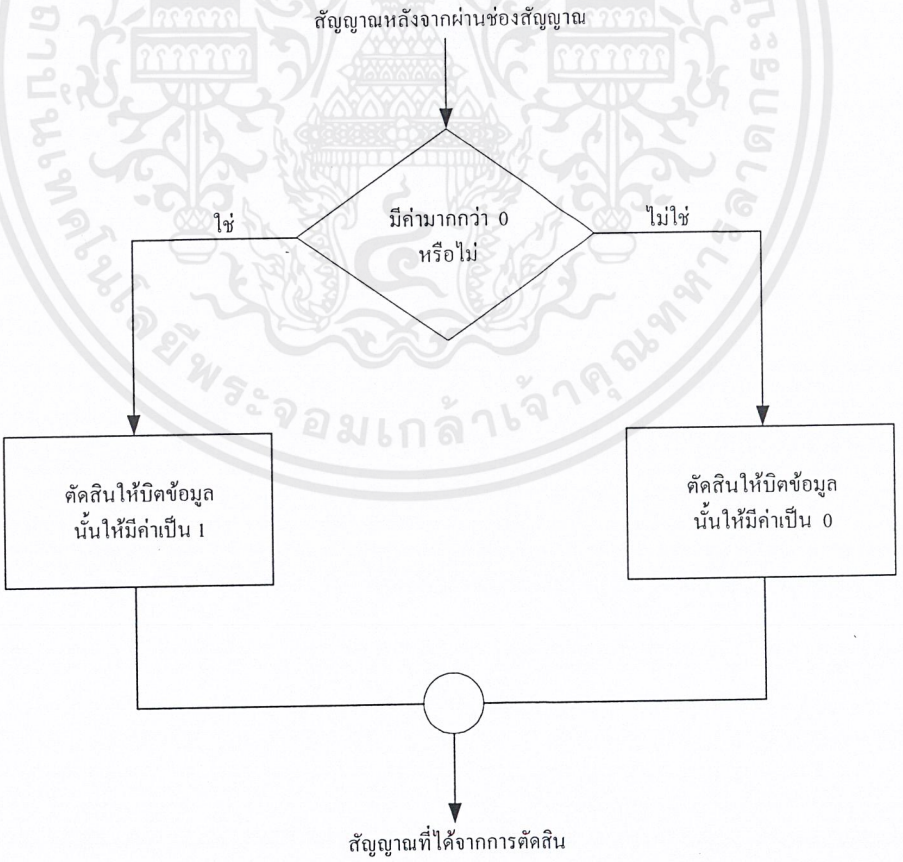
ซึ่งทำให้ทราบว่า $$(V_T - s_{01})^2 = (V_T - s_{02})^2$$

เพราะฉะนั้น การจะทำให้ค่า P_e มีค่าน้อยที่สุดนั้น เราจะต้องเลือกค่าระดับอ้างอิงดังนี้

$$\boxed{V_T = \frac{(s_{01} + s_{02})}{2}} \qquad (3.13)$$

ดังนั้นจากการกำหนดให้ค่า $s_{01} = 1$ และ $s_{02} = -1$ เราจะได้ค่าระดับอ้างอิง $V_T = 0$

และการทำงานของโปรแกรมในส่วนการตัดสินระดับสัญญาณ สามารถแสดงได้ดังรูปที่ (3.10)

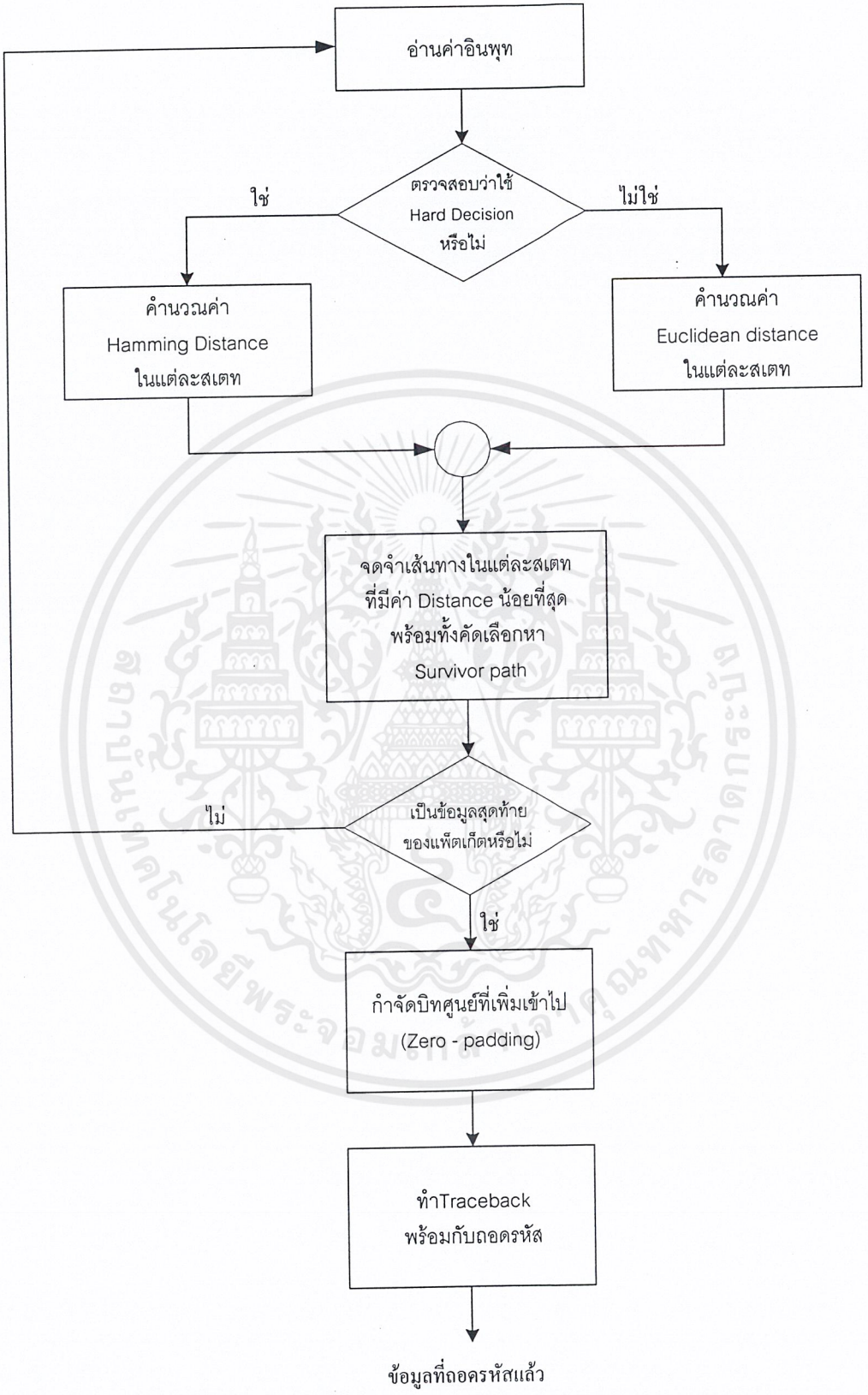


รูปที่ 3.10 แสดงโฟรชาร์ต(Flow Chart) ของโปรแกรมในส่วนตัดสินระดับสัญญาณ

แบบ Hard Decision

3.2.7 ส่วนถอดรหัสการประสาน

โปรแกรมจะใช้ฟังก์ชัน Decode (.) ในการถอดรหัส ซึ่งใช้เทคนิคของไวเทอร์บี โดยรับค่าเอาท์พุทจากส่วนตัดสินระดับสัญญาณ และใช้ ตาราง Trellis ที่ถูกสร้างขึ้นในการถอดรหัส โดยขั้นตอนแรกจะเริ่มจากที่สเตทศูนย์และเริ่มเลือกเส้นทางที่มีระยะห่างแฮมมิ่งสำหรับ Hard Decision หรือระยะห่างอูคลิดีน สำหรับ Soft Decision ที่น้อยที่สุด โดยแต่ละครั้งค่าระยะทางและเส้นทางจะถูกจดจำอย่างต่อเนื่องและเลือกเส้นทางที่อยู่รอด (Survival Path) จนกระทั่งสิ้นสุดข้อมูลลำดับสุดท้ายในแต่ละแพ็คเก็ต และจากตัวเข้ารหัสนั้นข้อมูลจะถูกทำการเติมบิตศูนย์หรือ Zero Padding เข้าไปด้วย เพื่อให้แน่ใจว่าเมื่อสิ้นสุดในแต่ละกระบวนการแล้ว มันจะกลับมาที่สเตทศูนย์อีกครั้ง ตัวถอดรหัสเองจะต้องใช้ Backward Trellis Table เพื่อทำการเลือกเส้นทางย้อนกลับ (Trace Back) ตามเส้นทางที่อยู่รอด (Survival Path) ที่มีค่าระยะห่างน้อยที่สุด และสำหรับการเติมบิตศูนย์เพิ่มเติมทางด้านส่งนั้น ตัวถอดรหัสเองก็จะต้องทำการกำจัดทิ้งทางด้านรับด้วย ซึ่งขั้นตอนของโปรแกรมแสดงได้ดังรูปที่ 3.11



รูปที่ 3.11 แสดงโฟรชาร์ต(Flow Chart) ของโปรแกรมในส่วนถอดรหัสการประสาน

จากที่กล่าวไว้ข้างต้นเส้นทางที่มาถึงในแต่ละโหนดมี 2 เส้นทาง ซึ่งจะมีเส้นทางหนึ่งที่ถูกกำจัดทิ้งและจะได้เส้นทางที่เหลือซึ่งเมื่อเวลาผ่านไปได้ช่วงหนึ่ง เราจะสามารถทำการถอดรหัสแรกๆได้ ในการถอดรหัสนั้นเราจะทำการที่เรียกว่า Traceback ซึ่งจะพิจารณาเลือกเส้นทางย้อนกลับโดยพิจารณาเส้นทางที่มีค่าเมตริกซ์น้อยที่สุด แล้วทำการอ้างอิงกับตาราง Trellis หรือตารางแสดงสถานะของรีจีสเตอร์เพื่อดูค่าเอาท์พุทที่สอดคล้องทำให้ได้ลำดับเอาท์พุทที่ต้องการ โดยเวลาที่ผ่านไปช่วงหนึ่งๆใดๆที่เริ่มทำกระบวนการเลือกเส้นทางย้อนกลับ (Traceback) นี้ เรียกว่า Traceback Depth ซึ่งโดยปกติค่า Traceback Depth ที่เหมาะสมต่อการถอดรหัสนั้นไม่ควรมีค่าต่ำกว่า M x 5 เมื่อให้ M เป็นความยาวคอนสเตร็นท์

3.2.8 ส่วนเปรียบเทียบหาอัตราความผิดพลาด

หลังจากผ่านส่วนการถอดรหัสการประสานด้วยวิธีไวเทอร์บีแล้ว ก็จะได้ข้อมูลที่ถูกกู้หรือประเมินค่ากลับคืนมา ซึ่งเราจะนำข้อมูลดังกล่าวนี้ไปเปรียบเทียบบิทต่อบิท กับข้อมูลทางด้านส่งที่สุ่มขึ้นมาว่ามีจำนวนแตกต่างกันเท่าใด โดยโปรแกรมจะทำการคำนวณค่าอัตราความผิดพลาดของข้อมูล (Bit Error Rate) ที่แต่ละค่าของความหนาแน่นพลังงานต่อบิต (Eb/No) ซึ่งมีค่าตั้งแต่ 0 dB ถึง 6 dB โดยขั้นตอนการหาความน่าจะเป็นของความผิดพลาดนั้น เราจะนำข้อมูลที่ได้หลังจากการถอดรหัสแล้วที่ด้านรับมาเปรียบเทียบกับข้อมูลที่ทางด้านส่ง เพื่อหาจำนวนบิตผิดพลาดทั้งหมด ซึ่งค่า BER หาได้จาก

$$BER = \frac{\text{จำนวนบิตผิดพลาด}}{\text{จำนวนบิตทั้งหมดที่ใช้ในการทดสอบ}} \qquad (3.14)$$

3.2.9 ส่วนแสดงผล

เมื่อได้ค่าอัตราความผิดพลาด (BER) ในแต่ละค่าของ Eb/No แล้วโปรแกรมก็จะถามผู้ใช้ว่าต้องการที่จะใช้ฟังก์ชัน **Paste&Plot** หรือไม่ โดยส่วนของฟังก์ชันนี้โปรแกรม Visual C++ จะทำการเปิดโปรแกรม MATLAB ขึ้นมาอัตโนมัติ และเมื่อหน้าต่างรับคำสั่งของโปรแกรม MATLAB ปรากฏขึ้นมา เราก็เพียงคลิกเมาส์ปุ่มขวาเพื่อเลือกใช้การแปะวางหรือ **Paste** แล้วกด Enter โปรแกรม MATLAB ก็จะนำข้อมูลตัวเลขต่างๆที่ได้จากการจำลองเหตุการณ์ ไปพล็อต (**Plot**) กราฟเพื่อแสดงประสิทธิภาพของการเข้ารหัสการประสานออกทางหน้าจอ

3.3 ส่วนประยุกต์ใช้งาน (PA:Practical Application)

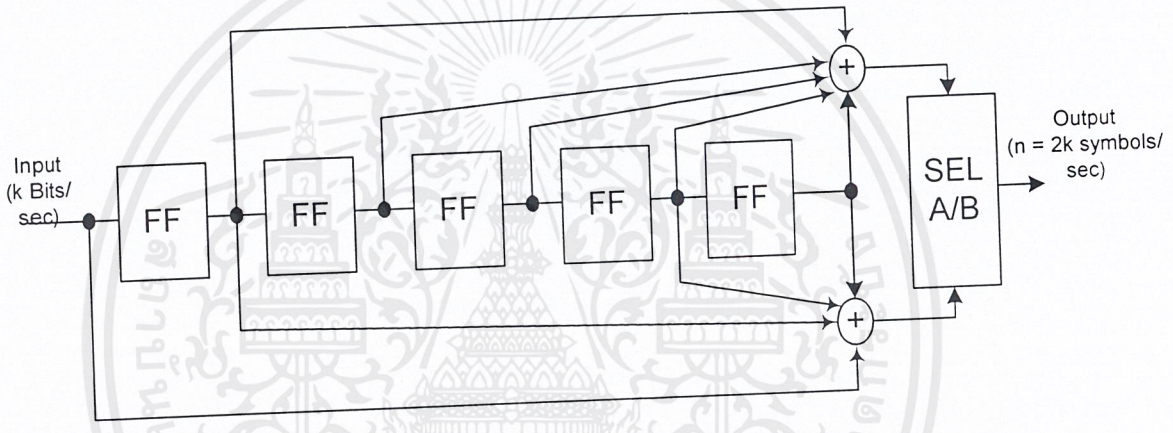
สำหรับส่วนนี้จะเป็นการนำโปรแกรมไปประยุกต์ใช้งานจริง จากรูปที่ 3.9 บริเวณกรอบเส้นประจะเห็นว่าประกอบไปด้วยโปรแกรม 3 ส่วน คือ

1. ส่วนเข้ารหัส-ถอดรหัส(Encoder - Decoder)
2. ส่วนรับ – ส่งข้อมูลผ่านพอร์ตอนุกรม(Serial Communication)
3. ส่วนเปรียบเทียบหาอัตราความผิดพลาด(BER)

โดยรายละเอียดของส่วนต่างๆ แสดงได้ดังนี้

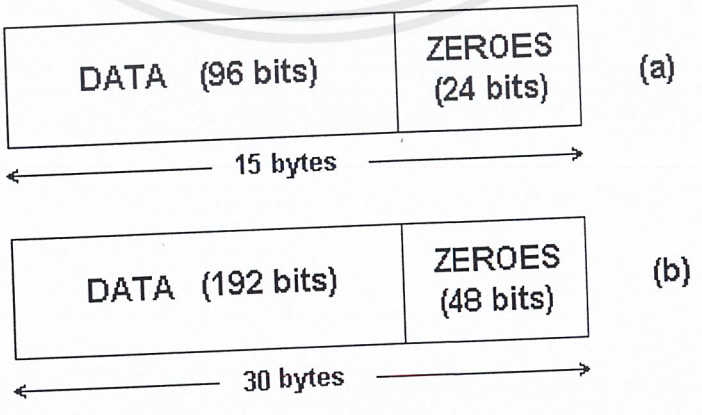
3.3.1 ส่วนเข้ารหัส – ถอดรหัส(Encoder – Decoder)

ส่วนนี้จะเป็นส่วนในการใช้งานหลัก โดยโปรแกรมจะทำการเปิดไฟล์ข้อมูลใดๆ เพื่อทำการอ่านสำหรับการเข้ารหัสหรือถอดรหัส โดยข้อมูลที่ถูกอ่านเข้ามานี้จะถูกแยกออกเป็นแพคเก็ต(Packets) ซึ่งข้อมูลนี้จะถูกกระทำการเข้ารหัสหรือถอดรหัสทีละแพคเก็ต โดยโครงสร้างของตัวเข้ารหัสที่ใช้นั้นแสดงในรูปที่ 3.12 ซึ่งมีอัตราการเข้ารหัสที่ ½ เมื่อดำเนินการเสร็จข้อมูลที่เป็นผลลัพธ์ก็จะถูกเก็บบันทึกลงไฟล์ใหม่ คือ ไฟล์ที่เข้ารหัสช่องสัญญาณ(Encoded file) หรือ ไฟล์ที่ถอดรหัสช่องสัญญาณแล้ว (Decoded file) สำหรับโครงสร้างของแพคเก็ตข้อมูลนั้น จะสามารถแสดงได้ในรูปที่ 3.13



รูปที่ 3.12 แสดงโครงสร้างตัวเข้ารหัสที่ใช้งานจริง โดยมีค่า Generator Sequence เท่ากับ

$G(0) = 59_8$, $G(1) = 65_8$



รูปที่ 3.13 (a) แสดงโครงสร้างแพคเก็ตข้อมูลก่อนทำการเข้ารหัสการประสาน

(b) แสดงโครงสร้างแพคเก็ตข้อมูลหลังทำการเข้ารหัสการประสาน

จากรูปที่ 3.13 (a) จะเห็นว่าจำนวนข้อมูลในแต่ละแพคเก็ตก่อนทำการเข้ารหัสมี 96 บิท และตามด้วยบิทศูนย์ต่อท้ายอีก 24 บิท หรือการทำ Zero-Padding เพื่อเคลียร์ยสถานะของรีจิสเตอร์ก่อนการที่จะเริ่มทำงานในแพคเก็ตต่อไป จากรูปที่ 3.13(b) จำนวนข้อมูลและบิทศูนย์ในแต่ละแพคเก็ตจะเพิ่มขึ้นเป็น 2 เท่าหลังจากผ่านการเข้ารหัสการประสาน ทั้งนี้เพราะเราใช้อัตราการเข้ารหัส (Code Rate) ที่ ½ ทำให้จำนวนบิททั้งหมดในแต่ละแพคเก็ตหลังการเข้ารหัสเท่ากับ 240 บิท ดังนั้นทางด้านรับหรือตัวถอดรหัสก็จะอ่านข้อมูลมาทำการถอดรหัสทีละ 240 บิทต่อแพคเก็ต

สำหรับการอ่านไฟล์เข้ามาที่มีขนาดแน่นอนในแต่ละแพคเก็ตปกตินั้น เนื่องจากขนาดของไฟล์ข้อมูลใดๆมีขนาดไม่เท่ากัน ดังนั้นจะเหลือเศษของข้อมูลซึ่งมีค่าน้อยกว่า 96 บิทในแพคเก็ตสุดท้ายจึงทำให้แพคเก็ตสุดท้ายนี้มีขนาดที่ไม่แน่นอน คือเปลี่ยนตามขนาดของข้อมูลที่เหลือจากการจัดแพคเก็ตปกติ

3.3.2 ส่วนรับ – ส่งข้อมูลผ่านพอร์ตอนุกรม(Serial Communication)

โปรแกรมส่วนนี้จะทำหน้าที่ในการรับส่งข้อมูลจากไฟล์ใดๆ ในหน้าต่าง MS-DOS ผ่านพอร์ตอนุกรม(พอร์ต COM1 เท่านั้น) โดยรายละเอียดข้อตกลงระหว่างการส่ง – รับข้อมูลแสดงในตารางที่ 3.1

ตารางที่ 3.1 แสดงรูปแบบการส่ง – รับข้อมูลผ่านพอร์ตอนุกรม

อัตราการส่งสัญญาณ (Baud Rate)	9600 สัญลักษณ์ต่อวินาที (Symbol/Second)
จำนวนบิทข้อมูล	8 บิท
จำนวน Stop Bit	1 บิท
พาริตี้บิท	ไม่มี
แฮนด์เช็คกิ้ง (HandShaking)	ซอฟท์แวร์
พอร์ทติดต่อ	COM 1 เท่านั้น

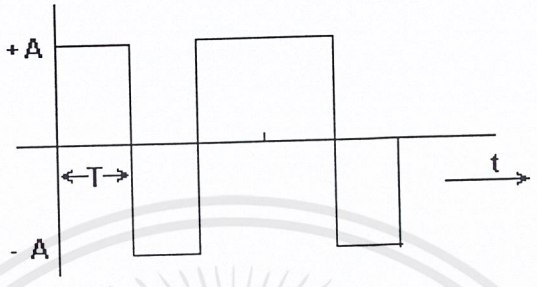
การส่ง – รับข้อมูลนี้จะต้องรันโปรแกรมด้านส่งก่อน คือด้านส่งจะต้องมีข้อความ “Transmitter is waiting…….” หลังจากนั้นค่อยรันโปรแกรมทางด้านรับ โดยโปรแกรมด้านส่งจะรอคำสั่งพร้อมที่มาจากทางด้านรับ เมื่อด้านรับส่งการติดต่อครั้งแรกมา(Hit the first contact) ขั้นตอนในการส่ง – รับข้อมูลก็จะเริ่มขึ้น โดยด้านส่งจะส่งขนาดของไฟล์ข้อมูลไปก่อนหลังจากนั้นก็เริ่มส่ง – รับข้อมูลทีละไบต์ผ่านพอร์ตอนุกรม(COM1) ซึ่งระหว่างการส่ง – รับข้อมูลจะมีการทำแฮนด์เช็คกิ้งตลอดเวลา

3.3.3 ส่วนเปรียบเทียบหาอัตราความผิดพลาด (BER)

โปรแกรมส่วนนี้จะทำการเปิดไฟล์ใดๆ 2 ไฟล์ นำข้อมูลมาเทียบกันบิทต่อบิทเพื่อดูจำนวนบิททั้งหมดที่แตกต่างกันว่ามีเท่าใด ซึ่งจะนำค่านี้ไปหารด้วยขนาดของไฟล์ที่เปิด จะได้ค่า BER ตามสมการที่ (3.14) โดยทั่วไปแล้วไฟล์ที่จะนำมาเทียบกันคือ ไฟล์ข้อมูลทางด้านส่ง(Source file) กับไฟล์ที่ถอดรหัสการประสานทางด้านรับแล้ว(Decoded file)

3.3.4 การหาค่าอัตราส่วนสัญญาณพลังงานต่อความหนาแน่นสัญญาณรบกวนใน 1 บิท(Eb/No)

เราจะหาค่าอัตราส่วน (Eb/No) นี้เพื่อใช้ในการวิเคราะห์ประสิทธิภาพการเข้ารหัสการประสาน ซึ่งจะเห็นผลการทดลองได้ในบทที่ 4 ในการส่ง-รับข้อมูลผ่านพอร์ตอนุกรมตามมาตรฐาน RS-232 นั้น ระดับของสัญญาณไฟฟ้าที่ออกจากพอร์ตอนุกรมจะมีลักษณะเป็นแบบสัญญาณมีขั้ว (Polar Signaling) ดังแสดงในรูปที่ 3.14



รูปที่ 3.14 แสดงลักษณะของสัญญาณไฟฟ้าแบบมีขั้ว(Polar Signaling)

ซึ่งจากรูปเราสามารถอธิบายลักษณะคลื่นในเชิงคณิตศาสตร์ได้เป็น

$$S_1(t) = +A \quad , \quad 0 < t \leq T \tag{3.15}$$

$$S_2(t) = -A \quad , \quad 0 < t \leq T \tag{3.16}$$

โดยสัญญาณมีขั้วจะเป็นแบบขั้วตรงข้าม(Antipodal signal)เมื่อ $S_1(t) = -S_2(t)$ และหากเราให้ทางเครื่องรับมีลักษณะแบบแมทช์ฟิลเตอร์(Matched Filter) ดังนั้นเราสามารถหาค่าโอกาสผิดพลาดของการตัดสินสัญญาณได้ดังนี้

$$P_e = Q\left(\sqrt{\frac{2A^2T}{N_o}}\right) = Q\left(\sqrt{2\left[\frac{E_b}{N_o}\right]}\right) \tag{3.17}$$

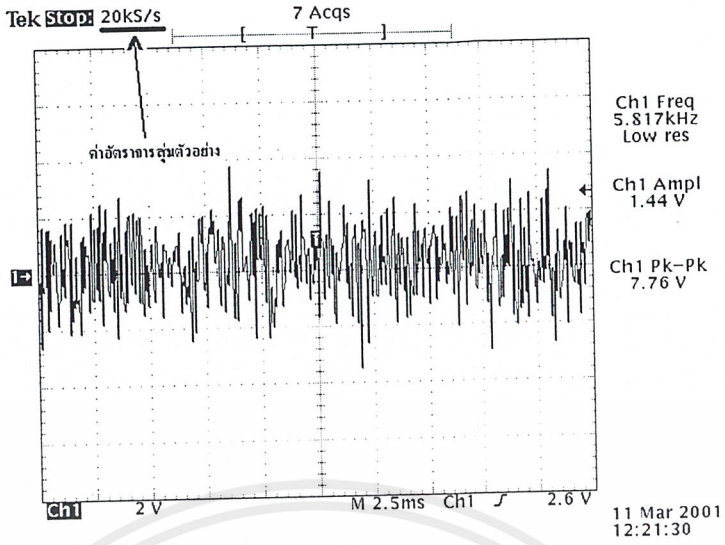
โดยค่าพลังงานต่อบิทเฉลี่ยคือ $E_b = A^2T$ (3.18)

หรือ $E_b = A^2/R$ (3.19)

เมื่อ R คืออัตราการส่งผ่านข้อมูล

และเมื่อคิดในหน่วยเดซิเบลจะได้ว่า $E_{b\,(dB)} = 10\log_{10}(E_b)$ (3.20)

เพราะฉะนั้นหากกำหนดให้ค่า No มีค่าคงที่แล้ว เราสามารถเปลี่ยนแปลงอัตราส่วน Eb/No ได้โดยการเปลี่ยนค่า Eb ซึ่งจะสัมพันธ์กับการเปลี่ยนค่า A ในสมการที่ (3.19) จากการใช้งานจริงเราจะกำหนดให้ค่า R เท่ากับ 9600 Symbols/Sec ส่วนค่า No นั้นคือค่าความหนาแน่นกำลังเชิงสเปกตรัมของสัญญาณรบกวน หรือ พีเอสดีของสัญญาณรบกวน โดยในการทดลองนั้นเราจะนำสัญญาณรบกวนหรือค่าตัวแปรสุ่มมาจากเครื่องกำเนิดสัญญาณ(Function Generator) ในโหมด Noise แล้วใช้สโคปจับบันทึกสัญญาณแบบไฟล์สเปรดชีท(Spread Sheet)จำนวน 10 ครั้ง โดยแต่ละครั้งจะใช้อัตราการสุ่มค่าตัวอย่างที่ 20 KS/s ดังรูปที่ 3.15



รูปที่ 3.15 แสดงค่าอัตราการสุ่มตัวอย่าง

เราใช้ค่าอัตราการสุ่มตัวอย่างที่ 20KS/s เพราะหากเราพิจารณาที่อัตราการส่งสัญญาณซึ่งมีค่าเท่ากับ 9600 Symbols/Sec และจากตารางที่ 3.2

ตารางที่ 3.2 แสดงค่าแบนด์วิทของรูปแบบสัญญาณชนิดต่างๆ เมื่อ R คืออัตราการส่งผ่านสัญญาณ

Code Type	First - Null Bandwidth (Hz)
Unipolar NRZ	R
Polar NRZ	R
Unipolar RZ	2R
Bipolar RZ	R
Manchester NRZ	2R

จากตารางจะเห็นว่าค่า First – Null Bandwidth ของสัญญาณแบบ Polar NRZ มีค่าเท่ากับอัตราการส่งผ่านสัญญาณ ดังนั้นเราสามารถประมาณค่าแบนด์วิทของสัญญาณที่ส่งได้เท่ากับ 9,600 เฮิร์ต หรือประมาณ 10,000 เฮิร์ต เพราะฉะนั้นจากทฤษฎีการสุ่มตัวอย่าง(Sampling Theory) เราจึงใช้ค่าอัตราการสุ่มตัวอย่างของสัญญาณรบกวนเป็นสองเท่าของแบนด์วิทของสัญญาณที่ส่ง คือ 2 x 10,000 = 20 KHz หรือ 20 KS/s ซึ่งก็คือค่าประมาณของแบนด์วิทของสัญญาณรบกวนที่มีผลต่อระบบหรือสัญญาณที่ส่งจากด้านส่งไปยังด้านรับด้วยอัตราการส่งผ่าน 9,600 Symbols/Sec

จากการเก็บตัวอย่างของสัญญาณสุ่มทั้ง 10 ครั้งนี้ เราจะนำค่าทั้งหมดไปวิเคราะห์โดยใช้โปรแกรม MATLAB เพื่อหาค่าฮิสโทรแกรม(Histogram), ค่าเฉลี่ย , ค่าเบี่ยงเบนมาตรฐาน รวมทั้งค่าประมาณพีเอสดีของสัญญาณสุ่ม ซึ่งค่าประมาณพีเอสดีของสัญญาณสุ่มหรือที่เรียกว่า Periodogram นี้สามารถหาได้จากความสัมพันธ์ดังสมการที่ (3.21)

$$P_T(f) = |X_T(f)|^2 / T \tag{3.21}$$

เมื่อ T คือช่วงเวลาของการสังเกต(Observation Interval)จากสัญญาณสุ่ม X(t)

ซึ่งค่าเฉลี่ยพีเอสดีของสัญญาณสุ่มจากตัวอย่างทั้ง 10 ไฟล์นี้ก็คือ No ที่เราต้องการหา และเมื่อคิดในหน่วยเดซิเบลจะได้

$$N_{o\,(dB)} = 10\log_{10}(N_o) \tag{3.22}$$

เพราะฉะนั้นอัตราส่วน Eb/No ในหน่วยเดซิเบลคือ

$$(Eb/No)_{dB} = E_{b\,(dB)} - N_{o\,(dB)} \tag{3.23}$$

บทที่ 4

การทดลองและผลการทดลอง

สำหรับผลการทดลองเราสามารถแบ่งออกได้เป็น 2 ส่วน คือ ผลการทดลองจากการจำลองเหตุการณ์(Simulation) และผลการทดลองผ่านสภาพแวดล้อมจริง

4.1 แสดงผลการทดลองจากการจำลองเหตุการณ์ (Simulation)

4.1.1 แสดงตัวอย่างของตาราง Trellis

เมื่อกำหนดค่า R = ½ , K=3 และ ค่า Generator Sequence เท่ากับ $G(0) = 7_8$, $G(1) = 5_8$

**********forward trellis look-up table**********

current state, input, next state, output

0, 0, 0, 00
0, 1, 2, 11
1, 0, 0, 11
1, 1, 2, 00
2, 0, 1, 10
2, 1, 3, 01
3, 0, 1, 01
3, 1, 3, 10

**********Backward trellis look-up table**********

current state, input, previous state, output

0, 0, 0, 00
0, 0, 1, 11
1, 0, 2, 10
1, 0, 3, 01
2, 1, 0, 11
2, 1, 1, 00
3, 1, 2, 01
3, 1, 3, 10

Total state : 4

4.1.2 แสดงตัวอย่างของประสิทธิภาพเชิงตัวเลขในการเข้ารหัสการประสาน

AWGN channel :using Hard decision

At Eb/N0: 0.00 dB,BER : 1.962750e-01

At Eb/N0: 0.50 dB,BER : 1.630520e-01

At Eb/N0: 1.00 dB,BER : 1.294145e-01

At Eb/N0: 1.50 dB,BER : 9.818600e-02

At Eb/N0: 2.00 dB,BER : 7.284550e-02

At Eb/N0: 2.50 dB,BER : 4.964100e-02

At Eb/N0: 3.00 dB,BER : 3.277200e-02

At Eb/N0: 3.50 dB,BER : 1.978250e-02

At Eb/N0: 4.00 dB,BER : 1.129750e-02

At Eb/N0: 4.50 dB,BER : 6.151500e-03

At Eb/N0: 5.00 dB,BER : 3.213000e-03

At Eb/N0: 5.50 dB,BER : 1.446500e-03

At Eb/N0: 6.00 dB,BER : 5.935000e-04

AWGN channel :using Soft decision(no quantization)

At Eb/N0: 0.00 dB,BER : 9.149500e-02

At Eb/N0: 0.50 dB,BER : 6.414900e-02

At Eb/N0: 1.00 dB,BER : 4.197600e-02

At Eb/N0: 1.50 dB,BER : 2.541000e-02

At Eb/N0: 2.00 dB,BER : 1.426200e-02

At Eb/N0: 2.50 dB,BER : 7.340500e-03

At Eb/N0: 3.00 dB,BER : 3.415500e-03

At Eb/N0: 3.50 dB,BER : 1.637000e-03

At Eb/N0: 4.00 dB,BER : 6.270000e-04

At Eb/N0: 4.50 dB,BER : 2.370000e-04

At Eb/N0: 5.00 dB,BER : 9.500000e-05

At Eb/N0: 5.50 dB,BER : 2.700000e-05

At Eb/N0: 6.00 dB,BER : 8.000000e-06

BSC channel :BSC(p)

At Eb/N0: 0.00 dB,p : 1.590072e-01,BER : 1.977160e-01

At Eb/N0: 0.50 dB,p : 1.450987e-01,BER : 1.639685e-01

At Eb/N0: 1.00 dB,p : 1.312818e-01,BER : 1.304775e-01

At Eb/N0: 1.50 dB,p : 1.176636e-01,BER : 9.989800e-02

At Eb/N0: 2.00 dB,p : 1.043594e-01,BER : 7.256650e-02

At Eb/N0: 2.50 dB,p : 9.149038e-02,BER : 5.013500e-02

At Eb/N0: 3.00 dB,p : 7.918023e-02,BER : 3.280850e-02

At Eb/N0: 3.50 dB,p : 6.755114e-02,BER : 1.985350e-02

At Eb/N0: 4.00 dB,p : 5.671859e-02,BER : 1.170200e-02

At Eb/N0: 4.50 dB,p : 4.678569e-02,BER : 6.297000e-03

At Eb/N0: 5.00 dB,p : 3.783725e-02,BER : 3.205000e-03

At Eb/N0: 5.50 dB,p : 2.993384e-02,BER : 1.483500e-03

At Eb/N0: 6.00 dB,p : 2.310681e-02,BER : 6.570000e-04

At Eb/No: 3.00 dB,p : 7.918023e-02,BER : 3.216800e-02

At Eb/No: 3.50 dB,p : 6.755114e-02,BER : 1.990200e-02

At Eb/No: 4.00 dB,p : 5.671859e-02,BER : 1.150500e-02

At Eb/No: 4.50 dB,p : 4.678569e-02,BER : 6.110000e-03

At Eb/No: 5.00 dB,p : 3.783725e-02,BER : 3.050500e-03

At Eb/No: 5.50 dB,p : 2.993384e-02,BER : 1.514000e-03

At Eb/No: 6.00 dB,p : 2.310681e-02,BER : 6.230000e-04

AWGN channel:BER for uncoded BPSK

At Eb/No: 0.00 dB,BER: 7.893339e-02

At Eb/No: 0.50 dB,BER: 6.731958e-02

At Eb/No: 1.00 dB,BER: 5.650457e-02

At Eb/No: 1.50 dB,BER: 4.659117e-02

At Eb/No: 2.00 dB,BER: 3.766373e-02

At Eb/No: 2.50 dB,BER: 2.978229e-02

At Eb/No: 3.00 dB,BER: 2.297754e-02

At Eb/No: 3.50 dB,BER: 1.724729e-02

At Eb/No: 4.00 dB,BER: 1.255511e-02

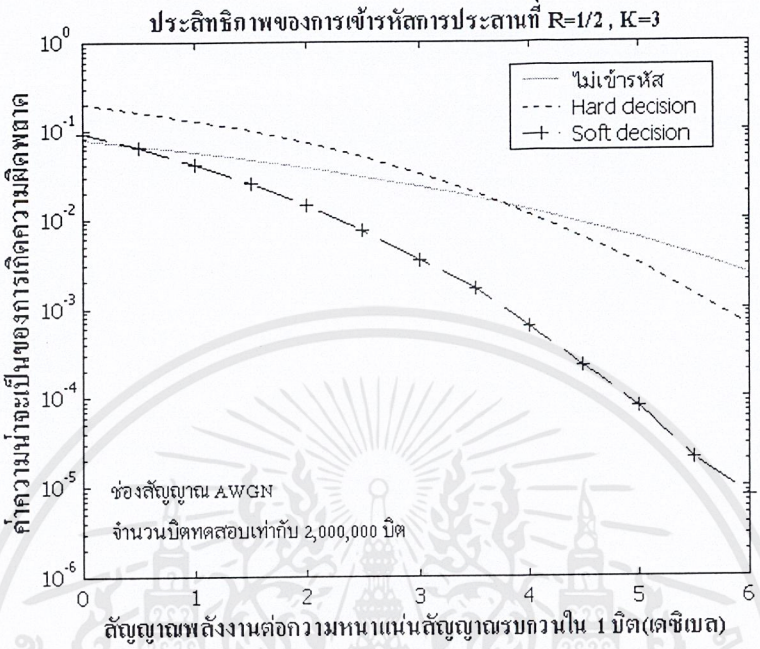
At Eb/No: 4.50 dB,BER: 8.831655e-03

At Eb/No: 5.00 dB,BER: 5.979092e-03

At Eb/No: 5.50 dB,BER: 3.878238e-03

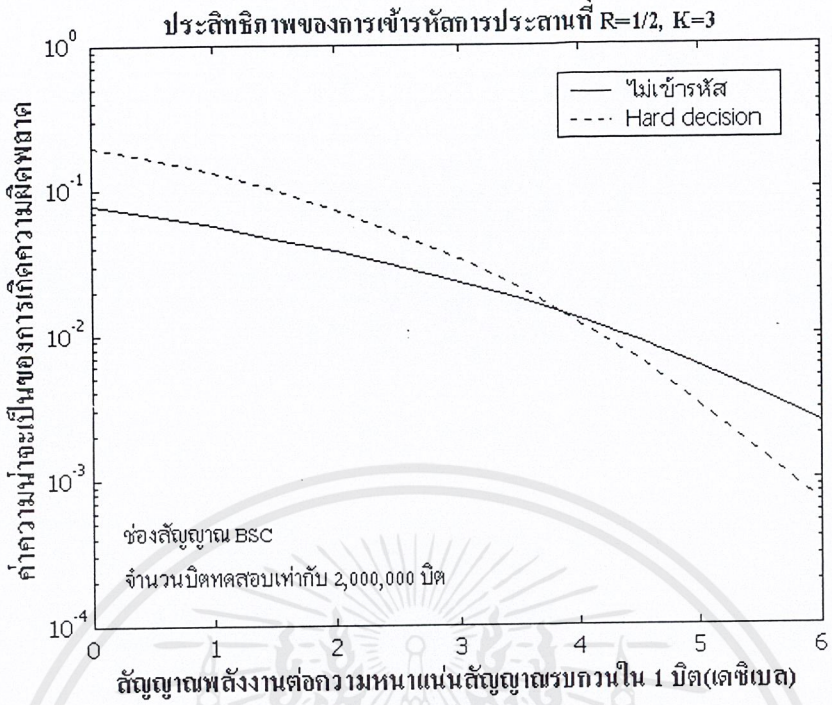
At Eb/No: 6.00 dB,BER: 2.397914e-03

4.1.3 แสดงกราฟเปรียบเทียบประสิทธิภาพของการเข้ารหัสการประสาน

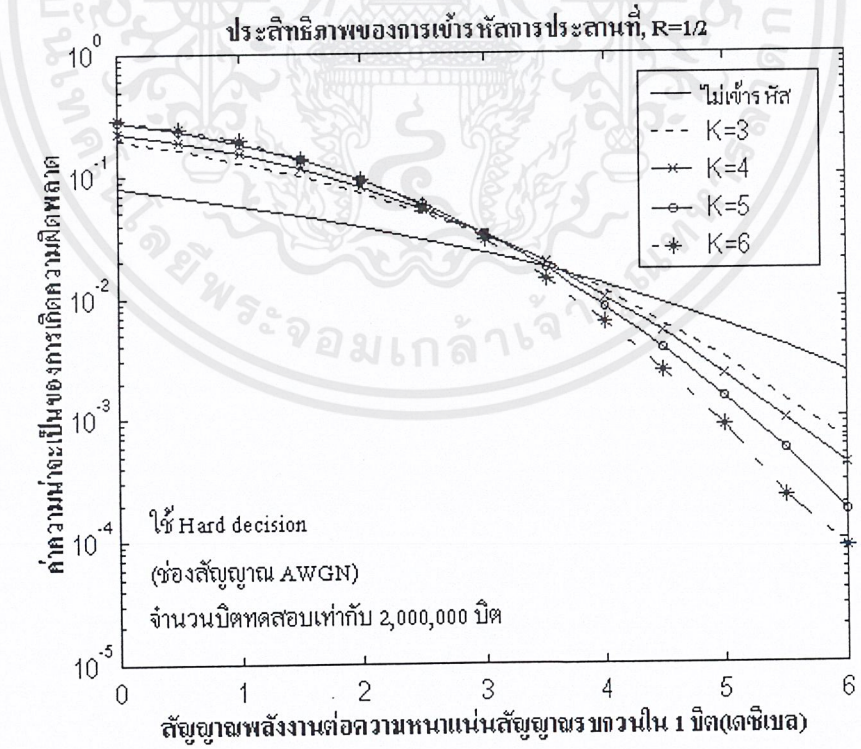


รูปที่ 4 . 1 แสดงการเปรียบเทียบประสิทธิภาพของการเข้ารหัสการประสาน เมื่อใช้การตัดสินสัญญาณทางด้านรับแบบ Hard decision และ Soft decision ผ่านช่องสัญญาณแบบ AWGN

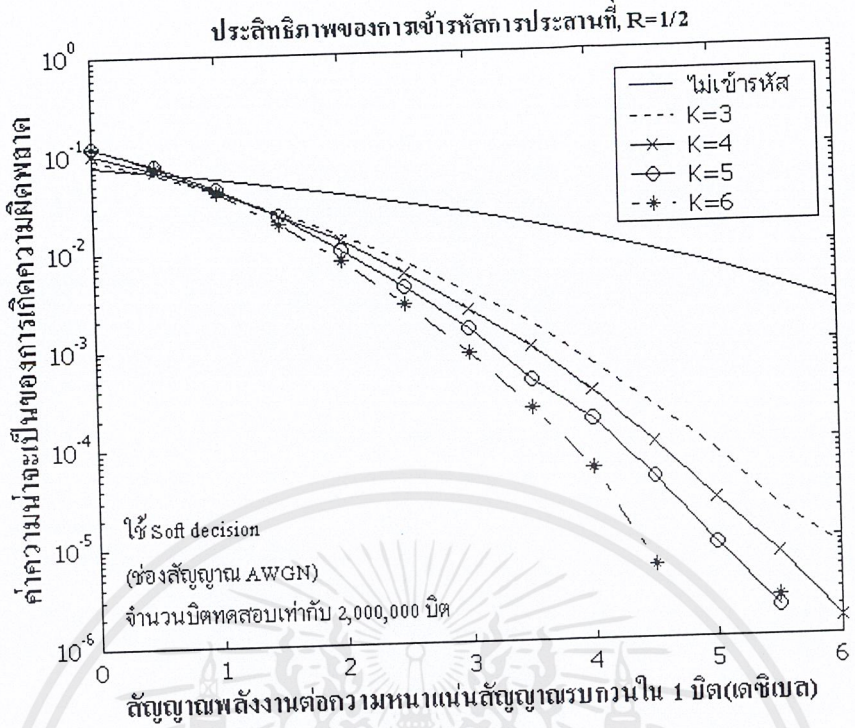
จากรูป 4 . 1 จะเห็นได้ว่าสัญญาณข้อมูลที่ทำการเข้ารหัสควบคุมความผิดพลาดนั้นจะมีประสิทธิภาพความน่าเชื่อถือที่ดีกว่าสัญญาณที่ไม่ได้เข้ารหัสควบคุมความผิดพลาด และจากรูปที่ 4 .1 นี้จะพบว่าการตัดสินสัญญาณทางด้านรับนั้น หากใช้การตัดสินแบบ Soft decision จะให้ประสิทธิภาพที่ดีกว่าการตัดสินสัญญาณแบบ Hard decision



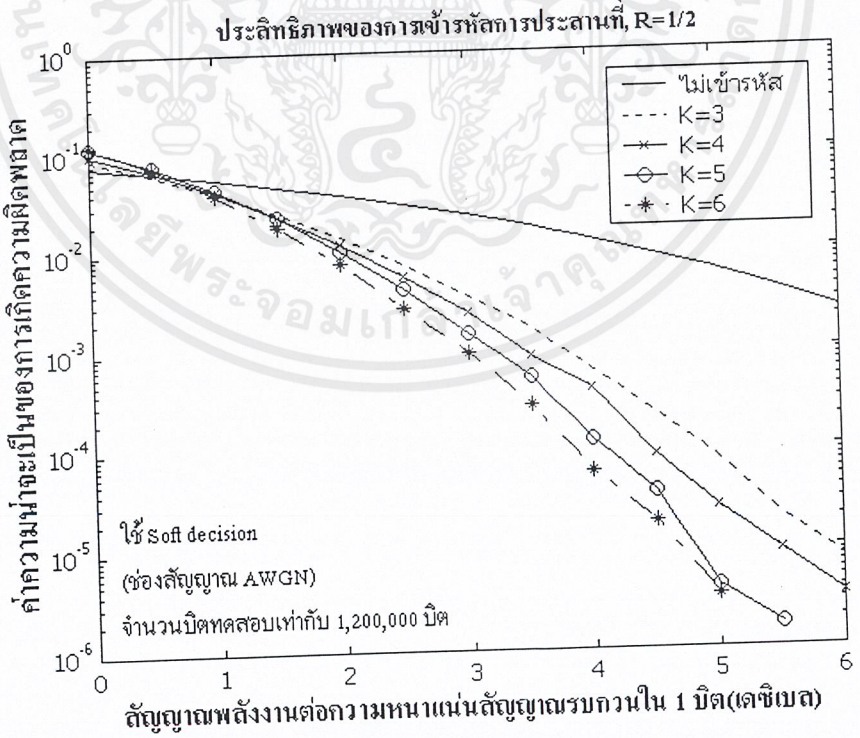
รูปที่ 4 .2 แสดงประสิทธิภาพของการเข้ารหัสการประสานเมื่อส่งสัญญาณผ่านช่องสัญญาณแบบ BSC ซึ่งจะใช้การตัดสินสัญญาณแบบ Hard decision



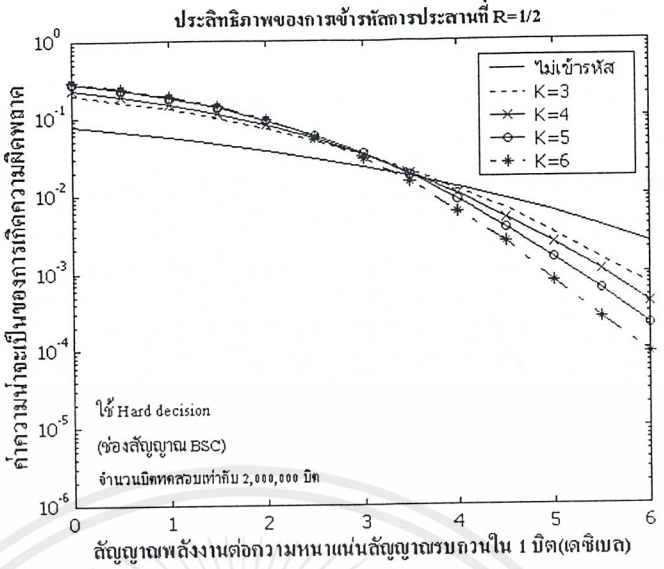
รูปที่ 4 . 3 แสดงประสิทธิภาพการเข้ารหัสการประสานที่ค่าความยาวคอนสเตรนท์(K) ต่างๆ ผ่านช่องสัญญาณ AWGN โดยใช้การตัดสินสัญญาณแบบ Hard decision



รูปที่ 4 . 4 แสดงประสิทธิภาพการเข้ารหัสการประสานที่ค่าความยาวคอนสเตรนท์(K) ต่างๆ ผ่านช่องสัญญาณ AWGN โดยใช้การตัดสินสัญญาณแบบ Soft decision

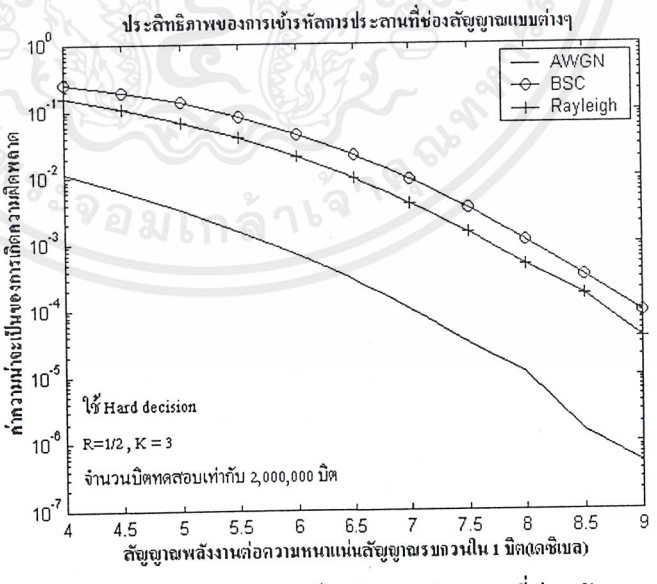


รูปที่ 4 .5 แสดงค่าความคลาดเคลื่อนของกราฟเมื่อเทียบกับรูปที่ 4 .5 เพราะใช้จำนวนบิตในการทดสอบแค่ 1,200,000 บิต

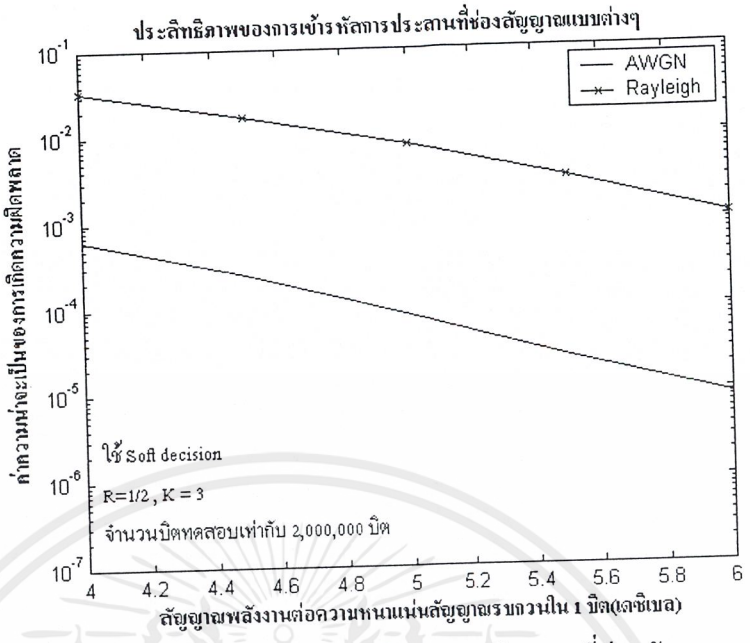


รูปที่ 4.6 แสดงประสิทธิภาพการเข้ารหัสการประสานที่ค่าความยาวคอนสเตรนท์(K) ต่างๆ ผ่านช่องสัญญาณ BSC โดยใช้การตัดสินสัญญาณแบบ Hard decision

จากรูปที่ 4.3 – 4.6 นั้นเราจะพบว่าค่าความน่าจะเป็นของการเกิดความผิดพลาด จะมีค่าน้อยลงเมื่อเราใช้การเข้ารหัสการประสานที่ค่าความยาวคอนสเตรนท์ (K) เพิ่มขึ้น นอกจากนี้จำนวนบิทข้อมูลที่ใช้ทดสอบในการจำลองเหตุการณ์ (Simulation) นั้นยังมีผลต่อกราฟที่ใช้แสดงประสิทธิภาพด้วย ซึ่งหากเราต้องการให้กราฟแสดงผลลัพธ์ที่มีความน่าเชื่อถือ เราก็จำเป็นต้องเพิ่มจำนวนบิทข้อมูลที่ใช้ในการทดสอบให้มากขึ้นด้วย



รูปที่ 4.7 แสดงประสิทธิภาพของการเข้ารหัสการประสานที่ช่องสัญญาณแบบต่างๆ เมื่อใช้การตัดสินสัญญาณแบบ Hard decision

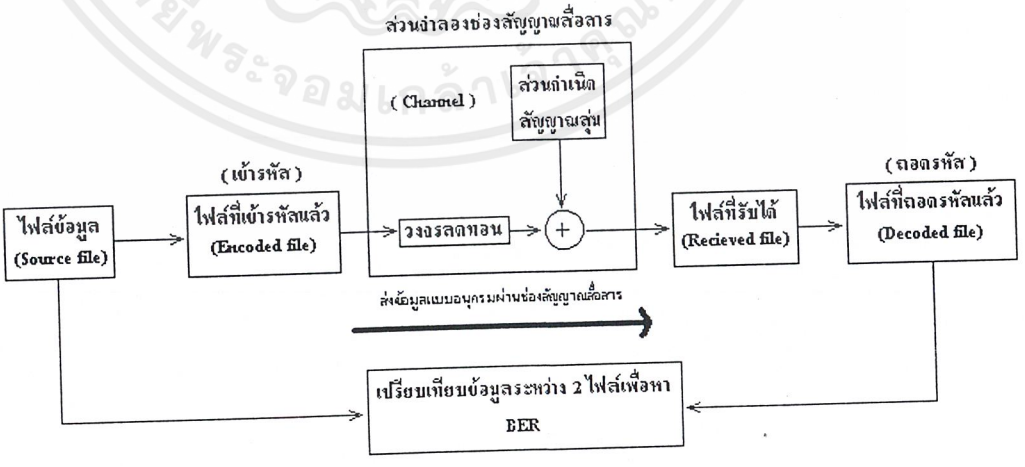


รูปที่ 4.8 แสดงประสิทธิภาพของการเข้ารหัสการประสานที่ช่องสัญญาณแบบต่างๆ เมื่อใช้การตัดสินสัญญาณแบบ Soft decision

จากรูปที่ 4.7-4.8 นั้น แสดงประสิทธิภาพของการเข้ารหัสการประสานบนช่องสัญญาณแบบต่างๆ ซึ่งเมื่อเปรียบเทียบดูแล้วจะเห็นว่า การเข้ารหัสการประสานนั้นจะให้ประสิทธิภาพที่ดีกว่าเมื่อส่งข้อมูลผ่านช่องสัญญาณแบบ AWGN หากเทียบกับช่องสัญญาณแบบ BSC และ Rayleigh

4.2 แสดงผลการทดลองผ่านสภาพแวดล้อมจริง

สำหรับการทดลองผ่านสภาพแวดล้อมจริงนี้มีขั้นตอนแสดงดังรูปที่ 4.9



รูปที่ 4.9 แสดงขั้นตอนการทดลองผ่านสภาพแวดล้อมจริง

4.2.1 แสดงตาราง Trellis

ค่าพารามิเตอร์ที่ใช้งานคือ R = ½ , K=6 Generator Sequence : G(0) = 59_8 , G(1) = 65_8

**********Forward trellis look-up table*********

**********At constraint length 6**********

current state,	input,	next state,	output
0,	0,	0,	0 0
0,	1,	16,	1 1
1,	0,	0,	1 1
1,	1,	16,	0 0
2,	0,	1,	1 0
2,	1,	17,	0 1
3,	0,	1,	0 1
3,	1,	17,	1 0
4,	0,	2,	1 1
4,	1,	18,	0 0
5,	0,	2,	0 0
5,	1,	18,	1 1
6,	0,	3,	0 1
6,	1,	19,	1 0
7,	0,	3,	1 0
7,	1,	19,	0 1
8,	0,	4,	1 0
8,	1,	20,	0 1
9,	0,	4,	0 1
9,	1,	20,	1 0
10,	0,	5,	0 0
10,	1,	21,	1 1
11,	0,	5,	1 1
11,	1,	21,	0 0
12,	0,	6,	0 1
12,	1,	22,	1 0
13,	0,	6,	1 0
13,	1,	22,	0 1

14,	0,	7,	1 1
14,	1,	23,	0 0
15,	0,	7,	0 0
15,	1,	23,	1 1
16,	0,	8,	0 1
16,	1,	24,	1 0
17,	0,	8,	1 0
17,	1,	24,	0 1
18,	0,	9,	1 1
18,	1,	25,	0 0
19,	0,	9,	0 0
19,	1,	25,	1 1
20,	0,	10,	1 0
20,	1,	26,	0 1
21,	0,	10,	0 1
21,	1,	26,	1 0
22,	0,	11,	0 0
22,	1,	27,	1 1
23,	0,	11,	1 1
23,	1,	27,	0 0
24,	0,	12,	1 1
24,	1,	28,	0 0
25,	0,	12,	0 0
25,	1,	28,	1 1
26,	0,	13,	0 1
26,	1,	29,	1 0
27,	0,	13,	1 0
27,	1,	29,	0 1
28,	0,	14,	0 0
28,	1,	30,	1 1
29,	0,	14,	1 1
29,	1,	30,	0 0
30,	0,	15,	1 0
30,	1,	31,	0 1

31,	0,	15,	0 1
31,	1,	31,	1 0

Total state: 32

***************Backward trellis look-up table***********

***************At constraint length 6****************

current state,	input,	next state,	output
0,	0,	0,	0 0
0,	0,	1,	1 1
1,	0,	2,	1 0
1,	0,	3,	0 1
2,	0,	4,	1 1
2,	0,	5,	0 0
3,	0,	6,	0 1
3,	0,	7,	1 0
4,	0,	8,	1 0
4,	0,	9,	0 1
5,	0,	10,	0 0
5,	0,	11,	1 1
6,	0,	12,	0 1
6,	0,	13,	1 0
7,	0,	14,	1 1
7,	0,	15,	0 0
8,	0,	16,	0 1
8,	0,	17,	1 0
9,	0,	18,	1 1
9,	0,	19,	0 0
10,	0,	20,	1 0
10,	0,	21,	0 1
11,	0,	22,	0 0
11,	0,	23,	1 1
12,	0,	24,	1 1
12,	0,	25,	0 0

13, 0, 26, 0 1
13, 0, 27, 1 0
14, 0, 28, 0 0
14, 0, 29, 1 1
15, 0, 30, 1 0
15, 0, 31, 0 1
16, 1, 0, 1 1
16, 1, 1, 0 0
17, 1, 2, 0 1
17, 1, 3, 1 0
18, 1, 4, 0 0
18, 1, 5, 1 1
19, 1, 6, 1 0
19, 1, 7, 0 1
20, 1, 8, 0 1
20, 1, 9, 1 0
21, 1, 10, 1 1
21, 1, 11, 0 0
22, 1, 12, 1 0
22, 1, 13, 0 1
23, 1, 14, 0 0
23, 1, 15, 1 1
24, 1, 16, 1 0
24, 1, 17, 0 1
25, 1, 18, 0 0
25, 1, 19, 1 1
26, 1, 20, 0 1
26, 1, 21, 1 0
27, 1, 22, 1 1
27, 1, 23, 0 0
28, 1, 24, 0 0
28, 1, 25, 1 1
29, 1, 26, 1 0
29, 1, 27, 0 1

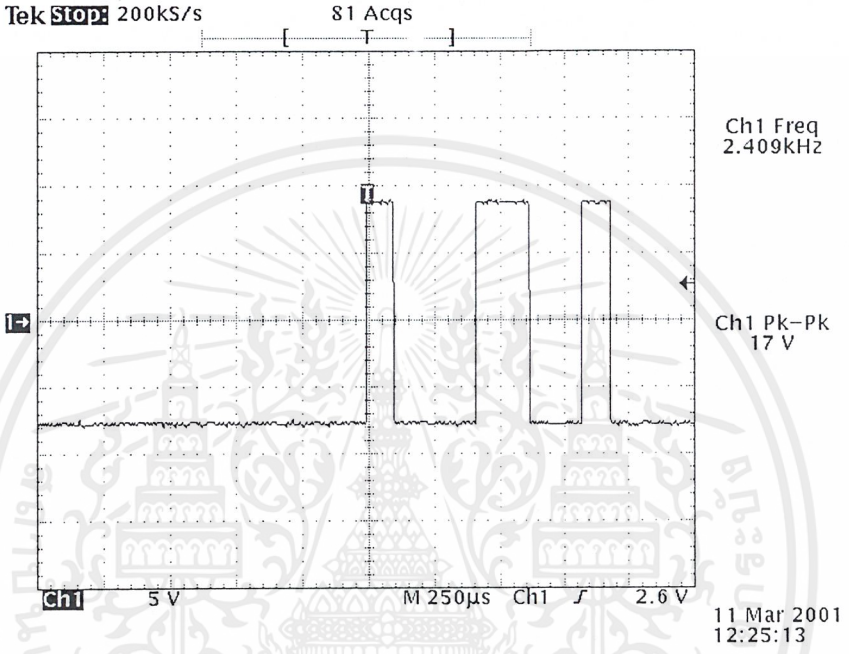
30, 1, 28, 1 1

30, 1, 29, 0 0

31, 1, 30, 0 1

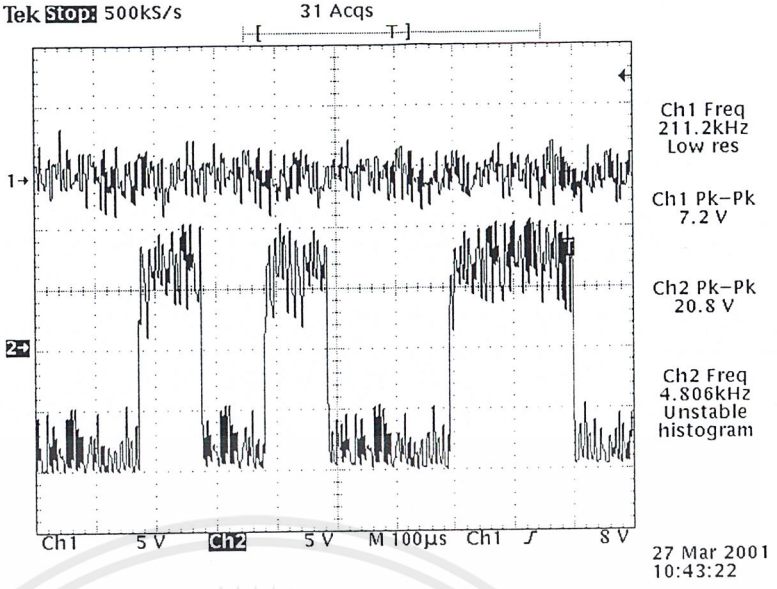
31, 1, 31, 1 0

Total state: 32



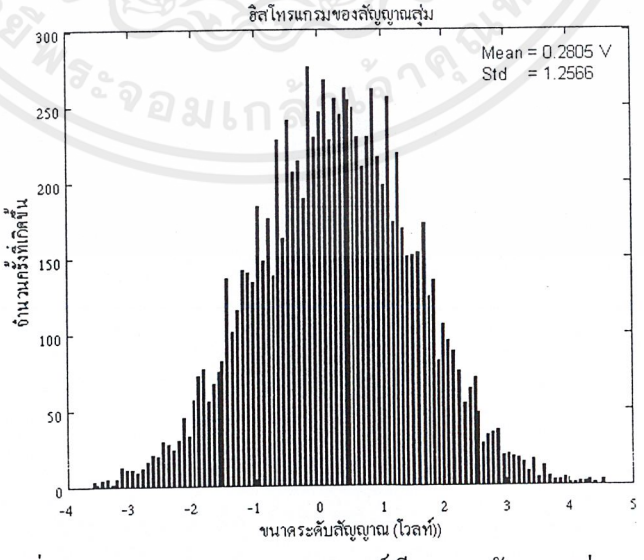
รูปที่ 4.10 แสดงระดับแรงดันไฟฟ้าของสัญญาณที่ออกจากพอร์ตอนุกรม

จากรูปที่ 4.10 นั้นแสดงระดับสัญญาณไฟฟ้าของข้อมูลที่ออกจากพอร์ตอนุกรมของเครื่องคอมพิวเตอร์ ซึ่งมีลักษณะตามมาตรฐาน RS-232 คือ รูปร่างของสัญญาณเป็นแบบสัญญาณมีขั้ว(Polar Signaling) โดยสัญญาณของข้อมูลเหล่านี้จะถูกส่งออกตามสายส่งไปยังคอมพิวเตอร์ปลายทางที่เป็นตัวรอรับข้อมูล



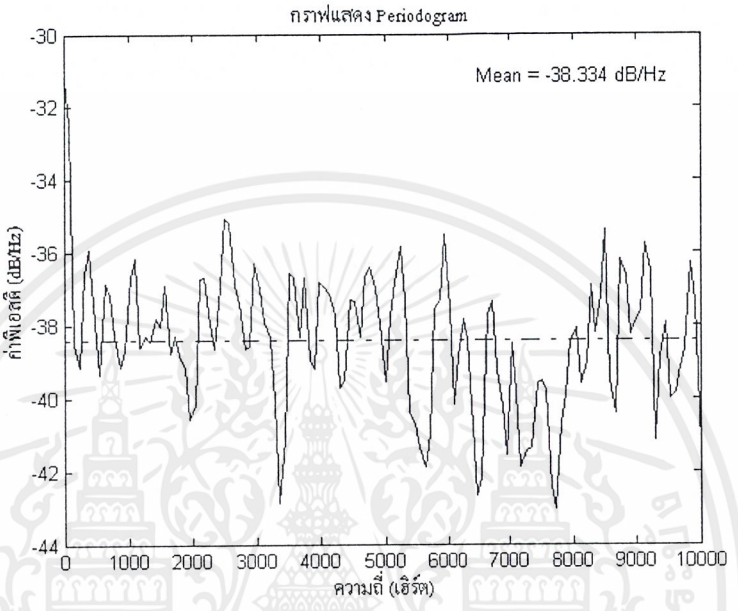
รูปที่ 4.11 ช่องสัญญาณที่ 1 แสดงลักษณะของสัญญาณสุ่ม
ช่องสัญญาณที่ 2 แสดงลักษณะของสัญญาณข้อมูลเมื่อรวมกับสัญญาณสุ่ม

จากรูปที่ 4.11 นี้ สำหรับช่องสัญญาณที่ 1 คือสัญญาณสุ่มที่เรานำมาจากเครื่องกำเนิดสัญญาณ (Function Generator) เพื่อที่จะนำมาเป็นสัญญาณรบกวนที่เข้าไปรวมกับสัญญาณข้อมูลระหว่างการส่งข้อมูลผ่านสายนำสัญญาณแบบทองแดง โดยเราจะนำสัญญาณข้อมูลจากรูปที่ 4.10 และสัญญาณสุ่มจากรูปที่ 4.11 มาผ่านวงจรรวมสัญญาณ(Summing Amplifier) ซึ่งจะได้สัญญาณดังช่องสัญญาณที่ 2 หรือสัญญาณที่ผ่านตัวกลางแบบมีสัญญาณรบกวนนั่นเอง



รูปที่ 4.12 แสดงการกระจายแบบเกาส์เซียนของสัญญาณสุ่ม

รูปที่ 4.12 นั้นแสดงการกระจายแบบเกาส์เซียนของสัญญาณสุ่ม ซึ่งแสดงลักษณะของสัญญาณสุ่มในรูปที่ 4.11 บนช่องสัญญาณที่ 1 โดยเราจะบันทึกตัวอย่างของสัญญาณสุ่มทั้งหมด 10 ครั้ง แล้วนำข้อมูลที่ได้จากการบันทึกไปวิเคราะห์โดยใช้โปรแกรม MATLAB พบว่าสัญญาณสุ่มที่นำมาใช้ในการทดลองนี้มีลักษณะการกระจายแบบปกติ (Normal Distribution) หรือการกระจายแบบเกาส์เซียน(Gaussian Distribution) ซึ่งเป็นลักษณะของช่องสัญญาณสื่อสารโดยทั่วไป

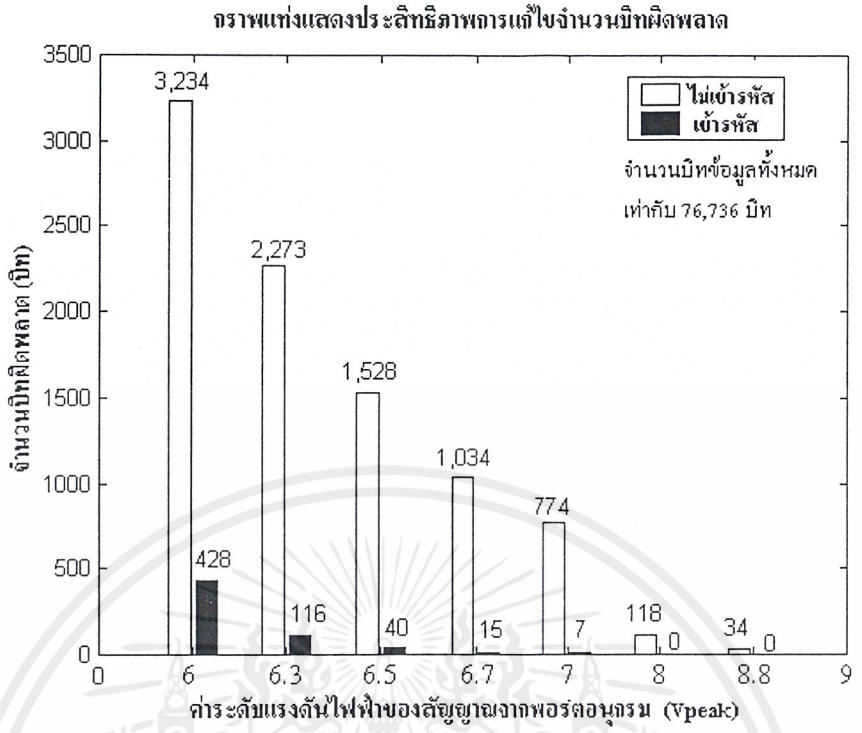


รูปที่ 4.13 แสดงค่าประมาณความหนาแน่นกำลังเชิงสเปกตรัม (PSD) ของสัญญาณสุ่ม

รูปที่ 4.13 นี้ก็แสดงคุณลักษณะของสัญญาณสุ่มคล้ายกับรูปที่ 4.12 โดยการวิเคราะห์จากโปรแกรม MATLAB ค่า PSD นี้จะมีประโยชน์ในการหาค่าอัตราส่วนสัญญาณพลังงานต่อความหนาแน่นสัญญาณรบกวนใน 1 บิท (E_b/N_o) ซึ่งค่า PSD นั้น ก็คือค่า N_o นั่นเอง

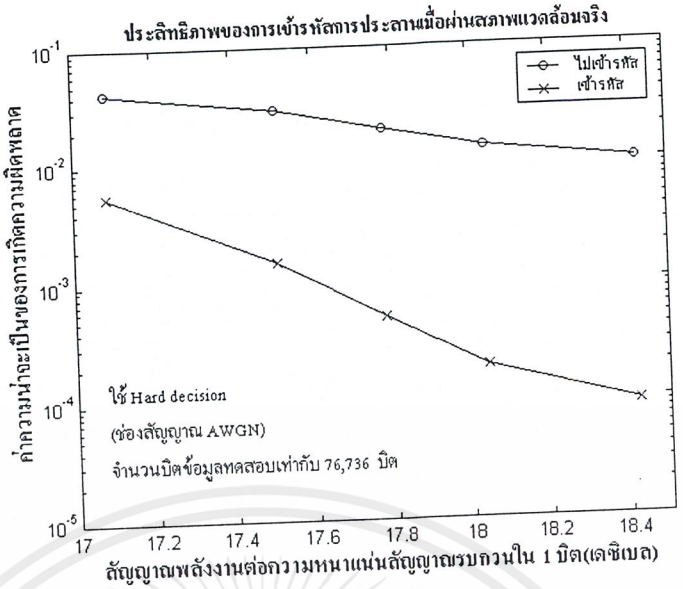
ระดับแรงดันไฟฟ้าของสัญญาณจากพอร์ตอนุกรม (Vpeak)	ไม่เข้ารหัส		เข้ารหัส	
	จำนวนบิทผิดพลาด(บิท)	BER	จำนวนบิทผิดพลาด(บิท)	BER
8.8	34	0.000443	0	0
8	118	0.001538	0	0
7	774	0.010087	7	0.000091
6.7	1,034	0.013475	15	0.000195
6.5	1,528	0.019912	40	0.000521
6.3	2,273	0.029621	116	0.001512
6	3,234	0.042144	428	0.005578

ตารางที่ 4.1 แสดงการเปรียบเทียบจำนวนบิทผิดพลาดและค่า BER

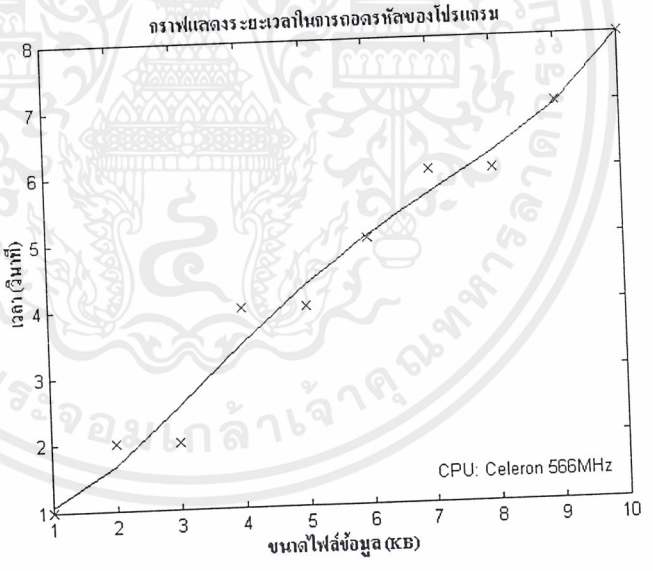


รูปที่ 4.14 กราฟแท่งแสดงประสิทธิภาพในการแก้ไขความผิดพลาด

จากตารางที่ 4.1 นั้น แสดงจำนวนบิทผิดพลาดที่เกิดขึ้นระหว่างการส่ง-รับข้อมูลของคอมพิวเตอร์ผ่านตัวกลางสายส่งที่มีสัญญาณรบกวน โดยข้อมูลที่เป็นตัวเลขเหล่านี้เมื่อนำมาแสดงเป็นกราฟแท่งดังรูปที่ 4.14 เพื่อให้สามารถมองเห็นประสิทธิภาพในการแก้ไขความผิดพลาดที่เกิดขึ้นของการเข้ารหัสช่องสัญญาณ โดยใช้รหัสการประสานหรือคอนโวลูชั่นแนลโค๊ด (Convolutional Code) จะเห็นว่ากราฟแท่งของข้อมูลที่ไม่ได้ทำการเข้ารหัสช่องสัญญาณนั้น จะมีจำนวนบิทผิดพลาดเกิดขึ้นมากกว่าข้อมูลที่ทำการเข้ารหัสช่องสัญญาณและถอดรหัสแล้ว นอกจากนี้ระดับของสัญญาณไฟฟ้าที่ออกจากพอร์ตอนุกรมยังมีผลต่อจำนวนบิทผิดพลาดด้วย กล่าวคือ หากเราเพิ่มระดับของสัญญาณไฟฟ้าที่ออกจากพอร์ตอนุกรมนั่นหมายถึงอัตราส่วนของสัญญาณพลังงานต่อความหนาแน่นสัญญาณรบกวนใน 1 บิท (E_b/N_o) ที่เพิ่มขึ้นด้วย ทำให้จำนวนบิทผิดพลาดที่เกิดขึ้นน้อยลง



รูปที่ 4.15 แสดงประสิทธิภาพการเข้ารหัสการประสาน
เมื่อส่งข้อมูลผ่านสภาพแวดล้อมจริง



รูปที่ 4.16 แสดงระยะเวลาการทำงานของโปรแกรม
ในการถอดรหัสการประสาน

จากรูปที่ 4.16 นั้นจะแสดงระยะเวลาในการถอดรหัส ซึ่งขั้นตอนการทำงานของโปรแกรมจะใช้เวลานานที่สุดตรงกระบวนการถอดรหัสการประสาน โดยหากค่าความยาวคอนสเตรนท์(K) ยิ่งเพิ่มขึ้น ระยะเวลาในการประมวลผลก็จะเพิ่มขึ้นด้วย แต่ประสิทธิภาพของการแก้ไขความผิดพลาดก็จะดีขึ้น

บทที่ 5
บทวิจารณ์และบทสรุป

5.1 ส่วนจำลองเหตุการณ์ (Simulation)

จากการจำลองเหตุการณ์ในการส่ง-รับข้อมูลผ่านช่องสัญญาณที่มีการรบกวน โดยใช้โปรแกรมนั้น เราจะพบว่ากราฟเปรียบเทียบประสิทธิภาพในการส่ง-รับข้อมูลระหว่างข้อมูลที่ไม่ได้ทำการเข้ารหัสกับข้อมูลที่ทำการเข้ารหัสนั้น ข้อมูลที่ทำการเข้ารหัสช่องสัญญาณหรือรหัสควบคุมความผิดพลาดจะให้ผลของค่าความน่าจะเป็นของการเกิดความผิดพลาด มีค่าที่น้อยกว่าข้อมูลที่ไม่ได้ทำการเข้ารหัส เมื่อเปรียบเทียบที่อัตราส่วนของสัญญาณพลังงานต่อความหนาแน่นสัญญาณรบกวนใน 1 บิท (Eb/No) ที่ค่าเดียวกัน และหากเปรียบเทียบระหว่างข้อมูลที่เข้ารหัสการประสาน แต่ใช้วิธีในการตัดสินระดับสัญญาณทางด้านรับที่ต่างกันคือ ตัดสินแบบฮาร์ด (Hard Decision) หรือ ตัดสินแบบซอร์ฟ(Soft Decision) จะพบว่าหากใช้การตัดสินแบบซอร์ฟนั้น จะให้ประสิทธิภาพที่ดีกว่าคือ ให้ค่าความน่าจะเป็นของการเกิดความผิดพลาดน้อยกว่าการใช้การตัดสินระดับสัญญาณแบบฮาร์ด นอกจากนี้ประสิทธิภาพในการลดความน่าจะเป็นของการเกิดความผิดพลาดจะเพิ่มขึ้นตามจำนวนรีจิสเตอร์หรือความยาวคอนสเตร็นท์(K)ที่ใช้ แต่ก็หมายถึงจำนวนพื้นที่หน่วยความจำทางด้านรับที่จำเป็นต้องจองไว้ใช้สำหรับจดจำเส้นทางเพิ่มขึ้นด้วย เพื่อทำการถอดรหัสช่องสัญญาณด้วยวิธีไวเทอร์บี(Viterbi) โดยขั้นตอนในการถอดรหัสนั้น จำนวนขั้นตอนในการประมวลผลหรือจำนวนสเตทของตาราง Trellis มีค่าเท่ากับ 2^{M-1} ดังนั้นเมื่อเพิ่มความยาวคอนสเตร็นท์ ทำให้การคำนวณเพื่อเลือกเส้นทางที่ถูกต้องในแต่ละขั้นตอน จะมีค่าที่เพิ่มขึ้นเป็นเอ็กซ์โปเนนเชียล ดังนั้นข้อจำกัดในการจำลองเหตุการณ์(Simulation)ครั้งนี้ จึงอยู่ที่ขนาดพื้นที่หน่วยความจำของคอมพิวเตอร์ ซึ่งทำให้การเข้ารหัสการประสานนั้นสามารถใช้ค่าความยาวคอนสเตร็นท์ได้สูงสุดเพียง 6 นอกจากนี้ระยะเวลาที่โปรแกรมทำงานในการจำลองเหตุการณ์แต่ละครั้งนั้น ใช้เวลานานพอสมควรซึ่งระยะเวลาส่วนใหญ่จะตกอยู่ที่กระบวนการถอดรหัสการประสานเพราะจะต้องมีการคำนวณและตัดสินใจมาก ควรที่จะหาทางแก้ไขให้มีการทำงานที่เร็วกว่านี้

5.2 ส่วนประยุกต์ใช้งาน (Practical Application)

สำหรับส่วนประยุกต์ใช้งานที่มีการทดลองส่งข้อมูลผ่านสภาพแวดล้อมจริงนั้น จากผลการทดลองจะพบว่าการเข้ารหัสการประสานนั้นสามารถช่วยปรับปรุงค่าอัตราส่วนความผิดพลาด(BER) ให้ดีขึ้นได้จริง และมีผลสอดคล้องกับส่วนจำลองเหตุการณ์คือในกราฟเปรียบเทียบประสิทธิภาพการเข้ารหัสการประสานนั้น กราฟของข้อมูลที่เข้ารหัสจะมีความชันมากกว่ากราฟของข้อมูลที่ไม่ได้เข้ารหัส ดังนั้นโปรแกรมนี้จึงสามารถนำไปประยุกต์ใช้งานได้จริง แต่จำเป็นต้องนำไฟล์ข้อมูลใดๆไปทำการเข้ารหัสก่อนแล้วนำไฟล์ทั้งหมดที่เข้ารหัสแล้ว ส่งผ่านช่องสัญญาณสื่อสารไปยังด้านรับ แล้วด้านรับต้องรับไฟล์ทั้งหมดนำไปถอดรหัสอีกครั้งหนึ่ง จะเห็นว่าเราไม่สามารถทำการเข้ารหัสและถอดรหัสทางด้านรับได้ทันที ซึ่งโปรแกรมจะต้องได้รับการพัฒนาและแก้ไขต่อไป นอกจากนี้การเข้ารหัสการประสานของโปรแกรม

สามารถที่จะแก้ไขความผิดพลาดที่เกิดขึ้นในลักษณะสุ่ม(Random Error Pattern)เท่านั้น หากบิทผิดพลาดเกิดขึ้นทางด้านรับติดๆกันหลายบิท(Burst Error Pattern) โปรแกรมก็ไม่อาจจะแก้ไขข้อมูลให้ถูกต้องได้ทั้งหมด

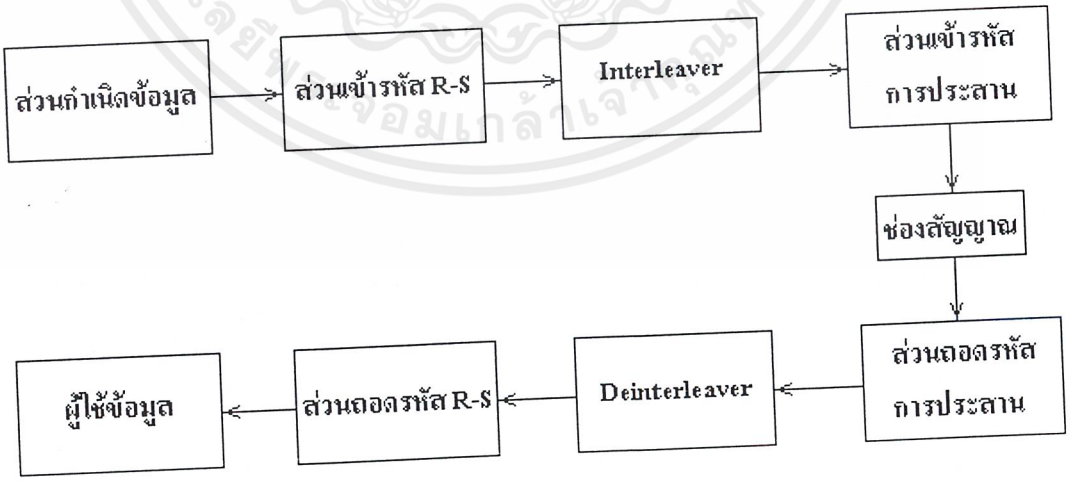
5.3 แนวทางในการพัฒนา

สำหรับแนวทางในการพัฒนานั้น เราสามารถจะพิจารณาได้ 2 ลักษณะคือ

1. ความเร็วในการทำงาน
2. ประสิทธิภาพในการแก้ไขความผิดพลาด

สำหรับความเร็วในการทำงานของการเข้ารหัสและถอดรหัสการประสานนั้น เราสามารถจะปรับปรุงให้ดีขึ้นได้โดยการประยุกต์ใช้อุปกรณ์ฮาร์ดแวร์ ช่วยในการเข้าและถอดรหัสการประสาน ซึ่งอาจจะนำชุดไมโครโปรเซสเซอร์ที่มีความเร็วสูงและมีคำสั่งที่สนับสนุนในการใช้งาน เช่นชุดฮาร์ดแวร์ตระกูล TMSC5X ของบริษัท Texas Instruments มาประยุกต์ใช้กับโครงงาน

ส่วนในเรื่องของประสิทธิภาพในการแก้ไขความผิดพลาดนั้น เราสามารถจะปรับปรุงให้ดีขึ้นได้โดยเพิ่มความยาวคอนสเตรนท์และปรับโครงสร้างของตัวเข้ารหัสการประสาน แต่นั้นก็หมายถึงจำนวนหน่วยความจำที่เพิ่มขึ้นทางด้านรับสำหรับการจดจำเส้นทางเพื่อใช้ในการถอดรหัสการประสาน หรืออาจจะนำบล็อคโค๊ดมาใช้ในการเข้ารหัสและถอดรหัสร่วมกับรหัสการประสานเพื่อเพิ่มประสิทธิภาพ (Concatenated Coding) และจากลักษณะของการแก้ไขความผิดพลาดที่เกิดขึ้นแบบสุ่มนี้ (Random Error Pattern) เราสามารถจะเพิ่มประสิทธิภาพให้ทำการแก้ไขความผิดพลาดที่เกิดขึ้นแบบต่อเนื่อง(Burst Error Pattern)ได้ โดยนำรหัสบล็อคโค๊ดแบบ Reed –Solomon และเทคนิค Interleaving มาประยุกต์ใช้ร่วมกับรหัสการประสาน ดังแสดงในรูปที่ 5.1



รูปที่ 5.1 แสดงการประยุกต์ใช้งานสำหรับการแก้ไขความผิดพลาดแบบต่อเนื่อง(Burst Error Pattern)

ภาคผนวก

โปรแกรมส่วนประกาศตัวแปร(Header File)

// ConvoDlg.h : header file (ส่วนประยุกต์ใช้งาน)

```
//

#if
!defined(AFX_CONVODLG_H__6ED49A46_DB3A_11D4_8159_CF2AF96A8331__INCLUD
ED_)
#define
AFX_CONVODLG_H__6ED49A46_DB3A_11D4_8159_CF2AF96A8331__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////
////////
// CConvoDlg dialog

class CConvoDlg : public CDialog
{
// Construction
public:
      CConvoDlg(CWnd* pParent = NULL);    // standard constructor

    struct trellis{
      unsigned int state;
      int input;
      int code[2];
      };
      struct cumulative_matrix{
            unsigned int prev_state;
            int hamming;
            int input;
      };

      trellis **imatrix(int row, int col);
      void free_imatrix(trellis **matrix, int row);
      void create_table();
      void AddDlgItemText(int nid, LPCTSTR input);
      void encode(int input,int *output);
      void decode(int *code,int Is_soft,int LAP);
      void ClearStatus();
      trellis **F_table,**B_table;
    int ig,k,m,n,iv,xt;
    int *data_bit;
    int curr_state,state;
    FILE *fp,*wp;
    char ch;
    CString WriteFileName;
    int *output;
    CString OutputFileName;
      char *Ong ;
    /*****************************************************/
      //unsigned int DATA_REG = (unsigned int)(0x3F8);
    //unsigned  int filesize(FILE *fp);
    //void sport(int port,unsigned char c),send_file(char
*fname),rec_file();
    //void port_init(int port,unsigned char code);
    //void wait(int port);
    //char rport(int port);
```

```
    //int check_stat(int port);
    /*****************************************************/

// Dialog Data
      //{{AFX_DATA(CConvoDlg)
      enum { IDD = IDD_CONVO_DIALOG };
      CButton     m_button;
      CAnimateCtrl      m_animate;
      CProgressCtrl     m_progress;
      //}}AFX_DATA

      // ClassWizard generated virtual function overrides
      //{{AFX_VIRTUAL(CConvoDlg)
      protected:
      virtual void DoDataExchange(CDataExchange* pDX);      //
DDX/DDV support
      //}}AFX_VIRTUAL

// Implementation
protected:
      HICON m_hIcon;

      // Generated message map functions
      //{{AFX_MSG(CConvoDlg)
      virtual BOOL OnInitDialog();
      afx_msg void OnPaint();
      afx_msg HCURSOR OnQueryDragIcon();
      afx_msg void OnEncode();
      afx_msg void OnDecode();
      afx_msg void OnSimulate();
      virtual void OnCancel();
      afx_msg void OnAbout();
      afx_msg void OnComm();
      afx_msg void OnFinderr();
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.

#endif //
!defined(AFX_CONVODLG_H__6ED49A46_DB3A_11D4_8159_CF2AF96A8331__INCLUD
ED_)
```

// ConvoSimu.h : header file(ส่วน Simulation)

```
#if
!defined(AFX_CONVOSIMU_H__6F7ACB90_DD8E_11D4_8BC4_8447EF216B20__INCLU
DED_)
#define
AFX_CONVOSIMU_H__6F7ACB90_DD8E_11D4_8BC4_8447EF216B20__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000
// ConvoSimu.h : header file
//
```

```
////////////////////////////////////////////////////////////////////////
////////
// CConvoSimu dialog

class CConvoSimu : public CDialog
{
// Construction
public:
      CConvoSimu(UINT nID,CWnd* pParent = NULL);    // standard
constructor
    struct trellis{
      unsigned int state;
      int input;
      int code[8];
      };
      struct cumulative_matrix{
            unsigned int prev_state;
            double hamming;
            int input;
      };
trellis **imatrix(int row, int col);
void free_imatrix(trellis **matrix, int row);
void AddDlgItemText(int nid, LPCTSTR input);
void create_table(void);
void pasteplot(void);
void encode(int input,double *output);
void decode(double *code,int Is_soft);
void modulator(double *input,double *output);
void Soft_demodulator(double input,double *output);
void Hard_demodulator(double input,double *output);
int totaldata;
double normal_rand(double mean, double std_dev);
double Q(double x);
trellis **F_table,**B_table;
int k,m,n,*data_bit,MAX,NO_ROUTE,PACKET,iz;
double M_PI;
unsigned int curr_state,state;
unsigned long error;
char *Ong,*Tle,*Tong ;
int A,B,C,D;
// Dialog Data
      //{{AFX_DATA(CConvoSimu)
      enum { IDD = IDD_SIMULATION };
      CAnimateCtrl      m_animate1;
      CProgressCtrl     m_progress2;
      CSliderCtrl m_slider;
      int         m_slidervalue;
      int         m_editslider;
      //}}AFX_DATA


// Overrides
      // ClassWizard generated virtual function overrides
      //{{AFX_VIRTUAL(CConvoSimu)
      protected:
      virtual void DoDataExchange(CDataExchange* pDX);     // DDX/DDV
support
      //}}AFX_VIRTUAL

// Implementation
```

```
protected:
    HICON m_hIcon;
      // Generated message map functions
      //{{AFX_MSG(CConvoSimu)
      virtual BOOL OnInitDialog();
      afx_msg void OnPaint();
      afx_msg HCURSOR OnQueryDragIcon();
      afx_msg void OnRadio1();
      afx_msg void OnRun();
      afx_msg void OnRadio2();
      afx_msg void OnRadio3();
      afx_msg void OnRadio4();
      afx_msg void OnRadio5();
      afx_msg void OnRadio6();
      afx_msg void OnRadio7();
      afx_msg void OnHScroll(UINT nSBCode, UINT nPos, CScrollBar*
pScrollBar);
      //}}AFX_MSG
      DECLARE_MESSAGE_MAP()
};


//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations
immediately before the previous line.

#endif //
!defined(AFX_CONVOSIMU_H__6F7ACB90_DD8E_11D4_8BC4_8447EF216B20__INCLU
DED_)
```

โปรแกรมส่วนจำลองเหตุการณ์(Simulation)

```
// ConvoSimu.cpp : implementation file
//

#include "stdafx.h"
#include "afx.h"
#include "Convo.h"
#include "math.h"
#include "ConvoSimu.h"
#include "math.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////
////////
// CConvoSimu dialog
CTime dse;

CConvoSimu::CConvoSimu(UINT nID,CWnd* pParent):CDialog(nID,pParent)
{
      //{{AFX_DATA_INIT(CConvoSimu)
```

```
	m_slidervalue = 3;
	m_editslider = 3;
	//}}AFX_DATA_INIT
	m_hIcon = AfxGetApp()->LoadIcon(IDI_ICON5);
}


void CConvoSimu::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CConvoSimu)
	DDX_Control(pDX, IDC_ANIMATE1, m_animate1);
	DDX_Control(pDX, IDC_PROGRESS2, m_progress2);
	DDX_Control(pDX, IDC_SLIDER, m_slider);
	DDX_Slider(pDX, IDC_SLIDER, m_slidervalue);
	DDX_Text(pDX, IDC_EDITSLIDE, m_editslider);
	//}}AFX_DATA_MAP
}


BEGIN_MESSAGE_MAP(CConvoSimu, CDialog)
	//{{AFX_MSG_MAP(CConvoSimu)
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_RADIO1, OnRadio1)
	ON_BN_CLICKED(ID_RUN, OnRun)
	ON_BN_CLICKED(IDC_RADIO2, OnRadio2)
	ON_BN_CLICKED(IDC_RADIO3, OnRadio3)
	ON_BN_CLICKED(IDC_RADIO4, OnRadio4)
	ON_BN_CLICKED(IDC_RADIO5, OnRadio5)
	ON_BN_CLICKED(IDC_RADIO6, OnRadio6)
	ON_BN_CLICKED(IDC_RADIO7, OnRadio7)
	ON_WM_HSCROLL()
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
///////
// CConvoSimu message handlers

BOOL CConvoSimu::OnInitDialog()
{	CDialog::OnInitDialog();
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon
    m_slider.SetRange(3,6);
	MAX =100000;
    PACKET= 20;
    NO_ROUTE= 300000;
    M_PI =3.141592654;
	Ong = new char[100];
	Tle = new char[100];
	Tong= new char[100];
	m_progress2.SetRange(0,13);
	m_progress2.SetStep(1);
	m_progress2.SetPos(0);
	return true;
}

void CConvoSimu::OnPaint()
{
	if (IsIconic())
```

```
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(),
0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

HCURSOR CConvoSimu::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}



void CConvoSimu::OnRadio1()
{
	CButton *yesbutton1;
	yesbutton1=(CButton*)GetDlgItem(IDC_RADIO1);
	if ((yesbutton1->GetCheck())==1)
		n=2;
}


void CConvoSimu::OnRadio2()
{
	CButton *yesbutton2;
	yesbutton2=(CButton*)GetDlgItem(IDC_RADIO2);
    if ((yesbutton2->GetCheck())==1)
		n=3;

}

void CConvoSimu::OnRadio3()
{
	CButton *yesbutton3;
	yesbutton3=(CButton*)GetDlgItem(IDC_RADIO3);
    if ((yesbutton3->GetCheck())==1)
		n=4;
}

void CConvoSimu::OnRadio4()
{
    CButton *yesbutton4;
	yesbutton4=(CButton*)GetDlgItem(IDC_RADIO4);
```

```
    if ((yesbutton4->GetCheck())==1)
            A=1;

}

void CConvoSimu::OnRadio5()
{
      CButton *yesbutton5;
      yesbutton5=(CButton*)GetDlgItem(IDC_RADIO5);
    if ((yesbutton5->GetCheck())==1)
            B=1;

}

void CConvoSimu::OnRadio6()
{
    CButton *yesbutton6;
      yesbutton6=(CButton*)GetDlgItem(IDC_RADIO6);
    if ((yesbutton6->GetCheck())==1)
            C=1;

}

void CConvoSimu::OnRadio7()
{
      CButton *yesbutton7;
      yesbutton7=(CButton*)GetDlgItem(IDC_RADIO7);
    if ((yesbutton7->GetCheck())==1)
            D=1;

}


void CConvoSimu::OnHScroll(UINT nSBCode, UINT nPos, CScrollBar*
pScrollBar)
{
    CSliderCtrl *s;
      s = (CSliderCtrl*)pScrollBar;
      if (s==&m_slider){
            UpdateData(TRUE);
            m_editslider = m_slidervalue;
            UpdateData(FALSE);
      }
      CDialog::OnHScroll(nSBCode, nPos, pScrollBar);
}


void CConvoSimu::create_table()
{
 int it,iu,ix,iy,iv,tmp,mem[9];
 int generator[8];
 unsigned int *idx;
 state=(unsigned int)pow(2,m);
 if(n==2)
 {
   switch(m){
    case 2:generator[0]=7;
          generator[1]=5;
                break;
    case 3:generator[0]=60;
          generator[1]=52;
```

```
                break;
    case 4:generator[0]=58;
           generator[1]=38;
             break;
    case 5:generator[0]=47;
           generator[1]=53;
             break;
   }
}
if(n==3)
{
  switch(m){
   case 2:generator[0]=7;
          generator[1]=7;
          generator[2]=5;
          break;
     case 3:generator[0]=60;
          generator[1]=52;
          generator[2]=44;
          break;
     case 4:generator[0]=62;
          generator[1]=54;
          generator[2]=42;
          break;
     case 5:generator[0]=61;
          generator[1]=43;
          generator[2]=39;
          break;
   }
 }
 if(n==4)
 {
   switch(m){
    case 2:generator[0]=7;
          generator[1]=7;
          generator[2]=7;
          generator[3]=5;
          break;
      case 3:generator[0]=60;
          generator[1]=52;
          generator[2]=52;
          generator[3]=44;
          break;
      case 4:generator[0]=62;
          generator[1]=54;
          generator[2]=46;
          generator[3]=42;
          break;
      case 5:generator[0]=61;
          generator[1]=57;
          generator[2]=55;
          generator[3]=43;
          break;
    }
  }

  /** begin trellis table generation part***************/

  idx = (unsigned int *)malloc(sizeof(int)*state);
  memset(idx,0,sizeof(int)*state);
  CConvoSimu::F_table = CConvoSimu::imatrix(state,2);
```

```
 CConvoSimu::B_table = CConvoSimu::imatrix(state,2);
 for(ix=0;ix<state;ix++) //state = 2^m
  for(iy=0;iy<2;iy++)    //each state has 2^k branches (k = 1)
  {
    mem[0] = iy;
    CConvoSimu::F_table[ix][iy].input = iy;
    iv = (ix>>1)|(iy<<(m-1));
    CConvoSimu::F_table[ix][iy].state = iv;  //set next state value
    CConvoSimu::B_table[iv][idx[iv]].state = ix;  //set previous
state value
    CConvoSimu::B_table[iv][idx[iv]].input = iy;
    for(it=0;it<m;it++)
    {
      if(ix&(1<<it))
      mem[m-it] = 1;
      else mem[m-it] = 0;
    }
    for(it=0;it<n;it++)
    {
      tmp = 0;
      for(iu=0;iu<=m;iu++)
       if(generator[it]&(1<<(m-iu)))
      tmp += mem[iu];
      tmp %= 2;
      CConvoSimu::F_table[ix][iy].code[it]=tmp;
      CConvoSimu::B_table[iv][idx[iv]].code[it]=tmp;
    }
    idx[iv]++;
  }
 free(idx);

   AddDlgItemText(IDC_EDIT1," \r\n");
   AddDlgItemText(IDC_EDIT1,"*********Forward trellis look-up
table*********\r\n");
   AddDlgItemText(IDC_EDIT1,"current state,   input,   next state,
output\r\n");

   for(ix = 0;ix < state;ix++)
    for(iy = 0;iy< 2;iy++)
      {
        sprintf(Ong,"      %d,
%d,",ix,CConvoSimu::F_table[ix][iy].input);
      AddDlgItemText(IDC_EDIT1,Ong);
        sprintf(Ong,"                %2d,
",CConvoSimu::F_table[ix][iy].state);
        AddDlgItemText(IDC_EDIT1,Ong);
      for(it = 0;it < n;it++)
        {   sprintf(Ong,"%d ",CConvoSimu::F_table[ix][iy].code[it]);
            AddDlgItemText(IDC_EDIT1,Ong);
          UpdateWindow( );
        }
        AddDlgItemText(IDC_EDIT1," \r\n");

      }


   AddDlgItemText(IDC_EDIT1," \r\n");
   AddDlgItemText(IDC_EDIT1,"*********Backward trellis look-up
table*********\r\n");
   AddDlgItemText(IDC_EDIT1,"current state,   input,   next state,
output\r\n");
```

```
   for(ix = 0;ix < state;ix++)
    for(iy = 0;iy< 2;iy++)
      {
        sprintf(Ong,"      %d,
%d,",ix,CConvoSimu::B_table[ix][iy].input);
      AddDlgItemText(IDC_EDIT1,Ong);
        sprintf(Ong,"           %2d,
",CConvoSimu::B_table[ix][iy].state);
        AddDlgItemText(IDC_EDIT1,Ong);
      for(it = 0;it < n;it++)
        {   sprintf(Ong,"%d ",CConvoSimu::B_table[ix][iy].code[it]);
            AddDlgItemText(IDC_EDIT1,Ong);
        }
        AddDlgItemText(IDC_EDIT1," \r\n");
      }


}

void CConvoSimu::AddDlgItemText(int nid,LPCTSTR input)
{     CString temp;
      GetDlgItemText(nid,temp);
      SetDlgItemText(nid,temp+input);
}

CConvoSimu::trellis **CConvoSimu::imatrix(int row,int col)
{
   int i;
   CConvoSimu::trellis **m;
   m= (CConvoSimu::trellis **)
malloc(row*sizeof(CConvoSimu::trellis)) ;
   if (!m) AfxMessageBox("Memory allocation failure.") ;
   for (i=0;i<row; i++)
   {
       m[i] = (CConvoSimu::trellis *) malloc(col*sizeof
(CConvoSimu::trellis)) ;
     if (!m[i]) AfxMessageBox("Memory alloation failure.");
   }
     return m ;
}
void CConvoSimu::free_imatrix(CConvoSimu::trellis **matrix, int row)
{
  int i ;
  for (i = row - 1 ; i >= 0 ; i--)
    free((char*) matrix[i]) ;
  free((char*) matrix) ;
}

void CConvoSimu::encode(int input,double *output)
{
 for(int ix=0;ix<2;ix++)
  {
       if(CConvoSimu::F_table[CConvoSimu::curr_state][ix].input ==
input)
    {
     for(int iy=0;iy<n;iy++)
           output[iy] = CConvoSimu::F_table[CConvoSimu::curr state]
[ix].code[iy];
       CConvoSimu::curr_state =
CConvoSimu::F_table[CConvoSimu::curr_state][ix].state;
```

```
    }
  }

}

void CConvoSimu::decode(double *code,int Is_soft)
{
 int iu,iv,ix,iy,tmp,input;
 double distance,H;
 unsigned int idx;
 CConvoSimu::cumulative_matrix *s;
 s=(CConvoSimu::cumulative_matrix
*)malloc(sizeof(CConvoSimu::cumulative_matrix)*(MAX+3*
(m+1)+1)*state);
 if(s==NULL)
 {
   AfxMessageBox("Memory allocation (Cumulative_matrix) failure.");
   return;
 }

 s[0].prev_state = 0;
 s[0].hamming = 0;
 s[0].input = 0;
 for(ix=1;ix<state;ix++)
 {
  s[ix].prev_state = 0;
  s[ix].hamming = NO_ROUTE;
  s[ix].input = -1;
 }

 for(iu=1;iu< (MAX+3*(m+1)+1);iu++)
 {
  for(ix=0;ix<state;ix++)
  {
    H=NO_ROUTE;
    tmp = -1;
    for(iy=0;iy<2;iy++)
    {
      idx = CConvoSimu::B_table[ix][iy].state;  //idx :previous state
      distance = s[(iu-1)*state+idx].hamming;
      for(iv=0;iv<n;iv++)
      {
      if(Is_soft)
      { // Soft decision :calculate Euclidean distance
        if(CConvoSimu::B_table[ix][iy].code[iv])
          distance += pow(1-code[(iu-1)*n+iv],2);
        else
          distance += pow(-1-code[(iu-1)*n+iv],2);
      }else
      {  // Hard decision :calculate Hamming distance
       distance += pow(CConvoSimu::B_table[ix][iy].code[iv]-code[(iu-
1)*n+iv],2);
      }
      }
      if(distance<=H)
      {  //choose the branch that has the least hamming distance
       H=distance;
       tmp=idx;
       input=CConvoSimu::B_table[ix][iy].input;
      }
    }
```

```
    s[iu*state+ix].prev_state = tmp;
    s[iu*state+ix].hamming = H;
    s[iu*state+ix].input = input;
  }
 }
 idx=0; // set to state 0
 // readout zero-padding length 3 bytes or (24 bits)

 for(ix=MAX+3*(m+1)-1;ix>=MAX;ix--)
   idx = s[(ix+1)*state+idx].prev_state;
 //begin to readout real data part
 for(ix=MAX-1;ix>=0;ix--)
 {
  tmp = s[(ix+1)*state+idx].input;
  idx = s[(ix+1)*state+idx].prev_state;
  if(data_bit[ix]^tmp)
        error += 1;
 }
 free(s);

}

void CConvoSimu::modulator(double *input,double *output)
{
      if(*input)
    *output = 1;
    else
    *output = -1;
}

/* BPSK demodulator - Soft decision no quantization*/
void CConvoSimu::Soft_demodulator(double input,double *output)
{

  *output = input;
}
/*BPSK demodulator - Hard decision*/
void CConvoSimu::Hard_demodulator(double input,double *output)
{
  if(input >= 0)
    *output = 1;
  else
    *output = 0;
}
/* Generate gaussian random double with specified mean and std_dev */

double CConvoSimu::normal_rand(double mean, double std_dev)
{
 double fac,rsq,v1,v2;
 static double gset;
 static int iset;
 if(iset){
   /* Already got one */
   iset = 0;
   return mean + std_dev*gset;
 }
  /* Generate two evenly distributed numbers between -1 and +1
   * that are inside the unit circle
   */
 do {
```

```
    v1 = 2.0 * (double)rand() / RAND_MAX - 1;
    v2 = 2.0 * (double)rand() / RAND_MAX - 1;
    rsq = v1*v1 + v2*v2;
   } while(rsq >= 1.0 || rsq == 0.0);
 fac = sqrt(-2.0*log(rsq)/rsq);
 gset = v1*fac;
 iset++;
 return mean + std_dev*v2*fac;
}
/* This function is used to calculate Q-function
double CConvoSimu::Q(double x)
{
  double a=0.344,b = 5.334;
  double q;
  q = (1-a)*x + a*sqrt(pow(x,2)+b);
  a = exp(-pow(x,2)/2)/sqrt(2*M_PI);
  return a/q;
}


void CConvoSimu::OnRun()
{
 unsigned long int t;
 char msg[100];
 int iv,ix,iy;
 double *output,lo;
 double BER,noise,code,ebn0,esn0,Pe;
 CTime t1 = CTime::GetCurrentTime();
 k = 1;
 m = m_editslider-1;
 CConvoSimu::curr_state=0;
 CConvoSimu::data_bit = (int *)malloc(sizeof(int)*(MAX+3*(m+1)));
 output = (double *)malloc(sizeof(double)*n*(MAX+3*(m+1)));
 if(CConvoSimu::data_bit == NULL || output == NULL)
 {
   AfxMessageBox("Memory allocation failure");
   return;
 }
 m_animate1.Open(IDR_AVI1);
 m_animate1.Play(0,-1,-1);
 state=(unsigned int)pow(2,m);
 SetDlgItemText(IDC_EDIT2,"Creating table...\r\n");
 UpdateWindow();
 CConvoSimu::create_table();
 sprintf(Ong,"Total state : %u\r\n",state);
 AddDlgItemText(IDC_EDIT1,Ong);
 UpdateWindow( );

 if ((A==1)&&(C==1))
       iz=0;
 if ((A==1)&&(D==1))
       iz=1;
 if (B==1)
       iz=2;
 switch(iz){
  case 0: AddDlgItemText(IDC_EDIT1,"\r\nAWGN channel :using Hard
Decision\r\n");
          AddDlgItemText(IDC_HIDE,"y1=[\r\n");
          SetDlgItemText(IDC_EDIT2,"Simulation on AWGN channel");
            UpdateWindow();
            break;
```

```
  case 1: AddDlgItemText(IDC_EDIT1,"\r\nAWGN channel :using Soft
Decision(no quantization)\r\n");
              AddDlgItemText(IDC_HIDE,"y1=[\r\n");
            SetDlgItemText(IDC_EDIT2,"Simulation on AWGN channel");
              UpdateWindow();
              break;
  case 2: AddDlgItemText(IDC_EDIT1,"\r\nBSC  channel :using Hard
Decision\r\n");
              AddDlgItemText(IDC_HIDE,"y1=[\r\n");
              SetDlgItemText(IDC_EDIT2,"Simulation on BSC channel");
              UpdateWindow();
 }

 for(ebn0 = 0;ebn0 <= 6 ;ebn0 += 0.5)
 {

  esn0 = ebn0 + 10*log10(1.0/n);
  noise = sqrt(0.5/pow(10.,esn0/10.));
  error = 0;
  lo=sqrt(2*pow(10,esn0/10));
  Pe = CConvoSimu::Q(lo);

  for(iv = 0;iv < PACKET;iv++)
  {
      sprintf(Tong,"PACKET #%d,\r\nTOTAL:%d",iv+1,PACKET);
      SetDlgItemText(IDC_EDIT3,Tong);
    UpdateWindow( );
    sprintf(Tong,"At Eb/No: %3.2f dB\r\nGenerating random
data...",ebn0);
      SetDlgItemText(IDC_EDIT2,Tong);
      UpdateWindow();
      for(iy = 0; iy < MAX;iy++)          /* randomly generated input
data bits */
    {
            CConvoSimu::data_bit[iy] = rand() % 2;
            CConvoSimu::encode(CConvoSimu::data_bit[iy],&output
[iy*n]);
    }

    sprintf(Tong,"At Eb/No: %3.2f dB\r\nEncoding...",ebn0);
      SetDlgItemText(IDC_EDIT2,Tong);
      UpdateWindow();

    /* pad the zero input 3*(m+1) bits */
      sprintf(Tong,"At Eb/No: %3.2f dB\r\nZero Padding...",ebn0);
      SetDlgItemText(IDC_EDIT2,Tong);
      UpdateWindow();
    for(iy = MAX; iy < (MAX+3*(m+1));iy++)
    {
            CConvoSimu::data_bit[iy] = 0;
            CConvoSimu::encode(CConvoSimu::data_bit[iy],&output
[iy*n]);
    }
    // modulates input data bits using BPSK
    sprintf(Tong,"At Eb/No: %3.2f dB\r\nModulating(BPSK)...",ebn0);
      SetDlgItemText(IDC_EDIT2,Tong);
      UpdateWindow();
    for(ix = 0;ix < (MAX+3*(m+1));ix++)
      for(iy =0;iy <n;iy++)
          CConvoSimu::modulator(&output[ix*n+iy],&output[ix*n+iy]);
    /* pass modulated signal through channel model*/
```

```
      sprintf(Tong,"At Eb/No: %3.2f dB\r\nPassing signal through
channel...",ebn0);
      SetDlgItemText(IDC_EDIT2,Tong);
      UpdateWindow();
    for(ix = 0;ix < (MAX+3*(m+1));ix++)
      for(iy =0;iy <n;iy++)
      {
          switch(iz){
          case 0:  // AWGN  channel
          case 1:  // AWGN  channel
            if(output[ix*n+iy] == 1)
             output[ix*n+iy] = CConvoSimu::normal_rand(1,noise);
            else
             output[ix*n+iy] = CConvoSimu::normal_rand(-1,noise);
            break;
          case 2: // BSC(Pe)
             if(((double)rand()/RAND_MAX) < Pe)
             { //flip the bit
                 if(output[ix*n+iy] == 1)
                 output[ix*n+iy] = -1;
                 else
                 output[ix*n+iy] = 1;

             }
            }
      }
    // demodulator part
      sprintf(Tong,"At Eb/No: %3.2f dB\r\nDemodulating...",ebn0);
      SetDlgItemText(IDC_EDIT2,Tong);
      UpdateWindow();
    for(ix = 0;ix < (MAX+3*(m+1));ix++)
      for(iy =0;iy <n;iy++)
      switch(iz){
       case 2:        // binary symmetric channel
       case 0: CConvoSimu::Hard_demodulator(output[ix*n+iy],&output
[ix*n+iy]);
             break; // hard decision,AWGN ch.
       case 1: CConvoSimu::Soft_demodulator(output[ix*n+iy],&output
[ix*n+iy]);
             break; // soft decision,AWGN ch.
       }
   sprintf(Tong,"At Eb/No: %3.2f dB\r\nDecoding...",ebn0);
   SetDlgItemText(IDC_EDIT2,Tong);
   UpdateWindow();
   switch(iz){
     case 2:
     case 0: CConvoSimu::decode(output,0);  // using hamming distance
          break;
     case 1: CConvoSimu::decode(output,1);  // using Euclidean
distance
          break;
   }
  }
  sprintf(Tong,"At Eb/No: %3.2f dB\r\nFinding BER...",ebn0);
  SetDlgItemText(IDC_EDIT2,Tong);
  UpdateWindow();
  BER = error/((double)MAX*iv);
  totaldata=(MAX*iv)/8;
  switch(iz){
  case 0:
```

```
  case 1: sprintf(Ong,"At Eb/No: %3.2f dB,BER : %e ,ERROR: %u bits
from data %u bytes\r\n",ebn0,BER,error,totaldata);
            AddDlgItemText(IDC_EDIT1,Ong);
          UpdateWindow( );
            sprintf(Tle,"\r\n%e",BER);
          AddDlgItemText(IDC_HIDE,Tle);
            m_progress2.StepIt();
            break;
  case 2: sprintf(Ong,"At Eb/No: %3.2f dB,p : %e,BER : %e,ERROR: %u
bits from data %u bytes\r\n",ebn0,Pe,BER,error,totaldata);
            AddDlgItemText(IDC_EDIT1,Ong);
            UpdateWindow();
            sprintf(Tle,"\r\n%e",BER);
          AddDlgItemText(IDC_HIDE,Tle);
            m_progress2.StepIt();
  }
 }
 AddDlgItemText(IDC_HIDE,"\r\n];");
 CConvoSimu::pasteplot();
 AddDlgItemText(IDC_EDIT1,"\r\n\r\nAWGN channel:BER for uncoded
BPSK\r\n");
 sprintf(Tong,"At Eb/No: %3.2f dB\r\nSimulating for uncoded
signal...",ebn0);
 SetDlgItemText(IDC_EDIT2,Tong);
 UpdateWindow();
 for (ebn0 = 0 ;ebn0 <= 6 ; ebn0 += 0.5)
 {
   BER = CConvoSimu::Q(sqrt(2*pow(10,ebn0/10)));
   sprintf(Ong,"At Eb/No: %3.2f dB,BER: %e\r\n",ebn0,BER);
   AddDlgItemText(IDC_EDIT1,Ong);

 }
 m_animate1.Stop();
 m_animate1.Close();
 SetDlgItemText(IDC_EDIT2," ");
 SetDlgItemText(IDC_EDIT3," ");
 CTime t2 = CTime::GetCurrentTime();
 CTimeSpan ts = t2 - t1;  // Subtract 2 CTimest=t2-t1;
 t= ts.GetTotalSeconds();
 float tt;
 tt=t/60;
 sprintf(msg,"Simulating Process Succeeded.\n\rTotal times = %d secs
(%2.1f mins).",t,tt);
 MessageBox(msg);
 m_progress2.SetPos(0);


if(MessageBox("Paste&Plot?","Paste&Plot",MB_ICONQUESTION|MB_YESNO)==I
DYES)
       WinExec("matlab.exe",SW_SHOW);

 free(data_bit);
 free(output);
 free_imatrix(F_table,state);
 free_imatrix(B_table,state);

}

void CConvoSimu::pasteplot(void)
{
```

```
      CEdit *textbox;
      AddDlgItemText(IDC_HIDE,"\r\nx=[0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5
5.5 6];");
      AddDlgItemText(IDC_HIDE,"\r\ny2=[0.078933 0.06732 0.056505
0.046591 0.037664 0.029782 0.022978 0.017247 0.012555 0.008832
0.005979 0.003878 0.002398];");
      //AddDlgItemText(IDC_HIDE,"\r\nx=[4 4.5 5 5.5 6 6.5 7 7.5 8 8.5
9 9.5 10];");
      //AddDlgItemText(IDC_HIDE,"\r\ny2=[0.012555 0.008831655
0.005979092 0.00387824 0.002398 0.00140526 0.0007756 0.00040022
0.00019156 0.00008427 0.00003373 0.000012143 0.000003882];");
      AddDlgItemText(IDC_HIDE,"\r\nh=semilogy(x,y1,x,y2,'--');grid
on;");
      AddDlgItemText(IDC_HIDE,"\r\nxlabel('Eb/No (dB)');ylabel('Prob.
of error');");
    AddDlgItemText(IDC_HIDE,"\r\nlegend(h,'Encoded','Uncoded');");


      switch(iz){
    case 0: sprintf(Tle,"\r\ntitle('AWGN channel :Using Hard
Decision,R=1/%d,K=%d');",n,m+1);
                AddDlgItemText(IDC_HIDE,Tle);
               break;
    case 1: sprintf(Tle,"\r\ntitle('AWGN channel :Using Soft Decision
(no quantization),R=1/%d,K=%d');",n,m+1);
                AddDlgItemText(IDC_HIDE,Tle);
              break;
    case 2: sprintf(Tle,"\r\ntitle('BSC  channel :Using Hard
Decision,R=1/%d,K=%d');",n,m+1);
                AddDlgItemText(IDC_HIDE,Tle);
      }

      textbox=(CEdit*)GetDlgItem(IDC_HIDE);
      textbox->SetSel(0,-1);
      textbox->Copy();

}
```

โปรแกรมส่วนประยุกต์ใช้งาน (Practical Application)

```
// ConvoDlg.cpp : implementation file
//

#include "stdafx.h"
#include "afx.h"
#include "Convo.h"
#include "ConvoDlg.h"
#include "ConvoSimu.h"
#include "math.h"
#include "iostream.h"
#include "ComPort.h"
#include "ctype.h"
#include "mscomm.h"
#include "fcntl.h"
#include "Ferr.h"
#include "io.h"
#include "mmsystem.h"
#define NO_ROUTE 30000
#define PORT 0
```

```
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////
////////
// CConvoDlg dialog

CConvoDlg::CConvoDlg(CWnd* pParent /*=NULL*/)
      : CDialog(CConvoDlg::IDD, pParent)
{
      //{{AFX_DATA_INIT(CConvoDlg)
            // NOTE: the ClassWizard will add member initialization
here
      //}}AFX_DATA_INIT
      m_hIcon = AfxGetApp()->LoadIcon(IDI_ICON1);
}

void CConvoDlg::DoDataExchange(CDataExchange* pDX)
{
      CDialog::DoDataExchange(pDX);
      //{{AFX_DATA_MAP(CConvoDlg)
      DDX_Control(pDX, IDC_BUTTON1, m_button);
      DDX_Control(pDX, IDC_ANIMATE, m_animate);
      DDX_Control(pDX, IDC_PROGRESS1, m_progress);
      //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CConvoDlg, CDialog)
      //{{AFX_MSG_MAP(CConvoDlg)
      ON_WM_PAINT()
      ON_WM_QUERYDRAGICON()
      ON_BN_CLICKED(IDENCODE, OnEncode)
      ON_BN_CLICKED(IDC_DECODE, OnDecode)
      ON_BN_CLICKED(IDC_SIMULATE, OnSimulate)
      ON_BN_CLICKED(IDC_ABOUT, OnAbout)
      ON_BN_CLICKED(IDC_BUTTON1, OnComm)
      ON_BN_CLICKED(IDC_FINDERR, OnFinderr)
      //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////
////////
// CConvoDlg message handlers

BOOL CConvoDlg::OnInitDialog()
{
      CDialog::OnInitDialog();

      SetIcon(m_hIcon, TRUE);            // Set big icon
      SetIcon(m_hIcon, FALSE);           // Set small icon
      m_button.SetIcon(AfxGetApp()->LoadIcon(IDI_ICON6));
      k=1,m=5,n=2;Ong = new char[100];
      m_progress.SetRange(0,100);
      m_progress.SetStep(10);
      m_progress.SetPos(0);
      return TRUE;  // return TRUE  unless you set the focus to a
control
}
```

```
// If you add a minimize button to your dialog, you will need the
code below
//  to draw the icon.  For MFC applications using the document/view
model,
//  this is automatically done for you by the framework.

void CConvoDlg::OnPaint()
{
      if (IsIconic())
      {
            CPaintDC dc(this); // device context for painting

            SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(),
0);

            // Center icon in client rectangle
            int cxIcon = GetSystemMetrics(SM_CXICON);
            int cyIcon = GetSystemMetrics(SM_CYICON);
            CRect rect;
            GetClientRect(&rect);
            int x = (rect.Width() - cxIcon + 1) / 2;
            int y = (rect.Height() - cyIcon + 1) / 2;

            // Draw the icon
            dc.DrawIcon(x, y, m_hIcon);
      }
      else
      {
            CDialog::OnPaint();
      }
}

HCURSOR CConvoDlg::OnQueryDragIcon()
{
      return (HCURSOR) m_hIcon;
}

void CConvoDlg::OnEncode()
{
  unsigned long int t;
  char msg[100];
  unsigned long int MAX,pack,ja,iy,last_pack;
  unsigned long int handle,sourcesize;
  xt=1;
  CConvoDlg::curr_state=0;
  if(!OutputFileName.IsEmpty())
        OutputFileName.Empty();
  CConvoDlg::ClearStatus();

  CFileDialog fd(TRUE,NULL,NULL,OFN_OVERWRITEPROMPT,"ALL FILES
(*.*)|*.*||");
  CFileDialog fs(FALSE,NULL,NULL,OFN_OVERWRITEPROMPT,"ALL FILES
(*.*)|*.*||");
  if (fd.DoModal()==IDOK)
      {
            OutputFileName = fd.GetPathName();

      }
  else {
        return;
```

```
    }


    SetDlgItemText(IDC_EDITSOURCE,OutputFileName);
    handle=_open(OutputFileName,_O_RDONLY);
    if( handle == -1 )
    {
     AfxMessageBox( "Open failed on input file." );
     return;
    }

    sprintf(Ong,"\r\nThe read file is %ld bytes long.\r\n",_filelength
(handle));
    AddDlgItemText(IDC_EDIT1,Ong);
    sourcesize=_filelength(handle);
    MAX = 8*_filelength(handle);
    _close(handle);

    ja=MAX/96;    /*   remind that 96 bits/packet      */
    pack=(unsigned long)floor(ja); /*amount of packet -> Total packet
is pack+1*/
    last_pack = MAX-(pack*96);/*amount of bits in last pack*/

    sprintf(Ong,"Total %ld bits\r\n",MAX);
    AddDlgItemText(IDC_EDIT1,Ong);
    sprintf(Ong,"Divided into %ld packets \r\n",pack+1);
    AddDlgItemText(IDC_EDIT1,Ong);
    sprintf(Ong,"96 bits/packet                    -> %ld
packets\r\n",pack);
    AddDlgItemText(IDC_EDIT1,Ong);
    sprintf(Ong,"%d   bits for the last packet -> 1
packet\r\n",last_pack);
    AddDlgItemText(IDC_EDIT1,Ong);
    MessageBox("Please Identify the output file name.","Output for
encoded file",MB_OK|MB_ICONINFORMATION);

    if(!WriteFileName.IsEmpty())
          WriteFileName.Empty();

    if (fs.DoModal()==IDOK)
    {
        WriteFileName = fs.GetPathName();

    }
    else {
          return;
    }
    SetDlgItemText(IDC_EDIT3,WriteFileName);



/*****************************************************************
/
    CTime t1 = CTime::GetCurrentTime();
    CConvoDlg::data_bit = (int *)malloc(sizeof(int)*96);
    CConvoDlg::output = (int *)malloc(sizeof(int)*n*120); //96+(zero 3
bytes)
    if(CConvoDlg::data_bit == NULL || CConvoDlg::output == NULL){
     AfxMessageBox("Memory allocation failure");
     return;}
```

```
 for(iu=1;iu<jh;iu++)
 {
  for(ix=0;ix<state;ix++)
  {
    H=NO_ROUTE;
    tmp = -1;
    for(iy=0;iy<2;iy++)
    {
      idx = CConvoDlg::B_table[ix][iy].state;  //idx :previous state
      distance = s[(iu-1)*state+idx].hamming;
      for(iv=0;iv<n;iv++)
      {
      if(Is_soft)
      { // Soft decision :calculate Euclidean distance
        if(CConvoDlg::B_table[ix][iy].code[iv])
          distance += pow(1-code[(iu-1)*n+iv],2);
        else
          distance += pow(-1-code[(iu-1)*n+iv],2);
      }else
      {  // Hard decision :calculate Hamming distance
       distance += pow(CConvoDlg::B_table[ix][iy].code[iv]-code[(iu-
1)*n+iv],2);
      }
      }
      if(distance<=H)
      {  //choose the branch that has the least hamming distance
       H=distance;
       tmp=idx;
       input=CConvoDlg::B_table[ix][iy].input;
      }
    }

    s[iu*state+ix].prev_state = tmp;
    s[iu*state+ix].hamming = H;
    s[iu*state+ix].input = input;
  }
 }
 idx=0; // set to state 0
 // readout zero-padding length 3 bytes or (24 bits)

 for(ix=LAP+24-1;ix>=LAP;ix--)
   idx = s[(ix+1)*state+idx].prev_state;
 //begin to readout real data part
 for(ix=LAP-1;ix>=0;ix--)
 {
  tmp = s[(ix+1)*state+idx].input;
  idx = s[(ix+1)*state+idx].prev_state;
  CConvoDlg::data_bit[ix] = tmp;
 }
 free(s);

}

void CConvoDlg::ClearStatus()
{
    SetDlgItemText(IDC_EDIT1," ");
      SetDlgItemText(IDC_EDITSOURCE," ");
      SetDlgItemText(IDC_EDIT3," ");
      SetDlgItemText(IDC_EDIT4," ");
}
```

```
void CConvoDlg::OnCancel()
{
      CDialog::OnCancel();
}

class CAboutDlg:public CDialog{
 public:
       CAboutDlg(UINT nID,CWnd* pParent=NULL);
};
CAboutDlg::CAboutDlg(UINT nID,CWnd* pParent):CDialog(nID,pParent)
{

}


void CConvoDlg::OnAbout()
{
    CAboutDlg adlg(IDD_ABOUT,this);
      adlg.DoModal();
}



void CConvoDlg::OnSimulate()
{
  CConvoSimu adlg(IDD_SIMULATION,this);
  adlg.DoModal();

}


void CConvoDlg::OnComm()
{
  CComPort adlg(IDD_PORT,this);
  adlg.DoModal();

}

void CConvoDlg::OnFinderr()
{
  CFerr adlg(IDD_FINDERR,this);
  adlg.DoModal();

}
```

โปรแกรมส่วนส่ง – รับข้อมูลผ่านพอร์ตอนุกรม

```
/* Program for transfering file using software handshaking
   At baud rate = 9600
   No parity bit ,Data byte = 8 bits,Stop bit = 1 bits */
// PORT 0 is COM1,PORT 1 is COM2, and if we change the port
// Don't forget to change the DATA_REG too!
// ***********************************************************//
// Table for port_init function -> code
// 9600 baud,1 stop bit,no parity,8 bit data  -> 227
// 4800 baud,1 stop bit,no parity,8 bit data  -> 199
// ***********************************************************//
// For DATA_REG
// COM1 -> 0x3F8  , (PORT=0)
// COM2 -> 0x2F8  , (PORT=1)
// ***********************************************************//
#define PORT 0
#include <conio.h>
#include <ctype.h>
#include <stdlib.h>
#include <dos.h>
#include <iostream.h>
#include <stdio.h>
#include <fstream.h>
unsigned int DATA_REG = (unsigned int)(0x3F8); //Look at upper table
unsigned  int filesize(FILE *fp);
void sport(int port,unsigned char c),send_file(char *fname),rec_file
();
//void send_file_name(char *f);
//void get_file_name(char *f);
void port_init(int port,unsigned char code);
void wait(int port);
void send_init();
char rport(int port);
int check_stat(int port);
char OutputFileName[100],Choice[5];

main()
{
  clrscr();
  cout<<"Please type 's' to send or any character to receive file->
";
  cin>>Choice;
  //printf("File transfer program in operation. To abort, \n");
  //printf("press any key. \n\n");
  port_init(PORT,227);  /* Initialize begining status of serial
port*/
  if(tolower(Choice[0]) == 's') send_init();
  else rec_file();
  return(0);
}
```

```
/*************************************************************************
*/
/* Sending file  */
void send_init()
{
  cout<< "Enter the location of file to send. ->  ";
  cin>> OutputFileName;
  send_file(OutputFileName);
}

void send_file(char *fname)
{
  FILE *fp;
  int i;
  char ch;
  i=0;
  union {
     char c[2];
     unsigned int count;
  } cnt;
  if (!(fp=fopen(fname,"rb"))) {
    printf("Cannot open input file. \n");
    exit(-1);
  }

  printf("Transmitter is waiting.....\n");

  wait(DATA_REG); //Waiting response for the first contact!!

  printf("Sending %s\n\n",fname);
  cnt.count = filesize(fp);
  printf("Size = %d bytes.\n",cnt.count);
  /*************************************************************/
  sport(DATA_REG,cnt.c[0]);  //Send the size of file(use 2 bytes)
  wait(DATA_REG);            //So we have to send twice.
  sport(DATA_REG,cnt.c[1]);
  /*************************************************************/
  do
  {
    ch=getc(fp);       //Read data from the source file.
    if (ferror(fp))
    {
       printf("Error reading input file.");
       break;
    }
    if(!feof(fp))
    {
      wait(DATA_REG);    //Send data to target.
      sport(DATA_REG,ch);
      i++;
      printf("Data byte#%d has been sent.\r",i);
    }
  }while (!feof(fp));
    wait(DATA_REG);
    fclose(fp);
    printf("\nDone!...\007");
    getchar();

}
/* receiving file */
```

```
void rec_file()
{
   FILE *fp;
   char ch;
   int i;
   i=0;
   union {
       char c[2];
       unsigned  int count;
   } cnt;
   cout<< "Enter the name for receiving file. ->  ";
   cin>> OutputFileName;
   if (!(fp=fopen(OutputFileName,"wb"))) {
     printf("Cannot open output file\n");
     exit(-1);
   }
   printf("\nHit the first contact.\n");
   sport(DATA_REG,'.'); //Hit the first contact!!
   /*************************************************************/
   cnt.c[0] = rport(DATA_REG);  //Receive the size of source file.
   sport(DATA_REG,'.');
   cnt.c[1] = rport(DATA_REG);
   /*************************************************************/
   printf("\nThe size of file has received.\n");
   printf("Total = %d bytes.\n",cnt.count);
   printf("\nSTATUS : HOLD ON! Press enter to continue. :) ");
   getchar();
   printf("\nReceiving file...\n");
   sport(DATA_REG,'.');


   for(;cnt.count; cnt.count--)//Get the data according to size of
file.
   {
     ch = rport(DATA_REG);
     putc(ch,fp);
     i++;
     printf("Data byte#%d has been received.\r",i);
     if(ferror(fp))
     {
       printf("Error writing file");
       exit(-1);
     }
     sport(DATA_REG,'.');
   }
   fclose(fp);
   printf("\n\nDone!.....\n\007");
   getchar();
}

/* calculating the size of file */
unsigned  int filesize(FILE *fp)
{
   unsigned long int i;
   i=0;
   do {
      getc(fp);
      i++;
   } while(!feof(fp));
   rewind(fp);
   return i-1;
```

```
}


/* wait for replying*/
void wait(int port)
{
   if(rport(port) != '.') {
     printf("Communication error (not '.' response)\n");
     exit(-1);
   }
}

/* send data to output port */
void sport(int port,unsigned char c)
{
   outportb(port,c);

}

/* read data from serial port */
char rport(int port)
{
   unsigned char a;
   while(!(check_stat(PORT)&256))
   if(kbhit())
      {
      getchar();
      exit(-1);
      }
   a=inportb(port);
   return a;

}

/* check the status of port*/
int check_stat(int port)
{
   union REGS r;
   r.x.dx = port;
   r.h.ah = 3;
   int86(0x14,&r,&r);
   return r.x.ax;
}

/* Initialize status of port */
void port_init(int port,unsigned char code)
{
   union REGS r;
   r.x.dx = port;
   r.h.ah = 0;
   r.h.al = code;
   int86(0x14,&r,&r);
}
```

กิตติกรรมประกาศ

ปริญญานิพนธ์ชิ้นนี้ สำเร็จลุล่วงได้ด้วยดี ด้วยความช่วยเหลือจากบุคคลหลายๆท่าน ทั้งในด้านความรู้ ความช่วยเหลือ เครื่องและอุปกรณ์ต่างๆ ตลอดจนถึงการสั่งสอน ให้คำแนะนำ อีกทั้งยังคอยเป็นกำลังใจในปริญญานิพนธ์นี้ ซึ่งต้องกล่าวถึง

บิดา มารดา ที่คอยเป็นกำลังใจและสนับสนุนในเรื่องต่างๆ ในการทำปริญญานิพนธ์ชิ้นนี้

อาจารย์กฤษณ์ วงรุจิระ และดร.สุทธิชัย นพนาคีพงษ์ อาจารย์ที่ปรึกษาในการทำปริญญานิพนธ์ที่คอยให้คำแนะนำ ความช่วยเหลือ ให้คำปรึกษาในเรื่องต่างๆ และยังให้ความสนับสนุนในทุกๆด้าน

คุณเศรษฐกร กาเมือง , คุณชนินทร์ มหารักษ์ , ที่คอยให้คำแนะนำต่างๆให้คำปรึกษาในเรื่องข้อมูลที่เป็นประโยชน์อย่างสูงแก่ปริญญานิพนธ์ชิ้นนี้

นายเกรียงศักดิ์ บันดาลศิริกุล ,คุณวริทธิ สันธิเดช , คุณอรนุช เลิศสุวรรณกิจ , คุณวรุฒม์ บุญธรรม , คุณอังคณา โชติเวชกุล ,คุณฤชิษฐ์ หิตะวัฒนกุล , คุณปัฐวิกร พลอยประเสริฐ , คุณภิรดี พิทยาธิกุล , คุณเจ จันทร์ศุภฤกษ์, คุณธีรดา จันทรัศมี ที่คอยให้กำลังใจให้เสมอมานับตั้งแต่เริ่มปริญญานิพนธ์ชิ้นนี้

คณะผู้จัดทำ

21 มีนาคม 2544

เอกสารอ้างอิง

[1] Shu Lin, Daniel J. Costello,JR. 1994. **Error Control Coding: Fundamental and Application,** Prentice Hall.

[2] Djimitri Wiggert, 1988. **Codes for Error Control and Synchronization,** Artech House.

[3] Arnold M. Michelson, Allen H. Levesque, 1984. **Error – Control Techniques for Digital Communication,** A Wiley Interscience Publication

[4] Martin Bossert, 1999. **Channel Coding for Telecommunication,** A Wiley Interscience Publication

[5] Steven Roman, 1996. **Introduction to Coding and Information Theory,** Springer.

[6] Charistian Schlegel, 1997. **Trellis Coding,** IEEE Press.

[7] Thomas M. Cover, Joy A. Thomas, 1991. **Element of Information Theory,** A Wiley Interscience Publication.

[8] Michel Daoud Yacoub, 1993. **Foundation of Mobile Radio Engineering,** CRC Press.

[9] William C.Y. Lee, 1997. **Mobile Communication,** 2nd Edition, McGraw – Hill.

[10] Joel Adams, Sanford Leestma, Larry Nyhoff, 1996. **Turbo C++ An Introduction to Computing,** Prentice Hall.

[11] Herbert Schildt, 1990. **C: The Complete Reference,** 2nd Edition, McGrew – Hill.

[12] John R. Taylor, **An Introduction to Error Analysis,** University Science Books.

[13] Leon W. Couch II, **Digital and Analog Communication Systems,** 5th Edition, Prentice Hall.

[14] ดร. วิทยาเรืองพรวิสุทธิ์, 2538. **คู่มือโปรแกรมภาษา C สำหรับผู้เริ่มต้น,** สำนักพิมพ์ บริษัท ซีเอ็ดยูเคชั่น จำกัดมหาชน.

[15] ขจร โรจนเมธินทร์, 2538. **เทคนิคการเขียนโปรแกรม Turbo C++ สำหรับวินโดวส์,** สำนักพิมพ์ซีเอ็ด

[16] ศ.ดร. วิวัฒน์ กิรานนท์, 2540. **วิศวกรรมสื่อสาร,** พิมพ์ครั้งที่ 1, อักษรสยามการพิมพ์.

[17] ปริญญา เรืองศิริไพศาล, **การวิเคราะห์สมรรถนะของการเข้ารหัสแบบคอนโวลูชัน และการไม่เข้ารหัสของระบบไดเร็กซีเควน CDMA บนช่องสัญญาณนากากามิ,** วิทยานิพนธ์ ปีการศึกษา 2543, ภาควิชาวิศวกรรมโทรคมนาคม, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

[18] นิรุธ อำนวยศิลป์, **คู่มือการเขียนโปรแกรม Microsoft Visual C++ Version 6.0 ฉบับเพื่อการใช้งานจริง,** พิมพ์ครั้งที่ 3, บริษัท ซัคเซส มีเดีย จำกัด.