ปีการศึกษา 2533

การจดจำเสียง
(voice recognition)

โดย

นาย ไพบูลย์ นาคมหาชลาสินธุ์
นาย ภาณุ เกตุมาลา
นาย มนัส ศิริวัฒน์

อาจารย์ที่ปรึกษา

อาจารย์ พูศักดิ์ ชีวสุวิทย์

ปริญญานิพนธ์ปีการศึกษา 2533

ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

เรื่อง การจดจำเสียง (voice recognition)

ผู้จัดทำ

1.นายไพบูลย์ นาคมหาชลาสินธุ์ 30.1200

2.นายภาณุ เกตุมาลา 30.1208

3.นายมนัส ศิริวัฒน์ 30.1214

.....อาจารย์ที่ปรึกษา

(อาจารย์ ฬศักดิ์ ชีวสุวิทย์)

สารบัญ

การจดจำเสียงพูด

ผู้จัดทำ

1.นายไพบูลย์ นาคมหาชลาสินธุ์

2.นายภาณุ เกตุมาลา

3.นายมนัส ศิริวัฒน์

อาจารย์ที่ปรึกษา

รศ.ดร.ฟูศักดิ์ ชิวสุวิทย์

ปีการศึกษา 2533

บทคัดย่อ

ปริญญานิพนธ์นี้ กล่าวถึงการเก็บข้อมูลเสียงพูดในเครื่องไมโครคอมพิวเตอร์ โดยวิธีอินเตอร์รัพท์ เพื่อเก็บลงบนหน่วยความจำ และนำข้อมูลที่เก็บไว้ไปทำการวิเคราะห์โดยใช้เทคนิคการแปลงฟาสต์ฟูเรียร์(FFT) เพื่อเปลี่ยนสัญญาณเสียงอินพุทในโดเมนเวลาให้เป็นสเปคตรัมในโดเมนความถี่ และหาคุณลักษณะของเสียงพูดโดยการตรวจจับความถี่จุดสุ่มที่มีขนาดสูงสุดในโดเมนความถี่ และแสดงผลเสียที่เกิดจากการรั่วไหลของลอนช้าง เมื่อใช้ฟังก์ชันถ่วงน้ำหนักข้อมูลแบบสี่เหลี่ยม

ABSTRACT

This thesis describes a speech-data storage method in microcomputer by interrupting to keep into memory. The stored data are analysed by using " Fast Fourier Transform (FFT) " technique to transform input speech signal in time domain to spectrum in frequency domain and find the feature of speech signal by detecting the maximum peak of all frequency sampled points. Then showing the leakage affective from sidelobe when using rectangular data-weighting function.

บทที่ 1

บทนำ

เมื่อมนุษย์เปล่งเสียงที่เป็นคำพูดออกมา มนุษย์ด้วยกันเองสามารถที่จะจำความหมายของคำที่เปล่งออกมาได้เมื่อมนุษย์นั้น ๆ อยู่ในสิ่งแวดล้อมเดียวกัน พื้นฐานเดียวกัน เนื่องมาจากมนุษย์มีสมองที่มีความสลับซับซ้อน และมีประสิทธิภาพในการจดจำ การคิดเปรียบเทียบที่เหนือกว่าสัตว์ และเครื่องจักรกล ดังนั้นมนุษย์จึงพยายามที่จะให้เครื่องจักร สามารถที่จะทำการเปรียบเทียบ และคิดได้ใกล้เคียงกับคนจึงทำให้เกิดการวิเคราะห์เสียงเพื่อหาคุณลักษณะที่สำคัญของสัญญาณเสียงที่จะมาเปรียบเทียบความแตกต่างของสัญญาณเสียงที่เกิดขึ้น เพื่อให้ทราบว่าเป็นคำใด ซึ่งนั่นหมายถึงว่าเครื่องจักรสามารถจะจดจำได้

ในปริญญานิพนธ์นี้จะกล่าวถึงเรื่องต่างๆที่เกี่ยวข้องกับสัญญาณเสียงพอจะกล่าวโดยสรุปได้ดังนี้

ในบทที่ 2 เป็นเกี่ยวการสุ่มตัวอย่างและการเข้ารหัสของเสียง เพื่อที่จะสามารถนำสัญญาณเสียงนั้นวิเคราะห์ได้ง่ายขึ้น

ในบทที่ 3 จะกล่าวถึงการได้ยินและการรับรู้เสียงของมนุษย์ รวมถึงโครงสร้างทางกายภาพของการได้ยิน

ในบทที่ 4 และ 5 จะกล่าวถึงวิธีการวิเคราะห์โดยใช้การแปลงฟูเรียร์และอัลกอริทึมฟาสต์ฟูเรียร์เข้าช่วยในการวิเคราะห์

ในบทที่ 6 และ 7 กล่าวถึงการ วิเคราะห์และ การจดจำ ว่ามีรูปแบบคะไรบ้างและมีวิธีการอย่างไร

ส่วนในบทท้ายๆ จะกล่าวถึงการทดลอง ผลการทดลอง ปัญหาที่เกิดขึ้นจากการทำงานและสรุปผล

ส่วนของโปรแกรมและรายละเอียดของวงจรได้แสดงไว้ในภาคผนวก

บทที่ 2

ทฤษฎีอัตราการสุ่ม
และการเข้าระหัสข้อมูลแบบพัลส์โค้ดมอดดูเลชัน

2.1 ทฤษฎีอัตราการสุ่ม (Sampling rate theory)

ในระบบที่ต่อเนื่องอินพุท เอาท์พุทและตัวแปรอื่นๆในระบบทั้งหมดในระบบทั้งหมดเราสามารถที่จะวัดได้เมื่อเวลาใดๆ และสามารถเปลี่ยนแปลงไปได้อย่างต่อเนื่องได้ตามเวลา ในระบบที่ไม่ต่อเนื่องนั้นตัวแปรต่างๆในระบบเราไม่อาจคาดคะเนได้ตลอดช่วงเวลา เพราะระบบทำงานไม่ต่อเนื่อง หรือทำงานเป็นช่วงที่แน่นอน เมื่อเราหาค่าตัวแปรใดๆในเวลาช่วงหนึ่งๆ ก็จะได้ตัวแปรค่าหนึ่งออกมา ซึ่งค่าที่ได้มานี้อาจจะเป็นค่าที่อาจใกล้เคียงกับค่าจริงๆในระบบที่ต่อเนื่องนั้น เพราะมันจะมีค่าเท่ากันกับในระบบต่อเนื่อง เฉพาะในขณะที่ทำการแปลงข้อมูลเป็นดิจิตอลเท่านั้น และจะเป็นจริงก็ต่อเมื่อมีข้อมูลขนาดจำนวนบิทเท่ากับอนันต์เท่านั้น ดังนั้นจึงเป็นการยากลำบากหอสมควรที่จะนำเอา คอมพิวเตอร์ไปทำงานกับสัญญาณหรือระบบที่เปลี่ยนไปได้ระบบ อย่างต่อเนื่องตามเวลาเพราะว่าการทำงานของคอมพิวเตอร์นั้นเป็นแบบการทำงานที่ไม่ต่อเนื่อง ฉะนั้นจึงมีความจำเป็นที่สัญญาณของระบบที่ต่อเนื่องเหล่านั้นจะต้องถูกทำการสุ่มตามช่วงเวลาที่ไม่ต่อเนื่องราวว่ามันเองก็การทำงานที่ไม่ต่อเนื่องเช่นกัน เราจะเรียกการทำงานของระบบที่ทำการสุ่มข้อมูล (sample-data system) ซึ่งสัญญาณที่ต่อเนื่องหรือสัญญาณอนาลอกใดๆ ซึ่งถูกสุ่มและเปลี่ยนให้เป็นข้อมูล ในเชิงตัวเลขว่า การดิจิไตซ์ จากที่ได้กล่างมาทั้งหมดนี้ เรารู้จักกันในรูปของการแปลงสัญญาณ อนาลอกเป็นสัญญาณดิจิตอล ส่วนในกระบวนการทำงานที่ตรงกันข้ามกับกระบวนการดังกล่าว ก็คือ การแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอลนั่นเอง

สัญญาณที่ถูกนำเข้ามาผ่านระบบทำการสุ่มข้อมูล บางทีก็มีความหมายเป็นลำดับ (sequence) แล้วแต่ว่าจะเป็นลำดับเข้าหรือลำดับออก โดยธรรมชาติแล้งสัญญาณใดๆนั้นมีความยาวเป็นอนันต์ ดังนั้นเราจึงมักจะใช้เวกเตอร์ เป็นตัวที่จะใช้แยกข้อมูลให้มีความยาวที่แน่นอน ซึ่งจะทำให้ การวิเคราะห์นั้นสามารถทำได้ง่ายขึ้น

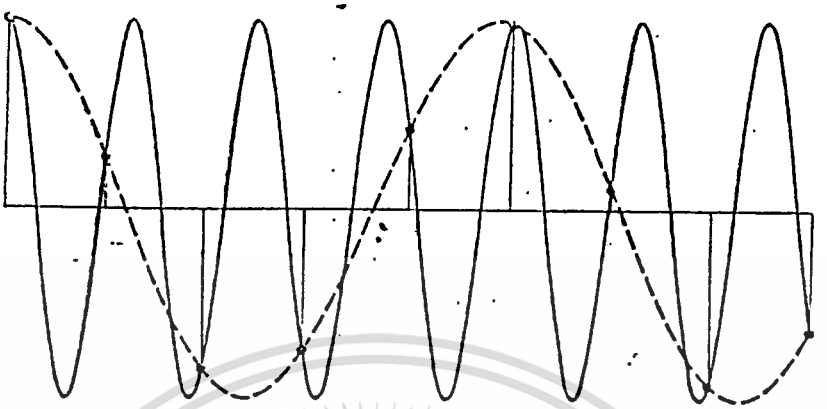
โดยการคำนวณด้วยระบบดิจิตอล

จากทฤษฎีดังกล่าวทำให้เราอ้างได้ว่า ข้อมูลเชิงตัวเลขที่เราสุ่มมาเพื่อเป็นตัวแทนสัญญาณต่างๆทางอนาลอกนั้นย่อมมีจุดที่สุดที่แน่นอน หรือเราสามารถใช้ข้อมูลเชิงตัวเลขต่างๆนั้นเป็นตัวแทนสัญญาณต่างๆ ทางอนาลอกได้เสมอและมีจำนวนที่แน่นอนที่สามารถควบคุมได้ เราเรียกความแตกต่างระหว่างข้อมูลเชิงตัวเลขที่เราสามารถกำหนดได้นี้ว่า ข้อมูลควันไตซ์(quantize)

ซึ่งจากการสุ่มข้อมูลออกมาแบบนี้ก็อาจเกิดความคลาดเคลื่อนบ้าง เราเรียกความผิดพลาดทางควันไตซ์ ซึ่งความผิดพลาดทางควันไตซ์นี้เป็นปัญหาในเรื่องสัญญาณรบกวน จะเห็นได้ว่าเป็นสิ่งที่เกิดขึ้นปกติของการทำงานแบบดิจิตอลเอง และมักก่อให้เกิดความไม่เที่ยงตรงในระบบที่ทำงานเกี่ยวกับสัญญาดิจิตอลนี้

ในระบบที่เรากำลังศึกษานี้จะใช้ช่วงเวลาที่ทำการสุ่มข้อมูลแต่ละครั้งเท่ากันเสมอฉะนั้นอัตราสุ่มข้อมูลของระบบจะเท่ากับส่วนกลับของเวลาที่แตกต่างกันระหว่างการสุ่มข้อมูลแต่ละครั้ง

ในการสุ่มข้อมูลจากสัญญาณทางอนาลอกใดๆ ควรจะประมาณของอัตราสุ่มไว้ให้สูงพอสมควร เพราะถ้าใช้อัตราสุ่มที่ค่อนข้างต่ำ อาจก่อให้เกิดปัญหาในระหว่างที่ทำการสุ่มข้อมูลจากสัญญาณต่อเนื่อง เพราะในสัญญาณนั้นต่างๆอาจมีข้อมูลต่างๆที่สำคัญมาก เช่นสัญญาณในหัวใจในคนไข้ที่มีปัญหาอาจมีบางช่วงที่มีลักษณะของความถี่สูง หากใช้อัตราสุ่มที่มีค่าสูงกว่าสัญญาณหัวใจเพียงสองหรือสามเท่า สัญญาณดังกล่าวก็จะไม่ได้ถูกสุ่มเข้ามาหรือระบบทำการสุ่มไม่สามารถเก็บรายระเอียดของสัญญาณได้อย่างเพียงพอ ผลลัพธ์ที่ตามมาก็คือ เมื่อเราทำการแปลงข้อมูลกลับไปเป็นสัญญาณอนาลอกแล้ว เราจะได้สัญญาณที่ผิดเพี้ยนไปจากสัญญาณต้นฉบับมาก ในทฤษฎีการสุ่มข้อมูลจากสัญญาณที่ต่อเนื่อง กำหนดให้ใช้อัตราสุ่มอย่างน้อยสองเท่าของความถี่สูงสุดของสัญญาณที่เราจะนำมาทำการสุ่ม เราเรียกอัตราการสุ่มที่น้อยที่สุดที่ยอมนับได้นี้ว่า NYQUIST RATE แต่หากต้องการรายละเอียดของสัญญาณสูง เช่นในการวิเคราะห์สัญญาณหัวใจต้องเลือกอัตราสุ่มสูงมากกว่าสองเท่าขึ้นไป ดังนั้นสำหรับสัญญาณต่างๆ ที่จะนำมาทำการสุ่มข้อมูลที่ความถี่ประมาณ 3-4 กิโลเฮิรตซ์ อัตราการสุ่มที่ควรจะใช้มากสุดคือ 8-10 กิโลเฮิรตซ์



รูปที่ 2.1 *แสดงตัวอย่างของการกำหนดอัตราสุ่มที่ต่ำกว่ามาตราฐาน เส้นหนา แสดงถึงสัญญาณอินพุทที่ความถี่ประมาณ 795 Hz จุดแสดงถึงการสุ่มที่ ความถี่การสุ่มที่ความถี่ประมาณ 1 kHz เส้นประแสดงถึงสัญญาณเอาท์พุทที่ ได้ออกมา มีความถี่เท่ากับผลต่างของอัตราสุ่มกับความถี่จองสัญญาณอิน พุทประมาณ 1000-795 =205 Hz*

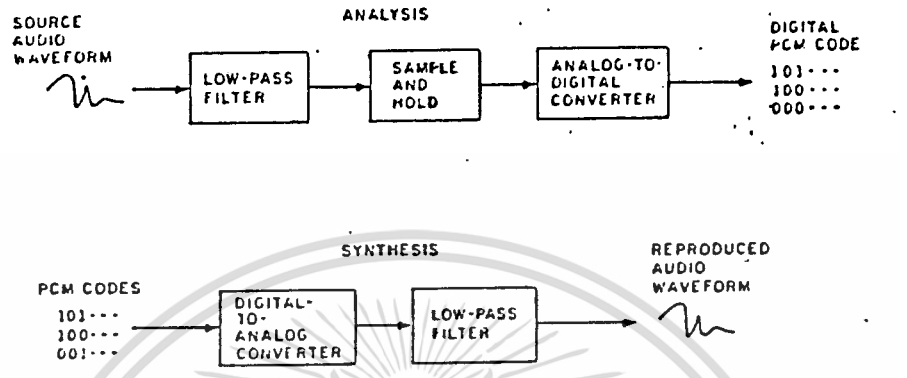
ในระบบใดๆที่ใช้อัตราสุ่มน้อยกว่า NYQUIST RATE ที่เป็นมาตรฐานนั้นเราเรียกว่า Undersampling สำหรับสัญญาณที่ถูกสุ่มแบบ Undersampling นั้น จะได้ผลลัพธ์ออกมาเป็นสัญญาณใหม่ที่มีความถี่เท่ากับผลต่างของความถี่ของอัตราสุ่มกับความถี่ของสัญญาณที่ป้อนเข้ามา ซึ่งจะมีความเพี้ยนไปจากเดิมอย่างมากดังตัวอย่างดังรูป

แต่ถ้ากำหนดให้อัตราสุ่มมีค่าสูงเกินไป ก็ไม่ใช่ผลดีขึ้นมาเท่าไรนัก เพราะปริมาณข้อมูลที่ทำการสุ่มออกมาได้นั้น อาจจะมากเกินความจำเป็น ซึ่งทำให้สิ้นเปลืองเนื้อที่ของหน่วยความจำอย่างมาก ฉะนั้น เราควรเลือกความถี่ของอัตราสุ่มให้ถูกต้องตามมาตราฐาน และเมื่อเลือกใช้อัตราสุ่มค่าหนึ่งแล้ว เราอาจจะใช้อุปกรณ์ประเภทกรองความถี่ผ่านได้ด้วย เพื่อป้องกันให้ข้อมูลที่สำคัญๆพลาดไป

โดยการออกแบบวงจรกรองความถี่ที่มากว่าครึ่งหนึ่งของอัตราสุ่มที่เรากำหนดอัตราสุ่ม (Fs/2) แต่ในการออกแบบวงจรกรองความถี่ผ่านนั้น ไม่ควรให้จุดที่ให้คัทออฟนั้น ทำการคัทออฟอย่างรวดเร็วเกินไป ควรเลือกให้การตอบสนองของวงจรกรองความถี่ผ่านนั้นลดลงอย่างช้าๆจะดีกว่ามาก

2.1 การเข้ารหัสแบบพัลส์โค้ดมอดดูเลชัน(pulse code modulation)

พัลส์โค้ดมอดดูเลชัน คือวิธีที่ใช้กันทั่วไปในการแทนสัญญาณทางอนาลอกให้อยู่ในรูปแบบสัญญาณทางดิจิตอล สำหรับจุดใดๆบนสัญญาณทางอนาลอกที่ต่อเนื่องกันจะถูกแทนได้เป็นข้อมูลทางดิจิตอลค่าหนึ่งในระบบเลขฐานสอง เมื่อไรที่ค่าหนึ่งบนสัญญาณทางอนาลอกถูกทำการถูกทำการวัด ค่าๆนั้นจะนำมาเปรียบเทียบว่าจะต้องแทนด้วยโค้ดอะไรในระบบเลขฐาน 2 ซึ่งเรียกว่า พัลส์โค้ด ถ้าหากว่าค่านั้นอยู่ระหว่างสองค่าในระบบเลขฐานสอง จะถือว่า ค่าเลขฐานสองที่อยู่ใกล้มากที่สุดนั้นเป็นโค้ดสำหรับแทนสัญญาณจุดนั้นของสัญญาณอนาลอกที่ทำการอินพุทเข้า วิธีการแบบนี้เรียกว่า ควันไตเซชั่น(Quantization) การแบ่งคลื่นออกเป็นช่วงย่อยๆ โดยแต่ละช่วงจะถูกกำหนดให้เท่ากับโค้ดในระบบเลขฐานสองค่าหนึ่ง ซึ่งเรียกว่าพัลส์โค้ด ฉะนั้นในการวิเคราะห์สัญญาณเสียงใดๆเมื่อสัญญาณเสียงผ่านกระบวนการเหล่านี้แล้วชุดข้อมูลที่เป็น พัลซ์ โค้ด จะถูกส่งออกมาอย่างต่อเนื่องเป็นขบวน ผลที่ได้นี้คือ สัญญาณพัลส์โค้ดมอดดูเลชั่น แต่เพราะว่าสัญญาณเสียงที่จะนำมาวิเคราะห์และทำการบันทึกไว้นั้นจะถูกนำมาเก็บในรูปของสัญญาณดิจิตอล ฉะนั้นในการที่จะนำเอาสัญญาณเสียงนั้นๆมาผ่านกระบวนต่างๆเราจึงต้องทำการปรับปรุงแต่งสัญญาณนั้นให้เหมาะสมเสียก่อนซึ่งจะแสดงวิธีการต่างๆดังรูป 5.1 ซึ่งแสดงถึงแผนผังการทำงานของเครื่องสังเคราะห์สัญญาณเสียงหรือ Speech Synthesizer ซึ่งเครื่องนี้จะทำการแปลงสัญญาณเสียงที่เราต้องการที่จะเก็บบันทึกไว้ให้อยู่ในรูปของข้อมูลทางดิจิตอลที่เราเรียกว่า พัลส์โค้ดมอดดูเลชั่น



รูปที่2.2 แสดงไดอะแกรมของระบบการวิเคราะห์และสังเคราะห์สัญญาณเสียงโดยใช้วิธี พัลส์โค้ดมอดดูเลชั่น

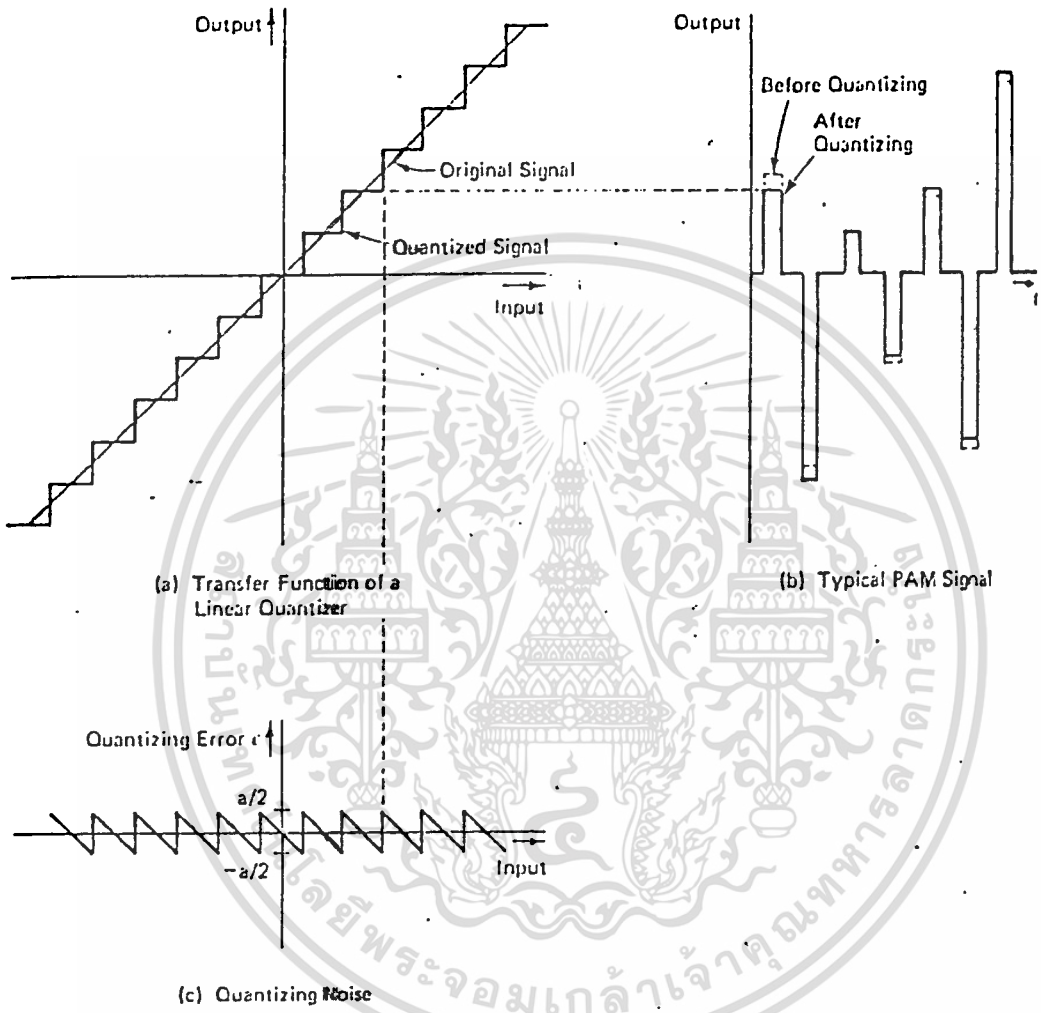
จากรูปสามารถอธิบายการทำงานได้ดังนี้คือ สัญญาณเสียงโดยทั่วไปซึ่งบางส่วนของสัญญาณอาจมีความถี่สูงเกินไป ทำให้ผลต่างของระดับสัญญาณระหว่างสองจุดใดๆ ในขณะทำการสุ่มมีค่ามากเกินกว่าค่าสูงสุดของพัลส์โค้ด ที่เราได้กำหนดไว้ มักทำให้เกิดความผิดเพี้ยนของสัญญาณขึ้นได้ที่เราเรียกว่า Compliance error ฉะนั้นเราจึงต้องนำเอาสัญญาณที่เราต้องการจะวิเคราะห์และบันทึกไว้มาผ่านวงจรกรองความถี่ต่ำผ่าน ซึ่งจะสามารถช่วยลดปัญหานี้ได้เพราะสัญญาณความถี่สูงได้ถูกกรองทิ้งไปแล้ว จากนั้นสัญญาณที่ได้จะถูกนำมาเข้าวงจรส่วนที่เรียกว่าวงจรสุ่มและหน่วงสัญญาณ(sample&hold-circuit) เหตุที่ต้องใช้วงจรสุ่มและหน่วงสัญญาณนี้ก็เพราะในวงจรนี้เราได้เลือกใช้อุปกรณ์ที่ทำการแปลงสัญญาณอนาลอกให้เป็นสัญญาณดิจิตอลมที่ทำงานแบบซักเซสทีฟแอมพรอกซิเมชั่น (sucessice approximation) ซึ่งอุปกรณ์แปลงสัญญาณนี้จะทำการแปลงสัญญาณออกมาทีละบิท แล้วกลับไปเช็คสัญญาณอินพุทใหม่อีกแล้วทำการแปลงไล่ไปจนครบทุกบิท จึงจะถือว่าเสร็จกระบวนการทำงานใน 1 ครั้ง ถ้าหากเราไม่มีวงจรสุ่มและหน่วงสัญญาณไว้แล้วจะทำให้สัญญาณทางอินพุทของ ADC ไม่ใช่สัญญาณ

เดิม ในระหว่างกระบวนการทำการแปลงข้อมูลเหล่านั้นกลับไปเป็นสัญญาณเสียง มักจะมีความผิดเพี้ยนขึ้นมาเสมอ จากนั้นข้อมูลทางดิจิตอลที่ได้รับออกมาแต่ละไบท์จะนำมาผ่านกระบวนการทางซอฟแวร์โดยไมโครโปรเซสเซอร์ เพื่อทำการแปลงสัญญาณออกมาให้อยู่ในรูปของ พัลส์โค้ดแล้วนำข้อมูลเหล่านี้ไปเก็บไว้ในหน่วยความจำ มาถึงจุดนี้ก็คือเสร็จสิ้นขั้นตอนในการวิเคราะห์สัญญาณเสียงเพื่อเก็บบันทึกในรูปแบบของสัญญาณดิจิตอลแบบ พัลส์โค้ดมอดดูเลชั่น

สำหรับขั้นตอนในการสังเคราะห์สัญญาณเสียงก็คือการนำข้อมูลพัลซ์ โค้ดที่เรามีอยู่ในหน่วยความจำมาทำการแปลงกลับให้เป็นสัญญาณเสียงทำได้โดยการนำเอาข้อมูลที่เก็บไว้ซึ่งจะถูกเก็บไว้ในลักษณะ พัลซ์ โค้ด มาผ่านกระบวนการแปลงสัญญาณเหล่านั้นกลับออกมาให้อยู่ในรูปของข้อมูล 8 บิท โดยวิธีการทางซอฟแวร์ จากนั้นจึงนำเอาข้อมูลเหล่านั้นมาผ่านวงจรแปลงสัญญาณดิจิตอลให้เป็นสัญญาณอนาลอก แต่สัญญาณที่ออกมาจากวงจรแปลงสัญญาณดังกล่าวมักจะไม่เรียบ หรือเป็นสัญญาณที่มีความเป็นเชิงเส้นในลักษณะทางสัญญาณอนาลอกเท่าที่ควร จึงมีความจำเป็นที่จะต้องนำเอาวงจรกรองความถี่ต่ำผ่านมาใช้ เพื่อจะทำให้สัญญาณที่ได้มานั้นมีความเป็นเชิงเส้นอย่างสมบูรณ์ จึงเสร็จสิ้นทุกขั้นตอนในการทำงานของระบบการวิเคราะห์และสังเคราะห์สัญญาณเสียงโดยวิธี พัลส์โค้ดมอดดูเลชั่น

หลักจากการสุ่มข้อมูล ขนาดของแอมพลิจูดที่ได้ จะเรียกว่า ควันไตซ์ (quantize) ซึ่งมีขนาดแบบไม่ต่อเนื่อง สัญญาณใหม่ที่ไพด้นี้จะเป็นสัญญาณแบบคร่าวๆของสัญญาณเดิม ซึ่งแน่นอนคือสัญญาณทั้งสองย่อมไม่เหมือนกันทุกส่วน ทำให้เกิดความผิดพลาดขึ้น ความแตกต่างระหว่างสองสัญญาณนี้เรียกว่า quantization error ในระบบการส่งข้อมูลแบบพัลส์โค้ดมอดดูเลชัน สัญญาณรบกวนที่เกิดจากการควันไตซ์(quantizing noise) เป็นสัญญาณรบกวนที่สำคัญที่สุดซึ่งต้องระวังอย่างมาก

สัญญาณรบกวนที่เกิดจากควันไตซ์นี้ สามารถลดได้โดยการลดขนาดของช่วงการควันไตซ์ลง หรือ เพิ่มระดับให้มากขึ้นอีก สำหรับการสื่อสารนั้นมักจะใช้การเข้ารหัสในรูปของข้อมูล 8 บิท ซึ่งมีอยู่ทั้งหมด 256 ระดับ



รูปที่2.3 *แสดงการควันไตซ์สัญญาณอนาลอกและความผิดพลาดจากการควันไตซ์*

ถ้าใช้รหัส N บิท จะได้ช่วงการควันไตซ์ทั้งหมด 2^n ระดับ ซึ่งได้ค่า

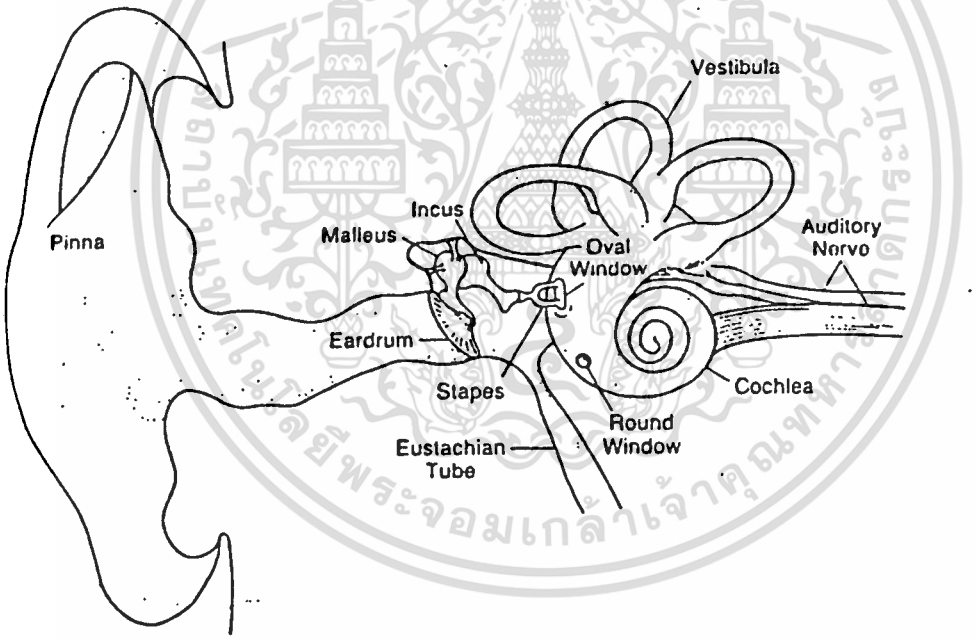
rms S/N ratio = 1.8 + 6N เดซิเบล

ยกตัวอย่างเช่น การเข้าระหัส 8 บิท อัตราส่วนของสัญญาณกันน้อยส์คือ 49.8 เดซิเบล สำหรับสมการนี้ได้กำหนดแอมพลิจูดที่สูงที่สุดที่ระบบจะรับได้โดยไม่ถูกขลิบ(clip)

บทที่ 3
การได้ยินและการรับรู้เสียงของมนุษย์

3.1การได้ยิน(Hearing)

ระบบการฟังหมายถึงสิ่งที่ได้รับมาถูกเปลี่ยนไปเป็นสัญญาณกระตุ้น กระตุ้นไปยังระบบประสาท เริ่มจากโครงสร้างของหูซึ่งแบ่งออกเป็น 3 ส่วน ได้แก่ หูชั้นนอก หูชั้นในและหูชั้นใน



รูปที่3.1 *แสดงส่วนประกอบของหู*

3.1.1 หูชั้นนอก

หูชั้นนอกประกอบไปด้วยใบหู(pinna) ซึ่งสามารถเห็นจากภายนอกได้ ท่อนำเสียง(external canal หรือexternal auditory meatus) และ

แก้วหู (eardrum) ใบหูเป็นส่วนปิดกั้นจากสิ่งต่างๆภายนอก และเป็นตัวรับสัญญาณเสียง ท่อนำเสียงจะมีลักษณะค่อนข้างเป็นท่อตรง ขนาดยาวประมาณ 2.7 cm และกว้าง 0.7 cm เพื่อที่จะนำเสียงสู่แก้วหูต่อไป ท่อนี้มีความรีโซแนนท์ที่ประมาณ 3 kHz ซึ่งเป็นช่วงของความถี่เสียงคนพูดปกติ แก้วหูเป็นโครงสร้างที่มีลักษณะเป็นรูปกรวย อยู่ปลายสุดของท่อนำเสียง แก้วหูจะสั่นเมื่อได้รับการตอบสนองจากสัญญาณเสียง และเป็นการเชื่อมโยงสัญญาณอันดับแรกของโครงสร้างในการส่งผ่านเสียงไปสู่ตัวเปลี่ยนเป็นสัญญาณของระบบประสาทในหูชั้นใน

3.1.2 หูชั้นกลาง

หูชั้นกลางเป็นช่องที่มีอากาศบรรจุอยู่ภายใน ซึ่งแยกออกจากหูชั้นนอกโดยแก้วหูและต่ออยู่กับหูชั้นใน ซึ่งเป็นช่องเปิด 2 ช่อง ที่เรียกว่า ช่องรูปวงรีและรูปกลม (oval and round windows) หูชั้นกลางนี้ยังต่ออยู่กับส่วนภายนอกหูโดยผ่านทางท่อยูสเตเชียน (eustachaintube)ซึ่งเป็นตัวปรับสมดุลของความดันอากาศ ระหว่างหูชั้นกลางและบรรยากาศภายนอก

หูชั้นกลางประกอบด้วยกระดูกชิ้นเล็กๆ 3 ชิ้น ซึ่งจะเป็นตัวผ่านของสัญญาณระหว่างแก้วหูและช่องรูปวงรี กระดูกเหล่านี้เรียกว่ากระดูกค้อน (malleus or hammer)กระดูกทั่ง(incus or anvil) กระดูกโกรน (stapes or stirrup) กระดูกค้อนจะติดกับแก้วหู กระดกโกรนติดกับช่องรูปวงรี และกระดูกทั่งจะติดกับกระดูกทั้งสองชนิด

หน้าที่ของกระดูกเหล่านี้คือเป็นตัวกั้นการส่งผ่านของสัญญาณและเป็นตัวจำกัดแอมปลิจูด

- ตัวกั้นการส่งผ่านสัญญาณ(Impedance transformation) ช่วยให้การส่งพลังงานเสียงจากอากาศไปสู่ของเหลวในหูชั้นในให้มีประสิทธิภาพมากขึ้น

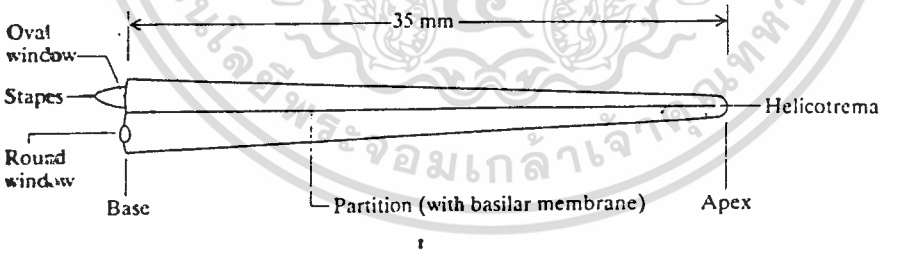
- การจำกัดและป้องกันระดับเสียงที่สูงเกินไป ซึ่งจะทำโดยกล้ามเนื้อของหูชั้นใน

3.1.3 หูชั้นใน

หูชั้นในประกอบด้วยอวัยวะที่เกี่ยวกับการได้ยิน 2 ส่วนคือ

- Cochlea หมายถึงอวัยวะรับเสียงขดแบบก้นหอย อยู่ภายในหลอดเยื่อคดเคี้ยวเรียกว่า Membranous labyrimth ซึ่งภายในมีน้ำเหลืองเรียกว่า Endolymph ทำหน้าที่รับคลื่นเสียงที่สั่นสะเทือนไปชนอวัยวะที่เรียกว่า organ of corti มีเส้นประสาทสมองคู่ที่ 8 มาเลี้ยงและรับคลื่นเสียงไปสู่สมองอีกต่อหนึ่ง

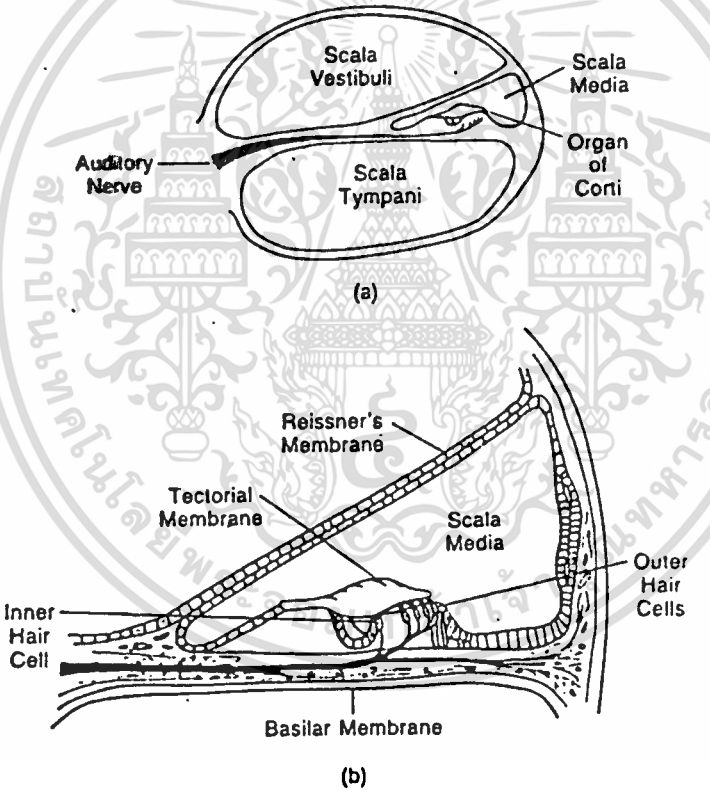
- Semicircular canal หมายถึงอวัยวะเกี่ยวกับการทรงตัว มีลักษณะเป็นหลอดครึ่งวงกลม 3 วง (กระดูกอ่อน) ซึ่งตั้งฉากซึ่งกันและกัน ติดอยู่กับถุงที่เรียกว่า Ultriculus ภายในกระดูกครึ่งวงกลม 3 วงนี้จะมี Endolymph เมื่อเรขยับเอียงไปทางใดก็จะเอียงไปทางนั้นเพื่อรักษาสมดุลย์ของร่างกาย และบอกความรู้สึกไปสู่สมองทาง เส้นประสาทรับความรู้สึก (Auditor nerve)



รูปที่ 3.2 *แสดงลักษณะการแผ่ออกของ Cochlea*

Cochlea มีรูปร่างคล้ายก้นหอย สัญญาณการสื่อสารผ่านมาจากหูชั้นกลางทางช่องเปิดกลมและรูปวงรี ซึ่งจะมีตัวแปลงสัญญาณเสียงจากการสั่น เป็นสัญญาณกระตุ้นระบบประสาท ถ้าเอา Cochlea มาแผ่ออกจะได้ดังรูปที่ 3.9 และภาคตัดขวางจะได้ดังรูป 3.10 cochlea จะถูกแบ่งแผ่นเยื่อที่ยืดหยุ่น ที่เรียกว่า basilar membrane และมีเยื่อที่บางกว่าที่เรียกว่า Peissner'membrane

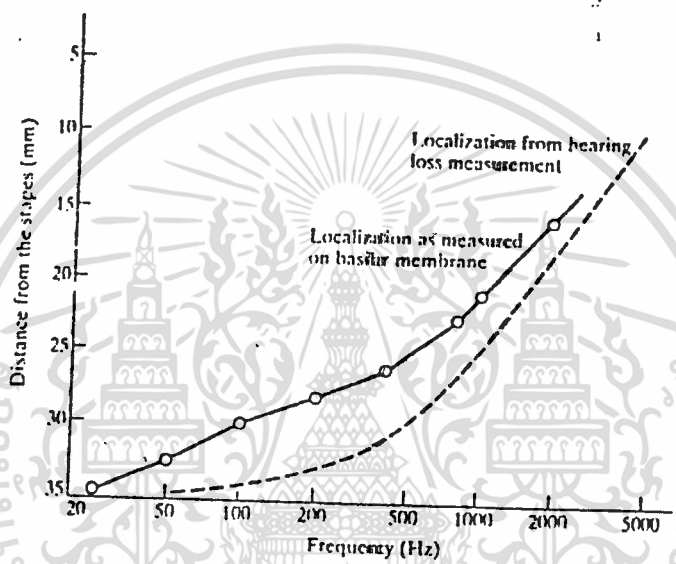
ส่วนที่กั้นคู่นี้จะแบ่ง cochlea ออกเป็น 2 ส่วน คือ scala vestibali และ scala tympani สองส่วนนี้จะต่อกันที่บริเวณปลายเปิดของ cochlea ที่เรียกว่า helicotrema พลังงานเสียงผ่านทางช่องรูปวงรีที่ซึ่งได้รับสัญญาณจากกระดูกโกรนอีกทีหนึ่ง เสียงจะเดินทางลงสู่ด้านหนึ่งของ cochlea (scala vestibali) ผ่านไปยังอีกด้านหนึ่ง (scala tympani) โดยผ่านทาง helicotreme และออกไปที่ช่องรูปกลม แผ่นเยื่อบาร์ซิลล่า (basilar membrane) จะสั่นเมื่อได้รับผลของเสียง



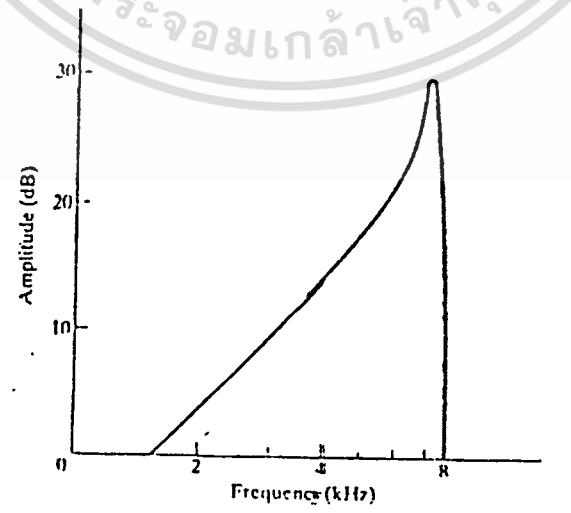
รูปที่ 3.3 (a)*แสดงภาพตัดขวางของ Cochlea*

(b)*แสดงโครงสร้างภายในของ Scala media*

ส่วนที่ทำหน้าที่เปลี่ยนสัญญาณเสียง (organofcorti) และ tectorial membrane จะวางตัวตามแนวยาวของแผ่นเยื่อบาร์ซิลล่า โดยที่ tectorial membrane จะอยู่เหนือ organofcorti ซึ่ง tectorial membrane จะมีเซลขน (haircell) ซึ่งกระจาย อยู่บริเวณช่องว่างระหว่าง organ ofcorti และ tectorialmembrane ขนเหล่านี้จะรับการสั่นสะเทือนของแผ่นเยื่อส่งผ่านไปยังส่วนของเซลขน ซึ่งเป็นที่เชื่อมต่อไปยังเส้นประสาท ที่จะส่งสัญญาณไปยังสมอง



รูปที่ 3.4 *ตำแหน่งของแอมพลิจูดสูงสุดตามเยื่อบาร์ซิลล่า*



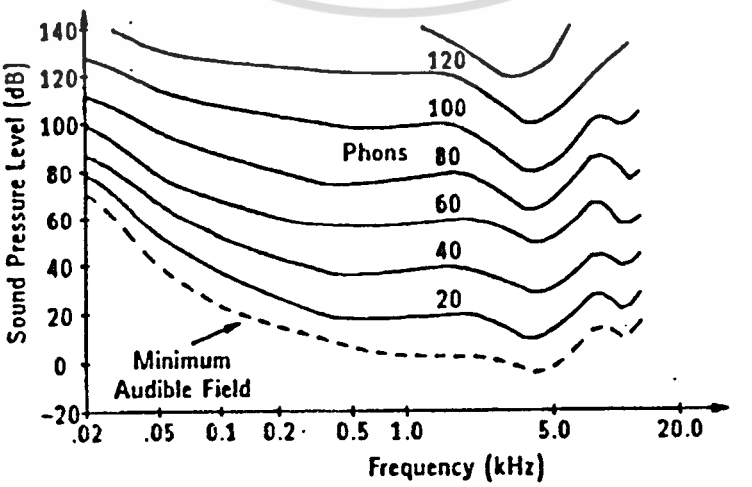
รูปที่ 3.5 *แสดงถึงการตอบสนองความถี่ของจุดบนเยื่อบาร์ซิลล่า*

หน้าที่การทำงานของโครงสร้างช่วยในการกระจายไปสู่ที่ว่างของความถี่ที่ประกอบขึ้นตามความยาวของแผ่นเยื่อบาร์ซิลล่าจากการวิเคราห์การสั่นเนื่องจากการตอบสนองของเสียงถึงจุดสูงสุดที่จุด 1จุด ตามความยาวของมันและจุดนี้จะเป็นจุดที่มีค่าความถี่สูงสุดของความถี่ ของฟังก์ชันเสียง จุดที่ขนาดสูงสุดนำไปวาดกราฟกับความถี่ จะได้ดังรูป 3.4 ดังนั้น cochlea จึงเป็นตัวรับความถี่จากทางกลเป็นสัญญาณระบบประสาท

ความถี่ธรรมชาติที่ตอบสนองกับจุดๆเดียวบนแผ่นเยื่อบาร์ซิลล่า แสดงในรูปที่ 3.5

3.2 ความเข้าใจ(Perception)

หน้าที่ของการรับเสียงและความสามรถ โดยปกติจะอยู่ในช่วง 16 Hz ถึง 16 kHz ในเด็กอาจถึง 20 kHz ยังมีอยู่ การรับฟังจะลดสมรรถภาพลงตามอายุ อาจจะถึง 10 kHz การได้ยินในช่วงความถี่ด้านต่ำสัญญาณที่ได้ยินจะเป็นแบบขบวนของสัญญาณ (pulse train) ที่ความถี่ด้านสูงการได้ยินจะค่อยๆจางลง จนไม่ได้ยินช่วงของความเข้มอยู่ในช่วงของ 1-130 dB ที่ความเข้มต่ำเสียงจะเบาจนไม่ได้ยิน ส่วนที่ความเข้มสูงจะทำให้เกิดความเจ็บปวดแก้วหูได้



รูปที่3.6 *แสดงเส้นแสดงระดับความเท่ากันของระดับความดัง*

3.2.1 **ความดัง** การรับรู้ของความดังเป็นหน้าที่ของทั้งความถี่และระดับของเสียง

โดยการเปรียบเทียบโทน(tone)ของเสียงที่ต่างกันของความถี่และแอมพลิจูด เส้นแสดงระดับความดังแสดงได้จากรูปที่3.6

3.2.2 **การเข้าใจเสียงพูด** ปัญหาพื้นฐานที่กำหนดโดยทฤษฎีเสียงพูดคือ สัญญาณเสียงที่เข้ามาที่หู จะแปลโดยสมองเป็นเสียงพูดได้อย่างไร จากการทดลองหลายๆครั้ง แล้วการรวบรวมก็พอได้คำตอบบางส่วน อย่างไรก็ตามคำตอบที่สมบูรณ์ก็ยังไม่ปรากฏอย่างเด่นชัด เราสามารถสรุปความสอดคล้องของบางส่วนที่หาข้อมูลมาได้ ดังนี้

ปรากฏว่า สมองจะสร้างความแตกต่างระหว่างเสียงพูดกับสัญญาณที่ไม่ใช่เสียงพูด การทดลองพบว่าเมื่อผู้ฟังได้ยินเสียงที่สังเคราะห์ขึ้น ไม่ใช่เสียงพูด แต่ทำคล้ายเสียงพูด จะพบขอบเขตที่มีการเปลี่ยนแปลงกระทันหันข้างใดข้างหนึ่ง จะถูกเข้าใจว่าเป็นเสียงพูด ส่วนอีกข้างหนึ่งจะไม่เป็นเสียงพูด

สมองสามารถแยกเสียงที่เป็นเสียงพูดและไม่ใช่เสียงพูดได้ เชื่อว่า ศูนย์ประมวลผลเสียงพูดจะอยู่บริเวณสมองซีกซ้าย ที่เป็นเช่นนี้ เพราะหูข้างขวาจะแสดงสมรรถภาพที่ดีกว่าหูซ้ายในการรับฟัง มีความเชื่อในทางที่ว่า ระบบประสาทของมนุษย์ ใช้ในการถอดรหัสเสียงที่เข้ามา สมองจะทำการแยกจำพวกของเสียงพูด นอกจากจะแบ่งเป็นสัญญาณเสียงพูด และไม่ใช่เสียงพูดแล้ว เสียงพูด โดยจะแยกความถี่มูลฐานของความถี่ส่วนฮาร์โมนิค และแยกออกมาเป็นเสียงพูดหลายชนิด หน่วยประมวลผลกลางจะแสดงการจดจำโดยใช้ผลอันนี้ในการรับสัญญาณเข้ามา

การวิจัย ชี้ให้เห็นว่า การเข้าใจเสียงพูดขึ้นกับคุณลักษณะความชัดเจนในการออกเสียงและการวิเคราะห์เสียงพูด

บทที่ 4

การแปลงฟูเรียร์(Fourier Transform)

การแปลงฟูเรียร์เป็นเครื่องมือวิเคราะห์ในงานวิทยาศาสตร์สาขาต่างๆ ในปัจจุบัน การประยุกต์ใช้งานที่นิยมใช้วิเคราะห์ระบบเชิงเส้นที่ไม่มีการแปรผันตามเวลา(linear time-invariant system) แต่การแปลงฟูเรียร์ที่จะพูดถึงต่อไปนี้ เป็นหลักหรือคุณสมบัติสำคัญ ซึ่งสามารถใช้วิเคราะห์งานได้ทั่วไป

4.1 ฟูเรียร์อินทิกรัล

ฟูเรียร์อินทิกรัล มีคำจำกัดความ เป็นสมการว่า

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{j2\pi ft} dt \qquad (2.1)$$

h(t) เป็นเทอมของฟังก์ชันของตัวแปรเวลา และ H(t) เป็นเทอมของฟังก์ชันของตัวแปรความถี่ การแปลงฟูเรียร์ของฟังก์ชันเวลา จะแทนด้วยตัวอักษรตัวใหญ่

ในรูปทั่วไป การแปลงฟูเรียร์ มีรูปเป็นจำนวนเชิงซ้อน(complex number)

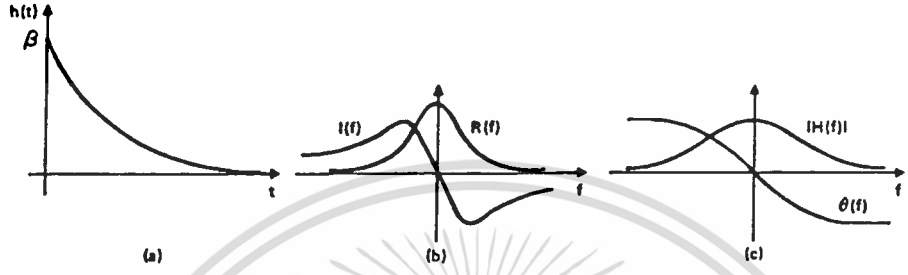
$$H(f) = R(f) + jI(f) = |H(f)| e^{j\theta(f)} \qquad (2.2)$$

เมื่อ R(f) เป็นส่วนจริงของการแปลงฟูเรียร์

I(f) เป็นส่วนจินตภาพของการแปลงฟูเรียร์

H(f) เป็นแอมพลิจูดหรือฟูเรียร์สเปคตรัมของ h(t) ซึ่งได้จาก $\sqrt{R(f)^2+I(f)^2}$

(t) เป็นมุมเฟสของการแปลงฟูเรียร์ ซึ่งได้ $\tan^{-1}[I(f)/R(f)]$



รูปที่2.1 (a) *ตัวอย่างของฟังก์ชันในโดเมนเวลา*

(b) *ส่วนจริงและส่วนจินตภาพของการแปลงฟูเรียร์*

(c) *แอมพลิจูดและเฟสของการแปลงฟูเรียร์*

4.2 การแปลงอินเวอร์สฟูเรียร์(inverse fourier transform)

การแปลงอินเวอร์สฟูเรียร์ มีค่าจากัดความ ดังนี้

$$h(t) = \int_{-\infty}^{\infty} H(f)\, e^{j2\pi ft} df \qquad (2.3)$$

4.3 การแปลงฟูเรียร์แบบไม่ต่อเนื่อง(discreate fourier transform หรือ DFT)

DFT เป็นลำดับของความถี่ไม่ต่อเนื่อง ที่ได้จากการสุ่มในหนึ่งคาบเวลาของการแปลงฟูเรียร์

เมื่อกำหนดให้จำนวนการสุ่มเท่ากับ N สุ่มบนคาบเวลา $0 < w < 2\pi$

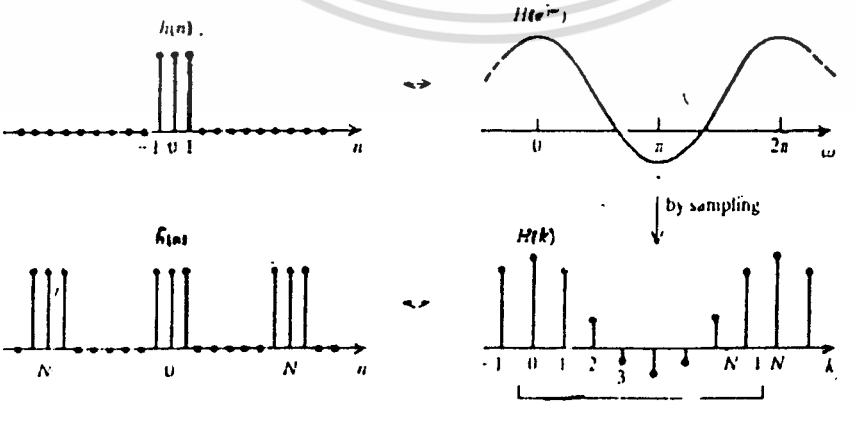
ดังนั้น

$$w_k = 2\pi k/N \qquad \text{เมื่อ } 0<k<N-1 \qquad (2.4)$$

ถ้ากำหนดให้ {h(n)} เป็นลำดับของ discreat time และลำดับของ Fourier Transform $H(e^{j\omega})$ ให้เท่ากับ {H(k)} โดยที่

$$H(k) = H(e^{j\omega}) \qquad \text{เมื่อ } w=w_k=2\pi k/N \text{ และ } 0<k<N-1 \qquad (2.5)$$

ลำดับของ DFT เริ่มต้นที่ k=0 หรือที่ w=0 แต่ไม่รวมจุด k=N หรือ w=2π เราจะเห็นว่า$H(e^{j\omega})$ มีคาบเวลาเท่ากับ w หรือ 2π ซึ่งได้จากการกระจายอนุกรมฟูเรียร์และคำนวณค่าสัมประสิทธิ์จากลำดับ{h(n)} แสดงว่าเมื่อสัญญาณเวลาต่อเนื่องถูกสุ่มด้วยคาบเวลาการสุ่มT_s สเปคตรัมของผลลัพธ์จากการแปลงฟูเรียร์จะมีฟังก์ชันคาบเวลาของความถี่เท่ากับ $2\pi/T_s$ จากรูปที่1 เมื่อ H (e^{j}) ถูกสุ่มด้วยคาบเวลาการสุ่ม $w_s=2\pi/N$ เมื่อแปลงกลับเป็นลำดับของฟังก์ชันเวลาไม่ต่อเนื่อง {h(n)}จะมีคาบเวลา $2\pi/w_s=N$



รูปที่4.2 *แสดงผลลัพธ์ในลำดับของเวลา จาก sampled spectrum ในกรณีนี้ N=8 และอยู่ในช่วง 0 < w < 2π*

ดังนั้นลำดับ discrete time ที่อยู่ในฟังก์ชันคาบ สามารถหาในเทอมของ {h(n)} ได้

$$\hat{h}(n) = \sum_{m=-\infty}^{\infty} h(n+mN)$$

ลำดับ {h(n)} เรียกว่า periodic extention ของ {h(n)} จำนวนจุดสุ่มในหนึ่งคาบของ spectrum N จะมีค่าเท่ากับคาบของ {h(n)}

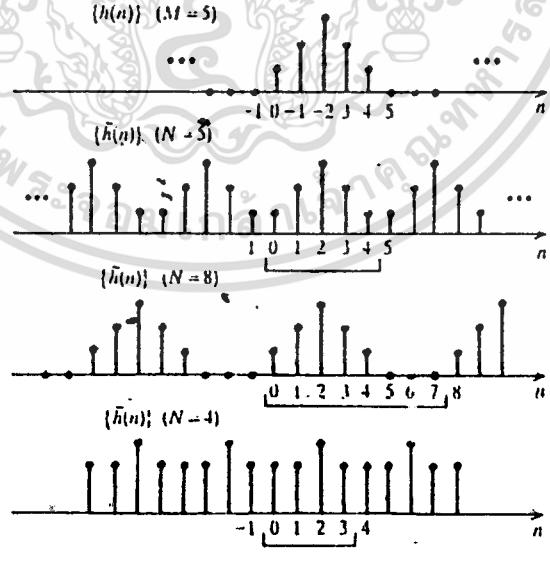
เราสามารถเลือกค่า N เท่าใดก็ได้ ซึ่งเป็นจำนวนการสุ่มของ $H(e^{j\omega})$ ภายใต้ $0 \leq \omega < 2$ เมื่อ N เป็นคาบของ {h(n)} เราต้องเลือกค่าที่ไม่น้อยเกินไป ในลักษณะที่ {h(n)}เป็นฟังก์ชันคาบเวลาแบบไม่จำกัด ทำให้เกิดลักษณะสัญญาณที่ถูกสุ่มทับกันสัญญาณที่ได้จะผิดพลาดไป สามารถแสดงจากตัวอย่างข้างล่าง

กำหนด

$h(n) = h(n)$ เมื่อ $0 <= n =< M-1$

$= 0$ เมื่อ $M <= n =< N-1$

จะได้จุดที่เกิดจากการสุ่ม เมื่อ N มีค่าต่าง ๆ ดังรูป



รูปที่4.3 *แสดงความสัมพันธ์ของช่วงเวลาของลำดับ M และจำนวนจุดในการสุ่มสเปคตรัม N*

จากรูป จะเห็นว่า ถ้า ค่า M>N รูปสัญญาณจะทับกัน (overlap) หรือเกิด time-aliasing ซึ่งสามารถป้องกันได้โดยเลือกค่า N >= M

บทที่ 5
การแปลงฟาสต์ฟูเรียร์ (FFT)
และการประยุกต์ใช้งาน

FFT(fast fourier transform) เป็นอัลกอริทึม(algorithm) ที่ช่วยในการคำนวน DFT มีประสิทธิภาพมากขึ้น ลดจำนวนการกระทำทางคณิตศาสตร์และขนาดของหน่วยความจำที่ต้องใช้ในการคำนวณ DFT อัลกอริทึมของ FFT จึงเป็นเทคนิคที่ถูกเลือกใช้ในการประยุกต์ใช้งานด้านต่างๆ

5.1 ลักษณะของปัญหา

ให้ $\{X(m)\}$ เป็นลำดับของข้อมูล $X(m), m = 0,1,\ldots,N-1$ ที่ได้จากการสุ่มในช่วงจำกัดของสัญญาณ $x(t)$ เรามีอัลกอริทึมที่ใช้คำนวณค่าสัมประสิทธิ์ $C_x(k)$ ดังสมการนี้

$$C_x(k) = 1/N\sum_{m=0}^{N-1} X(m)W^{km}, \quad k = 0,1,\ldots,N-1 \qquad (5.1\text{-}1)$$

เมื่อ $W = e^{-i2\pi/N}$, $i = \sqrt{-1}$ เราเรียกสมการ(1) นี้ว่า DFT ของ $\{X(m)\}$ อัลกอริทึมซึ่งเราจะได้กล่าวถึงต่อไปเรียกว่าการแปลงฟาสต์ฟูเรียร์ (FFT) เป็นอัลกอริทึมที่พัฒนาขึ้นโดย Cooley และ Tukey บางทีอาจจะเรียกว่าอัลกอริทึมของ Cooley และ Tukey ก็ได้ ตั้งแต่นี้ไป เราจะตั้งข้อสมมุติว่า

$$N = 2^n, \qquad n = 1,2,\ldots,n_{max}$$

จากที่เราได้เข้าใจกันแล้ว ถ้าไม่ต้องการให้ข้อมูลสูญหาย ต้องมีขนาด N ที่เหมาะสม ตามทฤษฎีการสุ่ม(Sampling theorem) นั่นคือ $N \geq 2BL$ เมื่อ B เป็นแบนวิดธ์ของ $x(t)$ ในหน่วย Hz และ L คือช่วงเวลาของมัน สำหรับในกรณี N ไม่เท่ากับ 2^n เราจะได้พูดถึงกันต่อไป

.2 เข้าสู่การหาอัลกอริทึม

เราจะพิจารณาในกรณีที่ {X(m)} เป็นค่าจำนวนจริง และ N = 8 จากคุณสมบัติคอมเพล็กซ์คอนจูเกต(complex conjugate property) เรามี

$$C_x(4+l) = C_x(4-l), \quad l = 1,2,3$$

ดังนั้นเมื่อเทียบกับสมการที่ (5.1-1) ทำให้การคำนวณเหลือเพียง

$$C_x(k) = 1/8\sum_{m=0}^{7}X(m)W^{km}, \quad k = 0,1,\ldots,4 \qquad (5.2\text{-}1)$$

เมื่อ $W = e^{-i2\pi/4}$

จาก สมการ(1) จะได้ว่า

$$8C_x(k) = \sum_{m=0}^{7}X(m)\cos(mk\pi/4) - i\sum_{m=0}^{7}X(m)\cos(mk\pi/4) = A(k)-B(k)$$

เมื่อ

$$A(k) = \sum_{m=0}^{7}X(m)\cos(mk\pi/4),$$

และ

$$B(k) = \sum_{m=0}^{7}X(m)\cos(mk\pi/4), \quad k = 0,1,\ldots,4 \qquad (5.2\text{-}2)$$

ใช้สัญญลักษณ์เมทริกซ์ สมการที่(2) จะเป็น

$$A = CX$$

$$B = SX$$

เมื่อ A และB มีมิติ (5x1) , X มีมิติ (8x1) และ Cและ S มีมิติ (5x8) ดังนี้

$$S = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/\sqrt{2} & 1 & 1/\sqrt{2} & 0 & -1/\sqrt{2} & -1 & -1/\sqrt{2} \\ 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 \\ 0 & 1/\sqrt{2} & -1 & 1/\sqrt{2} & 0 & -1/\sqrt{2} & 1 & -1/\sqrt{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

และ

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1/\sqrt{2} & 0 & -1/\sqrt{2} & -1 & -1/\sqrt{2} & 0 & 1/\sqrt{2} \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 \\ 1 & -1/\sqrt{2} & 0 & 1/\sqrt{2} & -1 & 1/\sqrt{2} & 0 & -1/\sqrt{2} \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}$$

(5.2-3)

เมื่อพิจารณาดูสมการที่(3) จะเห็นว่า สมาชิกของเมทริกซ์ Cและs มีค่าซ้ำกัน ซึ่งเป็นเหตุผล 2 ประการที่พัฒนาอัลกอริทึมฟาสต์ฟูเรียร์ ดังนี้

1.ความเป็นซายน์(sinusoidal) จัดกลุ่มด้วยลักษณะเด่น ซึ่งจะเกิดขึ้นได้เมื่อเลือกจุดสุ่มที่ได้อย่างถูกต้อง

2.ถ้า N เป็น เป็นจำนวนของจุดสุ่มของ x(t) ที่มีรูปแบบไม่แน่นอนแล้ว การเกิดซ้ำกันของเมทริกซ์ Cและs จะเพิ่มขึ้นตามจำนวนตัวประกอบของ $N = 2^n = 2x2x...x2$ ดังนั้น เมื่อจำนวนของข้อมูลมาก การใช้อัลกอริทึมฟาสต์ฟูเรียร์จะช่วยประหยัดเวลาและหน่วยความจำอย่างมาก

5.3 หนทางการพัฒนาอัลกอริทึม

หนทางที่จะพัฒนาอัลกอริทึมที่ต้องการคือการหาความสัมพันธ์กับการซ้ำกันในรูปแบบทั่วไปของเมทริกซ์ Cและs กับ กำลังของ W หรือหลักรากที่ N ของหนึ่ง (principle N-th root of unity) เพื่อหาสัญลักษณ์ของรูปทั่วไป

สัญลักษณ์ แสดงค่าในฐานสิบแต่ละตัวของ $m, 0 \leq m \leq N-1$ ให้อยู่ในรูปของเลขฐานสองหรือไบนารี่(binary) ดังนี้

$$m = m_{n-1}2^{n-1} + m_{n-2}2^{n-2} + \ldots + m_1 2^1 + m_0 2^0,$$

เมื่อ

$$m_v = 0 \text{ หรือ } 1, \quad v = 0,1,\ldots,n-1 \quad , n = \log_2 N$$

ในทำนองเดียวกัน ค่าในฐานสิบแต่ละตัวของ $k, \quad 0 \leq m \leq N-1$ เปลี่ยนรูปเป็น

$$k = k_{n-1}2^{n-1}+k_{n-2}2^{n-2}+\ldots+k_1 2^1+k_0 2^0,$$

เมื่อ

$$k_v = 0 \text{ หรือ } 1 , \quad v = 0,1,\ldots,n-1$$

ให้รูปแบบไบนารี่ของ $X(m)$ แทนด้วย $X(m)$ จะได้ว่า

$$\begin{aligned} X(m) &= X(m_{n-1}2^{n-1}+m_{n-2}2^{n-2}+\ldots+m_1 2^1+m_0 2^0) \\ &= X(m_{n-1},m_{n-2},\ldots,m_1,m_0) \end{aligned} \qquad (5.3\text{-}1)$$

เปลี่ยนรูปของสมการ (1) ตามความสัมพันธ์ที่ได้ ดังนี้

$$\sum_{m=0}^{N-1} X(m)W^{km} = \sum_{m_0}\sum_{m_1}\cdots\sum_{m_{n-1}} X(m_{n-1},m_{n-2},\ldots,m_1,m_0)W^{k(m_{n-1}2^{n-1}+\ldots+m_0 2^0)} \qquad (5.3\text{-}2)$$

5.4 การพัฒนาอัลกอริทึม

ขั้นตอนการพัฒนา algolithm จะอธิบายในกรณี $N = 8 = 2^3$ ดังนั้นจะได้

$$C_x(k) = (1/8)\sum_{m=0}^{7} X(m)W^{km}, \quad k=0,1,..,7 \qquad (5.4\text{-}1)$$

เมื่อ $W = e^{-i\pi/4}$ เราแสดงค่าของ m ในรูปไบนารี่ ดังนี้

$$m = m_2 2^2+m_1 2^1+m_0 2^0 \qquad (5.4\text{-}2)$$

จากสมการ (5.3-2),สมการที่(1) และ(2) จะได้ว่า

$$8C_x(k) = \sum_{m_0}\sum_{m_1}\sum_{m_2} \tilde{X}(m_2,m_1,m_0)W^{k(4m_2+2m_1+m_0)}$$

$$= \sum_{m_0,m_1,m_2} \sum \tilde{X}(m_2, m_1, m_0) W^{4km_2} W^{2km_1} W^{km_0} \qquad (5.4\text{-}3)$$

จากสมการ(3) ผลบวกในสุด ของ m_2 แทนด้วยสัญญลักษณ์ M_2 จะได้ว่า

$$M_2 = \sum_{m_2} \tilde{X}(m_2, m_1, m_0) W^{4km_2}$$

แทนค่าของ k ในรูปของไบนารี่ และ $W^4 = -1$ เราจะได้ว่า

$$M_2 = \sum_{m_2} \tilde{X}(m_2, m_1, m_0)(-1)^{m_2[4k_2+2k_1+k_0)}$$

เพราะว่า $(-1)^m [4k +2k +k) = 1$ M_2 สามารถแสดงในรูปของ

$$M_2 = \sum_{m_2} \tilde{X}(m_2, m_1, m_0)(-1)^{k_0 m_2} \qquad (5.4\text{-}4)$$

จากสมการ(4) ผลบวกบน m_2แสดงค่าที่รวมได้ในรูปฟังก์ชันของ k_0, m_1, m_0

$$M_2 = \sum_{m_2} \tilde{X}(m_2, m_1, m_0)(-1)^{k_0 m_2} = \tilde{X}_1(k_0, m_1, m_0) \qquad (5.4\text{-}5)$$

แทน สมการ(5) ลงใน สมการ(3)

$$8C_x(k) = \sum_{m_0} \sum_{m_1} \tilde{X}_1(k_0, m_1, m_0) W^{2km_1} W^{km_0} \qquad (5.4\text{-}6)$$

ทำซ้ำอีกครั้ง ในส่วนในสุดของสมการ(6) และแทนด้วยสัญญลักษณ์ M_1

$$M_1 = \sum_{m_1} \tilde{X}_1(k_0, m_1, m_0) W^{2km}$$
$$= \sum_{m_1} \tilde{X}_1(k_0, m_1, m_0)(-i)^{(4k_2+2k_1+k_0)m_2} \qquad (5.4\text{-}7)$$

ในสมการ(7) จำนวน $(-i)^{4k_2 m_1} = 1$ จะได้รูปแบบที่ง่ายขึ้น ดังนี้

$$M_1 = \sum_{m_1} \tilde{X}_1(k_0, m_1, m_0)(-i)^{(2k_1+k_0)m_1} \qquad (5.4\text{-}8)$$

ผลบวก บน m_1จะได้ผลลัพธ์ M_1ที่เป็นฟังก์ชันของ k_0, k_1, m_0

$$M_1 = \tilde{X}_2(k_0, k_1, m_0) \qquad (5.4\text{-}9)$$

แทน สมการ(9) ลงใน สมการ(6) จะได้

$$8C_x(k) = \sum_{m_0} x_2(k_0, k_1, m_0) W^{m_0 k}$$

เหมือนกับ M_2 และ M_1 เราแทนสัญญลักษณ์ผลบวกบน m_0 ด้วย M_0

$$M_0 = \sum_{m_0} \tilde{X}_2(k_0, k_1, m_0) W^{m_0 k},$$

ซึ่งจะได้

$$M_0 = \sum_{m_0} \tilde{X}_2(k_0, k_1, m_0')[(1-i)/\sqrt{2}]^{(4k_2+2k_1+k_0)m_0} \qquad (5.4\text{-}10)$$

การทำผลบวกบน m_0 จะได้ M_0 ที่เป็นฟังก์ชันของ k_0, k_1, k_2 ดังนี้

$$M_0 = \tilde{X}_3(k_0, k_1, k_2) \qquad (5.4\text{-}11)$$

$$8C_x(k) = 8\tilde{C}_x(k_2, k_1, k_0) = \tilde{X}_3(k_0, k_1, k_2) \qquad (5.4\text{-}12)$$

ขั้นแรกในการหาคำตอบพิจารณา

1. $N = 8$ ทำให้ผล $\log_2 N = 3$

2. coefficient $(-1), (-i), [(1-i)/\sqrt{2}]$ มีความสัมพันธ์กับรากที่ 2,4,8 ตามลำดับ ของ unity $e^{-i2\pi}$

ดังนั้นจึงกำหนดสัญลักษณ์ใหม่ ให้

$$A_2 = e^{-i2\pi/2^r}, \quad r = 1, 2, \ldots, \log_2 N \qquad (5.4\text{-}13)$$

เมื่อ A_2 เป็นรากที่ 2^r ของ unity โดยมีคุณสมบัติดังนี้

$$1.\ A_2 = W^{N/2}\ ,\ W = e^{-i2\pi/n} \qquad (5.4\text{-}14a)$$

$$2.\ (A_2)^{+l} = -(A_2)^{l}$$

$$;\ r = 1,2,\ldots\log_2 N, l = 0,1,..,2^{r-1} \qquad (5.4\text{-}14b)$$

เมื่อ $= 2^{r-1}$

$$3.\ (A_N)^{N/2} = -1 \qquad (5.4\text{-}14c)$$

จากสมการข้างต้นสามารถเขียนในเทอม A_2 จะได้

$$X_1(k_0, m_1, m_2) = \sum_{m_2} \tilde{X}(m_2, m_1, m_0) A_2^{k_0 m_2} \qquad (5.4\text{-}15)$$

ในสมการ(15) k_0 มีค่า 0 หรือ 1 ทำให้ได้สมการ 4 สมการ ในแต่ละค่า k_0

กรณี 1 : $k_0 = 0$ (5.4-16a)

$X_1(0,0,0) = X(0,0,0)+X(1,0,0) \Longrightarrow X_1(0) = X(0)+X(4)$

$X_1(0,0,1) = X(0,0,1)+X(1,0,1) \Longrightarrow X_1(1) = X(1)+X(5)$

$X_1(0,1,0) = X(0,1,0)+X(1,1,0 \Longrightarrow X_1(2) = X(2)+X(6)$

$X_1(0,1,1) = X(0,1,1)+X(1,1,1) \Longrightarrow X_1(3) = X(3)+X(7)$

กรณี 2 : $k_0 = 1$ (5.4-16b)

$X_1(1,0,0) = X(0,0,0)+X(1,0,0) \Longrightarrow X_1(4) = X(0)-X(4)$

$X_1(1,0,1) = X(0,0,1)+X(1,0,1) \Longrightarrow X_1(5) = X(1)-X(5)$

$X_1(1,1,0) = X(0,1,0)+X(1,1,0) \Longrightarrow X_1(6) = X(2)-X(6)$

$X_1(1,1,1) = X(0,1,1)+X(1,1,2) \Longrightarrow X_1(7) = X(3)+X(7)$

จากสมการ (9) ในเทอม A_4

$$\tilde{X}_2(k_0,k_1,m_0) = \tilde{X}_1(k_0,0,m_0)+\tilde{X}_1(k_0,1,m_0)A_4^{(2k+k)}$$

กรณี 1 :$(k_1,k_0)=(0,0)$ (5.4-17a)

$$\tilde{X}_2(0,0,0) = \tilde{X}_1(0,0,0)+\tilde{X}_1(0,1,0)\Longrightarrow X_2(0) = X_1(0)+{}_1X(2)$$
$$\tilde{X}_2(0,0,1) = X_1(0,0,1)+X_1(0,1,1)\Longrightarrow X_2(1) = X_1(1)+X_1(3)$$

กรณี 2 :$(k_1,k_0)=(0,1)$ (5.4-17b)

$$X_2(1,0,0) = X_1(1,0,0)+A_4\ X_1(1,1,0)\Longrightarrow X_2(4)=X_1(4)+A_4\ X_1(6)$$
$$X_2(1,0,1) = X_1(1,0,1)+A_4\ X_1(1,1,1)\Longrightarrow X_2(1)=X_1(5)+A_4\ X_1(7)$$

กรณี 3 :$(k_1,k_0)=(1,0)$ (5.4-17c)

$$X_2(0,1,0) = X_1(0,0,0)+A_4\ X_1(0,1,0)\Longrightarrow X_2(2) = X_1(0)-X_1(2)$$
$$X_2(0,1,1) = X_1(0,0,1)+A_4\ X_1(0,1,1)\Longrightarrow X_2(3) = X_1(1)-X_1(3)$$

กรณี 4 :$(k_1,k_0)=(1,1)$ (5.4-17d)

$$X_2(1,1,0) = X_1(1,0,0)+A_4\ X_1(1,1,0)\Longrightarrow X_2(6)=X_1(4)-A_4\ X_1(6)$$
$$X_2(1,1,1) = X_1(1,0,1)+A_4\ X_1(1,1,1)\Longrightarrow X_2(7)=X_1(5)-A_4\ X_1(7)$$

สมการ (11) เขียนในเทอม A_8 จะได้

$$X_2(k_0,k_1,k_2) = X_2(k_0,k_1,0)+X_2(k_0,k_1,1)A_8^{(4k+2k+k)} \quad (5.4\text{-}18)$$

กรณี 1 : $(k_2,k_1,k_0)=(0,0,0)$ (5.4-19a)

$$X_3(0,0,0)=X_2(0,0,0)+X_2(0,0,1)\Longrightarrow X_3(0)=X_2(0)+X_2(1)$$

กรณี 2 : $(k_2,k_1,k_0)=(0,0,1)$ (5.4-19b)

$$X_3(1,0,0)=X_2(1,0,0)+A_8\ X_2(1,0,1)\Longrightarrow X_3(4)=X_2(4)+A_8\ X_2(5)$$

กรณี 3 : $(k_2,k_1,k_0)=(0,1,0)$ (5.4-19c)

$$X_3(0,1,0)=X_2(0,1,0)+A_8\ X_2(0,1,1)\Longrightarrow X_3(2)=X_2(2)+A_8\ X_2(3)$$

กรณี 4 : $(k_2,k_1,k_0)=(0,1,1)$ (5.4-19d)

$$X_3(1,1,0)=X_2(1,1,0)+A_8\ X_2(1,1,1)\Longrightarrow X_3(6)=X_2(6)+A_8\ X_2(7)$$

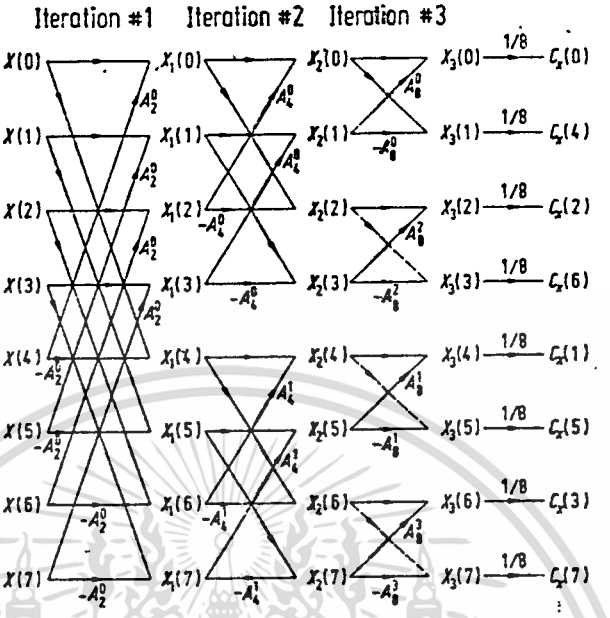
กรณี 5 : $(k_2,k_1,k_0)=(1,0,0)$ (5.4-19e)

$$X_3(0,0,1)=X_2(0,0,0)+A_8\ X_2(0,0,1)\Longrightarrow X_3(1)=X_2(0)-X_2(1)$$

ลำดับการกระทำทางคณิตศาสตร์ตั้งแต่สมการ(19a) ถึง สมการ(19e) ได้แสดงไว้ในรูปที่5.1 และการทำซ้ำ(iteration) ครั้งที่3 ก็คือ r = 3 ในสมการที่(13)

เพื่อให้ได้ค่า $C_x(k)$ ตามที่ต้องการ ใช้สมการ

$$8C_x(k_2,k_1,k_0) = X_3(k_0,k_1,k_2) \qquad (5.4\text{-}20)$$



รูปที่5.1 *แสดงแผนภาพการไหล(signal flow-graph) ของสัญญาณ FFT เมื่อ N = 8*

จะเห็นว่ามีการสลับตำแหน่งของสัมประสิทธิ์ไบนารี่ k_0, k_1 และ k_2 ใน $X_3()$ เมื่อเทียบกับตำแหน่งจริง $C_x(k)$ เราเรียกลักษณะเช่นนี้ว่า *ปรากฏการณ์การกลับบิท (bit reversal)* $X_3()$ มีความสัมพันธ์กับ $C_x(k)$ ตามสมการข้างล่างนี้

$$\begin{aligned} X_3(0) &= 8C_x(0) \\ X_3(4) &= 8C_x(1) \\ X_3(2) &= 8C_x(2) \\ X_3(6) &= 8C_x(3) \\ X_3(1) &= 8C_x(4) \end{aligned} \qquad (5.4\text{-}21)$$

ผลลัพธ์ของสัมประสิทธิ์ $C_x(k)$, $k = 5,6,7$ เราหาได้จาก

$$C_x(4+l) = C_x(4-l), \quad l = 1,2,3$$

จะได้

$$C_x(5) = C_x(3)$$

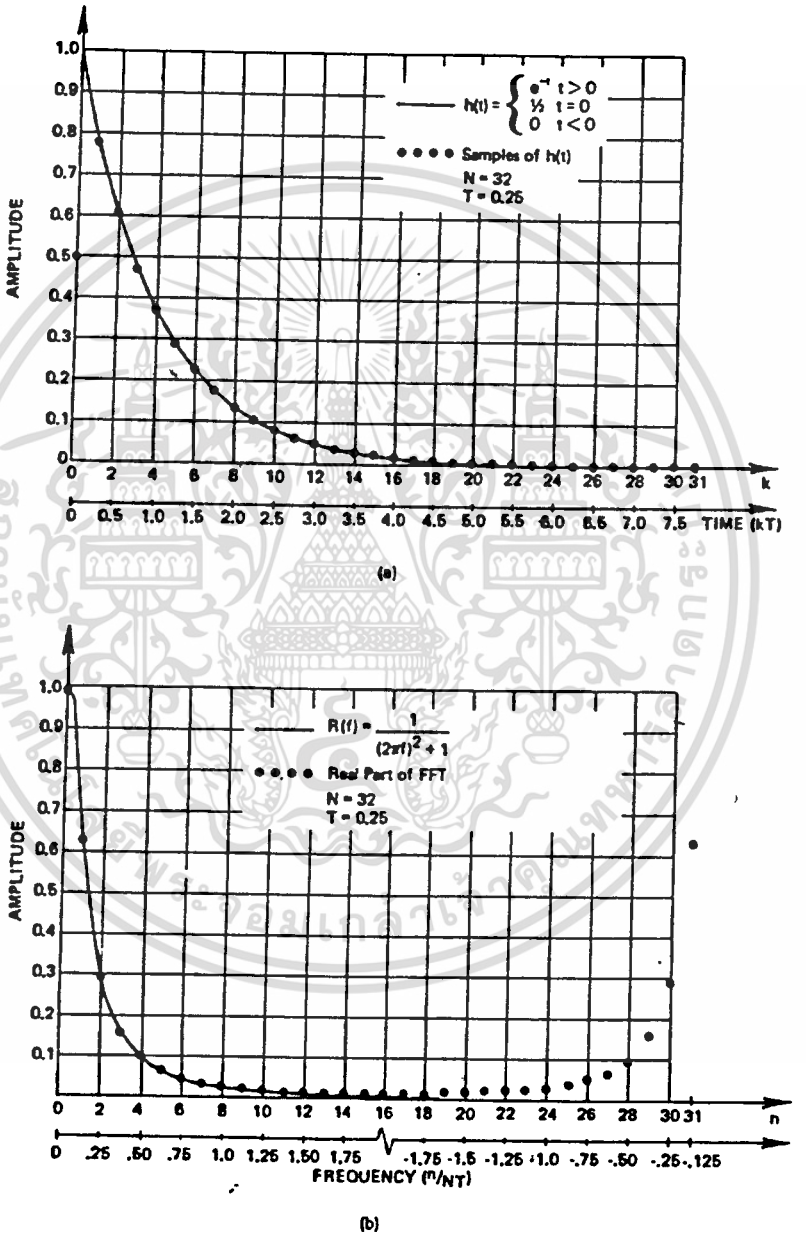
$$C_x(6) = C_x(2)$$

$$C_x(7) = C_x(1)$$

$C_x(5), C_x(6), C_x(7)$ สามารถหาได้จากสมการ (12) และ (18)

$$\begin{aligned} 8C_x(5) &= X_3(5) = X_2(4) - A_8X_2(5) \\ 8C_x(6) &= X_3(3) = X_2(2) - A_8X_2(3) \\ 8C_x(5) &= X_3(7) = X_2(6) - A_8X_2(7) \end{aligned} \qquad (5.4\text{-}22)$$

จากขั้นตอนข้างต้นเป็นขั้นตอนและแนวทางในการเขียน โปรแกรม FFT



รูปที่ 5.2 *แสดงตัวอย่างการแปลงฟูเรียร์ โดยผ่าน FFT*

5.5 การประยุกต์ใช้งาน FFT

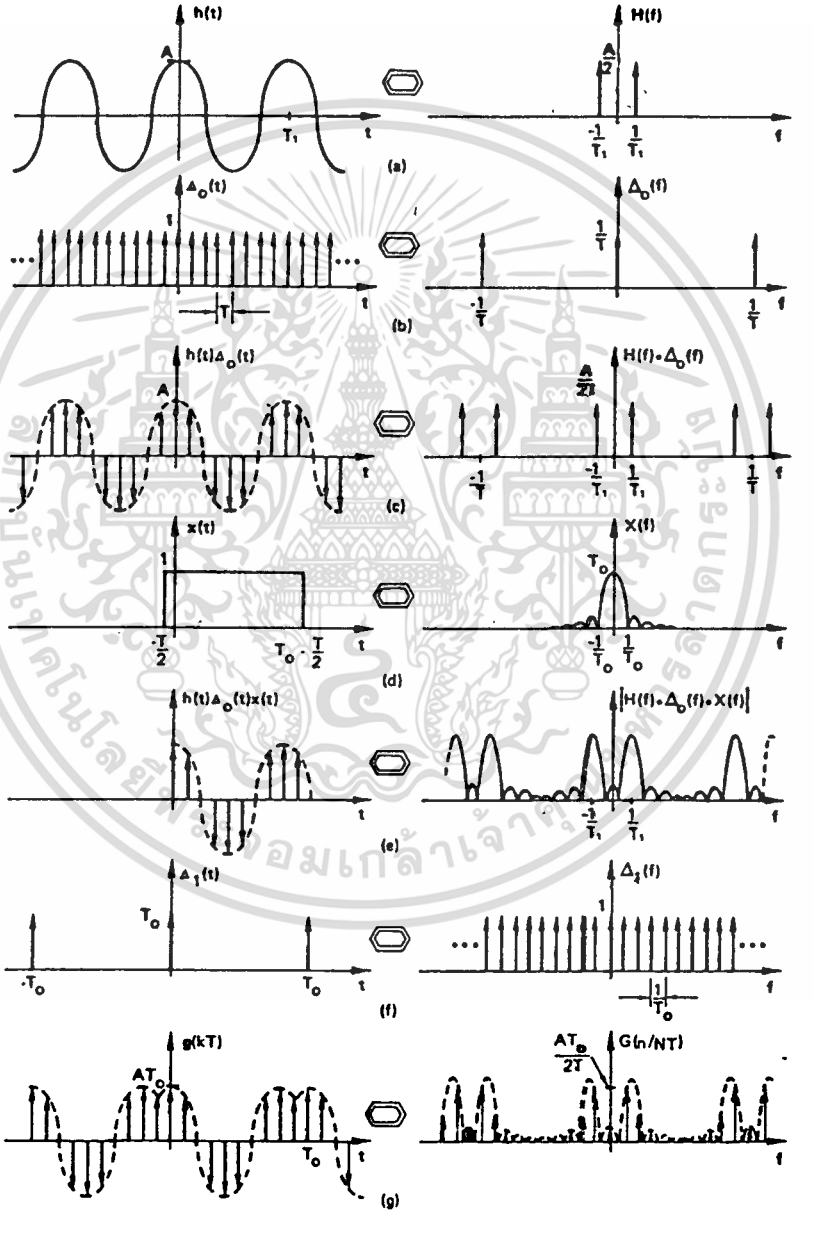
5.5.1 ความละเอียด(resolution)ของ FFT

ผลลัพธ์ของ FFT ในรูปที่5.2(b) และ(c) แต่ละความถี่มีระยะห่าง f_o = 1/NT ดังนั้นจุดสุ่มในโดเมนความถี่จึงมีค่าตั้ง 0/NT,1/NT,2/NT,...,(N/2)/NT สำหรับด้านความถี่บวก ระยะห่างของความถี่ f_o = 1/NTเป็นเทอมที่แสดงถึงความละเอียดของFFT แต่ละสมาชิกของความถี่ที่ได้เรียกว่า สมาชิกความละเอียด(resolution element) หรือเซลความละเอียด (resolution cell) เราอาจจะคิดว่า ความละเอียดเป็นเทอมที่ทำให้เราสามารถแยกความแตกต่างของแต่ละความถี่ โดยประมาณค่าความถี่ที่ได้จากการแปลงเป็นความถี่ตั้งแต่ 0/NT,1/NT,2/NT, จนถึง(N/2)/NT ซึ่งเราอาจจะลดระยะห่างระหว่างแต่ละความถี่ได้ โดยการเพิ่มจำนวนข้อมูล N ซึ่งก็คือการเพิ่มระยะการตัดทอน(truncation) ของฟังก์ชันที่ต้องการแปลง ถ้าค่าของ N เพิ่มขึ้นเป็นสองเท่า ระยะห่างของความถี่ก็ลดลงสองเท่าเช่นกัน

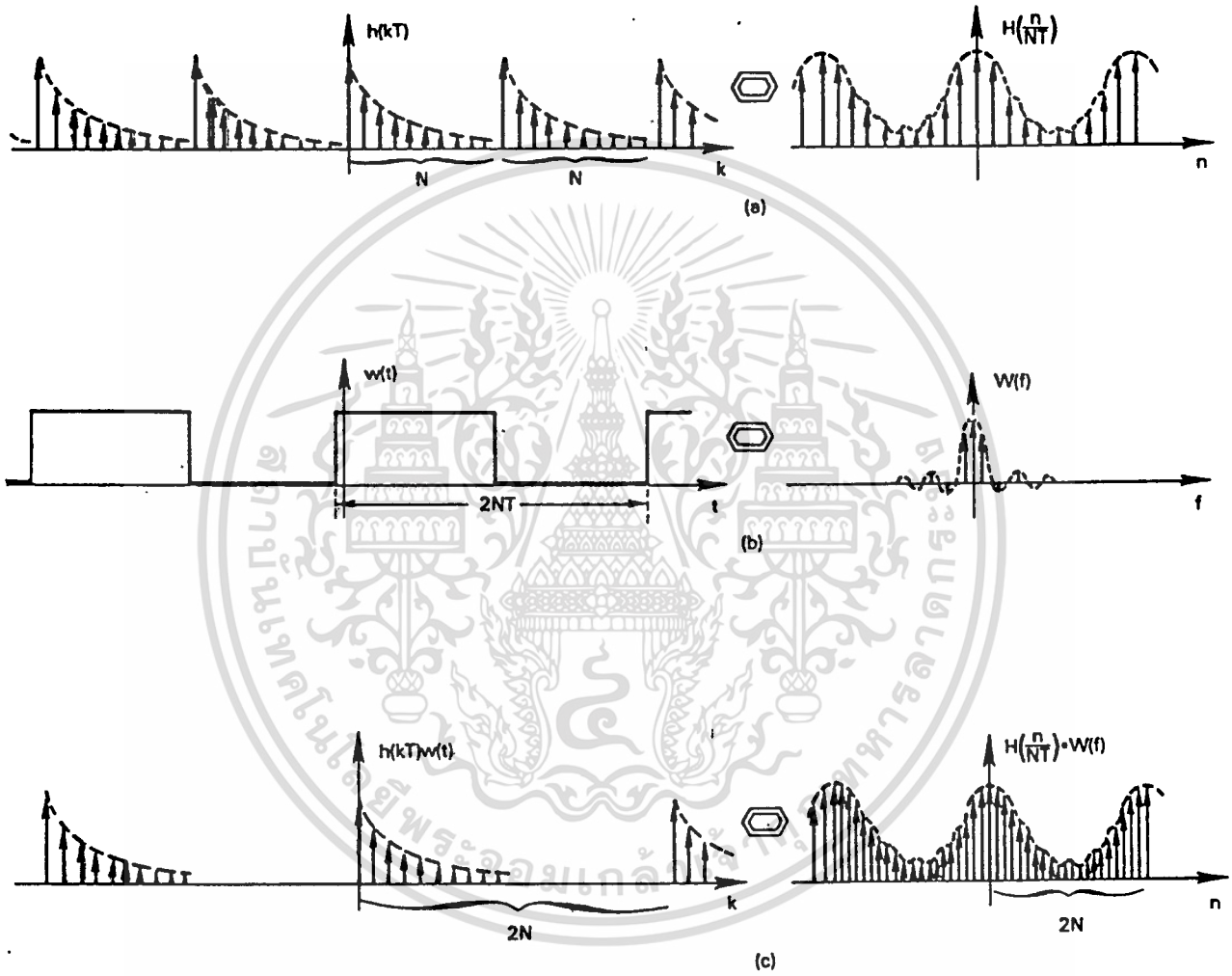
จากรูปที่5.3 ระยะห่างของความถี่(ความละเอียด) ของการแปลงดิสครีตฟูเรียร์(discreate fourier transform) ถูกกำหนดจากความกว้างของสี่เหลี่ยมที่ไปคูณหรือตัดทอนฟังก์ชันก่อนทำการแปลง การตัดทอนในโดเมนเวลาก็คือการทำคอนวูลูชั่นของฟังก์ชัน [sin(f)/f] กับผลการแปลงฟูเรียร์ของฟังก์ชันเริ่มต้น การทำคอนวูลูชันกับฟังก์ชัน[sin(f)/f] ทำให้ผลจากการแปลงคลุมเคลือและเลือนลาง ถ้าฟังก์ชันที่ตัดทอนในโดเมนเวลามีความกว้างมากขึ้น จะทำให้ผลของฟังก์ชัน[sin(f)/f]แคบลง และความคลุมเคลือของความถี่ลดน้อยลงด้วย ความคลุมเคลือของความถี่ยิ่งน้อยเท่าใด ก็ยิ่งทำให้ประสิทธิภาพของการใช้ FFT ในการแก้ปัญหาในแต่ละงาน เป็นไปได้มากยิ่งขึ้น

ความเข้าใจผิด โดยทั่วๆไปของผู้ใช้ FFT คือ การเพิ่มจำนวนข้อมูล N โดยการต่อเติมศูนย์และตัดทอนฟังก์ชัน และตีความหมายของฟังก์ชันความถี่ที่ได้เป็นฟังก์ชันความถี่ที่มีความละเอียดสูง ตอนนี้เรามาเปรียบเทียบรูปที่6.2 กับรูปที่5.4 ในรูปที่5.4(a) แสดงผลจากการแปลงดิสครีตฟูเรียร์ เช่นเดียวกับรูปที่6.2(ข)

เราต้องการตรวจสอบผลของการเพิ่มศูนย์ให้กับฟังก์ชันเวลาของรูปที่5.4(a) สมมติว่าจำนวนของศูนย์ที่เพิ่มเข้าไปนั้น



รูปที่5.3 *แสดง DFT ของสัญญาณคาบเวลา โดยมีช่วงการตัดทอนน้อยกว่า 1 คาบเวลา*



รูปที่ 5.4 *แสดงตัวอย่างของการเพิ่มความละเอียดของ FFT โดยการต่อเติมศูนย์*

เท่ากับ N ซึ่งทำได้โดยการด้วยฟังก์ชันคาบเวลาในโดเมนเวลาในรูปที่5.4(b) ผลของความถี่ที่ได้ แสดงในรูปถัดมา การคูณทำให้ฟังก์ชันมีคาบเวลาเท่ากับ 2N โดยที่จุดที่ไม่ใช่ศูนย์เท่ากับจุดสุ่ม N จำนวนในรูปที่ 5.4(a) ผลของฟังก์ชันความถี่ที่ได้ก็คือการทำคอนโวลูชั่นของฟังก์ชันความถี่ของรูปที่5.4(a) และรูปที่5.4(b) แต่ความละเอียดของความถี่ได้ถูกกำหนดไว้ในรูปที่5.4(a)เรียบร้อยแล้ว การทำคอนโวลูชั่นเป็นการเพิ่มจำนวนจุดสุ่มในโดเมนความถี่โดยการเฉลี่ยค่า (interpolation) ด้วยฟังก์ชัน[sin(f)/f] กับผลการแปลงฟูเรียร์ของฟังก์ชันเริ่มต้น ดังนั้น ถึงแม้ว่าระยะห่างของผลลัพธ์จาก FFT จะมีระยะใกล้ขึ้น โดยการต่อแต้มศูนย์ ก็ตาม แต่ความละเอียดก็ไม่ได้เปลี่ยนแปลง *ความละเอียดของ FFT ไม่สามารถทำให้ความละเอียดของข้อมูลเพิ่มขึ้น โดยการต่อเติมศูนย์ นอกเสียจากว่า ฟังก์ชันที่จะทำการต่อเติมศูนย์มีค่าศูนย์ครอบคลุมช่วงนั้นอยู่แล้ว*

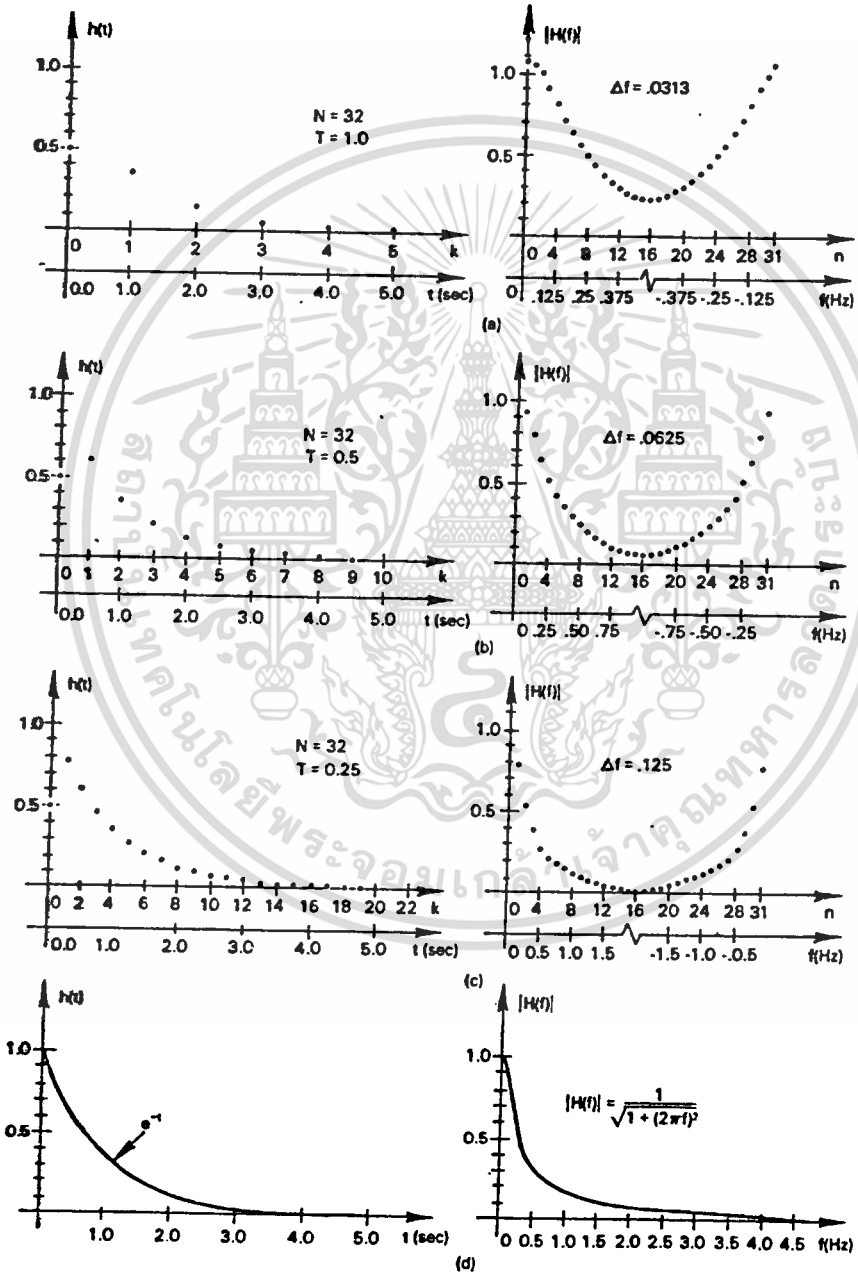
จากหัวข้อนี้ เราอาจจะสรุปได้ว่า ความละเอียดนั้นต้องพิจารณาจากช่วงเวลาในโดเมนเวลาของสัญญาณ และในการประยุกต์ใช้ FFT ช่วงเวลาของสัญญาณถูกตั้งใหม่เป็นช่วงระยะห่างของการตัดทอนข้อมูล

5.5.2ผลความผิดเพี้ยนจาก FFT(FFT alising)

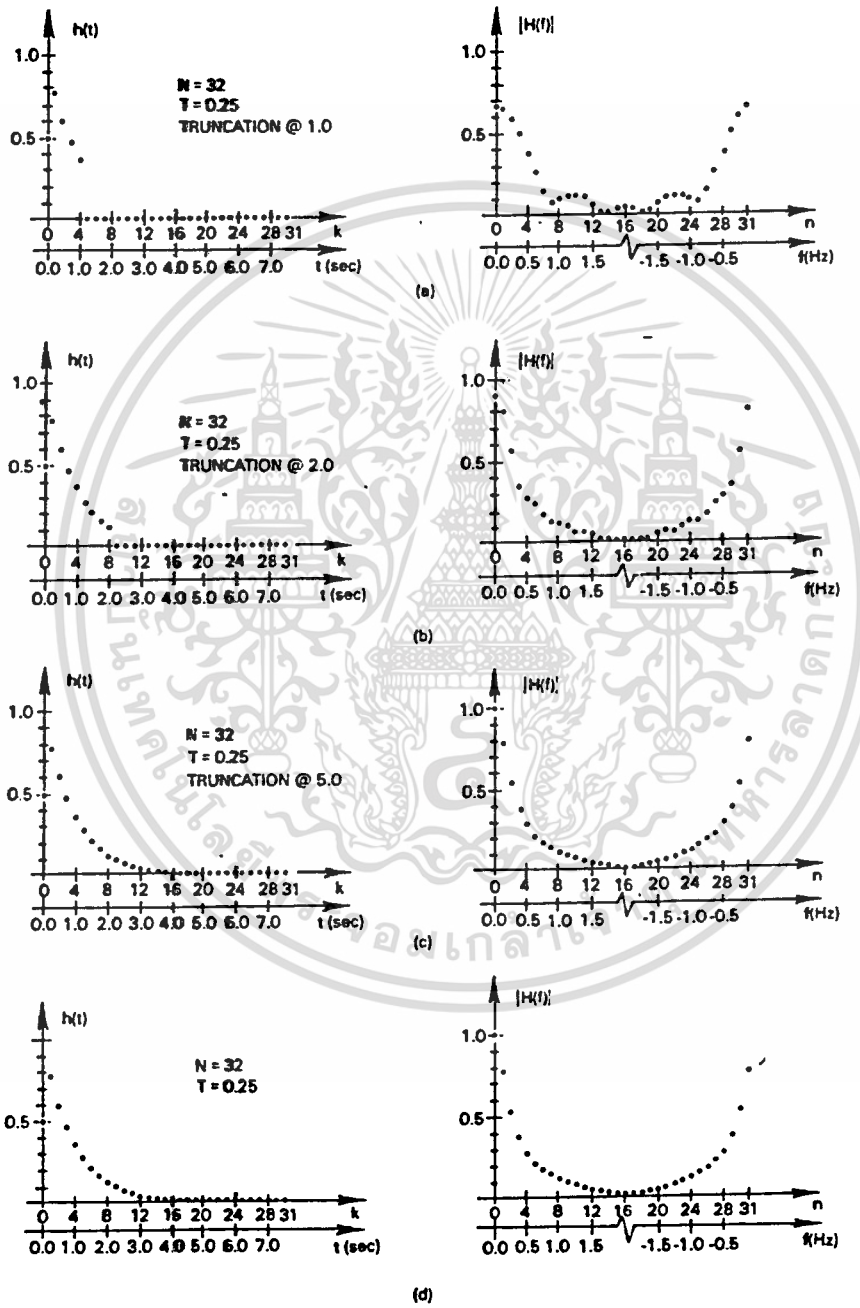
ปัญหาหนึ่งที่เราพบในการคำนวณการแปลงฟูเรียร์จาก FFT คือความผิดเพี้ยน (aliasing) จากที่เคยกล่าวมาแล้ว ในการสุ่มข้อมูล ความผิดเพี้ยนจะเกิดขึ้นเมื่อจุดสุ่มของฟังก์ชันเวลามีระยะห่างของการเก็บมากเกินไป ผลของฟังก์ชันความถี่ที่ผิดเพี้ยนนี้เรียกว่า การพับ(fold)หรือการซ้อน(overlap) ในตัวมันเอง ซึ่งได้แสดงตัวอย่างไว้ในรูปที่5.5

ในรูปที่5.5 เรามีจุดสุ่มของฟังก์ชัน $h(t) = e^{-t}, t>0$ โดยช่วงเวลาการสุ่ม T = 1.0,0.5 และ 0.25 ตามลำดับ จำนวนข้อมูลตั้งให้เท่ากับ 32 สำหรับแต่ละกรณี ขนาดของการแปลงฟูเรียร์ที่ได้ คำนวณโดยใช้ FFT ซึ่งได้แสดงรวมไว้ในรูปที่5.5(a) ถึง5.5(c) แล้ว จากผลที่ได้จาก FFT จะเห็นว่า เมื่อ T = 1.0 ผลที่ได้มีความผิดเพี้ยนมากที่สุด (ขนาดของการแปลงฟูเรียร์แบบต่อเนื่องได้แสดงไว้ในรูปที่5.5(d)) จะเห็นว่าความผิดเพี้ยนมีค่าลดลง เมื่อมีคาบเวลาเท่า

กับ 0.5 และเมื่อคาบเวลาเท่ากับ 0.25 ผลที่ได้จะมีความคล้ายคลึงกับผลที่ได้จากการคำนวณแบบต่อเนื่องมากที่สุด



รูปที่ 5.5 แสดงตัวอย่างของความผิดเพี้ยนในโดเมนความถี่จากอัตราการสุ่ม



รูปที่ 5.6 *แสดงการตัดทอนในโดเมนเวลา*

ในรูปที่5.5 ได้แสดงหลักการลดความผิดเพี้ยน โดยการลดช่วงการสุ่ม ในกรณีนี้จะเห็นว่าไม่มีผลของการตัดทอน เนื่องจาก T มีค่าต่ำ จึงทำให้ NT มีค่ามากกว่า ความกว้างของช่วงไม่เป็นศูนย์(nonzero interval)ของฟังก์ชันมาก

5.5.3การตัดทอนของ FFT ในโดเมนเวลา(FFT Time-Domain Truncation)

ปัญหาความผิดพลาดอื่นที่มักพบจากการประยุกต์ใช้ FFT ในการแปลงฟูเรียร์ คือความผิดพลาดเนื่องจากการตัดทอนข้อมูลในโดเมนเวลา(time-domain trumcation) ความผิดพลาดนี้เกิดขึ้นจากการเลือกกลุ่มข้อมูลจุดสุ่มเพื่อหาลักษณะของฟังก์ชันเวลาไปตัดทอนลักษณะบางส่วนของรูปคลื่นเริ่มต้น รูปที่5.6(a) และรูปที่5.6(b) แสดงตัวอย่างของจุดนี้ เมื่อเราตัดทอนฟังก์ชัน h(t) ที่ NT = 1,2,5 S. ตามลำดับ ขนาดของการแปลงฟูเรียร์ที่ผ่าน FFT ได้แสดงในรูป

การตัดทอนที่ 1 วินาที ทำให้ผลที่ได้จาก FFT มีการกระเพื่อม(rippling) เมื่อใช้ช่วงเวลาของการตัดทอนนานขึ้น เท่ากับ 2 วินาที ผลของ FFT ที่ได้มีการกระเพื่อมน้อยลง และเมื่อเพิ่มช่วงเวลาการตัดทอนให้เป็น 5 วินาที ผลของ FFT ที่ได้ไม่ปรากฏผลของการเกิดการกระเพื่อม เหมือนกันรูปที่ถูกต้องในรูปที่ 5.6(c)

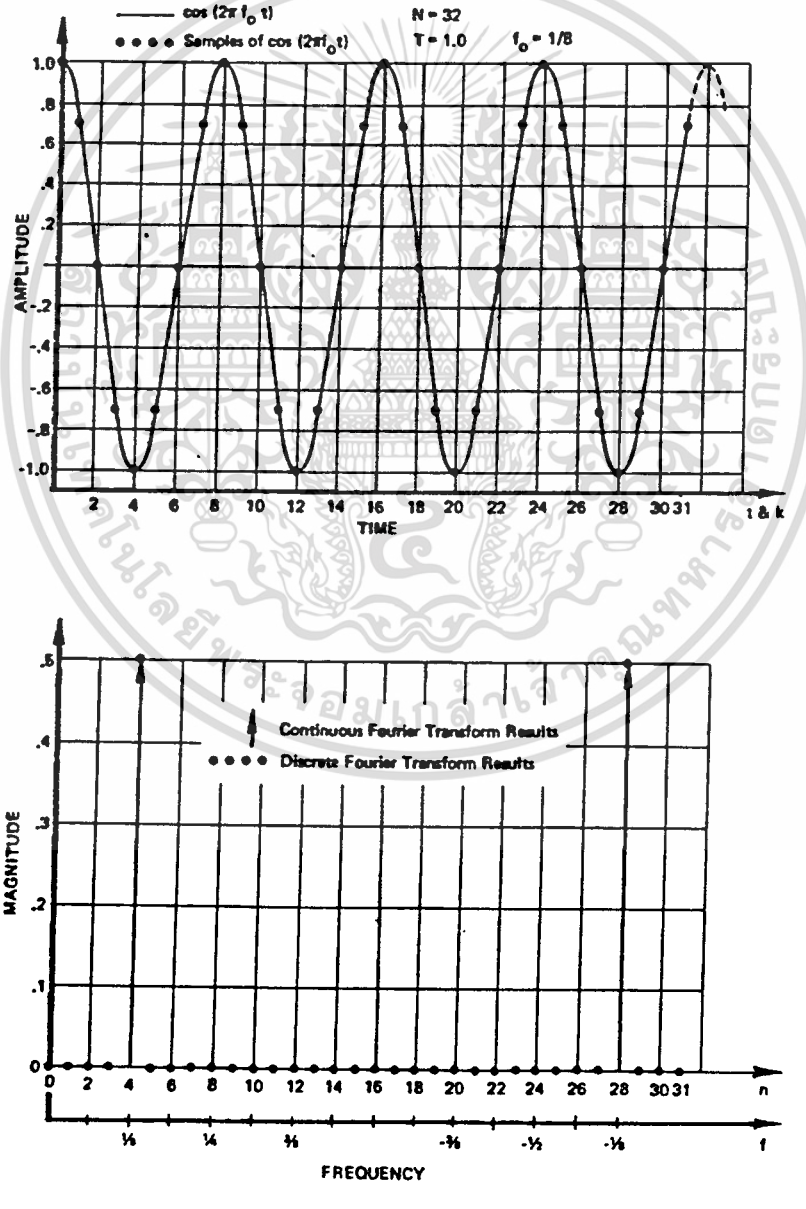
รูปที่5.6 แสดงผลของการทดลอง โดยการพิจารณาหาช่วงเวลาการตัดทอนที่ดีที่สุด โดยการเพิ่มช่วงเวลาการตัดทอน ซึ่งเราก็ได้เห็นผลของการลดลงของการกระเพื่อม

5.5.4 FFT ของฟังก์ชันคาบเวลา

การคำนวณหา FFT ของฟังก์ชันคาบเวลา เราจะต้องเลือกช่วงการสุ่มและช่วงการตัดทอน จากที่กล่าวมาแล้ว การเลือก T ต้องเลือกให้เหมาะสมเพื่อไม่ให้เกิดความผิดเพี้ยน การตัดทอนของฟังก์ชันเวลาคาบเวลาเป็นปัญหาใหม่ที่ไม่ได้อธิบายไว้ในหัวข้ออื่นมาก่อน

ถ้าเราเลือกจำนวนจุดสุ่ม N ของการแปลงฟูเรียร์ เท่ากับ หนึ่งคาบเวลาพอดีหรือเท่ากับจำนวนเท่าของคาบเวลา ผลของความถี่ที่ได้จะตรงกับสัญญาณเริ่มต้น

เพื่ออธิบายจุดนี้ เราทำการคำนวณ FFT ของฟังก์ชันโดซายน์ในรูปที่5.7(a) โดยมีช่วงการสุ่ม T = 1.0 S และจำนวนจุดสุ่มเท่ากับ 32 และแสดงผลลัพธ์ที่ได้ในรูปที่5.7(b) โดยแสดงเป็นขนาดที่ได้จาก FFT ของจุดเหล่านี้ จากรูปจะเห็นว่า จุดอื่นเป็นศูนย์หมด นอกจากจุดที่เป็นความถี่จริง



รูปที่5.7 *แสดงผลจาก FFT ของฟังก์ชันคาบเวลา*
โดยมีช่วงการตัดทอนเท่ากับจำนวนเท่าของคาบเวลา

5.6ฟังก์ชันถ่วงน้ำหนักข้อมูล(data weighting function)

จากหัวข้อที่แล้ว การตัดทอน(truncation) อาจจะทำให้ผลจากการแปลงฟูเรียร์คลาดเคลื่อน การประมวลผลข้อมูลจำนวนจำกัด N ของฟังก์ชันเวลาที่ไม่รู้ชนิดเวลาแน่นอน จำเป็นต้องใช้วินโดว์ของข้อมูลหรือที่เรียกว่าฟังก์ชันถ่วงน้ำหนักข้อมูล ในหัวข้อนี้ เราจะได้อธิบายเทคนิคการใช้ฟังก์ชันถ่วงน้ำหนักเพื่อลดผลเสียที่เกิดจากการตัดทอนให้เหลือน้อยที่สุด

5.6.1ฟังก์ชันถ่วงน้ำหนักสี่เหลี่ยม(rectangular weighting function)

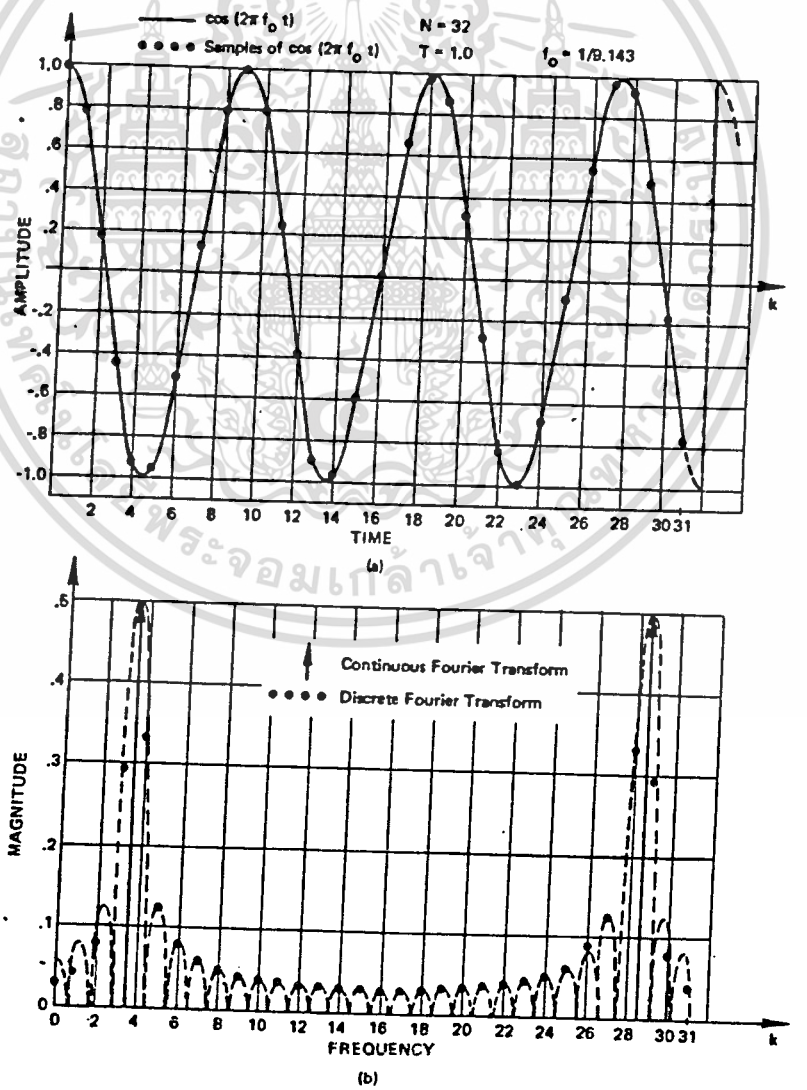
จากรูปที่5.2(a) เราสุ่มสัญญาณซายน์โดยการคูณด้วยอิมพัลส์ฟังก์ชัน(impulse function) ซึ่งได้แสดงไว้ในรูปที่5.2(b) นำผลที่ได้(รูปที่5.2(c))ไปคูณด้วยฟังก์ชันตัดทอนสี่เหลี่ยม ซึ่งแสดงไว้ในรูปที่5.2(d) เพื่อจำกัดจำนวนของจุดสุ่มให้เหลือเท่ากับ N ในที่นี้การตัดทอนของโดเมนเวลา(time domain) ก็คือการถ่วงน้ำหนักข้อมูลโดยคูณด้วยฟังก์ชันสี่เหลี่ยม ผลของการตัดทอนโดเมนเวลาได้แสดงไว้ในรูปที่5.2(e) เรารู้ว่าฟังก์ชันโดเมนความถี่(frequency domain)ของฟังก์ชันสี่เหลี่ยม คือฟังก์ชัน[sin(f)/f] ดังนั้นการแปลงเป็นโดเมนความถี่ของฟังก์ชันเวลาที่ตัดทอนแล้ว จึงเป็นการทำคอนโวลูชั่น(convolution) ของฟังก์ชันเวลากับฟังก์ชันสี่เหลี่ยม ทำให้ผลที่ได้มีองค์ประกอบของความถี่เพิ่มขึ้นมา ซึ่งก็คือลอนข้าง(side lobe)ของฟังก์ชันสี่เหลี่ยม องค์ประกอบที่เพิ่มขึ้นมานี้เรียกว่า ส่วนรั่วไหล(leakage) ทำให้อิมพัลส์ของความถี่ของฟังก์ชันรั่วออกไปที่ลอนข้างของฟังก์ชัน [sin(f)]/f

ถึงแม้ว่าฟังก์ชันในโดเมนเวลาเริ่มต้นเป็นสัญญาณซายน์ แต่เมื่อผ่านการสุ่มแล้ว รูปคลื่นของสัญญาณที่ถูกสุ่มในโดเมนเวลาจะไม่เป็นสัญญาณซายน์อีก เพราะว่าช่วงของการตัดทอน(truncation interval) ไม่เท่ากับคาบเวลาหรือจำนวนเท่าของคาบเวลา(เป็นจำนวนเต็ม) ทำให้คอนโวลูชั่นในโดเมนเวลาในรูปที่5.2(e)และ(f) ไม่ตรงกับฟังก์ชันคาบเวลาเริ่มต้น เนื่องจากการทำคอนโวลูชั่นของฟังก์ชันเวลาเป็นแบบไม่ต่อเนื่อง(discontinuous) จึงได้แสดงผลของการเกิด

การกระเพื่อม(rippling effect) ไว้ในแบบไม่ต่อเนื่องในรูปที่5.2(g) ด้วย

ต่อไป เราสาธิตตัวอย่างของผลที่เกิดจากการใช้ฟังก์ชันถ่วงน้ำหนักสี่เหลี่ยมดังนี้

สมมติว่า เราต้องคำนวณ FFT ของฟังก์ชันโคซาน์(cosine function) ซึ่งแสดงไว้ในรูปที่9.7(a) โดยมีคาบเวลา T = 1.0 และ N = 32 ในรูปที่9.7 (b) เราได้แสดงขนาดของการแปลงดิสครีตฟูเรียร์(discrete fourier transform) ของจุดสุ่มในรูปที่9.7(a) FFT จะทำให้ผลเป็นความถี่ไม่ต่อเนื่องทางด้านบวกทั้งหมด จากที่ได้อธิบายในหัวข้อที่แล้ว องค์ประกอบของความถี่เพิ่มขึ้นมาที่เรียกว่าส่วนรั่วไหลเป็นผลมาจากลักษณะการมีลอนข้างของฟังก์ชัน [sin(f)] /f



รูปที่5.8 *แสดงผลจาก FFT ของฟังก์ชันคาบเวลา โดยมีช่วงการตัดทอนไม่เท่ากับจำนวนเท่าของคาบเวลา*

เพื่อลดผลจากการรั่วไหล เราจำเป็นต้องใช้วิธีการตัดทอนโดเมนเวลาหรือฟังก์ชันถ่วงน้ำหนักที่มีลักษณะของลอนข้างในโดเมนความถี่เล็กกว่าของฟังก์ชันสี่เหลี่ยม ลอนข้างที่เล็กกว่านี้ทำให้การรั่วไหลของผลลัพธ์จาก FFT ลดน้อยลง เพื่อให้เข้าใจจุดนี้ได้ชัดเจนยิ่งขึ้น ขอให้พิจารณารูปที่5.2 อีกครั้งหนึ่ง ถ้าเปลี่ยนฟังก์ชันตัดทอนในรูปที่5.2(d)เป็นฟังก์ชันที่มีลอนข้างต่ำ จะทำให้การประมาณค่าของการแปลงฟูเรียร์ดีขึ้น * การใช้ฟังก์ชันถ่วงน้ำหนักข้อมูลเพื่อตัดทอนและถ่วงน้ำหนักข้อมูลต้องทำกับจุดสุ่ม N ก่อนคำนวณ FFT

5.6.2 ลักษณะของฟังก์ชันถ่วงน้ำหนักชนิดต่างๆ

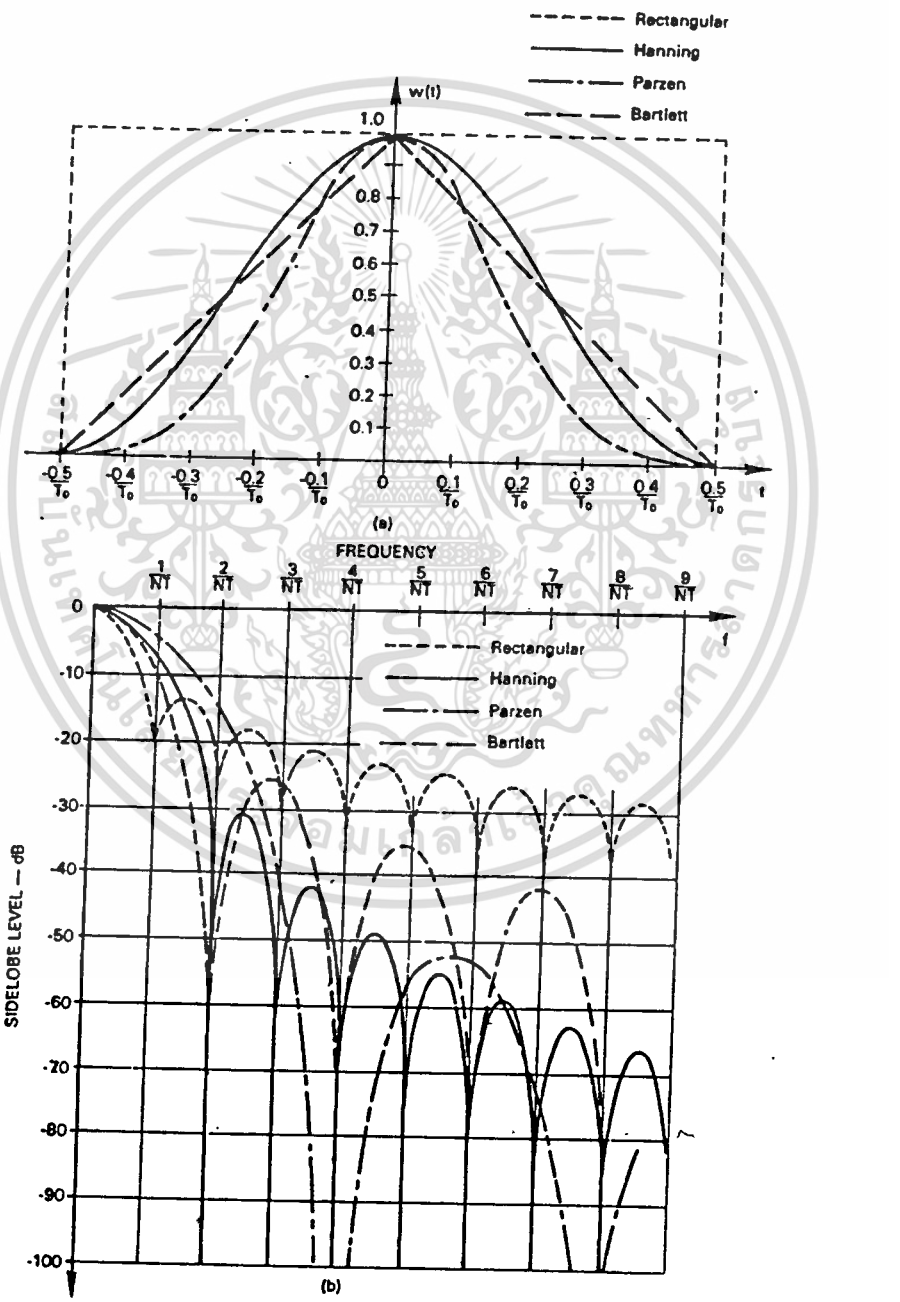
ในรูปที่5.9(a) ได้แสดงฟังก์ชันตัดทอนหรือถ่วงน้ำหนักที่นิยมใช้กับ FFT ผลของการตอบสนองความถี่ของฟังก์ชันได้แสดงไว้ในรูปที่5.9(b)และในตารางที่ 5.2 แสดงข้อมูลการเปรียบเทียบลักษณะของฟังก์ชันถ่วงน้ำหนักแต่ละตัวทั้งในโดเมนเวลาและโดเมนความถี่ โดยมีจุดศูนย์กลางอยู่ที่จุดศูนย์เพื่อให้ง่ายต่อการสังเกต

จากรูปที่5.9(b) จะเห็นว่า ฟังก์ชันถ่วงน้ำหนักทั้งหมดมีลอนข้างในโดเมนความถี่ที่มีขนาดเล็กฟังก์ชันถ่วงน้ำหนักสี่เหลี่ยมทั้งนั้น ซึ่งจะช่วยให้ผลของการรั่วไหลลดน้อยลง แต่อย่างไรก็ตาม ฟังก์ชันถ่วงน้ำหนักทั้งหมดนี้มีลักษณะของลอนหลัก (main lobe) ที่กว้างกว่า ลองดูรูปที่6.4(d)และ(e)อีกครั้ง ลองคิดเปรียบเทียบผลที่ได้เมื่อใช้ฟังก์ชันถ่วงน้ำหนักตามรูปที่5.9(b) จะเห็นว่า ยิ่งลอนหลักมีความกว้างมากขึ้นเท่าใด ผลที่ได้จาก FFT ยิ่งมีความคลุมเคลือมากยิ่งขึ้น นั่นก็หมายความว่าถ้าลอนหลักมีขนาดกว้าง จะทำให้แยกความแตกต่างของแต่ละความถี่ได้ยาก

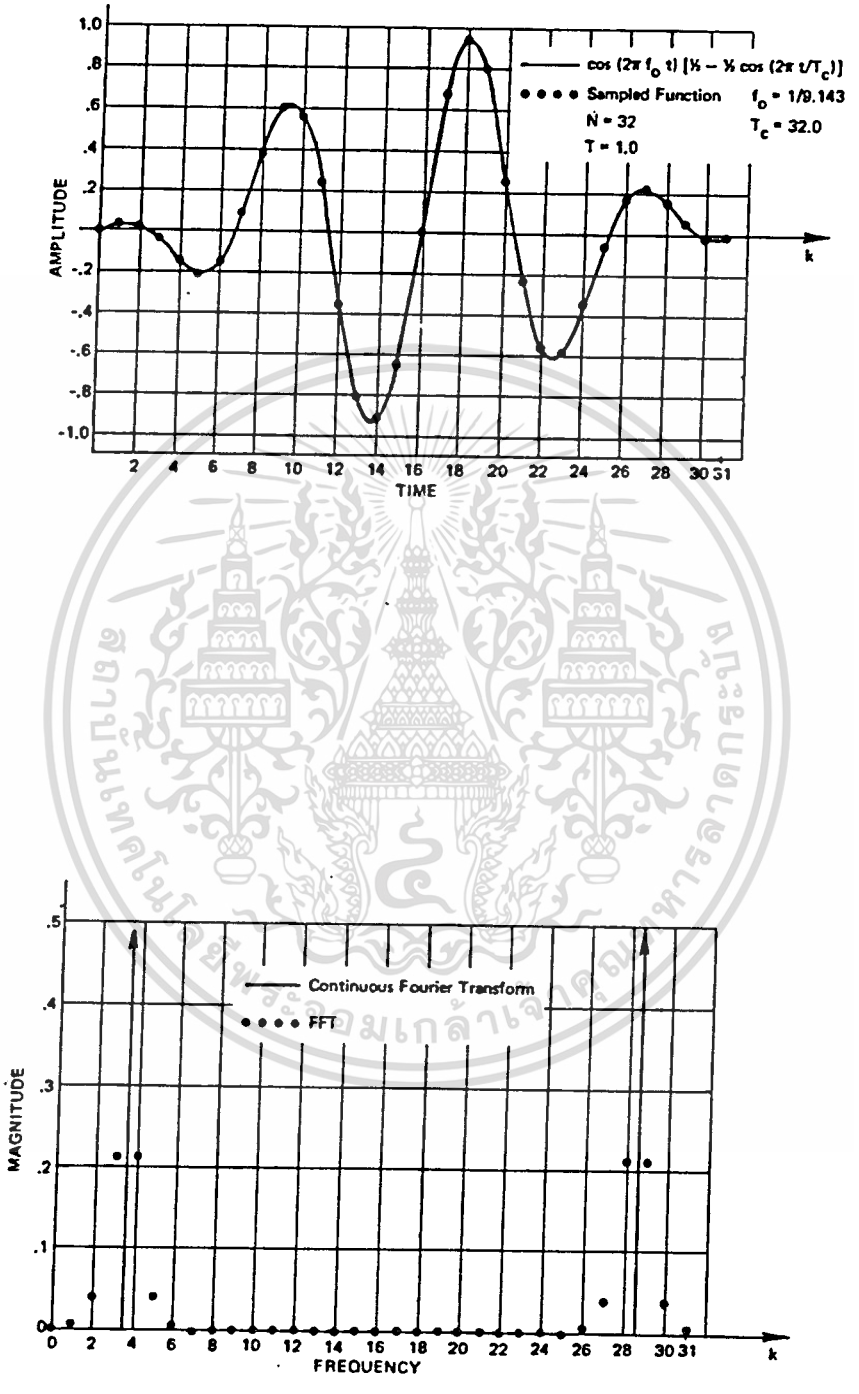
ข้อได้เปรียบเสียเปรียบระหว่างส่วนรั่วไหล(ระดับการรั่วไหล) และความละเอียด(แบนด์วิดธ์ของลอนหลัก) เป็นที่รู้จักกันดีในงานวิทยาศาสตร์สาขาต่างๆ ในตารางที่5.2 แสดงระดับสูงสุดของลอนข้าง และแบนวิดธ์(bandwidth) 3-dB สำหรับแต่ละฟังก์ชันถ่วงน้ำหนัก สำหรับงานการทดลองทั่วไป นิยมใช้ฟังก์ชันแฮนนิ่ง (Hanning function) เพราะว่าโครงสร้างของมันง่ายกว่าแบบอื่น การเลือกใช้ฟังก์ชันถ่วงน้ำหนักที่ให้ผลดีที่สุดต้องขึ้นกับลักษณะของงานที่เรานำไปใช้งานด้วย

เป็นสำคัญ

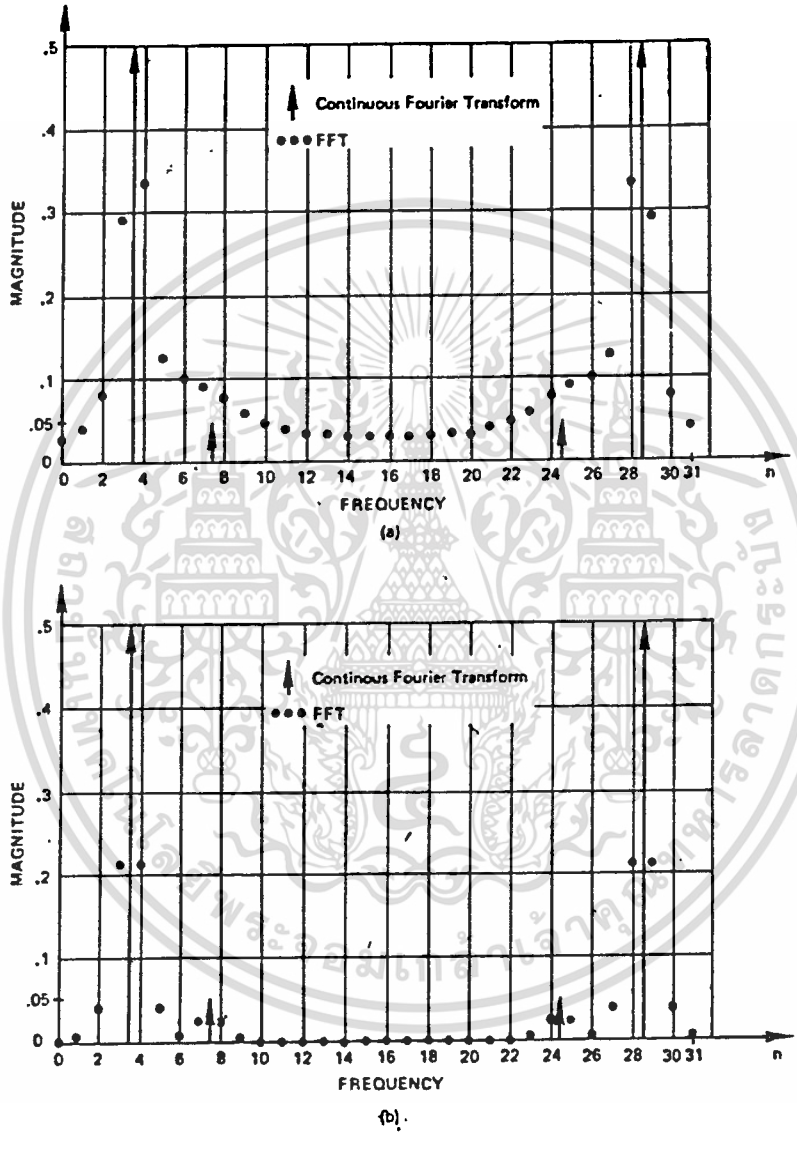
ผลที่ได้จากFFTจะมีช่วงความถี่ $f_o = 1/NT$ ซึ่งอาจทำให้เกิดความสับสนกับค่า $1/NT$ ของฟังก์ชันถ่วงน้ำหนักที่เลือกใช้ ผลของความละเอียดของความถี่ของผลลัพธ์ที่ได้



รูปที่5.9 แสดงฟังก์ชันถ่วงน้ำหนักหรือวินโดว์ของ FFT



รูปที่ 5.10 *แสดงตัวอย่างการใช้แฮนนิ่งฟังก์ชัน เพื่อลดส่วนรั่วไหลของการคำนวณด้วย FFT*



รูปที่5.11 (a) *แสดงตัวอย่างที่เลือนลางเนื่องจากผลจากส่วนรั่วไหลของลอนข้าง*
(b) *สัญญาณที่ตรวจจับได้หลังจากผ่านแฮนนิ่งฟังก์ชันแล้ว*

จาก FFT เป็นเพียงฟังก์ชันของแบนวิดธ์ของฟังก์ชันถ่วงน้ำหนักแต่ละตัวเท่านั้น(ดูรูปที่ 5.9(b)) ดังนั้นการใช้คำนิยามของความละเอียด 1/NT ต้องทำอย่างระมัดระวัง และระลึกอยู่เสมอว่า มันเป็นเพียงช่องว่างระหว่างความถี่ที่เกิดจากผลที่ได้จาก FFT เท่านั้นและไม่เกี่ยวกับ 1/NT ของวินโดว์ที่ใช้

ต่อไป เราจะแสดงผลจากการใช้ฟังก์ชันถ่วงน้ำหนักที่มีลอนข้างต่ำ เพื่อให้เห็นลักษณะการนำไปใช้ลดการรั่วไหลของผลจากการตัดทอนในโดเมนเวลา ในรูปที่ 5.10(a) เราได้แสดงรูปคลื่นโคซายน์ที่แสดงไว้ในรูปที่9.7(a)ที่คูณด้วยฟังก์ถ่วงน้ำหนักแฮนนิ่งที่แสดงไว้ในรูปที่5.9(a)แล้ว

รูปที่5.10(b) แสดงจุดสุ่มจาก FFT ของรูปที่5.10(a) จากรูปจะเห็นว่า ส่วนรั่วไหลลดลงไปอย่างมาก แต่องค์ประกอบหลักของความถี่ มีระยะกว้างขึ้นหรือคลุมเคลือเมื่อเปรียบเทียบกับจุดของอิมพัลส์ฟังก์ชัน เพราะผลจากการคอนโวลูชั่นความถี่ของอิมพัลส์ฟังก์ชันกับผลการแปลงฟูเรียร์ของฟังก์ชันถ่วงน้ำหนัก

บทที่ 6

การวิเคราะห์เสียงพูด(Speech Analysis)

ในการเก็บหรือจดจำเสียงพูด เราต้องพยายามลดช่วงของสัญญาณเสียงพูดที่เกิดซ้ำกัน โดยใช้ตัวแทนของเสียงพูด ที่สามารถแทนลักษณะสำคัญของเสียงพูดนั้นในเทอมของค่าพารามิเตอร์(parameter) เพื่อให้จัดการกับข้อมูลได้ง่าย

ต่อไป เราจะได้อธิบายถึง วิธีการที่ใช้ในการวิเคราะห์เสียงพูด ทั้งในโดเมนเวลา(กระทำกับเสียงพูดที่เก็บไว้โดยตรง)และโดเมนความถี่(ผ่านการแปลงเป็นความถี่ก่อน) เพื่อหาตัวแทนของสัญญาณเสียงพูดในรูปของพารามิเตอร์ที่เหมาะสมที่สุด สามารถนำไปใช้งานได้ และมีข่าวสารของข้อมูลครบถ้วน การวิเคราะห์ในโดเมนเวลา เราต้องการการคำนวณเพียงเล็กน้อย ทำให้ถูกจำกัดให้วัดได้เฉพาะลักษณะง่ายๆเท่านั้น เช่นการวัดพลังงานและความเป็นคาบเวลาของสัญญาณ ในขณะที่การวิเคราะห์ในโดเมนความถี่จะให้ค่าพารามิเตอร์ที่มีประสิทธิภาพมากกว่า แต่การสุ่มสัญญาณเสียงพูดจะมีต้องมีความเที่ยงตรงสูง

เทคนิคที่ใช้ในการวิเคราะห์เสียงพูด อาจทำได้ทั้งแบบดิจิตอลและอนาลอก การประมวลผลสัญญาณอนาลอก เป็นการใช้วงจรทางด้านอิเลคทรอนิกส์ ซึ่งมีข้อดีอยู่ตรงที่มีความเร็วสูง แต่จำเป็นต้องใช้วงจรเฉพาะอย่าง ทำให้ต้องต่อสายใหม่และปรับค่าใหม่ทุกครั้งที่นำไปประยุกต์ใช้กับงานอื่น ในขณะที่เทคนิคทางดิจิตอลสร้างและเปลี่ยนแปลงได้ง่ายกว่า เพียงแต่ต้องการซอฟต์แวร์หรือโปรแกรม และฮาร์ดแวร์พิเศษ(ไมโครโปรเซสเซอร์และชิพ) ถึงแม้ว่า อาจจะมีปัญหาในเรื่องความเร็ว ไม่สามารถตอบสนองในเวลาจริงได้ แต่ในปัจจุบันเทคโนโลยีทางด้าน VLSI ได้พัฒนาไปอย่างมาก ทำให้ลดข้อเสียเปรียบของเทคนิคดิจิตอลไปได้มาก สำหรับงานในด้านการทดลอง ขอแนะนำให้ใช้ดิจิตอลเพราะแก้ไขได้ง่ายกว่า

6.1 การวิเคราะห์เสียงพูดในช่วงสั้น(Short-time speech analysis)

สัญญาณเสียงพูดเป็นสัญญาณที่เปลี่ยนไปตามเวลา โดยเกิดในลักษณะแบบสุ่ม

(random) แต่ก็ต้องขึ้นกับการควบคุมเสียงของผู้พูดด้วย เพราะเสียงที่เปล่งออกมาในช่วงระยะเวลาหนึ่งนั้น จะขึ้นกับรูปร่างของท่อกาธรเสียง (vocal tract) และลักษณะการสั่นของเส้นเสียง (vocal cord) สัญญาณของเสียงพูดจึงเป็นสัญญาณที่เป็นคาบเวลาชั่วขณะ (quasi-periodic) หมายความว่า สัญญาณเสียงพูดมีคาบเวลาคงที่ในระยะเวลาสั้นๆ และมีการเปลี่ยนแปลงในระหว่างระยะเวลาสั้นๆ นั้น

ถ้าเราพูดช้ามากๆ เสียงที่พูดที่ได้อาจจะเปลี่ยนแปลงอยู่ในช่วงมากกว่า 200 ms ก็ได้ แต่การพูดโดยทั่วไป ลักษณะการเปลี่ยนแปลงของเสียงพูดจะอยู่ช่วงค่าเฉลี่ยประมาณ 80 ms ในการวิเคราะห์ เราจะใช้วินโดว์ (window) วิเคราะห์สัญญาณเป็นช่วงๆ หรือเรียกว่า การวิเคราะห์เฟรม (frame) ช่วงเวลาของวินโดว์มีค่าไม่แน่นอน ในบางครั้งอาจต้องใช้ถึง 100 ms แต่ในบางครั้งเมื่อนำวิเคราะห์ สัญญาณเสียงพูดที่เร็วมากและไม่มีช่วงหยุดเลย เราอาจจะต้องใช้ช่วงเวลาของวินโดว์ต่ำถึง 5-10 ms

6.1.1 การใช้วินโดว์ (windowing)

รูปแบบการของการเฉลี่ย โดยปกติมีวิธีหลายอย่างเพื่อให้ได้เส้นของพารามิเตอร์ที่เป็นฟังก์ชันของเวลา แสดงได้อย่างถูกต้อง ทางเลือกทางหนึ่งที่เลือกใช้คือขนาดของวินโดว์ซึ่งมีปัจจัยที่เกี่ยวข้องคือ

(1) วินโดว์จะต้องเล็กพอๆ กับคุณลักษณะของคำพูด

(2) วินโดว์จะต้องยาวพอที่จะใช้ในการคำนวณหาพารามิเตอร์

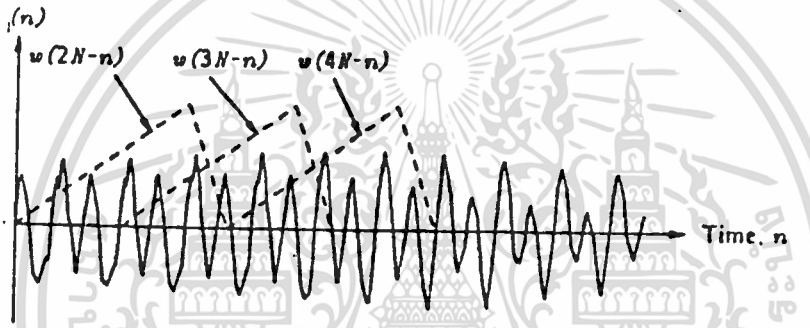
(3) วินโดว์ที่ดีจะต้องไม่สั้นไปจนข้ามบางช่องของคำพูด การวิเคราะห์จะทำเป็นคาบๆไป ซ้ำๆกันตลอดที่ได้สัญญาณ

เงื่อนไขท้ายนี้จะขึ้นกับ frame rate (จำนวนครั้งต่อวินาที ที่กระทำต่อสัญญาณที่วิเคราะห์) มากกว่าขนาดของวินโดว์ โดยปกติ frame rate ประมาณ 2 เท่าของความถี่เพื่อให้วินโดว์ทับกัน 50 %

การทำวินโดว์เป็นการคูณสัญญาณเสียงโดยวินโดว์ที่มีช่วงเวลาจำกัด (finite-duration window) w(n) ซึ่งกลุ่มของสัญญาณเสียงที่สุ่มมาจะถูกให้น้ำหนักโดยรูปของวินโดว์

วินโดว์อาจมีความยาวแบบไม่จำกัด แต่ในทางปฏิบัติจะใช้มีแบบจุดสิ้นสุด เพื่อให้ง่ายในการคำนวณโดยการใช้ฟังก์ชัน w(n) ตรวจสอบส่วนต่างๆของ S(n) โดยการเคลื่อนวินโดว์ ตามรูป 6.1 วินโดว์ที่มีรูปแบบง่ายที่สุดคือรูปแบบสี่เหลี่ยม r(n)

$$w(n) = r(n) = 1 \text{ เมื่อ } 0 \leq n \leq N-1 \qquad (6.1)$$
$$0 \text{ อื่นๆ}$$



รูปที่6.1 แสดงสัญญาณเสียง s(n) กับวางทับสัญญาณของวินโดว์ 3 ตัว โดยมีจุดเริ่มต้นของเวลาที่จุดสุ่ม 2N, 3N และ 4N

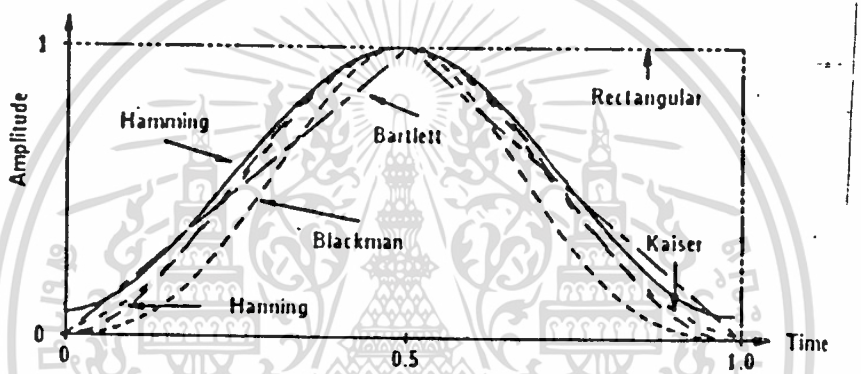
การวิเคราะห์แบบนี้เป็นการจำกัดช่วงจุดสุ่มให้เหลือ N โดยแต่ละจุดสุ่มมีการถ่วงน้ำหนักเท่าๆกัน

วินโดว์ที่ใช้ในการจริงนั้นต้องมีรูปร่างตามรูปที่ 6.2 สมมติว่าเสียงพูดประมาณว่าคงที่ในช่วง 10ms เราต้องใช้วินโดว์ที่ช่วงเวลาเท่ากับ 20 ms โดยให้จุดตรงกลาง 10 ms ถ่วงน้ำหนักมากกว่าจุดต้นและจุดปลาย เหตุผลที่ต้องถ่วงน้ำหนักจุดกลางมากกว่าจุดปลาย เพราะรูปร่างของวินโดว์มีผลต่อเอาท์พุทต่อพารามิเตอร์เสียงพูด เมื่อเราเลื่อนวินโดว์ในโดเมนเวลาเพื่อวิเคราะห์เฟรมของสัญญาณเสียงพูด ผลของพารามิเตอร์อาจเปลี่ยนแปลงอย่างมากถ้าใช้ฟังก์ชัน r(n) ต่างกัน

ตัวอย่างเช่นการพลังงานโดยการบวกผลกำลัง 2 ของจุดสุ่มในวินโดว์ของสี่เหลี่ยม ทำให้เกิดการกระเพื่อมรอบจุดความถี่ ดังนั้นเราจะใช้แฮนนิ่งวินโดว์ (Hanning window) แทนวินโดว์สี่เหลี่ยมซึ่งมีรูปร่างเท่ากัน

$$w(n) = h(n) = 0.54-0.46\cos(2\pi n/N-1) \text{ เมื่อ } 0 \leq n \leq N-1 \quad (6.2)$$

$$0 \text{ อื่นๆ}$$



รูปที่ 6.2 *รูปแบบของวินโดว์ ซึ่งมีช่วงเวลาเท่ากับหนึ่ง*

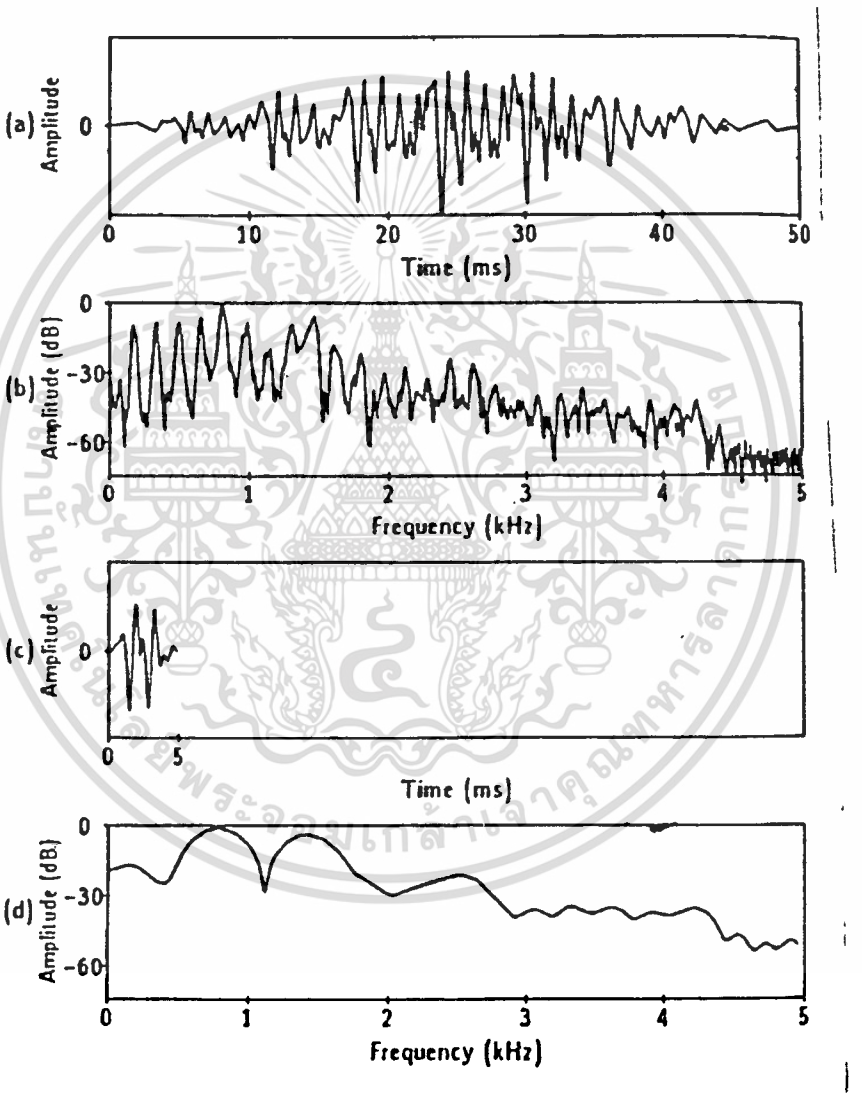
การทำให้ยอบของวินโดว์ลาดลง ในการวิเคราะห์เฟรมโดยการเลื่อนตลอดแนวความยาวของสัญญาณทั้งหมด ไม่มีผลเสียต่อพารามิเตอร์ของเสียงพูด

6.2 พารามิเตอร์ในโดเมนเวลา (Time-Domain Paramiter)

ในขบวนการของสัญญาณเสียงในฟังก์ชันเวลาจะมีความได้เปรียบในด้านความง่าย คำนวณเร็ว และง่ายในการแปลงความหมาย พารามิเตอร์ของเสียงสำหรับการเข้ารหัสและการจำสามารถจะทำได้จากการวิเคราะห์ฟังก์ชันเวลา เช่น พลังงาน (หรือแอมปลิจูด), เสียง

6.2.1 การวิเคราะห์ในโดเมนเวลา

การวิเคราะห์ฟังก์ชันเวลาที่เปลี่ยนจากสัญญาณเสียงสู่พารามิเตอร์ 1 ตัวหรือมากกว่า ซึ่งโดยปกติจะช้ากว่าสัญญาณเริ่มแรกมาก การเก็บและการ manipulate ใน



รูปที่6.3 (a) *แสดงสัญญาณที่ถูกคูณด้วยแฮมนิ่งวินโดว์ 50 ms*

(b) *สเปคตรัมของรูป(a)*

(c) *แสดงสัญญาณที่ถูกคูณด้วยแฮมนิ่งวินโดว์ 5 ms*

(d) *สเปคตรัมของรูป(c)*

กรณีของสัญญาณเสียงจะมีประสิทธิภาพมากกว่าการเก็บสัญญาณเสียงโดยตรง ตัวอย่างเช่น คำพูดที่ถูกสุ่มมาที่ 6000-10000 ตัวอย่าง/วินาที เพื่อที่จะเก็บช่วงความถี่ให้ถึง 3-5 kH_Z และโดยปกติแล้วการเปล่งเสียง 100 ms จะต้องการถึง 1000 ตัวอย่าง เพื่อให้ได้ความถี่ถูกต้องในการแสดงผลออกมา การสุ่มสัญญาณพารามิเตอร์โดยทั่วไปจะสุ่มที่ 40-100 ตัวอย่าง/s (มีบางระบบที่ต้องสุ่มถึง 200 ตัวอย่าง/s ดังนั้นเมื่อเปลี่ยนคลื่นเสียงไปสู่กลุ่มพารามิเตอร์ อัตราการสุ่มสามารถลดลง

เทคนิคการประมวลผลแบบ short-time ส่วนมาก จะสร้างพารามิเตอร์อยู่ในรูปแบบ

$$Q(n)=\sum_{m=-\infty}^{\infty} T[S(m)]w(n-m) \qquad (6.3)$$

สัญญาณเสียง S(n) ได้รับการเปลี่ยนเป็นฟังก์ชัน T ซึ่งถูกให้น้ำหนักโดยวินโดว์ w(n) และถูกรวมกัน (sum) เป็นๆ พารามิเตอร์ Q(n) ที่อัตราการสุ่มเริ่มต้นซึ่งจะเป็นตัวแทนแสดงสมบัติของคำพูด ที่ถูกเฉลี่ยภายใต้ช่วงของวินโดว์ Q(n) ได้มาจากการคอนโวลูชั่นของ T[S(n)] กับ w(n) เพื่อที่จะกระจายฟังก์ชันนั้น w(n) ใช้เป็น low pass filter ,Q(n) จะถูกทำให้ลายเรียบลงไปในลักษณะของฟังก์ชัน T[S(n)] เดิม เมื่อ Q(n) เป็นเอาท์พุทของวงจรกรองผ่านความถี่ต่ำ low pass filter (วินโดว์) ในกรณีส่วนใหญ่แบนด์วิดของ Q(n) จึงเหมือนกับของวงจรกรองผ่านความถี่ต่ำ w(n) เพื่อการทำงานและการจัดเก็บมีประสิทธิภาพมากขึ้น Q(n)

6.3 ค่าพารามิเตอร์ในโดเมนความถี่

ค่าพารามิเตอร์สำคัญจำนวนมากถูกพบในโดเมนความถี่ เพราะท่อกำเนิดเสียงให้กำเนิดสัญญาณที่สามารถวิเคราะห์เป็นความถี่ได้ง่ายกว่าทำในโดเมนเวลา

โดยตรง เสียงพูดของคนคนเดียวกันในแต่ละครั้ง เมื่อพิจารณาในโดเมนเวลาจะเห็นผลของความแตกต่างของสัญญาณ แต่เมื่อพิจารณาในโดเมนความถี่ สัญญาณที่ได้มีความคล้ายคลึงกันมาก

ระบบการได้ยินของมนุษย์จะตอบสนองต่อรูปร่างของสัญญาณเสียง (หรือขนาดที่กระจายอยู่ในโดเมนความถี่) มากกว่าที่เฟสของสัญญาณในโดเมนเวลา ด้วยเหตุผลเหล่านี้ การวิเคราะห์ความถี่จึงถูกใช้ในการดึงค่าพารามิเตอร์สำคัญจากสัญญาณเสียงพูด

ต่อไป จะได้อธิบายถึงวิธีการต่างๆ ที่จะนำไปใช้ในการวิเคราะห์ความถี่

6.3.1การวิเคราะห์ด้วยฟิลเตอร์แบงค์(filter bank)

การวิเคราะห์ความถี่ด้วยฟิลเตอร์แบงค์ เป็นเทคนิคที่นิยมกันมาก เพราะสามารถตอบสนองได้ในเวลาจริง โครงสร้างง่ายและไม่แพง โดยใช้ฟิลเตอร์แบงค์หรือกลุ่มของแบนพาสฟิลเตอร์(bandpass filter)ซึ่งอาจจะเป็นอนาลอกฟิลเตอร์หรือดิจิตอลฟิลเตอร์ก็ได้ แบนพาสฟิลเตอร์แต่ละตัวจะแยกวิเคราะห์ความถี่ของสัญญาณเสียงในช่วงๆ ในทางด้านการคำ วิธีฟิลเตอร์ มีความยืดหยุ่นมากกว่าการวิเคราะห์ด้วย DFT เพราะช่วงแบนวิดธ์(bandwidth) สามารถปรับช่วงตามขอบเขตการรับฟังเสียงของหูมนุษย์ได้มากกว่าการกำหนดตายตัวว่าต้องมีแบนวิดธ์กว้างแคบเท่านี้เท่านั้น

ในการประยุกต์ใช้งานจำนวนมาก ต้องการกลุ่มของพารามิเตอร์ความถี่ไม่มาก ในการอธิบายการกระจายของพลังงานของความถี่ โดยทั่วไปการใช้กลุ่มของแบนพาสฟิลเตอร์ 8-12 ตัว ก็สามารถให้ค่าตัวแทนของความถี่ที่ซับซ้อนและมีประสิทธิภาพมากกว่าการวิเคราะห์ด้วย DFT ซึ่งให้รายละเอียดปลีกย่อยของแต่ละความถี่ได้มากกว่า

ในวิธีการทั่วไป ฟิลเตอร์จะมีการวางระยะตามบาร์คสเกล(bark scale) คือ ให้มีระยะห่างความถี่ของฟิลเตอร์แต่ละตัวเท่าๆกัน และมีแบนวิดธ์ที่เพิ่มขึ้นแบบลอกการึทึม(loarithm) แต่ไม่ควรเกิน 1 kHz โดยทั่วไปนิยมใช้ฟิลเตอร์ 1/3 ออคเทฟ(one-third-octave filter) ในระบบการจดจำเสียงพูดจริงๆนั้น

อาจจะใช้ฟิลเตอร์แบงค์ถึง 2 ครั้ง ครั้งแรกเป็นการจำแนกเสียงอย่างคร่าวๆ ก่อน โดยใช้ฟิลเตอร์เพียงไม่กี่ตัว และวิเคราะห์หารายละเอียดปลีกย่อยอีกครั้ง โดยใช้กลุ่มของฟิลเตอร์มากขึ้น

6.3.2 การวิเคราะห์ฟูเรียร์ช่วงสั้น(Short_Time Fourier Analysis)

การวิเคราะห์ฟูเรียร์ช่วงสั้นเป็นเทคนิคการหาความถี่ที่ใช้กันมานานแล้ว การวิเคราะห์ฟูเรียร์ให้ตัวแทนของสัญญาณเสียงพูดเป็นฟังก์ชันความถี่ในเทอมของขนาดและเฟส เนื่องจากเสียงพูดไม่ได้นิ่งตลอดเวลา ดังนั้นจึงจำเป็นต้องใช้การวิเคราะห์ในช่วงสั้น โดยใช้วินโดว์

การแปลงฟูเรียร์ช่วงสั้น มีสมการว่า

$$S_n(e^{jw}) = \sum_{m=-\infty}^{\infty} s(m)(e^{-jwm})w(n-m) \qquad (6.4)$$

ในการคำนวณ เราต้องใช้ DFT แทนการแปลงฟูเรียร์แบบต่อเนื่อง โดยใช้ฟังก์ชันวินโดว์ w ลดข้อมูลแบบไม่ต่อเนื่องทั้งหมดให้เหลือจำนวน N ตัว (N คือช่วงเวลาหรือขนาดของวินโดว์ที่ใช้ในการแปลง DFT) ข่าวสารต่างๆใน $S_n(e^{jw})$ จะไม่สูญหายไปจากข้อมูลเดิม $s_n(e^{jw})$ ถ้าการแปลงนั้นสุ่มมาด้วยความถี่สูงเพียงพอ (คือ ช่วงระยะห่างระหว่าง N) และวินโดว์ w(n) ไม่มีจุดสุ่มที่เป็นศูนย์ตลอดช่วง N ตัวแปร N เป็นตัวแปรที่ต้องระวังมากเป็นพิเศษในการวิเคราะห์ความถี่ช่วงสั้น ถ้าค่าของ N ต่ำ(ใช้วินโดว์ช่วงสั้น) จะทำให้ความละเอียดในโดเมนความถี่จะหยาบมาก เพราะจะให้ผลที่ดีในโดเมนเวลา เพราะการเฉลี่ยถูกทำเฉพาะในช่วงสั้นๆเท่านั้น ในทางตรงกันข้าม ถ้า N มีขนาดใหญ่ จะให้ผลความละเอียดของเวลาที่แย่ แต่จะทำให้โดเมนความถี่มีความละเอียดสูงกว่า

บทที่ 7

การจดจำ(recognition)

7.1 ประเภทของการจดจำ

ในการประมวลผลสัญญาณเสียงพูด มีการประยุกต์ใช้ในงานสำคัญ 2 ด้าน คือ

1.การจดจำเสียงพูด(Speech recognition)

2.การจดจำผู้พูด(Speaker recognition)

ทั้งสองหัวข้อนี้เป็นการจดจำรูปแบบอย่างหนึ่ง การจดจำรูปแบบได้ถูกนำไปประยุกต์ใช้ในงานหลายๆด้าน เช่น การประมวลผลสัญญาณภาพ แต่ในที่นี้จะอธิบายถึงการจดจำรูปแบบที่เกี่ยวข้องกับการประมวลผลสัญญาณเสียงพูดเท่านั้น

7.2 การจดจำรูปแบบ(Pattern recognition)

การจดจำรูปแบบเป็นการเปรียบเทียบรูปแบบตรวจสอบ(test pattern) ซึ่งเป็นตัวแทนของสิ่งที่ไม่รู้เพื่อแยกแยะหรือจดจำ กับรูปแบบอ้างอิง(reference pattern) 1 ตัวหรือหลายตัว ซึ่งได้รู้ลักษณะมาก่อนแล้ว

ในแต่ละรูปแบบจะอยู่ในรูปของเวคเตอร์ แต่ละสมาชิกของเวคเตอร์เป็นค่าที่วัดได้ของลักษณะเด่น(feature) ลักษณะเด่นเป็นลักษณะที่วัดได้ของลักษณะสัญญาณอินพุทและสามารถนำไปใช้ประโยชน์ในการจดจำรูปแบบได้ เช่น ในระบบการจดจำเสียงพูดแบบแยกคำ รูปแบบต่างๆจะเก็บอยู่ในรูปของกลุ่มฟังก์ชันเวลา เพราะค่าของแต่ละลักษณะนั้นวัดได้นั้นไม่ได้วัดที่จุดใดจุดหนึ่งของคำพูดแต่มาจากการวัดทั้งคำ รูปแบบเหล่านี้เราเรียกว่า ต้นแบบ(template)

ในระบบการจดจำจะมีการทำงานคล้ายตัวอย่างดังต่อไปนี้

-สมมุติว่า ระบบการจดจำผู้พูดเป็นระบบที่ทำขึ้นสำหรับผู้พูด 50 คน เมื่อมีการพูดคำหรือวลี ระบบจะทำการประมวลผลสัญญาณเสียงพูดและดึงเอาลักษณะเด่นออกมา(โดยแยกคำและวัดค่า)เพื่อทำเป็นรูปแบบอ้างอิง ดังนั้นระบบนี้จึงมีไล

บรารี่รูปแบบอ้างอิงอยู่ 50 แบบ เมื่อทำการเปรียบเทียบรูปแบบตรวจสอบจะถูกเปรียบเทียบกับรูปแบบอ้างอิงในไลบรารี่ทั้งหมด เพื่อหารูปแบบอ้างอิงที่ใกล้เคียงกับรูปแบบตรวจสอบมากที่สุด เพื่อแยกแยะว่าเป็นผู้พูดคนใด

ระบบการจดจำอัตโนมัติ นอกจากใช้ในการจดจำคำพูดและผู้พูดแล้ว ยังอาจนำไปใช้ในงานด้านอื่นอีก โดยแยกสิ่งที่ไม่รู้ออกเป็นคลาส(class) เช่น ในระบบจดจำตัวเลข คลาสคือคำว่า {ศูนย์,หนึ่ง,สอง,สาม,...,เก้า} ระบบจะมีไลบรารี่ของรูปแบบอ้างอิงในแต่ละคลาส ขบวนการจดจำจะทำการเปรียบเทียบความคล้ายคลึงของสัญญาณไม่รู้กับคลาสแต่ละคลาส เพื่อหารูปแบบที่ใกล้เคียงที่สุด

การทำงานของการจดจำโดยทั่วไปมีอยู่ 2 ขั้นตอน ดังนี้

1.การเรียนรู้(หรืออาจจะเรียกว่าการฝึกก็ได้) เป็นการรวบรวมไลบรารี่ของรูปแบบอ้างอิง

2.การจดจำ เป็นการใช้ไลบรารี่ที่สมบูรณ์แบบเปรียบเทียบกับอินพุทที่ไม่รู้

ในการเรียนรู้ ข้อมูลสำหรับแต่ละคลาสจะถูกป้อนเข้าระบบ ระบบจะทำการสร้างรูปแบบอ้างอิงหรือต้นแบบของแต่ละคลาส โดยการเฉลี่ยรูปแบบจำนวนมากแล้วนำมารวมกันเป็นไลบรารี่โดยแยกเป็นคลาสต่างๆ ในการจดจำ ระบบจะทำการคำนวณรูปแบบของลักษณะเด่นของอินพุทที่ไม่รู้ และทำการตรวจสอบกับคลาสต่างๆ เพื่อหาคลาสที่มีรูปแบบอ้างอิงใกล้เคียงกับลักษณะเด่นที่ได้มากที่สุด

การออกแบบระบบการจดจำ มีปัญหาที่สำคัญอยู่ 2 ประการ คือ

1.การเลือกลักษณะเด่นและการคำนวณค่า

2.ทางเลือกของกฎการตัดสินใจ(decision rule)

ปัญหาแรกที่มีความสำคัญมากและยากมากกว่า เราจึงควรเริ่มต้นที่ปัญหาที่สองก่อน เพราะมีแนวความคิดหลายๆอย่างซึ่งจะช่วยในการอธิบายในหัวข้อการเลือกลักษณะเด่น

7.3 การจำเสียงพูด(Speech recognition)_

7.3.1 ประเภทของการจำเสียงพูด

การจำเสียงพูดโดยทั่วไปนิยมแบ่งออกเป็น 4 กลุ่ม โดยเรียงลำดับตามความยาก ดังต่อไปนี้

1.การจำแบบแยกคำ(Isolated-word recognition) เป็นการจำโดยต้องมีช่วงหยุดของแต่ละคำ

2.การจำเฉพาะคำ(Word spotting) ตรวจจับคำเฉพาะใดๆที่ปรากฏในประโยคนั้น

3.การจำแบบต่อเนื่อง(Connected speech recognition) จดจำคำพูดได้ทั้งประโยค โดยไม่ต้องมีช่วงหยุดระหว่างคำ

4.การเข้าใจเสียงพูด(Speech understanding) เป็นแบบสมบูรณ์ของกลุ่มที่3 โดยสร้างแหล่งข้อมูลของการออกเสียงและหลักของภาษานั้น จุดประสงค์หลักก็คือ การพูดที่ไม่ต้องถูกต้องมากนัก แต่ให้เข้าใจความหมายของประโยคนั้นได้ ในทางทฤษฎี ระบบเข้าใจเสียงพูด จะตัดทอนเอาเฉพาะความหมายของประโยค โดยไม่เข้มงวดเรื่องไวยากรณ์

7.3.2 การจำแบบแยกคำ(Isolated-word recognition)

การจำแบบแยกคำ เป็นจุดเริ่มต้นของการพัฒนาการจำเสียงพูด จนเป็นการจำเฉพาะคำและการจำแบบต่อเนื่องตามลำดับ ช่วงหยุดของแต่ละคำพูดทำให้การจดจำง่ายขึ้น เพราะสามารถแยกแยะจุดสิ้นสุดของแต่ละคำ(หมายรวมถึงจุดเริ่มต้นและจุดสิ้นสุดของคำนั้น) ทำให้ผลที่เกิดจากความไม่ชัดเจนของการพูดลดน้อยลงไปได้มาก การจำแบบแยกคำขณะที่ออกเสียงจะต้องระมัดระวังอย่างมาก เพราะต้องการช่วงหยุดระหว่างการพูดแต่ละคำ ทำให้การพูดขาดความไพเราะของภาษาไป ไม่เหมือนกับวิธีการอื่นๆ ซึ่งสามารถพูดเป็นธรรมชาติได้มากกว่าและใช้ความระมัดระวังในการพูดน้อยกว่า

การจำแบบแยกคำ มีหลักการการจดจำโดยใช้ลักษณะของสัญญาณเสียงพูด ดังต่อไปนี้

1. ขนาดหรือกำลังต่อเวลา

2. อัตราตัดผ่านศูนย์(zero-crossing rate)

3. สมดุลย์กลุ่มความถี่รวม(gross spectrum balance) เป็นการเปรียบเทียบพลังงานของกลุ่มความถี่สูงกับกลุ่มพลังงานความถี่ต่ำ

4. ความถี่สำคัญ อาจมาในรูปของ

ก. ความถี่จากการแปลง DFT

ข. F1,F2,F3

ค. พารามิเตอร์ LPCหรือสัมประสิทธิ์ PARCOR

ง. เอาท์พุทของฟิลเตอร์แบงค์(filter-bank) วิธีนี้เป็นวิธีที่นิยมกันที่สุดเพราะราคาถูกและความเร็วสูง

ขนาดนอกจากเป็นตัวบอกจุดสิ้นสุดของแต่ละคำ และยังเป็นตัวชี้ความแตกต่างของพยัญชนะและสระอีกด้วย

7.3.3 การตรวจจับจุดสิ้นสุด(Endpoint detection)

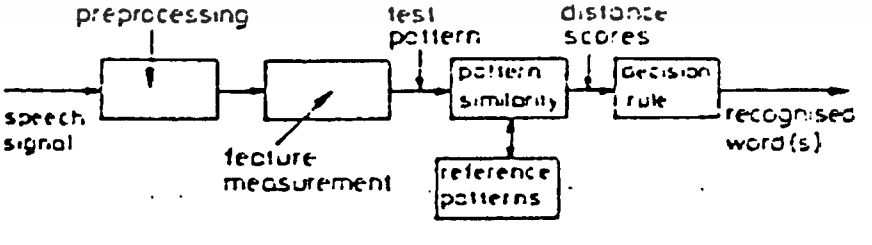
ปัญหาพื้นฐานของระบบจดจำเสียงพูด คือ เสียงที่เปล่งออกมาในขณะตรวจสอบนั้น อาจจะไม่เหมือนกันตอนที่เรียนรู้เสมอไป เสียงที่เปล่งออกมา 2 ครั้ง จะมีระยะเวลาที่ต่างกัน และระยะระหว่างการออกเสียงที่ไม่แน่นอน หมายความว่าคุณสมบัติที่ขึ้นกับเวลาจะทำให้การเปรียบเทียบไม่สามารถทำได้ เพราะคำอ้างอิงกับคำที่มาเปรียบเทียบเวลาจะต่างกัน ในกรณีเช่นนี้ ถึงแม้ว่าจะเปรียบเทียบกับคำที่ไม่รู้กับรูปแบบที่ถูกต้องของตัวมันเองแล้วก็ตาม แต่ก็อาจจะให้ผลแตกต่างได้พอๆกับที่เปรียบเทียบกับรูปแบบอ้างอิงของคำอื่น ซึ่งจะเป็นปัญหา ถ้าคำที่ถูกเปรียบเทียบเป็นการเปรียบเทียบทั้งคำต้นแบบ แทนที่จะเปรียบเทียบบางส่วนของคำโดยการเปรียบเทียบส่วนต่อส่วน ความจริงแม้ในระหว่างเสียงสองเสียงที่ใช้ในการเรียนรู้ รายละเอียดของเวลายังต่างกันเลย เมื่อเก็บข้อมูลเสียงจากหลายๆคน จึงต้องทำการเฉลี่ยในระหว่างการเรียนรู้ก่อนนำไปเปรียบเทียบ ในขั้นตอนการจดจำจะไม่มีความ

ในหลายๆกรณี ความแม่นยำของการปรับจะขึ้นอยู่กับความแม่นยำของการแสดงจุดสิ้นสุดของคำ ความผิดพลาดในการจดจำจุดเริ่มต้น และจุดสิ้นสุดของคำจะทำให้เกิดความยุ่งยากขึ้น โดยความผิดพลาดของจุดสิ้นสุดจะเกิดขึ้น ในคำที่เริ่มต้นและลงท้ายด้วยพยางค์ที่ออกเสียงเบา ผู้พูดบางคนอาจจบประโยคด้วยเสียงสั้น พยางค์เสียงสั้นนี้จะทำให้เกิดความผิดพลาดในการเปรียบเทียบคำ

ในห้องทดลอง การจดจำจุดสิ้นสุดอาจจะทำได้ง่าย เพราะข้อมูลเสียงพูดถูกรวบรวมภายใต้การควบคุมสภาวะ เพื่อหาจุดสิ้นสุด ยิ่งกว่านั้น ผู้พูดที่บันทึกข้อมูลในห้องทดลอง จะทำด้วยความระมัดระวัง แต่นอกห้องทดลอง ไม่ได้เป็นอย่างนั้น เสียงพูดจะมีการรบกวนจากกรอบข้างเพราะไม่มีการควบคุมสภาวะ และผู้พูดไม่มีแรงจูงใจที่จะช่วยในการจดจำ

คุณสมบัติหลักของจุดสิ้นสุด คือพลังงานดังนั้นวิธีง่ายๆที่ใช้ในการตรวจจับจุดสิ้นสุด ก็คือ การเปรียบเทียบพลังงานของสัญญาณกับค่าหรือระดับที่ตั้งไว้ โดยตรวจสอบจุดเริ่มต้นของคำจากตำแหน่งของคำที่มีพลังงานเริ่มมากกว่าระดับที่ตั้งไว้ และจุดสิ้นสุดจากตำแหน่งของคำที่มีพลังงานต่ำกว่าระดับที่ตั้งไว้ ค่าระดับที่ตั้งไว้นี้ต้องมีความเข้มระดับหนึ่ง ถ้าความแรงของสัญญาณเสียงพูดอ่อน อาจจะต้องตั้งระดับต่ำๆ

7.4 ระบบจดจำเสียงพูดแบบแยกคำและไม่ขึ้นกับผู้พูด ด้วยเทคนิค DTW



รูปที่7.1 *แสดงแบบจำลองของการจดจำรูปแบบในการจดจำเสียงพูด*

รูปที่7.1 ได้แสดงแบบจำลองของการจดจำรูปแบบของสัญญาณเสียงพูด ซึ่งมีลักษณะการทำงาน 2 อย่าง คือ การฝึก และการตรวจสอบ การฝึกเป็นการสอนศัพท์ให้กับระบบจดจำเสียงพูด โดยใช้ค่าเฉลี่ยของแต่ละคำ เป็นตัวแทนหรือต้นแบบอ้างอิง(reference template) ของคำนั้น การเฉลี่ยคำคำเดียวนั้นหลายครั้ง หลังจากการปรับแนวเวลา(time alignment) เป็นวิธีการอย่างหนึ่งเพื่อให้ได้ต้นแบบอ้างอิงตามที่ต้องการ

ในการตรวจสอบหรือการจดจำ เป็นการเปรียบเทียบคำที่ไม่รู้กับต้นแบบอ้างอิงที่เก็บไว้แล้ว เมื่อเปล่งเสียงออกมา คำพูดนั้นจะถูกบันทึกโดยผ่านตัวกรองความถี่ต่ำผ่าน(bandpass filter) ลดทอนสัญญาณความถี่ต่ำและจำกัดช่วงความถี่ เพื่อป้องกันความผิดเพี้ยนของสัญญาณจากการสุ่มเชิงเลข(digitize) หลังจากนั้น คำที่เปล่งออกมาจะถูกแยกช่วงสัญญาณที่ไม่มีเสียงพูด(silence)ออกไป และมีการตรวจสอบจุดสิ้นสุดของแต่ละคำ เพื่อสะดวกต่อการประมวลผลต่อไป แทนที่จะเก็บคำที่พูดออกมาทั้งหมด พารามิเตอร์เสียงพูด(speech parameter) ที่เป็นลักษณะเด่นของคำนั้นจะถูกแยกออกมา ในขั้นตอนนี้ต้องทำการลดขนาดของพารามิเตอร์ที่แยกได้ เพื่อทำการเปรียบเทียบกับรูปแบบอ้างอิง(reference pattern) หรือรูปแบบที่ใกล้เคียงกับรูปแบบของคำที่ตรวจสอบมากที่สุด วิธี NN (nearest neighbour) และKNN(K nearest neighbour) เป็นกฎการตัดสินใจ(decision rule)ที่นิยมกันมากที่สุด

โดยทั่วไป ผู้พูดไม่สามารถพูดซ้ำคำเดิมให้เหมือนเดิมทุกอย่างได้ ผลที่ได้คือช่วงของเสียงพูดที่เกิดขึ้นในเวลาใดๆ Tr อาจเกิดขึ้นที่เวลาต่างกับเวลา Tt ของแบบตรวจสอบ(test pattern) ทำให้เกิดปัญหาในการเปรียบเทียบรูปแบบ วิธีการหนึ่งที่จะเอาชนะปัญหายุ่งยากนี้ได้ ก็คือการเปลี่ยนอัตราส่วนแกนเวลาของรูปแบบตรวจสอบแบบไม่เชิงเส้น เพื่อให้ตรงกับรูปแบบอ้างอิงให้มากที่สุด วิธีนี้เรียกว่า DTW(dynamic time warping) ซึ่งเป็นวิธีที่รู้จักกันมากที่สุด

บทที่ 8

วงจรและโปรแกรมควบคุมการทำงาน

8.1อธิบายการทำงานของวงจรแต่ละส่วน

8.1.1 ส่วนดีโค้ดพอร์ท (DECODE PORT)

ในส่วนนี้ใช้ 74138 3 line to 8 line decoder เป็นตัว decode port โดยที่ port 300-303 decode ไปที่ 8255 และ port 304 decode ไปยัง A/D รายละเอียดแสดงดังตาราง

เลขพอร์ท	หน้าที่การทำงาน
300h	D/A converter (port A 8255)
301h	control signal (port B 8255)
302h	not used (port C 8255)
303h	control port 8255
304h	A/D converter

8.1.2 พอร์ทขนาน (PARALLEL PORT) 8255

ใช้เป็นตัวส่งผ่านข้อมูล โดยโหมดการทำงานเป็นโหมด 0 โดยมีการทำงานของแต่ละport ดังนี้

-พอร์ท A เป็น เอาท์พุท ที่ส่งข้อมูลไปยัง ตัวเปลี่ยนสัญญาณดิจิตอลเป็นอนาลอก

-พอร์ท B เป็น เอาท์พุท ที่ใช้ส่งสัญญาณควบคุมการทำงานของวงจรในการ์ด

-พอร์ท C ไม่ได้ใช้งาน

8.1.3 เฟสล็อกลูป 4046(CMOS 4046 phase lock loop)

4046 จะทำหน้าที่เป็น VCO(voltage control oscillator) สร้างสัญญาณสุ่ม โดยสามารถปรับอัตราการสุ่ม(sampling rate) ที่ VR2 ได้ตั้งแต่ 10k-100kHz

8.1.4 ส่วนแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล (A/D)

A/D converter ที่ใช้เป็นเบอร์ adc0820ซึ่งมีคุณสมบัติดังนี้

-ดิจิตอลเอาท์พุท 8bit หรือ มีความละเอียด 256 ระดับ ดังนั้นการผิดพลาดควันไตส์ซิ่ง(quantizing error หรือความผิดพลาดในการเปลี่ยนค่าจากสัญญาณอนาลอก เป็นสัญญาณดิจิตอล จึงอยู่ระหว่าง +0.5/256 และ -0.5 /256

-ช่วงเวลาในการเปลี่ยน(convertion-time)หรือเวลาที่ใช้ในการแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล ประมาณ1.5 μsec. ซึ่งเป็นความเร็วที่สูง เนื่องการทำงานของ adc0820 ที่เป็นแบบ half_flashโดยแบ่งเป็นflash A/D 2 ชุด ชุดละ 4 บิท แล้วนำเอาท์จาก flash A/D แต่ละชุดมาเข้า sucessive approximation A/D อีกครั้งหนึ่ง

ในวงจรนี้ adc0820 จะทำงานในโหมด write ซึ่งมีลักษณะการทำงานดังนี้

-เมื่อมีสัญญาณนาฬิกา(clock) จาก4046มาเข้าที่ขาwrite ของ adc0820 สัญญาณอนาลอกที่ขา V_{in} จะถูกแปลงเป็นสัญญาณดิจิตอล เมื่อแปลงสัญญาณเสร็จแล้ว จะเก็บไว้ที่บัฟเฟอร์ที่เอาท์พุทและมีสัญญาณอินเทอรัพท์ออกมาที่ขา int ซึ่งสัญญาณอินเทอร์รัพท์นี้จะไปอินเทอรัพท์คอมพิวเตอร์ให้มารับข้อมูล โดยการส่งสัญญาณ read เมื่อ adc0820 ได้รับสัญญาณ read ที่ขา-read แล้ว ข้อมูลที่บัฟเฟอร์จะออกมาที่บัสข้อมูล คอมพิวเตอร์ก็จะรับข้อมูลไปเก็บไว้

8.1.5 ส่วนกรองความถี่สัญญาณ

เป็นส่วนที่ใช้ขจัดสัญญาณที่ไม่ต้องการออกไป ในวงจรนี้ใช้ เป็นตัวกรองสัญญาณที่ไม่ได้อยู่ในช่วง 300 - 3 กิโลเฮร์ตออกไป นั่นคือสัญญาณที่เราต้องการและนำไปใช้จะอยู่ในช่วงนี้เท่านั้น ซึ่งในช่วงดังกล่าวเป็นช่วงความถี่ของเสียงคนโดยทั่วไปที่เราจะต้องนำมาใช้งานต่อไป ในส่วนของการคำนวน จะกล่าวในหัวข้อต่อไป

8.1.6 ส่วนแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาลอก (D/A)

ใช้ D/A เบอร์ dac0808-8bit D/A Converter ทำหน้าที่แปลงสัญญาณข้อมูลดิจิตอลที่ส่งมาจาก parallel port 8255 ให้เป็นสัญญาณอนาลอก เพื่อส่งต่อไปยัง output ซึ่งเป็นลำโพงแปลงสัญญาณเป็นเสียงอีกต่อหนึ่ง

dac0808 มีคุณสมบัติดังนี้

-ความละเอียด = 1/256

-maximum output error ไม่เกิน 1/2 เท่าของความละเอียด

-setting time หรือเวลาที่ใช้ในการเปลี่ยนสัญญาณ เมื่อเปลี่ยนสัญญาณจะเกิดการออสซิสเลตในช่วง 1/2 เท่าของความละเอียดก่อนจะถึงค่าที่ถูกต้อง

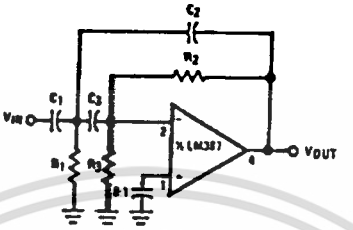
8.1.7 ส่วนวงจรลิมิตเตอร์และวงจรยกระดับสัญญาณ

เนื่องจากโวลท์เตจที่เข้า A/D จะต้องไม่เกินของ Vref(+) และ Vref(-) เพราะจะทำให้วงจรภายในของตัว adc0820 เสียหายได้ ดังนั้นจึงต้องมีส่วน จำกัดโวลท์เตจ เพื่อป้องกันการเสียหายของ adc0820 เมื่อสัญญาณเสียงมีขนาดใหญ่เกินค่าของ Vref(+) และ Vref(-) ในกรณีนี้ ตั้ง Vref(+) เท่ากับ 5V และ Vref(-)=0V

เนื่องจากตั้งระดับของ Vref ไว้ระหว่าง 0-5V จึงจำเป็นต้องยกระดับสัญญาณเสียงก่อนเข้า A/D ขึ้น2.5 volt(จากเต็มสเกล 5 volt) เพื่อให้รับสัญญาณได้เต็มสเกลที่ตั้งไว้

8.2 การออกแบบและการคำนวน ส่วนกรองความถี่สัญญาณ

8.2.1 ส่วนกรองความถี่สูงผ่าน (HIGHPASS FILTER)



รูปที่ 8.1 วงจรกรองความถี่สูงผ่าน

เมื่อ $w_c = 2\pi f_c$ ซึ่งเราจะหาค่า w_0 ได้โดยใช้สมการ (8.2.1) และคำนวนค่า Q โดยใช้สมการ (8.2.2) หรือใช้ตาราง (8.2.1)

เมื่อ $C_1 = C_3$

$$w_c = w_0 / \beta \qquad (8.2.1)$$

$$\beta = \sqrt{(1-(1/2Q^2))+\sqrt{(1-(1/2Q))^2+1}} \qquad (8.2.2)$$

Q	ω_c Low-Pass	ω_c High-Pass
0.707*	$1.000\omega_0$	$1.000\omega_0$
1	$1.272\omega_0$	$0.786\omega_0$
2	$1.498\omega_0$	$0.668\omega_0$
3	$1.523\omega_0$	$0.657\omega_0$
4	$1.537\omega_0$	$0.651\omega_0$
5	$1.543\omega_0$	$0.648\omega_0$
10	$1.551\omega_0$	$0.645\omega_0$
100	$1.554\omega_0$	$0.644\omega_0$

* Butterworth

ตาราง 8.2.1 แสดงค่า w_c กับ Q

เมื่อ $C_1 = \frac{Q}{W_0 R_2}$ (8.2.3)

$C_2 = \frac{C_1}{A_0}$ (8.2.4)

$R_1 = \frac{1}{QW_0C_1(2A_0+1)}$ (8.2.5)

ต้องการ กรองความถี่ที่สูงกว่า 300 เฮิร์ตผ่าน โดยเลือกใช้ค่า Q = 0.707 และ อัตราขยาย (A_0) = 1 โดยดูจากตาราง (8.2.1) ซึ่งจะได้ ค่า

$W_0 = W_c$

เลือกค่า R_3 = 240K

$R_2=((V_s/2.6)-1)R_3 = ((24/2.6)-1)240K$

$= 1.98*10^6$ กิโลโอห์ม

เลือกค่า R_2 เท่ากับ 2 เมกกะโอห์ม

เมื่อให้ค่า $C_1 = C_3$ ดังนั้นจากสมการ (8.2.3)จะได้

$C_1 = \frac{(0.707)(2+1)}{(2\pi)(300)(2*10^6)}$

$= 5.62 *10^{-10}$ F

ดังนั้นเลือกใช้ค่า C_1 = 560 pF

$C_1 = C_3$ =560 pF

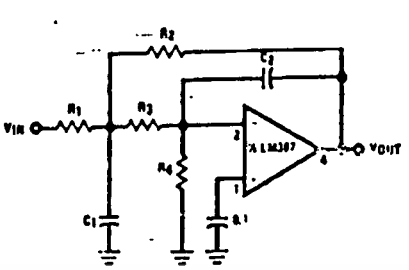
จากสมการ (8.5.6)

$R_1 = \frac{1}{(0.707)(2\pi)(300)(5.6 *10^{-10})}$

= 446 กิโลโอห์ม

เลือกค่า R_1 = 430 กิโลโอห์ม

8.2.2 ส่วนกรองความถี่ต่ำผ่าน(LOWPASS FILTER)



รูปที่ 8.2 วงจรกรองความถี่ต่ำผ่าน

F_c = 3 KHz $W_c = 2\pi f_c$ ใช้ค่า Q = 0.707 จะได้ $W_0 = W_c$ ที่ A_0 = 1

$$K = \frac{1}{4Q^2(A_0+2)} \qquad (8.2.7)$$

$$C_2 = KC_1 \qquad (8.2.8)$$

$$R_2 = \frac{1}{2QW_0C_1K} \qquad (8.2.9)$$

$$R_3 = \frac{R_2}{A_0+1} \qquad (8.2.10)$$

$$R_1 = \frac{R_2}{A_0} \qquad (8.2.11)$$

$$R_4 = \frac{R_2+R_3}{((V_S/2.6)-1)} \qquad (8.2.12)$$

จากค่าที่กำหนดให้สามารถหาค่า R C ได้ดังต่อไปนี้

$$K = \frac{1}{4(0.707)^2(1+1)}$$

$$= 0.25$$

เลือกค่า C_1 = 560 pF

ดังนั้น จะได้ $C_2 = KC_1 = 0.25(560\ pF)$

$$= 140\ pF$$

เลือกใช้ค่า C_2 เท่ากับ 150 pF

$$R_2 = \frac{1}{(2)(0.707)(2\pi)(3KHz)(560\ pF)(0.25)}$$

$$= 267.99 \text{ กิโลโอห์ม}$$

เลือกใช้ค่า R_2 เท่ากับ 270 กิโลโอห์ม

$$R_3 = \frac{R_2}{A_0+1} = \frac{270K}{(1+1)}$$

$$= 135 \text{ กิโลโอห์ม}$$

เลือกใช้ค่า R_3 เท่ากับ 130 กิโลโอห์ม

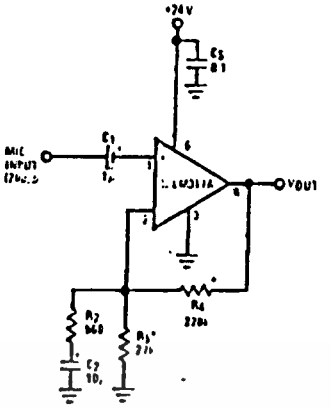
$$R_1 = R_2 / 1 = R_2 = 270 \text{ กิโลโอห์ม}$$

$$R_4 = \frac{270K + 130K}{((24/2.6)-1)} = 48.59\ K$$

เลือกใช้ค่า R_4 เท่ากับ 47 กิโลโอห์ม

ดังนั้นเมื่อนำทั้งสองวงจรมารวมกันจะได้วงจรกรองสัญญาณที่ยอมให้ช่วงความถี่ตั้งแต่ 300 เฮิร์ต จนถึง 3 กิโลเฮิร์ตผ่านไปได้

8.2.3 การคำนวนค่า R,C ค่าต่างๆใน ปรีแอมป์ (pre- amp)



รูปที่ 8.3 แสดงวงจร ปรีแอมป์

$R_4 = ((V_s/2.6)-1)R_s$

R_5 = มีค่าสูงสุดได้ 240 กิโลโอห์ม

โวลท์เตจเกน ของวงจร $(A_{vac}) = 1+ (R_4/R_5)$ เมื่อ $R_5 >> R_6$

$$C_2 = \frac{1}{2\pi f_0 R_6}$$

$$C_c = \frac{1}{2\pi f R_L}$$

เมื่อ f_0 คือ ค่าความถี่คัตออฟที่ความถี่ต่ำ

8.3 โปรแกรมควบคุมการทำงานของการ์ด

โปรแกรมมีโหมดการทำงานดังนี้

8.3.1 direct out mode

การทำงานของโหมดจะทำการรับข้อมูลจาก A/D port เมื่อได้รับการอินเทอรัพท์จาก ADC0820 และส่งผ่านไปยัง D/A portโดยตรง โดยไม่มีการเก็บข้อมูลไว้ในบัฟเฟอร์

8.3.2 record mode

การทำงานในโหมดนี้ โปรแกรมจะทำการบันทึกข้อมูลจาก A/D port ลงในบัฟเฟอร์ ซึ่งจองเนื้อที่หน่วยความจำของคอมพิวเตอร์ไว้เท่ากับที่ตั้งไว้ โดยโปร

แกรมจะออกจากโหมดการทำงานเอง เมื่อบันทึกข้อมูลครบตามที่ได้ตั้งไว้

ระยะเวลาการบันทึกข้อมูลขึ้นอยู่กับอัตราการสุ่มที่ตั้งไว้ที่VR2 โดยมีสูตรการคำนวณดังนี้

ระยะเวลาในการบันทึก = ขนาดของบัฟเฟอร์/อัตราการสุ่ม

8.3.3 save mode

ทำหน้าที่เก็บข้อมูลในบัฟเฟอร์ลงในดิสเกตต์ โดยทำการเก็บครั้งละ 64 kbyte ต่อหนึ่งไฟล์

8.3.4 load mode

ทำหน้าที่โหลดข้อมูลจากไฟล์ที่ต้องการและนำไปเก็บไว้ในบัฟเฟอร์ 64 kbyte ซึ่งเป็นที่เดียวกับในโหมด record

8.3.5 FFT(fast fourier transform)

จะนำข้อมูลจำนวนจริงที่เก็บไว้ในบัฟเฟอร์ มาเข้าขบวนการของ FFT ซึ่งจะให้ข้อมูลออกมาในรูปคอมเพล็กซ์(complex) ซึ่งจะมีทั้งจำนวนจริงและจำนวนจินตภาพ โดยเก็บจำนวนจริงไว้ใน rl_array และเก็บจำนวนจินตภาพไว้ใน im_array ในรูปแบบของ floating point ดังนั้น array แต่ละตัวจะใช้เนื้อที่หน่วยความจำเท่ากับสี่เท่าของขนาดบัฟเฟอร์ใน time domain เนื่องจากหน่วยความจำของคอมพิวเตอร์ไม่เพียงพอที่จะแปลงข้อมูลใน time domain จึงต้องใช้วินโดว์แปลงข้อมูลเป็นช่วง อาจจะเป็น 256 จุด หรือ 512 จุด

8.3.6 plot graph in time domain mode

การทำงานในโหมดกราฟฟิกของโปรแกรมนี้ จะทำในโหมดความละเอียดสูงสุดของจอภาพแต่ละชนิด

การทำงานในโหมดนี้ทำหน้าที่พล็อตกราฟของสัญญาณอินพุทใน time domain ซึ่งก็คือข้อมูลที่เก็บไว้ในบัฟเฟอร์ใน record mode หรือ load mode โดยสามารถกำหนดจุดเริ่มต้น จำนวนจุด และสเกลของขนาดในแกน y ได้

8.3.7 plot magnitude in frequency domain

นำข้อมูลที่ผ่านขบวนการ FFT มาหาขนาด ซึ่งหาได้จาก

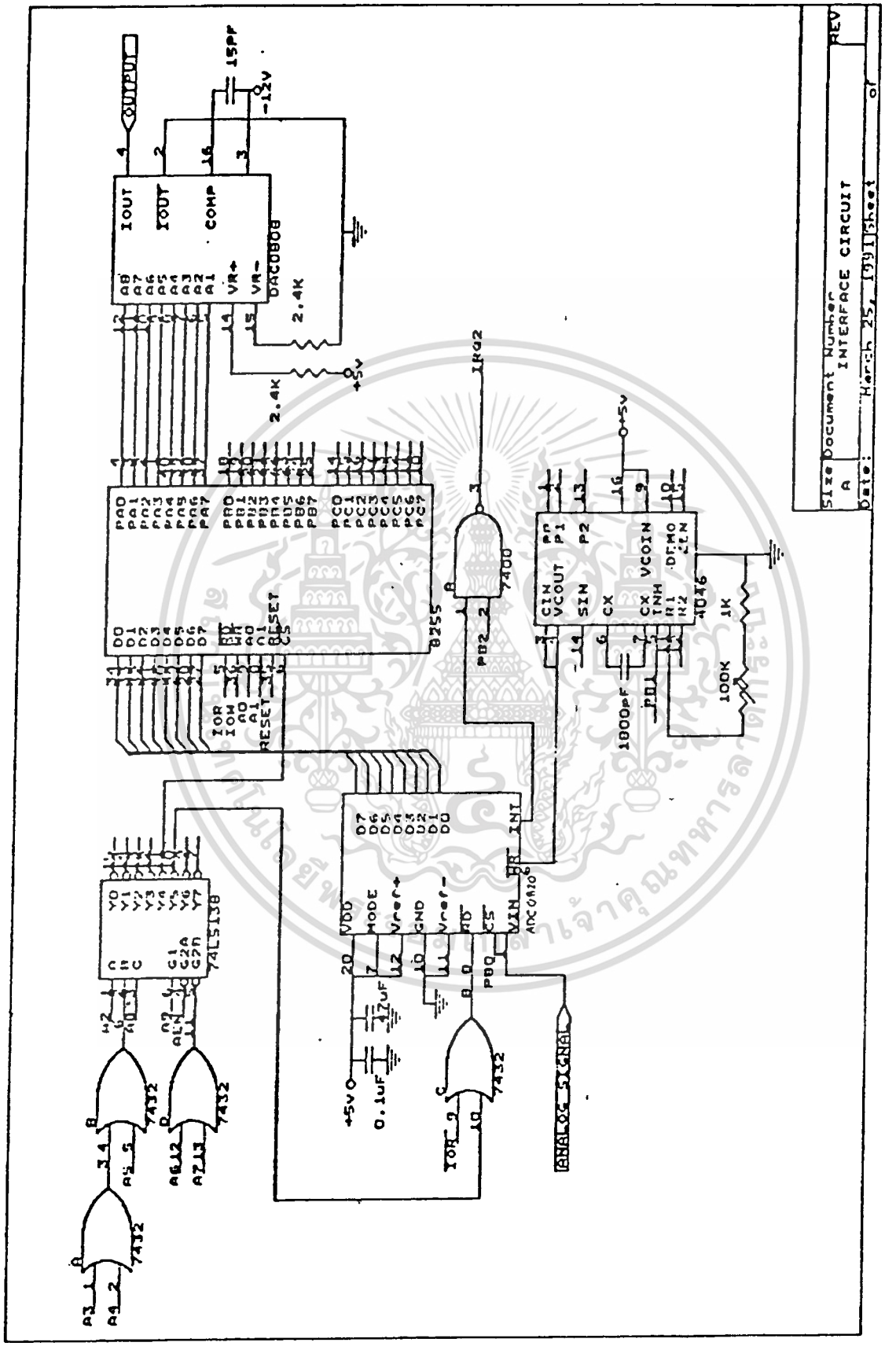
$$\text{magnitude} = \sqrt{(\text{real part})^2 + (\text{imaginary part})^2}$$

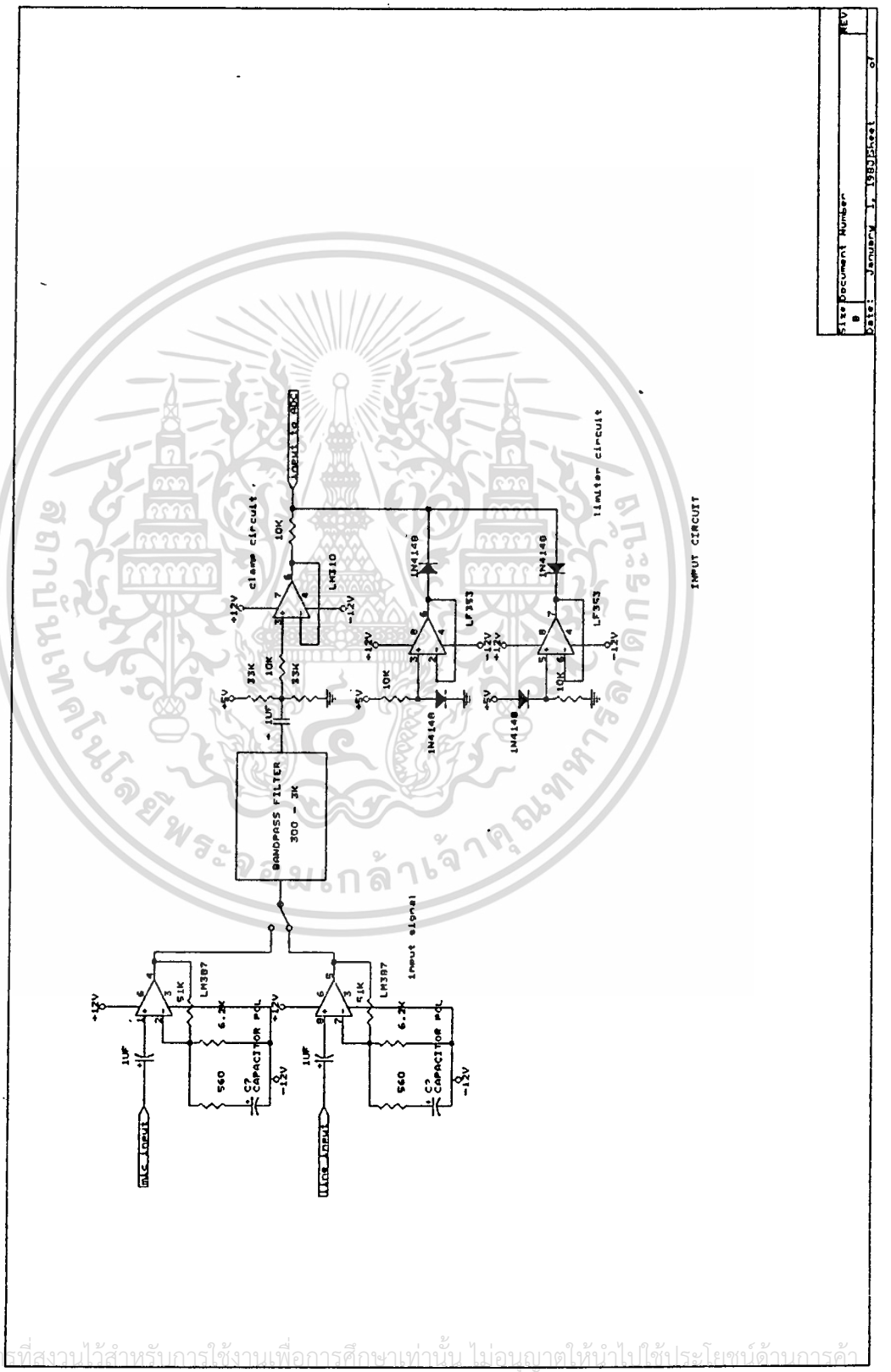
8.3.8 plot phase in frequency domain

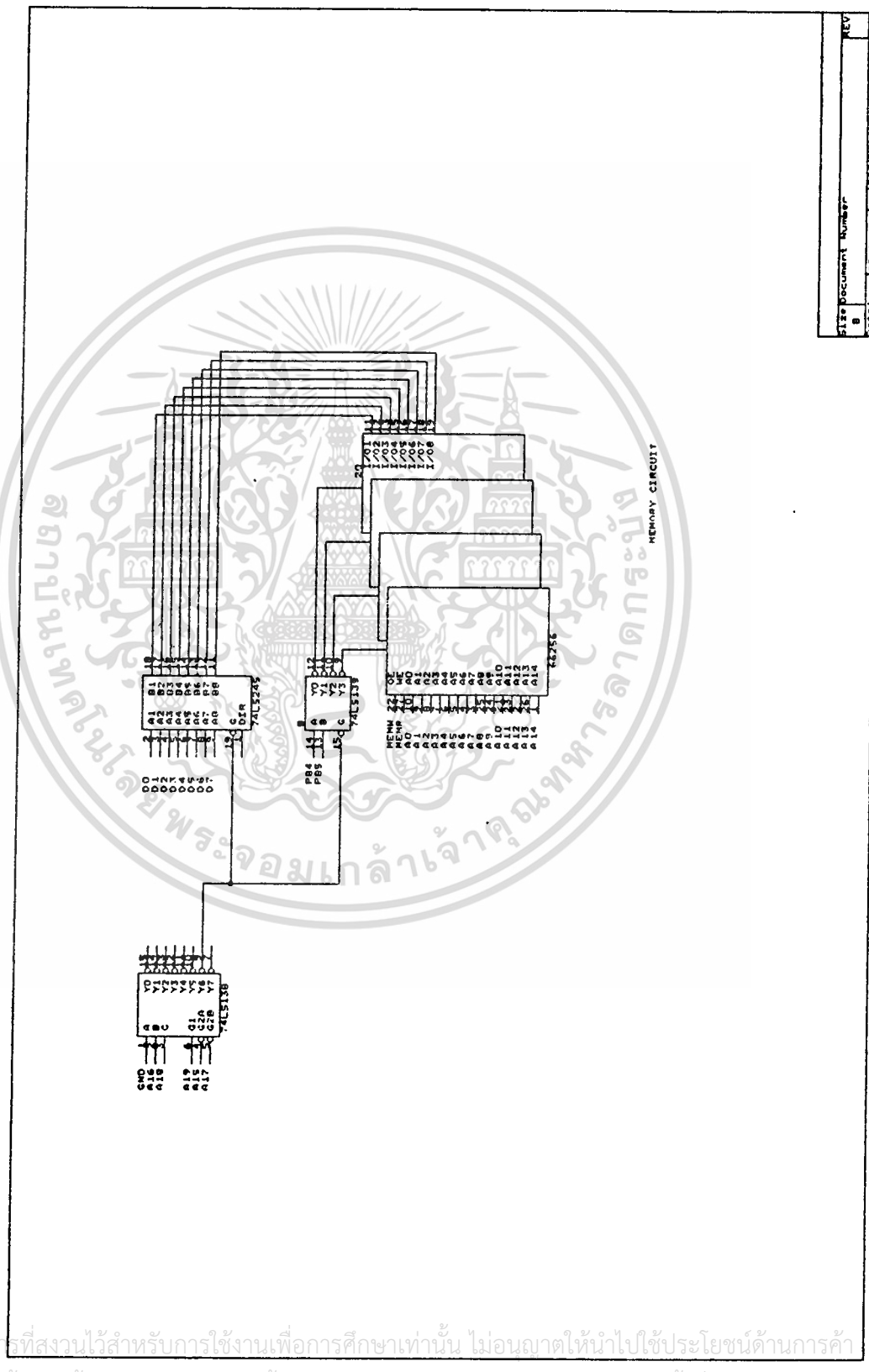
นำข้อมูลที่ผ่านขบวนการ FFT มาหาเฟส ซึ่งหาได้จาก

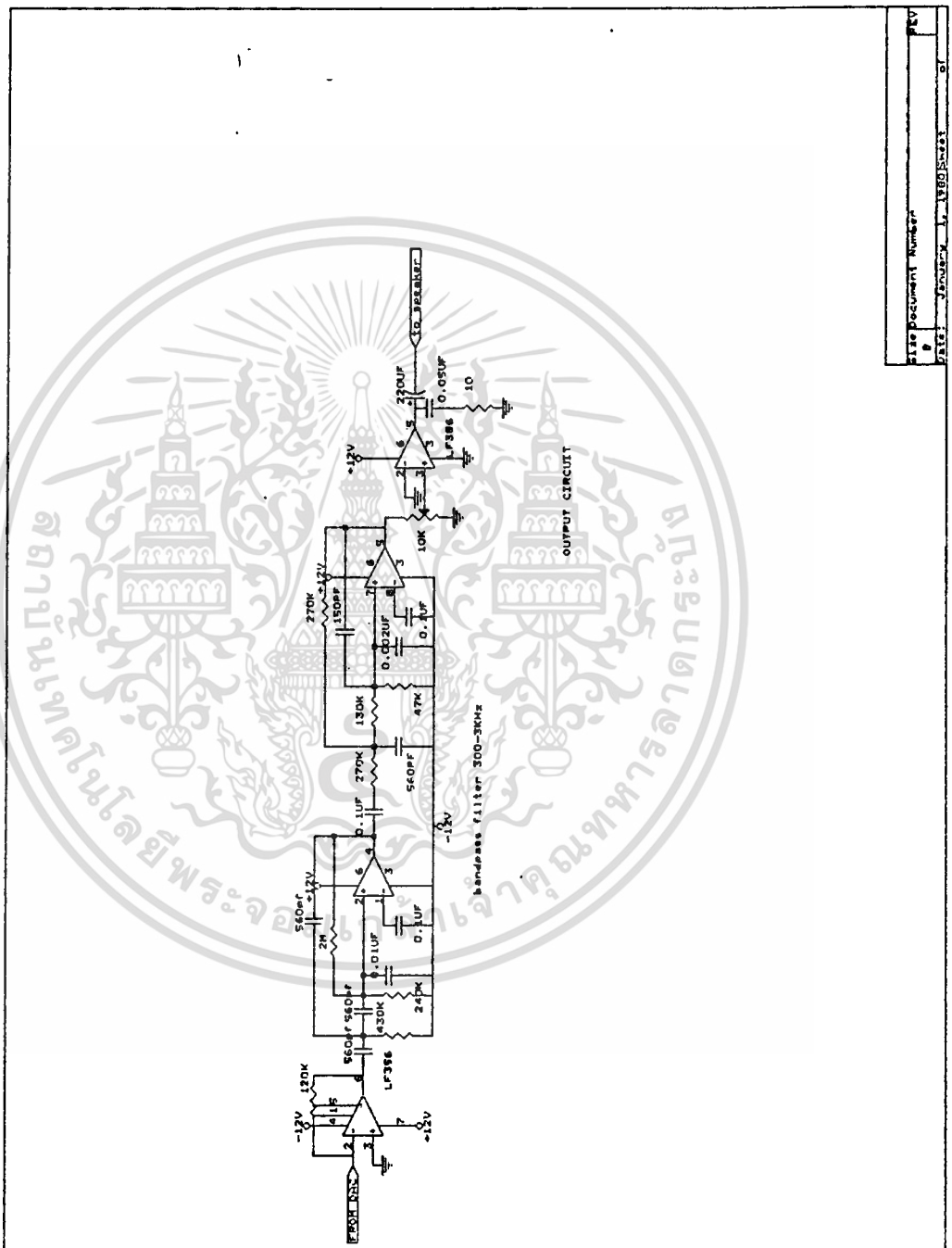
$$\text{phase} = \tan^{-1}(\text{imaginary part}/\text{real part})$$

โดยมีค่าอยู่ระหว่าง π และ $-\pi$









บทที่ 9

การทดลองและผลการทดลอง

9.1 การทดลองและผลการทดลองในภาคเรียนที่1

ในภาคเรียนที่1 ผู้จัดทำได้ทดลองทำการ์ดที่ใช้อินเตอร์เฟสกับเครื่องไมโครคอมพิวเตอร์ ตระกูล 8086 โดยต่อลงบนการ์ดต้นแบบ(prototype card) แต่ต้องประสบกับปัญหาใหญ่อยู่สองประการ คือ

1.เครื่องที่ทดลองมีความเร็วของการทำงานต่ำ ทำให้คอมพิวเตอร์รับอินเตอร์รัพท์ไม่ทัน

2.ปัญหาที่เกิดจากการจองหน่วยความจำของโปรแกรมภาษา C (ผู้จัดทำใช้ TurboC 2.0) ซึ่งมีความยุ่งยากมาก

งานในภาคเรียนที่1 จึงยังเก็บข้อมูลไม่ได้ ทำให้ไม่สามารถวิเคราะห์สัญญาณเสียงพูดได้ ผู้จัดทำจึงทดลองจำลองสัญญาณขึ้นมาใช้เอง เพื่อสังเกต ผลลัพธ์ที่ได้หลังผ่าน FFTและinverse FFT โดยใช้สัญญาณซายน์,โคซายน์ สัญญาณคลื่นสี่เหลี่ยม และสัญญาณสามเหลี่ยม โดยแสดงผลที่ได้บนจอภาพคอมพิวเตอร์

จากการทดลองพบว่า

1.FFT สามารถบอกความถี่ของสัญญาณอินพุทได้อย่างถูกต้อง เมื่อจำนวน N ที่ใช้ใน FFT เป็นจำนวนเท่าของจำนวนจุดสุ่มในหนึ่งคาบเวลา(ดูรายละเอียด ได้จากทฤษฎี) แต่เมื่อไม่เป็นจำนวนเท่า ผลลัพธ์ของ FFT ที่ได้จะปรากฏลอนข้าง (side lobe)

2.เมื่อทำการต่อเติมศูนย์(Appending Zero) ให้กับข้อมูลที่จะทำการแปลง เพื่อให้ได้จำนวนเท่ากับ 2^N ก่อนเข้า FFT พบว่าผลลัพธ์ที่ได้จะให้ความละเอียดของความถี่มากขึ้นทำให้เห็นของรูปคลื่น หลังผ่าน inverse FFT ได้ชัดเจนขึ้น

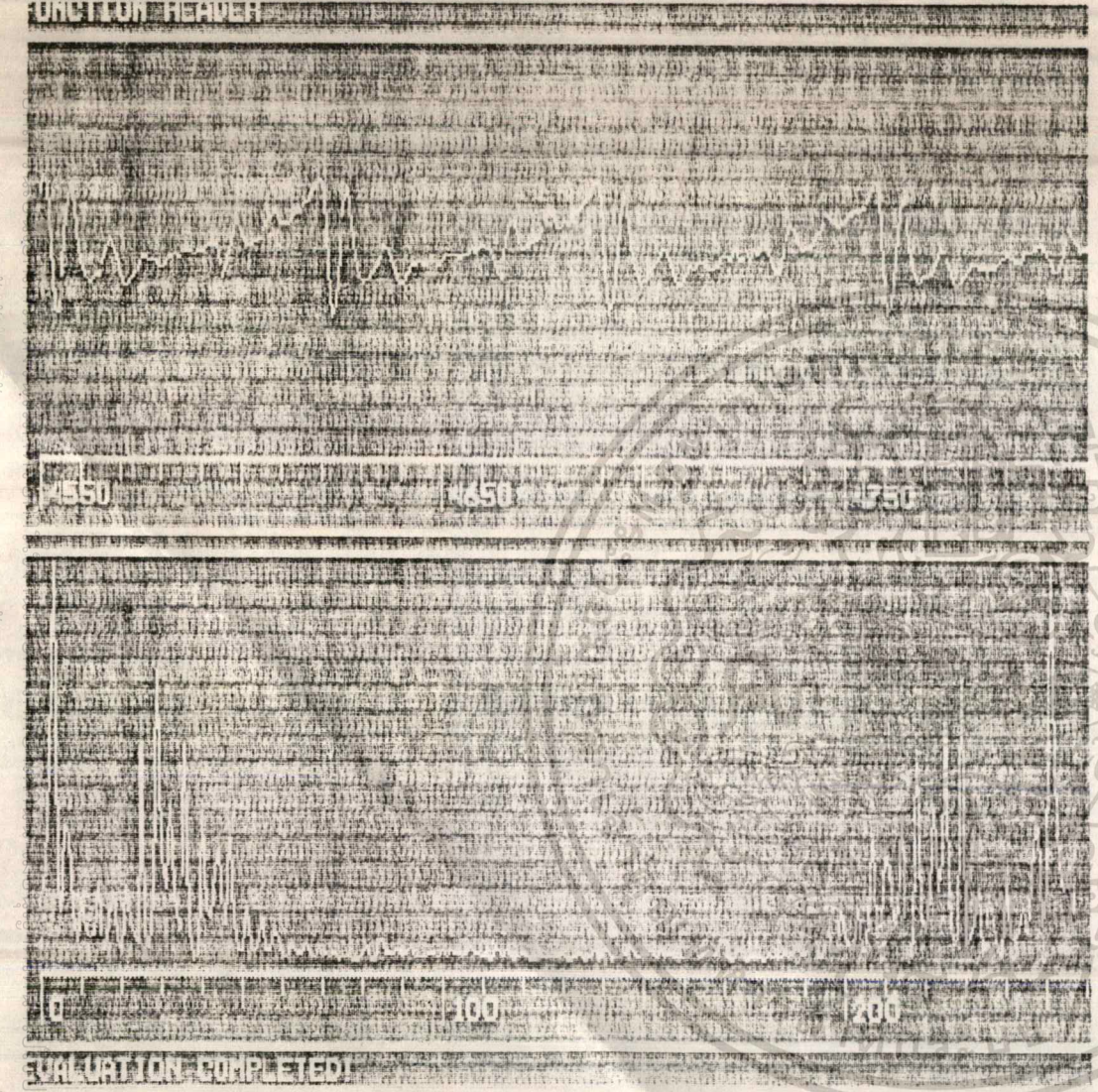
9.2 การทดลองและผลการทดลองในภาคเรียนที่2

ผู้จัดทำจิงทดลองการ์ดเดิมอีกครั้งโดยใช้โปรแกรมที่มีการตอบสนองการอินเตอรัพท์ที่เร็วขึ้น และพบว่าการ์ดสามารถทำงานและเก็บข้อมูลของเสียงพูดได้ แต่ยังมีสัญญาณรบกวนปนมากับสัญญาณเสียงจริง เนื่องจากการเดินสาย(wire wrap) ผู้จัดทำจิงเปลี่ยนเป็นใช้ปริ๊นท์แทนการเดินสาย เพื่อลดสัญญาณรบกวนที่เกิดขึ้น ซึ่งก็สามารถลดสัญญาณรบกวนไปได้มาก

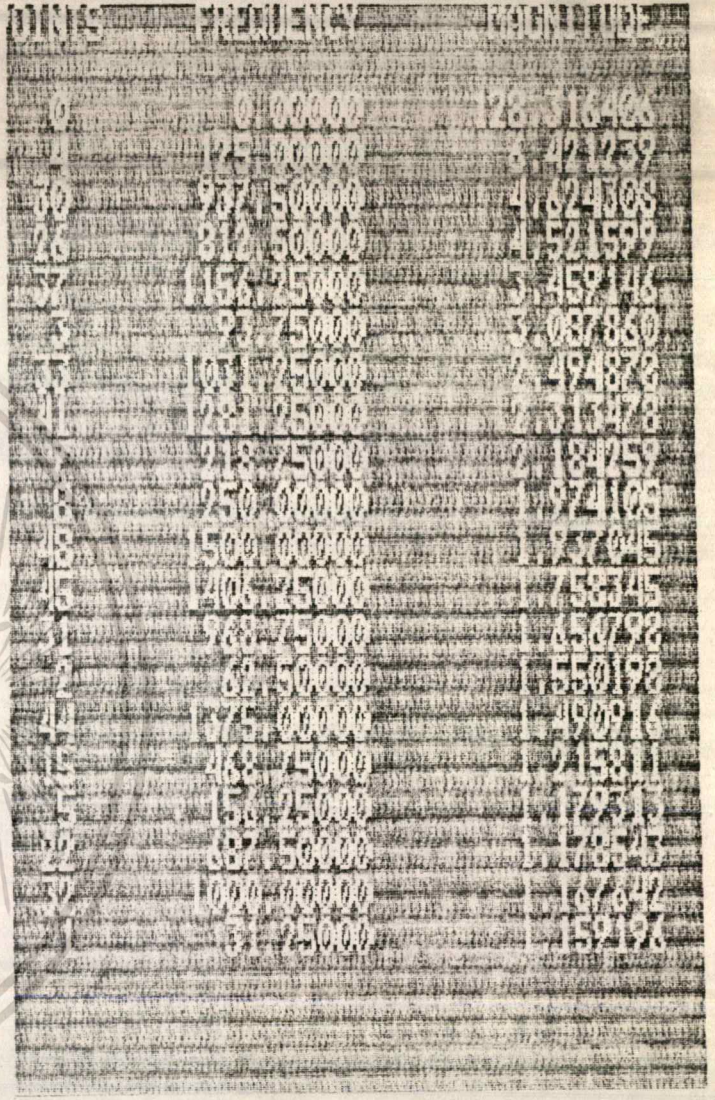
ผู้จัดทำได้ทดลองเก็บข้อมูลของเสียงพูดคำว่า (หนึ่ง,สอง,สาม,..,เก้า,ศูนย์) และนำมาผ่าน FFT เพื่อหาความถี่ของสัญญาณเสียงอินพุท ผลการทดลองที่ได้แสดงไว้ในรูปที่9.1-9.3

ในการหาช่วงของข้อมูลที่จะเข้า FFT ผู้จัดทำได้ทำโปรแกรมเลื่อนข้อมูลบนจอภาพคอมพิวเตอร์ เพื่อหาช่วงที่เป็นคาบเวลามากที่สุด แล้วจิงเข้า FFT นำผลลัพธ์ที่ได้มาหาจุดที่มีแอมพลิจูดสูงสุด เรียงลำดับไว้ 20 ความถี่ ซึ่งแสดงผลไว้ในรูปที่9.2

จากผลการทดลอง พบว่าที่จุดสุ่มเท่ากับ 0 มีค่าประมาณ 128 ซึ่งเป็นระดับ D.C. ของสัญญาณอินพุท ซึ่งต้องยกระดับสัญญาณขึ้นมา 2.5 V (ครึ่งหนึ่งของเรนจ์ 0-5V) เพื่อเข้า ADC



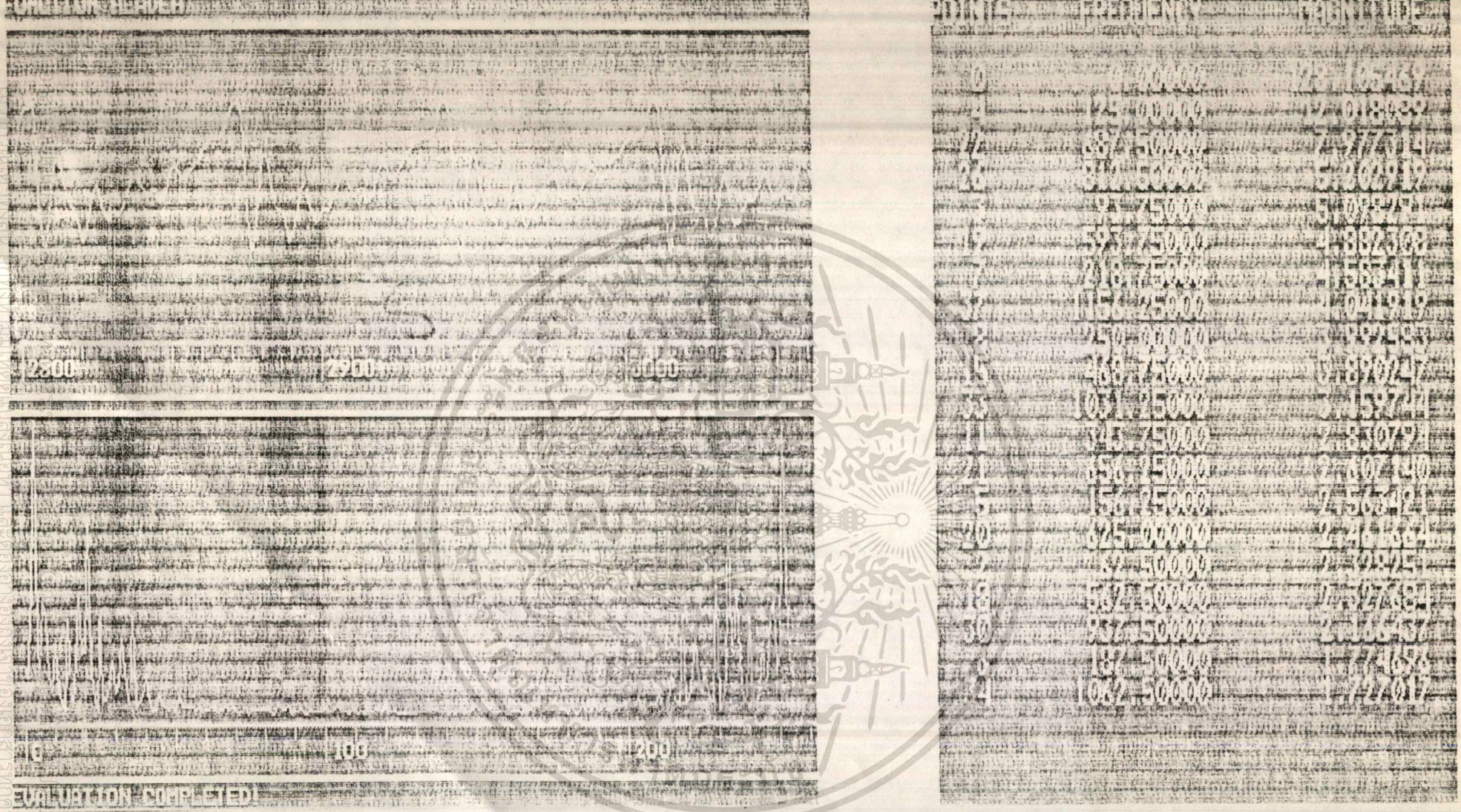
(a)



(b)

รูปที่9.3(a) *แสดงข้อมูลเสียงของคำว่า "สาม" ในโดเมนเวลา และในโดเมนความถี่หลังผ่าน FFT แล้ว โดยใช้วินโดว์สี่เหลี่ยมที่ขนาดของข้อมูล เท่ากับ 256*

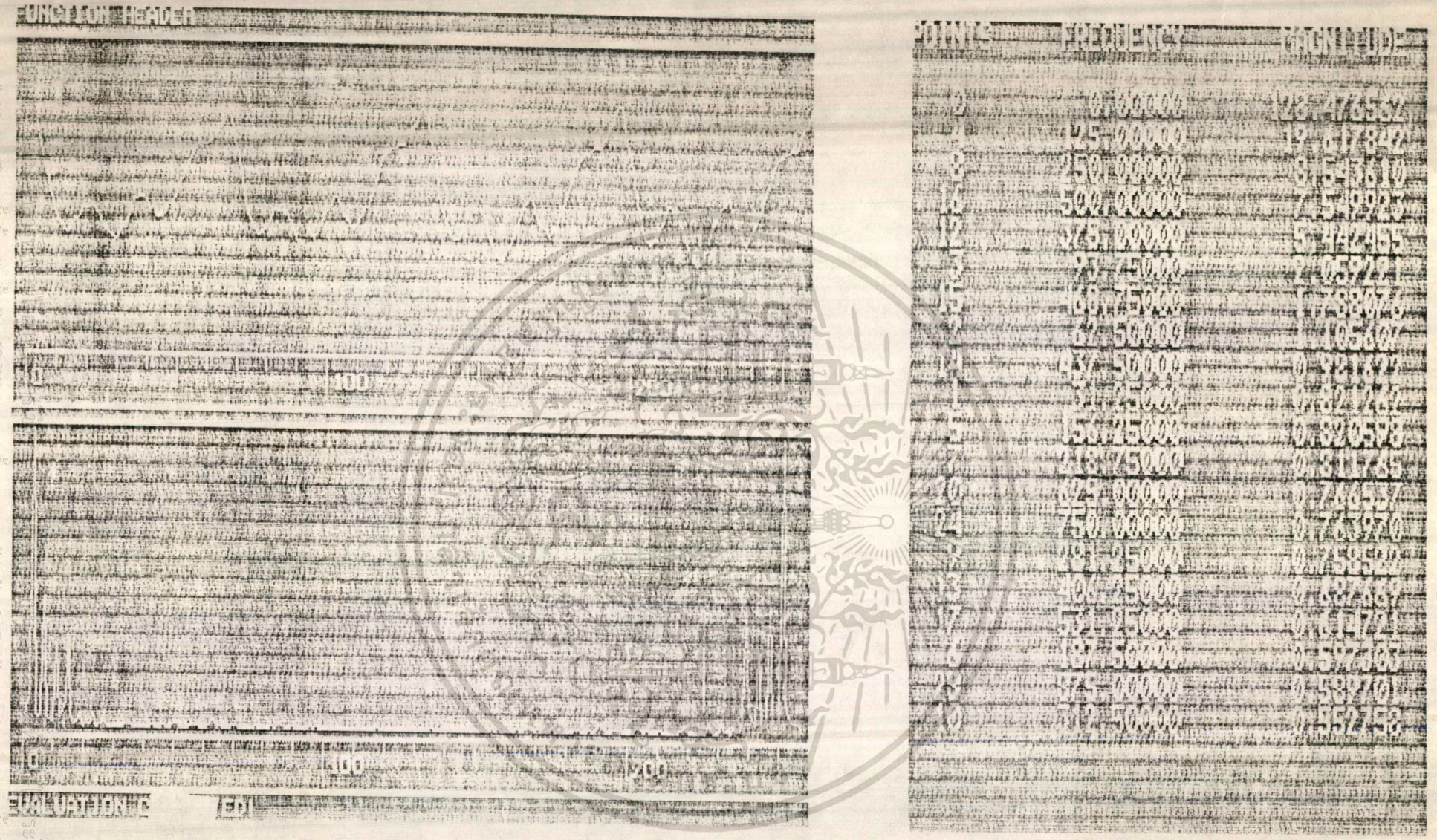
(b) *แสดงการลำดับจุดสุ่มในโดเมนความถี่ที่มีขนาดสูงสุด 20 อันดับแรก*



(a) (b)

รูปที่9.2(a) *แสดงข้อมูลเสียงของคำว่า "สอง" ในโดเมนเวลา และในโดเมนความถี่หลังผ่าน FFT แล้ว โดยใช้วินโดว์สี่เหลี่ยมที่ขนาดของข้อมูล เท่ากับ 256*

(b) *แสดงการลำดับจุดสุ่มในโดเมนความถี่ที่มีขนาดสูงสุด 20 อันดับแรก*



(b)

รูปที่9.1(a) แสดงข้อมูลเสียงของคำว่า "หนึ่ง" ในโดเมนเวลา และในโดเมนความถี่หลังผ่าน FFT แล้ว โดยใช้ในโดเมนสี่เหลี่ยมที่ขนาดของข้อมูล เท่ากับ 256

(b) แสดงการลำดับจุดสุ่มในโดเมนความถี่ที่มีขนาดสูงสุด 20 อันดับแรก

บทที่ 10

สรุปผลและวิจารณ์การทดลอง

10.1 สรุปผลการทดลอง

1.การวิเคราะห์สัญญาณเสียงพูด ควรใช้เครื่องมือที่มีประสิทธิภาพ ยิ่งถ้าเป็นการใช้เทคนิคทางดิจิตอลด้วยแล้ว ควรเลือกใช้เครื่องคอมพิวเตอร์ที่มีความเร็วในการทำงานสูงหรือชิพประมวลผลร่วมทางคณิตศาสตร์(Math coprocessor) จะทำให้การวิเคราะห์ทำได้ด้วยความรวดเร็ว

2.FFT(fast fourier transform) เป็นเครื่องวิเคราะห์สัญญาณในโดเมนเวลา โดยเปลี่ยให้เป็นสเปคตรัมในโดเมนความถี่ ที่ทำงานได้รวดเร็วกว่าและใช้เนื้อที่หน่วยความจำที่ต้องใช้ในการคำนวณน้อยกว่าการคำนวณด้วย DFT โดยตรง เพราะ FFT ใช้อัลกอริทึมที่ลดความซ้ำซากของการเกิดข้อมูล

3.เมื่อใช้วินโดว์สี่เหลี่ยม(rectangular window) วิเคราะห์ในช่วงที่ต้องการ จะมีผลของการรั่วไหลของลอนข้าง(side lobe) ซึ่งสามารถลดผลเสียนี้ได้ โดยเปลี่ยนเป็นวินโดว์ที่มีการถ่วงน้ำหนัก จุดตรงกลางมากที่สุดและลาดลงมาทางจุดเริ่มต้นและจุดสิ้นสุดเช่น แฮมมิ่งวินโดว์ แฮนนิ่งวินโดว์

10.2 วิจารณ์การทดลอง

ในการทดลอง ต้องรันกับเครื่องคอมพิวเตอร์ที่มีความเร็วต่ำ และไม่มีตัวประมวลผลร่วมทางคณิตศาสตร์ ทำให้การตอบสนองไม่เป็นเวลาจริง ทำให้การวิเคราะห์ลำบากมาก ซึ่งเราสามารถแก้ปัญหาจุดนี้ได้โดยใช้โปรเซสเซอร์ที่ออกแบบมาให้ใช้กับการประมวลผลสัญญาณโดยเฉพาะ เช่น TMS32020 ของเท็กซัสอินสตรูเมนท์ หรือเปลี่ยนเป็นใช้วงจรทางอนาลอก โดยใช้เทคนิคฟิลเตอร์แบงค์(filter bank) แบ่งเป็นกลุ่มของแบนพาสฟิลเตอร์ หรือแบ่งช่วงของความถี่ออกเป็นส่วนย่อย ซึ่งอาจใช้ฟิลเตอร์แบงค์สองชุด ชุดแรกเป็นการแบ่งกลุ่มความถี่ของเสียงอย่างคร่าวๆ ก่อนแล้วจึงซอยเป็นความถี่ปลีกย่อยต่อไป

ภาคผนวก ก.

โปรแกรมที่ใช้ควบคุมการทำงานของการ์ด มีรายละเอียดของการใช้งานดังนี้

1.save ทำหน้าที่บันทึกข้อมูลเสียงพูด เมื่อเรียกโปรแกรมนี้แล้ว โปรแกรมจะขึ้นข้อความให้กดปุ่มใดๆ เพื่อให้ลองเสียงก่อนเริ่มบันทึก เมื่อได้ตามที่ต้องการแล้ว ให้กดปุ่มใดๆเริ่มบันทึก และหยุดบันทึก โดยสามารถบันทึกข้อมูลเสียงได้สูงสุด 32 วินาที ที่อัตราการสุ่ม 10 kHz (ความยาวของข้อมูลที่บันทึกได้ขึ้นกับขนาดหน่วยความจำที่จองไว้บนเครื่อง ในโปรแกรมนี้ได้จองไว้ 50000hexa ไบท์) เมื่อเสร็จสิ้นการบันทึก โปรแกรมจะเล่นกลับข้อมูลที่บันทึกทันที เมื่อจบแล้วจะถามว่า ต้องการฟังซ้ำอีกครั้งหรือไม่ ถ้าต้องการฟังซ้ำอีก ให้กดปุ่ม 'y' ถ้าไม่ต้องการฟังอีกให้กดปุ่มใดๆ โปรแกรมจะขึ้นข้อความให้บันทึกชื่อไฟล์ เพื่อเก็บข้อมูลลงบนดิสก์(ควรใส่ชื่อไฟล์ที่สัมพันธ์กับประโยคเพื่อง่ายต่อการจดจำ) หลังจากนั้นโปรแกรมจะถามต่ออีกว่า ต้องการบันทึกประโยคอื่นอีกหรือไม่ ถ้าต้องการให้กดปุ่ม 'y' ถ้าไม่ต้องการให้กดปุ่มใดๆ เพื่อสิ้นสุดโปรแกรม

2.load ทำหน้าที่ดึงข้อมูลของเสียงที่บันทึกด้วยโปรแกรม "save" โดยจะขึ้นข้อความให้ระบุชื่อไฟล์ที่ต้องการเล่นกลับ ให้ใส่ชื่อไฟล์ที่ต้องการ โปรแกรมจะทำการดึงข้อมูลจากดิสก์มาเก็บไว้บนหน่วยความจำ เมื่อเสร็จสิ้นการดึงข้อมูล โปรแกรมจะเล่นกลับเสียงที่เก็บไว้ทันที เมื่อจบการเล่นกลับ โปรแกรมจะขึ้นข้อความถามว่า ต้องการฟังเสียงซ้ำอีกหรือไม่ ถ้าต้องการให้กดปุ่ม 'y' ถ้าไม่ต้องการให้กดปุ่มใดๆ โปรแกรมจะถามต่อว่า ต้องการเล่นกลับข้อมูลจากไฟล์อื่นอีกหรือไม่ถ้าต้องการให้กดปุ่ม 'y' ถ้าไม่ต้องการให้กดปุ่มใดๆ เพื่อสิ้นสุดโปรแกรม

3.displayทำหน้าที่แสดงลักษณะของข้อมูลเสียงที่บันทึกไว้ โดยพล็อตในโหมดกราฟฟิก ทั้งในโดเมนเวลา และโดเมนความถี่หลังจากที่ผ่าน FFT แล้ว เมื่อเรียกใช้โปรแกรมนี้ โปรแกรมจะให้ใส่ชื่อไฟล์ที่ต้องการตรวจสอบลักษณะ ให้ใส่ชื่อไฟล์ที่ต้องการ โปรแกรมจะแสดงข้อมูลเสียงที่บันทึกบนจอภาพส่วนบน ให้กดปุ่มลูกศรขวา เพื่อเลื่อนดูช่วงที่มีเสียงพูด และเลื่อนกลับโดยใช้ปุ่มลูกศรซ้าย เมื่อได้ช่วงเสียงที่ต้องการแปลงเป็นสเปคตรัมแล้ว(ในโปรแกรมตั้งไว้ให้แปลงครั้งละ 256 จุดสุ่ม ซึ่งสามารถเปลี่ยนแปลงได้) ให้เคาะคานวรรค เพื่อเริ่มวิเคราะห์ เมื่อวิเคราะห์เสร็จแล้ว โปรแกรมจะแสดงสเปคตรัมไว้ในจอภาพส่วนล่าง ให้

```
 1  #include <stdio.h>
 2  #include <alloc.h>
 3  #include <ctype.h>
 4  #include <conio.h>
 5  #include <dos.h>
 6  #include <stdlib.h>
 7  #include <math.h>
 8
 9  #define CMD_port  0x303
10  #define DAC_port  0x300
11  #define CNTL_port 0x301
12  #define ADC_port  0x304
13
14  #define START { outportb(CNTL_port,(inportb(CNTL_port)&
    0xf0)|0x04); \
15                  outportb(0x21,inportb(0x21)&0xfb); }
16  #define STOP  { outportb(CNTL_port,inportb(CNTL_port)|0
    x0f); \
17                  outportb(0x21,inportb(0x21)|0x04); }
18  #define EOI     outportb(0x20,0x20);
19
20  #define buf_size  0x500001
21  #define file_size 0x8000
22
23  typedef unsigned char byte;
24
25  int looping=1;
26
27  byte huge *ptr;
28  byte far  *buffer;
29  long index=0;
30  long act_size;
31  char filename[20];
32
33  void interrupt (*kbd_int)();
34  void interrupt (*timer_int)();
35
36  void interrupt keypressed();
37  void interrupt direct_routine();
38  void interrupt record_routine();
39  void interrupt playback_routine();
40
41  void start(void interrupt (*func)());
42  int save_file();
43
44  main()
45  {
46      if ( (buffer=(byte *)farcalloc(  buf_size,sizeof(by
    te) ))==NULL ) {
47          puts("Not enough free space.");
```

```
48          exit(1);
49       }
50       outport(CMD_port,0x80);
51       kbd_int = getvect(9);
52       do {
53       puts("Press any key to record ");
54       start(direct_routine);
55       start(record_routine);
56       act_size = index;
57       puts("Press any key to playback");
58       do {
59          start(playback_routine);
60          puts("Again (y/n) ");
61       } while(tolower(getch())=='y');
62       printf("enter filename : ");
63       scanf("%s",filename);
64       save_file();
65       puts("Record another word\? (y/n)");
66       } while (tolower(getch()) == 'y');
67       puts("Goodbye ! ");
68       farfree((byte *)buffer);
69    }
70
71    void interrupt direct_routine()
72    {
73       outportb(DAC_port,inportb(ADC_port));
74       EOI
75    }
76
77    void interrupt record_routine()
78    {
79       *ptr = inportb(ADC_port);
80       outportb(DAC_port,*ptr);
81       ptr++;
82       if (++index == buf_size ) {
83          looping = 0;
84          STOP
85          setvect(9,kbd_int);
86       }
87       EOI
88    }
89
90    void interrupt playback_routine()
91    {
92       outportb(DAC_port,*ptr++);
93       if (++index == act_size ) {
94          looping = 0;
95          STOP
96          setvect(9,kbd_int);
97       }
```

```
51       kbd_int = getvect(9);
52       do {
53       puts("Press any key to record ");
54       start(direct_routine);
55       start(record_routine);
56       act_size = index;
57       puts("Press any key to playback");
58       do {
59          start(playback_routine);
60          puts("Again (y/n) ");
61       } while(tolower(getch())=='y');
62       printf("enter filename : ");
63       scanf("%s",filename);
64       save_file();
65       puts("Record another word\? (y/n)");
66       } while (tolower(getch()) == 'y');
67       puts("Goodbye ! ");
68       farfree((byte *)buffer);
69    }
70
71    void interrupt direct_routine()
72    {
73       outportb(DAC_port,inportb(ADC_port));
74       EOI
75    }
76
77    void interrupt record_routine()
78    {
79       *ptr = inportb(ADC_port);
80       outportb(DAC_port,*ptr);
81       ptr++;
82       if (++index == buf_size ) {
83          looping = 0;
84          STOP
85          setvect(9,kbd_int);
86       }
87       EOI
88    }
89
90    void interrupt playback_routine()
91    {
92       outportb(DAC_port,*ptr++);
93       if (++index == act_size ) {
94          looping = 0;
95          STOP
96          setvect(9,kbd_int);
97       }
98       EOI
99    }
100
101   void start(void interrupt(*func)())
102   {
103      ptr = buffer;
```

```
104      looping = 1;
105      index = 0;
106      setvect(9,keypressed);
107      setvect(10,func);
108      START
109      while(looping);
110   }
111
112   int save_file()
113   {
114      FILE *stream;
115      int  remainder,times,i;
116
117      if ( (stream=fopen(filename,"wb"))==NULL ) {
118         puts("Can\'t open file");
119         return 1;
120      }
121      remainder = (int)(act_size%file_size);
122      times     = (int)(act_size/file_size);
123      ptr = buffer;
124      freopen(filename,"ab",stream);
125      for (i=0 ;i<times; i++) {
126         fwrite((byte*)ptr,sizeof(byte),file_size,stream);
127         ptr += file_size;
128      }
129      fwrite((byte*)ptr,sizeof(byte),remainder,stream);
130      fclose(stream);
131      return 0;
132   }
133
134   byte temp;
135
136   void interrupt keypressed()
137   {
138      inportb(0x60);
139      temp = inportb(0x61);
140      outportb(0x61,temp|0x80);
141      outportb(0x61,temp);
142      if (looping==1) looping++;
143      else if (looping==2) {
144               looping = 0;
145               STOP
146               setvect(9,kbd_int);
147            }
148      EOI
149   }
```

```
 1  #include <stdio.h>
 2  #include <alloc.h>
 3  #include <conio.h>
 4  #include <ctype.h>
 5  #include <dos.h>
 6  #include <stdlib.h>
 7  #include <math.h>
 8  #include <io.h>
 9
10  #define CMD_port  0x303
11  #define DAC_port  0x300
12  #define CNTL_port 0x301
13  #define ADC_port  0x304
14
15  #define START { outportb(CNTL_port,(inportb(CNTL_port)&
    0xf0)|0x04); \
16                  outportb(0x21,inportb(0x21)&0xfb); }
17  #define STOP  { outportb(CNTL_port,inportb(CNTL_port)|0
    x0f); \
18                  outportb(0x21,inportb(0x21)|0x04); }
19  #define EOI     outportb(0x20,0x20);
20
21  #define buf_size  0x500001
22  #define file_size 0x8000
23
24  typedef unsigned char byte;
25
26  int looping=1;
27
28  byte huge *ptr;
29  byte far  *buffer;
30  long index=0;
31  long act_size;
32  char filename[20];
33
34  void interrupt (*kbd_int)();
35  void interrupt (*timer_int)();
36
37  void interrupt keypressed();
38  void interrupt direct_routine();
39  void interrupt record_routine();
40  void interrupt playback_routine();
41
42  void start(void interrupt (*func)());
43  int load_file();
44
45  main()
46  {
47      if ( (buffer=(byte *)farmalloc(buf_size))==NULL ) {
48         puts("Not enough free space.");
49         exit(1);
50      }
51      outport(CMD_port,0x80);
52      kbd_int = getvect(9);
```

```
 53       do {
 54       printf("enter filename : ");
 55       scanf("%s",filename);
 56       load_file();
 57       puts("Press any key to playback");
 58       do {
 59          start(playback_routine);
 60          puts("Again (y/n) ");
 61       } while(tolower(getch())=='y');
 62       puts("Load another word (y/n)");
 63       } while (tolower(getch())=='y');
 64       puts("Goodbye ! ");
 65       farfree((byte *)buffer);
 66  }
 67
 68  void interrupt direct_routine()
 69  {
 70       outportb(DAC_port,inportb(ADC_port));
 71       EOI
 72  }
 73
 74  void interrupt record_routine()
 75  {
 76     *ptr = inportb(ADC_port);
 77     outportb(DAC_port,*ptr);
 78     ptr++;
 79     if (++index == buf_size ) {
 80       looping = 0;
 81       STOP
 82       setvect(9,kbd_int);
 83     }
 84     EOI
 85  }
 86
 87  void interrupt playback_routine()
 88  {
 89     outportb(DAC_port,*ptr++);
 90     if (++index == act_size ) {
 91        looping = 0;
 92        STOP
 93        setvect(9,kbd_int);
 94     }
 95     EOI
 96  }
 97
 98  void start(void interrupt(*func)())
 99  {
100     ptr = buffer;
101     looping = 1;
102     index = 0;
103     setvect(9,keypressed);
104     setvect(10,func);
105     START
106     while(looping);
```

```
107  }
108
109  int load_file()
110  {
111     int remainder,times,i;
112     FILE *stream;
113
114     if ( (stream=fopen(filename,"rb"))==NULL ) {
115        puts("Can\'t open file");
116        return 1;
117     }
118     act_size  = filelength(fileno(stream));
119     remainder = (int)(act_size%file_size);
120     times     = (int)(act_size/file_size);
121     ptr = buffer;
122     for (i=0 ;i<times; i++) {
123        fread((byte*)ptr,sizeof(byte),file_size,stream);
124        ptr += file_size;
125     }
126     fread((byte*)ptr,sizeof(byte),remainder,stream);
127     fclose(stream);
128     return 0;
129  }
130
131  byte temp;
132
133  void interrupt keypressed()
134  {
135     inportb(0x60);
136     temp = inportb(0x61);
137     outportb(0x61,temp|0x80);
138     outportb(0x61,temp);
139     if (looping==1) looping++;
140     else if (looping==2) {
141              looping = 0;
142              STOP
143              setvect(9,kbd_int);
144           }
145     EOI
146  }
```

DISPLAY.C Tuesday, March 26, 1991 8:09 am Page 1

```
 1  #include <graphics.h>
 2  #include <alloc.h>
 3  #include <stdio.h>
 4  #include <conio.h>
 5  #include <stdlib.h>
 6  #include <math.h>
 7
 8  #define file_size  0x8000
 9
10  typedef unsigned char byte;
11
12  struct data{
13     float rl;
14     int n;
15  };
16
17  struct data dmax[300];
18  FILE *stream;
19  char filename [80];
20  byte far *buffer;
21  byte huge *ptr;
22  byte eraser[320];
23  byte blank[960];
24  long max_size;
25  int r_margin;
26  float rl[300];
27  float im[300];
28  float mag[300];
29  float max;
30  void initialize();
31  void draw_frame();
32  void plot_time(int start);
33  void put_number(int start);
34  void fft(int n,float *xr,float *xi,int inverse);
35  void magnitude(int n,float *xr,float *xi,float *mag);
36  void search_max(int n,float *x,float *max);
37  int flcmp(const struct data *a,const struct data *b);
38
39  main()
40  {
41     char ch;
42     int begin=0,i,j;
43
44     do {
45     initialize();
46     draw_frame();
47     setviewport(10,15,r_margin+9,144,0);
48     plot_time(begin);
49     do {
50        if ((ch=getch())==0) ch=getch();
51        if (ch==77) {
52           begin+=350;
53           clearviewport();
```

```
54              plot_time(begin);
55              put_number(begin);
56           }
57           if (ch==75) {
58              begin-=350;
59              clearviewport();
60              plot_time(begin);
61              put_number(begin);
62           }
63        } while(ch!=32);
64        for (i=0;i<256;i++) {
65           rl[i]=ptr[begin+i];
66           im[i]=mag[i]=0.;
67        }
68        setviewport(0,0,r_margin+19,347,0);
69        putimage(0,338,blank,COPY_PUT);
70        outtextxy(0,347,"EVALUATION!");
71        fft(256,rl,im,0);
72        magnitude(256,rl,im,mag);
73        search_max(256,mag,&max);
74        setviewport(10,180,r_margin+9,309,0);
75        moveto(1,(max-mag[1])*128/max);
76        for (i=2,j=2;j<256;i+=2,j++)
77          lineto(i,(max-mag[j])*128/max);
78        setviewport(0,0,r_margin+19,347,0);
79        putimage(0,338,blank,COPY_PUT);
80        outtextxy(0,347,"EVALUATION COMPLETED!");
81        for (i=0;i<256;i++) {
82          dmax[i].rl = mag[i];
83          dmax[i].n = i;
84        }
85        qsort(&dmax,127,sizeof(dmax[0]),flcmp);
86        getch();
87        restorecrtmode();
88        printf("POINTS     FREQUENCY          MAGNITUDE\n\n");
89        for (i=0;i<20;i++)
90           printf("%4d %15.5f %15f\n",dmax[i].n,dmax[i].n*80
      00./256.,dmax[i].rl);
91        while (getch()!='a');
92        } while (getch()=='b');
93        closegraph();
94        farfree(buffer);
95     }
96
97     void initialize()
98     {
99        int i,gdriver,gmode,times;
100       unsigned int remainder;
101
102       clrscr();
103       printf("Enter file name : ");
104       scanf("%s",filename);
105       if ( ( stream=fopen(filename,"rb") ) == NULL ) {
```

```
106         puts("File not found");
107         exit(1);
108      }
109      max_size = filelength(fileno(stream));
110      remainder = (int)(max_size%file_size);
111      times     = (int)(max_size/file_size);
112      buffer = (byte*)farmalloc(max_size);
113      if (buffer==NULL) {
114         puts("Not enough free space");
115         exit(1);
116      }
117      ptr = buffer;
118      for (i=0 ;i<times; i++) {
119         fread((byte*)ptr,sizeof(byte),file_size,stream);
120         ptr += file_size;
121      }
122      fread((byte*)ptr,sizeof(byte),remainder,stream);
123      fclose(stream);
124      ptr = buffer;
125      detectgraph(&gdriver,&gmode);
126      if ((gdriver==EGA)||(gdriver==VGA)) r_margin=600;
127      else
128        if (gdriver==HERCMONO) r_margin=700;
129        else {
130           puts("Can't run on this monitor type");
131           exit(1);
132        }
133      initgraph(&gdriver,&gmode,"");
134      getimage(0,0,39,7,eraser);
135      getimage(0,0,119,7,blank);
136   }
137
138   void scale()
139   {
140      int i;
141
142      setlinestyle(SOLID_LINE,0,NORM_WIDTH);
143      line(0,132,r_margin-1,132);
144      for (i=0;i<r_margin;i+=2)
145         putpixel(i,133,1);
146      for (i=0;i<r_margin;i+=10)
147         line(i,134,i,136);
148      for (i=0;i<r_margin;i+=20)
149         line(i,137,i,138);
150      for (i=0;i<r_margin;i+=100)
151         line(i,139,i,142);
152      for (i=0;i<r_margin;i+=200)
153         line(i,143,i,148);
154      settextjustify(LEFT_TEXT,BOTTOM_TEXT);
155      settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
156   }
157
158   void put_number(int start)
```

```
159  {
160     int i,j;
161     char number[10];
162
163     for (i=2,j=start;i<r_margin;i+=200,j+=100) {
164        sprintf(number,"%d",j);
165        putimage(2+i,140,eraser,COPY_PUT);
166        outtextxy(2+i,148,number);
167     }
168  }
169
170  void draw_frame()
171  {
172     settextjustify(LEFT_TEXT,TOP_TEXT);
173     outtext("FUNCTION HEADER");
174     setlinestyle(SOLID_LINE,0,THICK_WIDTH);
175     rectangle(0,12,r_margin+19,170);
176     setviewport(10,15,r_margin+9,144,0);
177     scale();
178     put_number(0);
179     setviewport(0,0,r_margin+19,347,0);
180     setlinestyle(SOLID_LINE,0,THICK_WIDTH);
181     rectangle(0,177,r_margin+19,335);
182     outtextxy(0,347,"MESSAGE LINE");
183     setviewport(10,180,r_margin+9,309,0);
184     scale();
185     put_number(0);
186     setviewport(0,0,r_margin+19,347,0);
187  }
188
189  void plot_time(int start)
190  {
191     int i,j;
192
193     moveto(0,(256-ptr[start])/2);
194     for (i=1,j=start+1;i<r_margin;i+=2,j++)
195        lineto(j,(256-ptr[j])/2);
196  }
197
198  void fft(int n,float *xr,float *xi,int inverse)
199  {
200     int log2n,n2,nl,i,j,k,l,le,lel,id;
201     int sign = -1;
202     double ur,ui,wi,wr,tr,ti;
203
204     if (inverse==1) sign = 1;
205     log2n = log(n)/log(2);
206     n2 = n/2; nl = n-1;
207     for (i=j=0; i<nl; i++) {
208        if (i<j) {
209           tr = *(xr+j); ti = *(xi+j);
210           *(xr+j) = *(xr+i);
211           *(xi+j) = *(xi+i);
```

```
212             *(xr+i) = tr; *(xi+i) = ti;
213         }
214         k = n2;
215         while (k<=j) {
216             j -= k; k /= 2;
217         }
218         j += k;
219     }
220     for (k=1; k<=log2n; k++) {
221         le = pow(2.,k);
222         le1 = le/2;
223         ur = 1.; ui = 0.;
224         wr = cos(M_PI/le1);
225         wi = sign*sin(M_PI/le1);
226         for (j=0; j<le1; j++) {
227             for (i=j; i<n; i=i+le) {
228                 id=i+le1;
229                 tr = *(xr+id)*ur - *(xi+id)*ui;
230                 ti = *(xr+id)*ui + *(xi+id)*ur;
231                 *(xr+id) = *(xr+i) - tr;
232                 *(xi+id) = *(xi+i) - ti;
233                 *(xr+i) = *(xr+i) + tr;
234                 *(xi+i) = *(xi+i) + ti;
235             }
236             tr = ur*wr - ui*wi;
237             ti = ur*wi + ui*wr;
238             ur = tr; ui =ti;
239         }
240     }
241     if (inverse!=1)
242         for (i=0; i<n; i++) {
243             *(xr+i) /= n;
244             *(xi+i) /= n;
245         }
246 }
247
248 void magnitude(int n,float *xr,float *xi,float *mag)
249 {
250     int i;
251
252     for (i=0;i<n;i++)
253         mag[i] = sqrt(xr[i]*xr[i]+xi[i]*xi[i]);
254 }
255
256 void search_max(int n,float *x,float *max)
257 {
258     int i;
259
260     *max=x[1];
261     for (i=2;i<n;i++)
262         if (*max<x[i]) *max=x[i];
263 }
264
```

```
265
266  int flcmp(const struct data *a,const struct data *b)
267  {
268     if ( (a->rl) < (b->rl) ) return 1;
269        else if ( (a->rl) == (b->rl) ) return 0;
270             else return -1;
271  }
272
273
```

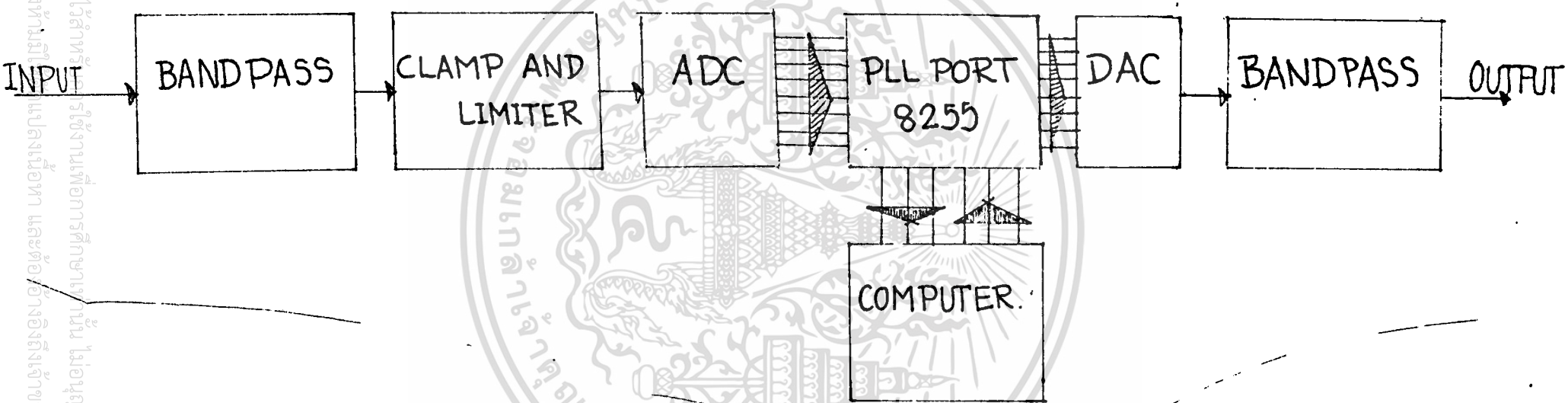
RATE.C Tuesday, March 26, 1991 9:03 am Page 1

```
 1  #include <stdio.h>
 2  #include <conio.h>
 3  #include <dos.h>
 4
 5  #define CMD_port  0x303
 6  #define DAC_port  0x300
 7  #define CNTL_port 0x301
 8  #define ADC_port  0x304
 9
10  #define START { outportb(CNTL_port,(inportb(CNTL_port)&
    0xf0)|0x04); \
11                  outportb(0x21,inportb(0x21)&0xfb); }
12  #define STOP  { outportb(CNTL_port,inportb(CNTL_port)|0
    x0f); \
13                  outportb(0x21,inportb(0x21)|0x04); }
14  #define EOI     outportb(0x20,0x20);
15
16
17  typedef unsigned char byte;
18
19  int looping=1;
20  int latch=0;
21  int count=0;
22
23  void interrupt (*kbd_int)();
24  void interrupt (*timer_int)();
25
26  void interrupt keypressed();
27  void interrupt time_routine();
28  void interrupt direct_routine();
29
30  main()
31  {
32      int x,y;
33
34      outport(CMD_port,0x80);
35      kbd_int = getvect(9);
36      timer_int = getvect(0x1c);
37      printf("Sampling rate = ");
38      x = wherex(); y = wherey();
39      printf("%10.5f kHz",latch*.00455);
40      setvect(9,keypressed);
41      setvect(0x1c,time_routine);
42      looping = 1;
43      while(looping) {
44         gotoxy(x,y);
45         printf("%10.5f",latch*.00455);
46      }
47      setvect(9,kbd_int);
48      setvect(0x1c,timer_int);
49      printf("\n");
50  }
51
```

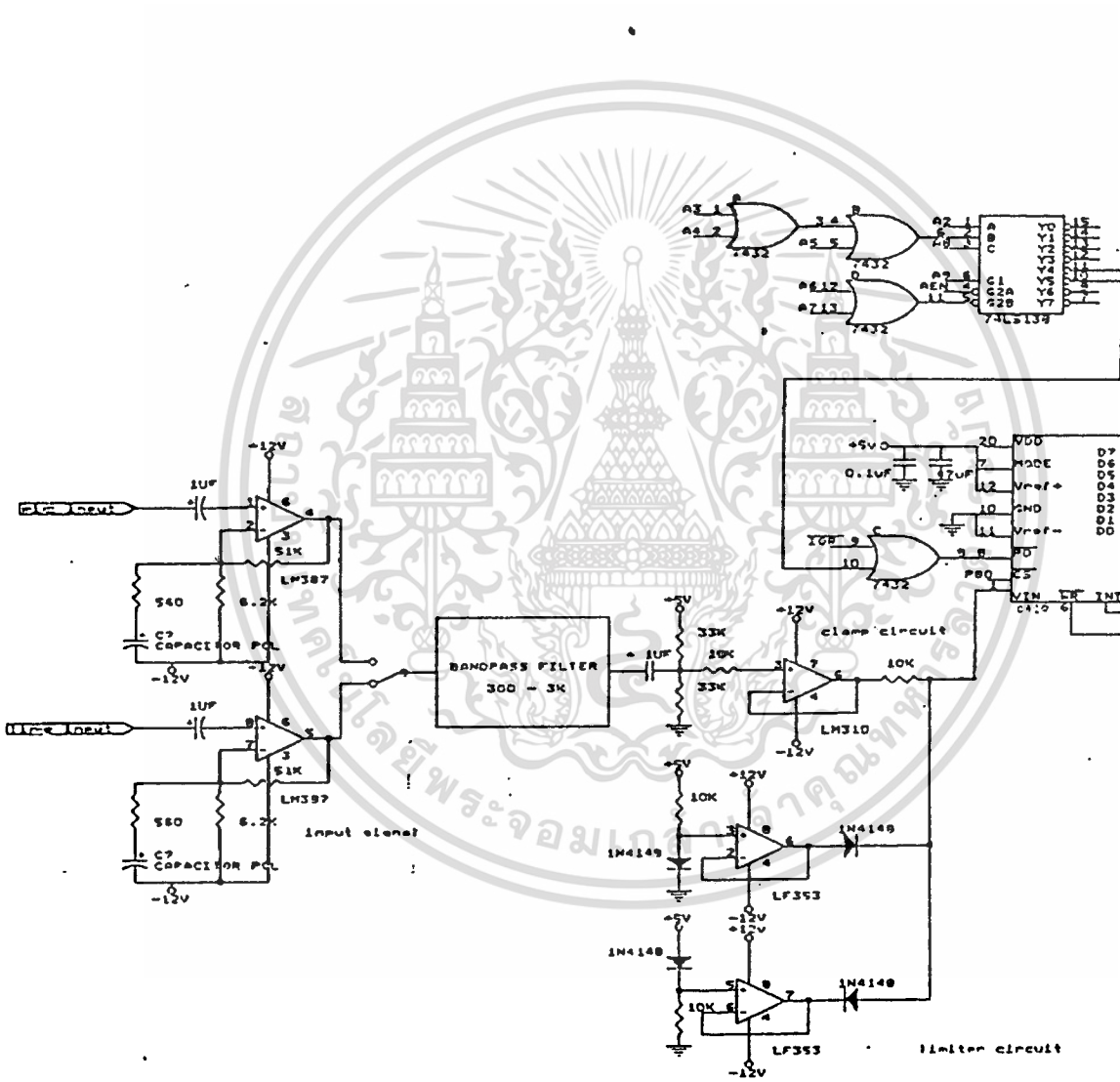
```
52  void interrupt direct_routine()
53  {
54      outportb(DAC_port,inportb(ADC_port));
55      count++;
56      EOI
57  }
58
59  byte temp;
60
61  void interrupt keypressed()
62  {
63     inportb(0x60);
64     temp = inportb(0x61);
65     outportb(0x61,temp|0x80);
66     outportb(0x61,temp);
67     if (looping==1) looping++;
68     else if (looping==2) {
69             looping = 0;
70             STOP
71             setvect(9,kbd_int);
72          }
73     EOI
74  }
75
76  int times = 0;
77
78  void interrupt time_routine()
79  {
80     if (!times) count = 0;
81     if (times==4) {
82             latch = count;
83             times = 0;
84     }
85     else times++;
86  }
87
```

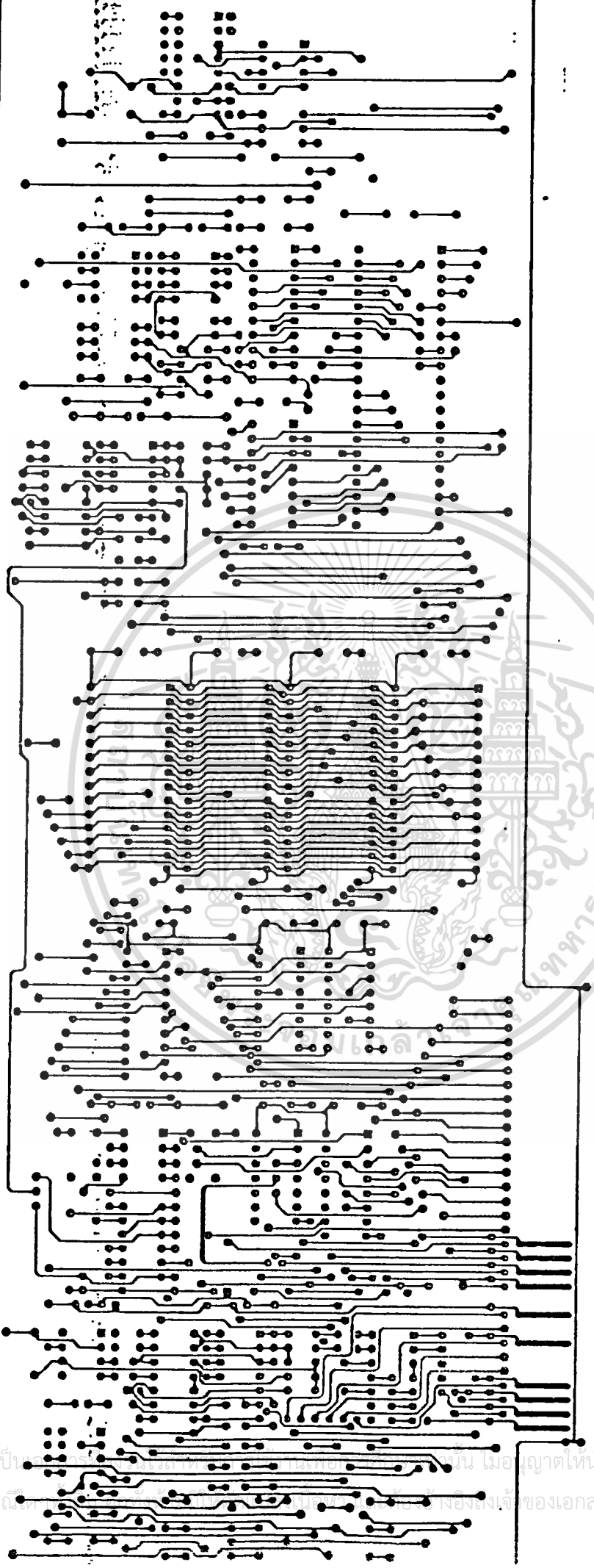
ภาคผนวก ข

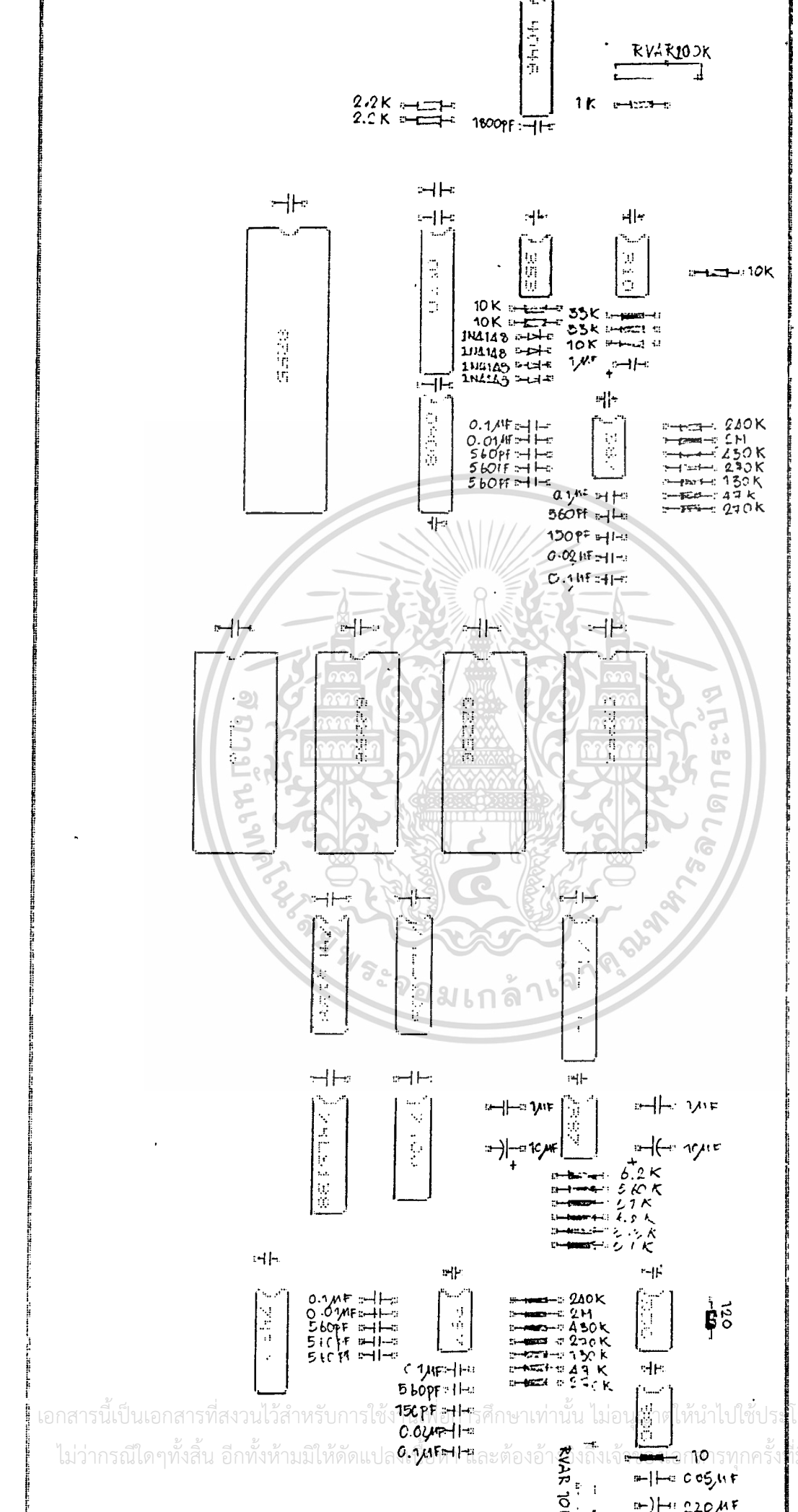
แสดงรายละเอียดของวงจรรวมที่ประกอบเป็นการ์ดทั้งหมด



ภาพแสดงบล็อคไดอะแกรมของการต่อวงจรฮาร์ดแวร์


+12V
1UF
LM387
560
6.2K
C2
CAPACITOR POL
-12V
input signal
BANDPASS FILTER
300 - 3K
33K
10K
clamp circuit
LM310
IN4148
LF353
limiter circuit
7432
74LS139
8255
RESET
100K
1K





กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จขึ้นได้ด้วยคำแนะนำของอาจารย์ฟูศักดิ์ ชีวสุวิทย์ พร้อมทั้งได้รับความช่วยเหลือทางด้านอุปกรณ์ในการทำงานบางส่วนจากอาจารย์วิทยา ทิพย์สุวรรณพร ซึ่งถ้าไม่ได้รับความช่วยเหลือดังกล่าวแล้วปริญญานิพนธ์ฉบับนี้คงจะเสร็จสมบูรณ์ลงมิได้ คณะผู้จัดทำจึงขอกราบขอบพระคุณมา ณ ที่นี้ด้วย พร้อมกับขอขอบคุณเพื่อนๆที่ได้ให้กำลังใจและคำปรึกษาด้วยดีมาโดยตลอดโดยเฉพาะคุณเครือคณิต ศิริสถิตย์กุล ที่ให้ความช่วยเหลือพิมพ์ปริญญานิพนธ์ จนเสร็จสมบูรณ์

คณะผู้จัดทำ

หนังสืออ้างอิง

1.D.O. Shaughnessy., "Speech Communication",Addison-Wesleypublishing company, pp.204-241,1987 .

2.O.E.Brigham., " The Fast Fourier Transform and its applications " ,Prentice-Hall international. Inc.,1988.

3.Martin Giles., "Audio/Radio Handbook" ,National Semi-conductor, pp 43-60 ,1986.

4.M.A.Rashwan and M.M.Fahmy., "New Technique for speakerindependent isolated-word recognition"IEE PROCEDDING, Vol.135 ,pp 251-257 ,1988

5.Roman Kuc.,"Introduction to digital signal processing". McGraw-Hill company. Inc,pp 120-138, 1988