อะแดปทีฟอัลกอริทึมสำหรับ IIR นอตช์ฟิลเตอร์ด้วยวิธีเกรเดียนต์ และการประยุกต์ใช้งาน

ADAPTIVE ALGORITHMS FOR IIR NOTCH FILTER USING GRADIENT OF OPTIMIZATION AND ITS APPLICATIONS

ราชู พันธ์ฉลาด

RACHU PUNCHALARD

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2543

ISBN 974-622-952-4

อะแดปทีฟอัลกอริทึมสำหรับ IIR นอตช์ฟิลเตอร์ด้วยวิธีเกรเดียนต์ และการประยุกต์ใช้งาน

ADAPTIVE ALGORITHMS FOR IIR NOTCH FILTER USING GRADIENT OF OPTIMIZATION AND ITS APPLICATIONS

ราชู พันธ์ฉลาด

RACHU PUNCHALARD

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2543

ISBN 974-622-952-4

ADAPTIVE ALGORITHMS FOR IIR NOTCH FILTER USING GRADIENT OF OPTIMIZATION AND ITS APPLICATIONS

RACHU PUNCHALARD

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2000

ISBN 974-622-952-4

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ อะแดปทีฟอัลกอริทึมสำหรับ IIR นอตช์ฟิลเตอร์ด้วยวิธีเกรเดียนต์และการประยุกต์ใช้งาน

ADAPTIVE ALGORITHMS FOR IIR NOTCH FILTER USING GRADIENT OF OPTIMIZATION AND ITS APPLICATIONS

ชื่อนักศึกษา นายราชู พันธ์ฉลาด

รหัสประจำตัว 41061152

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชา วิศวกรรมไฟฟ้า

อาจารย์ผู้ควบคุมวิทยานิพนธ์ รศ.ชวลิต เบญจางคประเสริฐ

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
ผศ.ดร.ปิติเขต	สู้รักษา	
รศ.ดร.ปราโมทย์	วาดเขียน	
รศ.ดร.กนก	เจนจิระพงศ์เวช	
รศ.ชวลิต	เบญจางคประเสริฐ	

วัน/เดือน/ปี ที่สอบ 6 ตุลาคม 2543 เวลา 12.00-13.00 น.

สถานที่สอบ ห้องสอบวิทยานิพนธ์ คณะวิศวกรรมศาสตร์ ตึก 12 ชั้น 4 ห้อง (E12-404)

บัณฑิตวิทยาลัยรับรองแล้ว

(รศ.ดร.บุญวัฒน์ อัตชู)

รักษาราชการแทนคณบดีบัณฑิตวิทยาลัย

วันที่ 17 **เดือน** ตุลาคม **พ.ศ.** 2543

หัวข้อวิทยานิพนธ์	อะแดปทีฟอัลกอริทึมสำหรับ IIR นอตช์ฟิลเตอร์ด้วยวิธีเกรเดียนต์และการประยุกต์ใช้งาน
นักศึกษา	นายราชู พันธ์ฉลาด
รหัสประจำตัว	41061152
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมไฟฟ้า
พ.ศ.	2543
ชื่ออาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ.ชวลิต เบญจางคประเสริฐ

บทคัดย่อ

วิทยานิพนธ์นี้นำเสนอ อะแดปทีฟอัลกอริทึม และ การประยุกต์ใช้งาน ตัวกรองอะแดปทีฟนอตช์แบบ IIR อันดับสอง สำหรับการประมาณค่าความถี่ในสัญญาณรบกวน และใช้สำหรับกำจัด สัญญาณคลื่นซายน์ความถี่ 50 Hz ของไฟฟ้ากระแสสลับ 220 โวลต์ ที่เหนี่ยวนำเข้าไปรบกวนสัญญาณคลื่นไฟฟ้าหัวใจ (ECG) ของผู้ป่วยขณะที่แพทย์กำลังบันทึก อะแดปทีฟอัลกอริทึมที่นำเสนอนี้ ประกอบด้วยสามแบบ คือ Quantized Least Mean *p*-Power (QLMP), Normalized Quatized Least Mean *p*-Power (NQLMP) และ Variable Step-Size Quantized Least Mean *p*-Power (VSQLMP) ซึ่งได้พัฒนามาจากอะแดปทีฟอัลกอริทึมแบบ Least Mean *p*-Power (LMP) อะแดปทีฟอัลกอริทึมทั้งสาม จะมีสมรรถนะในการทำงานดีกว่าแบบ LMP เช่น อะแดปทีฟอัลกอริทึมแบบ QLMP จะช่วยลดการคำนวณ อะแดปทีฟอัลกอริทึมแบบ NQLMP จะสามารถทำงานได้เร็ว และอะแดปทีฟอัลกอริทึมแบบ VSQLMP จะทำงานได้เร็วและมีความถูกต้องของคำตอบสูง จากผลการทดลอง โดยการจำลองการทำงานด้วยโปรแกรมคอมพิวเตอร์ และ การทดลองที่เวลาจริง (Real Time) พบว่า ผลการทดลองมีความสอดคล้องกัน ซึ่งยืนยันได้ว่าอะแดปทีฟอัลกอริทึมทั้งสามที่ได้พัฒนาขึ้น จะสามารถนำไปประยุกต์ใช้งานจริงได้

Thesis Title	Adaptive Algorithms For IIR Notch Filter Using Gradient of Optimization and Its Applications
Student	Mr. Rachu Punchalard
Student ID.	41061152
Degree	Master of Engineering
Programme	Electrical Engineering
Year	2000
Thesis Advisor	Assoc. Prof. Chawalit Benjangkaprasert King Mongkut's Institute of Technology Ladkrabang

ABSTRACT

The thesis introduces an adaptive algorithm and an application of second order adaptive IIR notch filter. This application is used for frequency estimation in background noise and for eliminating 50 Hz power line interference which induces to disturb the signal of electocardiogram (ECG) of patient while the physician is recording it. The presented adaptive algorithm consists of three types which are Quantized Least Mean *p*-Power (QLMP), Normalized Quatized Least Mean *p*-Power (NQLMP), and Variable step-size Quantized Least Mean *p*-Power (VSQLMP). All of them are developed from adaptive of Least Mean *p*-Power (LMP). They have more performance than LMP; for example, adaptive algorithm of QLMP will reduce the calculation, adaptive algorithm of NQLMP has fast convergence speed of rate, and adaptive algorithm of VSQLMP has fast convergence speed and highly correct solution. The application which simulates the working with program computer and real time application show that the result of both applications has consistency. The all of these developed algorithms can be used for the real application.

กิตติกรรมประกาศ

ผู้เขียนวิทยานิพนธ์ขอกราบขอบพระคุณ :รศ. ชวลิต เบญจางคประเสริฐ และ รศ. ดร. กนก เจนจิระพงศ์เวช ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ ที่ให้คำปรึกษา และแนวความคิดสำหรับการทำวิจัยเป็นอย่างดี

ขอขอบคุณ คุณพ่อ-คุณแม่ ที่ให้กำลังใจในการเรียน ภาควิชาวิศวกรรมโทรคมนาคม มหาวิทยาลัยเทคโนโลยีมหานครที่ให้ทุนในการศึกษาตลอดระยะเวลา 2 ปี และ สนับสนุนเครื่องมือสำหรับทำวิทยานิพนธ์ อาจารย์ประวิทย์ ชุมชู อาจารย์ประจำภาควิชาวิศวกรรมโทรคมนาคม ที่ให้คำปรึกษาในสิ่งที่เป็นประโยชน์ต่อการทำวิทยานิพนธ์ จนสำเร็จลงได้ด้วยดี ผศ.ดร.อธิคม ฤกษบุตร อาจารย์ ชาญชัย เลาหะเพ็ญแสง อาจารย์วรพงษ์ ศิลาพันธ์ อาจารย์อนุรี หล่อสวัสดิ์ศิริ อาจารย์ธวัชชัย เมธีวรัญญู อาจารย์อรรถพล ยิ้มสกุล อาจารย์พุทธพงศ์ สมใจ อาจารย์วุฒิพร เลิศวาสนา คุณพิมลวรรณ เรืองฤทธิ์ และ คุณสิริมดี ศรีเพ็ญ ที่ได้ให้ความช่วยเหลือในบางสิ่งที่เป็นประโยชน์ต่อการทำวิทยานิพนธ์

สารบัญ

หน้า

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

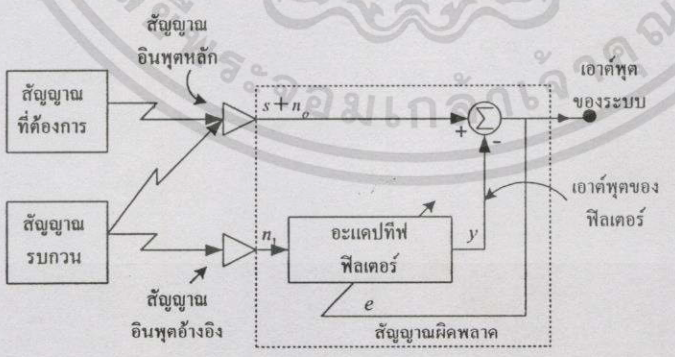
บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญต่อปัญหา

การประมาณค่าสัญญาณซายน์คลื่นเดี่ยวที่ปนอยู่ในสัญญาณรบกวนนั้น สามารถทำได้โดยการป้อนสัญญาณดังกล่าว ผ่านตัวกรองความถี่ ซึ่งตัวกรองความถี่จะทำการกดสัญญาณรบกวนให้มีขนาดต่ำลง และยอมให้สัญญาณที่ต้องการผ่านออกไป โดยไม่มีความผิดเพี้ยน ซึ่งตัวกรองความถี่ในลักษณะนี้จะเรียกว่า ตัวกรองที่เหมาะ (optimum filter) ซึ่งนำเสนอโดย Wiener หรือที่รู้จักกันทั่วไปว่า ตัวกรองความถี่วีนเนอร์ (Wiener filter)

ตัวกรองความถี่ที่จะนำมาใช้สำหรับประมาณค่าสัญญาณซายน์จะแบ่งเป็นสองชนิด คือ หนึ่ง ตัวกรองความถี่แบบคงที่ (Fixed filter) และ สอง ตัวกรองความถี่แบบปรับตัวได้ (Adaptive filter) การนำตัวกรองความถี่แบบคงที่มาใช้นั้น มีความจำเป็นจะต้องทราบคุณสมบัติของสัญญาณก่อน (prior knowledge) ทั้งสัญญาณที่ต้องการประมาณ และสัญญาณรบกวน แต่ถ้านำเอาตัวกรองความถี่แบบอะแดปทีฟมาใช้ ซึ่งสามารถปรับพารามิเตอร์ของตัวเองได้ ทำให้ไม่จำเป็นต้องทราบคุณสมบัติของสัญญาณ หรือต้องการเพียงเล็กน้อยเท่านั้น

ในปี ค.ศ. 1975 Widrow [1] ได้นำเสนอแนวความคิดในการกำจัดสัญญาณรบกวนแบบอะแดปทีฟขึ้นเป็นครั้งแรก และต่อมาได้ถูกนำมาประยุกต์ใช้กับระบบสื่อสาร และระบบการประมวลผลสัญญาณดิจิตอล หรือ ระบบควบคุม กันอย่างกว้างขวาง หลักการที่ Widrow ได้นำเสนอแสดงดังรูปที่ 1.1



รูปที่ 1.1 แนวความคิดในการกำจัดสัญญาณรบกวนแบบอะแดปทีฟ

จากรูปที่ 1.1 ตัวกรองแบบอะแดปทีฟ ต้องการสัญญาณอินพุตสองสัญญาณ คือสัญญาณอินพุตหลัก (primary input) และสัญญาณอินพุตอ้างอิง (reference input) ซึ่งสัญญาณอินพุตอ้างอิงจะได้จากอุปกรณ์ตรวจจับสัญญาณรบกวน (sensors) ซึ่งอาจจะมีหลาย ๆ ตัวโดยวางอยู่ในตำแหน่งที่มีสัญญาณรบกวน หรือตำแหน่งที่สัญญาณที่ต้องการมีขนาดต่ำ ซึ่งสัญญาณนี้จะถูกกรองโดยตัวกรองอะแดปทีฟ และนำไปลบกับสัญญาณอินพุตหลัก ซึ่งประกอบไปด้วยสัญญาณที่ต้องการและสัญญาณรบกวน เป็นผลให้สัญญาณรบกวนถูกลดทอนลง กระบวนการนี้แม้จะดูเหมือนว่าอันตราย เพราะถ้าหากกระทำอย่างไม่เหมาะสมแล้ว จะเป็นการเพิ่มกำลังงานของสัญญาณรบกวนที่สัญญาณเอาต์พุต แต่อย่างไรก็ตาม ถ้ากระบวนการกรอง และการลบถูกควบคุมอย่างเหมาะสมแล้ว กระบวนการทางอะแดปทีฟจะสามารถทำการลดสัญญาณรบกวนเป็นไปอย่างสมบูรณ์ โดยทำให้สัญญาณมีความผิดเพี้ยนน้อยที่สุด

การประยุกต์ใช้งานสำหรับการประมวลผลสัญญาณดิจิตอลในระบบสื่อสาร และเรดาร์นั้น ต้องการที่จะประมาณ (estimate), ตรวจวัด (detect) หรือยกระดับ (enhance) สัญญาณคลื่นซายน์ซึ่งถูกรบกวนด้วยสัญญาณรบกวน โดยจะขึ้นกับวัตถุประสงค์ของแต่ละการประยุกต์ใช้ ซึ่งปริมาณที่สนใจก็คือ ความถี่ ขนาด และ เฟส ของสัญญาณซายน์ อะแดปทีฟอัลกอริทึมที่ใช้สำหรับตรวจวัด หรือ ยกระดับปริมาณทั้งสามนี้จะเรียกว่า Adaptive Line Enhancers (ALE's) โดยสร้างจากตัวกรองอะแดปทีฟแบบ Finite Impulse Response (FIR) [1-3] ต่อมาได้มีการนำเอาโครงสร้างแบบ Infinite Impulse Response (IIR) มาใช้แทน FIR [4-8] เพราะมีข้อดีหลายประการเช่น ประสิทธิภาพในการคำนวณดีกว่า โดยเฉพาะเมื่อความถี่ของสัญญาณซายน์น้อยกว่าความถี่สุ่มมาก ๆ เมื่อนำอะแดปทีฟ FIR อันดับ 4 ไปใช้สำหรับตรวจวัดความถี่คลื่นซายน์ที่มีค่าต่ำ ๆ สองความถี่ หลังจากตรวจวัดได้แล้ว จะทำให้ตัวกรอง FIR อันดับ 4 กลายเป็นอันดับสอง ที่มีซีโร่ (zeros) วางอยู่ที่ตำแหน่งความถี่เดียวกัน ทั้งนี้เป็นเพราะความถี่ที่ตรวจวัดมีค่าต่ำนั่นเอง และเนื่องจากไม่มีโพล (pole) จะทำให้ฟังก์ชันถ่ายโอน (transfer function) มีอัตราการขยายที่สูงขึ้นเมื่อความถี่มีค่าสูงขึ้น ถ้าสัญญาณซายน์ที่เอาต์พุตถูกรบกวนด้วยสัญญาณรบกวนที่มีองค์ประกอบของความถี่สูงรวมอยู่ด้วย จะทำให้องค์ประกอบของสัญญาณรบกวนนี้ถูกขยายขึ้นด้วย ทำให้ความสามารถในการตรวจวัดความถี่ลดน้อยลง การแก้ปัญหาทำได้โดยการเพิ่มจำนวนอันดับของตัวกรองให้มากขึ้น ซึ่งก็จะทำให้การคำนวณเพิ่มมากขึ้นตามไปด้วย

วิธีการหนึ่งที่ใช้ได้ดีสำหรับการตรวจวัดความถี่คลื่นซายน์ความถี่เดียว คือใช้ตัวกรองความถี่อะแดปทีฟ IIR แบบนอตช์อันดับสอง [4-17] ทำงานร่วมกับอะแดปทีฟอัลกอริทึม แบบ IIR LMS (Least Mean Square) ซึ่งเป็นอะแดปทีฟอัลกอริทึมบนพื้นฐานของเกรเดียนต์ (Gradient-Based Algorithm) [18-20] แต่จะมีข้อเสียหลัก คือ การหาคำตอบทำได้ช้า จึงได้มีการทำวิจัยเพื่อปรับปรุงคุณสมบัติทางด้านความเร็วในการหาคำตอบ (convergence properties) เช่นใน [13] ได้ใช้วิธีหลายอัตราสุ่ม (multirate techniques) ซึ่งมีข้อดีที่สามารถตรวจวัดความถี่คลื่นซายน์หลาย ๆ

ความถี่ที่อยู่ใกล้ ๆ กันได้อย่างมีประสิทธิภาพ ในบทวิจัย [10] ได้ใช้วิธีเปลี่ยนแปลงโครงสร้างเป็นแบบแลตทิส (Lattice) ทำให้สามารถหาคำตอบได้เร็วขึ้น ในบทวิจัย [14] ได้ใช้วิธีการปรับโพลของฟังก์ชันถ่ายโอนและปรับฟอร์เกตติงแฟคเตอร์ (forgetting factor) ร่วมด้วย การปรับปรุงด้วยวิธีต่าง ๆ ย่อมจะต้องเพิ่มความซับซ้อนในการคำนวณตามไปด้วย บทวิจัย [12] ได้นำเสนออะแดปทีฟอัลกอริทึมแบบ IIR LMP (Least Mean p-Power Error Criterion) ซึ่งแตกต่างจากบทวิจัยที่ได้กล่าวมาทั้งหมด หลักการที่ใช้คือ จะพิจารณาค่าผิดพลาดเฉลี่ยของสัญญาณเอาต์พุตของตัวกรองอะแดปทีฟแบบนอตช์ยกกำลัง p แทนที่จะพิจารณาที่ยกกำลังสอง เหมือนกับ IIR LMS ผลปรากฏว่าอะแดปทีฟอัลกอริทึมสามารถทำงานเร็วขึ้น เมื่อกำลัง p มีค่าสูง แต่อย่างไรก็ตาม เมื่อค่า p สูงขึ้นจะทำให้ความสามารถในการกำจัดสัญญาณรบกวนแบบอิมพัลส์ลดน้อยลง นอกจากนี้เมื่อนำตัวกรองอะแดปทีฟแบบนอตช์ไปใช้สำหรับกำจัดสัญญาณคลื่นซายน์ความถี่ 50 Hz ของไฟฟ้ากระแสสลับ 220 โวลต์ ที่เหนี่ยวนำเข้าไปรบกวนคลื่นไฟฟ้าหัวใจ (Electrocardiogram : ECG) [19,21-23] ของผู้ป่วยขณะที่แพทย์กำลังบันทึก ที่ค่า p สูง จะทำให้ความสามารถในการกำจัดสัญญาณรบกวนแย่ลง และในตอนท้ายของบทวิจัย [12] ได้สรุปว่า การจะนำอะแดปทีฟอัลกอริทึมแบบ LMP ไปประยุกต์ใช้ในงานต่าง ๆ นั้นถ้าเลือกค่า p ที่เหมาะสมแล้วจะสามารถทำงานได้ดีว่าแบบ LMS

ปัญหาที่น่าสนใจก็คือถ้าทำให้ค่า p ไม่มีผลกระทบต่อการทำงานของอะแดปทีฟตัวกรองแล้วก็จะทำให้อะแดปทีฟอัลกอริทึมที่ได้มีคุณสมบัติที่ดีกว่าแบบ LMS และแบบ LMP ซึ่งแนวทางหนึ่งที่น่าสนใจได้ถูกนำเสนอโดยผู้เขียนวิทยานิพนธ์ ในบทวิจัย [15-16] จากบทวิจัยดังกล่าว ได้นำวิธีการควอนไตซ์มาใช้ ผลปรากฏว่าค่า p ไม่มีผลกระทบต่อการทำงานของตัวกรอง และยังพบข้อดีอื่นอีก คือสามารถลดการคำนวณลงได้หนึ่งครั้งต่อหนึ่งตัวอย่างเอาต์พุต และที่ค่า p เท่ากันจะสามารถหาคำตอบได้เร็วกว่า LMP นอกจากนี้ จะสามารถทนทานต่อสัญญาณรบกวนแบบอิมพัลส์ได้ดีกว่า ซึ่งข้อดีทั้งหมดดังกล่าวทำได้โดยการเลือกให้ $p = 1$ เพียงค่าเดียว และใช้ชื่อว่า QLMP

ต่อมาในบทวิจัย [17] ได้นำเสนออะแดปทีฟอัลกอริทึมโดยใช้วิธีการ ควอนไตซ์-นอร์มอลไลซ์เกรเดียนต์ (NQLMP) พบว่าอะแดปทีฟอัลกอริทึมสามารถตรวจวัดความถี่ได้เร็วเพิ่มขึ้นกว่า QLMP แต่มีข้อเสียคือ หลังจากที่อะแดปทีฟอัลกอริทึมเข้าสู่สภาวะคงตัว (steady state) แล้ว ค่าสัมประสิทธิ์ของตัวกรองที่ประมาณได้จะมีค่าสัญญาณรบกวนเกรเดียนต์ (Gradient noise) [19] มีค่าสูง อันเกิดจากกระบวนการรีเคอซีฟของอะแดปทีฟอัลกอริทึม เป็นผลให้ค่าความแปรปรวน (variance) ของสัมประสิทธิ์มีค่าสูงด้วย นอกจากนี้ เมื่อกำหนดให้ p มีค่าสูง จะทำให้สัญญาณรบกวนเกรเดียนต์ยิ่งเพิ่มสูงขึ้น ซึ่งในวิทยานิพนธ์นี้จะได้นำเสนอวิธีการสำหรับลดสัญญาณรบกวนเกรเดียนต์โดยการใช้วิธีการที่ทำให้ค่าสเต็ปไซส์เปลี่ยนตามเวลา ซึ่งจะทำให้สามารถลดค่าความแปรปรวนของสัมประสิทธิ์ลงได้ ซึ่งได้นำเสนอโดยผู้เขียนวิทยานิพนธ์ในบทวิจัย [43]

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

วิทยานิพนธ์นี้นำเสนออะแดปทีฟอัลกอริทึมสามแบบ แต่ละแบบจะพัฒนามาจากอะแดปทีฟอัลกอริทึมในบทวิจัย [12] ดังนี้คือ 1) แบบควอนไตซ์เกรเดียนต์ [15-16] 2) แบบควอนไตซ์-นอร์มอลไลซ์เกรเดียนต์ [17] และ 3) แบบปรับเปลี่ยนสเต็บไซส์-ควอนไตซ์เกรเดียนต์ [43] ทำงานร่วมกับตัวกรอง IIR แบบนอตช์อันดับสอง สำหรับการตรวจวัดความถี่สัญญาณคลื่นซายน์ความถี่เดียวที่ถูกรบกวนด้วยสัญญาณรบกวน ที่มีใช้ในระบบสื่อสาร, ระบบเรดาร์ และระบบควบคุม และนำไปใช้สำหรับกำจัดสัญญาณคลื่นซายน์ความถี่ 50 Hz ออกจากคลื่นไฟฟ้าหัวใจของผู้ป่วยในขณะกำลังบันทึก

1.3 แนวความคิดที่ใช้ในการวิจัย

แนวความคิดแรกที่ใช้ในบทวิจัย [15-16] คือทำการควอนไตซ์ค่าเกรเดียนต์ให้มีค่าเป็น 1 (เมื่อค่าเกรเดียนต์มากกว่าศูนย์) และ –1 (เมื่อค่าเกรเดียนต์น้อยกว่าศูนย์) ซึ่งค่าเกรเดียนต์นี้ จะคูณอยู่กับค่าสเต็ปไซส์ ซึ่งหลังจากควอนไตซ์แล้วจะทำให้อัตราความเร็วในการทำงานของอะแดปทีฟอัลกอริทึมขึ้นกับค่าสเต็บไซส์ ในขณะที่มีสัญญาณรบกวนอิมพัลส์เข้ามารบกวน แม้จะทำให้ค่าเกรเดียนต์เปลี่ยนไป แต่จะไม่มีผลต่อสัมประสิทธิ์ที่กำลังปรับ เพราะอะแดปทีฟอัลกอริทึมนี้ จะคิดเฉพาะเครื่องหมายของค่าเกรเดียนต์เท่านั้น (+1 และ –1) แนวความคิดที่สองได้พัฒนาจากบทวิจัย [15-16] ซึ่งนำเสนอในบทวิจัย [17] คือ นำเอาค่าพลังงานของค่าเกรเดียนต์ ไปหารค่าสเต็บไซส์ ทำให้ค่าสเต็บไซส์เปลี่ยนตามเวลา โดยในช่วงเริ่มต้นการทำงานค่าพลังงานจะมีค่าต่ำ ทำให้ค่าสเต็บไซ์มีค่าสูง ทำให้อะแดปทีฟอัลกอริทึมทำงานได้เร็ว แนวความคิดที่สามคือ จะทำการปรับค่าสเต็บไซส์ให้เปลี่ยนตามเวลาโดยไม่มีการหาร โดยจะใช้ค่าพลังงานของสัญญาณเอาต์พุตของตัวกรองมาใช้ในการปรับ วิธีนี้นอกจากจะสามารถทำงานได้ที่เวลาจริงแล้ว ยังจะทำให้อะแดปทีฟอัลกอริทึมทำงานได้เร็ว และสัมประสิทธิ์ที่ประมาณได้มีความเที่ยงตรงสูงอีกด้วย

1.4 ขอบเขตของการวิจัย

วิทยานิพนธ์จะขยายความจากบทวิจัย [15-17] โดยจะกล่าวถึงที่มาของหลักการที่ใช้ในบทวิจัย นอกจากนี้จะได้นำเสนอวิธีแก้ไขปัญหาที่เกิดขึ้นจากบทวิจัย [15-17] เพื่อเป็นแนวทางในการพัฒนาในอนาคต และใช้โปรแกรมคอมพิวเตอร์สำหรับจำลองการทำงานของอะแดปทีฟอัลกอริทึมแบบต่าง ๆ ที่ได้นำเสนอ ร่วมกับการใช้ตัวประมวลผลสัญญาณดิจิตอลเบอร์ TMS320C50 ของบริษัท Texas Instrument สำหรับทดลองการทำงานที่เวลาจริง ของตัวกรองความถี่แบบอะแดปทีฟตามที่ได้นำเสนอในวิทยานิพนธ์

บทที่ 2
ตัวกรองความถี่ดิจิตอล

ตัวกรองความถี่แบบดิจิตอล (digital filter) แบ่งได้เป็นสองชนิด [29] คือ ตัวกรองความถี่แบบ Finite Impulse Response (FIR filter) และตัวกรองความถี่แบบ Infinite Impulse Response (IIR filter) ตัวกรองความถี่แบบ FIR นั้น ฟังก์ชันถ่ายโอน (transfer function) จะมีเฉพาะซีโร่ (zero) ไม่มีโพล (pole) ทำให้ระบบมีความเสถียรภาพอย่างแน่นอน และมีผลตอบสนองทางเฟสเป็นเชิงเส้น (linear phase) แต่มีข้อเสียคือ ต้องใช้จำนวนอันดับ (order) สูงจึงจะทำให้ตัวกรองทำงานได้ดี ส่วนตัวกรองความถี่แบบ IIR นั้น ฟังก์ชันถ่ายโอนจะประกอบด้วย ทั้งซีโร่และโพล ทำให้มีปัญหาทางด้านเสถียรภาพ แต่ถ้ามีการออกแบบที่ดี จะทำให้ปัญหาดังกล่าวน้อยลง หรือไม่เกิดขึ้นเลย ข้อดีของตัวกรองความถี่แบบ IIR เมื่อเทียบกับแบบ FIR คือ ที่สมรรถนะของการทำงาน (performance) เท่ากัน ตัวกรองความถี่แบบ IIR จะใช้จำนวนอันดับน้อยกว่า ซึ่งทำให้การคำนวณน้อยกว่าด้วย ซึ่งความซับซ้อนในการคำนวณนี้ มีความสำคัญมากเมื่อนำตัวกรองความถี่ไปใช้งานที่เวลาจริง (real time) ด้วยไมโครโปรเซสเซอร์ ซึ่งมีข้อจำกัดทางด้านจำนวนบิต และความเร็วในการทำงาน ตัวกรองความถี่ที่มีจำนวนการคำนวณน้อยกว่า จะทำให้ไมโครโปรเซสเซอร์ทำงานน้อยลง และทำงานได้เร็วกว่า ในวิทยานิพนธ์นี้ได้นำเอาตัวกรองความถี่แบบ IIR มาใช้ ดังนั้นเนื้อหาที่จะได้กล่าวต่อไป จะได้กล่าวเฉพาะตัวกรองความถี่แบบ IIR เท่านั้น ส่วนตัวกรองความถี่แบบ FIR สามารถดูได้จากเอกสารอ้างอิง [24-28] การเลือกใช้งานระหว่าง FIR และแบบ IIR สามารถสรุปได้ [30] ดังต่อไปนี้คือ

1) การประยุกต์ใช้งานทางด้านการสื่อสารข้อมูล (data transmission), ทางการแพทย์ (biomedicine), และการประมวลผลสัญญาณภาพ จำเป็นต้องใช้ตัวกรองความถี่ที่ให้ผลตอบสนองทางเฟสเป็นเชิงเส้น ดังนั้น FIR จะเหมาะสมกว่าแบบ IIR
2) สำหรับการใช้งานที่เวลาจริงซึ่งต้องมีการจำกัดจำนวนบิตนั้น การคำนวณสัมประสิทธิ์ของตัวกรองจะต้องทำการปัดเศษ (round-off) ซึ่งจะทำให้เกิด round-off noise ขึ้นซึ่ง round-off noise นี้จะเกิดขึ้นใน FIR น้อยกว่าที่เกิดขึ้นใน IIR
3) ในกรณีที่มีความต้องการความชันในช่วงแถบหยุด (cutoff) สูงนั้น FIR จะต้องใช้จำนวนสัมประสิทธิ์มากกว่า IIR
4) IIR สามารถออกแบบได้โดยตรงจากตัวกรองต้นแบบในตัวกรองความถี่แบบแอนะลอกแต่ FIR ทำไม่ได้
5) ตัวกรองความถี่แบบ FIR จะสังเคราะห์ได้ค่อนข้างยากถ้าหากไม่ใช้คอมพิวเตอร์ (CAD) ช่วยในการออกแบบ

จากทั้งหมดที่กล่าวมาพอจะสรุปเป็นแนวทางสำหรับการใช้งานได้ดังนี้

- จะใช้ IIR ก็ต่อเมื่อในงานที่มีความต้องการความชันในช่วงแถบหยุดสูง ซึ่ง IIR จะใช้จำนวนสัมประสิทธิ์น้อยกว่า FIR
- จะใช้ FIR ถ้าจำนวนสัมประสิทธิ์ไม่มากจนเกินไป โดยเฉพาะสำหรับงานที่ต้องการความผิดเพี้ยนทางเฟสน้อยจะเหมาะสมอย่างยิ่ง

2.1 ตัวกรองความถี่แบบ IIR (IIR filters)

ตัวกรองความถี่แบบ IIR เป็นส่วนประกอบที่สำคัญอันหนึ่งในระบบการประมวลผลแบบไม่ต่อเนื่องทางเวลา (Discrete-time processing) มีข้อดีหลาย ๆ ข้อที่เหนือกว่าตัวกรองกรองแบบ FIR โดยเฉพาะอย่างยิ่ง ด้านผลตอบสนองทางขนาด เช่น เมื่อมีความต้องการให้ความกว้างของช่วงความถี่แถบผ่าน (pass band) หรือ ช่วงแถบความถี่ไม่ผ่าน (stop band) มีขนาดแคบมาก ๆ หรือต้องการใช้ช่วงแถบเปลี่ยน (transition band) มีค่าแคบมาก ๆ หรือมีความต้องการให้มีอัตราการลดทอนที่สูง ตัวกรองความถี่แบบ IIR เหมาะที่จะนำมาใช้งานมากกว่าแบบ FIR

ตัวกรองความถี่แบบ IIR จะมีสมการของสัญญาณเอาต์พุตที่เป็นฟังก์ชันของสัญญาณอินพุตปัจจุบัน อินพุตในอดีตและสัญญาณเอาต์พุตในอดีต (สัญญาณเอาต์พุตในอดีตได้จากการป้อนกลับ) ซึ่งสามารถเขียนให้อยู่ในรูปสมการผลต่าง (difference equation) ได้ดังนี้

$$y(n) = -a_1 y(n-1) - a_2 y(n-2) - \ldots - a_N y(n-N) + b_0 x(n) + b_1 x(n-1) + \ldots + b_M x(n-M) \qquad (2.1)$$

เมื่อ $x(n)$ คือลำดับของสัญญาณอินพุต และ $y(n)$ คือลำดับของสัญญาณเอาต์พุต N คือจำนวนตัวอย่าง (samples) ทั้งหมดของสัญญาณเอาต์พุตที่ทราบค่า สมการที่ (2.1) สามารถนำไปใช้ในการคำนวณหาสัญญาณเอาต์พุตต่อ ๆ กันไปได้ รูปแบบของการนำเอาสัญญาณเอาต์พุตก่อนหน้ามาใช้สำหรับคำนวณหาสัญญาณเอาต์พุตตัวต่อไปจะเรียกว่า รีเคอร์ซีฟ (recursive) โดยทั่วไป ตัวกรองความถี่แบบ IIR และตัวกรองความถี่แบบ recursive มักจะนำมาใช้ในความหมายอย่างเดียวกัน ทั้งนี้เพราะจากสมการที่ (2.1) สามารถนำไปใช้สร้างตัวกรองความถี่ทั้งสองแบบได้เหมือนกัน คำว่า IIR จะหมายถึงรูปแบบของผลตอบสนองอิมพัลส์ (impulse response) ของตัวกรองความถี่ ในขณะที่คำว่า recursive จะหมายถึงตัวกรองความถี่นี้ถูกสร้างขึ้นอย่างไร ตัวกรองความถี่แบบ FIR จะสามารถสร้างในรูปแบบของ recursive ได้ด้วยในขณะเดียวกันตัวกรองความถี่แบบ IIR ก็จะสามารถสร้างในรูปแบบ nonrecursive ได้เหมือนกัน

เพื่อความสะดวกจะนิยามตัวกรองความถี่แบบรีเคอร์ซีฟด้วยกับฟังก์ชันถ่ายโอน หรือฟังก์ชันของระบบ (system function, ฟังก์ชันของระบบก็คือการแปลง z (z-transform) ของผลตอบ

สนองอิมพัลส์ของตัวกรองความถี่) โดยฟังก์ชันของระบบนี้จะอยู่ในรูปเศษส่วน (rational function) ในตัวแปร z^{-1} ระบบตามสมการที่ (2.1) จะมีฟังก์ชันของระบบเป็นไปตามสมการ คือ

$$H(z)=\frac{\sum_{k=0}^{M} b_k z^{-k}}{1+\sum_{k=1}^{N} a_k z^{-k}} \tag{2.2}$$

จากสมการที่ (2.2) ถ้ากำหนดให้เงื่อนไขเริ่มต้นเป็นศูนย์ จะสามารถเขียนใหม่ได้ คือ

$$H(z)=G\frac{\prod_{k=0}^{M}(1-\beta_k z^{-1})}{\prod_{k=1}^{N}(1-\alpha_k z^{-1})} \tag{2.3}$$

ราก (roots) ของโพลิโนเมียลตัวเศษ β_k เรียกว่าซีโร่ของตัวกรองความถี่ และรากของตัวส่วน α_k เรียกว่าโพล และ G เป็นอัตราการขยาย ซึ่งมีค่าคงที่ โดยทั่วไป จำนวนของซีโร่ และโพลจะขึ้นกับความต้องการของผู้ใช้งาน จำนวนอันดับที่ N (order N) ของตัวกรองความถี่แบบ IIR จะหาได้จากจำนวนรากของโพลที่อยู่ในระนาบ z ที่มีค่าจำกัด (finite z-plane)

ตัวกรองความถี่ที่มีความเป็น เชิงเส้นไม่แปรตามเวลา (Linear Time-Invariance:LTI) จะมีคุณสมบัติเป็นคอซอล (causal) ถ้าผลตอบสนองอิมพัลส์มีค่าเท่ากับศูนย์เมื่อ $n < 0$ ตัวอย่างสัญญาณเอาต์พุตของตัวกรองความถี่ ที่เป็นคอซอล จะขึ้นกับตัวอย่างสัญญาณอินพุตปัจจุบัน และในอดีตเท่านั้น ถ้าจำกัดสัญญาณอินพุตให้มีค่าเป็นศูนย์ $x(n)=0$ สำหรับ $n < 0$ และค่าเริ่มต้นของ $y(-1)=y(-2)=\ldots=y(-N)=0$ จะทำให้ตัวกรองความถี่แบบรีเคอร์ซีฟตามสมการที่ (2.1) มีคุณสมบัติเป็นคอซอล ความเป็นคอซอลของตัวกรองความถี่ มีความสำคัญมากในการประยุกต์ใช้งานที่เวลาจริง (real time) เมื่อมีการทิก (tick) ของสัญญาณนาฬิกาจะได้ตัวอย่างอินพุต 1 ตัวอย่างจากนั้นตัวกรองความถี่จะต้องสร้างตัวอย่างของเอาต์พุตออกมา 1 ตัวอย่างด้วย (ในความเป็นจริงจะต้องไม่มีสัญญาณอินพุตก่อนเวลา $n = 0$ แน่นอน)

ข้อควรคำนึงถึงอีกอันหนึ่งของตัวกรองความถี่แบบ IIR ก็คือความมีเสถียรภาพ (stable) ถ้าตัวกรองความถี่ไม่เสถียรภาพ (unstable) จะทำให้ลำดับของสัญญาณเอาต์พุตเพิ่มขึ้นอย่างไม่มีขอบเขต ถ้าสัญญาณอินพุตยังคงป้อนให้อยู่ ซึ่งเสถียรภาพของตัวกรองความถี่แบบ IIR จะขึ้นอยู่กับตำแหน่งโพลของฟังก์ชันระบบในระนาบ z ตัวกรองความถี่แบบ IIR ที่คุณสมบัติเป็น causal LTI จะมีเสถียรภาพ ถ้าค่าโพลเป็นไปตามเงื่อนไข $|\alpha_k|<1$ ซึ่งหมายความว่าตำแหน่งโพลทุกตัวจะต้องอยู่ในวงกลมหนึ่งหน่วย (unit circle) บนระนาบ z นั่นเอง

2.2 โครงสร้างของตัวกรองความถี่แบบ IIR (Structures for IIR filter)

ในหัวข้อนี้จะกล่าวถึงโครงสร้างของตัวกรองความถี่แบบ IIR โดยแสดงในรูปของ signal flow graph [31] โดยจะกล่าวถึงเฉพาะโครงสร้างแบบ Direct Forms เนื่องจากเป็นโครงสร้างที่ใช้ในวิทยานิพนธ์ โครงสร้างที่ดีและเหมาะจะนำไปใช้งานจะต้องเป็นโครงสร้างที่มีความซับซ้อนน้อย คือ มีการคูณและตัวหน่วง (delay) ที่น้อย ทั้งนี้เพราะโดยทั่วไปแล้วการคูณ จะต้องใช้เวลามากและมีราคาสูงเมื่อนำไปทำเป็นฮาร์ดแวร์ และจำนวนตัวหน่วง หมายถึง จำนวนหน่วยความจำซึ่งหากมีตัวหน่วงน้อย หน่วยความจำที่ใช้ก็น้อยลงด้วย และสิ่งที่ได้จากลดจำนวนการคูณอีกอย่างก็คือจะทำให้ความเร็วในการทำงานของ CPU เร็วขึ้น ซึ่งจะได้กล่าวในรายละเอียดของแต่ละแบบ ดังต่อไปนี้

2.2.1 โครงสร้างแบบ Direct Forms

จากสมการที่ (2.1) และ (2.2) ประกอบไปด้วยการบวกจำนวน M+1 ครั้ง สำหรับพจน์อินพุต $x(n)$, $x(n\text{-}1)$,..., $x(n\text{-}M)$ และ N ครั้งสำหรับพจน์เอาต์พุต $y(n\text{-}1)$, $y(n\text{-}2)$,..., $y(n\text{-}N)$ และแต่ละพจน์จะถูกถ่วงน้ำหนักด้วยสัมประสิทธิ์ของตัวกรองความถี่ จากสมการที่ (2.1) สามารถเขียนใหม่ได้ว่า

$$u(n) = b_0 x(n) + b_1 x(n-1) + b_2 x(n-2) + ... + b_M x(n-M) \tag{2.4}$$

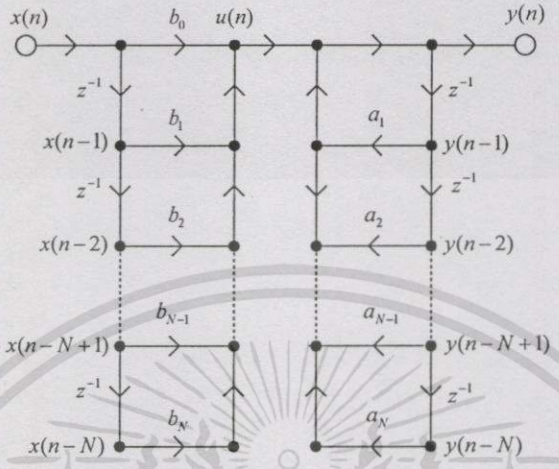
$$y(n) = -a_1 y(n-1) - a_2 y(n-2) - ... - a_N y(n-N) + u(n) \tag{2.5}$$

จากสมการ (2.4) และ (2.5) จะเห็นว่าตัวกรองความถี่แบบ IIR จะประกอบด้วยระบบย่อย (subsystem) สองระบบต่อกันเป็นชั้น (cascade) กัน กล่าวคือ ชั้นที่หนึ่งเป็นของฟังก์ชันเศษ ในสมการที่ (2.2) และชั้นที่สองเป็นของหนึ่งหารด้วยฟังก์ชันส่วนของสมการที่ (2.2) เมื่อนำสองส่วนนี้มารวมกันจะได้โครงสร้างแบบ Direct Form I ซึ่งสามารถเขียนเป็น signal flow graph สำหรับตัวกรองความถี่อันดับ N ตามรูปที่ 2.1

จะสังเกตเห็นว่าโครงสร้างแบบ Direct Form I จะมีจำนวนพจน์ทั้งหมด M+N+1 พจน์ คือ $x(n)$, $x(n\text{-}1)$,..., $x(n\text{-}M)$, $y(n\text{-}1)$, $y(n\text{-}2)$,..., $y(n\text{-}N)$ แต่ละพจน์จะถูกคูณด้วยสัมประสิทธิ์ และนำมาบวกกัน ซึ่งจำนวนของการคูณสำหรับตัวอย่างเอาต์พุตแต่ละค่าจะเท่ากับจำนวนสัมประสิทธิ์ที่เป็น nontrivial เท่านั้น (การคูณด้วย 1, -1 และ 0 จะไม่นำมาคิดเป็นการคูณ)

จากสมการที่ (2.2) ตัวเศษของฟังก์ชันระบบ ต้องการสัญญาณอินพุตเป็น $x(n)$ และให้สัญญาณเอาต์พุตเป็น $u(n)$ และตัวส่วน ต้องการสัญญาณอินพุตเป็น $u(n)$ และให้สัญญาณเอาต์พุตเป็น $y(n)$ จากรูปที่ 2.1 ถ้าทำการพลิก (reverse) แต่ละระบบย่อย จะทำให้ตัวอย่างข้อมูลที่ถูกเก็บในระบบย่อยที่ 1 สามารถเก็บไว้ร่วมกับระบบย่อยที่ 2 ได้ ซึ่งโครงสร้างลักษณะนี้จะเรียกว่า Direct

Form II หรือ Canonic Direct Form ซึ่งมีข้อดีคือต้องการจำนวนตัวหน่วงน้อยกว่า โดยจะมีค่าสูงสุดเท่ากับ M, N และจะใช้การคูณเท่ากับ $M+N+1$ ครั้ง โครงสร้างแบบ Direct Form II จะมีสมการผลต่าง คือ

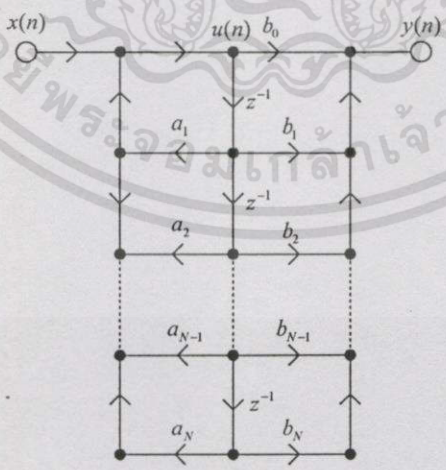


รูปที่ 2.1 Signal flow graph ของโครงสร้างแบบ Direct Form I สำหรับระบบอันดับ N

$$u(n) = x(n) - a_1 u(n-1) - a_2 u(n-2) - \ldots - a_N u(n-N) \tag{2.6}$$

$$y(n) = b_0 u(n) + b_1 u(n-1) + b_2 u(n-2) + \ldots + b_M u(n-M) \tag{2.7}$$

และ signal flow graph แสดงดังรูปที่ 2.2



รูปที่ 2.2 Signal flow graph ของโครงสร้างแบบ Direct Form II สำหรับระบบอันดับ N

ตัวกรองความถี่อะแดปทีฟแบบนอตช์ที่นำเสนอในวิทยานิพนธ์เป็นแบบ IIR อันดับสอง (2^{nd} order) และใช้โครงสร้างแบบ direct form I และเนื่องจากตัวกรองความถี่แบบ IIR จะมีปัญหาทางด้านความมีเสถียรภาพกล่าวคือ หากมีโพลเพียง 1 ตัวอยู่นอกวงกลมหนึ่งหน่วยก็จะทำให้ตัวกรองความถี่ไร้เสถียรภาพทันที ดังนั้นในหัวข้อถัดไปจะได้กล่าวถึงวิธีทดสอบความมีเสถียรภาพของตัวกรองความถี่แบบ IIR อันดับสอง

2.2.2 การทดสอบเสถียรภาพ (stability test) [32]

เพื่อที่จะทำให้ฟังก์ชันถ่ายโอนในสมการที่ (2.3) เป็นคอซอล และมีความเสถียรภาพแบบ BIBO (Bound In Bound Out) นั้น จะต้องทำให้โพลทุกตัวอยู่ในวงกลมหนึ่งหน่วย (unit circle) เท่านั้น การวิเคราะห์เพื่อหาตำแหน่งโพลของฟังก์ชันถ่ายโอนที่อันดับสูงกว่าสองนั้นทำได้ค่อนข้างยาก และจะต้องใช้โปรแกรมคอมพิวเตอร์ช่วยในการหาราก แต่สำหรับฟังก์ชันถ่ายโอนอันดับสองมีวิธีการง่าย ๆ ในการทดสอบเสถียรภาพดังนี้

ให้

$$D(z) = 1 + a_1 z^{-1} + a_2 z^{-2} \tag{2.8}$$

โดยที่ $D(z)$ คือตัวส่วนอันดับสองของสมการที่ (2.2) จากสมการที่ (2.8) จะสามารถเขียนให้อยู่ในรูปโพลดังสมการ

$$D(z) = (1-\alpha_1 z^{-1})(1-\alpha_2 z^{-1}) = 1-(\alpha_1+\alpha_2)z^{-1} + \alpha_1\alpha_2 z^{-2} \tag{2.9}$$

เปรียบเทียบสมการที่ (2.8) กับสมการที่ (2.9) จะได้

$$a_1 = -(\alpha_1+\alpha_2),\ a_2 = \alpha_1\alpha_2 \tag{2.10}$$

เพื่อให้ฟังก์ชันถ่ายโอนมีเสถียรภาพตำแหน่งโพลจะต้องอยู่ภายในวงกลมหนึ่งหน่วยนั่นคือ

$$|\alpha_1| < 1, \quad |\alpha_2| < 1$$

เนื่องจากสมการที่ (2.8) สัมประสิทธิ์ a_2 ถูกกำหนดเป็นผลคูณของโพลดังนั้นจะได้ว่า

$$|a_2| < 1 \tag{2.11}$$

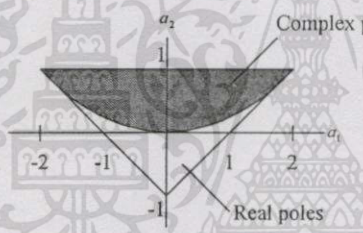
และจะสามารถกำหนดรากของโพลิโนเมียล $D(z)$ ดังนี้คือ

$$\alpha_1 = -\frac{a_1 + \sqrt{a_1^2 - 4a_2}}{2}, \quad \alpha_2 = -\frac{a_1 - \sqrt{a_1^2 - 4a_2}}{2} \tag{2.12}$$

และจากสมการที่ (2.12) จะได้ว่า

$$|a_1| < 1 + a_2 \tag{2.13}$$

ถ้าสัมประสิทธิ์ของตัวกรองความถี่เป็นไปตามสมการที่ (2.11) และสมการที่ (2.13) จะสามารถวาดขอบเขตของค่าสัมประสิทธิ์ที่ยังคงทำให้ตัวกรองความถี่มีความเสถียรเป็นรูปสามเหลี่ยมในระนาบ (a_1, a_2) ซึ่งรู้จักกันทั่วไปในชื่อของ สามเหลี่ยมเสถียรภาพ (stability triangle) สำหรับฟังก์ชันถ่ายโอนดิจิตอลอันดับสอง แสดงดังรูปที่ 2.3



รูปที่ 2.3 สามเหลี่ยมเสถียรภาพสำหรับฟังก์ชันถ่ายโอนแบบดิจิตอลอันดับสอง

โพลที่เกิดจากฟังก์ชันถ่ายโอนอันดับสองนั้นจะมีสามชนิด คือโพลคอนจุเกตเชิงซ้อน (complex conjugate poles), โพลค่าจริงที่ไม่ซ้ำกัน (real and unequal) และ โพลค่าจริงและเท่ากัน (real and equal) โพลที่เป็นคอนจุเกตเชิงซ้อนมักจะเกิดขึ้นถ้า $a_1^2 < 4a_2$ ซึ่งในกรณีนี้ โพลแต่ละค่าจะอยู่ที่ตำแหน่งรัศมี r จากจุดกำเนิดของวงกลมหนึ่งหน่วย และที่มุม θ ซึ่งกำหนดได้ตามสมการดังนี้คือ

$$\alpha_1 = r\angle\theta, \quad \alpha_2 = r\angle -\theta \tag{2.14}$$

เมื่อ

$$r = a_2^{1/2}, \quad \theta = \cos^{-1}\left(\frac{-a_1}{2r}\right) \tag{2.15}$$

จากสมการที่ (2.11) และ (2.13) สามารถใช้เป็นสูตรสำหรับประมาณจำนวนบิตเมื่อนำไปใช้งานที่เวลาจริง ที่ทำให้ระบบมีเสถียรภาพได้ซึ่งเป็นหนึ่งในหลาย ๆ วิธีที่มีใช้กัน สำหรับวิธีการหาเงื่อนไขของสัมประสิทธิ์ที่ทำให้มีเสถียรภาพของ ฟังก์ชันถ่ายโอนอันดับสูง ๆ สามารถดูได้จากเอกสารอ้างอิง [32]

2.3 ระบบตัวเลขในการประมวลผล [33]

การสร้างตัวกรองความถี่ด้วยฮาร์ดแวร์ หรือซอฟแวร์จะมีผลของ finite-word-length effect เกิดขึ้น หมายความว่า ตัวเลขที่ใช้คำนวณจะถูกแทนด้วยจำนวนบิตที่จำกัด ซึ่งจะทำให้ค่าตัวเลขที่ถูกแทนมีความแม่นยำจำกัด และนำไปสู่ความผิดพลาดจากการปัดเศษ (round-off errors) และความไม่เป็นเชิงเส้น (nonlinear effects) ใน performance ของตัวกรองความถี่ ในหัวข้อนี้จะกล่าวถึงผลกระทบจากการทำควอนไตซ์ที่เกิดขึ้นในระบบประมวลผลสัญญาณดิจิตอล โดยจะกล่าวถึงเฉพาะระบบตัวเลขแบบจำนวนเต็ม (fixed point number) เท่านั้นทั้งนี้เพราะในการทดลองที่เวลาจริงใช้การคำนวณด้วยตัวเลขดังกล่าวนี้

2.3.1 ระบบเลขจำนวนเต็ม (Fixed-point Numbers system)

คำว่า fixed-point จะสามารถแปลตามตรงได้ว่า คือ ระบบตัวเลขที่มีจุดทศนิยมคงที่ แต่เนื่องจากภายหลังจากการแปลงแล้วมองดูเหมือนเลขจำนวนเต็ม (คือไม่มีทศนิยม) ประกอบกับตัวแปรชนิดเลขจำนวนเต็ม (integer) ในภาษาต่าง ๆ ก็ใช้การแทนค่าด้วยระบบเลขชนิดนี้ จึงอนุโลมเรียกว่า ระบบเลขจำนวนเต็ม

ขออธิบายด้วยการยกตัวอย่าง สมมติว่าจะใช้เลขฐานสองจำนวน 8 บิตแทนค่าของตัวเลข ซึ่งเลขแปดบิตนี้เริ่มต้นที่ค่า 00000000_2 และสิ้นสุดที่ 11111111_2 ซึ่งรวมทั้งสิ้น 256 หรือ 2^8 ค่า ในการนำเลขฐานสองนี้ไปแทนค่าที่จะใช้ ทำได้โดยการสมมติตำแหน่งจุดทศนิยมขึ้นมา ซึ่งตำแหน่งของจุดทศนิยมจะมีผลต่อช่วงของค่าที่จะแทนได้ ดังนี้คือ

ถ้ากำหนดให้ N คือ จำนวนบิตที่อยู่ก่อนหน้าจุดทศนิยมและ M คือจำนวนบิตที่อยู่หลังจุดทศนิยมหรือเขียนเป็นสัญลักษณ์ได้ว่า

$$\overbrace{XXXXX}^{N\ Bits}.\overbrace{XXXXXXX}^{M\ Bits}$$

โดย X แทนตำแหน่งของบิต ค่าที่แต่ละบิตแทนคือ $2^{N-1}\ldots2^1\ 2^0\ .2^{-1}2^{-2}\ldots2^{-M}$ และค่าที่แทนได้คือ 0 และนับขึ้นไปทีละ 2^{-M} จนกระทั่งถึง 2^N-2^{-M} ซึ่งเป็นค่าตัวสุดท้าย ตัวอย่างเช่นสำหรับเลข 8 บิต ถ้า $N = 8, M = 0$ หรือจุดทศนิยมอยู่ขวามือสุดจะได้

เขียนเป็นสัญลักษณ์คือ $XXXXXXXX.$

ค่าของแต่ละบิตคือ $2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0$

ค่าที่สามารถแทนได้คือ 0, 1, 2, ..., 256

ถ้า $N = 0, M = 8$ หรือจุดทศนิยมอยู่ซ้ายมือสุดจะได้

เขียนเป็นสัญลักษณ์คือ $.XXXXXXXX$

ค่าของแต่ละบิตคือ $2^{-1} 2^{-2} 2^{-3} 2^{-4} 2^{-5} 2^{-6} 2^{-7} 2^{-8}$

ค่าที่สามารถแทนได้คือ $0, 2^{-8}, 2(2^{-8}), 3(2^{-8}), ..., 1-2^{-8}$

ซึ่งเท่ากับ 0, 0.00390625, 0.0078125, 0.01171875, ..., 0.99609375

ถ้า $N = 3, M = 5$ จะได้

เขียนเป็นสัญลักษณ์คือ $XXX.XXXXX$

ค่าของแต่ละบิตคือ $2^2 2^1 2^0 \quad 2^{-1} 2^{-2} 2^{-3} 2^{-4} 2^{-5}$

ค่าที่สามารถแทนได้คือ $0, 2^{-5}, 2(2^{-5}), 3(2^{-5}), ..., 2^2-2^{-5}$

ซึ่งเท่ากับ 0, 0.03125, 0.0625, 0.09375, ..., 3.96875

จากตัวอย่างข้างต้น จะเห็นว่าจำนวนตำแหน่งจุดทศนิยม จะมีผลต่อค่าที่แทนได้ ดังนั้นในการใช้ระบบเลขจำนวนเต็ม เพื่อสร้างตัวกรองความถี่ที่เวลาจริง เพื่อแทนค่าตัวเลขทศนิยม นอกจากจะต้องกำหนดจำนวนบิตที่ใช้แล้ว ยังต้องบอกว่าใช้จุดทศนิยมอยู่ที่ตำแหน่งไหน หรือบอกว่า N กับ M มีค่าเท่าใดด้วย และเมื่อเลือกตำแหน่งจุดทศนิยมแล้ว ให้คงตำแหน่งไว้ที่ตำแหน่งนั้นตลอดการคำนวณหนึ่ง ๆ ในกรณีที่ค่าที่ต้องการแทนไม่ลงตัวพอดี เป็นค่าที่สามารถแทนได้จะต้องทำการปัดเศษให้เป็นค่าที่ใกล้เคียงกับค่าที่ต้องการแทนที่สุด ซึ่งมีวิธีการทำดังนี้คือ

ถ้าให้ a เป็นค่าที่ต้องการแทน และ $0 \le a < 2^N$ และสมมติให้ a_{fix} คือหลังจากเปลี่ยนเป็นรูปแบบจำนวนเต็ม จะสามารถหา a_{fix} ซึ่งเป็นตัวแทนที่ใกล้ค่าของ a มากที่สุดดังนี้คือ

$$a_{fix} = round(a \times 2^M) \tag{2.16}$$

โดยนิยามให้ $round(.)$ คือ ฟังก์ชันในการปัดเศษเป็นจำนวนเต็ม ถ้าค่าที่ใส่ให้มีส่วนทศนิยมที่น้อยกว่า 0.5 ก็จะตัดทิ้ง แต่ถ้ามากว่าหรือเท่ากับ 0.5 ก็จะปัดเพิ่มเป็น 1 การปัดเศษนี้ทำให้เกิดความคลาดเคลื่อนขึ้น โดยค่าที่ a_{fix} แทนจะเป็น $\frac{a_{fix}}{2^M} - a$ ซึ่งไม่เท่ากับค่า a เริ่มต้นที่ต้องการ (จะ

เท่ากันในกรณีที่ $a \times 2^M$ ได้ค่าเป็นจำนวนเต็มเท่านั้น) ดังนั้นในการแปลงเป็นเลขจำนวนเต็มค่าความคลาดเคลื่อนที่เกิดขึ้นคือ

$$err = \frac{a_{fix}}{2^M} - a \tag{2.17}$$

ตัวอย่างเช่น ต้องการแปลง 0.66 เป็นเลขจำนวนเต็ม 8 บิต โดยทำสองกรณีคือ $M = 8$ และ $M = 7$

กรณีที่ $M = 8$ จะได้

$$\begin{aligned} a_{fix} &= round(0.66 \times 2^8) \\ &= round(168.96) \\ &= 169 \end{aligned}$$

ค่า a_{fix} = 169 นี้จะสามารถแทนได้ด้วยเลขฐานสองที่มีจำนวน 8 บิตได้แน่นอนซึ่งในที่นี้คือ a_{fix} = 10101001_2 ปกติแล้วไม่ต้องแปลงเป็นเลขฐานสอง ทั้งนี้เพราะซอฟท์แวร์ส่วนใหญ่ที่จะต้องนำค่าไปใช้ต่อสามารถรับค่าเป็นเลขจำนวนเต็มฐานสิบได้ ค่า a_{fix} =169 นี้เทียบกับค่าทศนิยมที่แทนอยู่คือ $\frac{a_{fix}}{2^8} = 0.66015625$ ซึ่งหมายถึงว่าค่า $a = 0.66$ นั้นไม่สามารถแทนได้พอดีด้วยรูปแบบจำนวนเต็ม 8 บิต แต่ค่าที่สามารถแทนได้ใกล้เคียงที่สุดคือ 0.66015625 ซึ่งตรงกับ a_{fix}=169 ดังนั้นความคลาดเคลื่อนที่เกิดขึ้น คือ

$$err = \frac{169}{2^8} - 0.66 = 0.00015625$$

กรณีที่ $M = 7$ จะได้

$$\begin{aligned} a_{fix} &= round(0.66 \times 2^7) \\ &= 84 \end{aligned}$$

และค่าความคลาดเคลื่อนคือ

$$err = \frac{84}{2^7} - 0.66 = -0.00375$$

จะเห็นได้ว่ากรณีที่ $M = 7$ จะสามารถแทนตัวเลขได้ในช่วงที่กว้างกว่าคือแทนได้ตั้งแต่ 0 ถึง 2 แต่กรณี $M = 8$ แทนได้เพียง 0 ถึง 1 แต่ทั้งสองกรณีถือว่ามีความละเอียดในการแทนค่าเท่ากัน เพราะใช้จำนวนบิตรวมเท่ากัน กล่าวคือ ถึงแม้กรณี $M = 7$ จะแทนค่าได้ในช่วงที่กว้างกว่าเท่าตัว การเลือกว่า จะใช้ M เท่าใดจึงจะเหมาะกับข้อมูลชุดหนึ่ง ๆ ก็ขึ้นอยู่กับว่าค่าต่ำสุดและสูงสุดของข้อมูลเป็นเท่าใด

ขนาดของขั้นจะมีค่าเท่ากับค่าของบิตขวาสุด (Least Significant Bit :LSB) ซึ่งก็คือ 2^{-M} และความคลาดเคลื่อนที่เกิดขึ้นในกรณีของการปัดเศษโดยวิธี rounding ก็จะมีขอบเขตไม่เกินครึ่งหนึ่งของขั้น นั่นคือ

$$-\text{ขนาดขั้น}/2 < err < \text{ขนาดขั้น}/2$$

$$-2^{-(M+1)} < err < 2^{-(M+1)}$$

ดังนั้นกรณี $M = 8$ จะแทนค่าได้ในช่วง $0 \le a < 1$ และ err ไม่เกิน $\pm 2^{-9}$ และกรณี $M = 7$ จะแทนค่าได้ในช่วง $0 \le a < 2$ และ err ไม่เกิน $\pm 2^{-8}$

โดยทั่วไปจะมีรูปแบบการแทนจำนวนบวกและจำนวนลบอยู่สองวิธีคือ

2.3.2 ระบบเลขจำนวนเต็มแบบมีเครื่องหมาย

ระบบเลขจำนวนเต็มแบบมีเครื่องหมายมีอยู่สองรูปแบบใหญ่ ๆ คือ

2.3.2.1 แบบ sign-magnitude

จะมีรูปแบบเหมือนเลขบวกแต่เพิ่มบิตพิเศษขึ้นมาอีก 1 บิต (โดยทั่วไปจะให้เป็นบิตซ้ายสุด) เป็นบิตบอกเครื่องหมาย (sign bit) โดยที่เมื่อบิตเครื่องหมายเป็น 0 หมายถึงเลขบวก และถ้าเป็น 1 หมายถึงเลขลบ จะขอยกตัวอย่างเลขแบบ sign-magnitude ในกรณีที่ $N = 0$, และ $M = 7$ เมื่อรวมบิตเครื่องหมายอีก 1 บิตจะต้องใช้จำนวนบิตทั้งสิ้น 8 บิตดังนี้

เขียนเป็นสัญลักษณ์ คือ $X.XXXXXXX$

ค่าของแต่ละบิตคือ sign $2^{-1}2^{-2}2^{-3}2^{-4}2^{-5}2^{-6}2^{-7}$

ตัวอย่างเช่น

$01010100_2 = 84_{10}$ มีค่าเท่ากับ 0.65625

และ $11010100_2 = -84_{10}$ มีค่าเท่ากับ –0.65625 เป็นต้น

จะเห็นได้ว่ารูปแบบ sign-magnitude สามารถเข้าใจได้ง่าย โดยเพียงแค่เติมบิตเครื่องหมายลงไปข้างหน้ารูปแบบปกติเท่านั้น อย่างไรก็ตามรูปแบบนี้ไม่เป็นที่นิยมในการใช้ เนื่องจากการบวกลบเลขทำได้ไม่สะดวกและไม่ดีเท่ารูปแบบ 2's complement ซึ่งจะได้กล่าวต่อไป

2.3.2.2 แบบ 2's complement

เป็นแบบที่นิยมใช้กันมากที่สุด โดยจะกำหนดให้บิตซ้ายมือสุดมีค่าเป็น -2^{N-1} ซึ่งเป็นบิตเดียวที่มีค่าติดลบ ส่วนบิตอื่น ๆ ยังมีความหมายเหมือนเดิม ยกตัวอย่างเช่น เลข 2's complement ขนาด 8 บิตที่มี $N = 1, 2$ และ 8 ไล่ตั้งแต่ค่าลบไปค่าบวก ขอให้สังเกตว่า ขนาดของขั้นในแต่ละกรณี ยังคงมีค่าเท่ากับบิตซ้ายมือสุดคือ 2^{-M} เหมือนกับกรณีการแทนเลขบวกอย่างเดียว ดังในตารางที่ 2.1

ตารางที่ 2.1 รูปแบบเลข 2's complement กรณี 8 บิต กับค่าที่แทนได้

ฐานสอง	ฐานสิบ (บวก)	ค่าที่แทน (กรณี $N = 8$)	ค่าที่แทน (กรณี $N = 1$)	ค่าที่แทน (กรณี $N = 2$)
1000 0000	128	-128	-1(ค่าลบมากที่สุด)	-2
1000 0001	129	-127	$-1+2^{-7} = -0.9921875$	$-2+2^{-6} = -1.984375$
1000 0010	130	-126	$-1+2(2^{-7}) = -0.984375$	$-2+2(2^{-6}) = -1.96875$
...	...	...	...	...
1111 1111	255	-1	$-2^{-7} = -0.0078125$	-2^{-6}
0000 0000	0	0	0	0
0000 0001	1	1	$2^{-7} = 0.0078125$	2^{-6}
0000 0010	2	2	$2(2^{-7}) = 0.015625$	$2(2^{-6})$
...	...	...	...	...
0111 1111	127	127	$1-2^{-7} = 0.9921875$ (ค่าบวกมากที่สุด)	$2-2^{-6} = 1.96875$

จะเห็นได้ว่า บิตซ้ายมือสุดยังสามารถมองเห็นเป็นบิตเครื่องหมายได้ เพราะถ้าบิตซ้ายมือสุดเป็น 1 จะได้เลขจำนวนนั้นเป็นค่าลบแน่นอน และถ้าเป็น 0 ก็จะได้เป็นค่าบวกแน่นอน

ในระบบประมวลผลสัญญาณ นิยมใช้รูปแบบ 2's complement ที่มีจุดทศนิยมหลังบิตซ้ายมือสุด 1 บิต ($N = 1$) ซึ่งจะแทนเลขได้จาก –1 ถึงประมาณ 1 และผลดีก็คือการคูณเลขสองจำนวนเข้าด้วยกันจะไม่เกิดโอเวอร์โฟลเพราะผลลัพธ์ที่ได้จะไม่มีทางเกินช่วง –1 ถึง 1 ยกตัวอย่าง

เช่น ต้องการแปลงเลข –0.595 เป็นจำนวนเต็มแบบ 2's complement 8 บิตโดยใช้ $M = 7$ สามารถทำได้ดังนี้

ถ้าให้ a คือค่าที่ต้องการจะแทน และ $-2^{N-1} \le a < 2^{N-1}$ สมมติให้ a_{fix} คือค่าที่ได้หลังจากเปลี่ยนเป็นรูปแบบจำนวนเต็มโดยที่ $M = 7$ และ $N = 1$ แล้วจะได้ว่า

$$\begin{aligned} a_{fix} &= round(a \times 2^{M}) \\ &= round(-0.595 \times 2^{7}) \\ &= round(-76.16) \\ &= -76 \end{aligned}$$

ค่า –76 สามารถแปลงกลับไปเป็นทศนิยมได้เป็น $-76/2^7 = -0.59375$ ซึ่งมีความคลาดเคลื่อนจาก a เท่ากับ

$$\frac{a_{fix}}{2^7} - a = -0.59375 - (-0.595) = 0.00125$$

ซอฟท์แวร์ส่วนใหญ่สามารถรับค่าจำนวนเต็มฐานสิบ (ไม่ว่าจะเป็นบวกหรือลบ) นี้ไปใช้งานต่อไปได้ โดยไม่จำเป็นต้องแปลงเป็นฐานสองให้ยุ่งยาก ซึ่งสำหรับการทดลองที่เวลาจริง ก็ใช้วิธีการดังกล่าวนี้ สำหรับกรณีที่ต้องการแปลงเป็นเลขฐานสองก็สามารถทำได้หลายวิธีดังนี้

วิธีที่ 1 แยกเลขเป็นส่วนลบและบวก จะได้ส่วนลบเป็นบิตซ้ายบิตเดียวมีค่า -2^{-B+1} เสมอ โดยที่ B เป็นจำนวนบิตทั้งหมด และส่วนบวกก็แปลงตามปกติ เช่น a_{fix} = -76 แตกเป็นส่วนลบกับส่วนบวกได้เป็น $a_{fix} = -128+52 = 10000000_2 + 00110100_2 = 10110100_2$

วิธีที่ 2 ถ้าเป็นเลขบวกให้แปลงตามปกติ แต่ถ้าเป็นเลขลบให้แปลงเป็นเลขบวกโดยการบวก 2^B เข้าไป ซึ่งมันจะทำหน้าที่เปลี่ยนความหมายของบิตซ้ายสุด จาก -2^{-B+1} เป็น 2^{-B+1} เช่นในที่นี้ a_{fix} = -76 แปลงเป็นเลขบวกโดยบวกด้วย 2^8 จะได้ $a_{fix} = -76+2^8 = 180 = 10110100_2$

จะสามารถสรุปข้อดีของ 2's complement ได้ดังนี้คือ

1) การบวกเลขทำได้เหมือนการบวกปกติโดยไม่ต้องคำนึงถึงว่าจะเป็นจำนวนบวก หรือลบ ดังตัวอย่างโดยใช้เลข 2's complement ขนาด 4 บิต และ $M = 0$ (ไม่มีส่วนทศนิยม) จะสามารถแทนค่าได้ในช่วง –8 ถึง 7 ตัวอย่างเช่น 3+(-5) จะได้ว่า

$$\begin{aligned} 0011_{+} &\Rightarrow 3_{+} \\ \underline{1011} &\Rightarrow \underline{-5} \\ \underline{\underline{1110}} &\Rightarrow \underline{\underline{-2}} \end{aligned}$$

จะเห็นได้ว่าการบวกเลขฐานสองที่เกิดขึ้น ทำเหมือนปกติ โดยไม่ต้องสนใจว่าจะเป็นเลขลบหรือเลขบวก ซึ่งจะได้ผลลัพธ์ถูกต้องเอง อีกตัวอย่างคือ –2+(-5) จะได้ว่า

$$\begin{aligned} 1110_{+} &\Rightarrow -2_{+} \\ \underline{1011} &\Rightarrow \underline{-5} \\ \underline{\underline{11110}} &\Rightarrow \underline{\underline{-7}} \end{aligned}$$

จากผลลัพธ์จะเห็นว่าเกิดบิตแรกเพิ่มขึ้นมา ซึ่งให้ตัดทิ้งได้ซึ่งในกรณีนี้ไม่ถือว่าโอเวอร์โฟล และอีกตัวอย่างคือ –4+(-5) จะได้ว่า

$$\begin{aligned} 1100_{+} &\Rightarrow -4_{+} \\ \underline{1011} &\Rightarrow \underline{-5} \\ \underline{\underline{10111}} &\Rightarrow \underline{\underline{7}} \end{aligned}$$

จะเห็นว่าได้ผลลัพธ์ผิด (ตัดบิตที่ 1 ทิ้ง) ผลลัพธ์ที่ถูกต้องคือ –9 ซึ่งเกินช่วงที่จะแทนค่าได้ และถือเป็นการเกิด โอเวอร์โฟล การตรวจสอบว่า เกิดโอเวอร์โฟลหรือไม่ อาจทำได้โดยตรวจสอบที่บิตเครื่องหมาย ถ้าจำนวนลบ บวก กับจำนวนลบ แล้วได้เป็นจำนวนบวก เหมือนในตัวอย่าง หรือจำนวนบวก บวก กับจำนวนบวก แล้วได้จำนวนลบ ถือว่าเกิดโอเวอร์โฟลขึ้น

2) การบวกเลขหลาย ๆ จำนวนไม่ต้องคำนึงถึงการเกิดโอเวอร์โฟลในผลลัพธ์กึ่งกลาง ถ้าผลลัพธ์สุดท้ายไม่เกิดโอเวอร์โฟล ตัวอย่างเช่น 4+5-3 โดยใช้ขนาด 4 บิต และ $M = 0$ ดังนี้คือ

$$\begin{aligned} 0100_{+} &\Rightarrow 4_{+} \\ 0101 &\Rightarrow \underline{5} \\ 1001_{+} &\Rightarrow -7_{+} \\ \underline{1101} &\Rightarrow \underline{-3} \\ \underline{\underline{10110}} &\Rightarrow \underline{\underline{6}} \end{aligned}$$

จะเห็นว่าได้ผลลัพธ์ตรงกึ่งกลางผิดแต่ผลลัพธ์สุดท้ายถูก ทั้งนี้เนื่องจาก โอเวอร์โฟลทางบวกและโอเวอร์โฟลทางลบชดเชยกันหายไปทำให้ได้ค่าผลลัพธ์ที่ถูกต้อง คุณสมบัติข้อนี้ของตัวเลข 2's complement มีผลดีมากต่อการประมวลผลสัญญาณ เพราะมีการประมวลผลหลาย ๆ อย่างที่ใช้การบวกสะสมค่าเพื่อหาค่าผลลัพธ์สุดท้าย เช่นการหาผลตอบของตัวกรอง FIR เป็นต้น

นอกจากเลขระบบจำนวนเต็มแล้ว ยังมีระบบเลขอ้างอิงดรรชนี (Floating-Point Number System) ซึ่งเป็นระบบเลขที่ตำแหน่งจุดทศนิยมลอยตัว คือ ตำแหน่งจุดทศนิยมไม่ติดอยู่ที่ตำแหน่งเดียวเหมือนอย่างระบบเลขจำนวนเต็ม (fixed-point) ซึ่งระบบเลขแบบอ้างอิงดรรชนีนี้มีข้อดีกว่าเนื่องจากมีการแบ่งขั้นที่ฉลาดกว่า เนื่องจากมีการปรับนัยสำคัญของเลขที่ใช้แทนให้เหมาะสมตามค่าขนาดของตัวเลข กล่าวคือ ที่ค่าของตัวเลขมาก ๆ ขนาดของขั้นจะกว้าง ส่วนที่ค่าตัวเลขน้อย ๆ ขนาดของขั้นจะลดลง การทำเช่นนี้ ทำให้ระบบเลขอ้างอิงดรรชนีที่จำนวนบิตมากพอสมควร จะมีความสามารถในการแทนตัวเลขได้ในช่วงที่กว้างกว่าระบบเลขจำนวนเต็มมาก และแทนได้แม่นยำกว่า เพราะมีนัยสำคัญที่เหมาะสมกับขนาดของตัวเลข และให้ค่าความคลาดเคลื่อนที่ต่ำกว่าด้วย รายละเอียดสามารถดูได้จาก [33]

อย่างไรก็ตามระบบเลขอ้างอิงดรรชนีมีข้อเสียในด้านต้นทุนที่แพงกว่า เพราะตัวกรองที่ใช้ในการประมวลผลของ เลขอิงดรรชนีทำได้ยากกว่าเลขจำนวนเต็ม นอกจากนี้ ตัวกรองของระบบเลขดรรชนียังใหญ่กว่า, ทำงานได้ช้ากว่า, และกินไฟมากกว่า ระบบเลขอิงดรรชนีจะทำงานได้ดีมากในระบบที่สัญญาณมีค่าสูง ๆ และ ต่ำ ๆ

2.4 ความคลาดเคลื่อนจากการใช้ระบบจำนวนเต็ม

ในหัวข้อนี้จะกล่าวถึงความคลาดเคลื่อนในระบบ เมื่อนำเอาระบบเลขจำนวนเต็มมาใช้แทนค่าสัญญาณ และมาใช้ในการประมวลผล ความคลาดเคลื่อนเหล่านี้ เกิดจากการแบ่งขั้นสัญญาณ, การปัดเศษสัมประสิทธิ์, โอเวอร์โฟล, และการปัดเศษหลังการคูณ การวิเคราะห์ในบางส่วนจะมีการใช้ทฤษฎีของตัวแปรสุ่ม (random variable) บ้าง

2.4.1 การแบ่งขั้นของสัญญาณ (Signal Quantization)

กระบวนการแรกที่เกิดความคลาดเคลื่อนขึ้นในระบบคือ การแบ่งขั้นของสัญญาณซึ่งกระทำหลังจากการสุ่มสัญญาณ การแบ่งขั้นสัญญาณ หมายถึง การแทนค่าสัญญาณที่ถูกสุ่มมาซึ่งความละเอียดไม่จำกัด (เพราะมาจากระดับสัญญาณแอนะลอก) ด้วยระบบเลขฐานสองที่มีจำนวนบิตจำกัด ซึ่งก็จะเกิดความคลาดเคลื่อนจากการแทนค่าขึ้น การแบ่งขั้นสัญญาณในทางปฏิบัติจะรวมอยู่ในตัวแปลงแอนะลอกเป็นดิจิตอล (A/D) แต่ในทางทฤษฎีจะแยกการแปลงสัญญาณแอนะลอก

และดิจิตอลออกเป็นสองกระบวนการ คือ การสุ่มซึ่งทำให้ได้สัญญาณไม่ต่อเนื่อง และการแบ่งขั้นสัญญาณซึ่งทำให้ได้สัญญาณที่มีความละเอียดทางขนาดจำกัดลง

ในที่นี้ จะขออธิบายเฉพาะกรณีที่ใช้ระบบเลขจำนวนเต็ม (fixed-point) ซึ่งจะทำให้ขนาดของขั้นคงที่ตลอดช่วงที่แทนค่าสัญญาณ การใช้จำนวนบิตที่มาก ย่อมทำให้ขั้นที่แบ่งมีขนาดเล็กลง มีความละเอียดมากขึ้น และแทนสัญญาณได้ถูกต้องมากขึ้น ถ้าให้ B คือจำนวนบิตที่ใช้แทน 1 ค่าของสัญญาณ, R เป็นช่วงของค่าสัญญาณที่สามารถแทนได้, และ Q แทนความกว้างของขั้น จะได้ว่า

$$\text{จำนวนขั้นทั้งหมด} = \frac{R}{Q} = 2^B \tag{2.18}$$

สัญญาณก่อนที่จะแบ่งขั้น จะต้องมีค่าอยู่ในช่วง $-R/2$ ถึง $R/2$ ถ้าพิจารณาตัวอย่างที่จุดหนึ่ง ๆ คือ $x(n)$ ซึ่งหลังจากถูกแบ่งขั้นแล้วจะมีค่าเปลี่ยนไป เพื่อให้ตรงกับขั้นที่แทนค่าได้ สมมติได้ค่าใหม่เป็น $x_q(n)$ ถ้าใช้กฎเกณฑ์การปัดเศษหาขั้นที่ใกล้ที่สุด (rounding) จะได้ว่า $x(n)$ ที่มีค่าอยู่ระหว่าง $x_q(n) - Q/2$ ถึง $x_q(n) + Q/2$ จะถูกปัดเป็นค่า $x_q(n)$ และ $x(n)$ มีค่าไม่เกิน $Q/2$ หรือ ครึ่งหนึ่งของขั้น

จะนิยาม $e(n)$ คือความคลาดเคลื่อนที่เกิดขึ้นจากการแบ่งขั้นสัญญาณ จะได้

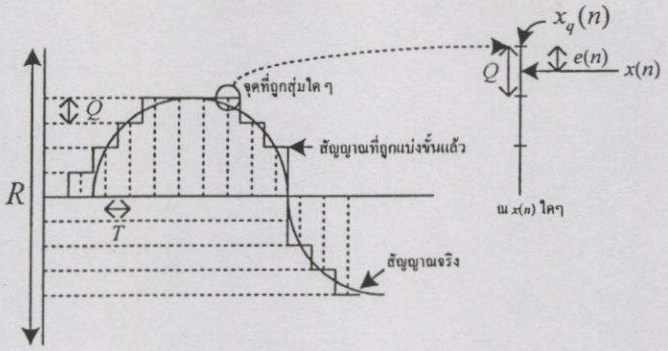
$$e(n) = x_q(n) - x(n) \tag{2.19}$$

โดยที่

$$-Q/2 < e(n) < Q/2 \tag{2.20}$$

สมการข้างบน เป็นค่าความคลาดเคลื่อนที่คิดทุก ๆ ของสัญญาณ และ Q มีค่าตามสมการที่ (2.18) โดยจะมีหน่วยเป็นแรงดันของสัญญาณแอนะลอก

ถ้าสมมติให้ $x_q(n)$ ซึ่งเป็นสัญญาณหลังจากที่ถูกแทนค่าด้วยจำนวนเต็ม อาจจะมองได้ว่าความคลาดเคลื่อนที่เกิดขึ้นกับ $x_q(n)$ เป็นเหมือนกับสัญญาณรบกวนที่ถูกเติมเข้าในสัญญาณเริ่มแรก นั่นคือ $x_q(n) = x(n) + e(n)$ ดังนั้น $e(n)$ จึงมีชื่อเรียกอีกอย่างหนึ่งว่า สัญญาณรบกวนที่เกิดจากการแบ่งขั้นสัญญาณ (quantization noise) รูปที่ 2.4 แสดงการแบ่งขั้นของสัญญาณ



รูปที่ 2.4 การแบ่งขั้นสัญญาณ (เส้นประในแนวตั้งคือตำแหน่งที่ถูกสุ่ม)

ฟังก์ชัน $e(n)$ เป็นฟังก์ชันที่มีค่าไม่แน่นอน (random) ชนิดหนึ่ง เพราะไม่สามารถรู้ล่วงหน้า ได้ว่า แต่ละค่าของมันจะมีค่าเท่าใดแน่ แต่สิ่งที่ทราบ คือ ความคลาดเคลื่อนที่เกิดขึ้นอยู่ระหว่าง $-Q/2$ ถึง $Q/2$ เท่านั้น เพื่อให้ง่ายต่อการวิเคราะห์ จะสมมติให้ค่าของ $e(n)$ มีการกระจายตัวสม่ำเสมอ (uniform distribution) ในช่วงค่าระหว่าง $-Q/2$ ถึง $Q/2$ หรือมีฟังก์ชันการกระจายตัวแบบโอกาส (probability density function) เป็น

$$p(e)=\begin{cases}1/Q, & -Q/2<e<Q/2\\ 0, & e=others\end{cases} \tag{2.21}$$

จากทฤษฎีความน่าจะเป็น จะได้ว่าค่าเฉลี่ยของ e มีค่าเท่ากับ 0 และค่าเฉลี่ยกำลังสองของ e มีค่าเท่ากับ $Q^2/12$ ซึ่งหาได้จาก

$$P_q=\int_{-Q/2}^{Q/2} e^2 p(e)de=\frac{Q^2}{12} \tag{2.22}$$

ค่าเฉลี่ยของกำลังสองของสัญญาณนี้ ก็คือ ค่ากำลังเฉลี่ยนั่นเอง ดังนั้น จึงใช้สัญลักษณ์แทนค่านี้ว่า P_q ซึ่งในที่นี้ จะสามารถหาค่ากำลังเฉลี่ยของสัญญาณรบกวนจากการแบ่งขั้นสัญญาณได้เท่ากับ $Q^2/12$ สังเกตว่า กำลังของสัญญาณรบกวนจะแปรตาม Q^2 ซึ่งแปรผกผันกับจำนวนบิต

ถ้ากำลังเฉลี่ยของสัญญาณมีค่าเท่ากับ P_s จะสามารถหาค่าอัตราส่วนระหว่างกำลังของสัญญาณต่อกำลังของสัญญาณรบกวน หรือ SNR ที่เนื่องจากสัญญาณรบกวนของการแบ่งขั้นสัญญาณได้ คือ

$$SNR_{A/D}=P_s/P_q$$

หรือ

$$SNR_{A/D}(dB) = P_s(dB) - P_q(dB) \tag{2.23}$$

P_s อาจหาได้จาก v_{rms}^2 ซึ่งค่า v_{rms} ของสัญญาณหนึ่ง ๆ นั้นขึ้นกับลักษณะรูปร่างของสัญญาณในการวิเคราะห์ต่อไป และอาจนิยามค่าคุณลักษณะของสัญญาณค่าหนึ่ง เพื่อบ่งบอกถึงค่า v_{rms} ของสัญญาณ ซึ่ง Loading factor หรือ LF ซึ่งเป็นค่าอัตราส่วนระหว่าง v_{rms} ต่อ v_p (Peak Voltage, ค่าสูงสุดของสัญญาณ) ของสัญญาณหนึ่ง ๆ นั่นคือ

$$LF = \frac{v_{rms}}{v_p} \tag{2.24}$$

สัญญาณที่มีระดับสัญญาณสูงสุดอยู่ห่างจากระดับสัญญาณ ณ เวลาอื่น ๆ มาก จะมีค่า LF ที่ต่ำกว่าสัญญาณที่มีระดับสัญญาณสูงสุดอยู่พอ ๆ กับระดับสัญญาณ ณ เวลาอื่น ๆ เช่นสัญญาณซายน์มี $v_{rms} = \frac{v_p}{\sqrt{2}}$ ดังนั้นจะมี $LF = \frac{1}{\sqrt{2}} = 0.707$ ส่วนสัญญาณสี่เหลี่ยมมี $v_{rms} = v_p$ ดังนั้น $LF = 1$ ซึ่งเป็นค่าที่มากที่สุดเท่าที่จะเป็นไปได้ สำหรับสัญญาณใด ๆ สำหรับสัญญาณเสียงพูด มีค่า $LF \approx 0.15 - 0.2$ (จากการทดลอง) เป็นต้น

จากนิยามของ LF จะสามารถหาค่ากำลังของสัญญาณได้เป็น

$$p_s = v_{rms}^2 = (LF)^2 v_p^2 \tag{2.25}$$

สมมติว่าทราบค่าสูงสุดของสัญญาณ คือ รู้ว่าสัญญาณมีค่าอยู่ระหว่าง $-v_p$ ถึง $+v_p$ และสามารถเลือกค่าในช่วงนี้เป็นช่วงที่ตัวแปลงแอนะลอกเป็นดิจิตอลสามารถแทนค่าได้พอดี นั่นคือค่า R ในรูปที่ 2.4 มีค่าเท่ากับ $2v_p$ เมื่อแทนค่า R ลงในสมการที่ (2.18) จะได้ว่า ขนาดของ 1 ขั้นของสัญญาณมีค่าเป็น

$$Q = \frac{2v_p}{2^B} \tag{2.26}$$

ดังนั้นได้ SNR อันเนื่องจากการแบ่งขั้นสัญญาณ มีค่าเป็น

$$SNR_{A/D} = \frac{P_s}{P_q} = \frac{(LF)^2 v_p^2}{\left(\frac{2v_p}{2^B}\right)^2 / 12}$$

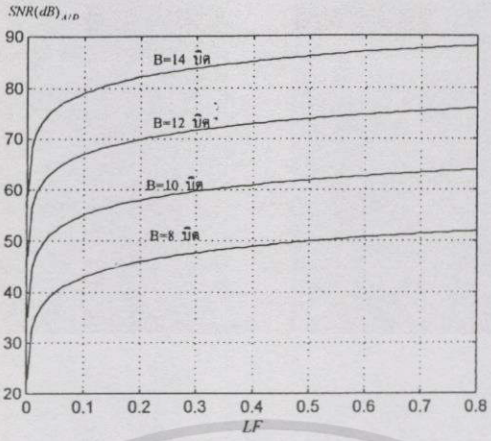
$$= 3(LF)^2 (2^{2B})$$

หรือ

$$SNR(dB)_{A/D} = 10\log\{3(LF)^2(2^{2B})\}$$
$$= 6.02B + 4.77 + 20\log(LF) \qquad (2.27)$$

ข้อสังเกตและพิจารณาสำหรับค่า $SNR_{A/D}$ ตามสมการที่ (2.27) นี้ คือ

1) $SNR_{A/D}$ มีค่าแปรตามจำนวนบิต (B) โดย $SNR_{A/D}$ จะมีค่ามากขึ้น 6 dB ต่อ 1 บิตที่เพิ่มขึ้น
2) $SNR_{A/D}$ ยังแปรตาม LF ดังแสดงในรูปที่ 2.5
3) ค่า $SNR_{A/D}$ ที่ได้นี้เป็นค่าที่มากเกินกว่าที่จะทำได้ในทางปฏิบัติ เนื่องจากในการหาค่า $SNR_{A/D}$ นี้ จะสมมติให้ค่าสูงสุดของสัญญาณ เป็นค่าสูงสุดที่ A/D สามารถแปลงค่าได้พอดี แต่ในทางปฏิบัติจะไม่สามารถทราบค่าสูงสุดของสัญญาณได้ จึงมักจะเผื่อให้ค่าสูงสุดที่ A/D สามารถแปลงได้ มีค่ามากกว่าช่วงที่จะใช้งานเล็กน้อย นอกจากนี้ ภายในตัวกรอง A/D เองในทางปฏิบัติก็จะมีความคลาดเคลื่อนในการแปลงค่าด้วย ซึ่งก็จะส่งผลให้มีสัญญาณรบกวนอันเนื่องมาจากความคลาดเคลื่อนของตัวกรอง A/D เองปนเข้ามาอีก
4) การเลือกว่า จะใช้ข้อมูลดิจิตอลกี่บิตแทน 1 ค่าสัญญาณ นอกจากขึ้นกับลักษณะของงานว่าสามารถยอมรับสัญญาณรบกวนได้ที่เท่าไรแล้ว ก็ยังขึ้นกับ SNR ของสัญญาณแอนะลอกอินพุตด้วย สัญญาณอินพุตโดยทั่วไปมักจะมีสัญญาณรบกวนปนมาจากแหล่งอื่นอยู่แล้ว ดังนั้น จะมีค่า SNR อยู่ค่าหนึ่ง ถ้าหากว่าใช้จำนวนบิตที่ให้ $SNR_{A/D}$ สูงกว่า SNR ของสัญญาณอินพุตมาก ๆ ก็อาจไม่มีประโยชน์มากนักเมื่อเทียบกับต้นทุนที่เพิ่มขึ้น เพราะไม่ว่าจำนวนบิตจะเป็นเท่าไรก็ตาม SNR ของสัญญาณเอาต์พุต ก็ยังถูกจำกัดด้วย SNR เริ่มต้นของสัญญาณอินพุต (ไม่มีทางได้ SNR ดีกว่า SNR อินพุต)



รูปที่ 2.5 ค่า $SNR_{A/D}$ ในฟังก์ชันของจำนวนบิต (B) และ loading factor (LF)

2.4.2 ความคลาดเคลื่อนจากการปัดเศษสัมประสิทธิ์ (Coefficient Rounding)

ความคลาดเคลื่อนจากการปัดเศษสัมประสิทธิ์ เกิดจากการที่นำค่าสัมประสิทธิ์ของระบบที่ออกแบบไว้ใช้ระบบเลขจำนวนเต็ม ซึ่งต้องแทนค่าสัมประสิทธิ์ด้วยจำนวนเต็มที่มีจำนวนบิตจำกัด (โดยทั่วไปใช้ 2's complement) ซึ่งก็แน่นอนว่าจะต้องเกิดความคลาดเคลื่อนกับค่าสัมประสิทธิ์เหล่านั้น การแทนค่าสัมประสิทธิ์ด้วยจำนวนเต็มนี้ เรียกอีกอย่างหนึ่งว่า การปัดเศษสัมประสิทธิ์ เพราะการปัดให้สัมประสิทธิ์ไปลงที่ค่าที่สามารถแทนค่าได้ด้วยจำนวนเต็ม

ความคลาดเคลื่อนที่เกิดขึ้นกับสัมประสิทธิ์ จะส่งผลถึงความคลาดเคลื่อนของลักษณะของระบบจากที่ได้ออกแบบไว้ ได้แก่ ผลตอบสนองเชิงความถี่มีรูปร่างเปลี่ยนไป, ความถี่ตัดมีค่าเปลี่ยนไป, ตำแหน่งของโพล และซีโร่เปลี่ยนไป จนถึงกระทั่งระบบเสียความมีเสถียรภาพไป ซึ่งผลดังกล่าวจะมีมากน้อยขึ้นอยู่กับ

1) จำนวนบิตที่ใช้ จำนวนบิตน้อยย่อมส่งผลกระทบมากขึ้น
2) ชนิดของตัวกรอง ผลของความคลาดเคลื่อนต่อตัวกรอง IIR จะมากกว่าตัวกรอง FIR มาก แต่สามารถแก้ได้โดยแตกตัวกรองที่มีอันดับสูง ให้อยู่ในโครงสร้างอนุกรม หรือขนานของตัวกรองอันดับสอง

โดยทั่วไปจะสามารถบอกได้เพียงว่าความคลาดเคลื่อนของสัมประสิทธิ์ทำให้คุณลักษณะของระบบเปลี่ยนไป แต่จะเปลี่ยนไปอย่างไร มากน้อยแค่ไหน และอยู่ในภาวะที่ยอมรับได้หรือไม่นั้น ยังไม่สามารถหาสูตรทั่วไปที่จะบอกได้ แนวทางที่ดีที่สุดก็คือ ลองปัดเศษสัมประสิทธิ์ดูเป็นเฉพาะกรณี ๆ ไป แล้วนำสัมประสิทธิ์ชุดใหม่ที่ได้หลังจากการปัดเศษไปตรวจสอบดู เช่น ถ้าเป็นตัวกรองดิจิตอลก็ตรวจดูว่าผลตอบสนองความถี่เปลี่ยนไปอย่างไร

สมมติ ในระบบมีสัมประสิทธิ์ค่าหนึ่งเท่ากับ a หากต้องการนำไปใช้กับตัวเลข 16 บิต (2's complement) ก่อนอื่นต้องเลือกตำแหน่งจุดทศนิยมของเลขจำนวนเต็มก่อน ในกรณีที่ค่าสัมประสิทธิ์ทุกค่ามีค่าอยู่ระหว่าง –1 ถึง 1 จะสามารถเลือกให้ $N = 1$, $M = 15$ ได้ แต่ถ้ามีสัมประสิทธิ์บางตัวที่มีค่าเกินช่วง –1 ถึง 1 ก็อาจต้องเลือกให้ N มากกว่า 1

หลังจากเลือกจำนวนบิต และจำนวนจุดทศนิยมแล้ว ก็สามารถหาค่าสัมประสิทธิ์ที่จะมีค่าเปลี่ยนไปได้ สมมติให้ $\hat{a}$ คือค่าของสัมประสิทธิ์หลังจากการปัดเศษ จะได้ว่า

$$\hat{a} = round(a \times 2^M) / 2^M \tag{2.28}$$

ซึ่งจะใช้สมการนี้กระทำกับสัมประสิทธิ์ทุกค่า และจะได้สัมประสิทธิ์ชุดใหม่ คือชุดหลังจากการปัดเศษ ซึ่งจะนำไปวิเคราะห์หาผลตอบสนองเชิงความถี่ หรือลักษณะอื่นๆ ต่อไป อย่าลืมว่าค่า $\hat{a}$ ที่ได้นี้เป็นค่าที่คำนวณขึ้นมาเพื่อไปใช้วิเคราะห์ผลของความคลาดเคลื่อนเท่านั้น แต่ค่าที่นำไปใช้จริงคือ $a_{fix} = round(a \times 2^M)$ ซึ่งเป็นค่าจำนวนเต็ม

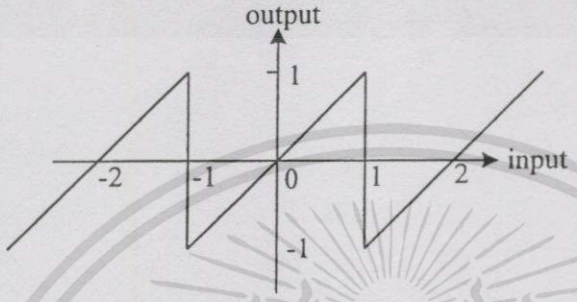
การวิเคราะห์ผลของความคลาดเคลื่อนจากการปัดเศษสัมประสิทธิ์เพียงอย่างเดียว มิได้เป็นหลักประกันร้อยเปอร์เซ็นว่า เมื่อนำไปใช้จะได้ระบบที่มีผลตอบสนองเชิงความถี่เปลี่ยนไปเหมือนที่ได้วิเคราะห์ไว้ โดยเฉพาะอย่างยิ่งถ้านำไปใช้ที่จำนวนบิตต่ำ ๆ ทั้งนี้เนื่องจาก ยังจะมีผลจากความคลาดเคลื่อนอื่นในการประมวลผลรวมเข้ามาอีกด้วย ซึ่งจะได้กล่าวถึงในหัวข้อถัดไป การวิเคราะห์ความคลาดเคลื่อนจากการปัดเศษเป็นการวิเคราะห์เบื้องต้น ซึ่งทำได้ง่ายและรวดเร็ว การวิเคราะห์โดยละเอียดอาจต้องทำโดยการจำลองระบบให้ประมวลผลด้วยระบบเลขจำนวนเต็ม ตามที่จะใช้จริง ๆ แล้วดูผลลัพธ์ว่าเป็นอย่างไร

2.4.3 ความคลาดเคลื่อนจากโอเวอร์โฟล (Overflow)

โอเวอร์โฟล คือ เหตุการณ์ที่ผลลัพธ์ของการประมวลผลมีค่าเกินช่วงที่สามารถแทนได้ โอเวอร์โฟลที่จะกล่าวถึงนี้คือ โอเวอร์โฟลที่เกิดจากผลลัพธ์จากการบวก จะยกตัวอย่างโอเวอร์โฟลที่เกิดจากการบวกเลข 2's complement สองจำนวนเข้าด้วยกัน ขนาด 8 บิต ที่มี $M = 0$ ซึ่งจะสามารถแทนค่าได้ในช่วง –128 ถึง 127 เช่น ต้องการหาผลลัพธ์ของ 100 + 87 ซึ่งผลลัพธ์ที่ได้คือ 187 ซึ่งเกินช่วงที่จะสามารถแทนได้ นั่นคือ

$$\begin{array}{ll} 01100100_{+} & \Rightarrow 100_{+} \\ \underline{01011011} & \Rightarrow \underline{87} \\ \underline{\underline{10111011}} & \Rightarrow \underline{\underline{-69}} \end{array}$$

ปรากฏว่าผลลัพธ์ที่ได้คือ –69 หรือมาจากผลลัพธ์ที่ถูกต้องลบด้วย 2^8 นั่นคือ 187-256 = -69 โดยทั่วไปโอเวอร์โฟลที่เกิดจากการบวกกันระหว่างเลข 2's complement ที่มีค่าเกินช่วงที่จะแทนได้จะเป็นดังรูปที่ 2.6 โดยค่าบวกที่อินพุตมาก ๆ จะทำให้เกิดค่าลบที่เอาต์พุตมาก ๆ หรือกลับกัน



รูปที่ 2.6 คุณลักษณะของโอเวอร์โฟลในการกระทำทางคณิตศาสตร์เลข 2's complement ในสภาวะที่อินพุตเกินช่วงที่แทนได้ (-1,1)

โอเวอร์โฟลทำให้ผลลัพธ์ที่ได้เปลี่ยนจากหน้ามือเป็นหลังมือ ไม่ได้เป็นเหมือนสัญญาณรบกวน หรือความเพี้ยนที่เกิดขึ้นเหมือนกับความคลาดเคลื่อนที่มาจากสาเหตุอื่น ๆ ดังนั้นโอเวอร์โฟล อาจทำให้สัญญาณเอาต์พุตเสียรูปร่างจนดูไม่รู้เรื่องไปเลยก็ได้ ถ้าหากปล่อยให้เกิดขึ้น

อย่างไรก็ตาม โอเวอร์โฟลเป็นความคลาดเคลื่อนที่สามารถป้องกันได้ โดย

1) ลดขนาดของสัญญาณอินพุต หรือใช้ตัวคูณลดทอนสัญญาณอินพุต หรือใช้วิธีเลื่อนบิตไปทางซ้าย (left shift) ซึ่งการเลื่อนบิตไปทางซ้าย 1 บิตเท่ากับการหารด้วย 2
2) สเกลค่าสัมประสิทธิ์ที่เป็นตัวคูณสัญญาณไปข้างหน้าลง (สัมประสิทธิ์ที่ไม่ใช่ตัวคูณของสัญญาณป้อนกลับ) การดูว่าต้องลดค่าสัมประสิทธิ์ตัวใดบ้างนั้นขึ้นกับโครงสร้างของระบบ

การปรับลดขนาดของสัญญาณ หรือสัมประสิทธิ์ข้างต้น ถ้าลดมากเกินไปจะทำให้สัญญาณมีขนาดลดต่ำลงเกินความจำเป็น ซึ่งส่งผลให้ *SNR* ต่ำลง การพิจารณาว่าจะต้องลดค่าลงเท่าไร อาจทำได้โดยการทดลอง หรือจำลองใส่สัญญาณอินพุตที่จะใช้จริงดู หรืออาจประมาณค่าที่ต้องลดล่วงหน้าโดยทำการวิเคราะห์ก่อนก็ได้ ซึ่งรายละเอียดสามารถดูได้จาก [30-33]

2.4.4 ความคลาดเคลื่อนจากการปัดเศษหลังการคูณ (Product Rounding)

ก่อนอื่นขอทำความเข้าใจเรื่องการคูณในระบบเลขจำนวนเต็มก่อน และเหตุผลที่ว่าทำไมจึงต้องมีการปัดเศษหลังการคูณ สมมติว่ามีเลขบวกขนาด 5 บิต สองจำนวน ซึ่งใช้ $M = 0$ ดังนั้นแต่ละจำนวนสามารถแทนตัวเลขได้ระหว่าง 0 ถึง 32 ถ้านำเลขสองจำนวนนี้มาคูณกัน จะได้ผลลัพธ์ที่มีค่าระหว่าง 0 ถึง 1024 ซึ่งต้องใช้บิตในการแทนถึง 10 บิต ($M = 0$) ดังนี้

$$\begin{array}{r} XXXXX. \\ \times\ XXXXX. \\ \hline XXXXXXXXXX. \end{array}$$

ถ้าผลคูณจะต้องถูกนำเอาไปใช้คำนวณต่อไป เราต้องการให้ผลคูณมีขนาด 5 บิตเหมือนตัวตั้ง (มิฉะนั้นก็จะต้องใช้ตัวประมวลที่มีจำนวนบิตเพิ่มขึ้นเป็นสองเท่า) แต่ในกรณีนี้ทำไม่ได้ เพราะผลคูณจำเป็นต้องใช้จำนวนบิตในการแทนถึง 10 บิต การคูณในกรณีนี้ถือว่าเกิดโอเวอร์โฟลจากการคูณขึ้น ซึ่งทำให้ใช้งานไม่ได้ในระบบประมวลผล

ถ้าพิจารณากรณีที่ตัวคูณเป็นรูปแบบเลข 5 บิต ที่มี $M = 5$ และ $N = 0$ โดยผลลัพธ์ที่ได้เป็น 10 บิตเช่นเดิม แต่เป็น 10 บิตที่แทนค่าได้ในช่วง 0 ถึง 32 เท่านั้น เนื่องจากตัวตั้งมีค่า 0 ถึง 32 และตัวคูณมีค่าตั้งแต่ 0 ถึง 1 ดังนั้นผลคูณไม่มีทางได้ค่าเกิน 32 ซึ่งจะเป็นรูปแบบ 10 บิต ที่มี $M = 5$ และ $N = 5$ ดังนี้

$$\begin{array}{r} XXXXX. \\ \times\ .XXXXX \\ \hline XXXXX.XXXXX \end{array}$$

ดังนั้น ถ้าต้องการนำผลคูณนี้ไปใช้คำนวณต่อ ก็สามารถทำได้โดยการตัดเศษ (truncate) 5 บิตหลังทิ้ง หรือ ปัดเศษ (round) 5 บิตหลัง การตัดเศษเทียบเท่ากับการตัด 5 บิตหลังทิ้ง ซึ่งทำได้ง่ายกว่าการปัดเศษในการใช้งานจริง ซึ่งจะได้ผลลัพธ์สุดท้ายหลังการตัดเศษ หรือปัดเศษเป็นรูปแบบ 5 บิตที่มี $M = 5$ เหมือนตัวตั้งโดยไม่มีทางเกิดโอเวอร์โฟล และจะสามารถนำไปคำนวณต่อไปได้ โดยใช้ตัวประมวลผลที่มีจำนวนบิตเท่าเดิม สิ่งที่เสียไปก็ คือ นัยสำคัญหรือความละเอียดของผลลัพธ์ 5 บิตล่าง ซึ่งก็ทำให้เกิดความคลาดเคลื่อนขึ้น

ในกรณีที่ตัวคูณมีค่ามากกว่า 1 ด้วย เช่น $M = 3$ และ $N = 2$ ผลลัพธ์ที่ได้จะเป็นรูปแบบ 10 บิตที่มี $M = 3$ โดยจะต้องทำการปัดเศษ 3 บิตหลังทิ้ง จุดที่พิเศษขึ้นมา คือ กรณีนี้ผลลัพธ์มีโอกาสเกิดโอเวอร์โฟลได้ ถ้า 2 บิตหน้าของผลลัพธ์ไม่เท่ากับศูนย์ ดังนั้น ตัวตั้งและตัวคูณต้องมีขนาดไม่ใหญ่จนทำให้ผลลัพธ์มากเกินกว่าที่จะแทนค่าได้

$$\begin{array}{r} XXXXX._{\times} \\ XX.XXX \\ \hline XXXXXXX.XXX \end{array}$$

สรุปได้ว่า การคูณเลขจำนวนเต็ม ตัวตั้งจะมีจุดทศนิยมตรงไหนก็ได้ไม่สนใจ ส่วนตัวคูณถ้ามีจำนวนบิตอยู่หลังจุดทศนิยมเท่ากับ M บิต หลังจากคูณแล้วต้องปัดเศษผลลัพธ์ทิ้ง M บิต วิธีการจะทำให้ได้ผลลัพธ์หลังการปัดเศษถูกต้อง และเป็นรูปแบบเดียวกับตัวตั้ง

จะเห็นได้ว่า ทุกครั้งที่มีการคูณจะทำให้เกิดความคลาดเคลื่อนขึ้น โดยลักษณะความคลาดเคลื่อนของการปัดเศษหลังการคูณนี้ คล้ายคลึงกับความคลาดเคลื่อนจากการแบ่งขั้นสัญญาณ อาจมองได้ว่า การปัดเศษหลังคูณ คือ การที่ต้องการแทนค่าผลลัพธ์จากการคูณซึ่งมีความละเอียดมากด้วยจำนวนบิตที่จำกัดลง (เท่ากับจำนวนบิตของสัญญาณเดิม) สมมติว่าขนาดของ 1 ขั้น ของสัญญาณมีค่าเท่ากับ Q หน่วยเป็นแรงดัน มีค่าตามสมการ

$$Q = \frac{R}{2^B} \tag{2.29}$$

ถ้าให้ $y(n)$ เป็นสัญญาณที่เป็นผลลัพธ์ของการคูณก่อนปัดเศษ และ $y_e(n)$ เป็นผลลัพธ์หลังปัดเศษ อาจมองได้ว่า ความคลาดเคลื่อนที่เกิดขึ้นเป็นเหมือนสัญญาณรบกวนที่ถูกเติมเข้ามาในระบบ และเรียกมันว่าสัญญาณรบกวนจากการคูณ (multiplication noise) ขอใช้สัญลักษณ์แทนว่า $e_e(n)$ ซึ่งจะได้

$$e(n) = y_e(n) - y(n) \tag{2.30}$$

ในกรณีของการปัดเศษ (rounding) สัญญาณรบกวนนี้มีค่าไม่เกินครึ่งหนึ่งของขั้น หรือ

$$-Q/2 < e_{round}(n) < Q/2 \tag{2.31}$$

เช่นเดียวกันกับการวิเคราะห์สัญญาณรบกวนจากการแบ่งขั้นสัญญาณ ซึ่งจะได้ว่าค่ากำลังเฉลี่ย หรือค่าเฉลี่ยของกำลังสองของสัญญาณ คือ

$$P_{e,round} = \frac{Q^2}{12} \tag{2.32}$$

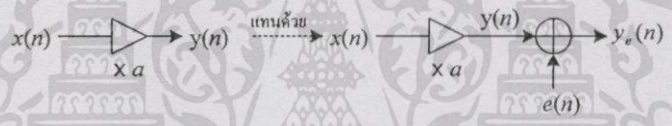
ในกรณีของการตัดเศษ (truncation) สัญญาณรบกวนนี้จะมีค่าเป็นลบเสมอ (ค่าจะถูกบังคับเป็นขั้นที่ต่ำลง) ดังนั้น

$$-Q < e_{trunc}(n) < 0 \tag{2.33}$$

และกำลังเฉลี่ยของสัญญาณรบกวนสำหรับกรณีนี้คือ

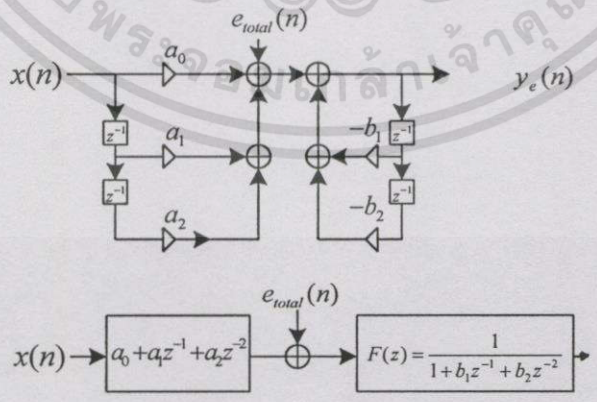
$$P_{e,trunc} = \frac{Q^2}{3} \tag{2.34}$$

ในการวิเคราะห์สัญญาณรบกวนจากการคูณ จะต้องแทนตัวคูณทุกตัวในระบบด้วยตัวคูณที่มีสัญญาณรบกวนปนที่เอาต์พุต ดังแสดงในรูปที่ 2.7 จากนั้น จึงหาผลรวมของสัญญาณรบกวนทั้งหมดที่จะปรากฏที่สัญญาณเอาต์พุตของระบบ



รูปที่ 2.7 การแทนตัวคูณด้วยตัวคูณที่มีสัญญาณรบกวนปนเพื่อใช้ในการวิเคราะห์

จะขอยกตัวอย่างการวิเคราะห์สัญญาณรบกวนจากการคูณในตัวกรอง IIR โครงสร้าง direct form I อันดับสองเพื่อเป็นแนวทาง แสดงดังรูปที่ 2.8



รูปที่ 2.8 โครงสร้างแบบ direct form I ที่ใช้ในการวิเคราะห์

จากรูปพบว่า มีตัวคูณอยู่ 5 ตัว ซึ่งสัญญาณรบกวนที่เกิดจากตัวคูณทุกตัวสามารถมารวมกันได้เป็น $e_{total}(n)$ โดยที่ $e_{total}(n)$ ที่เกิดขึ้นในโครงสร้าง direct form I นี้ ถึงแม้มองดูเหมือนเป็นสัญญาณรบกวนที่เกิดขึ้นที่เอาต์พุตเหมือนตัวกรองแบบ FIR แต่จริงแล้วไม่ใช่ เนื่องจากสัญญาณรบกวนนี้ถูกป้อนกลับเข้ามาในระบบอีก นี่เองที่เป็นเหตุผลที่ทำให้สัญญาณรบกวนจากการคูณมีผลต่อตัวกรอง IIR มากว่าตัวกรอง FIR

โดยจะทำการหาว่าสัญญาณรบกวนจากตัวคูณที่ปรากฏออกมาที่เอาต์พุตจริง ๆ รวมแล้วมีค่า เท่าไร จะสมมติให้ $y(n)$ คือสัญญาณเอาต์พุตขณะไม่มีสัญญาณรบกวนเกิดขึ้น และ $y_e(n)$ คือสัญญาณเอาต์พุตที่มีสัญญาณรบกวน ดังนี้

$$y_e(n) = y(n) + e_{total}(n) \tag{2.35}$$

สมการผลต่างของ $y_e(n)$ เขียนได้เป็น

$$y_e(n) = a_o x(n) + a_1 x(n-1) + a_2 x(n-2) - b_1 y_e(n-1) - b_2 y_e(n-2) + e_{total}(n) \tag{2.36}$$

ทำการแปลง z จะได้ว่า

$$Y_e(z)\left[1 + b_1 z^{-1} + b_2 z^{-2}\right] = X(z)\left[a_o + a_1 z^{-1} + a_2 z^{-2}\right] + E_{total}(z)$$

$$Y_e(z) = \frac{a_o + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}} X(z) + \frac{E_{total}(z)}{1 + b_1 z^{-1} + b_2 z^{-2}} \tag{2.37}$$

จะได้

$$E_{out}(z) \frac{E_{total}(z)}{1 + b_1 z^{-1} + b_2 z^{-2}} = F(z) E_{total}(z) \tag{2.38}$$

นั่นคือ สัญญาณรบกวนที่เกิดขึ้น จะผ่านฟังก์ชัน $F(z)$ ออกมาที่สัญญาณเอาต์พุตตามรูปที่ 2.8 สิ่งที่สนใจก็คือ กำลังของสัญญาณ $e_{out}(n)$ ว่าเป็นเท่าไร มีกฎว่า ถ้าสัญญาณอินพุตของระบบหนึ่งเป็นสัญญาณรบกวนขาว (white noise) กำลังของสัญญาณเอาต์พุตจะมีค่าเท่ากับกำลังของสัญญาณอินพุตคูณด้วยผลบวกกำลังสองของผลตอบสนองต่ออิมพัลส์ของระบบ โดยจะสามารถ

พิจารณาว่าสัญญาณ $e_{total}(n)$ มีลักษณะเป็นสัญญาณรบกวนขาว ซึ่งจะได้กำลังของสัญญาณรบกวนที่เอาต์พุตเป็น

$$P_{e,out} = P_{e,total} \sum_{n=0}^{\infty} f^2(n) \tag{2.39}$$

โดย $f(n)$ คือผลตอบสนองอิมพัลส์ของ $F(z)$ ถ้าสัมประสิทธิ์ทุกตัวไม่เท่ากับศูนย์ จะได้กำลังรวมของสัญญาณรบกวนเป็น $P_{e,total} = 5P_e$ ซึ่งจะได้กำลังของสัญญาณรบกวนที่เอาต์พุตเป็น

$$P_{e,out} = 5P_e \sum_{n=0}^{\infty} f^2(n) \tag{2.40}$$

โดยที่ $P_e = Q^2/12$ (สำหรับสัญญาณรบกวนที่เกิดจากการปัดเศษ) จะขอยกตัวอย่างผลการศึกษาเลขจำนวนเต็มขนาด 8 บิต กับตัวกรอง LPF บัตเตอร์เวิร์ธ ที่มีความถี่ตัดที่ 0.6π โดยเปรียบเทียบผลตอบสนองอิมพัลส์ และสเปกตรัมของมัน

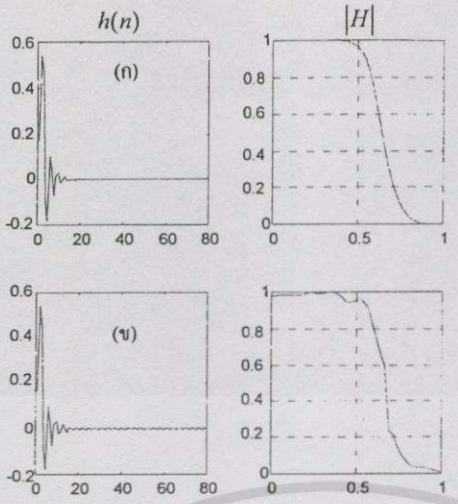
ทำการหาสัมประสิทธิ์ของตัวกรองโดยใช้ฟังก์ชันสำเร็จรูปใน DSP Toolbox ชื่อ butter ในโปรแกรม MATLAB ดังนี้

[a,b] = butter(4,0.6) ซึ่งจะได้สัมประสิทธิ์ของ LPF อันดับ 4 และความถี่ตัด 0.6π คือ

a = [0.1672 0.6687 1.0031 0.6687 0.1672]

b = [1.0000 0.7821 0.6800 0.1827 0.0301]

ผลลัพธ์ที่ได้จะไม่มีความคลาดเคลื่อนเนื่องจาก MATLAB ใช้ตัวเลขอิงดรรชนีในการคำนวณ ผลตอบสนองแสดงดังในรูปที่ 2.9 ก



รูปที่ 2.9 ผลตอบสนองอิมพัลส์ และสเปกตรัมของมัน (ก) เมื่อไม่มีความคลาดเคลื่อน (ข) เมื่อใช้กับระบบเลข 8 บิต

ในกรณีที่ใช้ระบบเลข 8 บิต จากสัมประสิทธิ์ข้างต้นพบว่า ทุกตัวมีขนาดไม่เกิน 1 ยกเว้น a(3) = 1.0031 ซึ่งจะอนุโลมปัดให้ a(3) = 1 ดังนั้น สามารถเลือกให้ $M = 7$ และ $N = 1$ โดยใช้โปรแกรม MATLAB คำนวณเป็นแบบเลขจำนวนเต็ม จะได้ผลตามรูปที่ 2.9 (ข) ซึ่งจะสังเกตุเห็นถึงความผิดเพี้ยนที่เกิดขึ้นกับผลตอบสนองเชิงความถี่ของระบบ จุดที่ต้องระวัง ก็คือ สเปกตรัมขอบผลตอบสนองต่ออิมพัลส์ในกรณีนี้ จะไม่ใช่ผลตอบสนองเชิงความถี่ของระบบที่จะได้ ทั้งนี้เนื่องจากผลของสัญญาณรบกวนจากการคูณเป็นผลที่ไม่เป็นเชิงเส้น การเกิดผลทางความถี่ต่อผลตอบสนองต่ออิมพัลส์ในลักษณะหนึ่ง ไม่ได้หมายความว่า ผลนั้นจะเหมือนกันเมื่อสัญญาณอินพุตเป็นสัญญาณอื่น ดังนั้น รูป 2.9 (ข) จะใช้เปรียบเทียบกันเท่านั้น ไม่สามารถถือได้ว่าเป็นผลตอบสนองเชิงความถี่ของระบบที่เปลี่ยนไป ผลในรูป (ข) มีลักษณะไม่ถูกต้องคือ ผลตอบสนองไม่ลู่เข้าสู่ศูนย์ ทั้งที่สัญญาณอินพุตเป็นศูนย์ไปนานแล้ว โดยผลตอบสนองจะแกว่งเป็นคาบ มีค่า (ก่อนหารกลับเป็นทศนิยม) เป็น ..., 0, -1, 1, 0,-1, 1, 0, -1, 1, 0,... ปรากฏการณ์นี้เรียกว่า limit cycle oscillation ซึ่งเป็นผลจากความคลาดเคลื่อนจากการคูณที่เกิดขึ้นกับระบบที่มีการป้อนกลับในบางกรณีเท่านั้น

2.5 สรุป

ในบทนี้ได้กล่าวถึง ตัวกรองความถี่ดิจิตอลแบบ IIR และโครงสร้างแบบ Direct Form ซึ่งใช้ในวิทยานิพนธ์ โครงสร้างแบบ Direct Form แบ่งออกเป็นสองแบบคือ Direct Form I และ Direct Form II โครงสร้างแบบ Direct Form I จะมีโอกาสเกิดโอเวอร์โฟลน้อยกว่าโครงสร้างแบบ Direct Form II ทั้งนี้เพราะมีตัวบวกเพียงชุดเดียว แต่โครงสร้างแบบ Direct Form II ก็มีข้อดีตรงที่มีการคำนวณน้อยกว่าและใช้หน่วยความจำน้อยกว่า

การสร้างตัวกรองด้วยฮาร์ดแวร์หรือซอฟแวร์จะต้องคำนึงผลของ finite-word-length effect ที่เกิดขึ้นด้วย ทั้งนี้เนื่องจาก ข้อมูลที่ต้องการจะถูกแทนด้วยจำนวนบิตที่จำกัด ซึ่งทำให้ค่าความ แม่นยำลดน้อยลง ตัวกรองแบบ IIR จะมีข้อควรระวังในเรื่องของตำแหน่งโพล ซึ่งจะต้องบังคับให้อยู่ในวงกลมหนึ่งหน่วยในระนาบ z โดยทั่วไป ตัวกรองแบบ IIR อันดับสูง ๆ จะนิยมสร้างจากตัวกรองแบบ IIR อันดับสองต่ออนุกรมกัน ทั้งนี้เพื่อต้องการลดความคลาดเคลื่อนจากการคูณ ซึ่งมีผลดีตามมาคือ จะสามารถทำการทดสอบเสถียรภาพได้ง่าย โดยที่ตัวกรอง IIR อันดับสองจะบังคับให้โพลทุกตัวจะต้องอยู่ภายในสามเหลี่ยมเสถียรภาพ ซึ่งง่ายและรวดเร็วในการทดสอบ

การสร้างตัวกรองด้วยจำนวนบิตจำกัดทำให้เกิดระบบตัวเลขสองระบบคือ ระบบตัวเลขจำนวนเต็ม (fixed point number system) และระบบตัวเลขอิงดรรชนี (floating point number system) โดยทั่วไปเลขอิงดรรชนีจะมีข้อดีกว่าเลขแบบจำนวนเต็ม ทั้งนี้เพราะจะสามารถแทนค่าของจำนวนได้แม่นยำกว่า แต่มีข้อเสียคือ มีราคาแพง และทำงานได้ช้า นอกจากนี้โครงสร้างฮาร์ดแวร์จะมีขนาดใหญ่ด้วย การใช้ตัวเลขในการประมวลผลแบบจำนวนเต็มแม้ว่าจะมีข้อดีตรงที่ราคาถูก ทำงานได้เร็ว แต่ก็มีข้อเสียอันเนื่องมาจาก finite-word-length effect อยู่หลายประการด้วยกันพอสรุปได้ดังนี้คือ

1) เกิดความคลาดเคลื่อนจากการแบ่งขั้นของสัญญาณ (Signal Quantization)
2) โอเวอร์โฟลที่เกิดจากการบวก
3) สัญญาณรบกวนจากการปัดเศษหลังการคูณ
4) Limit cycles oscillation

ดังนั้นการสร้างตัวกรองความถี่โดยใช้โครงสร้างแบบต่าง ๆ จะต้องคำนึงถึงผลกระทบทั้งสี่ข้อนี้ด้วย ยกตัวอย่างเช่น การจะสร้างตัวกรอง IIR อันดับสูง ควรจะสร้างจากตัวกรอง IIR อันดับสอง และโครงสร้างที่เหมาะสมคือ แบบ Direct Form II แต่จะต้องระวังในเรื่องโอเวอร์โฟลด้วย นอกจากนี้ ผลกระทบจากการปัดเศษจะเกิดขึ้นในโครงสร้างแบบ FIR น้อยกว่าแบบ IIR ทั้งนี้เพราะโครงสร้างแบบ FIR มีการคำนวณไปข้างหน้าอย่างเดียว แต่โครงสร้างแบบ IIR จะมีการป้อนกลับ ซึ่งทำให้สัญญาณรบกวนถูกป้อนกลับเข้ามายังระบบอีกครั้งหนึ่ง และสุดท้าย Limit cycles oscillation จะเกิดขึ้นเฉพาะในตัวกรอง IIR อันเนื่องมาจากการควอนไตซ์ และการปัดเศษของการคูณของระบบเลขจำนวนเต็ม ซึ่งเป็นผลให้ผลตอบสนองของระบบเปลี่ยนแปลงไป

บทที่ 3
กระบวนการสุ่มแบบไม่ต่อเนื่องทางเวลา

ในบทนี้จะกล่าวถึง กระบวนการสุ่มแบบไม่ต่อเนื่องทางเวลา (Discrete-Time Random Processes) ซึ่งเป็นวิชาที่มีความสำคัญมากต่อการศึกษา การประมวลผลสัญญาณแบบอะแดปทีฟ (Adaptive Signal Processing) ทั้งนี้เพราะ การประมวลผลสัญญาณแบบอะแดปทีฟ จะเกี่ยวข้องกับสัญญาณแบบสุ่ม ทั้งที่เป็นแบบ strictly stationary, wide sense stationary และ แบบ non stationary การวิเคราะห์คุณสมบัติของสัญญาณแบบสุ่ม จะต้องอาศัยหลักการทางสถิติ โดยอาจจะอธิบายในรูปของ ค่าเฉลี่ยเชิงสถิติ เช่น ค่ากลาง (mean) ค่าความแปรปรวน (variance) และ ค่าความเกี่ยวพันธ์ (correlation) เป็นต้น

3.1 ชนิดของสัญญาณ

สัญญาณที่พบกันทั่วไปจะสามารถแบ่งออกได้เป็น 3 ประเภทคือ [20]

3.1.1 สัญญาณแบบ Deterministic

สัญญาณแบบ Deterministic เป็นสัญญาณที่สามารถอธิบายได้ด้วยสมการทางคณิตศาสตร์ ซึ่งแบ่งเป็น สัญญาณรายคาบ และ สัญญาณไม่รายคาบ สัญญาณรายคาบ เช่น สัญญาณรูปซายน์ ซึ่งสามารถกำหนดด้วยสมการคือ

$$x(t) = \sin(2\pi ft + \phi) \tag{3.1}$$

หรือ

$$x(n) = \sin(2\pi fn + \phi) \tag{3.2}$$

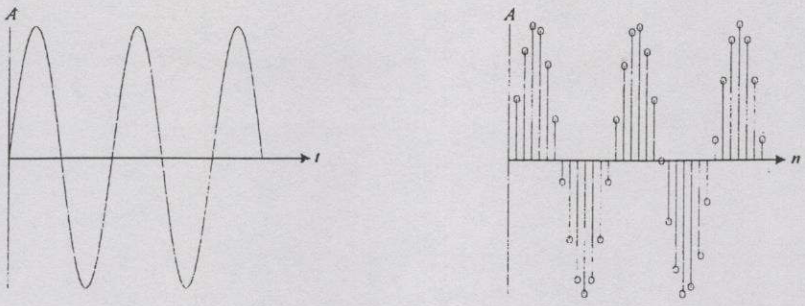
สมการที่ (3.1) คือสัญญาณซายน์ต่อเนื่องทางเวลา ส่วนสมการที่ (3.2) เป็นสัญญาณไม่ต่อเนื่องทางเวลา ซึ่งแสดงได้ตามรูปที่ 3.1 ส่วนสัญญาณแบบ Deterministic แบบไม่มีคาบ เช่น สัญญาณพัลส์สี่เหลี่ยม ซึ่งสามารถเขียนเป็นสมการได้ว่า

$$x(t) = rect(t/T) \tag{3.3}$$

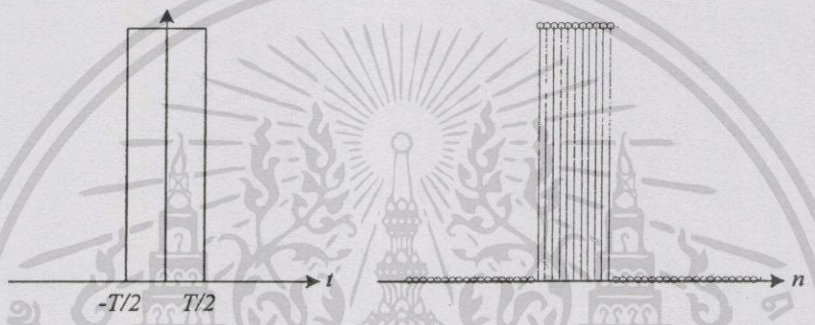
หรือ

$$x(n) = rect(n/T) \tag{3.4}$$

ซึ่งแสดงได้ดังรูปที่ 3.2



รูปที่ 3.1 สัญญาณ Deterministic แบบมีคาบ



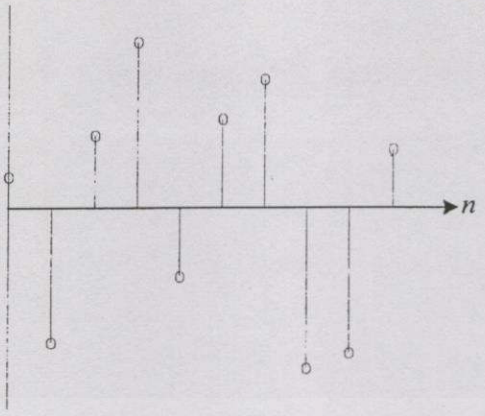
รูปที่ 3.2 สัญญาณแบบ Deterministic แบบไม่มีคาบ

3.1.2 สัญญาณแบบสุ่ม (Random หรือ Stochastic)

สัญญาณแบบสุ่ม เป็นสัญญาณที่ไม่อาจเดาล่วงหน้าได้ ว่าจะมีลักษณะเช่นใด ตัวอย่างสัญญาณแบบสุ่ม เช่น สัญญาณรบกวนที่เกิดจากการควอนไตซ์สัญญาณของวงจร A/D หรือ ที่เกิดจากตัวประมวลผลสัญญาณแบบ fixed point เป็นต้น ตัวอย่างสัญญาณแบบสุ่มแสดงดังรูปที่ 3.3

3.1.3 สัญญาณแบบ Chaotic

สัญญาณประเภทนี้ จะคล้ายคลึงกับสัญญาณแบบสุ่ม แต่จะมีรูปแบบ (pattern) เฉพาะตัว ตัวอย่าง เช่น การหมุนวนของน้ำทำให้เกิดสิ่งที่เรียกว่า Swirl, การพัดของลม หรือ แม้กระทั่งการหมุนเวียนของเลือดในเส้นเลือด ซึ่งสามารถหารูปแบบได้จากสิ่งที่เรียกว่า Attractor หรือตัวที่ทำให้เกิดปรากฏการณ์ Chaos นั่นเอง ซึ่งสัญญาณในลักษณะนี้ยังอยู่ในขั้นการทำวิจัย

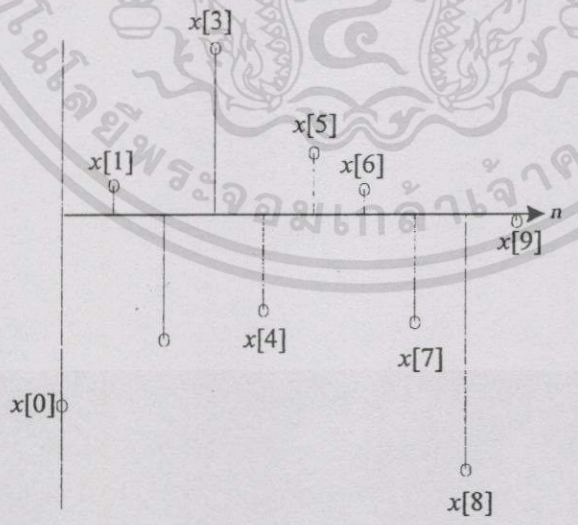


รูปที่ 3.3 สัญญาณแบบสุ่ม

3.2 ตัวแปรสุ่ม (Random Variable)

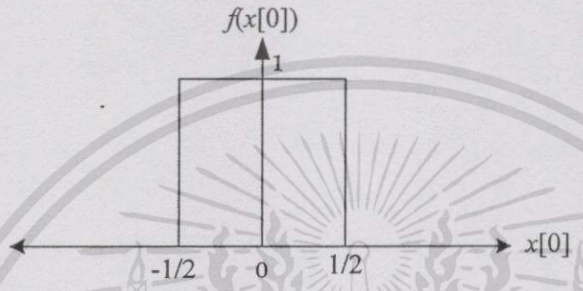
ในหัวข้อนี้จะกล่าวถึง แนวความคิดของตัวแปรสุ่ม และคุณลักษณะเฉพาะตัวต่าง ๆ ของมัน ซึ่งค่อนข้างมีความสำคัญ ด้วยเหตุผลสองประการ คือ หนึ่ง เนื่องจากตัวแปรสุ่มมีค่าที่กระจัดกระจายไม่แน่นอน และมักจะพบบ่อยในทางปฏิบัติ เช่น การตรวจวัดสัญญาณ (signal detection) ทฤษฎีการประมาณค่าความถี่ของสัญญาณ (spectrum estimation) ดังนั้นจึงมีความจำเป็นที่ต้องเข้าใจในเรื่องของตัวแปรสุ่ม เหตุผลที่สอง คือ การศึกษาเรื่องตัวแปรสุ่มเป็นจุดเริ่มต้นสำหรับผู้ที่จะศึกษาในเรื่องกระบวนการสุ่มในขั้นสูงต่อไป

สมมติว่า $x[n]$ คือสัญญาณแบบสุ่มที่ไม่ต่อเนื่องทางเวลา ดังแสดงในรูปที่ 3.4



รูปที่ 3.4 สัญญาณแบบสุ่มไม่ต่อเนื่องทางเวลา

$x[n]$ แต่ละตัวจะเรียกว่า ตัวแปรสุ่ม (random variable) หรือใช้ตัวย่อว่า *r.v.* สิ่งที่เป็นตัวกำหนดค่าตัวแปรสุ่มแต่ละตัวจะขึ้นกับค่าความเป็นไปได้ ซึ่งอยู่ในรูปของฟังก์ชัน ซึ่งรู้จักกันโดยทั่วไป คือ Probability Density Function หรือ เรียกย่อ ๆ ว่า Pdf เช่น $x[0]$ ซึ่งเป็นตัวแปรสุ่มหนึ่งตัวจะมี Pdf ส่วนตัวเป็น $f(x[0])$ หากกำหนดให้ $x[0]$ มีค่า Pdf มีค่าไม่ต่ำกว่า –1/2 และ มากสุดไม่เกิน 1/2 และทราบว่าค่า ความเป็นไปได้ (probability) ของฟังก์ชันมีค่าน้อยกว่าหรือเท่ากับ 1 เสมอ สิ่งที่ได้คือ กราฟของ Pdf ของ $x[0]$ หรือ $f(x[0])$ ตามรูปที่ 3.5



รูปที่ 3.5 Pdf ของตัวแปรสุ่ม $x[0]$

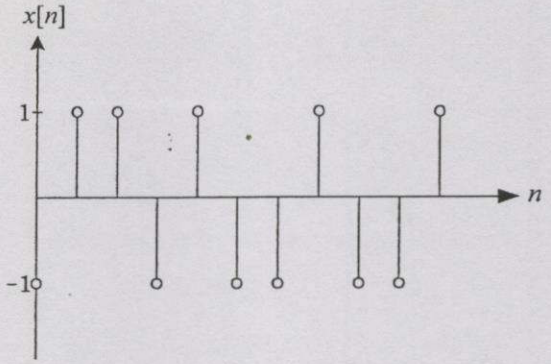
$x[0]$ จะมีค่าเท่าใดก็ได้ภายในกรอบสี่เหลี่ยม สังเกตว่าพื้นที่ภายในกรอบจะมีค่าเท่ากับ 1 เสมอ ซึ่งจะสามารถนำไปสู่การหาคุณสมบัติของ Pdf ได้

3.2.1 คุณสมบัติของ Pdf

ทราบว่าการอินทิเกรตรูปกราฟในรูปที่ 3.5 จะได้เป็นผลลัพธ์เป็นพื้นที่รูปกราฟนั้น ซึ่งเขียนเป็นสมการได้ว่า

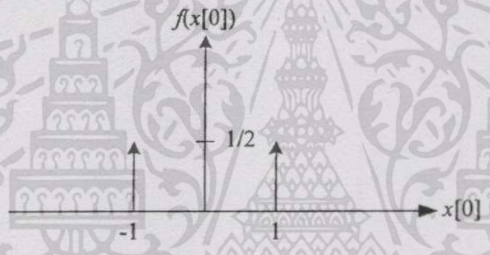
$$\int_{-\infty}^{\infty} f(x[0])dx[0] = 1 \tag{3.5}$$

ในกรณีที่สัญญาณ $x[n]$ เป็นสัญญาณแบบไบนารี ซึ่งจะมีค่าเป็น 1 และ –1 ดังแสดงในรูปที่ 3.6



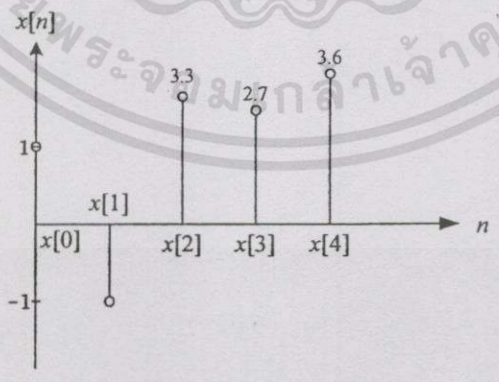
รูปที่ 3.6 สัญญาณแบบสุ่มที่มีค่า 1 และ –1

จะได้ Pdf ของ $x[0]$ เป็นดังรูปที่ 3.7



รูปที่ 3.7 Pdf ของสัญญาณสุ่มแบบไบนารี่

ยกตัวอย่าง ถ้ามีสัญญาณ $x[n]$ ดังรูปที่ 3.8



รูปที่ 3.8 ตัวอย่างสัญญาณแบบสุ่ม

จากรูป $x[0]$ และ $x[1]$ เป็นสัญญาณสุ่มแบบไบนารี่ ซึ่งจะให้ Pdf ดังในรูปที่ 3.7 ส่วน $x[2]$, $x[3]$, $x[4]$ มี Pdf ที่เรียกว่าเป็น Normal Distribution หรือ Gaussian Distribution สัญลักษณ์ที่ใช้แทนคือ

$$N(\underset{Mean}{3.3}, \underset{sJandard\ Deviation}{1})$$

โดย Gaussian distribution มีสมการเป็น

$$f(x[2]) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x[2]-\mu)^2}{2\sigma^2}\right) \tag{3.6}$$

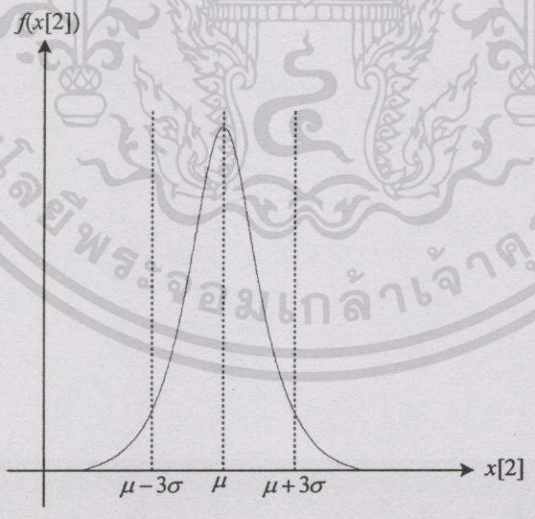
โดยที่

μ = *Mean*

σ = *Standard Deviation*

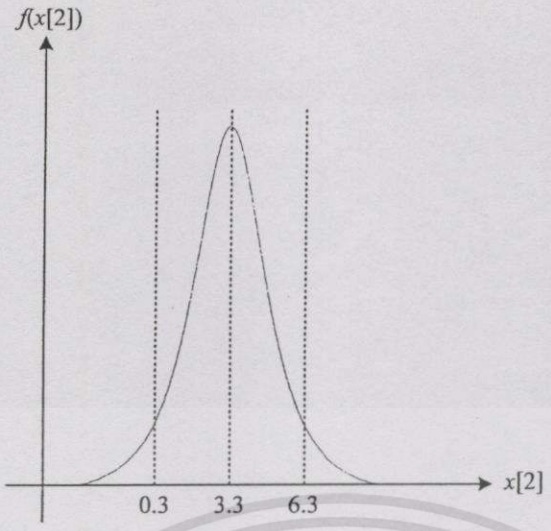
σ^2 = *Variance*

การกระจายของ x[2] มีรูปร่างดังรูปที่ 3.9



รูปที่ 3.9 Gaussian Distribution ของ $x[2]$

จากรูปที่ 3.9 พื้นที่ของกราฟ 99 เปอร์เซ็นต์ จะอยู่ในบริเวณ $\mu-3\sigma$ ถึง $\mu+3\sigma$ ดังนั้น ถ้า $N(3.3,1)$ ก็จะได้ดังรูปที่ 3.10



รูปที่ 3.10 Gaussian Distribution ของ $x[2]$ เมื่อ $N(3.3,1)$

ในกรณีที่ต้องการหาค่า Pdf ของ *r.v* ตัวหนึ่งที่มีค่าน้อยมาก ๆ ($\rightarrow -\infty$) จนถึงค่าหนึ่งของ ($x[n]$) จะได้ผลรวมของ Pdf เหล่านั้นเป็นอีกฟังก์ชันหนึ่ง ซึ่งจะเรียกฟังก์ชันนี้ว่า Cumulative Density Function หรือ Cdf และสัญลักษณ์ที่ใช้คือ $F_x(x[n])$ โดยที่

$$F_x(x[n]) = \text{Probability}\,(X[n] \le x[n]) \tag{3.7}$$

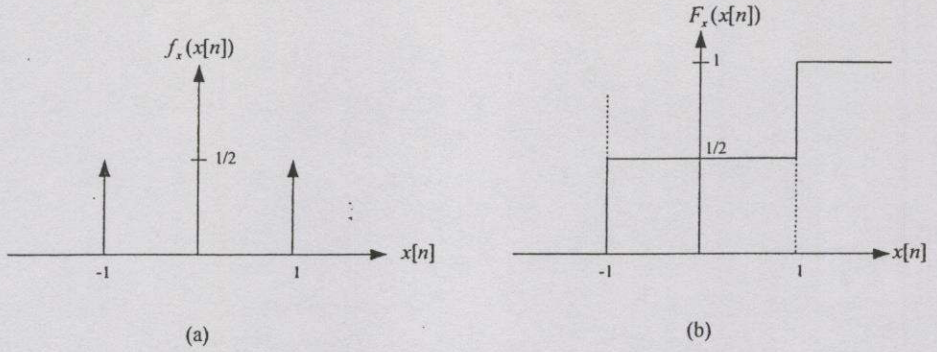
โดยที่ $X[n]$ คือตัวแปรสุ่ม หรือ *r.v* และ $x[n]$ คือค่าหนึ่ง ๆ จากสมการที่ (3.7) ก็คือผลรวมของ Pdf นั่นเอง ดังนั้นจะได้ว่า

$$F_x(x[n]) = \int_{-\infty}^{x[n]} f_x(z)dz \tag{3.8}$$

หรือ

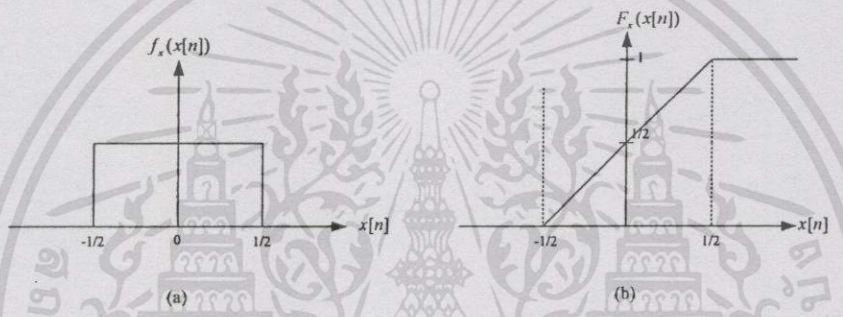
$$f_x(x[n]) = \frac{\partial F_x(x[n])}{\partial x[n]} \tag{3.9}$$

ตัวอย่างหากมี $f_x(x[n])$ ดังรูปที่ 3.11(a) จะได้ $F_x(x[n])$ เป็นดังรูปที่ 3.11(b) ตามลำดับ สังเกตว่าผลสุดท้ายของ $F_x(x[n])$ จะเป็น 1 เสมอ



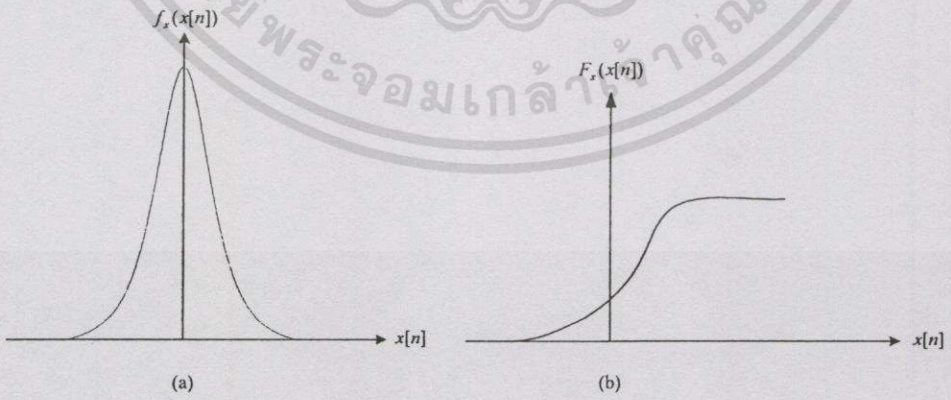
รูปที่ 3.11 (a) Pdf และ (b) Cdf ของตัวแปรสุ่ม $x[n]$

และหาก $f_x(x[n])$ เป็นดังรูปที่ 3.12 (a) จะได้ $F_x(x[n])$ ดังรูปที่ 3.12(b) ตามลำดับ



รูปที่ 3.12 (a) Pdf และ (b) Cdf ของ $x[n]$

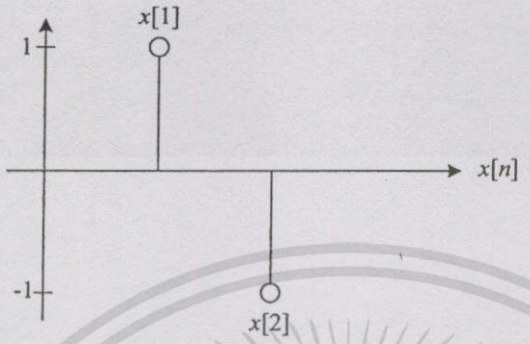
สำหรับ Gaussian Distribution จะได้ดังรูปที่ 3.13



รูปที่ 3.13 (a) Pdf และ (b) Cdf ของตัวแปรสุ่มที่เป็นแบบ Gaussian Distribution

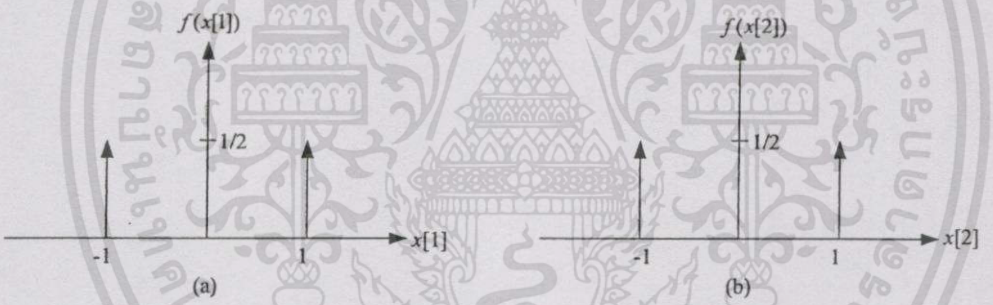
โดยทั่วไป ไม่ได้คาดหวังว่าค่า Pdf ของ $x[n_1]$ และ $x[n_2]$ จะเหมือนกัน ดังนั้นจะมีวิธีการหาความสัมพันธ์ของ $x[n_1]$ และ $x[n_2]$ ในรูปของ Pdf ร่วม หรือ Joint Pdf

สมมติให้ $x[1]$ และ $x[2]$ เป็นสัญญาณสุ่มแบบไบนารีแสดงดังรูปที่ 3.14



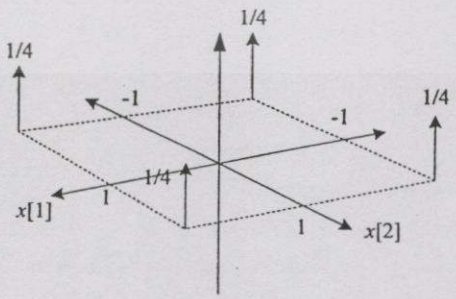
รูปที่ 3.14 สัญญาณแบบสุ่มไบนารี่

จะได้ Pdf สำหรับ $x[1]$ และ $x[2]$ ดังรูปที่ 3.15 (a) และ 3.15 (b) ตามลำดับ



รูปที่ 3.15 (a) Pdf ของ $x[1]$ และ (b) $x[2]$

และได้ Joint Pdf เป็น $f_x(x[1], x[2])$ แสดงดังรูปที่ 3.16



รูปที่ 3.16 Joint Pdf ของ $x[1]$ และ x[2]

หรือเขียนความสัมพันธ์นี้ในรูปสมการ Joint Cdf ดังนี้คือ

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_x(x[1],x[2])dx[1]dx[2] = 1 = F(x[n_1],x[n_2]) \tag{3.10}$$

และจะได้ Joint Pdf ดังสมการ

$$f(x[n_1],x[n_2]) = \frac{\partial^2 F(x[n_1],x[n_2])}{\partial x[n_1]\partial x[n_2]} \tag{3.11}$$

ดังนั้นทำให้สามารถคิดต่อไปได้ว่า หากเป็นการ Distribution แบบ *r.v* หลายตัว ก็จะได้เป็น *k-th order multivariate distribution* เป็น

$$F(x[n_1],x[n_2],\ldots x[n_k]) = \text{Prob}(X[n_1] \le x[n_2],\ldots,X[n_k] \le x[n_k]) \tag{3.12}$$

และได้ Pdf เป็น

$$f(x[n_1],\ldots,x[n_2]) = \frac{\partial^2 F(x[n_1],\ldots,x[n_k])}{\partial x[n_1]\ldots\partial x[n_k]} \tag{3.13}$$

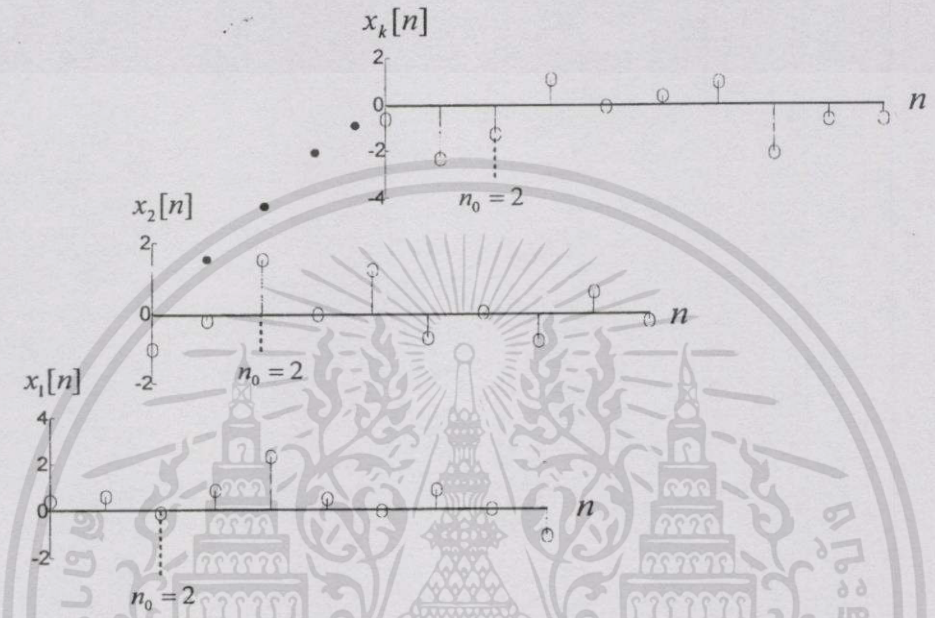
3.2.2 ค่าเฉลี่ยแบบ Ensemble

ที่ผ่านมาได้ทำการพิจารณาสัญญาณแบบสุ่ม $x[n]$ และได้ Pdf ของ $x[n]$ นั้น ๆ ถ้าหากมี $x[n]$ มากกว่า 1 จะต้องพิจารณา $x[n]$ ตั้งแต่ $x_1[n]$ ถึง $x_k[n]$ ที่เวลา $n = n_0$ เดียวกันดังแสดงในรูปที่ 3.17 ถ้าหากกำหนดให้ Probability ของ $x[n]$ ทุกตัวมีค่าเท่ากันจะได้ว่า

$$\text{Prob ของ } x_k[n] = \frac{1}{k} \quad ,\forall k \tag{3.14}$$

ค่าเฉลี่ยแบบ Ensemble เป็นการหาค่าเฉลี่ยของ $f(x[n_0])$ โดยที่ $x[n_0]$ เป็น $x_1[n_0]$, $x_2[n_0]$,..., $x_k[n_0]$ หรือ

$$\begin{aligned}\text{Ensemble average} &= x_1[n_0]\cdot \text{Prob}(x_1[n]) \\ &+ x_2[n_0]\cdot \text{Prob}(x_2[n]) \\ &+ \ldots \\ &+ x_k[n_0]\cdot \text{Prob}(x_k[n])\end{aligned} \tag{3.15}$$



รูปที่ 3.17 การหาค่าเฉลี่ยแบบ Ensemble ของตัวแปรสุ่มหลายตัว

3.2.3 ค่ากลาง (mean value)

มีข้อกำหนดในการหาค่ากลางของสัญญาณสุ่ม เป็น

$$\mu[n] = E\{x[n]\} \tag{3.16}$$

โดยที่ $E\{x[n]\}$ เป็น ค่าเฉลี่ย Ensemble หรือ ค่าคาดหวัง (Expected value) คือ

$$E\{x[n]\} = \int_{-\infty}^{\infty} x[n] f(x[n]) dx[n] \tag{3.17}$$

โดยที่ $f(x[n])$ เป็น Pdf ค่า mean มีชื่อเรียกอีกอย่างว่า First moment และจะสามารถเขียนค่า mean ในรูปของ relative frequency ได้เป็น

$$\mu[n] = \lim_{N\to\infty}\left\{\frac{1}{N}\sum_{i=1}^{N} x_i[n]\right\} \tag{3.18}$$

ซึ่งจะใช้แทน ค่าเฉลี่ย Ensemble ได้ แต่โดยทั่วไป หาก Pdf ของ $x[n]$ ไม่เท่ากับ Pdf ของ $x[m]$ ก็จะได้ว่า

$$\mu[n] \neq \mu[m] \qquad \text{ที่ } n \neq m \tag{3.19}$$

ตัวกระทำ Expectation ที่น่าสนใจ ซึ่งพอจะสรุปได้ดังนี้คือ

i) ความเป็นเชิงเส้น

$$E\{ax[n] + by[m]\} = aE\{x[n]\} + bE\{y[m]\} \tag{3.20}$$

ii) ความไม่เป็นอิสระต่อกัน

$$E\{x[n]y[m]\} \neq E\{x[n]\} \cdot E\{y[m]\} \tag{3.21}$$

คุณสมบัติข้อนี้จะเท่ากันก็ต่อเมื่อ $x[n]$ และ $y[m]$ นั้นเป็นอิสระต่อกัน

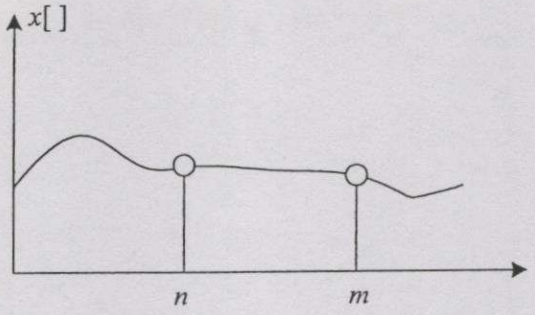
iii) หาก $y[n] = g(x[n])$ และ Pdf ของ $x[n]$ เป็น $f(x[n])$ แล้วจะสามารถหา $E\{y[n]\}$ โดยที่ Pdf ที่ต้องการทราบมีเพียง $f(x[n])$ เท่านั้น จะได้ว่า

$$E\{y[n]\} = \int_{-\infty}^{\infty} g(x[n]) \cdot f(x[n]) dx[n] \tag{3.22}$$

3.2.4 ค่าความเกี่ยวพันธ์ (Correlation)

Correlation เป็นตัวบอกความเหมือนของ *r.v* 2 ตัว ดังตัวอย่างในรูปที่ 3.18 ตัวแปรสุ่มมีสองตัวคือ $x[n]$ และ $x[m]$ ความเกี่ยวเนื่องกันของตัวแปรทั้งสองจะสามารถเขียนให้อยู่ในรูปของ

$$r(m,n) = E\{x[m] \cdot x[n]\} \tag{3.23}$$



รูปที่ 3.18 ตัวแปรสุ่มสองตัว

หรือ

$$r(m,n) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x[m]x[n]f(x[m],x[n])dx[m]dx[n] \tag{3.24}$$

และในรูปของ relative frequency จะได้ว่า

$$r(m,n) = \lim_{N\to\infty}\left\{\frac{1}{N}\sum_{i=1}^{N} x_i[m],x_i[n]\right\} \tag{3.25}$$

กรณีพิเศษเมื่อ $n = m$ จะได้

$$r(m,n) = E\{x^2[n]\} \tag{3.26}$$

ซึ่งเป็นค่ากำลังงาน หรือ พลังงานเฉลี่ยของสัญญาณใด ๆ ค่าในสมการที่ (3.23-3.26) โดยทั่วไปจะเรียกว่า Autocorrelation

3.2.5 ค่าความแปรปรวนร่วม (Covariance)

ค่า covariance กำหนดได้ตามสมการ

$$c(m,n) = E\{(x[m]-\mu[m])(x[n]-\mu[n])\} \tag{3.27}$$

ถ้า $\mu[n] = \mu[m] = 0$ จะได้ว่า covariance จะมีค่าเท่ากับ Autocorrelation จากสมการที่ (3.27) จะได้ว่า

$$c(m,n) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x[m]-\mu[m])(x[n]-\mu[n])f(x[m],x[n]dx[m]dx[n]$$
$$= E\{x[m]x[n]\} - \mu[n]\mu[m] \tag{3.28}$$

จากสมการที่ (3.27) ถ้าหากว่า $m = n$ แล้วจะได้ว่า

$$c(n,n) = E\{(x[n]-\mu[n])^2\}$$
$$= \sigma_n^2 \tag{3.29}$$

จากสมการที่ (3.29) เรียกว่า ค่าความแปรปรวนหรือ variance นั่นเอง และจากสมการที่ (3.29) จะสามารถหาค่า variance ได้ดังนี้คือ

$$\sigma_n^2 = \int(x[n]-\mu[n])^2 f[x]dx$$
$$= E\{x^2[n]\} - (E\{x[n]\})^2 \tag{3.30}$$

ในกรณีของกระบวนการสุ่มแบบ Gaussian จะได้ว่า

$$\sigma_n^2 = 1 \tag{3.31}$$

3.2.6 ความเป็นอิสระ (Independence)

จะกล่าวว่า $\{x_1[n]\}$ และ $\{x_2[n]\}$ เป็นอิสระต่อกันก็ต่อเมื่อ

$$f(x_1[n]x_2[n]) = f(x_1[n]) \cdot f(x_2[n]) \tag{3.32}$$

และ

$$E\{x_1[n], x_2[n]\} = E\{x_1[n]\} \cdot E\{x_2[n]\} \tag{3.33}$$

หรือแม้กระทั่ง $x_1[n]$ และ $x_2[n]$ จะเป็นสัญญาณ $x[n]$ เดียวกันก็ได้ โดยที่ $x_1[n]= x_2[m]$ และ $x_2[n] = x[m+n]$ จากสมการที่ (3.28) หากกล่าวว่า $x[m]$ และ $x[n]$ เป็นอิสระต่อกันแล้วจะได้ว่า

$$c(m,n) = E\{x[m]x[n]\} - \mu[n]\mu[m] = 0 \tag{3.34}$$

หรือกล่าวอีกนัยหนึ่งก็คือ ตัวแปรสุ่มทั้งสอง ไม่เกี่ยวพันธ์กัน (uncorrelated) สรุปก็คือ independent samples จะ uncorrelated ส่วน uncorrelated samples อาจจะไม่ independent หากมันไม่เป็น Gaussian distribution

3.2.7 Orthogonality

หาก x[m] และ x[n] นั้น uncorrelated กัน และ $\mu = 0$ จะกล่าวว่ามัน orthogonal กัน สัญลักษณ์ที่ใช้ คือ $\perp$ หรือเขียนในเทอมของ $E\{\cdot\}$ จะได้ว่า

$$E\{x[m] \cdot x[n]\} = 0 \tag{3.35}$$

3.2.8 Stationarity

ความหมายที่แท้จริงของ Stationary คือ ค่าของ $f_x(x[n])$ จะต้องมีค่าเท่ากัน สำหรับทุก ๆ ค่าของ n หรือจะมีชื่อเรียกอีกชื่อหนึ่งว่า Strictly Stationary จะสามารถเขียนเป็นสมการได้ว่า

$$f(x[n_1 + n_0], x[n_2 + n_0]),\ldots, x[n_k + n_0]) = f(x[n_1],\ldots, x[n_k]) \tag{3.36}$$

โดยทั่วไปในทางปฏิบัติจะไม่ค่อยมี Strict Stationarity ดังนั้นจึงลดข้อกำหนดลงเหลือเพียงการดูที่ mean และ variance ซึ่งเป็น 1^{st} และ 2^{nd} moments เท่านั้น และเรียกชื่อใหม่ว่า Wide-Sense Stationary หรือ WSS สำหรับ $x[n]$ ซึ่งเป็น WSS จะมีคุณสมบัติดังต่อไปนี้คือ

i)

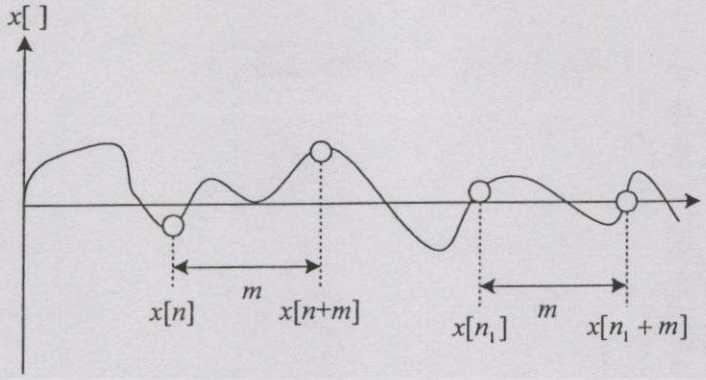
$$E\{x[m]\} = E\{x[n]\} = \mu \tag{3.37}$$

เช่น $x[n]$ เป็น Gaussian จะได้ว่า $\mu = 0$

ii)

$$r(m) = E\{x[n] \cdot x[n+m]\} \tag{3.38}$$

หมายความว่า ค่า Autocorrelation จะขึ้นกับค่าของ m เท่านั้น เช่นในรูปที่ 3.19 จะได้ว่า Autocorrelation ของ $x[n]$ กับ $x[n+m]$ มีค่าเท่ากับ Autocorrelation ของ $x[n_1]$ กับ $x[n_1+m]$



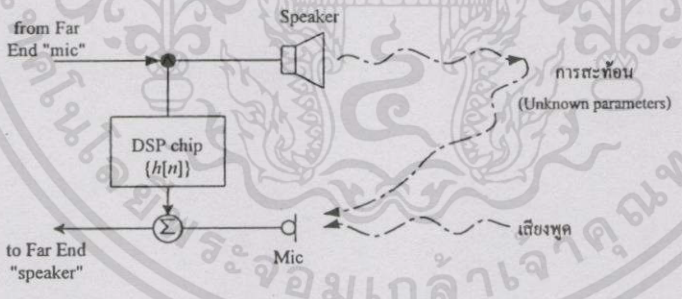
รูปที่ 3.19 อธิบายความหมายของ Autocorrelation

3.2.9 ความเกี่ยวพันธ์แบบไขว้ (Cross-correlation)

Cross-correlation จะมีนิยามตามสมการ คือ

$$r_{xy}(m) = E\{x[n]y[n+m]\} = r_{xy}[-m] \tag{3.39}$$

เพื่อให้สามารถเข้าใจได้โดยง่ายจะขอยกตัวอย่างการประยุกต์ใช้งาน Cross-correlation โดยนำมาทำ System Identification ของ Hand-free Mobile phone ดังแสดงในรูปที่ 3.20 และรูปที่ 3.21



รูปที่ 3.20 Hand-free Mobile phone

สิ่งที่ต้องการคือ $\{h[n]\}$ ที่ทำให้ $y[n]$ เหมือน $d[n]$ มากที่สุด โดยตัววัดความเหมือนกันคือ $e[n]$ โดยจะใช้วิธีการลด $E\{e^2[n]\}$ ให้มีค่าน้อยที่สุด หรือหมายความว่า $\underline{h}$ จะได้จาก

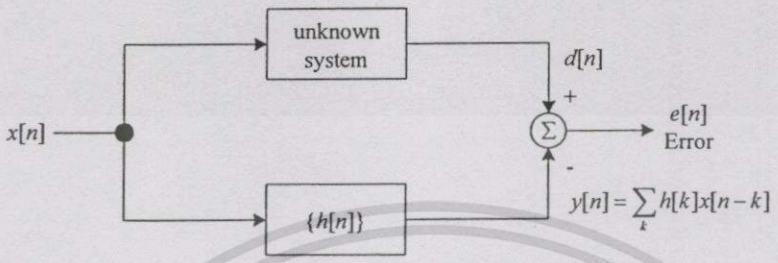
$$\underline{h} = R_{xx}^{-1}\underline{r}_{dx} \tag{3.40}$$

เมื่อ

$\underline{h}$ = Impulse Response ของ Unknown System

R_{xx} = Autocorrelation ของสัญญาณอินพุต

$\underline{r}_{dx}$ = Cross-correlation ของ $d[n]$ และ $x[n]$



รูปที่ 3.21 บล๊อกไดอะแกรมของรูปที่ 3.20

สามารถศึกษารายละเอียดได้จากเอกสารอ้างอิง [20] และเนื่องจาก Correlation และ Covariance จะอธิบายความเกี่ยวพันธ์กันระหว่างสัญญาณสองสัญญาณ โดยที่ ถ้าสัญญาณทั้งสองเป็นสัญญาณเดียวกันแล้วจะได้ Autocorrelation และ Autocovariance และถ้าสัญญาณทั้งสองมาจากคนละแห่งแล้วจะได้ Cross-correlation และ Cross-covariance ซึ่งจะสามารถสรุปได้ดังนี้คือ

สมมติว่ามี สัญญาณแบบสุ่มที่มีคุณสมบัติเป็น stationary สองสัญญาณคือ $x[n]$ และ $y[n]$ จะสามารถหาค่า Autocorrelation, Autocovariance, Cross-correlation และ Cross-covariance ตามลำดับ ได้ตามสมการต่อไปนี้ คือ

$$r_{xx}(m) = E\{x[n]x[n+m]\} \tag{3.41}$$

$$c_{xx}(m) = E\{(x[n]-\mu_x)(x[n+m]-\mu_x)\} \tag{3.42}$$

$$r_{xy}(m) = E\{x[n]y[n+m]\} \tag{3.43}$$

$$c_{xy}(m) = E\{(x[n]-\mu_x)(y[n+m]-\mu_y)\} \tag{3.44}$$

เมื่อ μ_x และ μ_y คือค่า mean ของสัญญาณ $x[n]$ และ $y[n]$ ตามลำดับ ซึ่งมีคุณสมบัติที่น่าสนใจดังนี้ คือ

i)

$$c_{xx}(m) = r_{xx}(m) - \mu_x^2 \tag{3.45}$$

$$c_{xy}(m) = r_{xy}(m) - \mu_x\mu_y \tag{3.46}$$

จากสมการที่ (3.45) และสมการที่ (3.46) ถ้า $\mu_x = 0$ จะทำให้ Correlation และ Covariance มีค่าเท่ากัน

ii)

$$r_{xx}(0) = E\{x^2[n]\} \text{ (ค่า กำลังสองเฉลี่ย)} \tag{3.47}$$

$$c_{xx}(0) = \sigma_x^2 \text{ (ค่า variance)} \tag{3.48}$$

iii)

$$r_{xx}(m) = r_{xx}(-m) \tag{3.49}$$

$$c_{xx}(m) = c_{xx}(-m) \tag{3.50}$$

$$r_{xy}(m) = r_{xy}(-m) \tag{3.51}$$

$$c_{xy}(m) = c_{xy}(-m) \tag{3.52}$$

iv) ถ้า $y[n] = x[n - n_0]$ แล้วจะได้ว่า

$$r_{yy}(m) = r_{xx}(m) \tag{3.53a}$$

$$c_{yy}(m) = c_{xx}(m) \tag{3.53b}$$

v) สำหรับกระบวนการสุ่มใด ๆ เมื่อ $m \to \infty$ แล้วจะทำให้ตัวแปรสุ่มมีการ Uncorrelated กันน้อยมากดังนั้นจะได้

$$\lim_{m\to\infty} r_{xx}(m) = (E\{x[n]\})^2 = \mu_x^2 \tag{3.54}$$

$$\lim_{m\to\infty} c_{xx}(m) = 0 \tag{3.55}$$

$$\lim_{m\to\infty} r_{xy}(m) = \mu_x \mu_y \tag{3.56a}$$

$$\lim_{m\to\infty} c_{xy}(m) = 0 \tag{3.56b}$$

3.3 การแทนสัญญาณที่มีพลังงานไม่จำกัดด้วยสเปกตรัม [26]

แม้ว่าจะไม่สามารถทำการแปลง z (z-transform) กับสัญญาณที่มีพลังงานไม่จำกัด (Infinite-Energy Signals) ได้เนื่องจากสัญญาณดังกล่าวมีลักษณะไม่เป็นรายคาบ (Aperiodic) อย่างไรก็ตามเมื่อนำสัญญาณนี้มาทำ Autocorrelation และ ทำ Autocovariance ค่าที่ได้จะมีลักษณะ

เป็นรายคาบเกิดขึ้น จึงทำให้สามารถ นำเอาการแปลง z และ การแปลงฟูริเย (Fourier Transform) มาใช้สำหรับหาสเปกตรัมของสัญญาณที่มีพลังงานไม่จำกัดเหล่านี้ได้ การแทนสัญญาณเหล่านี้ด้วยสเปกตรัม มีความสำคัญมากต่อ การอธิบายความสัมพันธ์ระหว่างสัญญาณอินพุต และ สัญญาณเอาต์พุตของระบบ เชิงเส้นไม่แปรตามเวลา หรือ LTI เมื่อสัญญาณอินพุตเป็นสัญญาณที่มีพลังงานไม่จำกัด

3.3.1 การแปลง z ของ Correlation และ Covariance

กำหนดให้ $R_{xx}(z)$, $C_{xx}(z)$, $R_{xy}(z)$ และ $C_{xy}(z)$ คือการแปลง z ของ $r_{xx}(m)$, $c_{xx}(m)$, $r_{xy}(m)$ และ $c_{xy}(m)$ ตามลำดับ จากสมการที่ (3.54) และ (3.56) จะทราบได้ทันทีว่า การแปลง z จะมีค่าก็ต่อเมื่อ $\mu_x = 0$ เท่านั้น ซึ่งจะทำให้ $R_{xx}(z) = C_{xx}(z)$ และ $R_{xy}(z) = C_{xy}(z)$ ซึ่งมีคุณสมบัติที่น่าสนใจดังนี้

i)

$$\sigma_x^2 = \frac{1}{2\pi j}\oint_{Closed} C_{xx}(z)z^{-1}dz \tag{3.57}$$

เมื่อ *Closed* เป็นเส้นทางปิดในบริเวณที่ $C_{xx}(z)$ ลู่เข้า (region of convergence:ROC)

ii)

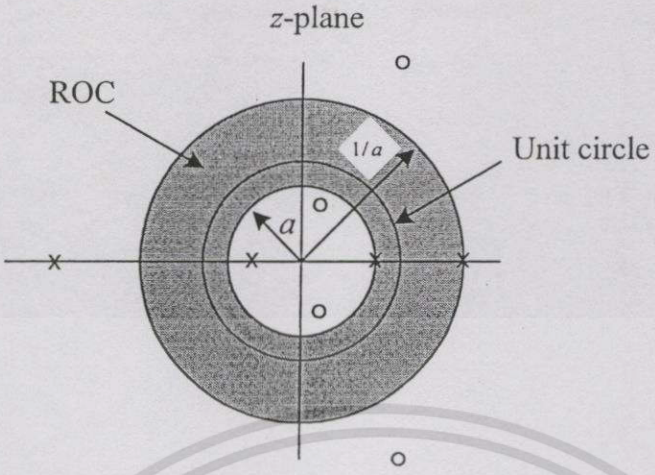
$$C_{xx}(z) = C_{xx}(1/z) \tag{3.58a}$$

$$C_{xy}(z) = C_{yx}^*(1/z^*) \tag{3.58b}$$

สมการที่ (3.58) ได้รับโดยตรงจากการแปลง z สมการที่ (3.49)-(3.52) และ ROC ของ $C_{xx}(z)$ จะเป็นไปตามเงื่อนไขดังนี้ คือ

$$a < |z| < \frac{1}{a}$$

เมื่อ $0 < a < 1$ เนื่องจาก $C_{xx}(z) \to 0$ เมื่อ $m \to \infty$ ROC จะต้องอยู่ภายในวงกลมหนึ่งหน่วย ในกรณีที่ $C_{xx}(z)$ เป็นฟังก์ชันแบบเศษส่วน (rational function) ของตัวแปร z เป็นนัยว่า โพล และ ซีโร่ของ $C_{xx}(z)$ จะเกิดเป็นคู่ Complex conjugate กัน ดังแสดงในรูปที่ 3.22



รูปที่ 3.22 ROC และ ตำแหน่งโพลและซีโร่ที่ได้จากการแปลง z ของ $C_{xx}(z)$

3.3.2 สเปกตรัมกำลังงาน

เนื่องจากว่า ROC จะอยู่ภายในวงกลมหนึ่งหน่วย จะทำให้สามารถเขียนสมการที่ (3.57) ใหม่ได้ดังนี้

$$\sigma_x^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} P_{xx}(\omega)d\omega \tag{3.59}$$

โดยที่

$$P_{xx}(\omega) = C_{xx}(e^{j\omega}) \tag{3.60}$$

เมื่อ $\mu_x = 0$ จะเรียกว่า variance ซึ่งก็คือค่า mean-square หรือ กำลังงานเฉลี่ยนั่นเอง ดังนั้น พื้นที่ภายใน $P_{xx}(\omega)$ สำหรับ $-\pi \le \omega \le \pi$ จะเป็นสัดส่วนโดยตรงกับกำลังงานเฉลี่ยของสัญญาณ และเช่นเดียวกัน การอินทิเกรต $P_{xx}(\omega)$ ตลอดช่วงความถี่หนึ่ง ก็จะเป็นสัดส่วนโดยตรงกับกำลังงานเฉลี่ยของสัญญาณตลอดช่วงความถี่นั้นด้วย จากเหตุผลนี้เอง จึงเรียก $P_{xx}(\omega)$ ว่า สเปกตรัมความหนาแน่นกำลังงาน (power density spectrum) หรือเรียกย่อ ๆ ว่า สเปกตรัม (spectrum) จะพบว่า สามารถคำนวณหาค่าความหนาแน่นกำลังงานได้จากการแปลงฟูริเย ของ Autocorrelation หรือ Autocovariance อย่างใดอย่างหนึ่ง อย่างไรก็ตามการใช้วิธีนี้จะทำได้ยากเมื่อ $\mu_x \neq 0$, เนื่องจาก $r_{xx}(m) \to \mu_x^2$ เมื่อ $m \to \infty$ ในกรณีเช่นนี้ จะเลือกใช้สมการที่ (3.61) มาใช้

แทน (3.60) จากคุณสมบัติข้อ ii) ในหัวข้อ 3.3.1 ทำให้ทราบว่า $P_{xx}(\omega)$ เป็นฟังก์ชันแบบสมมาตร คือ $P_{xx}(\omega)=P_{xx}(-\omega)$ และก็เป็นความจริงที่ว่า ความหนาแน่นกำลังงานไม่มีค่าเป็นลบแน่นอน

ในทำนองเดียวกันจะสามารถนิยาม ความหนาแน่นกำลังงานแบบไขว้ได้ดังนี้คือ

$$P_{xy}(\omega) = C_{xy}(e^{j\omega}) \tag{3.61}$$

ในทำนองเดียวกัน

$$P_{xy}(\omega) = P_{yx}(-\omega) \tag{3.62}$$

3.3.3 ผลตอบสนองของระบบเชิงเส้นต่อสัญญาณสุ่ม

พิจารณาระบบ LTI ซึ่งมีความเสถียรภาพ และมีผลตอบสนองของระบบต่อสัญญาณอิมพัลส์เป็น $h[n]$ และให้ $x[n]$ คือ ลำดับของสัญญาณอินพุต เป็นสัญญาณแบบสุ่ม มีคุณสมบัติเป็น WSS แล้วจะได้สัญญาณเอาต์พุตของระบบ LTI ดังสมการ คือ

$$y[n] = \sum_{k=-\infty}^{\infty} h[n-k]x[k] = \sum_{k=-\infty}^{\infty} h[k]x[n-k] \tag{3.63}$$

สัญญาณอินพุตอาจจะกำหนดได้ด้วยค่า mean คือ μ_x และฟังก์ชัน Autocorrelation $r_{xx}(m)$ และจะแสดงให้เห็นว่า เมื่อสัญญาณอินพุตเป็น stationary แล้ว สัญญาณเอาต์พุต $y[n]$ ก็จะเป็น stationary ด้วย ซึ่งจะสามารถกำหนดได้ด้วยค่าทางสถิติเช่นเดียวกับสัญญาณอินพุต ในการประยุกต์ใช้งานหลาย ๆ อย่าง เพียงพอแล้วที่จะกำหนดคุณลักษณะของสัญญาณอินพุต และสัญญาณเอาต์พุต ในเทอมของค่าเฉลี่ยอย่างง่าย ๆ เช่น mean, variance และ autocorrelation ซึ่งจะสามารถหาความสัมพันธ์ระหว่างปริมาณทั้งสองได้ดังนี้

กำหนดให้ค่า mean ของสัญญาณเอาต์พุตเป็น

$$\begin{aligned} \mu_y = E\{y[n]\} &= \sum_{k=-\infty}^{\infty} h[k]E\{x[n-k]\} \\ &= \mu_x \sum_{k=-\infty}^{\infty} h[k] \end{aligned} \tag{3.64}$$

จากความจริงที่ว่า ค่าคาดหวังของผลบวก จะเท่ากับผลบวกของค่าคาดหวัง ในเทอมของฟังก์ชันระบบ จะเขียนได้ดังสมการ คือ

$$\mu_y = H(e^{j0})\mu_x \tag{3.65}$$

จากสมการที่ (3.65) เป็นจริงทั้งนี้เพราะ สัญญาณอินพุตเป็น stationary จึงทำให้สัญญาณเอาต์พุตคงที่ด้วย

ต่อไปจะสมมติว่า สัญญาณเอาต์พุตมีลักษณะเป็น nonstationary ชั่วคราว จะสามารถกำหนดค่า ฟังก์ชัน autocorrelation ของสัญญาณเอาต์พุตได้ดังสมการ คือ

$$\begin{aligned} r_{yy}(n, n+m) &= E\{y[n]y[n+m]\} \\ &= E\left\{\sum_{k=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} h[k]h[j]x[n-k]x[n+m-j]\right\} \\ &= \sum_{k=-\infty}^{\infty} h[k]\sum_{j=-\infty}^{\infty} h[j]E\{x[n-k]x[n+m-j]\} \end{aligned} \tag{3.66}$$

เนื่องจากสมมติว่า $x[n]$ เป็น stationary แล้ว $E\{x[n-k]x[n+m-j]$ จะขึ้นกับความแตกต่างของ $m+k-j$ ดังนั้น

$$r_{yy}(n, n+m) = \sum_{k=-\infty}^{\infty} h[k]\sum_{j=-\infty}^{\infty} h[j]r_{xx}[m+k-j] = r_{yy}[m] \tag{3.67}$$

นั่นคือ ฟังก์ชัน autocorrelation ของสัญญาณเอาต์พุต ก็จะขึ้นกับ ค่าความต่างของเวลา m ด้วย ดังนั้นสรุปได้ว่า สำหรับระบบที่มีคุณสมบัติเป็น LTI ถ้าสัญญาณอินพุตเป็น stationary แล้ว จะได้สัญญาณเอาต์พุตเป็น stationary ด้วย

ถ้าแทน $l = j - k$ ลงในสมการที่ (3.67) แล้วจะสามารถเขียนใหม่ได้ดังสมการ คือ

$$\begin{aligned} r_{yy}(m) &= \sum_{l=-\infty}^{\infty} r_{xx}[m-l]\sum_{k=-\infty}^{\infty} h[k]h[l+k] \\ &= \sum_{k-\infty}^{\infty} r_{xx}[m-l]v[l] \end{aligned} \tag{3.68}$$

เมื่อกำหนดให้

$$v[l] = \sum_{k=-\infty}^{\infty} h[k]h[l+k] \tag{3.69}$$

สมการที่ (3.69) เรียกว่า aperiodic autocorrelation sequence หรือเรียกง่าย ๆ ว่า autocorrelation ของ $h[n]$ จริงแล้ว สมการที่ (3.69) ก็คือ การคอนโวลูชันแบบไม่ต่อเนื่องทางเวลาของ $h[n]$ กับ $h[-n]$ นั่นเอง

สมมติว่า $\mu_x = 0$ ทำการแปลง z สมการที่ (3.68) และ สมการที่ (3.69) จะได้ว่า

$$\begin{aligned} R_{yy}(z) &= V(z)R_{xx}(z) \\ &= H(z)H(z^{-1})R_{xx}(z) \end{aligned} \tag{3.70}$$

ในเทอมของความหนาแน่นกำลังงานจะได้ว่า

$$P_{yy}(\omega) = \left|H(e^{j\omega})\right|^2 P_{xx}(\omega) \tag{3.71}$$

สมมติว่า $\mu_x = 0$ และ จากสมการที่ (3.69), $\mu_y = 0$ เหมือนกัน ดังนั้นจะได้ว่า

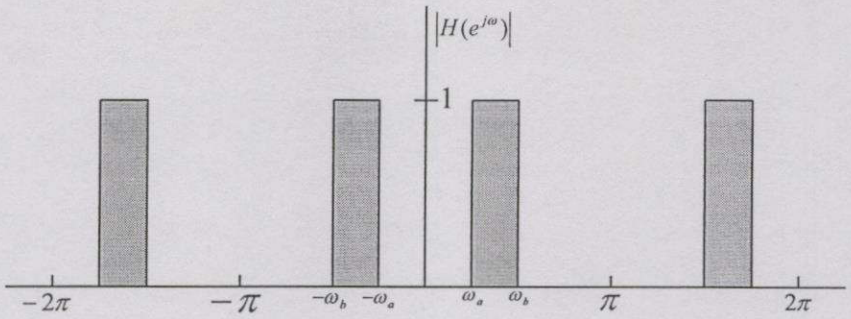
$$\begin{aligned} r_{yy}(0) &= \frac{1}{2\pi}\int_{-\pi}^{\pi} P_{yy}(\omega)d\omega \\ &= \text{กำลังงานเฉลี่ยรวมที่เอาต์พุต} \end{aligned} \tag{3.72}$$

แทนสมการที่ (3.71) ลงในสมการที่ (3.72) จะได้

$$r_{yy}(0) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \left|H(e^{j\omega})\right|^2 P_{xx}(\omega)d\omega \tag{3.73}$$

สมมติว่า $H(e^{j\omega})$ เป็นตัวกรองแบบแถบผ่านในอุดมคติ ดังแสดงในรูปที่ 3.23 จากที่ทราบว่า $r_{xx}(m)$ เป็นฟังก์ชันคู่ดังนั้น

$$P_{xx}(\omega) = P_{xx}(-\omega) \tag{3.74}$$



รูปที่ 3.23 ผลตอบสนองทางความถี่ของตัวกรองแบบแถบผ่านในอุดมคติ

และในทำนองเดียวกัน $|H(e^{j\omega})|^2$ ก็เป็นฟังก์ชันคู่ในตัวแปร ω ดังนั้นจะได้ว่า

$$\begin{aligned} r_{yy}(0) &= \text{กำลังเฉลี่ยเอาต์พุต} \\ &= \frac{1}{\pi}\int_{\omega_a}^{\omega_b} P_{xx}(\omega)d\omega \end{aligned} \tag{3.75}$$

ดังนั้นพื้นที่ภายใต้ $P_{xx}(\omega)$ ระหว่าง ω_a ถึง ω_b จะสามารถใช้แทน กำลังงานเฉลี่ยของสัญญาณอินพุต ในช่วงความถี่หนึ่งนั่นเอง

3.4 สรุป

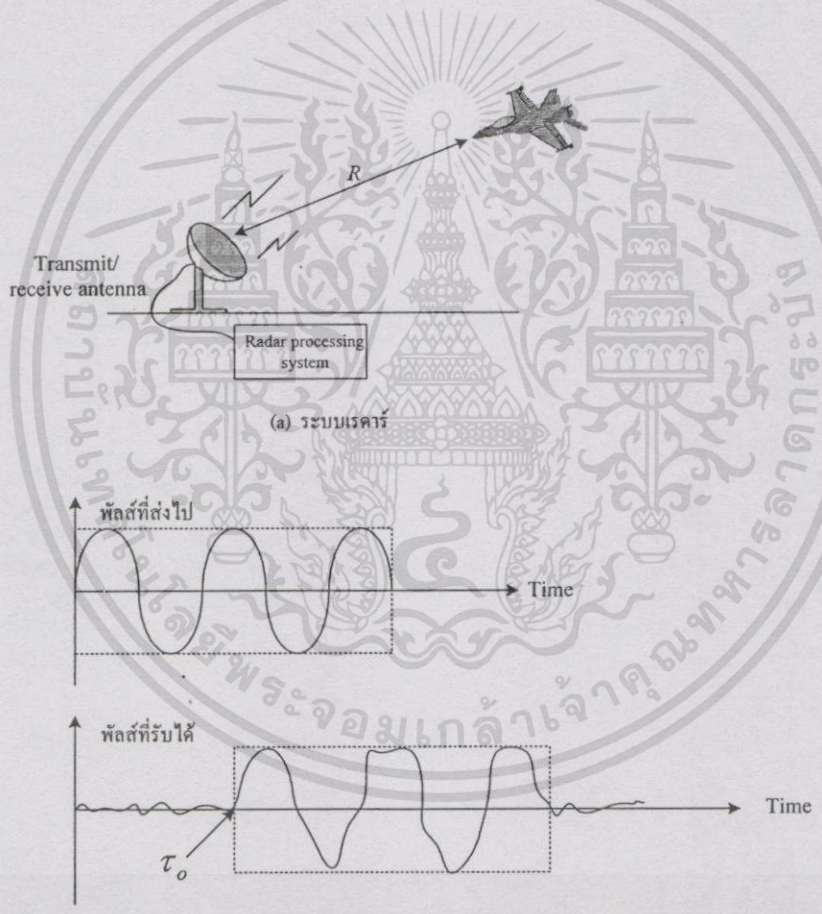
ในบทนี้ได้กล่าวถึง ทฤษฎีพื้นฐานของกระบวนการสุ่มแบบไม่ต่อเนื่องทางเวลา โดยเริ่มจาก การกล่าวทั่ว ๆ ไปเกี่ยวกับตัวแปรสุ่ม และกระบวนการสุ่มของตัวแปรสุ่มหลายตัว ถึงแม้ว่าการที่จะอธิบายถึงคุณสมบัติของกระบวนการสุ่มได้อย่างสมบูรณ์นั้นจำเป็นจะต้องทราบ Joint distribution หรือ density function อย่างไรก็ตาม จะสามารถอธิบายกระบวนการสุ่มได้ด้วยค่า mean, variance และ autocorrelation โดยการใช้ค่าเฉลี่ยแบบ ensemble แทน density function ได้ จากนั้นได้กล่าวถึง กระบวนการสุ่มที่มีคุณสมบัติเป็น stationary และแบบ wide-sense stationary (WSS) ซึ่งพบว่า ค่า mean ของกระบวนการสุ่มนี้จะมีค่าคงที่ไม่ขึ้นกับเวลา นอกจากนี้ค่า autocorrelation $E\{x[n]x[n+m]\}$ จะขึ้นกับค่าความแตกต่างทางเวลา m เท่านั้น ถึงแม้ว่าในทางปฏิบัติเงื่อนไขที่เป็น WSS ยากที่จะเกิดขึ้น แต่ถ้าพิจารณาในช่วงเวลาสั้น ๆ ก็อาจจะสมมติได้ว่า กระบวนการสุ่มนั้นมีคุณสมบัติเป็น WSS ได้ ในบางปัญหามีความจำเป็นจะต้องทราบค่าทางสถิติอันดับ 1 (mean, variance) ค่าทางสถิติอันดับสอง (correlation, covariance) ของกระบวนการสุ่ม ซึ่งเป็นไปไม่ได้ที่จะทราบล่วงหน้าได้ ดังนั้น จำเป็นจะต้องใช้วิธีการประมาณค่าทั้งสอง ซึ่งมีด้วยกันสองแบบ

คือ โดยใช้การเฉลี่ยเอ็นเซ็มเบิล (ensemble average) และ ค่าเฉลี่ยทางเวลา (time average) ในกรณีที่ค่าเฉลี่ยทั้งสองเท่ากัน จะเรียกกระบวนการสุ่มนั้นว่า มีคุณสมบัติเป็นเออร์กอดิก (ergodic)

ปริมาณทางสถิติที่สำคัญอีกอันหนึ่งที่ได้กล่าวถึงในบทนี้คือ สเปกตรัมกำลังงาน ซึ่งเป็นการทำการแปลงฟูริเยของกระบวนการสุ่มแบบ WSS และยังได้แสดงให้เห็นถึงความสัมพันธ์ระหว่าง สัญญาณอินพุต และสัญญาณเอาต์พุตจากระบบ LTI ให้เห็นอีกด้วย ซึ่งทฤษฎีทั้งหมดที่ได้กล่าวถึงในบทนี้ จะเป็นทฤษฎีที่ต้องใช้สำหรับการศึกษา การประมวลผลสัญญาณแบบอะแดปทีฟ ซึ่งจะได้กล่าวในบทต่อไป

บทที่ 4
ทฤษฎีการประมาณค่า

ทฤษฎีการประมาณค่า (Estimation Theory) จะพบในระบบการประมวลผลสัญญาณดิจิตอลหลาย ๆ ระบบ [34] เช่น ระบบเรดาร์ (Radar) ระบบโซนา (Sonar) ระบบประมวลผลสัญญาณเสียง (Speech) การวิเคราะห์สัญญาณภาพ (Image analysis) ทางการแพทย์ (Biomedicine) ระบบสื่อสาร (Communications) และ ระบบควบคุม (Control) เป็นต้น ซึ่งระบบเหล่านี้จะทำการประมาณ พารามิเตอร์ต่าง ๆ ในระบบออกมา ยกตัวอย่างเช่น ในระบบเรดาร์ ซึ่งใช้สำหรับหาตำแหน่งของเครื่องบินในสนามบิน ดังแสดงในรูปที่ 4.1



รูปที่ 4.1 ระบบเรดาร์

จากรูป เพื่อที่จะหาระยะ R จะเริ่มจากการส่งพัลส์ในรูปของสนามแม่เหล็กไฟฟ้าออกไป เมื่อพัลส์เดินทางไปกระทบกับเครื่องบิน ก็จะเกิดการสะท้อนกลับ แต่พัลส์ที่สะท้อนกลับนี้จะถูกหน่วงเวลาไปเป็น τ_0 วินาที ดังนั้น ระยะ R จะสามารถหาได้จากสมการ คือ

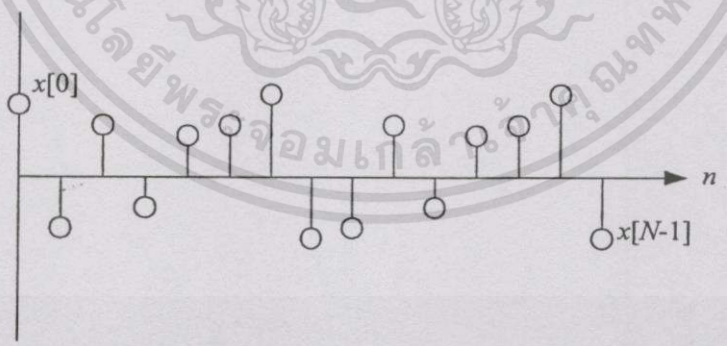
$$R = \frac{\tau_0 c}{2} \tag{4.1}$$

เมื่อ c คือความเร็วของคลื่นแม่เหล็กไฟฟ้าที่เดินทางไปมีค่า 3×10^8 m/s จะเห็นได้ชัดว่าถ้าสามารถวัดค่า τ_0 ได้ จะสามารถหาระยะห่างระหว่างสนามบินกับเครื่องบินได้อย่างแน่นอน จากรูปที่ 4.1 (b) เป็นสัญญาณที่รับได้ สังเกตเห็นว่า พัลส์จะมีขนาดที่ลดลง เนื่องจากเกิดการสูญเสียจากการเดินทาง นอกจากนี้ยังถูกสัญญาณที่ไม่ต้องการเข้ามารบกวนอีกด้วย นอกจากนี้อุปกรณ์อิเล็กทรอนิกส์ภายในเครื่องรับ ก็จะมีผลทำให้เกิดการหน่วงของสัญญาณเพิ่มขึ้นได้เหมือนกัน ดังนั้น การที่จะทราบค่า τ_0 นั้น เฉพาะการดึงกำลังงานจากเครื่องรับมาทำการวัดนั้นคงไม่เพียงพอ ทั้งนี้เพราะจะต้องคิดตัวแปรอื่น ๆ ที่มีผลต่อสัญญาณที่รับได้ด้วย

ในบทนี้ จะกล่าวถึงทฤษฎีการประมาณค่าพารามิเตอร์ในระบบต่าง ๆ ซึ่งเป็นพื้นฐานของการประมวลผลสัญญาณแบบอะแดปทีฟ ซึ่งจะมีกล่าวถึงในบทต่อไป

4.1 หลักการประมาณค่า

สมมติว่ามีสัญญาณ $x[n]$ ที่ได้จากการวัด N ค่า ดังรูปที่ 4.2



รูปที่ 4.2 สัญญาณที่ได้จากการวัด

เซตของการวัดที่ได้นี้ อาจมีพารามิเตอร์บางตัว ที่ต้องการทราบ เช่น ค่า mean เป็นต้น ดังนั้นในกรณีนี้ค่า mean จึงเป็นค่าที่ต้องการประมาณ (estimate) ถ้าแทนพารามิเตอร์ที่ต้องการ

ประมาณด้วย θ ส่วนฟังก์ชันที่จะใช้หาค่าการประมาณของ θ เรียกว่า estimation หรือใช้สัญลักษณ์ $g(\cdot)$ และสำหรับเซตของ $x[n]$ จะได้ว่า

$$\hat{\theta} = g(x[0], x[1], ..., x[N-1]) \tag{4.2}$$

$\hat{\theta}$ คือ ค่าประมาณของ θ ซึ่งเป็นค่าเฉพาะของแต่ละเซตของ $x[n]$ เพื่อให้เกิดความกระจ่างในการแยกความหมายของค่า estimate($\hat{\theta}$) และ estimator($g(\cdot)$) จะแสดงตัวอย่างดังข้างล่าง

สมมติว่า พารามิเตอร์ที่ต้องการทราบคือ ค่า mean ดังนั้นค่าประมาณของ mean ก็คือ $\hat{\mu}$ กำหนดได้ดังนี้คือ

$$\hat{\mu} = \frac{1}{N}\sum_{n=0}^{N-1} x[n] \tag{4.3}$$

ซึ่งก็หมายความว่า estimator, $g(\cdot)$ ก็คือ การรวม (sum) ทุก ๆ ค่าของ $x[n]$ และหารด้วย N นั่นคือ

$$g(\cdot) = \frac{1}{N}\sum_{n=0}^{N-1}(\cdot) \tag{4.4}$$

ดังนั้นสรุปได้ว่า $\hat{\mu}$, ค่าประมาณของ μ, เป็นค่าที่ได้จากการใช้ฟังก์ชัน $g(\cdot)$, ซึ่งเป็นตัวประมาณ ซึ่งเป็นฟังก์ชันทางคณิตศาสตร์ กระทำกับเซตของการวัดหนึ่ง ๆ

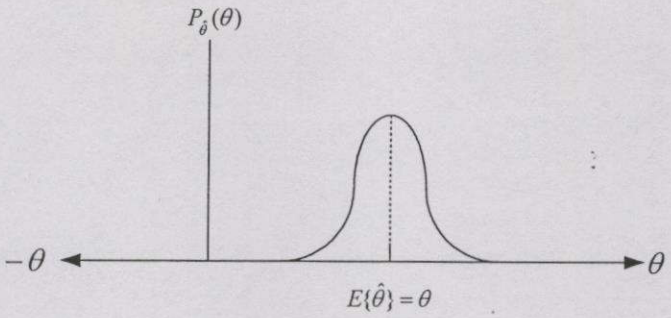
ค่า $x[n]$ ที่วัดได้นั้นเป็นสัญญาณสุ่ม ดังนั้นค่าที่ประมาณได้จึงเป็นค่าสุ่มด้วย หากให้ $x[n]$ มีการ distribution แบบ Gaussian แล้วจะได้ Pdf ของ $\hat{\theta}$ เป็น Gaussian เช่นกัน

สมมติว่า บังเอิญทราบค่าในเซตจากค่าที่วัดได้ คือ {$x[0]$, $x[1]$,..., $x[N\text{-}1]$} ค่าที่ได้นี้จะมีค่า mean เป็น θ ซึ่งเป็นค่าจริงของการวัดนี้ ในการประมาณค่า ก็จะมีความหวังว่า จะได้ค่าประมาณเฉลี่ย หรือ $E\{\hat{\theta}\}$ เท่ากับค่า θ ด้วย ดังแสดงตามรูปที่ 4.3 และหาก $E\{\hat{\theta}\} = \theta$ จะเรียกตัว estimator ว่าเป็น unbiased estimator

นอกจากนี้แล้วค่า $P_{\hat{\theta}}(\theta)$ ควรจะมีการกระจายที่น้อยนั่นคือค่า variance ของค่าที่ประมาณได้จะต้องน้อยมาก ๆ ดังนั้นสรุปได้ว่า estimator ที่ดีก็ควรจะ

i) เป็น unbiased

ii) variance ต่ำ



รูปที่ 4.3 Pdf ของพารามิเตอร์ที่ประมาณได้

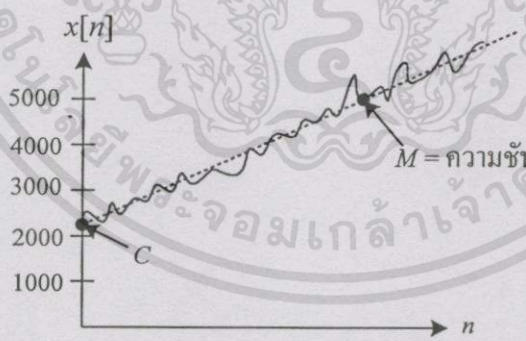
4.2 ชนิดของการประมาณ

การประมาณค่าจะสามารถแบ่งออกได้เป็นสองแบบคือ

i) Classical Estimation พารามิเตอร์ที่ไม่ทราบค่าเป็น deterministic

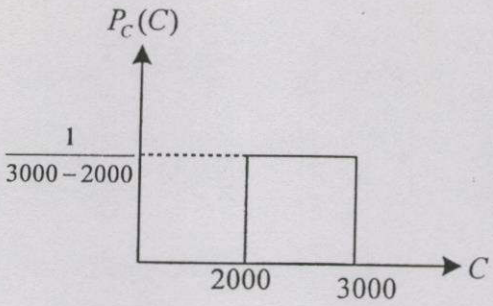
ii) Bayesian Estimation พารามิเตอร์ที่ไม่ทราบค่าเป็น random นั่นคือค่า Pdf ของ พารามิเตอร์ที่ไม่ทราบค่า ต้องบอกมาล่วงหน้า (a priori) พิจารณาตัวอย่างต่อไปนี้

มีความต้องการประมาณลักษณะของ index ก็คือต้องการประมาณค่า M และ C ดังรูปที่ 4.4



รูปที่ 4.4 ค่า C และ M ที่ต้องการประมาณ

ในกรณีของ Bayesian Estimation นั้น ทั้ง M และ C เป็น random ซึ่งหากเขียน Pdf ของ C หรือ $P_C(C)$ จะได้ดังรูปที่ 4.5 ส่วนของ $P_M(M)$ ก็จะได้เช่นเดียวกัน



รูปที่ 4.5 Pdf ของ C (ของ M ก็เช่นเดียวกัน)

ดังนั้น สำหรับ Bayesian estimation จะได้ joint Pdf สำหรับ $\underline{x}$ และ θ เป็น

$$P(\underline{x},\theta) = P(\underline{x}|\theta) \cdot P(\theta) \tag{4.5}$$

โดยที่ θ เป็นพารามิเตอร์ที่ไม่ทราบค่า สังเกตเครื่องหมาย "," ระหว่าง $\underline{x}$ และ θ ใน $P(\underline{x},\theta)$ จะแตกต่างจาก $P(\underline{x};\theta)$ เพราะในกรณี Bayesian ทั้ง $\underline{x}$ และ θ ต่างก็เป็น random ในกรณีการหา $P(\underline{x},C)$ จะได้ว่า

$$P(\underline{x},C) = P(\underline{x}|C) \cdot P_C(C) \tag{4.6}$$

ในทำนองเดียวกัน

$$P(\underline{x},M) = P(\underline{x}|M) \cdot P_M(M) \tag{4.7}$$

4.3 คุณภาพของตัวประมาณ

เมื่อได้ตัวประมาณมาแล้ว สิ่งที่ต้องการต่อไปคือ performance ของมัน ดังที่ได้กล่าวมาแล้วว่า ข้อมูลข่าวสารทางสถิติ ของค่าประมาณ คือ Bias และ Variance โดยตัวประมาณค่าที่ดีจะต้องเป็น unbiased และมี Variance น้อย ๆ

สมมติว่ามีข้อมูล $x[n]$ เป็น

$$x[n] = A + w[n] \tag{4.8}$$

เมื่อ

$$w[n] \approx N(0, \sigma_w^2)$$
$$x[n] = \{x[0], x[1], ..., x[N-1]\}$$

หากให้ตัวประมาณเป็น mean ก็จะได้ว่า

$$\hat{A} = \frac{1}{N}\sum_{n=0}^{N-1} x[n] \tag{4.9}$$

ซึ่งค่า mean ของ $x[n]$ คือ $E\{x[n]\}$ หรือ

$$\begin{aligned} E\{x[n]\} &= E\{A\} + E\{w[n]\} \\ &= E\{A\} = A \end{aligned} \tag{4.10}$$

และหากมีตัวประมาณ $\hat{A}'$ อีกหนึ่งตัว โดยที่ $\hat{A}' = x[0]$, ซึ่งก็คือ การเลือกเฉพาะ $x[0]$ เป็น ตัวประมาณ ทั้งการวัด (measurement) จะได้ว่า ค่า mean ของ $x[0]$ คือ

$$E\{x[0]\} = A \tag{4.11}$$

นั่นคือ

$$E\{\hat{A}'\} = E\{\hat{A}\} = A \tag{4.12}$$

ซึ่งจะพบว่า ตัวประมาณค่าทั้งสองให้ค่า mean ที่เท่ากัน แต่หากพิจารณาค่า variance ในกรณี $\hat{A}$ ค่า $\text{var}\{\hat{A}\}$ ก็คือ

$$\begin{aligned} E\{\hat{A}^2\} &= \text{var}\left\{\frac{1}{N}\sum_{n=0}^{N-1} x[n]\right\} \\ &= E\left\{\left(\frac{1}{N}\sum_{n=0}^{N-1} x[n]\right)\left(\frac{1}{N}\sum_{n=0}^{N-1} x[n]\right)\right\} \end{aligned} \tag{4.13}$$

หาก $w[n]$ เป็นอิสระทางสถิติ (Independent, Identically Distributed random variables :IID) จะได้ว่า

$$E\{\hat{A}^2\} = \frac{1}{N^2}\sum_{n=0}^{N-1} E\{x^2[n]\}$$
$$= \frac{1}{N^2} N\sigma^2 = \frac{\sigma^2}{N} \qquad (4.14)$$

ส่วนกรณีของ $\hat{A}'$ ค่า $\text{var}\{\hat{A}'\}$ คือ

$$E\{\hat{A}^{2'}\} = \text{var}\{x[0]\}$$
$$= E\{x^2[0]\} \qquad (4.15)$$
$$= \sigma^2$$

นั่นคือ ค่า $\text{var}\{\hat{A}\}$ มีค่าน้อยกว่า $\text{var}\{\hat{A}'\}$ แต่อย่างไรก็ตาม $\hat{A}$ ก็จะมีความซับซ้อนในการคำนวณมากกว่า $\hat{A}'$ เช่นกัน

ประเด็นสำคัญของการทำการประมาณคือ

i) เนื่องจากผลที่ได้มาจากการใช้ตัวประมาณกับเซตของข้อมูลที่วัดได้นั้น เป็น random ดังนั้นการทดสอบ performance ก็ขึ้นกับรูปแบบ random ด้วย เช่น อาจดูได้จาก Pdf เป็นต้น

ii) performance ของการประมาณได้มาจากการคำนวณโดยตรง ไม่สามารถใช้การ simulation ได้

iii) performance และ ความซับซ้อนในการคำนวณเป็น trade-off ซึ่งกันและกัน ดังนั้นในบางครั้ง ตัวประมาณค่าที่ใช้ก็อาจจะไม่ใช่ตัวประมาณค่าที่ดีที่สุด (Optimal Estimator)

สรุปได้ว่า การจะดูว่าตัวประมาณค่าตัวใด มีประสิทธิภาพดีนั้น จะดูได้จากข้อมูลทางสถิติ ก็คือ Bias หรือ Variance ถ้าหากตัวประมาณนั้นเป็นแบบ Unbiased อยู่แล้ว ก็จะหันมาพิจารณาที่ Variance อย่างเดียว

เกณฑ์ในการพิจารณา Variance ของตัวประมาณนั้นว่าน้อยที่สุด (Minimum Variance) ว่าเป็นเท่าใดนั้น จะใช้ Cramer Rao Bound (CRB) [34] เป็นตัวเปรียบเทียบ นั่นหมายความว่า หากหา minimum variance ของตัวประมาณออกมาได้แล้วได้เท่ากับ CRB ก็จะกล่าวว่า ตัวประมาณตัวนั้นเป็น Minimum Variance Unbiased Estimator (MVUE)

จากตัวอย่างในเรื่องของการหา variance ของตัวประมาณ ของค่า mean จากสัญญาณ $x[n] = A+w[n]$ โดยที่ $w[n] \approx N(0,\sigma^2)$ พบว่า มีความจำเป็นจะต้องทราบ Pdf หรือ Joint Pdf, $P(\underline{x};\theta)$ ก่อน นั่นหมายความว่า การที่จะได้ MVUE จะต้องทราบ Pdf

ในทางปฏิบัติ บางครั้งไม่ทราบ Pdf แต่ทราบ mean และ autocorrelation function เท่านั้น เมื่อเป็นอย่างนี้จะต้องใช้ตัวประมาณอีกแบบ ที่เรียกว่า Best Linear Unbiased Estimator (BLUE) หรือถ้าหากไม่ทราบข้อมูลทางสถิติใด ๆ เลย ก็จะยังคงสามารถหาตัวประมาณได้ด้วย Least Square Estimator (LSE)

4.4 ชนิดของตัวประมาณ

ชนิดของตัวประมาณสามารถแบ่งออกได้เป็น 2 ประเภทหลัก คือ

4.4.1 Block-Based Estimator

หมายความว่า จะต้องมีข้อมูลทั้งหมดก่อนจึงจะทำการประมาณ เช่น สำหรับ Sample mean estimator นั่นคือ

$$\hat{\mu} = \frac{1}{N}\sum_{n=0}^{N-1} x[n] \tag{4.16}$$

4.4.2 Sequential Estimator

เป็นการประมาณโดยอาศัยข้อมูลเก่า หรือ อาจกล่าวได้ว่าเป็นการประมาณที่เวลา N เป็นฟังก์ชันของข้อมูลก่อนหน้า N นั่นคือ

$$\hat{\mu}(N) = f(\hat{\mu}(N-1)) \tag{4.17}$$

เช่น สำหรับ sample mean ดังตัวอย่างที่แล้ว หากตัวประมาณเป็นแบบ sequential และ จะพิจารณา $\hat{\mu}(N)$ จะได้

$$\begin{aligned}\hat{\mu}(N) &= \frac{1}{N+1}\sum_{n=0}^{N} x[n] \\ &= \frac{1}{N+1}\left(\sum_{n=0}^{N-1} x[n] + x[N]\right)\end{aligned}$$

สังเกตว่า $\sum_{n=0}^{N-1} x[n]$ นั้นคือ $\hat{\mu}(N-1)$ ดังนั้น หากแยกพจน์ทั้งสองออกจากกันจะได้

$$= \frac{N}{N+1}\left(\frac{1}{N}\sum_{n=0}^{N-1} x[n]\right) + \frac{x[N]}{N+1}$$

ทราบว่า $\frac{N}{N+1} = 1 - \frac{1}{N+1}$ ดังนั้นจะได้ว่า

$$\begin{aligned} &= \left(1 - \frac{1}{N+1}\right)\hat{\mu}(N-1) + \frac{1}{N+1}x[N] \\ &= \hat{\mu}(N-1) + \frac{1}{N+1}(x[N] - \hat{\mu}(N-1)) \end{aligned} \tag{4.18}$$

จะสังเกตเห็นว่าพจน์สุดท้ายขวามือคือ $(x[N] - \hat{\mu}(N-1)$ ในสมการที่ (4.18) นั้น เป็นความแตกต่างของ $x[N]$ ซึ่งเป็นค่าใหม่ กับ $\hat{\mu}(N-1)$ ซึ่งเป็นค่า mean ของเวลาก่อนหน้า ส่วน $\frac{1}{N+1}$ เป็น gain factor สำหรับความแตกต่างนี้ หากเขียนสมการข้างบนในรูปแบบทั่ว ๆ ไป จะได้

$$New\ value = Old\ value + gain[f(error)] \tag{4.19}$$

error นั้นตามทฤษฎีของการประมาณ มีความหมายว่าเป็น Innovation ในขณะที่ f ในตัวอย่างนี้เป็น direct function สมการนี้เป็นสมการพื้นฐานของ Adaptive Filtering โดยที่ อัลกอริทึมทางอะแดปทีฟฟิลเตอร์อยู่ด้วยกันหลายแบบ เช่น

i) Least Mean Square (LMS)

ii) Normalised LMS (NLMS)

iii) Recursive Least Square (RLS)

iv) Kalman filter

โดยสิ่งที่เป็นตัวกำหนดปัจจัยในการเลือกใช้อัลกอริทึมมีอยู่ 4 ปัจจัยดังนี้

i) ความเร็วในการหาคำตอบ

อัลกอริทึมต่างแบบ จะให้ความเร็วในการหาคำตอบต่างกัน โดยทั่วไป RLS จะเร็วกว่า LMS

ii) ความถูกต้องของคำตอบ

ก็คือ variance และ คำตอบขณะจำนวนค่าในการคำนวณเข้าสู่อนันต์

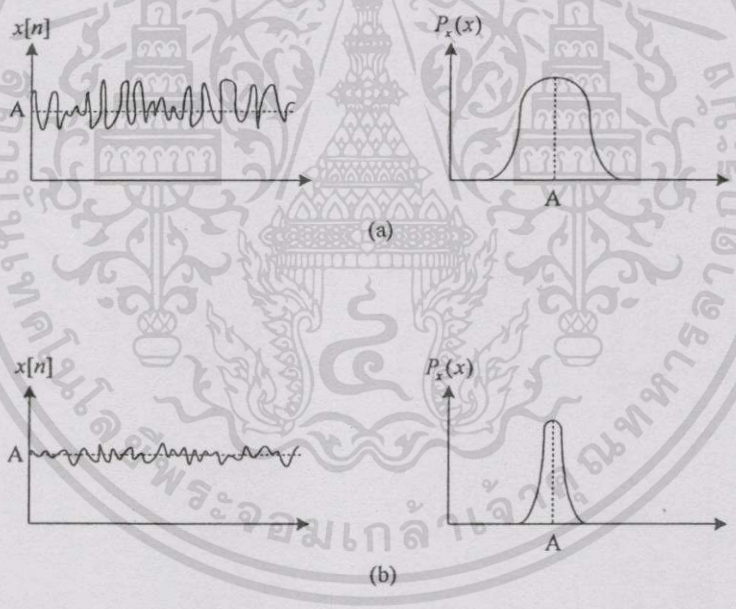
iii) ความสามารถในการติดตามสัญญาณ Non-stationary

iv) ความซับซ้อนในการคำนวณ

จากที่ได้กล่าวมาข้างต้นว่า ตัวประมาณค่านั้นมีหลายแบบ การจะเลือกใช้แบบในนั้นจะขึ้นอยู่กับลักษณะงานและข้อมูลทางสถิติ เนื่องจากตัวประมาณค่านั้นมีหลายแบบ และแต่ละแบบก็จะมีการวิเคราะห์ที่ซับซ้อน ซึ่งสามารถดูรายละเอียดได้จากเอกสารอ้างอิง [34]

4.5 Cramer-Rao Lower Bound (CRLB)

เครเมอร์-เรโอ บาวนด์ เป็นการหาขอบเขตค่า variance ที่น้อยที่สุด ที่จะเป็นไปได้สำหรับ unbiased estimator จากที่ทราบมาแล้วว่า ตัวประมาณค่า จะดีแค่ไหนนั้นขึ้นอยู่กับ variance ของสัญญาณรบกวน ตัวอย่างเช่นในการประมาณค่า mean ของสัญญาณ $x[n]$ ซึ่งมีสัญญาณรบกวน $w[n]$ ปนอยู่หรือ $x[n] = A + w[n]$, $w[n] \approx N(0,\sigma^2)$ จะได้ว่า หาก variance ของสัญญาณรบกวนมีค่ามาก จะได้ $x[n]$ และ Pdf ของ $x[n]$ เป็นดังรูปที่ 4.6 (a) และถ้า variance ของสัญญาณรบกวนมีค่าน้อยก็จะได้ดังรูปที่ 4.6 (b) จะเห็นว่า ในกรณีที่ variance น้อย ค่า $P_x(x)$ จะแคบ ดังนั้น CRLB นั้นจะอาศัยการดูที่ Pdf ของ $x[n]$ ว่าแคบเพียงใด

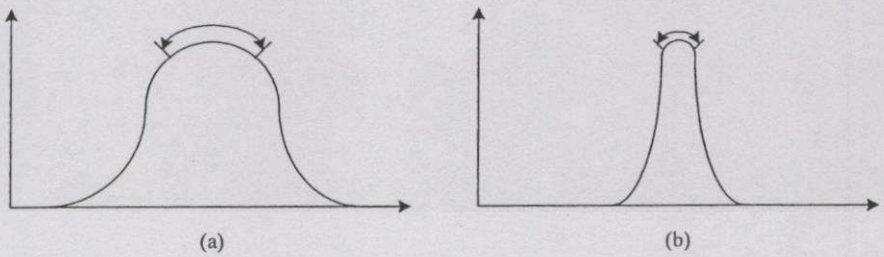


รูปที่ 4.6 สัญญาณรบกวนและ Pdf ของมัน

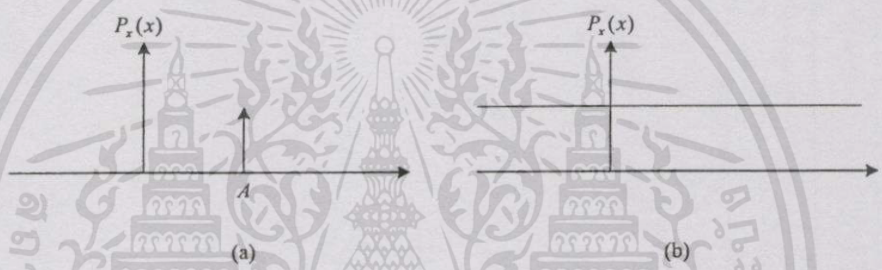
การวัดความแคบของ Pdf จะอาศัยสิ่งที่เรียกว่า Curvature ของ Pdf ที่ค่าจริง (true value) ดังแสดงในรูปที่ 4.7 (a) variance มาก และ (b) variance น้อย สรุปก็คือ ยิ่ง Curvature มีค่ามากเท่าใด ค่าที่ประมาณได้ก็ยิ่งใกล้เคียงกับค่าจริงมากขึ้นเท่านั้น

หาก $\sigma^2 = 0$ จะได้ว่า Pdf เป็นอิมพัลส์ (impulse) ที่ค่าจริง ดังในรูปที่ 4.8(a) ซึ่งกรณีนี้ Curvature มีค่าเป็นอนันต์ หรือได้ความถูกต้องสูงมาก แต่หาก Pdf เป็นลักษณะราบเรียบตลอด ดังในรูปที่ 4.8(b) กรณีนี้ค่า Curvature มีค่าเป็นศูนย์ หรือความถูกต้องไม่มีเลย นั่นหมายความว่า ค่า

variance ของตัวประมาณจะแปรผกผันกับค่า Curvature ซึ่ง Curvature นี้จะหาได้จาก การคำนวณค่าลบ ของอนุพันธ์อันดับสอง ของ log-likelihood function ที่ true parameter θ



รูปที่ 4.7 Curvature ของ Pdf



รูปที่ 4.8 (a) Curvature มีค่าเป็นอนันต์ (b) Curvature มีค่าเป็นศูนย์

Likelihood function คือ ค่า Pdf ของ actual measurement ใด ๆ เช่น $x[0]$ หรือก็คือ

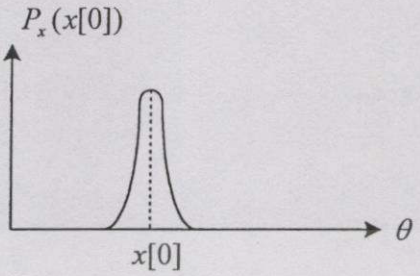
$$P_x(x)\Big|_{x=x[0]}$$

นั้นคือ จะแทน x ซึ่งเป็นสัญญาณใด ๆ ทั่วไปด้วย $x[0]$ ซึ่งเป็นค่า actual measurement ค่า Pdf ขณะนี้ ซึ่งเป็น likelihood function แล้วจะเป็นดังในรูปที่ 4.9

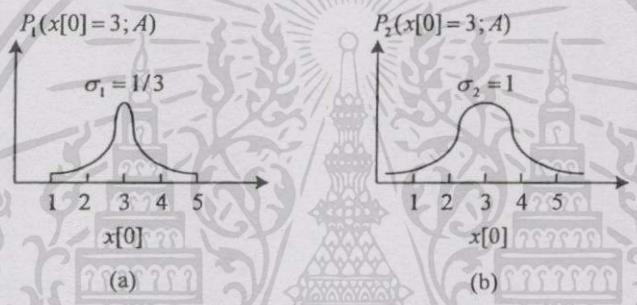
ตัวอย่างก็คือ หากทำการพิจารณา $x[0] = A+w[0]$ โดยที่ $w[n] \approx N(0,\sigma^2)$ และต้องการประมาณค่า A จะได้ว่า ค่าประมาณ $\hat{A} = x[0]$ และจะเห็นว่า ตัวประมาณค่านี้จะเป็นแบบ unbiased และมีค่า variance เป็น σ^2 ดังนั้นจะได้ likelihood function เป็นดังสมการ คือ

$$P_i(x[0];A) = \frac{1}{\sqrt{2\pi\sigma_i^2}}\exp\left[-\frac{1}{2\sigma_i^2}(x[0]-A)^2\right] \qquad (4.20)$$

หากกำหนดให้ $x[0]$ เป็นค่าคงที่ เช่น $x[0] = 3$ ให้ $i = 1, 2$ เพื่อแสดงว่าเมื่อ $\sigma_1 = 1/3$ และ $\sigma_2 = 1$ จะทำให้เกิด likelihood function 2 แบบ ดังรูปที่ 4.10



รูปที่ 4.9 Pdf ของ $x[0]$



รูปที่ 4.10 Likelihood function (a) เมื่อ $\sigma_1 = 1/3$ (b) เมื่อ $\sigma_2 = 1$

ในกรณีรูปที่ 4.10 (a) เนื่องจาก variance ต่ำ จึงทำให้ คำตอบที่ได้มีความถูกต้องมากกว่าแบบใน (b) ซึ่ง variance สูงว่า และจะสามารถหา Curvature ได้ดังนี้ คือ

$$\ln(P(x[0];A)) = -\ln(\sqrt{2\pi\sigma^2}) - \frac{1}{2\sigma^2}(x[0]-A)^2 \tag{4.21}$$

จากนั้นหา $\frac{\partial}{\partial A}$ ทั้งสองข้างของสมการที่ (4.21) จะได้

$$\frac{\partial \ln(P(x[0];A)}{\partial A} = \frac{1}{\sigma^2}(x[0]-A) \tag{4.22}$$

และได้ค่า Curvature เป็น

$$-\frac{\partial^2 \ln P(x[0];A)}{\partial A^2} = \frac{1}{\sigma^2} \tag{4.23}$$

พบว่า

$$\text{Curvature} = \frac{1}{\sigma^2} \tag{4.24}$$

และก็ทราบว่า ค่า variance ของตัวประมาณมีค่าเป็น σ^2 หรือจะได้ว่า

$$\text{var}\{\hat{A}\} = \frac{1}{-\dfrac{\partial^2 \ln P(x[0];A)}{\partial A^2}} \tag{4.25}$$

แต่โดยทั่วไปแล้วค่าอนุพันธ์อันดับสองนั้นขึ้นอยู่กับ $x[0]$ ซึ่งเป็น random ดังนั้นจะใช้ $E\{\cdot\}$ เพื่อทำให้ Curvature เป็น deterministic หรือ

$$I(A) = -E\left[\frac{\partial^2 \ln P(x[0];A)}{\partial A^2}\right] \tag{4.26}$$

ซึ่งหมายถึง ค่าเฉลี่ยของ Curvature ของ log-likelihood function ดังนั้นจะได้สมการสุดท้ายสำหรับคำนวณ CRLB ของตัวประมาณใด ๆ คือ

$$\text{var}\{\hat{A}\} \geq \frac{1}{I(A)} \tag{4.27}$$

หรือเขียนในรูปทั่วไป คือ

$$\text{var}\{\hat{\theta}\} \geq \frac{1}{I(\theta)} \tag{4.28}$$

โดยทั่วไปแล้ว สัญญาณมักจะมีรูปแบบตามสมการ คือ

$$x[n] = S[n;\theta] + w[n] \quad , n = 0,\ldots,N-1 \tag{4.29}$$

โดยที่ $S[n;\theta]$ นั้นอาจเป็นสัญลักษณ์ที่เป็นเชิงเส้นกับพารามิเตอร์บางตัว อย่างในกรณีที่ $S[n;\theta]$ เป็น $an+b, a\sin(2\pi fn)$ หรือ ar^n นั้น $S[n;\theta]$ เป็นเชิงเส้นกับ a แต่ในบางกรณี พารามิเตอร์ที่กำลังพิจารณา ก็อาจไม่ใช่พารามิเตอร์ที่เป็นเชิงเส้นก็ได้เช่น $S[n;\theta]$ เป็น $a\sin(2\pi fn+\phi)$ จะเห็นได้ชัดว่า f และ ϕ นั้นไม่เชิงเส้นกับ $S[n;\theta]$ จริง ๆ แล้วพารามิเตอร์ที่ต้องการประมาณไม่จำเป็นจะต้องเป็นเชิงเส้นก็ได้ ดังตัวอย่างต่อไปนี้

การหา CRLB ของตัวประมาณ $\hat{\theta}$ ของสัญญาณ $S[n;\theta]$ นั้นมีรูปแบบทั่ว ๆ ไปเป็น

$$\text{var}\{\hat{\theta}\} \geq \frac{\sigma^2}{\sum_{n=0}^{N-1}\left(\frac{\partial S[n;\theta]}{\partial\theta}\right)^2} \tag{4.30}$$

สังเกตว่า หาก σ^2 หรือ variance ของ $w[n]$ ไม่เปลี่ยนแปลง ก็หมายความว่าส่วนตัวหาร หรือ $\frac{\partial S[n;\theta]}{\partial\theta}$ ซึ่งคืออัตราส่วนการเปลี่ยนแปลงของสัญญาณ $S[n;\theta]$ เทียบกับ θ จะเป็นตัวกำหนดค่า variance ในสมการที่ (4.30) จะสรุปในแบบทั่ว ๆ ได้ว่า หากสัญญาณมีการเปลี่ยนแปลงอย่างรวดเร็ว ตามการเปลี่ยนแปลงค่าของพารามิเตอร์ที่ไม่ทราบค่า ที่กำลังประมาณ จะทำให้ได้ตัวประมาณที่ค่อนข้างเที่ยงตรง

เช่น หาก $\hat{\theta}$ นั้น เป็น $\hat{f}_o$ ซึ่งหมายถึงว่า จะกำลังทำการประมาณความถี่ f_o ของ $S[n;f_o]$ ซึ่งถูกบวกด้วย $w[n]$ และ $S[n;\theta]$ จะมีสมการเป็น

$$S[n;\theta] = A\cos(2\pi f_o n+\phi), 0<f<1/2 \tag{4.31}$$

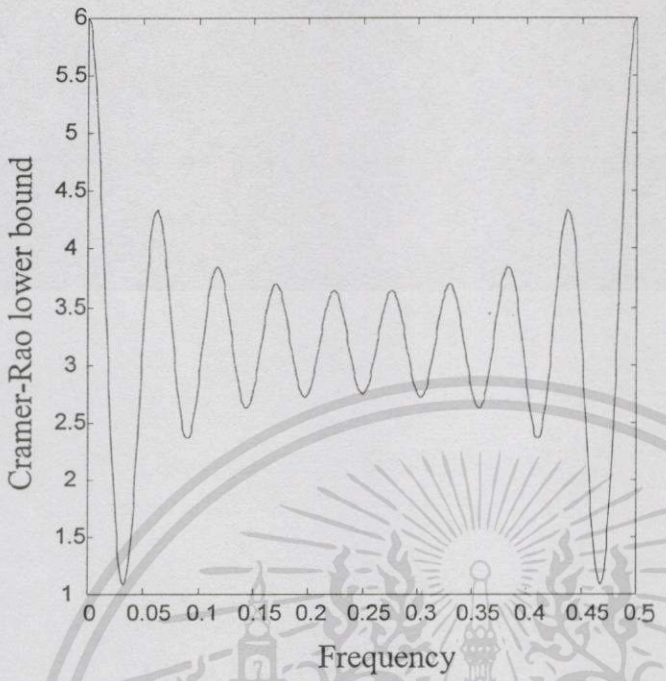
ทำการหาค่าอนุพันธ์ $\frac{\partial}{\partial f_o}$ โดยสมมติว่าทราบค่า A และ ϕ จะได้

$$\frac{\partial S[n;f_o]}{\partial f_o} = -2\pi n\sin(2\pi f_o n+\phi) \tag{4.32}$$

หรือได้ CRLB เป็น

$$v\{\hat{f}_o\} \geq \frac{\sigma^2}{A^2\sum_{n=0}^{N-1}[2\pi n\sin(2\pi f_o n+\phi)]^2} \tag{4.33}$$

หากพล็อตกราฟ var$\{\hat{f}_o\}$ เทียบกับ f จะได้ดังรูปที่ 4.11



รูปที่ 4.11 Cramer-Rao lower bound สำหรับการประมาณความถี่สัญญาณซายน์

จากรูปจะเห็นว่า var$\{\hat{f}_o\}$ จะมากที่สุดที่บริเวณความถี่ประมาณศูนย์ (DC) และที่ความถี่ Nyquist ส่วนที่ความถี่อื่น ๆ ค่า var$\{\hat{f}_o\}$ ก็จะมีค่าสูง ๆ ต่ำ ๆ สลับกัน จะเห็นว่าประโยชน์ของ CRLB ก็คือ CRLB บอกถึงตำแหน่งที่ควรจะทำการประมาณความถี่ f_o ตลอดย่านความถี่ $0 < f < 1/2$

ตัวประมาณค่าแบบ MVUE เป็น optimal estimator การนำตัวประมาณค่านี้มาใช้ จำเป็นต้องทราบ Pdf ของข้อมูลที่วัดได้ก่อน ส่วน BLUE ก็เป็น optimal estimator แบบเชิงเส้น ที่ไม่จำเป็นต้องทราบ Pdf แต่ต้องใช้ 1st และ 2nd moments เพื่อการประมาณค่า ส่วนตัวประมาณค่าแบบ Least Square (LSE) ไม่จำเป็นต้องทราบข้อมูลทางสถิติใด ๆ ของข้อมูลที่วัดได้เลย ใช้แต่ Signal model ที่กำหนดขึ้นมาเพื่อหาตัวประมาณค่าเท่านั้น ข้อเสียของ LSE ก็คือ ไม่มีคุณสมบัติเป็น optimal estimator ดังเช่น MVUE และ BLUE ในทางปฏิบัติ เป็นการยากที่จะทราบทั้ง Pdf, 1st และ 2nd moments ของข้อมูลที่วัดได้ ดังนั้น ในทางปฏิบัติจึงนิยมใช้ตัวประมาณแบบ LSE ซึ่งจะได้กล่าวรายละเอียดต่อไป

4.6 ตัวประมาณค่าแบบ Least Square

กำหนด data model ของ x[n] เป็นดังสมการ คือ

$$x[n] = \theta S[n], n = 0,1,...,N-1 \tag{4.34}$$

เมื่อ θ คือพารามิเตอร์ที่ต้องการประมาณและ $S[n]$ เป็นสัญญาณแบบ deterministic เช่น 1, n, หรือ $\sin(2\pi fn)$ ซึ่งถูกส่งมาจาก signal model สังเกตว่า ในสมการของ data model, $x[n]$ จะไม่มีส่วนประกอบของสัญญาณรบกวน $w[n]$ เลย ดังนั้น model สำหรับใช้หาตัวประมาณ จึงใช้ในการประมาณค่าพารามิเตอร์เท่านั้น

ข้อตัดสินที่ใช้ในการประมาณค่าของพารามิเตอร์คือ หาค่าผิดพลาด (error) ระหว่าง data model, $x[n]$ และ signal model, $s[n]$

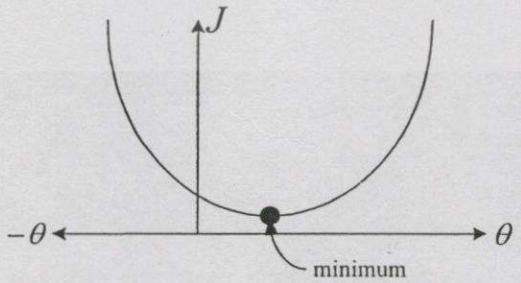
การหาตัวประมาณค่าแบบ LS จะใช้การ minimization ของ Cost function, J โดยที่

$$J = \sum_{n=0}^{N-1} e^2[n] \tag{4.35}$$

และค่า minimum ของ J คือ จุดที่ θ ทำให้ค่าอนุพันธ์ของ J เทียบกับ θ มีค่าเป็นศูนย์ นั่นคือ

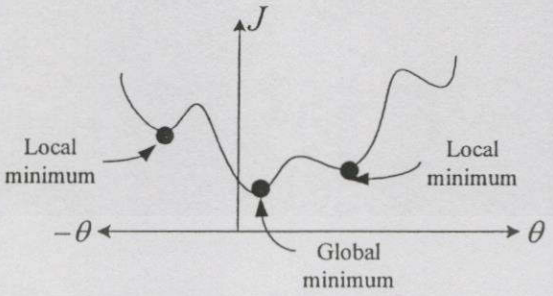
$$\frac{\partial J}{\partial \theta} = 0 \tag{4.36}$$

ถ้าให้ error function เป็นฟังก์ชันแบบเชิงเส้นจะได้ว่า $\frac{\partial^2 J}{\partial \theta^2}$ เป็นค่าบวกอย่างเดียว ดังนั้น จุดที่ θ ทำให้ $\frac{\partial J}{\partial \theta} = 0$ จะเป็นจุดต่ำสุดเพียงจุดเดียว ดังแสดงในรูปที่ 4.12



รูปที่ 4.12 Cost function ที่เป็นเชิงเส้นกับ θ

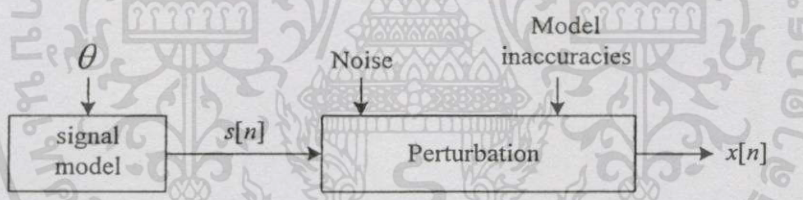
แต่ถ้า error function, $e[n]$ ไม่เป็นเชิงเส้น เช่น $e[n] = x[n] - \cos(2\pi n\theta)$ แล้วจะได้จุดต่ำสุดหลายจุด ดังแสดงในรูปที่ 4.13



รูปที่ 4.13 Cost function ที่ไม่เป็นเชิงเส้นกับ θ

ซึ่งขั้นตอนการประมาณก็จะยุ่งยากมากขึ้น ในที่นี้จะขอกล่าวเฉพาะ ตัวประมาณแบบเชิงเส้นเท่านั้น

จาก data model, $x[n]$ ซึ่งสัมพันธ์กับ signal model, $s[n]$ จะมีบล็อกไดอะแกรมแสดงดังรูปที่ 4.14



รูปที่ 4.14 ความสัมพันธ์ระหว่าง $x[n]$ กับ $s[n]$

$s[n]$ คือ สัญญาณแบบ deterministic ได้มาจาก signal model ซึ่งเป็นฟังก์ชันของ θ ส่วน $x[n]$ หรือ observed data นั้นคือ $s[n]$ ที่ถูกรบกวนด้วยสัญญาณรบกวน และความไม่สมบูรณ์ของ signal model การเลือก LSE ก็คือ การหาค่า $s[n]$ ที่ใกล้เคียงกับ $x[n]$ มากที่สุด โดยข้อตัดสินที่ใช้คือ ลดค่ากำลังสองของ error function ให้เหลือน้อยที่สุด ค่าผลรวมของ error function ยกกำลังสอง นั้นคือ $J(\theta)$ ดังนั้นจะได้ว่า

$$J(\theta) = \sum_{n=0}^{N-1} (x[n] - s[n])^2 \tag{4.37}$$

และจะได้ $\hat{\theta}$ เมื่อ $\hat{\theta}$ นั้นทำให้ $\frac{\partial J(\theta)}{\partial \theta} = 0$ สังเกตว่า ในการหา LSE นั้นไม่จำเป็นต้องทราบข้อมูลทางสถิติของข้อมูลที่วัดได้เลย

4.7 Sequential Least Square

จากที่ผ่านมา จะทำการหา LSE จากข้อมูลที่เป็น N samples ในบางกรณี (ส่วนใหญ่) จะได้ข้อมูลมาจาก sample ซึ่งถูกเก็บมาทีละ sample และต้องการหา LSE ในทันทีทันใด ซึ่งจะเรียกวิธีหา LSE จากข้อมูลที่เก็บไว้แล้ว จำนวน samples เรียกว่า Sequential Least Squares Estimator ส่วนวิธีหา LSE จากข้อมูลที่เก็บไว้แล้วจำนวน N samples เรียกว่า Bleck-based LSE ตัวอย่างเช่น ต้องการ ประมาณค่าระดับ DC ในสัญญาณ $x[n]$ โดยที่

$$x[n] = A + w[n] \, , n = 0,1,...,N-1 \tag{4.38}$$

หรือเขียนในรูป matrix

$$\underline{X} = \underline{A} + \underline{W} \tag{4.39}$$

และจัดอยู่ในรูป

$$\underline{X} = \underline{S} + \underline{W} \tag{4.40}$$

โดยที่

$$\underline{S} = \underline{H\theta} \tag{4.41}$$

เมื่อ

$\underline{\theta}$ = พารามิเตอร์ที่ต้องการประมาณ = $\underline{A}$

$\underline{H}$ = observation matrix = $\underline{1} = [11...1]^t$, (length N)

t = transpose จะได้ LSE คือ

$$\begin{aligned}\underline{\hat{\theta}}_{LS} &= (\underline{H}'\underline{H})^{-1}\underline{H}'\underline{X} \\ &= (\underline{1}'\underline{1})^{-1}\underline{1}'\underline{x} = \frac{\underline{1}'\underline{X}}{\underline{1}'\underline{1}} \\ &= \frac{1}{N}\sum_{n=0}^{N-1} x[n]\end{aligned} \tag{4.42}$$

$\underline{\hat{\theta}}_{LS}$ เป็นตัวประมาณของ DC สำหรับ $x[0], x[1], \ldots, x[\text{N-1}]$ ซึ่งจะเรียกว่า เป็น LSE ของเวลาล่าสุด หรือ ปัจจุบัน ซึ่งจะเขียนแทนด้วย $\hat{A}[N-1]$ ดังนั้น

$$\hat{A}[N-1] = \frac{1}{N}\sum_{n=0}^{N-1} x[n] \tag{4.43}$$

หากมีข้อมูลใหม่เข้ามาหรือเรียกว่าเป็น $x[N]$ จะได้ LSE ขณะที่เวลา N เป็น

$$\begin{aligned}\hat{A}[N] &= \frac{1}{N+1}\sum_{n=0}^{N} x[n] \\ &= \frac{1}{N+1}\left(\sum_{n=0}^{N-1} x[n] + x[N]\right) \\ &= \frac{N}{N+1}\sum_{n=0}^{N-1} x[n] + \frac{1}{N+1}x[N] \\ &= \frac{N}{N+1}\hat{A}[N-1] + \frac{1}{N+1}x[N]\end{aligned} \tag{4.44}$$

$\hat{A}[N]$ นั้นที่แท้แล้วก็คือ ผลรวมของค่าประมาณเดิม, $\hat{A}[N-1]$ และ ค่าที่วัดได้ใหม่ล่าสุด หากแยก factor ออกมา จะได้

$$\begin{aligned}\hat{A}[N] &= \frac{(N+1)}{(N+1)}\hat{A}[N-1] - \frac{1}{N+1}\hat{A}[N-1] + \frac{1}{N+1}x[N] \\ &= \hat{A}[N-1] + \frac{1}{N+1}(x[N] - \hat{A}[N-1])\end{aligned} \tag{4.45}$$

ซึ่งอยู่ในรูปแบบของ

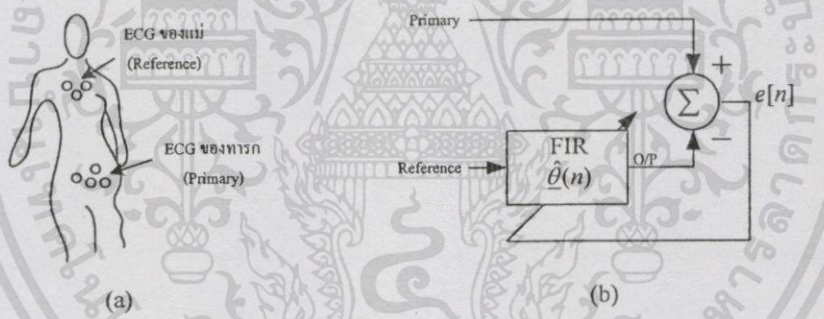
$$\textit{New Estimate} = \textit{Old Estimate} + \textit{Gain}(\textit{Error}) \tag{4.47}$$

หากค่า Sum Square Error (J) ที่มีค่าน้อยที่สุดจะได้ว่า

$$J_{\min}(N) = J_{\min}(N-1) + J_{\min}[error]$$
$$= J_{\min}(N-1) + \frac{1}{N+1}\sum_{n=0}^{N}(x[N] - \hat{A}[N-1])^2 \quad (4.48)$$

หากวิเคราะห์แล้วจะเห็นว่า สมการ $J_{\min}(N)$ ที่ได้นี้ น่าจะมีความถูกต้อง เนื่องจาก $J_{\min}(N)$ ซึ่งเป็นค่า LSE ของข้อมูล N+1 ค่า ควรจะต้องมากกว่า $J_{\min}(N-1)$ เสมอ ย้อนกลับไปดูสมการ $\hat{A}[N]$ พบว่า เทอม $\frac{1}{N+1}$ นั้นมีชื่อเรียกว่าเป็น Gain ซึ่ง Gain นี้จะเป็นตัวบอกถึง ความน่าเชื่อถือของค่าที่วัดได้ใหม่, x[N] ที่ได้มาว่าควรจะนำมาใช้คำนวณหา LSE หรือไม่

การประยุกต์ใช้งานของ Least Square ก็เช่น การแยกสัญญาณคลื่นไฟฟ้าหัวใจ (ECG) ของทารก (fetus) ออกจาก ECG ของแม่ (mother) ดังในรูปที่ 4.15 (a) และบล๊อกไดอะแกรมดังในรูปที่ 4.15 (b) ตามลำดับ



รูปที่ 4.15 การประยุกต์ใช้งาน LSE

ตัวกรอง FIR ซึ่งมีสัมประสิทธิ์เป็น $\hat{\underline{\theta}}(n)$ นั้น ซึ่งก็คือค่าประมาณของ ฟังก์ชันถ่ายโอนของหน้าท้องกับ อกของแม่นั่นเอง ส่วน $e[n]$ ที่ได้ จึงเป็นสัญญาณของทารก เพียงสัญญาณเดียว การประมาณค่าของ $\hat{\underline{\theta}}(n)$ ก็คือการ minimize ค่า Cost function, J ซึ่งในวิธีของ Least Square จะได้

$$J[N] = \sum_{k=0}^{N} e^2[k]$$
$$= \sum_{k=0}^{N} (primary - reference)^2 \quad (4.49)$$

และนี่คือ LS algorithm จริง ๆแล้ว มีอัลกอริทึมอยู่ 2 แนวทาง ที่ใช้กันทั่วไปในด้าน Adaptive Signal Processing คือ

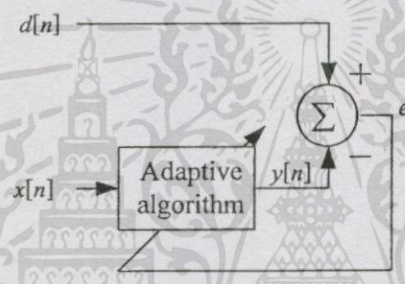
i) Least Mean Square (LMS)

ii) Recursive Least Square (RLS)

ซึ่งสิ่งที่เป็นตัวกำหนดการเลือกใช้ อัลกอริทึมมีอยู่ 4 ปัจจัยดังได้กล่าวมาแล้วข้างต้น ซึ่งในที่นี้จะกล่าวถึงเฉพาะ LMS อัลกอริทึมเท่านั้น

4.8 Least Mean Square (LMS) อัลกอริทึม

อัลกอริทึมนี้จะเป็นการ minimize ค่า Mean Square Error อย่างประมาณ (Approximately) บล๊อกไดอะแกรมซึ่งใช้อธิบายจะแสดงได้ดังรูปที่ 4.16



รูปที่ 4.16 บล๊อกไดอะแกรมของ LMS อัลกอริทึม

อะแดปทีฟอัลกอริทึมนั้นจะเป็นตัวกรองแบบ FIR ซึ่งมีสัมประสิทธิ์เป็นไปตามสมการ คือ

$$\underline{W} = \begin{bmatrix} w_o \\ w_1 \\ . \\ . \\ w_{N-1} \end{bmatrix} \tag{4.50}$$

ดังนั้น $e[n]$ ซึ่งเป็นสัญญาณผิดพลาดจะมีค่าตามสมการ คือ

$$e[n] = d[n] - \sum_{k=0}^{N-1} w[k]x[n-k] \tag{4.51}$$

สำหรับ LMS อัลกอริทึม ค่าพารามิเตอร์ของตัวกรอง FIR หาได้จากการ minimize ค่า J โดยที่ J เป็นค่า mean ของค่า error ยกกำลังสอง นั่นคือ

$$\begin{aligned} J &= E\{e^2[n]\} \\ &= E\{(d[n]-y[n])^2\} \\ &= E\{(d[n]-\underline{W}'\,\underline{X})^2\} \end{aligned} \tag{4.52}$$

เมื่อ $\underline{W}$ มีค่าตามสมการที่ (4.50) และ $\underline{X} = \begin{bmatrix} x[n] \\ x[n-1] \\ . \\ . \\ x[n-N+1] \end{bmatrix}$

หากกำหนด $x[n]$ และ $d[n]$ เป็น zero mean และเป็น WSS ดังนั้น J จะเป็น

$$J = E\{d^2[n]\} - 2E\{d[n]\underline{X}'\}\underline{W} + \underline{W}'E\{\underline{X}\,\underline{X}'\}\underline{W} \tag{4.53}$$

เมื่อ

$\{d^2[n]\}$ =Autocorrelation ของ $d[n] = \sigma_d^2$

$\{d[n]\underline{X}'\}$ = Crosscorrelation ของ $d[n]$ และ $\underline{X}' = \underline{P}'$

$\{\underline{X}\,\underline{X}'\}$ = Autocorrelation ของ $\underline{X} = R_{xx}$

ดังนั้นจาก (4.53) จะเขียนใหม่ได้ว่า

$$J = \sigma_d^2 - 2\underline{P}'\underline{W} + \underline{W}'R_{xx}\underline{W} \tag{4.54}$$

จากนั้นหา $\min J \rightarrow \dfrac{\partial J}{\partial \underline{W}} = 0$ จะได้ว่า

$$\begin{aligned} \frac{\partial J}{\partial \underline{W}} &= -2\underline{P} + 2R_{xx}\underline{W} \\ 0 &= -2\underline{P} + 2R_{xx}\underline{W}_{opt} \end{aligned} \tag{4.55}$$

$$\frac{\partial J}{\partial \underline{W}} = -2\underline{P} + 2R_{xx}\underline{W}$$
$$0 \quad = -2\underline{P} + 2R_{xx}\underline{W}_{opt} \tag{4.55}$$

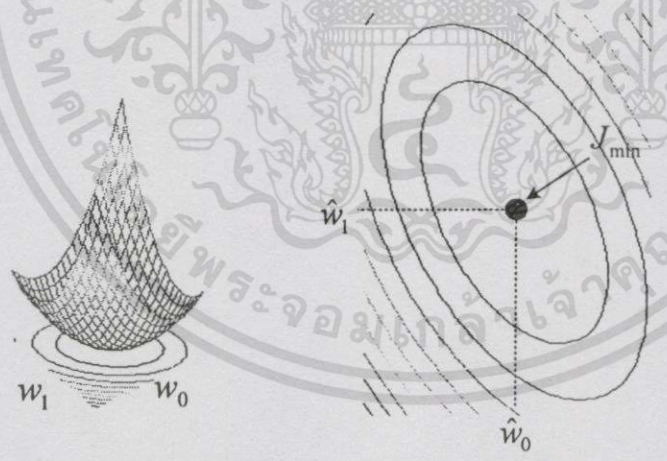
หรือจะได้ว่า

$$\underline{W}_{opt} = R_{xx}^{-1}\underline{P} \tag{4.56}$$

สมการนี้เรียกว่า Wiener Solution หากดูที่ 2nd derivative $\frac{\partial^2 J}{\partial \underline{W}^2}$ จะได้

$$\frac{\partial^2 J}{\partial \underline{W}^2} = 2R_{xx} \tag{4.57}$$

ซึ่งเป็น positive definite ดังนั้น $\frac{\partial J}{\partial \underline{W}} = 0$ จึงเป็นจุดต่ำสุด (minimum point) เช่น หาก สมมติให้ $\underline{W} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$ จะสามารถเขียนพื้นผิว (surface) ของ J ได้ดังรูปที่ 4.17



รูปที่ 4.17 พื้นผิวค่าผิดพลาดและกราฟ contour

หรือจะได้ $\underline{W}_{opt} = \begin{bmatrix} \hat{w}_0 \\ \hat{w}_1 \end{bmatrix}$ หากดูที่ $J_{\min}$ จะได้ว่า จากสมการที่ (4.54) เมื่อแทน $\underline{W}$ ด้วย $\underline{W}_{opt}$ จะได้

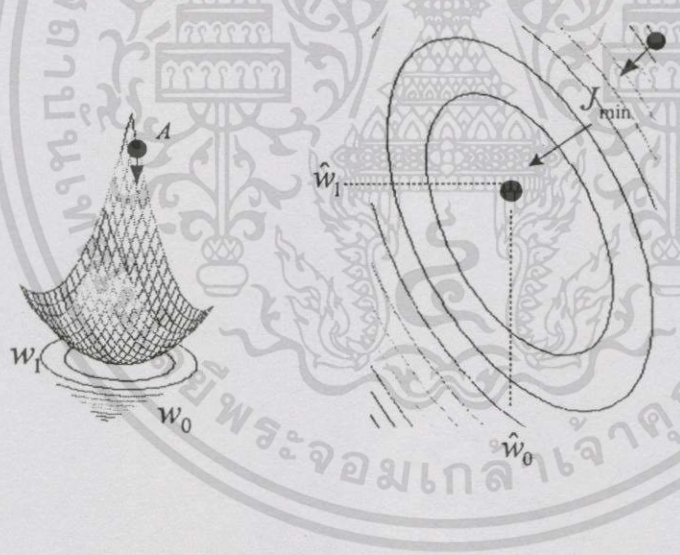
$$\begin{aligned} J &= \sigma_d^2 - 2\underline{P}' R_{xx}^{-1}\underline{P} + \underline{P}'(R_{xx}^{-1})' R_{xx} R_{xx}^{-1}\underline{P} \\ &= \sigma_d^2 - 2\underline{P}' R_{xx}^{-1}\underline{P} + \underline{P}'(R_{xx}^{-1})'\underline{P} \\ &= \sigma_d^2 - \underline{P}' R_{xx}^{-1}\underline{P} \\ &\vdots \end{aligned} \tag{4.58}$$

ข้อสังเกต $\underline{P}$ ซึ่งเป็น Cross correlation ของ $d[n]$ และ $x[n]$ ดังนั้นถ้า $\underline{P} = 0$ จะทำให้

$$J_{\min} = \sigma_d^2 \tag{4.59}$$

ซึ่งมีค่ามาก ดังนั้นหาก $x[n]$ ตัวใดที่ไม่มีความสัมพันธ์กับ $d[n]$ จะทำให้ error function มีค่ามาก ดังนั้นจะไม่มีการใช้ $x[n]$ นั้นในการหา $\underline{W}$

จะพบว่าประเด็นสำคัญของการหา Wiener Solution นั้นก็คือ จะต้องทราบค่าของ R_{xx} และ $\underline{P}$ เสียก่อน แต่โชคร้ายที่ในทางปฏิบัติมักจะไม่ทราบ ดังนั้นจึงต้องใช้วิธีการอื่น เพื่อช่วยหา $\underline{W}_{opt}$ จากการสังเกตพื้นผิวซึ่งเป็นรูปชาม หากเริ่มต้นหา $J_{\min}$ จาก A ตามรูปที่ 4.18



รูปที่ 4.18 การหาจุดต่ำสุดของพื้นผิวค่าผิดพลาดด้วยวิธี Steepest Descent

จะต้องอาศัยการเคลื่อนที่ไปในทิศทางขาลง หรือต้องใช้ negative gradient ของ J เขียนได้ด้วย $-\nabla J$ โดยที่

$$-\nabla J = -\frac{\partial J}{\partial \underline{W}} \tag{4.60}$$

ซึ่งวิธีการนี้จะเรียกว่า Steepest Descent algorithm การเคลื่อนที่จากจุด A จะเป็นไปทีละ step ซึ่งจะกำหนดขั้นของ step ด้วย μ ดังนั้นในแต่ละค่า n ที่เพิ่มขึ้น ค่า $\underline{W}$ จะถูกปรับ ดังนั้น

$$\underline{W}_{n+1} = \underline{W}_n - \mu \nabla J \Big|_{\underline{W}_n} \tag{4.61}$$

หากแทน $\nabla J = -2\underline{P} + 2R_{xx}\underline{W}_n$ จะได้

$$\begin{aligned} \underline{W}_{n+1} &= \underline{W}_n - \mu(-2\underline{P} + 2R_{xx}\underline{W}_n) \\ &= \underline{W}_n + 2\mu(\underline{P} - R_{xx}\underline{W}_n) \end{aligned} \tag{4.62}$$

ขั้นตอนต่อไปนี้เป็นขั้นตอนที่สำคัญที่สุดในการพัฒนาอัลกอริทึมที่เรียกว่า Least Mean Square โดยเริ่มจาก แทนที่จะหาค่า $E\{d\underline{X}\}$ และ $E\{\underline{X}\underline{X}'\}$ จะใช้การประมาณว่า $d\underline{X} = E\{d\underline{X}\}$ และ $\underline{X}\underline{X}' = E\{\underline{X}\underline{X}'\}$ หรือเขียนแทนด้วย

$$\hat{\underline{P}} = d\underline{X} \tag{4.63}$$

$$\hat{\underline{R}}_{xx} = \underline{X}\underline{X}' \tag{4.64}$$

ดังนั้นจากสมการที่ (4.62) จะกลายเป็น

$$\begin{aligned} \underline{W}_{n+1} &= \underline{W}_n + 2\mu(d\underline{X} - \underline{X}\underline{X}'\underline{W}_n) \\ &= \underline{W}_n + 2\mu\underline{X}(d - \underline{X}'\underline{W}_n) \end{aligned} \tag{4.65}$$

หรือจะได้สมการที่สำคัญสองสมการสำหรับ LMS อัลกอริทึม คือ

$$\underline{W}_{n+1} = \underline{W}_n + 2\mu\underline{X}\xi) \tag{4.67}$$

$$\xi = d - \underline{X}'\underline{W}_n \tag{4.68}$$

LMS อัลกอริทึม จะเป็นที่นิยมใช้กันมากที่สุดในทางปฏิบัติ เนื่องจากไม่ซับซ้อนในการคำนวณ

นอกจากอัลกอริทึมดังที่ได้กล่าวมาแล้ว ยังมีอัลกอริทึมอีกชนิดหนึ่งที่มีใช้กันในทางปฏิบัติ คือ Recursive Least Square (RLS) อัลกอริทึม ซึ่งโดยทั่วไปแล้ว RLS จะทำงานได้เร็วกว่า LMS แต่จะไม่ขอกล่าว ณ ที่นี้ รายละเอียดสามารถดูได้จากเอกสารอ้างอิง [19,36]

4.9 สรุป

ในบทนี้ ได้กล่าวถึงทฤษฎีทั่ว ๆ ไปเกี่ยวกับการประมาณค่าพารามิเตอร์ ที่พบได้ในระบบการประมวลผลสัญญาณต่าง ๆ เช่น ระบบเรดาร์ ระบบโซนา สัญญาณเสียง สัญญาณภาพ ทางการแพทย์ และ ระบบควบคุม เป็นต้น ตัวประมาณค่าที่ดีจะต้องมีคุณสมบัติสองประการคือ หนึ่งเป็น Unbiased และ สอง มีค่า Variance ต่ำ หรือเรียกว่า Minimum Variance Unbiased Estimator (MVUE) แต่ตัวประมาณค่านี้ จะต้องทราบ Pdf ของข้อมูลที่วัดได้ จึงจะสามารถคำนวณพารามิเตอร์ออกมาได้ ในกรณีที่ไม่ทราบ Pdf แต่ทราบ 1^{st} และ 2^{nd} moments ของสัญญาณที่วัดได้ก็จะสามารถใช้ตัวประมาณค่าแบบ Best Linear Unbiased Estimator (BLUE) แทนได้ และสุดท้าย ถ้าไม่ทราบอะไรเลย ก็จะต้องใช้ตัวประมาณแบบ Least Square (LS) ซึ่งต้องการเพียงแค่ signal model เท่านั้น อย่างไรก็ตาม ตัวประมาณค่าแบบ LS นี้จะไม่สามารถเรียกว่าเป็น Optimum Estimator เหมือนอย่าง MVUE และ BLUE ได้ ข้อตัดสินที่ใช้หาค่าพารามิเตอร์สำหรับ LS คือ การพิจารณาค่าผิดพลาดระหว่าง signal model, $s[n]$ กับ observation data, $x[n]$ ยกกำลังสองที่น้อยที่สุด ซึ่ง $s[n]$ นี้จะเป็นฟังก์ชันของพารามิเตอร์ที่จะประมาณ พารามิเตอร์ที่ทำให้ค่าความผิดพลาดระหว่างสัญญาณทั้งสองยกกำลังสองน้อยที่สุด ก็คือคำตอบของการประมาณนั้น ๆ ส่วนข้อตัดสินอีกอย่างหนึ่งที่ใช้คือ Least Mean Square (LMS) วิธีนี้จะตัดสินที่ค่าผิดพลาดเฉลี่ย ยกกำลังสองน้อยที่สุด จากการศึกษาพบว่าพื้นผิวค่าผิดพลาดของ ฟังก์ชันค่าผิดพลาด (error function) ที่เป็นเชิงเส้นกับพารามิเตอร์นั้นจะมีลักษณะเป็นรูปชาม การค้นหาจุดต่ำสุด (Global minimum) ของพื้นผิวนี้จะใช้หลักการของ Steepest Decent algorithm คือ ต้อง Negative Gradient ของ Cost function เทียบกับพารามิเตอร์ และให้เท่ากับศูนย์ วิธีการพิสูจน์ว่า เป็นจุดต่ำสุดหรือไม่นั้นจะดูที่ อนุพันธ์อันดับสองของ Cost function เทียบกับพารามิเตอร์ หากผลออกมาเป็น Positive definite ค่า Negative Gradient ก็เป็นจุดต่ำสุด อะแดปทีฟอัลกอริทึมแบบ LMS จะเป็นที่นิยมใช้กันมากที่สุด ทั้งนี้เพราะไม่ยุ่งยากในการคำนวณ ทำให้สามารถนำไปใช้ที่เวลาจริงได้สะดวก

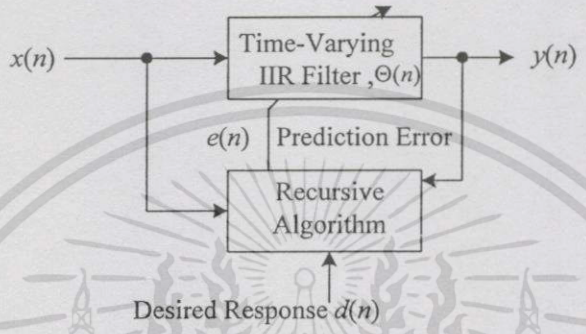
บทที่ 5
ตัวกรองความถี่แบบอะแดปทีฟ

ตัวกรองความถี่แบบอะแดปทีฟ (Adaptive Filtering) ได้เข้ามามีบทบาทเป็นอย่างมากในงานประมวลผลสัญญาณดิจิตอล และระบบควบคุม ทั้งนี้เพราะ ความสามารถปรับเปลี่ยนคุณลักษณะ (characteristics) ประจำตัวของระบบให้สอดคล้องกับสัญญาณที่รับเข้ามาได้ด้วยตัวเอง ซึ่งจะสามารถทำงานกับสัญญาณที่มีลักษณะเป็น nonstationary ซึ่งเป็นสัญญาณที่เกิดขึ้นจริงในทางปฏิบัติ ตัวกรองความถี่แบบอะแดปทีฟนั้นมีสองชนิด คือ ตัวกรองอะแดปทีฟแบบ FIR และแบบ IIR ตัวกรองอะแดปทีฟแบบ FIR จะปรับเฉพาะซีโร่เท่านั้น ทั้งนี้เพราะไม่มีโพล ส่วนตัวกรองอะแดปทีฟแบบ IIR จะปรับทั้งซีโร่ และโพลในเวลาเดียวกัน ตัวกรองอะแดปทีฟแบบ IIR ได้ถูกนำมาใช้ในงานต่าง ๆ แทนตัวกรองอะแดปทีฟแบบ FIR มากขึ้น โดยเฉพาะในงานที่เมื่อใช้ตัวกรองแบบ FIR ต้องใช้จำนวนอันดับ (order) มากโดยไม่จำเป็น เมื่อเปลี่ยนมาใช้ตัวกรองแบบ IIR แล้วจะสามารถลดอันดับลงได้ โดยที่สมรรถนะยังคงเดิม ตัวอย่างเช่น ในระบบประมวลผลสัญญาณ คือ Acoustic Echo Cancellation และ Data Equalization ในระบบควบคุมและหุ่นยนต์ คือ Modeling of Dynamical Systems เหตุผลที่สำคัญที่นำตัวกรองอะแดปทีฟ IIR มาใช้มากขึ้น คือ IIR จะมีลักษณะเป็น ฟังก์ชันแบบเศษส่วน (Rational function) ซึ่งสามารถทำ modeling ได้ดีกว่าแบบ FIR ที่มีลักษณะเป็นฟังก์ชันแบบโพลิโนเมียล (Polynomial function) อย่างไรก็ตาม ทฤษฎีสำหรับการวิเคราะห์ และสร้างตัวกรองอะแดปทีฟแบบ IIR ในปัจจุบันนี้ ยังอยู่ในกระบวนการทำวิจัย คือ ยังไม่สามารถเขียนออกมาเป็น ทฤษฎีที่เป็นสูตรสำเร็จได้ จึงทำให้การออกแบบและสร้างตัวกรองอะแดปทีฟแบบ IIR ตลอดจนการวิเคราะห์หาสมรรถนะ ของมัน ทำได้ค่อนข้างยาก

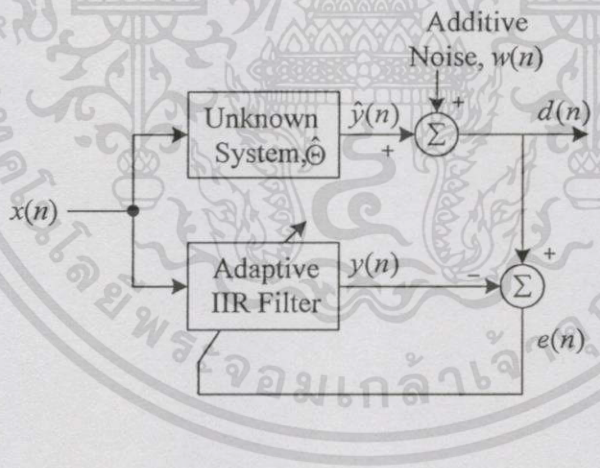
ในบทนี้จะกล่าวถึงทฤษฎีทั่วไปเกี่ยวกับตัวกรองความถี่อะแดปทีฟแบบ IIR โดยจะเน้นไปที่การหาสมการสำหรับค่าผิดพลาด (Equation error) ซึ่งสมการดังกล่าวจะถูกนำไปใช้สำหรับหาพื้นผิวค่าผิดพลาด (Error surface) และใช้กระบวนการทางอะแดปทีฟเพื่อค้นหาจุดต่ำสุดของพื้นผิวดังกล่าว ส่วนใหญ่แล้ว ตัวกรองความถี่อะแดปทีฟแบบ IIR จะใช้โครงสร้างแบบ Direct-form แต่โครงสร้างแบบ Direct-form จะมีข้อเสียในเรื่อง finite-precision effect และ การวิเคราะห์หาเสถียรภาพของตัวกรอง (complexity of stability monitoring) จากข้อเสียนี้เอง ทำให้มีการค้นหาโครงสร้างใหม่ ๆ เช่น โครงสร้างแบบ Lattice และ แบบ Parallel เป็นต้น

5.1 โครงสร้างของตัวกรองความถี่อะแดปทีฟแบบ IIR

รูปที่ 5.1 แสดงส่วนประกอบโครงสร้างทั่วไปของตัวกรองความถี่อะแดปทีฟแบบ IIR โดยมีสัญญาณอินพุต คือ $x(n)$ และสัญญาณเอาต์พุต คือ $y(n)$ จากรูป สังเกตได้ว่า จะประกอบด้วยตัวกรองความถี่แบบเปลี่ยนตามเวลา (Time-Varying Filter) โดยคุณลักษณะของตัวกรองจะถูกกำหนดด้วยพารามิเตอร์ $\Theta(n)$ และ Recursive Algorithm ใช้สำหรับปรับ $\Theta(n)$ จนทำให้สัญญาณเอาต์พุต $y(n)$ ใกล้เคียงกับ $d(n)$ ให้มากที่สุด ตัวอย่างการนำไปประยุกต์ใช้งานแสดงดังรูปที่ 5.2



รูปที่ 5.1 ส่วนประกอบของตัวกรองความถี่อะแดปทีฟแบบ IIR



รูปที่ 5.2 การประยุกต์ใช้งานเพื่อทำ System Identification

เป็นการทำ System Identification เมื่อ $\hat{\Theta}$ คือ พารามิเตอร์ที่ไม่ทราบค่า และ $d(n)$ คือ สัญญาณที่วัดได้ที่เอาต์พุตของระบบ ซึ่งโดยทั่วไปจะถูกรบกวนด้วยสัญญาณรบกวน $w(n)$ วัตถุประสงค์ของอัลกอริทึม คือ พยายามทำให้ค่าผิดพลาด $e(n)$ มีค่าต่ำที่สุด ซึ่งเป็นหลักการพื้นฐานของการทำ Prediction Error (PE) บางครั้งเรียกว่า Estimation Error ซึ่งกำหนดได้ดังนี้ คือ

$$e(n) = d(n) - y(n) \tag{5.1}$$

ข้อตัดสิน (Criterion) ที่นิยมใช้ คือ ดูที่ Mean Square Error (MSE) ของสมการที่ (5.1) ซึ่งกำหนดได้ดังสมการ คือ

$$\xi = E\{e^2(n)\} \tag{5.2}$$

เมื่อ $E\{\cdot\}$ คือตัวหาค่าคาดหวังทางสถิติ (statistical expectation) โดยพารามิเตอร์ $\hat{\Theta}$ จะถูกปรับด้วย อัลกอริทึมแบบ Stochastic Gradient หรือ อัลกอริทึมของ Gauss-Newton นอกจากนี้ยังมีข้อตัดสินอย่างอื่นอีก คือ วิธีการของ Least Square (LS) หรือ Recursive Least Square (RLS)

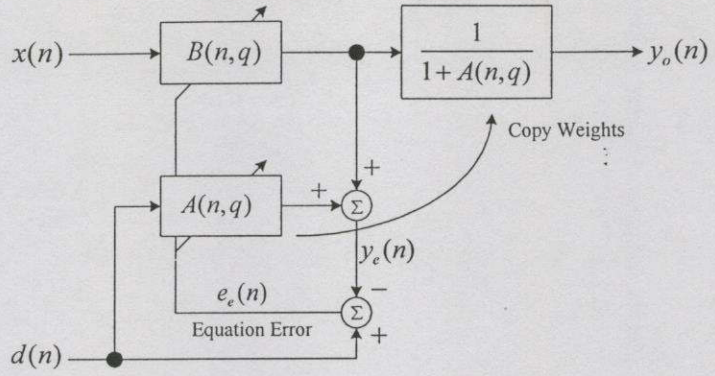
โดยพื้นฐานแล้ว จะสามารถสร้างสมการค่าผิดพลาด (Equation Error) เพื่อทำ Prediction Error ได้สองแบบคือ วิธี Equation Error และ วิธี Output Error โดยวิธีแรก สัมประสิทธิ์ที่ถูกป้อนกลับของตัวกรอง IIR จะถูกปรับในรูปแบบ FIR (all-zero) จากนั้นจะถูกคัดลอก (copy) ให้กับตัวกรองที่สอง ซึ่งจะสร้างในรูปของ IIR ที่เป็น all-pole วิธีการนี้ จะทำให้ตัวกรองอะแดปทีฟ IIR มีคุณสมบัติเป็น FIR ซึ่งง่ายต่อการทำความเข้าใจ และ วิเคราะห์เพื่อหาสมรรถนะของตัวกรอง แต่วิธีนี้จะมีข้อเสีย คือ อาจทำให้พารามิเตอร์ $\hat{\Theta}$ มีค่า bias เกิดขึ้นได้ ส่วนวิธีที่สองนั้น สัมประสิทธิ์ป้อนกลับจะถูกปรับในรูปแบบ recursive pole-zero ซึ่งไม่ทำให้เกิดค่า bias แต่จะทำให้ อะแดปทีฟอัลกอริทึม ลู่เข้าสู่จุด local minimum ของ ξ ทำให้ $\hat{\Theta}$ ผิดไปจากค่าที่ต้องการ และการวิเคราะห์คุณสมบัติการลู่เข้าจะทำได้ยาก ดังนั้น จากที่กล่าวมา จะต้องเลือกอย่างใดอย่างหนึ่ง

ข้อดีหลักของตัวกรองอะแดปทีฟ IIR ที่เหนือกว่า FIR คือ จะให้สมรรถนะดีกว่า FIR ที่จำนวนอันดับเท่ากัน และผลพลอยได้ที่สำคัญคือ สัญญาณเอาต์พุตที่ถูกป้อนกลับ จะสร้างผลตอบสนองต่อสัญญาณอิมพัลส์ที่มีค่าเป็นอนันต์ ทั้งที่จำนวนสัมประสิทธิ์มีค่าจำกัด นอกจากนี้จะสามารถประมาณผลตอบสนองทางความถี่ได้อย่างมีประสิทธิภาพมากกว่า เมื่อใช้สัญญาณเอาต์พุตจากระบบที่มีทั้งโพล และ ซีโร่ และ เนื่องจากตัวกรองอะแดปทีฟ IIR ใช้จำนวนสัมประสิทธิ์น้อยกว่า FIR จะเป็นผลให้ลดความซับซ้อนในการคำนวณลงได้ ดังนั้นในการประยุกต์ใช้งานหลาย ๆ อย่าง จึงหันมาใช้ตัวกรองอะแดปทีฟแบบ IIR แทน FIR มากขึ้น

5.2 Equation-Error Adaptive IIR filter

พิจารณาตัวกรองอะแดปทีฟ IIR แบบ Equation-Error ในรูปที่ 5.3 ซึ่งกำหนดได้ด้วย สมการผลต่างแบบ nonrecursive คือ

$$y_e(n) = \sum_{m=1}^{N-1} a_m(n)d(n-m) + \sum_{m=0}^{M-1} b_m(n)x(n-m) \tag{5.3}$$



รูปที่ 5.3 Equation-Error adaptive IIR filter

เมื่อ $\{a_m(n), b_m(n)\}$ คือ สัมประสิทธิ์ที่จะถูกปรับ และสับสคริบ e บอกให้ทราบว่าสัญญาณเอาต์พุตนี้เป็นของ equation-error formulation สังเกตสมการที่ (5.3) พบว่า จะประกอบด้วย สัญญาณอินพุตสองสัญญาณ และ สัญญาณเอาต์พุตสัญญาณเดียว ซึ่งขึ้นกับสัญญาณอินพุตในอดีต $x(n-m)$, $m = 0,\ldots, M-1$ และสัญญาณที่ต้องการ $d(n-m)$, $m = 0,\ldots, N-1$ เท่านั้น จะไม่ขึ้นกับสัญญาณเอาต์พุต $y(n-m)$ ในอดีต ดังนั้น จึงไม่มีการป้อนกลับ ทำให้สัญญาณเอาต์พุต มีความเป็นเชิงเส้นกับ สัมประสิทธิ์ของตัวกรอง ทำให้ง่ายต่อการค้นหาจุดต่ำสุดของพื้นผิวค่าผิดพลาด ที่อยู่บนพื้นฐานของ Gradient (Gradient-Based algorithm) เนื่องจาก $d(n)$ และ $x(n)$ ไม่เป็นฟังก์ชันของสัมประสิทธิ์ของตัวกรอง ทำให้การทำอนุพันธ์ (derivative) ของ $y_e(n)$ เทียบกับสัมประสิทธิ์ ได้ง่ายในการคำนวณ

เพื่อความสะดวก จะเขียนสมการที่ (5.3) ในรูปของสัญลักษณ์ของ delay-operator ดังนี้คือ

$$y_e(n) = A(n,q)d(n) + B(n,q)x(n) \tag{5.4}$$

เมื่อ

$$A(n,q) = \sum_{m=1}^{N-1} a_m(n)q^{-m} \tag{5.5a}$$

และ

$$B(n,q) = \sum_{m=0}^{M-1} b_m(n)q^{-m} \tag{5.5b}$$

จะสังเกตเห็นว่า ภารรวม (sum) ของ $A(n,q)$ จะเริ่มต้นที่ m = 1 นั่นหมายความว่า $A(n,q)d(n)$ จะขึ้นกับค่าในอดีตของ d คือ $d(n\text{-}m)$ เท่านั้นไม่ขึ้นกับ $d(n)$ ซึ่งเป็นค่าปัจจุบัน ส่วน n บอกให้ทราบว่าสัมประสิทธิ์มีการเปลี่ยนตามเวลา และ q^{-1} เป็น delay operator นั่นคือ $q^{-m}x(n) = x(n-m)$ ถ้าแทน q ด้วย z สมการที่ (5.5) จะกลายเป็นการแปลง z ถ้าสมมติว่าสัมประสิทธิ์ทุกตัวเป็นค่าคงที่ ไม่ขึ้นกับเวลา คือ $a_m(n) \to a_m$ และ $b_m(n) \to b_m$ ดังนั้น $A(n,q) \to A(z)$ และ $B(n,q) \to B(z)$ รูปแบบดังกล่าว จะสามารถใช้หา ซีโร่ ของตัวกรองอะแดปทีฟ ที่เวลาใด ๆ ได้ ตัวอย่างเช่น หลังจากที่สัมประสิทธิ์ถูกปรับแต่ละครั้ง และก่อนที่จะคัดลอกสัมประสิทธิ์ $\{a_m(n)\}$ ให้กับ inverse filter (รูปที่ 5.3) มีความจำเป็นจะต้องทราบ ซีโร่ของ 1- $A(z)$ ทั้งนี้เพราะ เมื่อคัดลอกไปที่ inverse filter จะกลายเป็น โพลของ inverse filter ซึ่งอาจทำให้ตัวกรองไม่เสถียรภาพ ถ้าหากโพลอยู่นอกวงกลมหนึ่งหน่วยในระนาบ z

สมการค่าผิดพลาด (Equation Error) กำหนดได้ตามสมการ คือ

$$e_e(n) = d(n) - y_e(n) \tag{5.6}$$

ที่เรียกเช่นนี้ก็เพราะว่า เป็นผลต่างของสมการผลต่างสองสมการคือ $[1 - A(n,q)]d(n)$ และ $B(n,q)x(n)$ โดยที่ $e_e(n)$ นี้จะเป็นเชิงเส้นกับสัมประสิทธิ์ของตัวกรอง จึงทำให้ค่า mean-square equation error (MSEE) กลายเป็นฟังก์ชันแบบ quadratic ซึ่งจะมีจุดต่ำสุด (Global minimum) เพียงจุดเดียว และให้ สมรรถนะ เหมือนกับตัวกรองอะแดปทีฟแบบ FIR

สมการที่ (5.3) สามารถเขียนให้อยู่ในรูปของของ inner product ดังนี้ คือ

$$y_e(n) = \Theta^t(n)\Phi_e(n) \tag{5.7}$$

เมื่อ

$$\Theta(n) = [a_1(n),...,a_{N-1}(n), b_0(n),...,b_{M-1}(n)]^t \tag{5.8}$$

และ

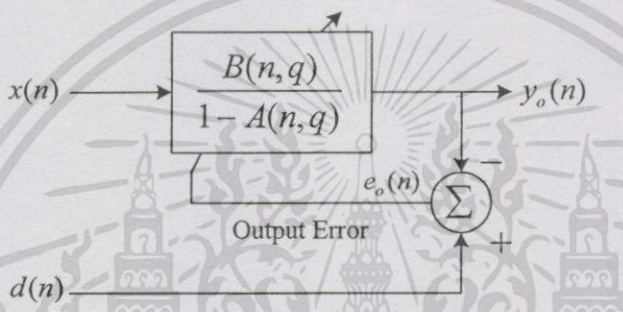
$$\Phi_e(n) = [d(n-1),...,d(n-N+1), x(n),..,x(n-M+1)]^t \tag{5.9}$$

จากสมการที่ (5.7) มีรูปแบบเป็น linear regression ซึ่งจะพบทั่วไปทางด้านสถิติ [36] เมื่อ $\Theta(n)$ คือพารามิเตอร์ที่ถูกประมาณ และ $\Phi_e(n)$ คือ regression เวกเตอร์ โดยที่ regressor จะเป็นอิสระจากพารามิเตอร์ เพราะ $x(n)$ และ $d(n)$ ไม่เป็นฟังก์ชันของ $A(n,q)$ หรือ $B(n,q)$ มีอัลกอริทึมมาก

มาย ที่ใช้สำหรับหาเซตของพารามิเตอร์ที่ทำให้ค่า MSEE ต่ำสุด เช่น Maximum likelihood [37], Maximum a posteriori [37], Least square [38] และ Mean-square error criterion ความจริงแล้ว อัลกอริทึมแบบ LMS (Least-mean-square) [39] ก็คืออัลกอริทึมแบบ Recursive gradient-decent ที่ใช้สำหรับหาค่าต่ำสุดของ MSEE

5.3 Output-Error Adaptive IIR filter

ตัวกรองอะแดปทีฟ IIR แบบ output-error แสดงดังรูปที่ 5.4 ถ้าใช้โครงสร้างแบบ Direct-form จะได้สมการผลต่างแบบ recursive ดังนี้ คือ



รูปที่ 5.4 Output error adaptive IIR filter

$$y_o(n) = \sum_{m=1}^{N-1} a_m(n) y_o(n-1) + \sum_{m=0}^{M-1} b_m(n) x(n-m) \tag{5.10}$$

จากสมการที่ (5.10) พบว่าสัญญาณเอาต์พุตปัจจุบันนั้น ขึ้นกับสัญญาณเอาต์พุตในอดีต $y_o(n-m)$, m = 1,..., $N-1$ สัญญาณเอาต์พุตที่ถูกป้อนกลับนี้ จะเป็นผลให้รูปของอะแดปทีฟอัลกอริทึม มีความซับซ้อนมากขึ้นเมื่อเทียบกับใช้วิธี equation error และจากสมการที่ (5.10) เขียนใหม่ได้ดังนี้ คือ

$$y_o(n) = \left(\frac{B(n,q)}{1 - A(n,q)} \right) x(n) \tag{5.11}$$

และเขียนในรูป ininer product คือ

$$y_o = \Theta^t(n) \Phi_o(n) \tag{5.12}$$

โดยที่ Θ มีค่าตามสมการที่ (5.8) และเวกเตอร์ของสัญญาณในวิธีนี้ คือ

$$\Phi_o(n) = [y_o(n-1),...,y_o(n-N+1),x(n),...,x(n-M+1)]^t \qquad (5.13)$$

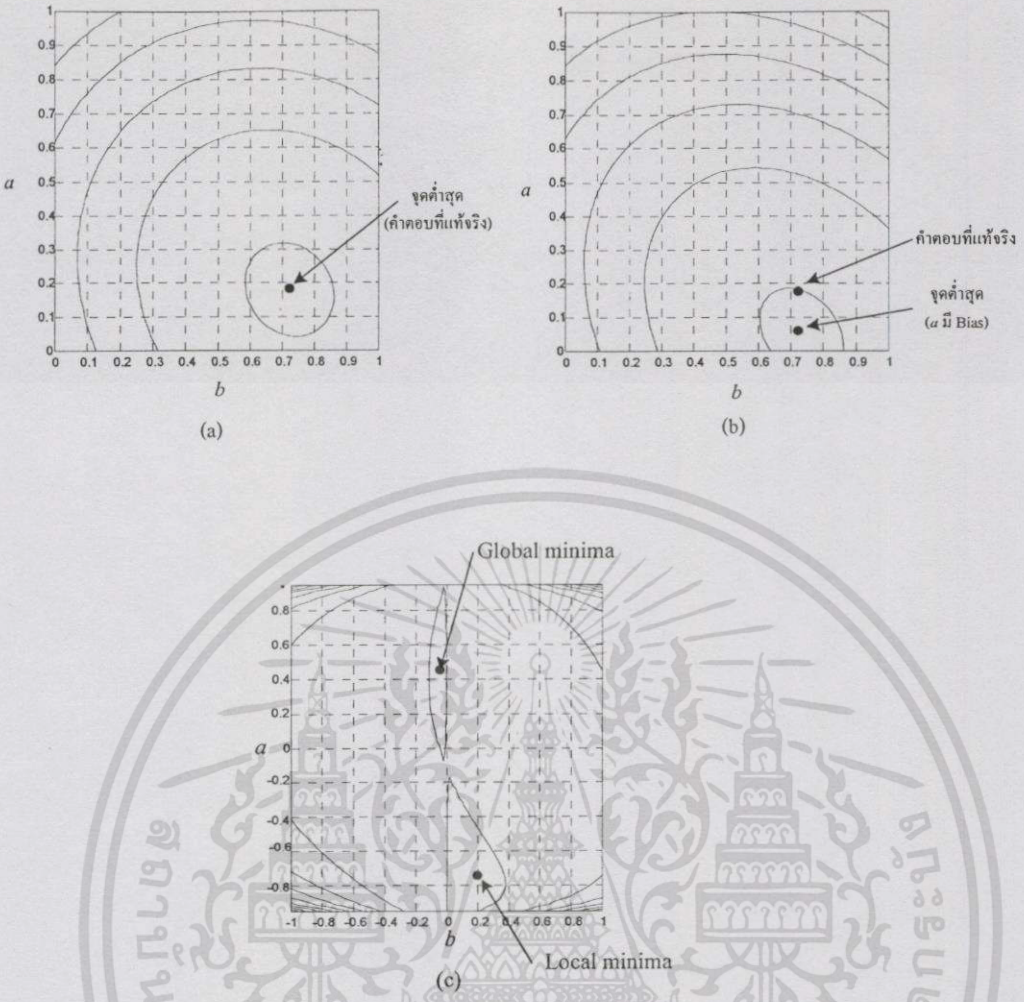
จะเห็นได้ชัดเจนว่า สัญญาณเอาต์พุตเป็นฟังก์ชันแบบไม่เชิงเส้นกับพารามิเตอร์ Θ ทั้งนี้เพราะ สัญญาณเอาต์พุตในอดีต $y_o(n-k)$ ของ Φ_o จะขึ้นกับสัมประสิทธิ์ของตัวกรองตัวก่อนหน้าที่ผ่านมา ดังนั้นสมการที่ (5.12) จะไม่เป็น linear regression แต่จะมีรูปแบบเหมือนกับสมการที่ (5.7) จึงเรียกสมการที่ (5.12) ว่าคือ pseudolinear regression และในทำนองเดียวกัน จะสามารถใช้อัลกอริทึม และวิธีการทางสถิติเพื่อหาเซตของสัมประสิทธิ์ที่ดีที่สุดได้ แต่อาจจะมีหลายคำตอบ เว้นแต่ ฟังก์ชันถ่ายโอนจะมีคุณสมบัติเป็น Strictly Positive Real (SPR) [40]

output error จะกำหนดได้ตามสมการ คือ

$$e_o(n) = d(n) - y_o(n) \qquad (5.14)$$

จะเห็นได้ชัดเจนว่า $e_o(n)$ เป็นฟังก์ชันแบบไม่เชิงเส้นกับ Θ ด้วย ดังนั้นจะทำให้ mean-square-output error (MSOE) ไม่เป็นฟังก์ชันแบบ quadratic ทำให้เกิด local minima จำนวนมากมาย เป็นผลให้อะแดปอัลกอริทึมลู่เข้าสู่จุดใดจุดหนึ่งได้ โดยที่ไม่ลู่เข้าสู่จุดต่ำสุดที่แท้จริง (Global minima) ทำให้คำตอบผิดไปจากที่ต้องการ รูปที่ 5.5 (a, b) แสดงตัวอย่างกราฟแบบ contour ของพื้นผิวค่าผิดพลาดของ Equation Error ในกรณีที่ไม่มีสัญญาณรบกวน และในกรณีที่มีสัญญาณรบกวนตามลำดับ และรูปที่ 5.5 (c) เป็นของ Output Error

ดังนั้น ในการนำไปใช้งาน จะต้องเลือกเอาอย่างใดอย่างหนึ่ง คือ ถ้าต้องการให้ cost ฟังก์ชันเป็นเชิงเส้นกับสัมประสิทธิ์ของตัวกรอง จะต้องใช้ตัวกรองอะแดปทีฟ IIR แบบ Equation Error แต่คำตอบที่ได้จะมีค่า bias เมื่อมีสัญญาณรบกวน แต่ถ้าเลือกแบบ Output Error จะทำให้ พื้นผิวค่าผิดพลาดเกิด Local minima ขึ้น ซึ่งคำตอบที่ได้ จะไม่ใช่คำตอบที่แท้จริง



รูปที่ 5.5 กราฟแบบ contour ของพื้นผิวค่าผิดพลาด (error surface) : (a) Equation Error เมื่อไม่สัญญาณรบกวน (b) Equation Error ขณะมีสัญญาณรบกวน และ (c) Output Error

5.4 อะแดปทีฟอัลกอริทึมบนพื้นฐานของเกรเดียนต์ (Gradient- Based Adaptive Algorithm)

เนื่องจาก อะแดปทีฟอัลกอริทึมที่ใช้ร่วมกับ ตัวกรองอะแดปทีฟ IIR แบบ Equation Error จะมีความคล้ายคลึงกันอย่างมากกับ ตัวกรองอะแดปทีฟ FIR โดยในเอกสารอ้างอิง [38] ได้กล่าวไว้อย่างละเอียด จึงไม่กล่าวถึง ณ ที่นี้ ดังนั้น จะกล่าวเฉพาะอะแดปทีฟอัลกอริทึมที่ใช้สำหรับตัวกรองอะแดปทีฟแบบ IIR แบบ ouput equation error เท่านั้น

5.4.1 อะแดปทีฟอัลกอริทึมแบบ Recursive Prediction Error (RPE)

อะแดปทีฟอัลกอริทึมแบบ RPE จะทำการปรับสัมประสิทธิ์ จนทำให้ MSOE ของ Cost ฟังก์ชัน $\zeta = E\{e_o^2(n)\}$ มีค่าต่ำสุด เมื่อ e_o คือ Output Error โดยทั่วไป จะไม่ทราบค่าของ ζ ทั้งนี้เพราะสัญญาณจะมีลักษณะเป็น nonstationary ดังนั้นจึงทำการประมาณค่า ζ ทุก ๆ ช่วงเวลาที่เพิ่มขึ้น ซึ่งกำหนดได้ดังสมการ คือ

$$\zeta(n) = e_o^2(n) \tag{5.15}$$

เห็นได้ชัดว่า การกระทำเช่นนี้ จะทำให้สัมประสิทธิ์ของตัวกรองเกิดสัญญาณรบกวนเนื่องจากการประมาณ (noisy estimates)

อัลกอริทึมแบบ RPE จะทำการปรับสัมประสิทธิ์ $\Theta(n)$ ในทิศทาง negative gradient ของ $\zeta(n)$ กำหนดได้ดังสมการ คือ

$$\nabla_{\Theta}[\zeta(n)] \equiv \frac{1}{2}\frac{\partial \zeta(n)}{\partial \Theta(n)} = e_o(n)\frac{\partial e_o(n)}{\partial \Theta(n)} = -e_o(n)\frac{\partial y_o(n)}{\partial \Theta(n)} \tag{5.16}$$

เมื่อ

$$\frac{\partial y_o(n)}{\partial \Theta(n)} = \left[\frac{\partial y_o(n)}{\partial a_1(n)},\ldots,\frac{\partial y_o(n)}{\partial a_{N-1}(n)},\frac{\partial y_o(n)}{\partial b_0(n)},\ldots,\frac{\partial y_o(n)}{\partial b_{M-1}(n)}\right]^t \tag{5.17}$$

จากสมการที่ (5.12) จะได้ว่า

$$\nabla_{\Theta}[y_o(n)] = \nabla[\Theta^t(n)\Phi_o(n)] \tag{5.18}$$

ถ้า Φ เป็นอิสระจาก Θ แล้วจะได้ว่า

$$\nabla_{\Theta}[y_o(n)] = \Phi_o(n) \tag{5.19}$$

แต่เนื่องจาก Φ_o ประกอบด้วยค่า y_o ในอดีต ซึ่งจะขึ้นกับสัมประสิทธิ์ของตัวกรอง จึงทำให้การคำนวณค่าตามสมการที่ (5.19) นั้น ทำได้ยาก แต่จะมีวิธีการประมาณที่เรียกว่า pseudolinear regression algorithm มาใช้คำนวณค่าในสมการที่ (5.19) ได้ ซึ่งจะกล่าวถัดไป ทำการ

หาอนุพันธ์ทั้งสองข้างของสมการที่ (5.10) เทียบกับ $a_k(n)$ โดยที่ $b_m(n)$ และ $x(n-m)$ เป็นอิสระจาก $a_k(n)$ จะได้ว่า

$$\frac{\partial y_o(n)}{\partial a_k(n)} = y_o(n-k) + \sum_{m=1}^{N-1} a_m(n) \frac{\partial y_o(n-m)}{\partial a_k(n)} \tag{5.20a}$$

ในทำนองเดียวกัน เมื่อทำการหาอนุพันธ์เทียบกับ $b_k(n)$ จะได้

$$\frac{\partial y_o(n)}{\partial b_k(n)} = x_o(n-k) + \sum_{m=1}^{N-1} a_m(n) \frac{\partial y_o(n-m)}{\partial b_k(n)} \tag{5.20b}$$

อนุพันธ์ย่อยทางขวามือในสมการที่ (5.20) เกิดขึ้นเพราะว่า ตัวกรองตามสมการที่ (5.10) มีการป้อนกลับ ทำให้เอาต์พุตก่อนหน้าขึ้นกับสัมประสิทธิ์ก่อนหน้าด้วย และจะเกี่ยวพันธ์กับค่าในปัจจุบันด้วย จะเป็นไปในลักษณะนี้จนจบกระบวนการปรับ สังเกตว่า การทำอนุพันธ์ตามสมการที่ (5.20) จะเทียบกับสัมประสิทธิ์ค่าปัจจุบันคือ $a_k(n)$ และ $b_k(n)$ ซึ่งไม่เป็นรูปแบบของ recursive เป็นผลให้ไม่สามารถเขียน ให้อยู่ในรูปของตัวกรอง ซึ่งใช้ ตัวหน่วงได้ อย่างไรก็ตาม ถ้าสเต็บไซส์ (step size (μ) ซึ่งเป็นพารามิเตอร์ที่ใช้สำหรับกำหนดอัตราความเร็วในการทำงานของ Gradient-based algorithm) มีขนาดเล็กพอ ทำให้การปรับสัมประสิทธิ์เป็นไปอย่างช้า ๆ แล้วจะสามารถประมาณได้ว่า

$$\Theta(n) \approx \Theta(n-1) \approx, ..., \approx \Theta(n-N+1) \tag{5.21}$$

ซึ่งการประมาณเช่นนี้ จะนำไปใช้กันในทางปฏิบัติ โดยเฉพาะในกรณีที่ N มีค่าน้อย อย่างไรก็ตาม การประมาณเช่นนี้จะไม่เป็นที่ยอมรับ ถ้าทำให้ประสิทธิภาพการทำงานของ RPE ลดลง ซึ่งจะสังเกตได้ในทางปฏิบัติเช่นกัน ดังนั้นจะสามารถแทนสมการที่ (5.20) ด้วยสมการต่อไปนี้คือ

$$\begin{aligned} \frac{\partial y_o(n)}{\partial a_k(n)} &\approx y_o(n-k) + \sum_{m=1}^{N-1} a_m(n) \frac{\partial y_o(n-m)}{\partial a_k(n-m)} \\ &= \left(\frac{1}{1 - A(n,q)} \right) y_o(n-k) \end{aligned} \tag{5.22a}$$

และ

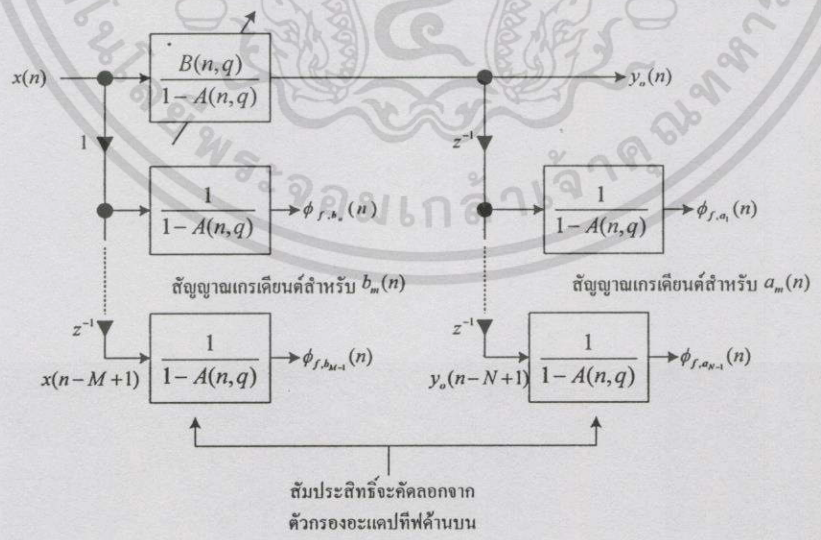
$$\frac{\partial y_o(n)}{\partial b_k(n)} \approx x(n-k) + \sum_{m=1}^{N-1} a_m(n) \frac{\partial y_o(n-m)}{\partial b_k(n-m)}$$

$$= \left(\frac{1}{1-A(n,q)} \right) x(n-k) \tag{5.22b}$$

การประมาณข้างต้น เป็นผลให้สมการที่ (5.22) มีรูปแบบเป็น recursive แต่ละองค์ประกอบของสัญญาณภายใน Φ_o จะถูกกรองโดย inverse pole-polynomial ของตัวกรองอะแดปทีฟ ซึ่งเป็นคุณลักษณะประจำตัวของตัวกรองอะแดปทีฟแบบ IIR เท่านั้น และอะแดปทีฟอัลกอริทึมแบบ Gradient-Based กำหนดได้ดังสมการ คือ

$$\Theta(n+1) = \Theta(n) + \mu I^{-1}(n+1) \left(\frac{1}{1-A(n,q)} \right) \Phi_o(n) e_o(n) \tag{5.23}$$

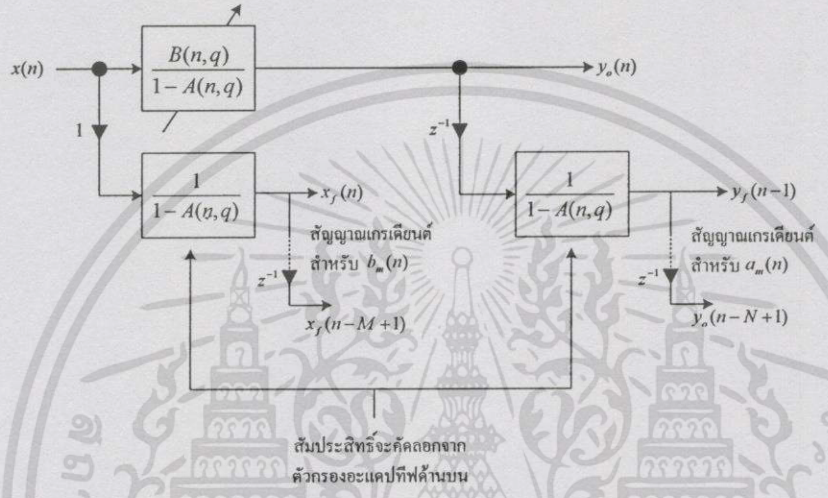
เมื่อ μ คือ สเต็ปไซส์ I คือเมตริกเอกลักษณ์ ถ้าให้ $F(n,q) = 1/[1-A(n,q)]$ และ $G(n,q) = 1$ จะพบว่า ตัวกรอง F จะสร้างสัญญาณเกรเดียนต์ออกมาเป็นจำนวน M+N-1 สัญญาณ ซึ่งตัวกรอง F นี้สร้างจากการคัดลอกเอาสัมประสิทธิ์ตัวส่วนของตัวกรองอะแดปทีฟหลังจากมีการปรับสัมประสิทธิ์แล้วมาใช้ โดยตัวกรอง F ทั้งหมดจะต่างกันตรงสัญญาณอินพุตเท่านั้น คือ ใช้สัญญาณอินพุตและเอาต์พุตปัจจุบันและในอดีตป้อนให้กับตัวกรอง F แต่ละตัว ดังแสดงในรูปที่ 5.6



รูปที่ 5.6 ตัวกรอง F ที่ใช้สำหรับสร้างสัญญาณเกรเดียนต์

จากรูปจะเห็นได้ชัดว่า ขณะคำนวณหา ϕ_f จะต้องใช้การคำนวณที่มาก และต้องการหน่วยความจำเพื่อเก็บข้อมูลเป็นจำนวนมากเพื่อเก็บข้อมูลในอดีต จากการประมาณค่าสัมประสิทธิ์ของตัวกรอง ดังที่ได้กล่าวมาแล้วนั้น พบว่า สามารถลดการคำนวณเพื่อ ϕ_f ลงได้ ดังแสดงในรูปที่ 5.7 จากรูปกำหนดให้

$$y_f(n-1) \equiv \frac{\partial y_o(n)}{\partial a_1(n)} \quad \text{และ} \quad x_f(n) \equiv \frac{\partial y_o(n)}{\partial b_0(n)} \tag{5.24}$$



รูปที่ 5.7 Simplified RPE algorithm

เป็นค่าเริ่มต้นของสัญญาณเกรเดียนต์ (พจน์แรกในสมการที่ (5.22a) และ (5.22b) ที่ $k = 1$ และ $k = 0$ ตามลำดับ) และสำหรับสัญญาณเกรเดียนต์ตัวถัดไป จะคำนวณโดยใช้การประมาณต่อไปนี้แทน คือ

$$\frac{\partial y_o}{\partial a_k(n)} \approx y_f(n-k) \quad \text{และ} \quad \frac{\partial y_o}{\partial b_k(n)} \approx x_f(n-k) \tag{5.25}$$

สำหรับ k =2,..., N-1 และ k =1,..., M-1 ตามลำดับ นั่นคือ แต่ละองค์ประกอบของสัญญาณเกรเดียนต์ คือ ค่าในอดีตอันใดอันหนึ่งของสัญญาณเกรเดียนต์ตามสมการที่ (5.24) ดังนั้น $y_f(n-k)$ และ $x_f(n-k)$ จะขึ้นกับสัมประสิทธิ์ในอดีตของ $A(n-k,q)$ แทนที่จะเป็นของ $A(n,q)$ ตามในรูปที่ 5.6 อัลกอริทึมตามรูปที่ 5.7 จะเรียกว่า Simplified RPE algorithm ดังนั้น จะสามารถแทนเวกเตอร์ข้อมูล ด้วยสมการต่อไปนี้ คือ

$$\Phi_f(n) = [y_f(n-1),..., y_f(n-N+1), x_f(n),..., x_f(n-M+1)]' \tag{5.26}$$

ซึ่งใช้ตัวกรองสำหรับสร้างสัญญาณเกรเดียนต์เพียงสองตัวเท่านั้น วิธีการนี้จะนิยมใช้กันในทางปฏิบัติ เพราะใช้หน่วยความจำน้อย และคำนวณน้อยกว่า

5.5 การพิจารณาเสถียรภาพของตัวกรอง

ข้อเสียหลักของอัลกอริทึม RPE ก็คือ pole polynomial ของตัวกรอง จะถูกนำไปใช้คำนวณสัญญาณเกรเดียนต์ ซึ่งอาจเกิดความไม่เสถียรภาพในระหว่างการปรับ โดยที่ ถ้า โพลของ 1 - $A(n,q)$ เกิดความผิดพลาดในขณะปรับ ออกไปอยู่นอกวงกลมหนึ่งหน่วย เป็นผลให้สัญญาณเอาต์พุตมากจนไม่สามารถควบคุมได้ ซึ่งเป็นการยากที่จะป้องกันไม่ให้เกิดขึ้นเลย ดังนั้นจึงต้องมีการทดสอบความมีเสถียรภาพของตัวกรองในขณะปรับด้วย และเนื่องจากได้นำ $\zeta(n)$ มาใช้แทน $E\{\zeta\}$ จึงเป็นสาเหตุให้เกิดสัญญาณรบกวนในขณะปรับ จึงเป็นไปได้ที่โพลจะหลุดออกนอกวงกลมหนึ่งหน่วยขณะทำการปรับ โดยเฉพาะในงานที่ต้องการให้ตำแหน่งโพลอยู่ใกล้ ๆ กับเส้นรอบวงกลมหนึ่งหน่วย ก็มีโอกาสที่โพลจะหลุดออกนอกวงกลมได้ง่าย รูปที่ 2.3 ในบทที่ 2 แสดงขอบเขตที่สัมประสิทธิ์ของตัวกรอง ซึ่งจะต้องอยู่ภายในสามเหลี่ยมเท่านั้น

วิธีที่ง่ายที่สุดสำหรับการทดสอบเสถียรภาพ คือ จะทำการตรวจสอบหลังจากการปรับแต่ละครั้งโดยตรวจสอบที่ผลบวกของ $|a_m(n)|$ จะต้องมีค่าน้อยกว่า 1 เสมอ นอกจากนี้ยังมีวิธีอื่น ๆ อีกหลายวิธีที่ใช้สำหรับพิจารณาเสถียรภาพของตัวกรองดูรายละเอียดได้จากเอกสารอ้างอิง [41-42] การจะยืนยันว่า ตัวกรองมีความเสถียรภาพ โดยพิจารณาจากตำแหน่งโพล ต้องอยู่ในวงกลมหนึ่งหน่วยนั้น จะใช้ได้กับเฉพาะตัวกรองที่มีคุณสมบัติเป็น LTI เท่านั้น แต่สำหรับตัวกรองที่ไม่มีคุณสมบัติเป็น LTI คือมีคุณสมบัติเป็น Time-Varying System เช่นตัวกรองแบบ IIR การพิจารณาโพลอย่างเดียวจะไม่เพียงพอ

5.6 สรุป

ในบทนี้ได้กล่าวทั่ว ๆ ไปเกี่ยวกับ โครงสร้างและอะแดปทีฟอัลกอริทึม ที่ใช้กับตัวกรองแบบ IIR ตัวกรองอะแดปทีฟ IIR แบบ output – error จะมีการป้อนกลับ ทำให้อะแดปทีฟอัลกอริทึมมีความซับซ้อนกว่าการใช้ตัวกรองอะแดปทีฟแบบ FIR นอกจากนี้ยังกล่าวถึงการพิจารณาเสถียรภาพของตัวกรองขณะทำการปรับสัมประสิทธิ์ โดยสัมประสิทธิ์ของตัวกรองจะต้องอยู่ภายในสามเหลี่ยมเสถียรภาพเท่านั้น เนื่องจากตัวกรองอะแดปทีฟแบบ IIR มีความเป็น nonlinear สูงจึงทำให้ ทฤษฎีของตัวกรองอะแดปทีฟแบบ IIR ยังไม่สมบูรณ์ แต่ได้มีการนำมาใช้กันพอสมควร โดยทั่วไปการศึกษาตัวกรองอะแดปทีฟแบบ IIR จะนำคอมพิวเตอร์มาใช้เพื่อจำลองการทำงานแทนการวิเคราะห์ตามหลักคณิตศาสตร์ ซึ่งค่อนข้างยาก เพราะระบบมีลักษณะไม่เป็นเชิงเส้นนั่นเอง

บทที่ 6
ตัวกรองอะแดปทีฟ IIR แบบนอตช์

วิทยานิพนธ์นี้ นำเสนอการประยุกต์ใช้งาน ตัวกรองอะแดปทีฟ IIR แบบนอตช์ (ANF) เพื่อการตรวจวัดความถี่ของสัญญาณคลื่นซายน์ ที่ไม่ทราบค่าในสัญญาณรบกวน ที่มีใช้ในระบบเรดาร์ หรือ โซนา, ใช้สำหรับกำจัดสัญญาณแบนด์แคบ (narrow band) ที่ปนมากับสัญญาณแบนด์กว้าง (broad band) ในระบบสื่อสาร หรือ ระบบควบคุม, ใช้สำหรับยกระดับสัญญาณคลื่นซายน์ในสัญญาณรบกวน ของระบบดีมอดูเลตในระบบสื่อสาร และใช้สำหรับกำจัดสัญญาณคลื่นซายน์ความถี่ 50 Hz ในสัญญาณคลื่นไฟฟ้าหัวใจ (Electrocardiogram : ECG) ของผู้ป่วย ซึ่งเหนี่ยวนำมาจากไฟฟ้ากระแสสลับ 220 โวลท์ ซึ่งทำให้สัญญาณ ECG มีความผิดเพี้ยนไป

ในบทนี้จะกล่าวถึงทฤษฎีทั่วไปเกี่ยวกับ ANF ที่ได้นำมาประยุกต์ใช้ดังกล่าวข้างต้น และอะแดปทีฟอัลกอริทึม ที่ใช้สำหรับปรับสัมประสิทธิ์ของตัวกรอง แบบ Gradient-Based Algorithm หรือเรียกย่อ ๆ ว่า GA โดย ในวิทยานิพนธ์นี้ จะมุ่งเน้นไปที่ การพัฒนาอะแดปทีฟอัลกอริทึมแบบ GA ให้ทำงานที่ดีขึ้น โดยทำการพัฒนาจากอะแดปทีฟอัลกอริทึมแบบ GA ที่พิจารณาค่าผิดพลาดเฉลี่ยกำลัง p น้อยที่สุด (โดยทั่วไปจะพิจารณากันเฉพาะค่าผิดพลาดเฉลี่ยกำลังสองเท่านั้น ซึ่งรู้จักกันทั่วไปคือ LMS) หรือ LMP ที่นำเสนอในบทวิจัย [12] โดยอะแดปทีฟอัลกอริทึมที่ได้พัฒนาขึ้นนี้ได้ถูกนำเสนอโดยผู้เขียนวิทยานิพนธ์ ในเอกสารอ้างอิง [15-17, 43] เนื้อหาในบทนี้จะแบ่งออกเป็นหกหัวข้อ คือ ในหัวข้อที่ 6.1 กล่าวนำเกี่ยวกับ ANF หัวข้อที่ 6.2 กล่าวถึงการประมาณค่า ANF หัวข้อที่ 6.3 กล่าวถึงข้อตัดสิน (Criterion) แบบ LMP หัวข้อ 6.4 กล่าวถึงอะแดปทีฟอัลกอริทึม LMP หัวข้อสุดท้าย 6.5 กล่าวถึง อะแดปทีฟอัลกอริทึมที่นำเสนอในวิทยานิพนธ์ และ หัวข้อที่ 6.6 กล่าวสรุป

6.1 กล่าวนำเกี่ยวกับ ANF

ANF ถูกนำมาใช้ประมาณสัญญาณคลื่นซายน์ที่ไม่ทราบความถี่ ซึ่งปนอยู่ในสัญญาณรบกวน ซึ่งสัญญาณดังกล่าว จะพบได้ในระบบสื่อสารหลาย ๆ ระบบ เช่น ในระบบ ดีมอดูเลต จะใช้สัญญาณคลื่นซายน์เป็นสัญญาณคลื่นพาห (carrier) ซึ่งเป็นไปได้ที่ สัญญาณคลื่นซายน์จะมีความถี่เปลี่ยนไปเมื่อเดินทางไปทางเครื่องรับ หรือ เฟสอาจเกิดการเลื่อน ทำให้เกิดปัญหาขึ้นทางด้านรับเพราะไม่สามารถกู้สัญญาณเดิมกลับมาได้ หรือ กู้ได้แต่เกิดความผิดเพี้ยน ซึ่งในทางปฏิบัติ จำเป็นจะต้องให้สัญญาณคลื่นพาหที่เครื่องส่ง จะต้องสัมพันธ์กัน (synchronized) ซึ่งกำหนดได้ตามสมการ [44] คือ

$$x_k(n) = \sum_{k=1}^{N} A_k \sin(\omega_k n + \theta_k) + w(n) \tag{6.1}$$

เมื่อ A_k คือขนาดของสัญญาณคลื่นซายน์ตัวที่ k, ω_k คือความถี่ที่ไม่ทราบค่ามีค่าอยู่ในช่วง $[0,\pi]$, θ_k คือเฟส เป็นตัวแปรสุ่ม ที่มีคุณสมบัติเป็น IID (independent, identically distribution) มีลักษณะ Pdf เป็นแบบ Uniform มีค่าอยู่ในช่วง $[0,2\pi]$ และ $w(n)$ คือสัญญาณรบกวน โดยสมมติว่า เป็นอิสระจากสัญญาณซายน์ ซึ่งโดยทั่วไปจะสมมติให้เป็นสัญญาณรบกวนแบบขาว มี Pdf แบบเกาส์เซียน (Gaussian White Noise) วัตถุประสงค์ที่ต้องการคือ ต้องการทราบความถี่ ω_k และ/หรือ กำจัดสัญญาณซายน์ทิ้งไป หลังจากที่ตรวจวัดความถี่ได้แล้ว A_k และ θ_k จะมีค่าเหมือนกับสัญญาณที่ทางด้านส่งทุกประการ (ในอุดมคติ)

ถ้าพิจารณาสัญญาณซายน์เพียงคลื่นเดียว และไม่มีสัญญาณรบกวนปน จะเขียนได้ดังสมการ คือ

$$x(n) = \sin(\omega_1 n + \theta) \tag{6.2}$$

จะสามารถหาฟังก์ชัน Autocorrelation ของสมการที่ (6.2) ได้ดังนี้ คือ

$$\begin{aligned} r_{xx}(m) &= E\{x(n)x(n-m)\} \\ &= \frac{1}{2\pi}\int_0^{2\pi} \sin(\omega_1 n + \theta)\sin[\omega_1(n-m)+\theta]d\theta \\ &= \frac{\cos(\omega_1 m)}{2} \end{aligned} \tag{6.3}$$

และฟังก์ชันความหนาแน่นสเปกตรัม คือ

$$\begin{aligned} P_{xx}(e^{j\omega}) &= \sum_{m=-\infty}^{\infty} r_{xx}(m)e^{-jm\omega} \\ &= \frac{\delta(\omega-\omega_1)+\delta(\omega+\omega_1)}{2} \end{aligned} \tag{6.4}$$

เมื่อ Dirac delta function $\delta(\cdot)$ ถูกนิยามจาก kernel property คือ

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} \delta(\omega-\omega_1) f(e^{j\omega})d\omega = f(e^{j\omega_1}) \tag{6.5}$$

โดยที่ $f(e^{j\omega_1})$ จะต้องถูกกำหนดมาก่อน

ทำการป้อนสัญญาณในสมการที่ (6.1) ให้กับตัวกรอง $H(z)$ จะได้สัญญาณเอาต์พุตจากตัวกรองตามสมการ คือ

$$y(n) = H(z)[x(n) + w(n)] \tag{6.6}$$

ค่า variance ของสัญญาณเอาต์พุตมีค่าตามสมการ คือ

$$E\{y^2(n)\} = \sum_{k=1}^{N}\left(\frac{A_k^2}{4\pi}\int_{-\pi}^{\pi}\delta(\omega \pm \omega_k)\left|H(e^{j\omega})\right|^2 d\omega\right) + \frac{1}{2\pi}\int_{-\pi}^{\pi}P_w(e^{j\omega})\left|H(e^{j\omega})\right|^2 d\omega \tag{6.7}$$

เมื่อ

$$P_w(z) = \sum_{m=-\infty}^{\infty}E\{w(n)w(n-m)\}z^{-k} \tag{6.8}$$

สมการที่ (6.8) คือความหนาแน่นสเปกตรัมของกระบวนการสุ่ม [$w(\cdot)$] โดยทั่วไปจะเขียนสมการค่า variance ของสัญญาณเอาต์พุตในรูปต่อไปนี้ คือ

$$E\{y^2(n)\} = \left\langle H(z), H(z)\right\rangle_{P_{xx}} + \left\langle H(z), P_w(z)H(z)\right\rangle \tag{6.9}$$

เมื่อ $\langle\cdot,\cdot\rangle_{P_{xx}}$ คือ inner product

เพื่อให้ง่าย จะกำหนดให้สัญญาณอินพุต เป็นสัญญาณซายน์คลื่นเดียวปนอยู่กับสัญญาณรบกวน ดังสมการ คือ

$$x(n) + w(n) = A_1\sin(\omega_1 n + \theta_1) + w(n) \tag{6.10}$$

ให้ $y(n) = H(z)[x(n) + w(n)]$ จะได้ variance ของสัญญาณเอาต์พุต คือ

$$E\{y^2(n)\} = A_1^2\left|H(e^{j\omega})\right|^2 + \frac{1}{2\pi}\int_{-\pi}^{\pi}P_w(e^{-j\omega})\left|H(e^{j\omega})\right|^2 d\omega \tag{6.11}$$

สมมติว่า ผลตอบสนองทางขนาดต่อความถี่ของ $H(e^{j\omega})$ เป็นดังนี้ คือ

$$\left|H(e^{j\omega})\right| = \begin{cases} 0, \omega = \omega_o \text{ and } \omega = -\omega_o \\ 1, otherwise \end{cases} \tag{6.12}$$

ซึ่งเป็นผลตอบสนองทางขนาดต่อความถี่ของ ตัวกรองแบบนอตช์ (Notch Filter) ในอุดมคติ ผลตอบสนองทางขนาดของตัวกรองไปตามวงกลมหนึ่งหน่วยมีค่าเป็น 1 เกือบทุกความถี่ ยกเว้นที่ $\omega = \pm\omega_o$ มีค่าเท่ากับ 0 ซึ่ง ω_o นี้เรียกว่า ความถี่นอตช์ (notch frequency) และจะได้ว่า

$$E\{y^2(n)\} = \begin{cases} E\{w^2(n)\}, & \omega_o = \omega_1 \\ A_1^2 + E\{w^2(n)\}, & otherwise \end{cases} \tag{6.13}$$

ดังนั้น จะทำให้ output error cost function จะมีจุด นอตช์เพียงจุดเดียว โดยการทำให้ ω_o เลื่อนติดตามตามความถี่ ω_1

ถ้าตัวกรองแบบแถบผ่าน (bandpass filter) จากตัวกรองนอตช์ในสมการที่ (6.12) ดังนี้ คือ

$$\left|G(e^{j\omega})\right|^2 = 1 - \left|H(e^{j\omega})\right|^2 = \begin{cases} 1, \omega = \pm\omega_o \\ 0, otherwise \end{cases} \tag{6.14}$$

และสัญญาณเอาต์พุตของตัวกรองกำหนดได้ คือ

$$y(n) = G(z)[x(n) + w(n)] \tag{6.15}$$

ซึ่งจะมีเฉพาะสัญญาณซายน์เท่านั้น จะสังเกตเห็นว่า ผลตอบสนองทางความถี่ของ $\left|H(e^{-j\omega})\right|$ และ $\left|G(e^{j\omega})\right|$ นั้นเป็นฟังก์ชันแบบไม่ต่อเนื่อง (ในอุดมคติ) และเนื่องจากจะไม่สามารถใช้ ฟังก์ชันแบบเศษส่วน สร้างตัวกรองในอุดมคติได้ ดังนั้นก่อนที่จะทำความเข้าใจว่า จะสามารถทำการประมาณผลตอบสนองทางความถี่ ด้วยฟังก์ชันแบบเศษส่วนได้อย่างไรนั้น ในขั้นแรก จะต้องทำให้ $H(e^{j\omega})$ มีความสัมพันธ์กับผลตอบสนองความถี่ของ stable, causal filter ก่อน ดังตัวอย่างต่อไปนี้ คือ

ตัวอย่างนี้จะแสดงให้เห็นว่า จะสร้าง stable, causal function $H(z)$ ที่ให้ผลตอบสนองทางความถี่เป็นแบบนอตช์ ได้อย่างไร ด้วยความเป็น causal ทำให้สามารถเขียน $H(z)$ ในรูปของอนุกรมได้ดังนี้ คือ

$$H(z) = h_o + h_1 z^{-1} + h_2 z^{-2} + \ldots, \quad |z| > 1 \tag{6.16}$$

เสถียรภาพของสมการที่ (6.16) หาได้จากการพิจารณาที่ L_2 norm ดังนี้ คือ

$$\sum_{m=0}^{\infty} h^2(m) < \infty \tag{6.17}$$

จากความสัมพันธ์ของ Parseval ด้วยกับความจริงที่ว่า

$$\sum_{m=0}^{\infty} h^2(m) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \left|H(e^{j\omega})\right|^2 d\omega = 1 \tag{6.18}$$

ซึ่งจะเป็นจริงถ้าผลตอบสนองทางความถี่เป็นดังสมการที่ (6.12)

พิจารณาฟังก์ชันต่อไปนี้

$$H_o(z) = \exp\left(\frac{z^{-1}+1}{z^{-1}-1}\right) \tag{6.19}$$

สำหรับ $|z|>1$ ฟังก์ชัน $(z^{-1}+1)/(z^{-1}-1)$ จะมีส่วนจริงเป็นลบ ซึ่งทำให้ $H_o(z)$ มีขอบเขต (bound) ดังนั้น อนุกรมตามสมการที่ (6.16) จะลู่เข้าเฉพาะกรณีที่ $|z|>1$

ผลตอบสนองทางความถี่ของสมการที่ (6.19) หาได้โดยการกำหนดให้ $z = \rho e^{j\omega}$ โดยที่ $0<\rho<1$ จากนั้นทำการ take ลิมิต ให้ $\rho \to 1$ ดังนั้น สำหรับที่ $\omega \neq 0$ พบว่า

$$\left|H_o(e^{j\omega})\right| = \left|\lim_{\rho\to 1} H_o(\rho e^{j\omega})\right| = 1, \quad \omega \neq 0 \tag{6.20}$$

เพราะว่า $(\rho e^{-j\omega}+1)/(\rho e^{-j\omega}-1)$ จะลู่เข้าสู่ค่าจินตภาพ เมื่อ $\rho \to 1$ ถ้า $\omega \neq 0$ ซึ่งแน่นอนว่า ค่า exponential ของ ส่วนจินตภาพนี้ จะมีค่า modulus เป็นหนึ่ง

สำหรับที่ $\omega = 0$ พบว่า

$$\lim_{\rho\to 1} \frac{\rho+1}{\rho-1} \to -\infty$$

ดังนั้น

$$H_o(e^{j0}) = \lim_{\rho\to\infty} H_o(\rho) = \exp(-\infty) = 0 \tag{6.21}$$

ดังนั้นผลตอบสนองความถี่ เมื่อทำการ take ลิมิตที่ค่า $\rho \to 1$ จะได้ว่า

$$\left|H_o(e^{j\omega})\right| = \begin{cases} 0, & \omega = 0; \\ 1, & otherwise \end{cases} \tag{6.22}$$

ทำการแปลง $z \to e^{-j\omega_o} z$ และทำการเลื่อน ω_o ไปตามรัศมีของวงกลมในระนาบ z ดังนั้น ฟังก์ชัน $H_o(e^{-j\omega_o} z)$ มี modulus เป็น 1 เกือบทุกจุดบนวงกลมหนึ่งหน่วย ยกเว้นที่ $z = e^{j\omega_o}$ จะมีค่าเท่ากับ 0 ดังสมการ คือ

$$\begin{aligned} H(z) &= H_o(e^{-j\omega_o} z) H_o(e^{j\omega_o} z) \\ &= \exp\left(\frac{e^{-j\omega_o} z^{-1} + 1}{e^{-j\omega_o} z^{-1} - 1} + \frac{e^{j\omega_o} z^{-1} + 1}{e^{j\omega_o} z^{-1} - 1} \right) \\ &= \exp\left(\frac{2(z^{-2} - 1)}{1 - 2\cos(\omega_o) z^{-1} + z^{-2}} \right) \end{aligned} \tag{6.23}$$

ทำการวิเคราะห์เฉพาะที่ $|z| > 1$ จะได้ผลตอบสนองทางความถี่ของสมการที่ (6.23) หลังจากที่ทำการ take ลิมิตให้ $\rho \to 1$ คือ

$$\left|H_o(e^{j\omega})\right| = \left|\lim_{\rho \to 1} H(\rho e^{-j\omega})\right| = \begin{cases} 0, & \omega = \pm\omega_o; \\ 1, & otherwise \end{cases} \tag{6.24}$$

จะพบว่า $H(z)$ มีรูปแบบที่ไม่เป็นฟังก์ชันแบบ เศษส่วน ของตัวแปร z แต่มีความต้องการที่จะสร้างตัวกรองให้อยู่ในรูปของ ฟังก์ชันแบบเศษส่วน ซึ่งจะสามารถทำได้โดยใช้การประมาณแบบ L_2 norm พิจารณาฟังก์ชันต่อไปนี้ คือ

$$\hat{H}(z) = \frac{N(z)}{D(z)} \tag{6.25}$$

เมื่อตัวเศษ $N(z)$ กำหนดให้เป็นโพลิโนเมียลในตัวแปร z อันดับสอง มีซีโร่ที่ $z = e^{\pm j\omega_o}$ และ $D(z)$ มีโพลอยู่ใกล้กับ $z = e^{\pm j\omega_o}$ ซึ่งวิธีการนี้ จะทำให้มี modulus ประมาณ 1 ยกเว้นในบริเวณที่ $z = e^{\pm j\omega_o}$ การได้มาซึ่ง $N(z)$ และ $D(z)$ จะกล่าวในหัวข้อที่ 6.2

6.2 การประมาณค่าตัวกรองแบบนอตช์

หัวข้อนี้จะกล่าวถึง การประมาณตัวกรองแบบนอตช์ จากผลตอบสนองทางอุดมคติ ให้อยู่ในรูปของ ฟังก์ชันแบบเศษส่วน (rational function) ที่มีความสัมพันธ์กับโครงสร้างของตัวกรองดิจิตอลแบบ Direct form [45-47] โดยจะสมมติให้สัญญาณอินพุต เป็นสัญญาณซายน์ความถี่เดียวปนมากับสัญญาณรบกวน

6.2.1 ตัวกรองแบบนอตช์ที่ใช้โครงสร้างแบบ Direct Form

ทำการประมาณ notch filter แบบ direct form ด้วยกับสมการต่อไปนี้ คือ

$$\hat{H}(z) = \frac{N(z)}{N(\rho z)} \tag{6.26}$$

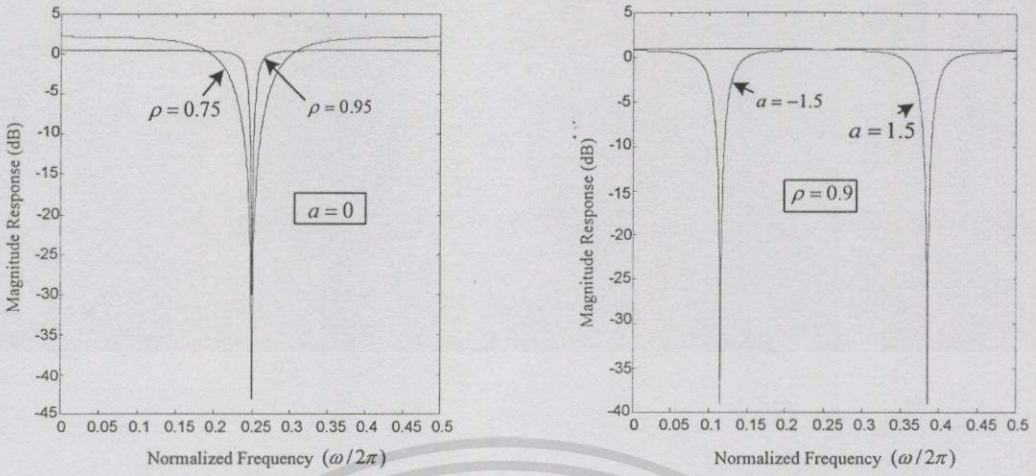
เมื่อ $N(z)$ เป็นโพลีโนเมียลที่มีซีโร่อยู่บนวงกลมหนึ่งหน่วย ในขณะที่ ρ เป็นค่าคงที่ มีค่าใกล้เคียงแต่น้อยกว่า 1 ดังนั้น โพลของตัวกรองจะยังคงอยู่ในบริเวณที่มีความเสถียรภาพ การออกแบบตัวกรองความถี่ด้วยกับฟังก์ชันแบบเศษส่วนนั้น จะนิยมสร้างจาก ตัวกรองความถี่อันดับสอง ถ้าต้องการจำนวนอับดับที่สูงขึ้น ก็จะนำเอาตัวกรองอันดับสองหลาย ๆ ตัวมาต่อ cascade กัน ดังนั้นจะได้ ฟังก์ชันถ่ายโอน (transfer function) ของ notch filter อันดับสองดังสมการ คือ

$$\hat{H}(z) = \frac{1 + az^{-1} + z^{-2}}{1 + \rho az^{-1} + \rho^2 z^{-2}} \quad ,0 < \rho < 1 \tag{6.27}$$

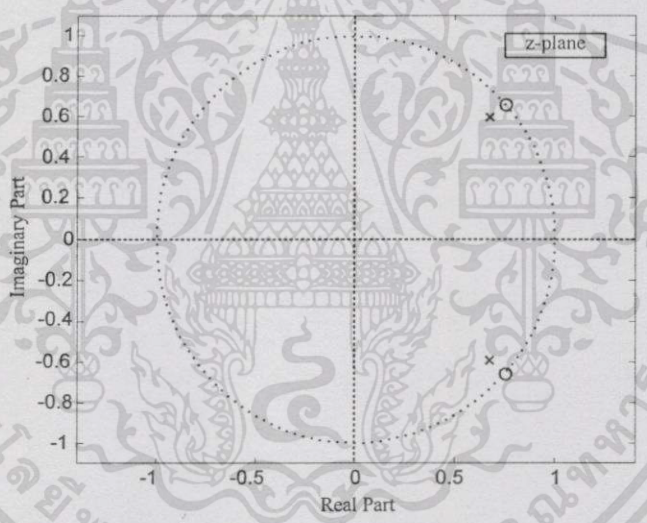
ความถี่นอตช์จะกำหนดได้ตามสมการ คือ

$$\omega_o = \cos^{-1}(-a/2) \tag{6.28}$$

จะได้ว่า $-2 < a < 2$ รูปที่ 6.1 แสดงผลตอบสนองทางขนาดต่อความถี่ $\left|\hat{H}(e^{j\omega})\right|$ โดยการปรับเปลี่ยนพารามิเตอร์ a และ ρ กราฟแสดงให้เห็นว่าเมื่อ $\rho \to 1$ จะทำให้แบนด์วิดของตัวกรองลดลง ทำให้ $\left|\hat{H}(e^{j\omega})\right|$ มีความใกล้เคียงกับผลตอบสนองทางความถี่ในอุดมคติ ของสมการที่ (6.16) และ a เป็นพารามิเตอร์ที่ใช้กำหนดความถี่นอตช์ตามสมการที่ (6.28) และรูปที่ 6.2 แสดงตำแหน่ง โพล-ซีโร่ ของสมการที่ (6.27) เมื่อกำหนดให้ a = -1.5 และ $\rho = 0.9$



รูปที่ 6.1 ผลตอบสนองทางขนาดต่อความถี่ $\left|\hat{H}(e^{j\omega})\right|$ เมื่อเปลี่ยนค่าสัมประสิทธิ์ ρ และ a



รูปที่ 6.2 ตำแหน่ง โพล-ซีโร่ ของ $\hat{H}(z)$ เมื่อ $\rho = 0.9, a = -1.5$

สมมติให้ความหนาแน่นสเปกตรัมของสัญญาณอินพุต เป็นดังสมการ

$$P_{xx}(e^{j\omega}) = \frac{A^2}{2}[\delta(\omega - \omega_1) + \delta(\omega + \omega_1)] + \sigma_w^2 \qquad (6.29)$$

ซึ่งประกอบด้วย สัญญาณคลื่นซายน์ และสัญญาณรบกวนขาว และ variance ของสัญญาณ เอาต์พุตจาก notch filter คือ

$$E\{\hat{y}^2(n)\} = \left\langle \hat{H}(z), \hat{H}(z) \right\rangle_{P_{xx}}$$
$$= A^2 \left| \hat{H}(e^{j\omega_1}) \right|^2 + \sigma_w^2 \left\| \hat{H}(z) \right\|_2^2 \tag{6.30}$$

กำหนดให้ ρ คงที่ และปรับเปลี่ยน a เพื่อให้ cost ฟังก์ชันตามสมการที่ (6.30) มีค่าต่ำสุด ในกรณีที่พิเศษ คือ ไม่มีสัญญาณรบกวน จะให้ค่า $\sigma_w^2 = 0$ และ $\left| \hat{H}(e^{j\omega_1}) \right|^2 = 0$ เท่านั้นจึงจะทำให้ cost ฟังก์ชันมีจุดต่ำสุด (Global minima) เพียงจุดเดียว หรือกล่าวอีกนัยหนึ่ง คือ ความถี่นอตช์ ω_o ของตัวกรอง ตรงกับความถี่ของสัญญาณอินพุต ω_1 พอดี สำหรับในทางปฏิบัติเมื่อ $\sigma_w^2 \neq 0$ จะต้องทำการพิจารณาสองข้อ คือ

i) เนื่องจาก cost ฟังก์ชัน $E\{\hat{y}^2(n)\}$ เป็น non-quadratic function ในตัวแปร a (เป็น nonlinear function) ดังนั้นจะต้องทำการตรวจสอบหา local minima ที่เกิดขึ้นด้วย

ii) ถ้ามีสัญญาณรบกวนเกิดขึ้น จะต้องทำการรวมพจน์ $\sigma_w^2 \left\| \hat{H}(z) \right\|_2^2$ เข้าไปในสมการของ cost ฟังก์ชันด้วย เพื่อตรวจสอบหาจุดต่ำสุดซึ่งมีการเปลี่ยนแปลงไปจากจุดเดิม

การพิจารณาในข้อ ii) พบว่า สัญญาณรบกวนอาจทำให้ ความถี่ที่ประมาณได้เกิดค่า bias ได้ จะแสดงให้เห็นว่า ในกรณีที่ไม่มีสัญญาณรบกวนนั้น cost ฟังก์ชันจะมีคุณสมบัติเป็น unimodal จากสมการที่ (6.30) กำหนดให้ $w(\cdot) \equiv 0$ และทำการหาอนุพันธ์ทั้งสองข้างเทียบกับสัมประสิทธิ์ a จะได้

$$\frac{\partial E\{\hat{y}^2(n)\}}{\partial a} = 2\left\langle \frac{\partial \hat{H}(z)}{\partial a}, \hat{H}(z) \right\rangle_{P_{xx}} = 0 \tag{6.31}$$

จาก [47] ได้แสดงวิธีการคำนวณสมการที่ (6.31) ไว้แล้ว นั่นคือ

$$\frac{\partial E\{\hat{y}^2(n)\}}{\partial a} = \frac{A^2}{2} \frac{(a + 2\cos\omega_1)[1 - \rho^3 + a\rho(1-\rho)\cos\omega_1 - \rho(1-\rho)\cos 2\omega_1]}{\left| D(e^{j\omega_1}) \right|^4} \tag{6.32}$$

เมื่อ $|D(\cdot)|$ คือขนาดของฟังก์ชันส่วน ของฟังก์ชันถ่ายโอนในสมการที่ (6.27) สมการที่ (6.32) จะเป็นศูนย์เมื่อ

$$a = -2\cos\omega_1 \tag{6.33}$$

และ

$$1 - \rho^3 + a\rho(1-\rho)\cos\omega_1 - \rho(1-\rho)\cos 2\omega_1 = 0 \tag{6.34}$$

อย่างไรก็ตาม สำหรับที่ $-2 < a < 2$ จะได้

$$1-\rho^3 + a\rho(1-\rho)\cos\omega_1 - \rho(1-\rho)\cos 2\omega_1 \geq (1-\rho^2) > 0 \qquad (6.35)$$

ถ้าเงื่อนไขเป็นไปตามสมการที่ (6.33) แล้ว จะสามารถหาสมการ cost ฟังก์ชัน สำหรับพารามิเตอร์ a ที่จุดต่ำสุดได้ ซึ่งจุดต่ำสุดเพียงจุดเดียวนี้จะเกิดขึ้นเฉพาะในกรณีที่ไม่มีสัญญาณรบกวนเท่านั้น (noise-free case)

ต่อไปจะพิจารณาถึงผลของสัญญาณรบกวน $\{w(\cdot)\}$ ถ้าสมมติให้เป็นสัญญาณรบกวนขาว จะได้ พจน์ของสัญญาณรบกวน จะเป็นดังนี้ คือ

$$\sigma_w^2 \left\| \hat{H}(z) \right\|_2^2$$

โดยที่ $\|\bullet\|_2$ คือ ตัวกระทำ norm-2 (L_2 -norm) และจะได้ว่า

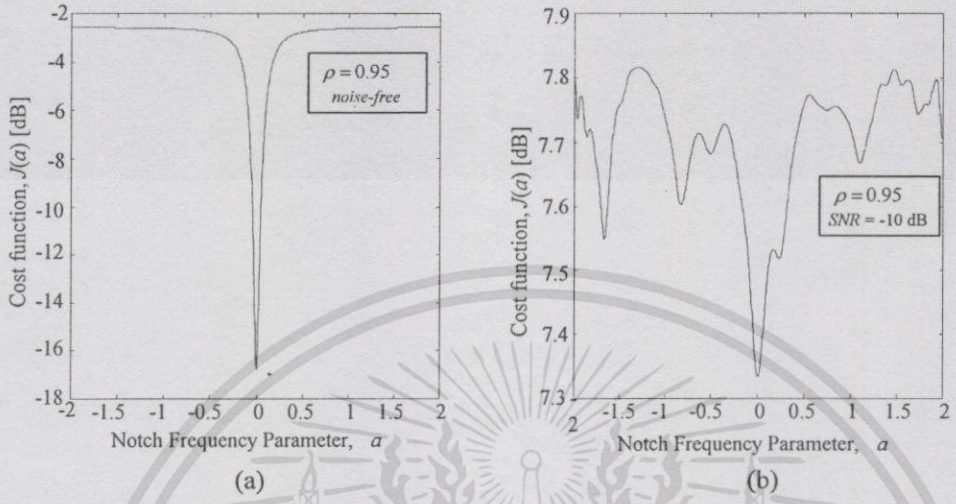
$$\left\| \hat{H}(z) \right\|_2^2 = 1 + \frac{(1-\rho)}{(1+\rho)(1+\rho^2)} \left(a^2 + \frac{[1+(1-a^2)\rho+\rho^2+\rho^3]^2}{1+(2-a^2)\rho^2+\rho^4} \right) \qquad (6.36)$$

ซึ่งในกรณีที่ $\rho < 1$ นั้น จะมีความยุ่งยากในการคำนวณ ทั้งนี้เพราะ จะต้องขึ้นกับอัตราการขยายของสัญญาณรบกวน $\left\| \hat{H}(z) \right\|_2^2$ ตามสมการที่ (6.36) ซึ่งเป็นฟังก์ชันของพารามิเตอร์ a ด้วย ในกรณีเช่นนี้ จุดต่ำสุดของ cost ฟังก์ชัน $E\{\hat{y}^2(n)\}$ จะอยู่ระหว่างจุดต่ำสุดขององค์ประกอบของ signal induced และ noise induced ของสัญญาณอินพุต ด้วยเหตุนี้เอง สัญญาณรบกวน $\{w(\cdot)\}$ ที่เกิดขึ้นนี้ จะทำให้ตำแหน่งจุดต่ำสุดมีการเปลี่ยนแปลงไปมา และจากสมการที่ (6.36) ถ้าทำการ take ลิมิตให้ $\rho \to 1$ แล้วจะได้ว่า

$$\lim_{\rho \to 1} \left\| \hat{H}(z) \right\|_2^2 = 1 \quad \text{สำหรับทุกค่า } a \in [-2,2] \qquad (6.37)$$

นั่นแสดงให้เห็นว่า สำหรับ $\rho \approx 1$ นั้น พจน์ของ noise induced ของ cost ฟังก์ชัน จะประมาณคงที่ เมื่อเทียบกับพารามิเตอร์ a ดังนั้น จุดต่ำสุดของ cost ฟังก์ชัน จะใกล้เคียงกับในกรณีที่ไม่มีสัญญาณรบกวน เพื่อให้ง่ายในการทำความเข้าใจ จะเขียน cost ฟังก์ชัน $E\{\hat{y}^2(n)\} \to J(a)$ ซึ่งหมายความว่า cost ฟังก์ชัน J นั้นเป็นฟังก์ชันของพารามิเตอร์ a รูปที่ 6.3 แสดง cost ฟังก์ชัน J (a) เมื่อความถี่ของสัญญาณอินพุตมีค่า $\omega_1 = \pi/2$ รูปที่ 6.3 (a) เป็นสภาวะที่ไม่มีสัญญาณรบกวน

และ รูปที่ 6.3 (b) เป็นสภาวะที่มีสัญญาณรบกวน, SNR = -10 dB จากรูปที่ 6.3 (b) จะเห็นว่า cost ฟังก์ชัน จะมี local minima เกิดขึ้นจำนวนมากมาย



รูปที่ 6.3 cost ฟังก์ชัน $J(a)$ ของ notch filter (a) noise-free case, (b) SNR = -10 dB

6.3 Least Mean *p*-Power Error Criterion (LMP)

จากหัวข้อ 6.2 ได้พิจารณาหา cost ฟังก์ชัน $J(a)$ ของ ANF อันดับสอง ซึ่ง cost ฟังก์ชันที่ได้ จะเป็นฟังก์ชันของพารามิเตอร์ a และ a เป็นพารามิเตอร์ที่ใช้กำหนดความถี่นอตช์ของตัวกรอง คำตอบของ a ที่ต้องการ คือ ค่า a ที่ทำให้ cost ฟังก์ชัน $J(a)$ มีค่าต่ำสุด (minimized) นั่นคือ จะทำการ minimize ค่าคาดหวังทางสถิติของสัญญาณเอาต์พุตยกกำลังสอง $E\{\hat{y}^2(n)\}$ ของตัวกรองนอตช์ ซึ่งข้อตัดสิน (criterion) ดังกล่าวนี้ จะเป็นที่รู้จักกันโดยทั่วไป นั่นคือ Least Mean Square (LMS) error sense เหตุผลที่ LMS เป็นที่นิยมใช้กันเนื่องจากความง่ายในทางคณิตศาสตร์ (mathematical simplicity)

ในช่วงหลายปีที่ผ่านมา การทำ minimize แบบ L_p -norm ($\|\bullet\|_p$:p-norm) [20, 35] และได้ถูกทดสอบถึงการทำงานค่อนข้างสมบูรณ์ และนำมาใช้กับการประยุกต์ใช้งานต่าง ๆ มากมาย เช่น การออกแบบตัวกรองความถี่, การทำ deconvolution, การเข้ารหัสเสียง และ การประมาณค่าความถี่ เป็นต้น โดยทั่วไป การออกแบบตัวกรองความถี่ จะต้องใช้การ minimize แบบ L_∞ norm [48] แต่เมื่อใช้สำหรับ ประมาณค่าความถี่ การ minimize แบบ L_1 -norm ($\|\bullet\|_1$:1-norm) จะดีกว่าโดยเฉพาะในกรณีที่สัญญาณรบกวนเป็นแบบอิมพัลส์ (impulse noise) [49] เนื่องจากการทำ minimize แบบ L_p มีทฤษฎีรองรับอย่างสมบูรณ์แล้ว จึงเป็นสิ่งที่น่าสนใจ หากนำมาใช้พัฒนาเป็นอะแดปทีฟอัลกอริทึมสำหรับ ANF ซึ่งโดยปกติใช้ข้อตัดสินแบบ LMS แต่จะใช้ ข้อตัดสินแบบ

Least Mean p-Power (LMP) Error Criterion แทน [12] ซึ่งพบว่าถ้าเลือกค่า p ที่เหมาะสมแล้ว เมื่อนำมาประยุกต์ใช้ในทางปฏิบัติหลาย ๆ อย่าง อะแดปทีฟอัลกอริทึมแบบ LMP จะมี สมรรถนะ ที่ดีกว่า อะแดปทีฟอัลกอริทึมแบบ LMS

เพื่อจะอธิบายข้อตัดสินแบบ LMP จะสมมติเช่นเดียวกับในหัวข้อ 6.2 คือ ให้สัญญาณอินพุตเป็นสัญญาณซายน์ความถี่เดียว ปนมากับสัญญาณรบกวนแบบขาว และ/หรือ สัญญาณรบกวนแบบอิมพัลส์ คือ

$$x(n) = A\sin(\omega_1 n + \theta) + v(n) \tag{6.38}$$

เมื่อ $v(n)$ คือ สัญญาณรบกวนแบบขาว หรือ อิมพัลส์ จากสมการที่ (6.27) พารามิเตอร์ a จะถูกปรับ ดังนั้นค่า mean p-Power error ของสัญญาณเอาต์พุตหรือ cost ฟังก์ชัน

$$J(a) = E\{|\hat{y}(n)|^p\} \tag{6.39}$$

จะถูกปรับจนมีค่าต่ำสุด สมการที่ (6.39) จะกลายเป็น LMS เมื่อให้ $p = 2$ การหาค่าของสมการที่ (6.39) ทำได้สองทาง คือ ใช้กระบวนการทางคณิตศาสตร์ตามหัวข้อ 6.2 และใช้การประมาณด้วยวิธีการของ Batch (Batch method) [50] ในที่นี้จะกล่าวเฉพาะ การประมาณตามวิธีที่สองเท่านั้น ดังนี้ คือ

ให้ N คือ จำนวนความยาวของข้อมูลอินพุต $x(n)$ และ $\hat{y}(n)$ คือ สัญญาณเอาต์พุตจากตัวกรอง แล้ว $J(a)$ จะสามารถประมาณได้ตามสมการ คือ

$$\hat{J}(a) = \frac{1}{N}\sum_{n=1}^{N}|\hat{y}(n)|^p \tag{6.40}$$

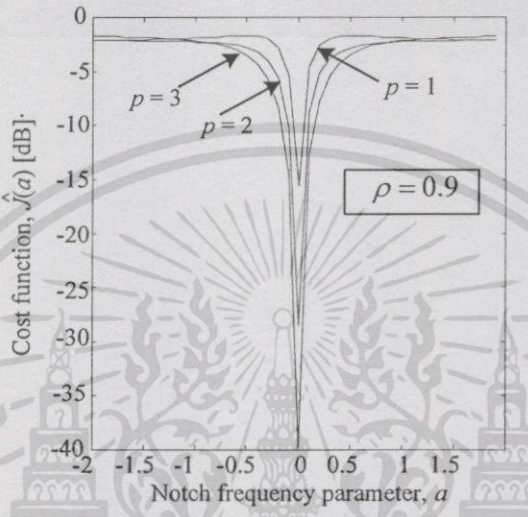
ดังนั้น จะสามารถประมาณคำตอบของพารามิเตอร์ a ได้ตามสมการ คือ

$$\hat{a} = \arg\min_a \hat{J}(a) \tag{6.41}$$

$\hat{a}$ หาได้โดยใช้ nonlinear programming optimization เช่น อัลกอริทึมแบบ steepest decent และ conjugate gradient เป็นต้น ต่อไปจะพิจารณาตัวอย่าง เมื่อสัญญาณอินพุตมีค่าตามสมการ คือ

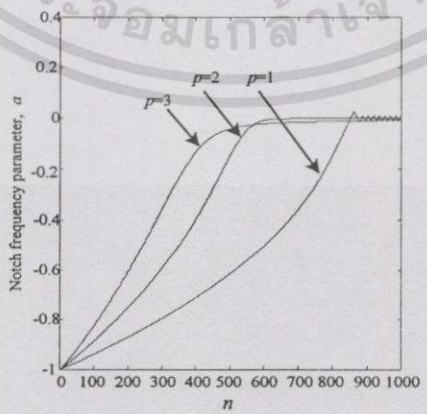
$$x(n) = \sin\left(\frac{\pi}{2}n + \frac{\pi}{2}\right) + v(n) \quad , n = 0,1,..., N \tag{6.42}$$

กำหนดให้ $v(n)$ เป็นสัญญาณรบกวนขาว มี Pdf เป็นแบบเกาส์เซียน มีค่า $\sigma_v^2 = 0.5$ ป้อนเข้าสู่ notch filter ในสมการที่ (6.27) โดยกำหนดให้ $\rho = 0.9$ รูปที่ 6.4 แสดงค่า $\hat{J}(a)$ เทียบกับ a ที่ค่า p =1, 2 และ 3



รูปที่ 6.4 ค่า mean p-Power error $\hat{J}(a)$ สำหรับ p = 1, 2 และ 3

จากรูปที่ 6.4 เห็นได้ชัดว่า ที่ p = 3 จะให้ค่า $\hat{J}(a)$ ต่ำสุด คือ –40 dB และ dip shape จะกว้างกว่าที่ค่า p = 2 และ p = 1 นั่นหมายความว่า เมื่อนำ steepest decent อัลกอริทึมมาใช้เพื่อหาจุดต่ำของ $\hat{J}(a)$ ที่ p มีค่าสูง จะทำให้อะแดปทีฟอัลกอริทึมทำงานได้เร็วเพิ่มขึ้น ดังแสดงในรูปที่ 6.5



รูปที่ 6.5 การลู่เข้าของพารามิเตอร์ a เมื่อใช้ steepest decent อัลกอริทึม ที่ p =1, 2 และ 3

6.4 Gradient-Based Algorithm แบบ Least Mean *p*-Power Error Criterion (LMP)

อะแดปทีฟอัลกอริทึมแบบ Gradient-Based เป็นรูปแบบหนึ่งของ Recursive Prediction Error (RPE) อัลกอริทึม ตามที่ได้กล่าวไว้ในบทที่ 5 อาศัยหลักการค้นหาจุดต่ำสุดของ cost ฟังก์ชัน ด้วยวิธีการแบบ steepest decent อัลกอริทึม [19] ซึ่งมีรูปแบบสมการดังนี้ คือ

$$a(n+1) = a(n) - \mu \frac{\partial J(a)}{\partial a} \tag{6.43}$$

เมื่อ μ คือ ค่าสเต็ปไซส์ ใช้กำหนดอัตราความเร็วของอะแดปทีฟอัลกอริทึม ขอบเขตของ μ สำหรับตัวกรองอะแดปทีฟแบบ IIR ยังไม่มีข้อสรุปที่แน่นอน ยังอยู่ในขั้นการค้นคว้าวิจัย ทั้งนี้เพราะ ความไม่เป็นเชิงเส้นของ cost ฟังก์ชัน ซึ่งจะไม่สามารถทำนายหรือทราบล่วงหน้าได้เลย จึงเป็นการยากที่จะสร้างเป็นสูตรสำเร็จเหมือนอย่างกับตัวกรองอะแดปทีฟแบบ FIR [38] ดังนั้น จึงต้องใช้วิธีการ ลองผิดลองถูก (trial and error) และ ปรับเปลี่ยนไปจนกว่าจะพอใจ

ในทางปฏิบัติ จะไม่สามารถสร้างอะแดปทีฟอัลกอริทึมตามสมการที่ (6.43) ได้ ทั้งนี้เพราะ cost ฟังก์ชัน $J(a) = E\{|\hat{y}(n)|^p\}$ เป็นค่าหวังทางสถิติ ซึ่งจะต้องคิดค่า $n = 0,1,2,...,\infty$ และเนื่องจาก สัญญาณในทางปฏิบัติ เป็นสัญญาณสุ่มแบบ nonstationary ซึ่งไม่สามารถคาดเดาอะไรได้มากนัก ดังนั้นจึงต้องใช้วิธีการประมาณแทน โดยการใช้ค่าของสัญญาณปัจจุบัน มาทำนายสัญญาณในอนาคต ค่าปัจจุบัน ณ เวลาหนึ่ง มักจะเรียกว่า instantaneous value โดย จะใช้ ค่า instantaneous ของ $J(a)$ เพื่อแทนค่าเฉลี่ย ensemble ของมัน ดังนั้น จะได้สมการสำหรับ ปรับค่าพารามิเตอร์ a ที่เป็นค่าประมาณของ steepest decent อัลกอริทึม ซึ่งเรียกชื่อใหม่ว่า Gradient-Based อัลกอริทึม ดังนี้ คือ

$$a(n+1) = a(n) - \mu \frac{\partial |\hat{y}(n)|^p}{\partial a} \tag{6.44}$$

สังเกตสมการที่ (6.31) ในหัวข้อที่ 6.2 ไม่มีตัวกระทำ $|\cdot|$ ทั้งนี้เพราะ ถึงอย่างไรค่า $\hat{y}^2(n)$ จะต้องเป็นค่าบวกทุกค่า ของ n อย่างแน่นอน ซึ่งแน่นอนทิศทางของ Gradient จะต้องเป็นลบเสมอ แต่สำหรับในกรณีของสมการที่ (6.44) p มีโอกาสเป็นเลขคี่ ซึ่ง ถ้าช่วงเวลาใดที่ $\hat{y}(n)$ มีค่าเป็นลบ และ p เป็นเลขคี่ ก็ทำให้ทิศทางของ Gradient เป็นบวก-ลบ สลับกัน ขึ้นกับสัญญาณ $\hat{y}(n)$ จะทำให้ อะแดปทีฟอัลกอริทึม ไม่สามารถค้นหาจุดต่ำสุดของ cost ฟังก์ชันได้เลย ดังนั้น จึงต้องใช้ตัวกระทำ $|\cdot|$ เพื่อบังคับให้ $\hat{y}^p(n)$ มีค่าเป็นบวก สำหรับทุกค่าของ p เพื่อทำให้ทิศทางของ Gradient เป็นลบตลอดการทำงาน ดังนั้น จะได้ว่า

$$|\hat{y}(n)|^p = \begin{cases} [\hat{y}(n)]^p & p:even \\ \text{sgn}[\hat{y}(n)]\cdot[\hat{y}(n)]^p & p:odd \end{cases} \tag{6.45}$$

ดังนั้นอนุพันธ์ย่อยของ $|\hat{y}(n)|^p$ เทียบกับพารามิเตอร์ a คือ

$$\frac{\partial|\hat{y}(n)|^p}{\partial a} = \begin{cases} p\cdot[\hat{y}(n)]^{p-1}\cdot\dfrac{\partial\hat{y}(n)}{\partial a} & p:even \\ p\cdot\text{sgn}[\hat{y}(n)]\cdot[\hat{y}(n)]^{p-1}\cdot\dfrac{\partial\hat{y}(n)}{\partial a} & p:odd \end{cases} \tag{6.46}$$

โดยที่ $\text{sgn}[x] \equiv \dfrac{x}{|x|}$ ดังนั้นสมการที่ (6.44) จะกลายเป็น

$$\begin{aligned} a(n+1) &= a(n) - \mu\cdot U_1(n) \qquad p:even \\ a(n+1) &= a(n) - \mu\cdot U_2(n) \qquad p:odd \end{aligned} \tag{6.47}$$

เมื่อ

$$\begin{aligned} U_1(n) &= p\cdot[\hat{y}(n)]^{p-1}\cdot\hat{e}(n) \\ U_2(n) &= p\cdot\text{sgn}[\hat{y}(n)]\cdot[\hat{y}(n)]^{p-1}\cdot\hat{e}(n) \end{aligned} \tag{6.48}$$

และ

$$\hat{e}(n) = \frac{\partial\hat{y}(n)}{\partial a} \tag{6.49}$$

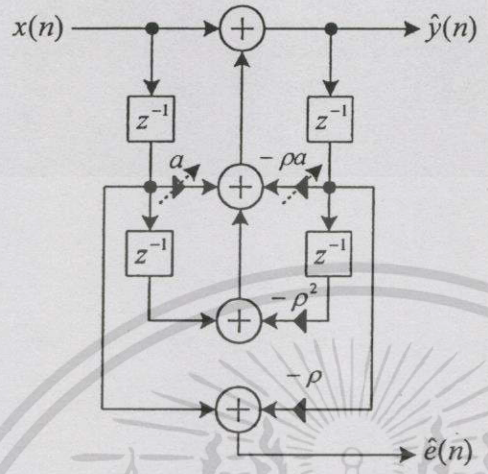
สมการของ $\hat{y}(n)$ หาได้โดยทำการแปลง z กลับ (inverse z-transform) ของสมการที่ (6.27) ดังนี้

$$\hat{y}(n) = x(n) + ax(n-1) + x(n-2) - \rho a y(n-1) - \rho^2 y(n-2) \tag{6.50}$$

และสมการของ $\hat{e}(n)$ หาได้โดยใช้วิธีการของ pseudolinear regression algorithm [18] ซึ่งอธิบายไว้ในบทที่ 5 ดังนี้ คือ

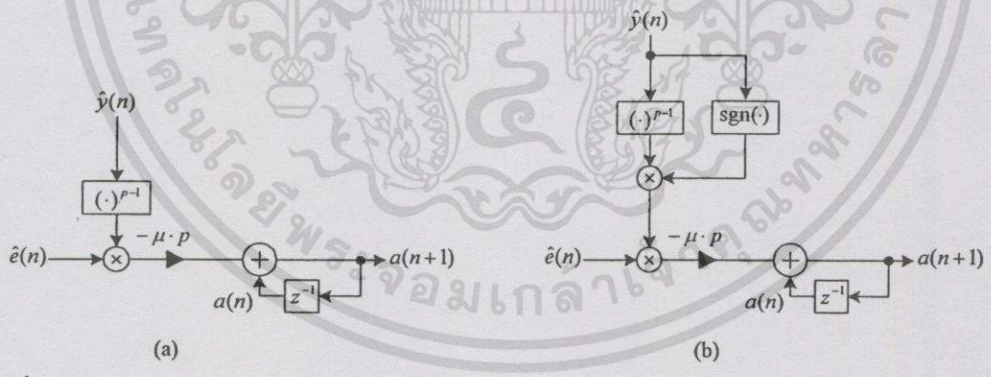
$$\hat{e}(n) = \frac{\partial\hat{y}(n)}{\partial a} \approx x(n-1) - \rho y(n-1) \tag{6.51}$$

จากที่กล่าวมานี้ คืออะแดปทีฟอัลกอริทึมที่นำเสนอในบทวิจัย [12] รูปที่ 6.6 แสดงโครงสร้างแบบ Direct form ของ ANF



รูปที่ 6.6 โครงสร้างแบบ Direct form ของตัวกรอง ANF

และรูปที่ 6.7 แสดงโครงสร้างของอะแดปทีฟอัลกอริทึม รูปที่ 6.8 (a) เมื่อ p เป็นเลขคู่ และ รูปที่ 6.7 (b) เมื่อ p เป็นเลขคี่



รูปที่ 6.7 โครงสร้างของอะแดปทีฟอัลกอริทึมแบบ LMP (a) เมื่อ p เป็นเลขคู่ (b) เมื่อ p เป็นเลขคี่

6.5 อะแดปทีฟอัลกอริทึมที่นำเสนอในวิทยานิพนธ์

วิทยานิพนธ์นี้ นำเสนออะแดปทีฟอัลกอริทึม ที่ได้ทำการดัดแปลงจาก LMP สามแบบ คือ QLMP (Quantized LMP) [15-16], NQLMP (Normalize Quantized LMP) [17] และ VSQLMP (Variable Step-size Quantized LMP) [43] อะแดปทีฟอัลกอริทึม แบบต่าง ๆ จะมีข้อดี ข้อเสีย แตก

ต่างกัน ขึ้นกับงานที่จะไปใช้ อย่างไรก็ตาม การจะเลือกว่าจะนำอะแดปทีฟอัลกอริทึมแบบใดมาใช้นั้น จะขึ้นกับเหตุผล 4 ประการ ดังกล่าวไว้ในบทที่ 4 คือ

i) ความเร็วในการหาคำตอบ

ii) ความถูกต้องของคำตอบ (ดูจาก variance และ bias)

iii) ความสามารถในการติดตามสัญญาณสุ่มแบบ nonstationary

iv) ความซับซ้อนในการคำนวณ

ดังนั้นหาก อะแดปทีฟอัลกอริทึมแบบใด ที่ให้คุณสมบัติทั้ง 4 ข้อนี้ ก็จะเลือกนำมาใช้งาน

6.5.1 อะแดปทีฟอัลกอริทึมแบบ QLMP

สมการสำหรับปรับพารามิเตอร์ a ของอะแดปทีฟอัลกอริทึมแบบ QLMP ที่ได้ดัดแปลงจาก LMP คือ

$$\begin{aligned} a(n+1) &= a(n) - \mu \cdot \text{sgn}[U_1(n)] \qquad p: even \\ a(n+1) &= a(n) - \mu \cdot \text{sgn}[U_2(n)] \qquad p: odd \end{aligned} \tag{6.52}$$

โดยที่ $U_i(n), i = 1,2$ มีค่าตามสมการที่ 6.48 วิธีการนี้ อาศัยตัวกระทำ sgn[·] สำหรับตรวจสอบเครื่องหมายของ $U_i(n)$ ถ้า $U_i(n)$ มีค่าเป็นบวก จะได้ค่า $\text{sgn}[U_i(n)] = 1$ และ ถ้า $U_i(n)$ มีค่าเป็นลบ จะได้ค่า $\text{sgn}[U_i(n)] = -1$ หรือกล่าวอีกนัยหนึ่ง คือ จะทำการ Quantize ค่า $U_i(n)$ ให้เพียงสองระดับ คือ 1 และ –1 ข้อดี คือ จะสามารถลดการคูณลงได้ 1 ครั้ง ต่อ 1 ตัวอย่าง (sample) สัญญาณเอาต์พุต (เทียบกับ LMP ที่ $p > 1$) เพราะ μ จะถูกคูณด้วย 1 หรือ –1 ซึ่งปกติแล้วจะไม่นับว่าเป็นการคูณ ในกรณีที่เลือกให้ $p = 1$ จะได้

$$U_i(n) = U_2(n) = \text{sgn}[\hat{y}(n)] \cdot \hat{e}(n) \tag{6.53}$$

และสมการสำหรับปรับพารามิเตอร์ a คือ

$$a(n+1) = a(n) - \mu \cdot \text{sgn}[\text{sgn}\{\hat{y}(n)\} \cdot \hat{e}(n)] \qquad p = 1 \tag{6.54}$$

ซึ่งจะลดการคูณลงได้ 2 ครั้ง (เทียบกับ LMP ที่ $p \neq 1$) นอกจากนี้ ยังพบว่า ความเร็วในการทำงานของอะแดปทีฟอัลกอริทึมจะขึ้นกับ ค่า μ เท่านั้น ดังนั้นในกรณีที่ใช้ μ เท่ากัน อะแดป

ทีฟอัลกอริทึมแบบ QLMP จะทำงานได้เร็วกว่า แบบ LMP และ ความเร็วจะไม่ขึ้นกับค่า p อีกด้วย ซึ่งจะต่างจากแบบ LMP เพราะเมื่อ p สูงขึ้น จะทำให้ความเร็วในการทำงานเพิ่มขึ้น แต่อย่างไรก็ตาม อะแดปทีฟอัลกอริทึมแบบ QLMP จะมีข้อเสีย คือ พารามิเตอร์ a จะมีค่า Gradient noise [19, 38] สูงกว่าแบบ LMP เมื่อค่า μ เท่ากัน นั่นหมายความว่า variance ของ พารามิเตอร์ a มีค่าสูงกว่านั่นเอง

สรุปได้ว่า ข้อดีสองประการของอะแดปทีฟอัลกอริทึมแบบ QLMP คือ (i) ลดความซับซ้อนในการคำนวณลงได้ โดยเฉพาะที่ $p = 1$ จะดีที่สุด ดังนั้น ในการใช้งานควรใช้ $p = 1$ (ii) คือ ค้นหาคำตอบได้เร็ว

6.5.2 อะแดปทีฟอัลกอริทึมแบบ NQLMP

อะแดปทีฟอัลกอริทึมในหัวข้อนี้ ดัดแปลงจาก อะแดปทีฟอัลกอริทึมในหัวข้อ 6.5.1 โดยใช้ ค่าพลังงาน (energy) ในแต่ละเวลา n ของ $U_i(n)$ มาทำการนอร์มอลไลซ์ (คือการหาร) ค่า μ ทำให้ step - size ของอะแดปทีฟอัลกอริทึมแบบ NQLMP เปลี่ยนแปลงตามเวลา ดังนี้คือ

เมื่อ $\mu \rightarrow \mu_n$

$$\mu_n = \frac{\mu}{\varepsilon + \varphi_i(n)} \tag{6.55}$$

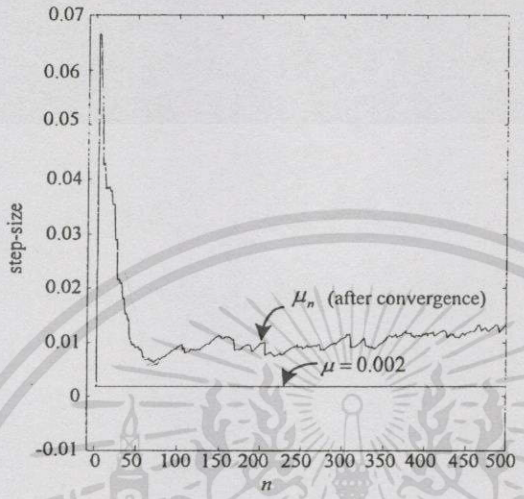
ε เป็นค่าคงที่น้อย ๆ มีไว้เพื่อป้องกันการหารด้วยศูนย์ และ

$$\varphi_i(n) = \alpha \cdot \varphi(n-1) + (1-\alpha) \cdot U_i^2(n) \quad ,i = 1, 2 \tag{6.56}$$

เป็นสมการค่าพลังงานแบบ recursive ของ $U_i(n)$, α เป็นค่าคงที่ เรียกว่า exponential weighting พารามิเตอร์ หรือ forgetting factor ใช้กำหนดคุณภาพของการประมาณ สำหรับสิ่งแวดล้อมที่เป็นแบบ stationary จะต้องกำหนดให้ $\alpha \approx 1$ แทนค่าในสมการที่ (6.55) ลงในสมการที่ (6.52) จะได้

$$\begin{aligned} a(n+1) &= a(n) - \mu_n \cdot \mathrm{sgn}[U_1(n)] \qquad p : even \\ a(n+1) &= a(n) - \mu_n \cdot \mathrm{sgn}[U_2(n)] \qquad p : odd \end{aligned} \tag{6.57}$$

สับสคริป n บอกว่า step-size เปลี่ยนตามเวลา ในช่วงแรกของการปรับ ค่า $\varphi_i(n)$ จะมีค่าน้อย ทำให้ μ_n มีขนาดสูง จึงทำให้อะแดปทีฟอัลกอริทึมนี้ หาคำตอบได้เร็ว หลังจากพบคำตอบแล้ว จะทำให้ $\varphi_i(n)$ มีค่าเพิ่มขึ้นอย่างรวดเร็ว เป็นผลให้ค่า μ_n มีค่าต่ำลง แต่ยังมีค่าสูงกว่า μ ตัวเดิมที่กำหนดไว้ ดังตัวอย่างในรูปที่ 6.8



รูปที่ 6.8 ค่า step – size ของอะแดปทีฟอัลกอริทึมแบบ QLMP และ NQLMP

จากรูปจะเห็นว่า หลังจากที่อะแดปทีฟอัลกอริทึมลู่เข้า จะทำให้ สเต็ปไซส์มีค่าสูงกว่าค่าเริ่มต้น ดังนั้นอะแดปทีฟอัลกอริทึม NQLMP จะมี Gradient noise สูงกว่า QLMP แต่

จะทำงานได้เร็วกว่า นอกจากนี้ อะแดปทีฟอัลกอริทึมแบบ NQLMP ยังต้องการคำนวณที่เพิ่มขึ้น คือ เพิ่มการคูณ 3 ครั้งและเพิ่มการหาร 1 ครั้ง ต่อหนึ่งตัวอย่าง และเนื่องจาก $U_i(n)$ เป็นฟังก์ชันของค่า p จึงทำให้ $\varphi_i(n)$ ขึ้นกับค่า p ด้วย โดยที่ค่า p สูง ๆ จะมี Gradient สูงตามไปด้วย สรุปได้ว่า อะแดปทีฟอัลกอริทึมแบบ NQLMP จะมีข้อดีคือ ทำงานได้เร็ว แต่ข้อเสียคือ มีความซับซ้อนในการคำนวณ และคำตอบมีค่า variance สูง หากจะนำไปใช้งาน ควรจะเลือกให้ $p = 1$ เช่นเดียวกับ QLMP

6.5.3 อะแดปทีฟอัลกอริทึมแบบ VSQLMP

อะแดปทีฟอัลกอริทึมแบบ VSQLMP นี้ได้นำข้อดีของการทำให้ step – size เปลี่ยนแปลงตามเวลา คือ จะทำงานได้เร็ว โดยจะสร้างสมการสำหรับ ปรับ step – size แบบ recursive โดยตรง ไม่ต้องใช้การหาร การทำเช่นนี้ จะสามารถลด Gradient noise ลงได้ เมื่อเทียบกับแบบ QLMP และ NQLMP นั่นหมายความว่า ค่า variance ของคำตอบมีค่าต่ำกว่าด้วย จากสมการที่ 6.52 ทำการแทน $\mu \rightarrow \mu(n)$ ดังนั้นจะได้สมการสำหรับปรับพารามิเตอร์ a ใหม่คือ

$$a(n+1) = a(n) - \mu(n) \cdot \text{sgn}[U_1(n)] \qquad p : even$$
$$a(n+1) = a(n) - \mu(n) \cdot \text{sgn}[U_2(n)] \qquad p : odd \tag{6.58}$$

เมื่อ

$$\mu(n+1) = \gamma \cdot \mu(n) + \lambda \cdot \Psi^2(n) \tag{6.59}$$

โดยที่

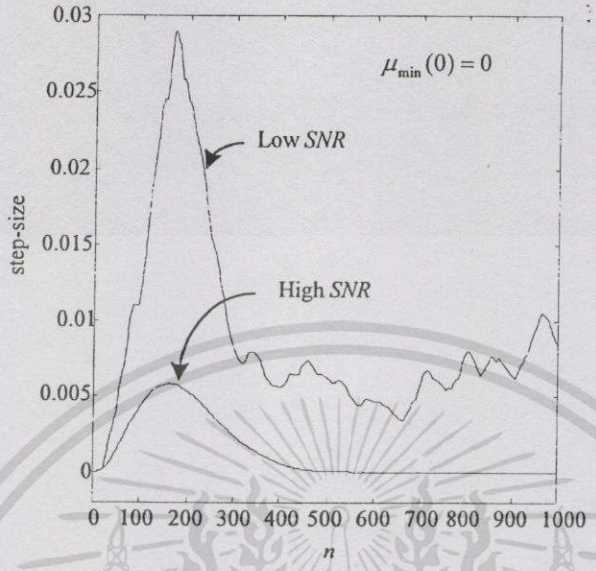
$$\Psi(n) = \alpha \cdot \Psi(n-1) + (1-\alpha) \cdot \hat{y}^2(n) \tag{6.60}$$

γ, λ เป็นค่าคงที่อยู่ในช่วง [0, 1] และ $\alpha \approx 1$ วิธีการนี้ จะใช้ค่าพลังงานประมาณ ของ $\hat{y}(n)$ มาใช้สำหรับควบคุมการปรับ step – size ให้เปลี่ยนตามเวลา ซึ่งได้ดัดแปลงมาจากบทวิจัย [51] โดยในบทวิจัย [51] ได้ใช้ค่า Autocorrelation ประมาณระหว่าง ค่าความผิดพลาด $e(n)$ กับ $e(n-1)$ โดยได้ทำการพัฒนาบนอะแดปทีฟอัลกอริทึมแบบ LMS ซึ่งใช้กับตัวกรองอะแดปทีฟแบบ FIR ในวิทยานิพนธ์ได้นำแนวความคิดดังกล่าวมาใช้กับตัวกรองอะแดปทีฟแบบ IIR โดยอยู่บนพื้นฐานอะแดปทีฟอัลกอริทึมแบบ LMP ซึ่งยังไม่เคยมีการนำมาใช้ก่อนหน้านี้เลย

และเนื่องจากค่า step – size จะปรับตัวตามค่าพลังงานของสัญญาณเอาต์พุต $\hat{y}(n)$ ในช่วงแรก ขณะที่ ANF เริ่มทำงาน ยังไม่สามารถตรวจวัดความถี่ได้ ค่าพลังงานของ $\hat{y}(n)$ มีค่าสูง เป็นผลให้ $\mu(n)$ มีค่าสูง จึงทำงานได้เร็ว หลังจากลู่เข้าแล้ว ค่าพลังงานของ $\hat{y}(n)$ มีค่าต่ำ โดยจะเหลือเฉพาะพลังงานของสัญญาณรบกวนเท่านั้น ดังนั้น ในกรณีที่สัญญาณรบกวนมีค่าต่ำ จะทำให้ $\mu(n)$ มีค่าต่ำมากด้วย ส่งผลให้ Gradient noise ลดลง และ $\mu(n)$ จะเพิ่มเมื่อสัญญาณรบกวนมีค่าพลังงานเพิ่มขึ้น ส่งผลให้ Gradient noise เพิ่มสูงตามด้วย แต่จะไม่มีผลต่อความเร็วในการทำงาน ข้อดีอีกอย่างหนึ่งก็คือ จะสามารถกำหนดค่า $\mu(0)$ ได้ตามต้องการ คือ $\mu_{\min}(0)$ หรือ $\mu_{\max}(0)$ แต่อย่างไรก็ตาม ถ้าให้ $\mu_{\max}(0)$ มากเกินไป ก็อาจทำให้ อะแดปทีฟอัลกอริทึมไม่สามารถลู่เข้าสู่คำตอบที่ต้องการได้ (divergence) เลย รูปที่ 6.9 แสดงตัวอย่างค่า $\mu(n)$ ของ VSQLMP ในกรณีที่สัญญาณรบกวนต่ำและในกรณีที่สัญญาณรบกวนมีค่าสูง

อะแดปทีฟอัลกอริทึมในหัวข้อนี้ พบว่า มีสองสมการที่เพิ่มเข้ามา คือ สมการที่ (6.59) และสมการที่ (6.60) นั่นหมายความว่า จะมีการคำนวณเพิ่มขึ้น จำนวนการคูณ หาร บวก และ ลบ ของแต่ละอัลกอริทึม ในกรณีที่ p = 1, 2 และ 3 แสดงดังตารางที่ 6.1 คือ พบว่า อะแดปทีฟอัลกอริทึมแบบ QLMP จะมีความซับซ้อนในการคำนวณน้อยที่สุด และ VSQLMP จะมีความซับซ้อนในการคำนวณมากที่สุด สำหรับการเลือกอะแดปทีฟอัลกอริทึมแบบ QLMP, NQLMP และ VSQLMP มาใช้นั้น ควรจะกำหนดให้ p = 1 จะเหมาะสมที่สุด เพราะค่า p ที่เพิ่มขึ้น ไม่ได้ทำให้ประสิทธิภาพของอัลกอริทึมดีขึ้น ในบางอัลกอริทึม อาจทำให้เลวลง เช่น NQLMP เป็นต้น ถึงแม้ว่า อะแดปทีฟอัลกอริทึมแบบ VSQLMP จะมีความซับซ้อนที่สุด แต่จากสมการที่ (6.59) หากกำหนดให้ γ และ λ มีค่าเท่ากับ 2^{-k} เมื่อ k อยู่ในช่วง $[0,\infty]$ จะสามารถลดการคูณลงได้ 2 ครั้ง โดยจะใช้การเลื่อน

(shift) บิตแทน ดังนั้นในกรณีที่เลือก $p = 1$ จะเหลือการคูณเพียง 4 ครั้ง สำหรับการนำไปประยุกต์ใช้งาน, ผลการจำลองการทำงาน และผลการทดลองจริง จะกล่าวถึงในบทที่ 7



รูปที่ 6.9 step-size $\mu(n)$ ในสภาวะสัญญาณรบกวนต่ำและสูง

ตารางที่ 6.1 ความซับซ้อนในการคำนวณของอัลกอริทึมแต่ละแบบสำหรับ p =1, 2 และ 3

$p = 1$	การคูณ	การหาร	การบวก-ลบ
LMP	2	-	2
QLMP	1	-	2
NQLMP	3	1	3
VSQLMP	6	-	4
$p = 2$	การคูณ	การหาร	การบวก-ลบ
LMP	8	-	6
QLMP	7	-	6
NQLMP	10	1	7
VSQLMP	13	-	8
$p = 3$	การคูณ	การหาร	การบวก-ลบ
LMP	9	-	6
QLMP	8	-	6
NQLMP	11	1	7
VSQLMP	14	-	8

6.6 สรุป

ในบทนี้ได้กล่าวถึง Adaptive IIR Notch Filter ที่ใช้โครงสร้างแบบ Direct form และอะแดปทีฟอัลกอริทึมแบบ Gradient-Based ที่ใช้ข้อตัดสินของคำตอบแบบ LMP (Least Mean p-Power Error Criterion) โดยใช้ฟังก์ชันถ่ายโอน $H(z)$ แบบ minimum parameter ANF [45] คือ ได้ออกแบบให้มีพารามิเตอร์เพื่อในการปรับเท่ากับจำนวนของสัญญาณคลื่นซายน์ทางอินพุตของตัวกรอง นั่นคือ ถ้าสัญญาณซายน์ที่ต้องการประมาณมีเพียงสัญญาณเดียว ก็จะมีพารามิเตอร์ที่ใช้ปรับเพียงตัวเดียว ฟังก์ชันถ่ายโอนที่นำมาใช้นี้ จะบังคับให้ ซีโร่ อยู่บนวงกลมหนึ่งหน่วย และ โพล ถูกบังคับให้อยู่ภายในวงกลมหนึ่งหน่วยใกล้กับซีโร่ และอยู่ในแนวรัศมีเดียวกัน การออกแบบในลักษณะนี้ มีข้อดีหลายประการ เช่น มีความเสถียรภาพสูง หมายความว่า โพลจะถูกคูณด้วยแฟคเตอร์ที่ทำให้ไม่สามารถออกนอกวงกลมหนึ่งหน่วยได้เลย และส่งผลให้ ไม่ต้องทำ stability monitoring ทำให้ลดความซับซ้อนในการคำนวณลงได้ คำตอบที่ได้มีความถูกต้องสูง หมายความว่า พารามิเตอร์ที่ประมาณได้มีค่า variance และ bias ต่ำ ทำงานได้เร็ว หมายความว่า ได้ทำการเชื่อมโยงพารามิเตอร์ระหว่าง ตัวเศษและตัวส่วนของฟังก์ชันถ่ายโอน ทำให้ สมการ output error equation มีความเป็นเชิงเส้นสูง และการที่โพลและซี่โร่ถูกบังคับให้อยู่ใกล้กัน และอยู่ในแนวรัศมีเดียวกัน จะทำให้แบนด์วิดแคบ ทำให้ ANF มีความเป็นอุดมคติมากขึ้น โดยทั้ง ซีโร่และโพล จะลู่เข้าหาสัญญาณซายน์เหมือนกัน

สำหรับอะแดปทีฟอัลกอริทึม ที่ได้นำเสนอในวิทยานิพนธ์นี้ ได้ปรับปรุงจากอะแดปทีฟอัลกอริทึมในบทวิจัย [12] ซึ่งเรียกว่า LMP โดยได้ทำการปรับปรุงเป็น 3 แบบ คือ QLMP, NQLMP และ VSQLMP ซึ่งการปรับปรุงอัลกอริทึมส่วนใหญ่ มักจะทำกันในลักษณะที่ต้องการเพิ่มความเร็วในการทำงาน แม้ว่าจะต้องเพิ่มความซับซ้อนในการคำนวณก็ตาม นอกจากนี้ ยังมีความต้องการลด Gradient noise ให้มีค่าต่ำที่สุด เพราะจะส่งผลให้ ค่า variance ของคำตอบมีค่าต่ำด้วย นอกจาก variance แล้ว ยังพิจารณาที่ค่า bias ด้วย นั่นหมายความว่า อะแดปทีฟอัลกอริทึมที่ดี จะต้องให้ ค่า variance และ bias เข้าใกล้ Cramer Rao bound (CRB) [34] อะแดปทีฟอัลกอริทึมแบบ QLMP จะมีความซับซ้อนในการคำนวณต่ำสุด, ทำงานได้เร็ว แต่ให้ค่า variance สูง อะแดปทีฟอัลกอริทึมแบบ NQLMP จะทำงานได้เร็ว แต่ให้ค่า variance สูงกว่า QLMP และยังมีการหาร ซึ่งจะมีปัญหา เมื่อใช้งานที่เวลาจริง (realtime) และ VSQLMP จะทำงานได้เร็วพอ ๆ กับ NQLMP แต่จะให้ค่า variance ต่ำมาก โดยเฉพาะเมื่อ SNR มีค่าสูง จะทำงานได้ดีมาก แต่จะมีความซับซ้อนมากที่สุด

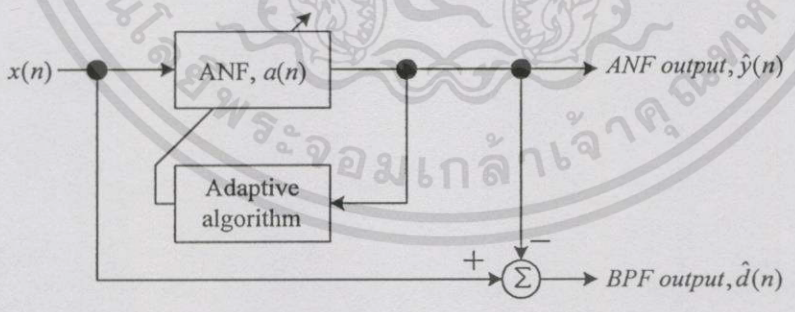
บทที่ 7
ผลการวิจัย

ในบทนี้จะกล่าวถึง ผลการทำงานของอะแดปทีฟอัลกอริทึม ที่ได้กล่าวไว้ในบทที่ 6 คือ ในหัวข้อ 6.4 และ หัวข้อ 6.5 โดยจะทดสอบการทำงานโดยการจำลองการทำงาน (simulation) ด้วยโปรแกรม MATLAB และ ทดลองที่เวลาจริง (realtime) ด้วยตัวประมวลผลสัญญาณดิจิตอล TMS320C50 DSP Starter Kit (DSK) บอร์ด เพื่อความชัดเจน จะทำการทดลองโดยการเลียนแบบการทำงาน และ ทดลองที่เวลาจริง ไปพร้อมกัน สำหรับรายละเอียดเกี่ยวกับ DSK ตลอดจนโปแกรมต่าง ๆ จะแสดงในภาคผนวก ก. และ ข. ตามลำดับ

7.1 การประยุกต์ใช้งาน ANF

7.1.1 การประมาณค่าสัญญาณซายน์คลื่นเดี่ยวในสัญญาณรบกวน

หัวข้อนี้ เป็นการประยุกต์ใช้งาน ANF สำหรับการประมาณค่าสัญญาณซายน์ที่ไม่ทราบความถี่ในสัญญาณรบกวนสองชนิด คือ สัญญาณรบกวนแบบเกาส์เซียน (White Gaussian Noise, WGN) และ สัญญาณรบกวนแบบอิมพัลส์ (impulse noise) โดยจะทำการเปรียบเทียบผลการทำงานของอะแดปทีฟอัลกอริทึมแต่ละแบบ โดยใช้เงื่อนไขในการทดสอบให้เหมือนกัน ซึ่งจะทำให้สามารถเปรียบเทียบสมรรถนะของอะแดปทีฟอัลกอริทึมแต่ละแบบได้อย่างชัดเจนบล๊อกไดอะแกรม สำหรับการประมาณค่าความถี่ แสดงดังรูปที่ 7.1



รูปที่ 7.1 บล๊อกไดอะแกรมสำหรับการประมาณค่าสัญญาณซายน์ในสัญญาณรบกวนโดยใช้ ANF

จากรูป $x(n)$ คือสัญญาณอินพุต ประกอบด้วยสัญญาณซายน์ความถี่เดียวและสัญญาณรบกวน

$$x(n) = A\sin(\omega_1 n + \theta) + v(n) \tag{7.1}$$

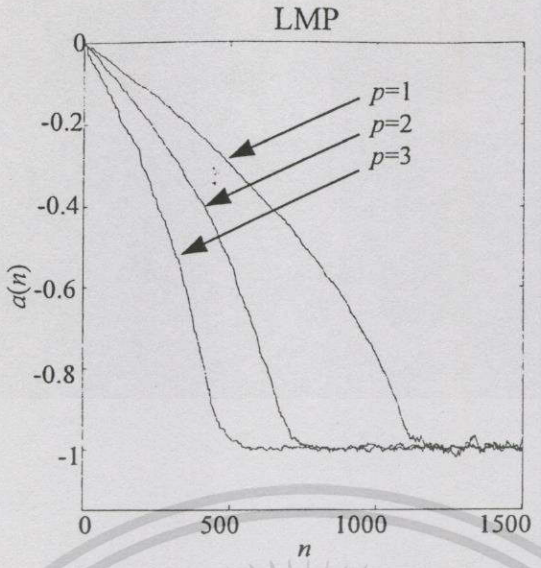
A คือขนาดของสัญญาณ, ω_1 เป็นความถี่ดิจิตอลที่ไม่ทราบค่า มีค่าอยู่ในช่วง $[0,\pi]$, θ คือ เฟสของสัญญาณมีค่าอยู่ในช่วง $[0,2\pi]$, $v(n)$ คือ สัญญาณรบกวนแบบเกาส์เซียน หรือ แบบอิมพัลส์, $\hat{y}(n)$ เป็นสัญญาณเอาต์พุตของ ANF ซึ่งหลังจากลู่เข้าแล้ว $\hat{y}(n) \approx v(n)$ ความถี่ที่ประมาณได้ จะดูได้จาก สัมประสิทธิ์ a และ $\hat{d}(n)$ เป็นสัญญาณเอาต์พุตของ Band-Pass Filter (BPF) ถ้าการประมาณทำได้อย่างสมบูรณ์ จะได้ $\hat{d}(n) \approx A\sin(\omega_1 n + \theta)$ ในการตรวจวัดความถี่จะสามารถดูได้ทั้งจาก สัมประสิทธิ์ a ซึ่งมีความสัมพันธ์กับ ความถี่นอตช์ ตามสมการที่ (6.28) และจาก $\hat{d}(n)$

7.1.2 ผลการจำลองการทำงาน และผลการทดลองที่เวลาจริง

สำหรับการทดลองที่เวลาจริง จะกำหนดใช้อัตราสุ่มเป็น 8 kHz จากทฤษฎีการสุ่ม (sampling theorem) ทำให้ได้ความถี่ที่สามารถตรวจวัดได้จะอยู่ในช่วงประมาณ 0-4 kHz ซึ่งเรียกความถี่ในช่วงนี้ว่า "ความถี่ไนควิส (Nyquist frequency)" ความถี่ 0-4 kHz นี้เมื่อแปลงเป็นสัญญาณดิจิตอลแล้วจะมีค่าอยู่ในช่วง $[0,\pi]$ ในการจำลองการทำงานด้วยคอมพิวเตอร์ จะกำหนดให้มีค่า $\omega_1 = \frac{\pi}{3}$ ซึ่งจะสัมพันธ์กับความถี่ของสัญญาณแอนะลอกอินพุตประมาณ 1 kHz (เมื่อใช้อัตราสุ่มเท่ากับ 8 kHz) และสัมประสิทธิ์ "a" ของ ANF จะมีค่าเท่ากับ –1 โดยในการทดลองนี้จะเปรียบเทียบ ความเร็วในการลู่เข้าสู่คำตอบของอะแดปทีฟอัลกอริทึมแบบต่าง ๆ ที่ได้นำเสนอในวิทยานิพนธ์

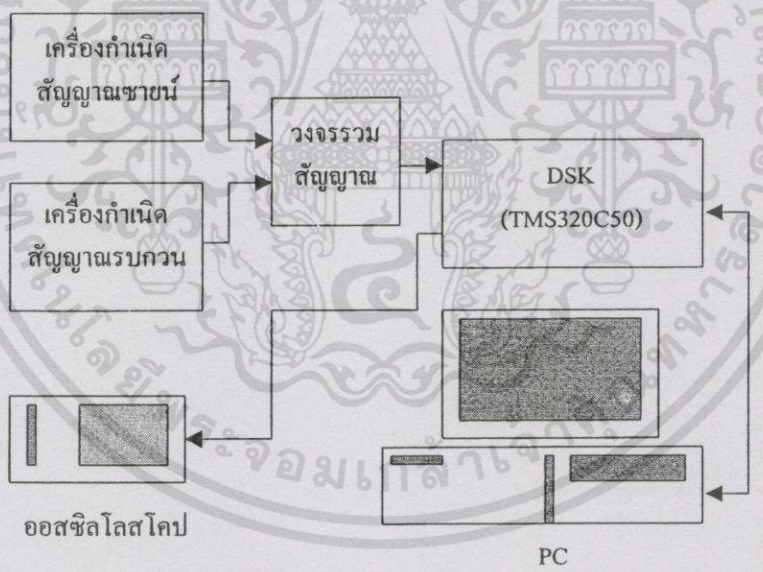
7.1.2.1 ผลการทำงานของอะแดปทีฟอัลกอริทึมแบบ LMP

กำหนดพารามิเตอร์ สำหรับการจำลองการทำงานดังนี้ คือ กำหนดความถี่ดิจิตอลของสัญญาณคลื่นซายน์เท่ากับ $\omega_1 = \frac{\pi}{3}$, $\mu = 0.005$, SNR = 8 dB, $\rho = 0.9$, N = 1500, $a(0)$ = 0 และ p =1, 2 และ 3 ผลการจำลองตรวจวัดความถี่ (ดูจากค่าสัมประสิทธิ์ "a") แสดงดังรูปที่ 7.2 จากรูป พบว่าเมื่อค่า p เพิ่มขึ้นจะทำให้อะแดปทีฟอัลกอริทึมทำงานได้เร็วขึ้น ซึ่งเป็นไปตามทฤษฎีดังที่ได้กล่าวไว้ในบทที่ 6



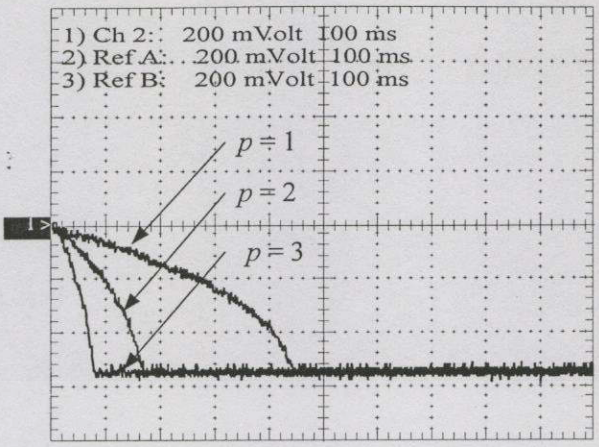
รูปที่ 7.2 การลู่เข้าของสัมประสิทธิ์ "a" สำหรับอะแดปทีฟอัลกอริทึมแบบ LMP เมื่อ p = 1, 2 และ 3

สำหรับการทดลองที่เวลาจริง จะทำการต่ออุปกรณ์ตามรูปที่ 7.3



รูปที่ 7.3 การต่ออุปกรณ์สำหรับการทดลองที่เวลาจริง

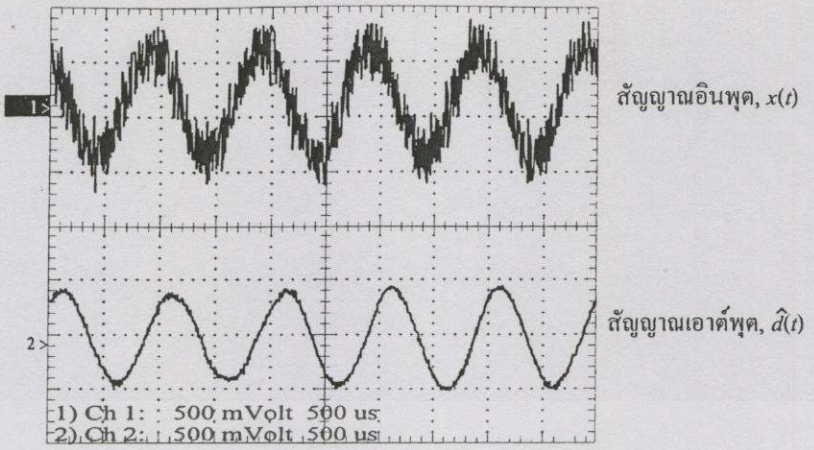
โดยกำหนดพารามิเตอร์ที่ใช้ในการทดลองใกล้เคียงกันกับการจำลองการทำงานด้วยโปรแกรมคอมพิวเตอร์ และกำหนดอัตราสุ่มเท่ากับ 8 kHz ทำการป้อนสัญญาณซายน์ความถี่ 1 kHz รวมกับสัญญาณรบกวนแบบเกาส์เซียน โดยจะปรับขนาดของสัญญาณอินพุตทั้งสองให้ได้ SNR ประมาณ 8 dB ผลการทดลอง โดยทำการส่งค่าสัมประสิทธิ์ "a" ออกทางเอาต์พุต แสดงดังรูปที่ 7.4



รูปที่ 7.4 การลู่เข้าของสัมประสิทธิ์ "a" สำหรับอะแดปทีฟอัลกอริทึม LMP เมื่อ p = 1, 2 และ 3 ทำงานที่เวลาจริง กับความถี่สัญญาณอินพุตเท่ากับ 1 kHz

จากรูปที่ 7.4 พบว่า การทำงานที่เวลาจริง จะมีความสอดคล้องกันกับการจำลองการทำงาน คือที่ ค่า p สูงขึ้น จะทำให้ อะแดปทีฟอัลกอริทึมทำงานได้เร็วเพิ่มขึ้น สังเกตรูปที่ 7.3 ที่ค่า p = 3 อะแดปทีฟอัลกอริทึมลู่เข้าที่ประมาณ n = 525 นั่นคือ เมื่อใช้ความถี่ในการสุ่มเท่ากับ 8 kHz จะได้ช่วงเวลาในการสุ่มเท่ากับ 125 ไมโครวินาที ดังนั้น อะแดปทีฟอัลกอริทึมจะใช้เวลาในการหาคำตอบทั้งสิ้นประมาณ 65.625 มิลลิวินาที ซึ่งจากการทดลองที่เวลาจริงตามรูปที่ 7.4 ที่ p = 3 จะใช้เวลาในการหาคำตอบประมาณ 70 มิลลิวินาที ซึ่งใกล้เคียงกัน ความผิดพลาดที่เกิดขึ้นเนื่องจากใช้ระบบตัวเลขในการประมวลผลต่างกัน โดยโปรแกรมคอมพิวเตอร์จะคำนวณโดยใช้ตัวเลขอิงดัชนี (floating point number) ขนาด 64 บิต แต่ไมโครโปรเซสเซอร์จะคำนวณโดยใช้ระบบเลขจำนวนเต็ม (fixed point number) จึงทำให้ผลที่ได้มีความแตกต่างกัน เล็กน้อย แต่อย่างไรก็ตามผลการทดลองทั้งสองก็ยังคงไปในทิศทางเดียวกัน โดยเมื่อเพิ่มค่า p จะทำให้ความเร็วในการหาคำตอบเพิ่มขึ้นแต่ในขณะเดียวกันก็จะต้องเพิ่มการคำนวณด้วย

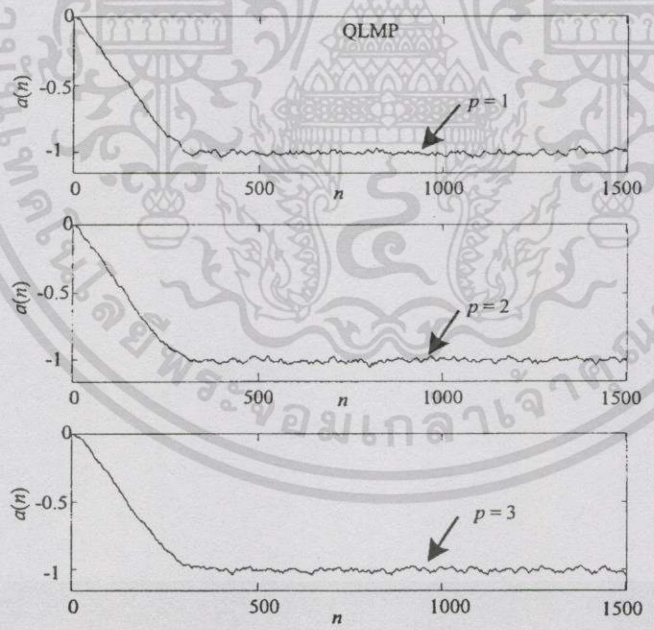
สำหรับการทดลองในรูปที่ 7.5 (บน) เป็นสัญญาณอินพุต $x(t)$ ความถี่ 1 kHz ถูกรบกวนด้วยสัญญาณรบกวนแบบเกาส์เซียน มี SNR ประมาณ 8 dB ส่วนรูปที่ 7.5 (ล่าง) แสดงสัญญาณเอาต์พุตของ BPF คือ $\hat{d}(t)$ จะเห็นว่า ความถี่ที่ประมาณได้ มีค่าใกล้เคียงกับสัญญาณทางอินพุต และสัญญาณรบกวนถูกกำจัดทิ้งไป นั่นคือ ANF สามารถตรวจวัดความถี่ได้สำเร็จนั่นเอง



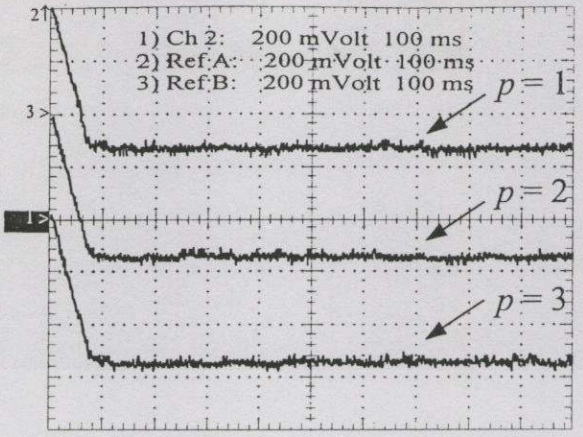
รูปที่ 7.5 สัญญาณอินพุต (บน) และสัญญาณที่ประมาณได้ (ล่าง)

7.1.2.2 ผลการทำงานของอะแดปทีฟอัลกอริทึมแบบ QLMP

กำหนดพารามิเตอร์ที่ใช้ในการทดลองเหมือนในหัวข้อที่ 7.1.2.2 ผลการทดลอง แสดงดังรูปที่ 7.6 (จำลองการทำงาน) และรูปที่ 7.7 (ที่เวลาจริง)



รูปที่ 7.6 การลู่เข้าของสัมประสิทธิ์ "a" สำหรับอะแดปทีฟอัลกอริทึม QLMP เมื่อ $p = 1, 2$ และ 3 ตามลำดับ

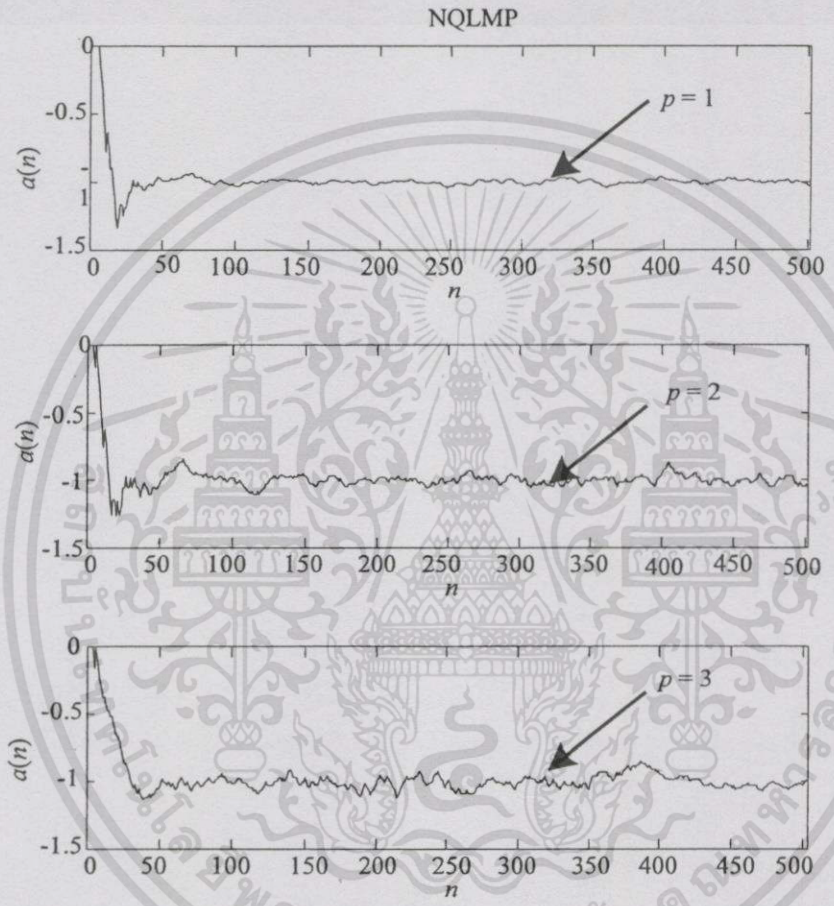


รูปที่ 7.7 การลู่เข้าของสัมประสิทธิ์ "a" สำหรับอะแดปทีฟอัลกอริทึม QLMP เมื่อ p = 1, 2 และ 3 ตามลำดับ ทำงานที่เวลาจริง กับความถี่สัญญาณอินพุตเท่ากับ 1 kHz

จากรูปที่ 7.6 และรูปที่ 7.7 ยืนยันได้ว่า ความเร็วในการหาคำตอบของอะแดปทีฟอัลกอริทึมแบบ QLMP นั้นไม่ขึ้นกับค่า p นอกจากนี้ยังมีการลู่เข้าสู่คำตอบที่เร็วกว่า LMP แม้จะใช้ค่า p = 1 ซึ่งในทางปฏิบัติ ก็ควรจะเลือกค่า p = 1 มาใช้งาน เพราะสามารถลดการคำนวณลงได้อีกด้วย

7.1.2.3 ผลการทำงานของอะแดปทีฟอัลกอริทึมแบบ NQLMP และ VSQLMP

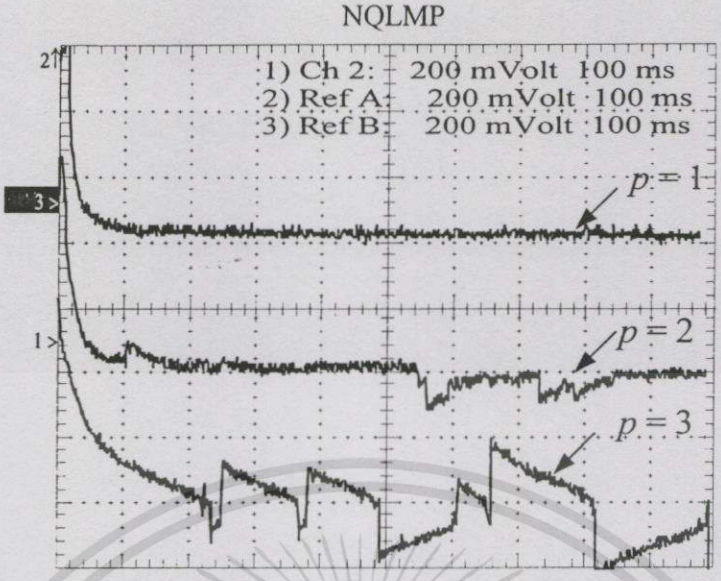
ในหัวข้อนี้จะทำการทดลองการทำงานของอะแดปทีฟอัลกอริทึมแบบ NQLMP และ VSQLMP โดยกำหนดค่าพารามิเตอร์ให้เหมือนกับการทดลองในหัวข้อที่ผ่านมา แต่จะมีพารามิเตอร์ที่เพิ่มเข้ามา คือ $\alpha = 0.98$ (forgetting factor), $\gamma = 0.97, \lambda = 0.001, \mu_{\max}(0) = 0.07$ ผลการจำลองการทำงานของอะแดปทีฟอัลกอริทึมแบบ NQLMP แสดงดังรูปที่ 7.8



รูปที่ 7.8 การลู่เข้าของสัมประสิทธิ์ "a" สำหรับอะแดปทีฟอัลกอริทึม NQLMP เมื่อ p = 1 (บน), p = 2 (กลาง) และ p = 3 (ล่าง)

จากรูปที่ 7.8 พบว่า ที่ค่า p มีค่าสูง อะแดปทีฟอัลกอริทึมแบบ NQLMP จะทำงานได้แย่ลงซึ่งสังเกตได้จากสัมประสิทธิ์ "a" ที่ประมาณได้ จะมีการแกว่งไปมา รอบค่า –1 มากกว่า ที่ p มีค่าน้อย และความเร็วในการทำงาน จะไม่ขึ้นกับค่า p ด้วย ทั้งนี้เพราะอะแดปทีฟอัลกอริทึมนี้ได้พัฒนามาจากอะแดปทีฟอัลกอริทึมแบบ QLMP แต่สามารถทำงานได้เร็วกว่า ดังนั้นหากจะนำอะแดปทีฟอัลกอริทึมนี้ไปใช้ ควรจะเลือก p = 1 จะเหมาะสมที่สุด

สำหรับการทดลองที่เวลาจริง แสดงดังรูปที่ 7.9

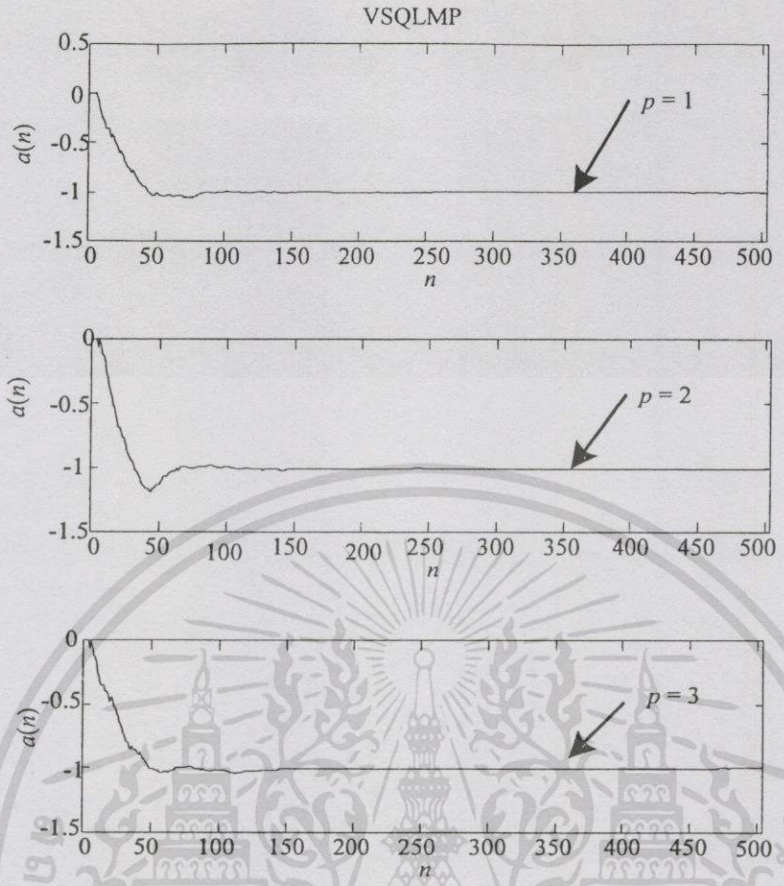


รูปที่ 7.9 การลู่เข้าของสัมประสิทธิ์ "a" สำหรับอะแดปทีฟอัลกอริทึม NQLMP เมื่อ $p = 1, 2$ และ 3 ทำงานที่เวลาจริง กับความถี่สัญญาณอินพุตเท่ากับ 1 kHz

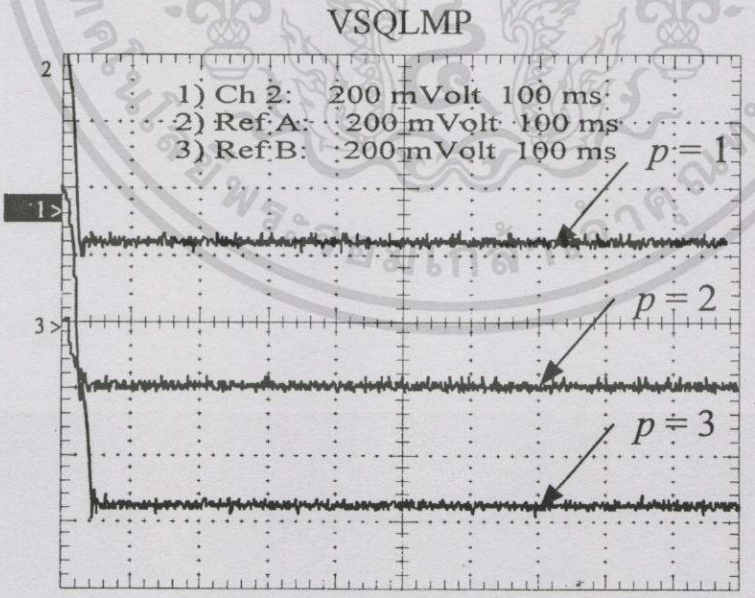
จากผลการทดลองตามรูปที่ 7.9 ยืนยันได้ว่า เมื่อค่า p สูงขึ้น จะทำให้อะแดปทีฟอัลกอริทึมแบบ NQLMP ทำงานได้แย่ลง ซึ่งสอดคล้องกับการจำลองการทำงานตามรูปที่ 7.8

สำหรับผลการจำลองการทำงานของอะแดปทีฟอัลกอริทึมแบบ VSQLMP แสดงดังรูปที่ 7.10 จากรูปพบว่าอะแดปทีฟอัลกอริทึมแบบ VSQLMP จะสามารถทำงานได้เร็ว และคำตอบที่ประมาณได้นั้น น่าจะมีความถูกต้องมากกว่าแบบ QLMP และ NQLMP โดยดูจากการแกว่งของค่าสัมประสิทธิ์ "a" ซึ่งเกือบเป็นเส้นตรง (การวิเคราะห์คำตอบจะกล่าวในหัวข้อถัดไป) นอกจากนี้ ค่า p ที่เปลี่ยนไป จะไม่ส่งผลการทบต่อการทำงานของอะแดปทีฟอัลกอริทึมอีกด้วย ดังนั้น ในการใช้งานจริง ควรจะเลือก $p = 1$ เช่นเดียวกัน

สำหรับผลการทดลองที่เวลาจริง แสดงดังรูปที่ 7.11 โดยผลการทดลองที่ได้ เหมือนกับที่ได้จากการจำลองการทำงาน



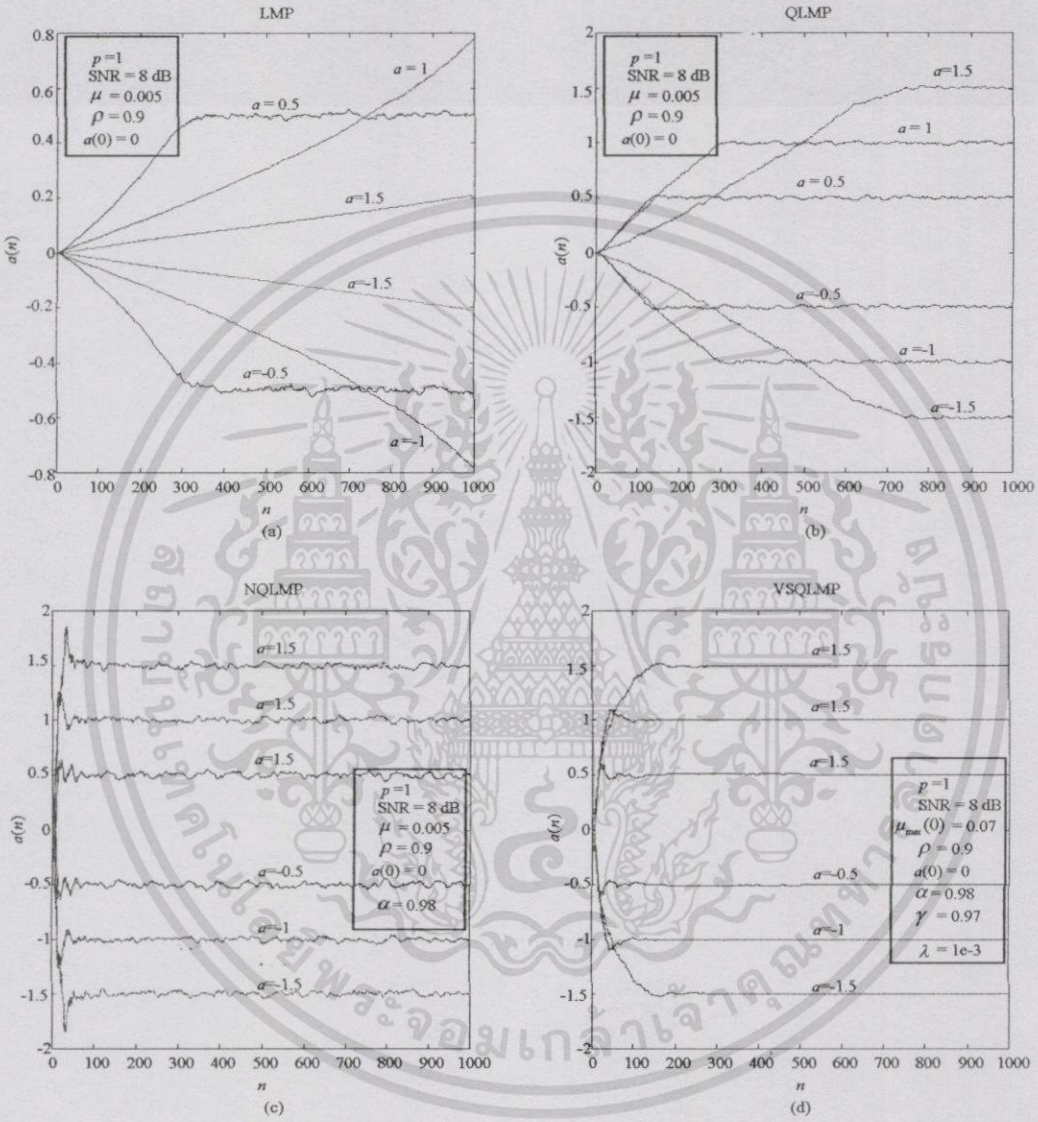
รูปที่ 7.10 การลู่เข้าของสัมประสิทธิ์ a สำหรับอะแดปทีฟอัลกอริทึม VSQLMP เมื่อ $p = 1$ (บน), $p = 2$ (กลาง) และ $p = 3$ (ล่าง)



รูปที่ 7.11 การลู่เข้าของสัมประสิทธิ์ "a" สำหรับอะแดปทีฟอัลกอริทึม VSQLMP เมื่อ $p = 1, 2$ และ 3 ทำงานที่เวลาจริง กับความถี่สัญญาณอินพุตเท่ากับ 1 kHz

จากรูปที่ 7.8 ถึงรูปที่ 7.11 พบว่า อะแดปทีฟอัลกอริทึมแบบ NQLMP และ VSQLMP มีความสามารถในการหาคำตอบได้เร็วกว่าแบบ LMP และ QLMP

ผลการจำลองการทำงานในรูปที่ 7.12 เป็นผลการทำงานที่ความถี่ต่าง ๆ ตลอดย่านความถี่ในควิส ($0-\pi$) ของอะแดปทีฟอัลกอริทึมทั้งสี่แบบที่ค่า $p=1$ เพียงค่าเดียว



รูปที่ 7.12 เปรียบเทียบผลการจำลองการประมาณค่าความถี่สัญญาณซายน์ย่านความถี่ในควิสในสัญญาณรบกวนแบบเกาส์เซียน (a) LMP (b) QLMP (c) NQLMP และ (d) VSQLMP

รูป (a) เป็นผลของ LMP พบว่า ถ้ากำหนดค่าเริ่มต้นให้กับสัมประสิทธิ์ "a" มีค่า ห่างจากคำตอบที่แท้จริง จะทำให้ อะแดปทีฟอัลกอริทึมทำงานได้ช้ามาก (จากรูปคือ ที่ a = 1.5 และ –1.5) สำหรับรูป (b) เป็นผลของ QLMP พบว่า ทำงานได้เร็วกว่า LMP ส่วนในรูป (c) และ รูป (d) เป็น

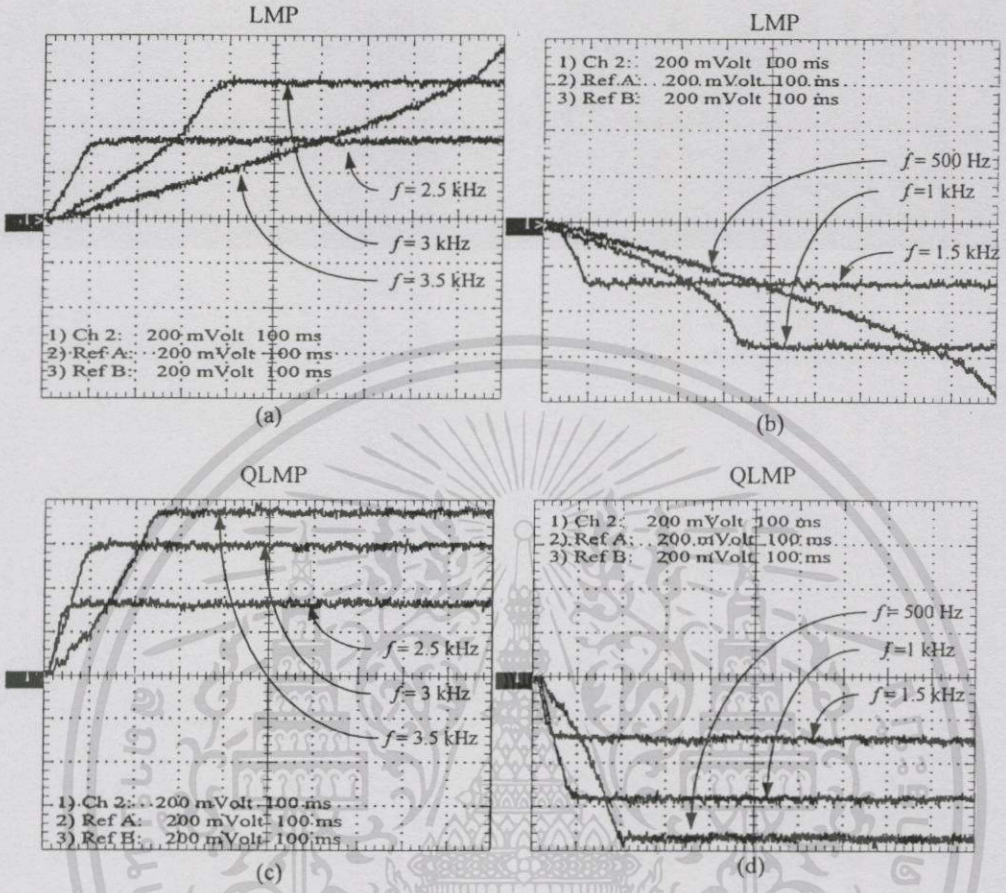
ของ NQLMP และ VSQLMP พบว่า ทำงานได้เร็วพอ ๆ กัน และเร็วกว่า LMP และ QLMP โดยผลของ VSQLMP จะให้คำตอบที่เรียบ (smooth) กว่า นั่นคือ ให้คุณสมบัติทางสถิติที่ดีกว่านั่นเอง

สำหรับความสัมพันธ์ระหว่างค่าสัมประสิทธิ์ "a", ความถี่ดิจิตอล และความถี่แอนะลอก เมื่อใช้อัตราสุ่ม 8 kHz แสดงดังตารางที่ 7.1 คือ

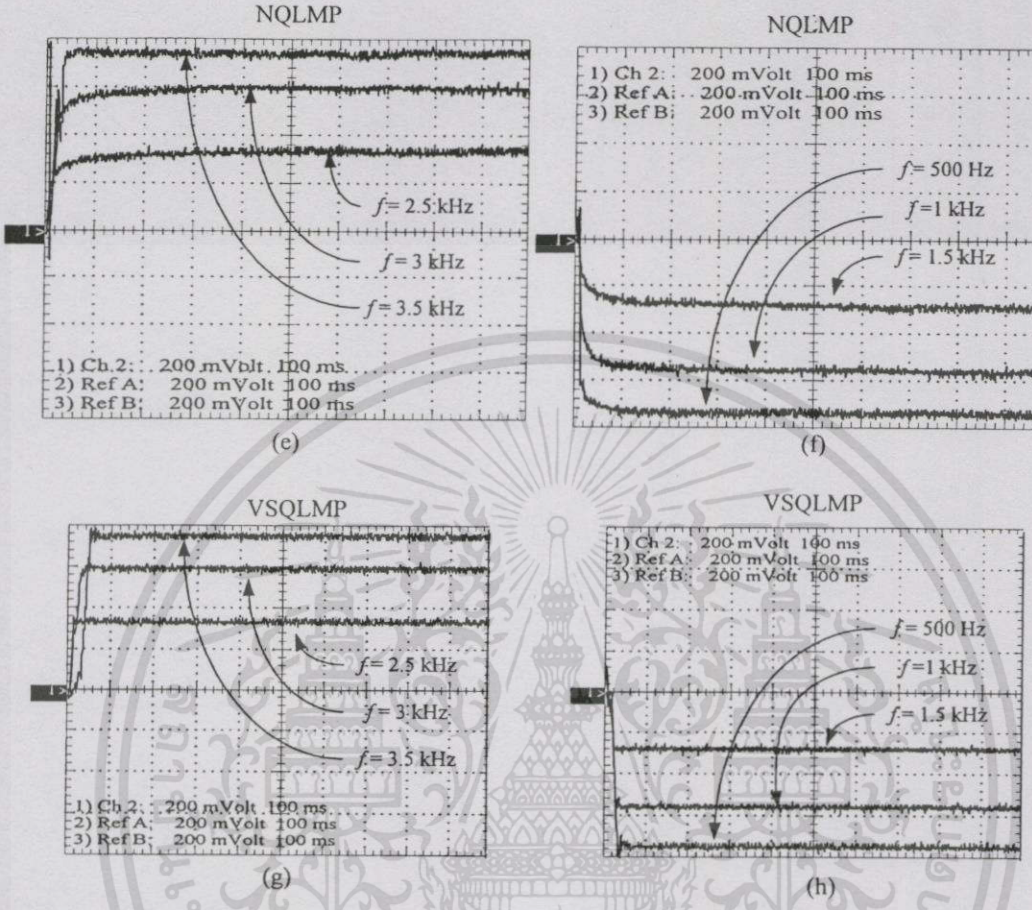
ตารางที่ 7.1 ความสัมพันธ์ของตัวแปรต่าง ๆ เมื่อใช้อัตราสุ่มเท่ากับ 8 kHz

สัมประสิทธิ์ "a"	ความถี่ดิจิตอล "ω_1"	ความถี่แอนะลอก "f_a"
-1.5	0.23π	500 Hz
-1	$\pi/3$	1 kHz
-0.5	0.42π	1.5 kHz
0.5	0.58π	2.5 kHz
1	$2\pi/3$	3 kHz
1.5	0.77π	3.5 kHz

ผลการทดลองที่เวลาจริงแสดงดังรูปที่ 7.13(a) ถึง รูปที่ 7.13(h)



รูปที่ 7.13 เปรียบเทียบการทำงานที่เวลาจริงในย่านความถี่ในควิส รูป (a)-(b) LMP และ รูป (c)-(d) QLMP



รูปที่ 7.13 (ต่อ) เปรียบเทียบการทำงานที่เวลาจริงในย่านความถี่ไนควิส รูป (e)-(f) NQLMP และ รูป (g)-(h) VSQLMP

เปรียบเทียบผลการทดลองในรูปที่ 7.12 กับรูปที่ 7.13 จะเห็นว่า สอดคล้องกัน นั่นหมายความว่า อะแดปทีฟอัลกอริทึมที่นำเสนอในวิทยานิพนธ์ สามารถนำไปใช้จริงได้ในทางปฏิบัติ

ความถี่ที่ได้จากการตรวจวัดนี้ คือความถี่ที่ถูกประมาณขึ้นด้วยกระบวนการทางอะแดปทีฟ ซึ่งในทางทฤษฎีกล่าวไว้ว่า ตัวประมาณค่า (estimator) ที่ดีนั้นจะต้องมีคุณสมบัติของคำตอบ สองประการคือ 1) เป็น Unbiased และ 2) Variance ต่ำ แต่เนื่องจากจะไม่สามารถวัดหาค่า Bias และ Variance ของสัมประสิทธิ์ได้ ดังนั้น การจะหาค่าทั้งสองนี้จะต้องใช้การวิเคราะห์ทางทฤษฎี หรือใช้คอมพิวเตอร์ช่วยในการคำนวณที่เรียกว่า Monte Carlo simulation [34] ซึ่งในหัวข้อต่อไปจะแสดงผลของค่าดังกล่าวแบบ Monte Carlo simulation

7.1.2.4 ค่า Bias และ Variance ของสัมประสิทธิ์ "a" โดยใช้วิธี Mote Carlo simulation

โดยการใช้วิธี Mote Carlo simulation เพื่อหาค่า variance และ bias อย่างประมาณของสัมประสิทธิ์ "a" นั้น ทำได้โดยกำหนดจำนวนความยาวของข้อมูลของแต่ละอัลกอริทึมมีค่า N = 2000 ทำการทดลองซ้ำเป็นจำนวน M = 100 ครั้ง จากนั้น จะนำค่าที่ได้ในแต่ละครั้ง มาหาค่าเฉลี่ยแบบแอนแซมเบิล (ensemble average) สำหรับพารามิเตอร์อื่น ๆ จะเหมือนกับการทดลองในหัวข้อที่ผ่านมา แต่จะทำการปรับเปลี่ยนค่า SNR ตั้งแต่ –2 dB จนถึง 20 dB

สำหรับสมการค่า mean ค่า variance และ ค่า bias ของสัมประสิทธิ์ "a" กำหนดได้ ตามลำดับดังนี้ คือ

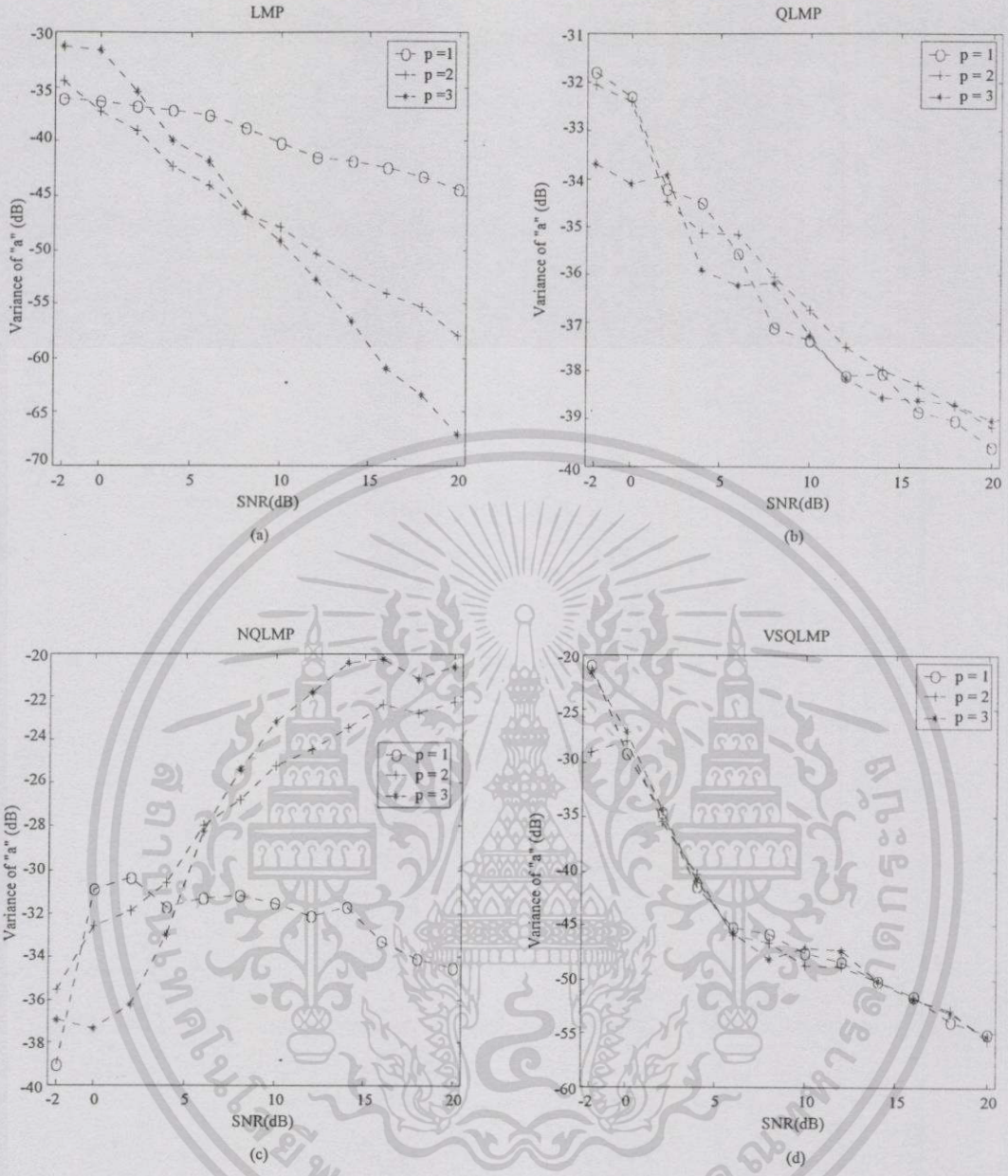
$$\hat{E}\{a\} = \frac{1}{M}\sum_{i=1}^{M} a_i(n) \quad ;n = 0,1,2...,N \tag{7.2}$$

$$\text{var}(a) = \frac{1}{M}\sum_{i=1}^{M}\left(a_i(n) - \hat{E}\{a\}\right)^2 \quad ;n = 0,1,2...,N \tag{7.3}$$

$$\text{bias}(a) = a_o - \hat{E}\{a\} \tag{7.4}$$

โดยที่ "a_o" คือค่าสัมประสิทธิ์ที่แท้จริง

ผลการทดลองในรูปที่ 7.14 แสดงการเปรียบเทียบค่า variance ของ สัมประสิทธิ์ "a" ของอะแดปทีฟอัลกอริทึมแต่ละแบบ ที่ p = 1, 2 และ 3



รูปที่ 7.14 ค่า variance ของสัมประสิทธิ์ "a" ที่ p =1, 2 และ 3 (a) LMP (b) QLMP (c) NQLMP และ (d) VSQLMP

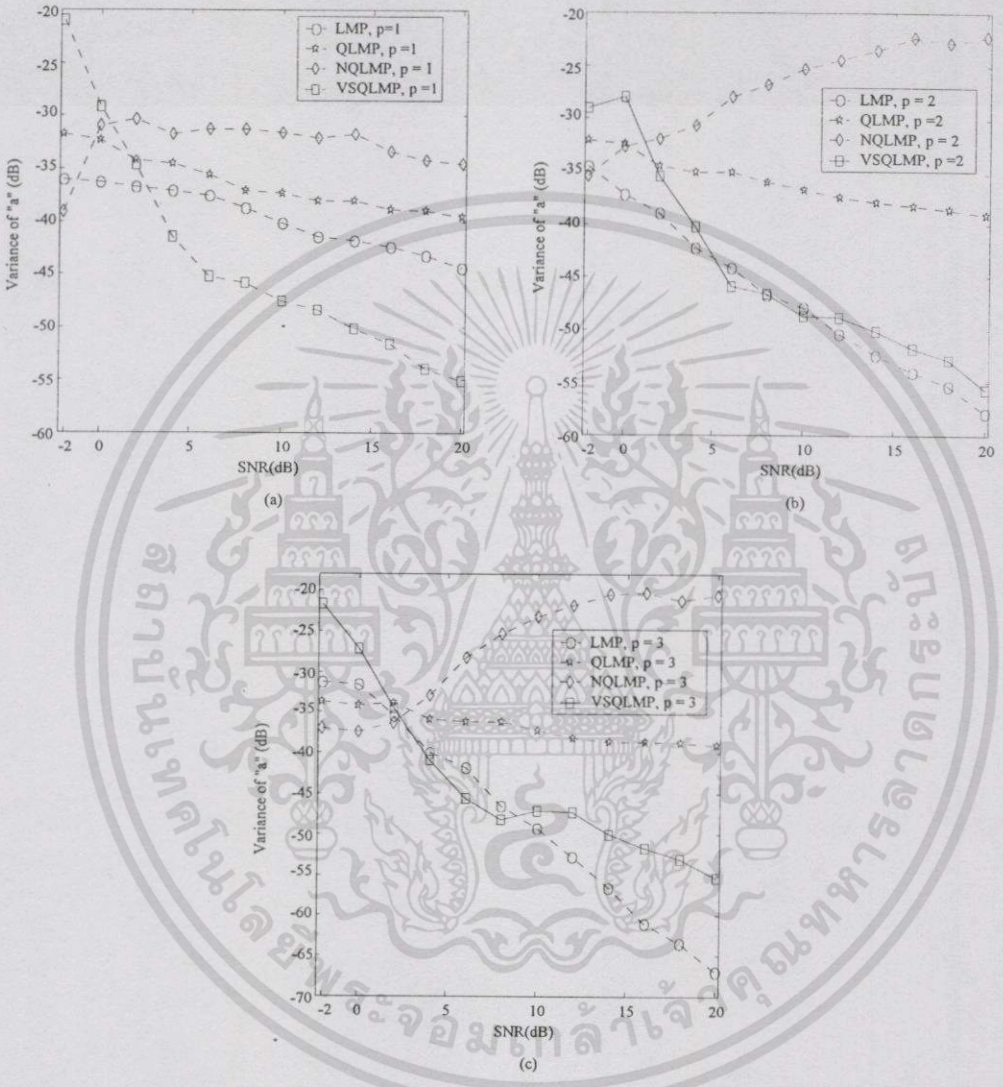
จากรูปที่ 7.14 (a) เป็นผลของ LMP พบว่า เมื่อค่า p สูงขึ้น และ ค่า SNR เพิ่มขึ้น จะทำให้ค่า variance ของสัมประสิทธิ์ลดลองกว่าเมื่อใช้ ค่า p น้อย

จากรูปที่ 7.14 (b) เป็นผลของ QLMP พบว่า ค่า variance มีค่าใกล้เคียงกันทุกค่าของ p และลดลองเมื่อค่า SNR เพิ่มขึ้น

จากรูปที่ 7.14 (c) เป็นผลของ NQLMP พบว่า ค่า variance จะเพิ่มสูงขึ้นเมื่อค่า p และ ค่า SNR มีค่าเพิ่มขึ้น ส่วนที่ค่า p มีค่าต่ำ ค่า variance มีแนวโน้มลดลงเมื่อ SNR เพิ่มขึ้น

และจากรูปที่ 7.14 (d) เป็นผลของ VSQLMP พบว่าที่ค่า SNR น้อย ค่า variance จะมีค่าสูง และที่ ค่า SNR สูง ค่า variance จะลดลงและ เกือบคงที่ที่ทุก ๆ ค่า p

เพื่อให้เปรียบเทียบให้ชัดเจนยิ่งขึ้น จะทำการเปรียบเทียบ ที่ค่า p เดียวกัน แสดงดังรูปที่ 7.15

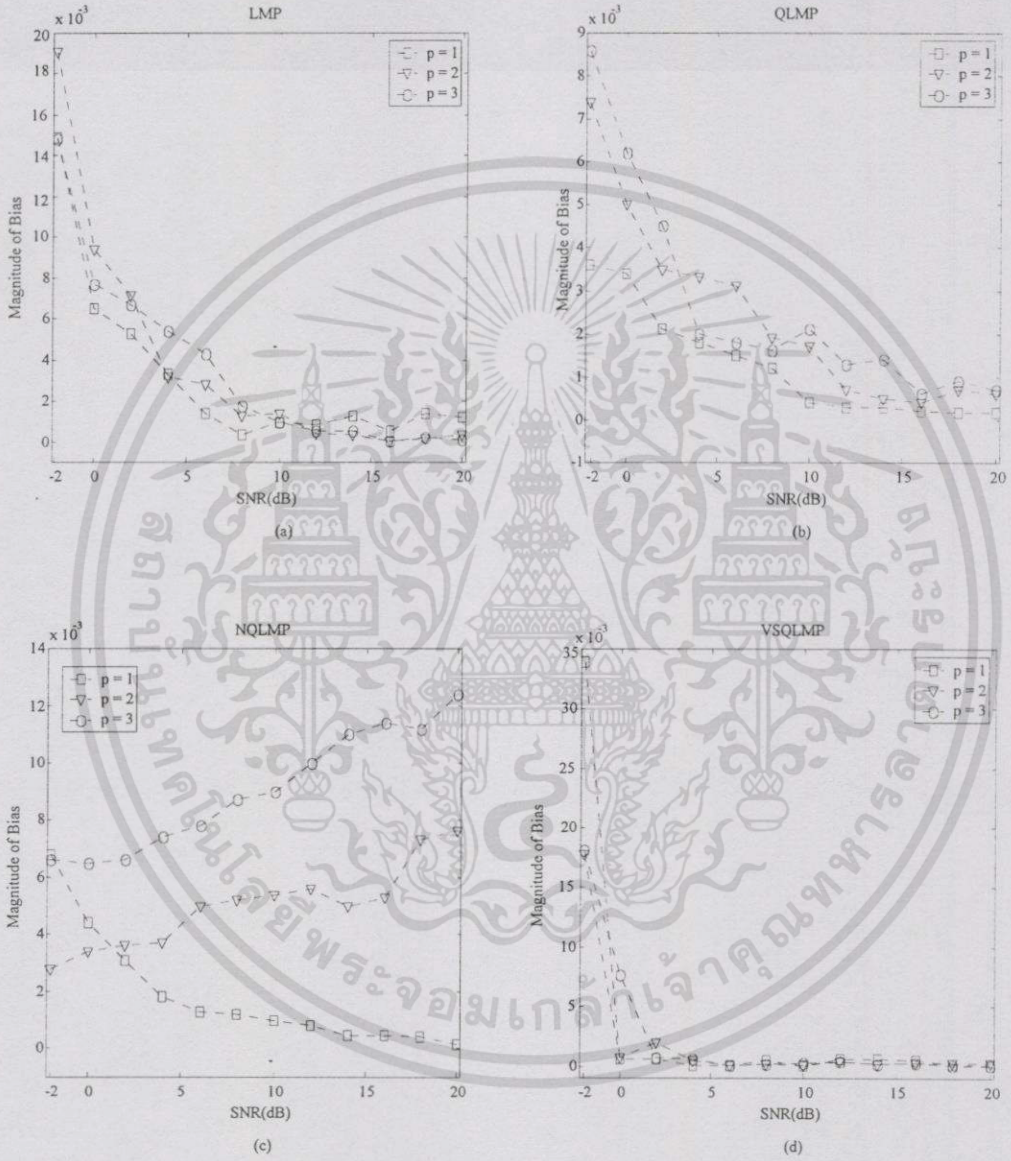


รูปที่ 7.15 เปรียบเทียบค่า variance ของ สัมประสิทธิ์ "a" (a) เมื่อ $p = 1$ (b) เมื่อ $p = 2$ และ (c) เมื่อ $p = 3$

จากรูปที่ 7.15 (a) เป็นผลค่า variance เมื่อ $p = 1$ พบว่า ที่ SNR ต่ำ VSQLMP จะให้ค่า variance สูงสุด แต่ที่ SNR สูงขึ้น VSQLMP จะให้ค่า variance ต่ำที่สุด นั่นคือ ที่ SNR สูง ๆ VSQLMP จะให้คุณสมบัติทางสถิติดีที่สุด

จากรูปที่ 7.15 (b) เป็นผลค่า variance เมื่อ $p = 2$ พบว่า NQLMP จะให้ค่า variance สูงขึ้น เมื่อ SNR มีค่าสูงขึ้น ส่วนค่า variance ของ LMP และ VSQLMP จะมีค่าใกล้เคียงกัน ส่วนรูปที่ 7.15 (c) จะคล้ายกับรูปที่ 7.15 (b)

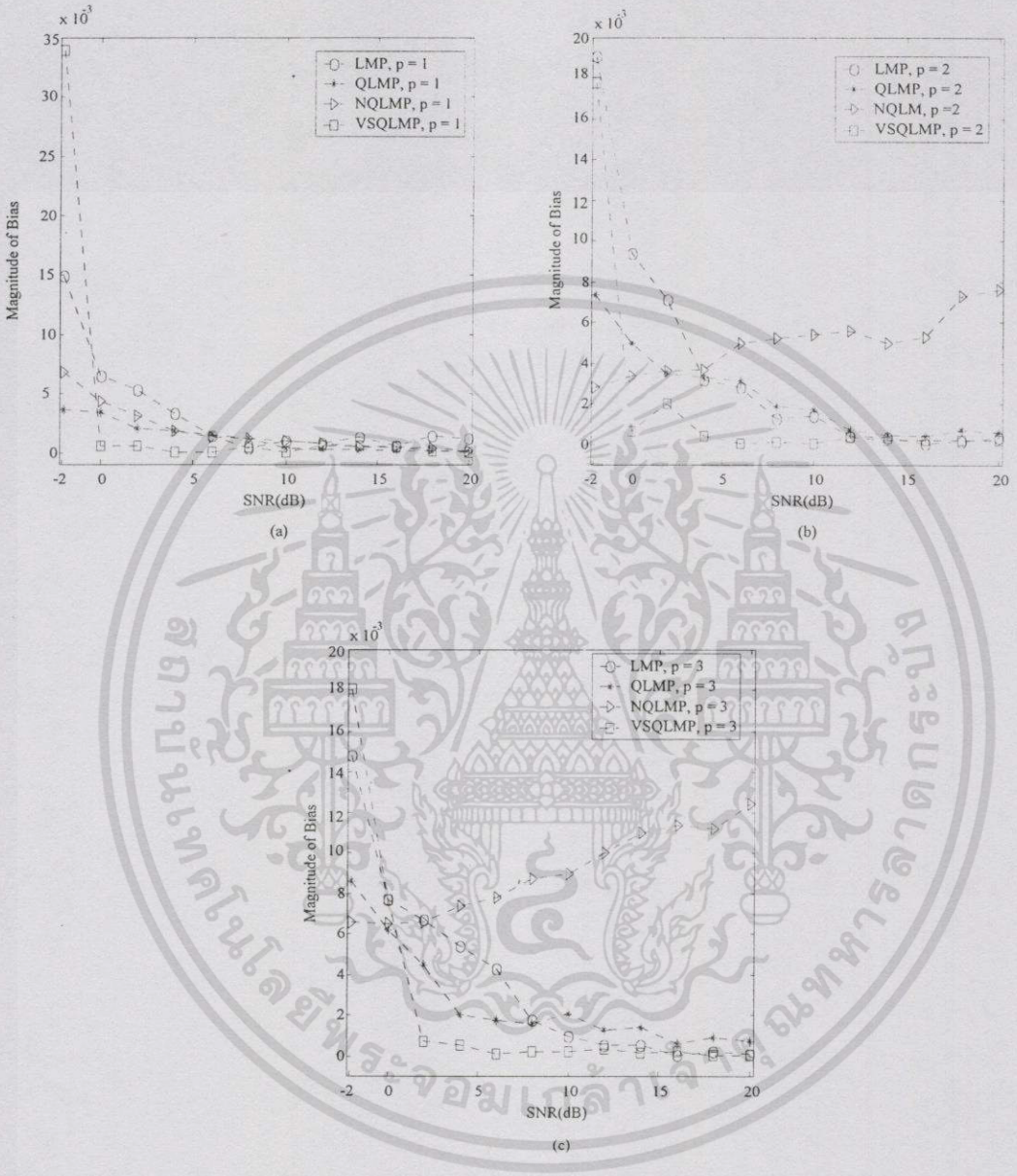
ผลการทดลองในรูปที่ 7.16 แสดงการเปรียบเทียบค่า bias ของอะแดปทีฟอัลกอริทึมแบบต่าง ๆ ที่ $p = 1, 2$ และ 3



รูปที่ 7.16 เปรียบเทียบค่า bias ที่ $p = 1, 2$ และ 3(a) LMP (b) QLMP (c) NQLMP และ (d)VSQLMP

จากรูปที่ 7.16 พบว่า อะแดปทีฟอัลกอริทึมแบบ NQLMP ที่ค่า p และ SNR เพิ่มขึ้น ทำให้ bias เพิ่มมากขึ้น ส่วนอะแดปทีฟอัลกอริทึมแบบอื่น จะมีค่า bias เกือบเท่ากันทุกค่า p และจะลดลง เมื่อ ค่า SNR เพิ่มขึ้น

เพื่อความชัดเจน จะแยกเปรียบเทียบ ตามค่า p ดังแสดงตามรูปที่ 7.17



รูปที่ 7.17 เปรียบเทียบค่า bias แยกตามค่า p, (a) เมื่อ $p = 1$ (b) เมื่อ $p = 2$ และ (c) เมื่อ $p = 3$

จากรูปที่ 7.17 จะเห็นได้ชัดเจนว่า ที่ SNR ต่ำ อะแดปทีฟอัลกอริทึมแบบ VSQLMP จะทำงานได้ไม่ดี และเมื่อ SRN มีค่าสูงระดับหนึ่ง (จากรูปคือที่ ประมาณ 0 dB) จะทำให้ VSQLMP ทำงานได้ดีขึ้นจนน่าพอใจ

จากการทดลองในหัวข้อนี้ พอจะสามารถสรุปได้ดังนี้

- อะแดปทีฟอัลกอริทึม แบบ LMP จะให้คุณสมบัติ ทางสถิติที่ดีขึ้น เมื่อ ค่า SNR และ ค่า p มีค่าสูง
- อะแดปทีฟอัลกอริทึม แบบ QLMP จะให้คุณสมบัติ ทางสถิติที่ดีขึ้น เมื่อ ค่า SNR มีค่าสูง และจะไม่ขึ้นกับค่า p แต่ จะแย่กว่า LMP แต่มีข้อดีที่ จะทำงานได้เร็ว
- อะแดปทีฟอัลกอริทึม แบบ NQLMP จะให้คุณสมบัติ ทางสถิติที่แย่ลง เมื่อ ค่า SNR และ ค่า p มีค่าสูง แต่มีข้อดีคือ ทำงานได้เร็วกว่า LMP และ QLMP
- อะแดปทีฟอัลกอริทึม แบบ VSQLMP จะให้คุณสมบัติ ทางสถิติที่ดีขึ้น เมื่อ ค่า SNR มีค่าสูง และจะไม่ขึ้นกับค่า p ส่วนที่ค่า SNR ต่ำ (ต่ำกว่า 0 dB) จะทำงานได้ไม่ดี อะแดปทีฟอัลกอริทึม แบบ VSQLMP จะทำงานได้เร็วพอ ๆ กับ NQLMP แต่จะให้คุณสมบัติทางสถิติที่ดีกว่ามาก

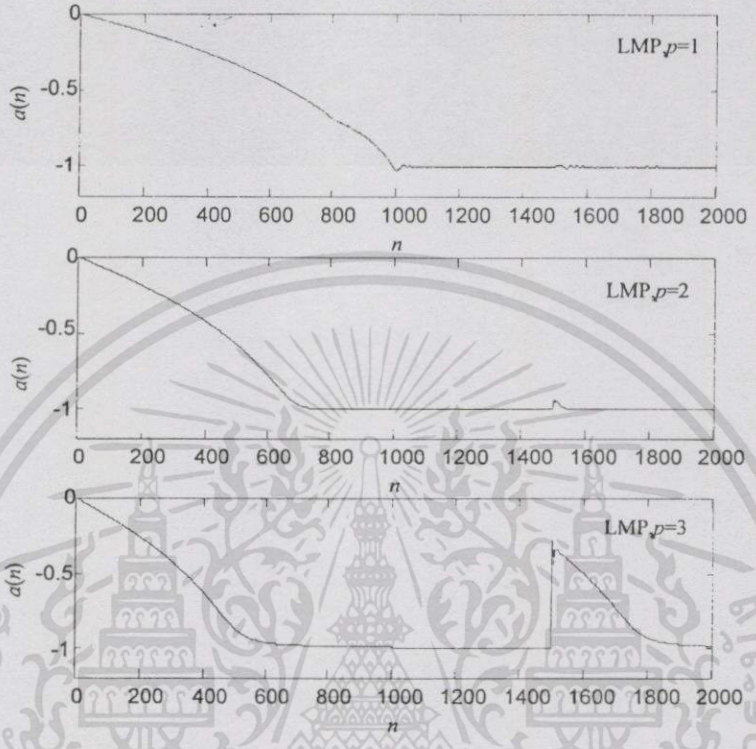
7.1.2.5 ผลการตรวจวัดความถี่สัญญาณซายน์ในสภาวะที่ถูกรบกวนด้วยสัญญาณรบกวนแบบแบบอิมพัลส์

สัญญาณรบกวนอิมพัลส์เป็นสัญญาณรบกวนที่มีขนาดสูงเกิดขึ้นในช่วงเวลาสั้น ๆ เช่นเกิดจากการจุดระเบิดของหัวเทียนรถยนต์ ฟ้าผ่า หรือการสปาร์คของปลั๊กไฟฟ้า เป็นต้น ซึ่งสัญญาณรบกวนนี้จะมีผลกระทบต่อการทำงานของอะแดปทีฟอัลกอริทึมด้วย ดั้งนั้นในหัวข้อนี้จะได้ทำการทดลองเพื่อดูถึงผลกระทบจากสัญญาณรบกวนดังกล่าว

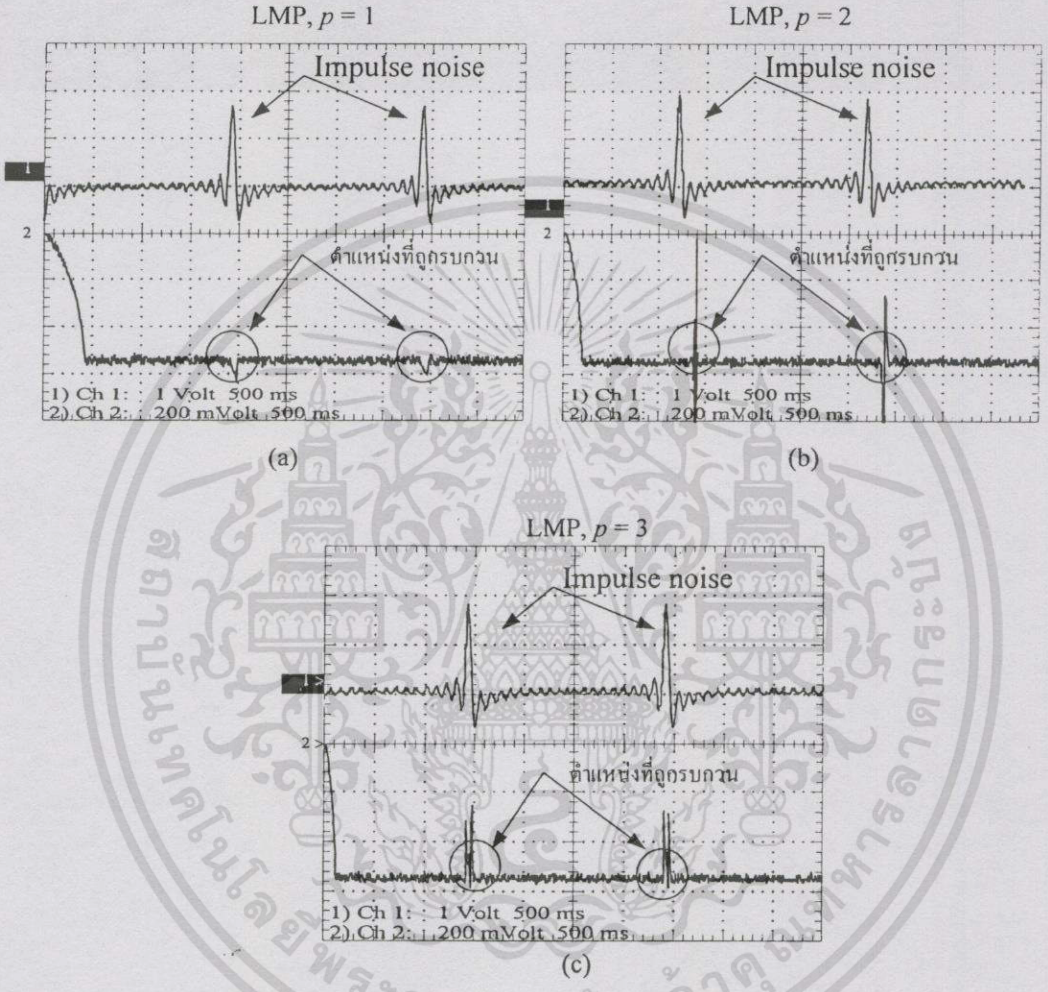
พารามิเตอร์ที่ใช้ในการทดลองจะเหมือนกับหัวข้อที่ผ่านมาโดยในการจำลองการทำงานด้วยโปรแกรมคอมพิวเตอร์นั้น จะทำงานบวกสัญญาณรบกวนแบบอิมพัลส์ที่ตำแหน่งมีการลู่เข้าสู่คำตอบแล้ว ส่วนการทดสอบที่เวลาจริงจะทำการบวกสัญญาณอิมพัลส์ที่มีลักษณะเป็นรายคาบไปตลอดการทำงาน

1) ผลของสัญญาณรบกวนอิมพัลส์ต่ออะแดปทีฟอัลกอริทึมแบบ LMP

อะแดปทีฟอัลกอริทึมแบบ LMP มีตัวแปร คือ p ดังนั้นในการทดลองจะได้ทดลองเปลี่ยนแปลงค่า p และทำการบวกสัญญาณอิมพัลส์ขนาด 10 หน่วยตรงตำแหน่งที่ 1500 รูปที่ 7.18 แสดงผลกระทบของสัญญาณรบกวนอิมพัลส์ต่อการทำงานของอะแดปทีฟอัลกอริทึมพบว่าที่ค่า p มีค่าสูง สัญญาณรบกวนอิมพัลส์จะมีผลกระทบมากเทียบกับค่า p ต่ำ แต่ค่า p ต่ำจะมีการลู่เข้าสู่คำตอบช้า สำหรับรูปที่ 7.19 เป็นการทดลองที่เวลาจริง จากรูปที่ 7.18 และรูปที่ 7.19 พบว่ามีความสอดคล้องกันสรุปได้ว่า ในสภาวะที่มีสัญญาณรบกวนแบบอิมพัลส์เกิดขึ้นบ่อยไม่ควรเลือกใช้ค่า p สูง ๆ เพราะอาจทำให้อะแดปทีฟอัลกอริทึมแบบ LMP หยุดทำงานได้



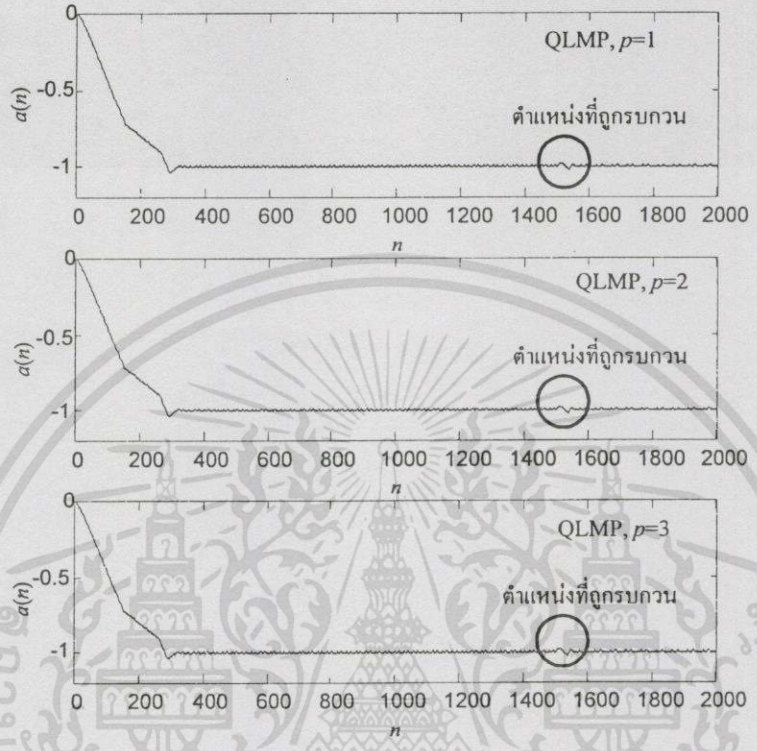
รูปที่ 7.18 ผลกระทบของสัญญาณรบกวนอิมพัลส์ต่ออะแดปทีฟอัลกอริทึมแบบ LMP ที่ได้จากการจำลองการทำงานเมื่อค่า p =1, 2 และ 3 ตามลำดับ



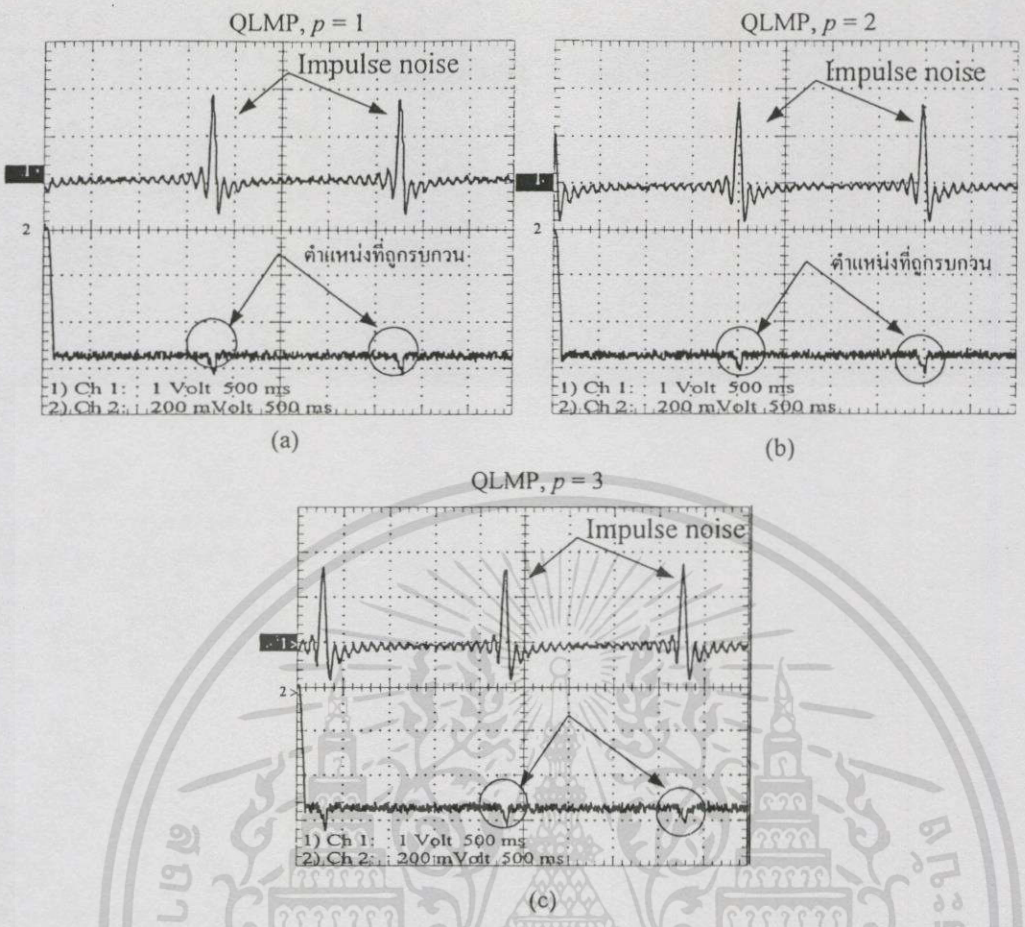
รูปที่ 7.19 ผลกระทบของสัญญาณรบกวนอิมพัลส์ที่เวลาจริง ต่ออะแดปทีฟอัลกอริทึมแบบ LMP, (a) p =1, (b) p = 2 และ (c) p = 3

2) ผลของสัญญาณรบกวนอิมพัลส์ต่ออะแดปทีฟอัลกอริทึมแบบ QLMP

โดยการกำหนดพารามิเตอร์ให้เหมือนกับการทดลองที่ผ่านมาจะได้ผลการทดลองตามรูปที่ 7.20 และ 7.21



รูปที่ 7.20 ผลกระทบของสัญญาณรบกวนอิมพัลส์ต่ออะแดปทีฟอัลกอริทึมแบบ QLMP ที่ได้จากการจำลองการทำงาน เมื่อค่า p =1, 2 และ 3 ตามลำดับ

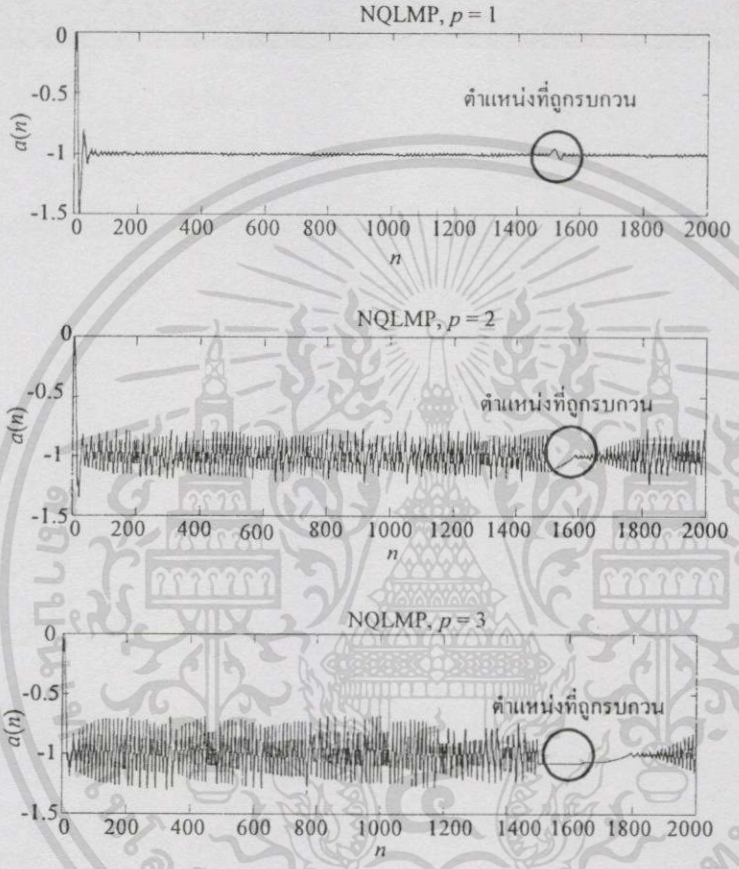


รูปที่ 7.21 ผลกระทบของสัญญาณรบกวนอิมพัลส์ที่เวลาจริง ต่ออะแดปทีฟอัลกอริทึมแบบ QLMP, (a) p =1, (b) p = 2 และ (c) p = 3

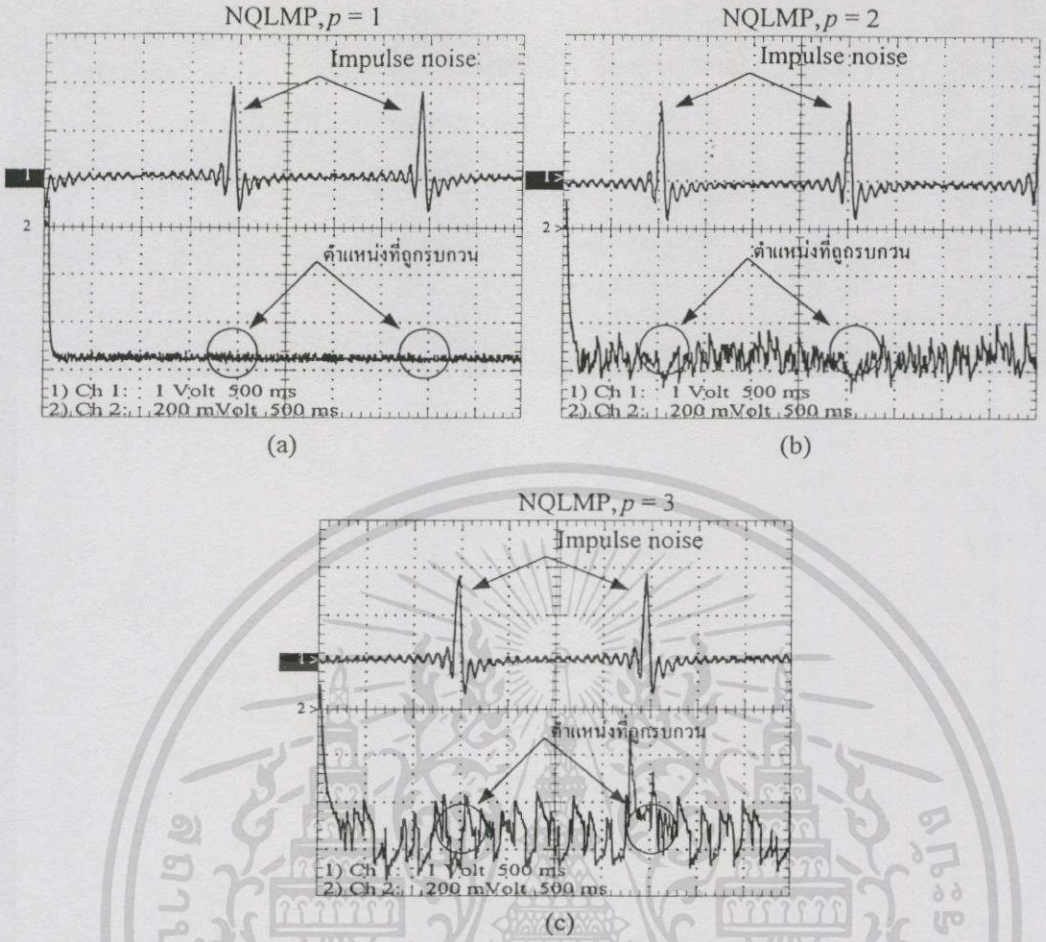
จากรูปที่ 7.20 และ 7.21 พบว่า อะแดปทีฟอัลกอริทึมแบบ QLMP จะสามารถทนต่อสัญญาณรบกวนอิมพัลส์ได้ดีกว่า LMP โดยค่า p ที่เพิ่มขึ้น จะไม่ส่งผลให้ การทำงานแย่ลง

3) ผลของสัญญาณรบกวนอิมพัลส์ต่ออะแดปทีฟอัลกอริทึมแบบ NQLMP และ แบบ VSQLMP

ในหัวข้อนี้จะทดสอบการทำงานของอะแดปทีฟอัลกอริทึมแบบ NQLMP และ แบบ VSQLMP โดยจะกำหนดพารามิเตอร์ที่ใช้ทดสอบเหมือนหัวข้อที่ผ่านมาทุกประการ ผลการทดลองของ NQLMP แสดงดังรูปที่ 7.22 และรูปที่ 7.23



รูปที่ 7.22 ผลกระทบของสัญญาณรบกวนอิมพัลส์ต่ออะแดปทีฟอัลกอริทึมแบบ NQLMP ที่ได้จากการจำลองการทำงาน เมื่อค่า p =1, 2 และ 3 ตามลำดับ

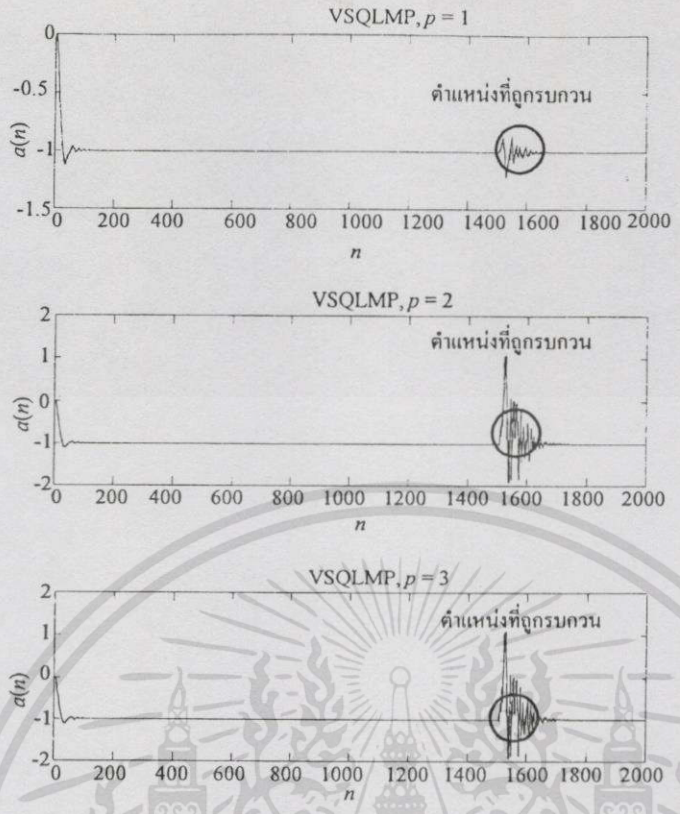


รูปที่ 7.23 ผลกระทบของสัญญาณรบกวนอิมพัลส์ที่เวลาจริง ต่ออะแดปทีฟอัลกอริทึมแบบ NQLMP, (a) p =1, (b) p = 2 และ (c) p = 3

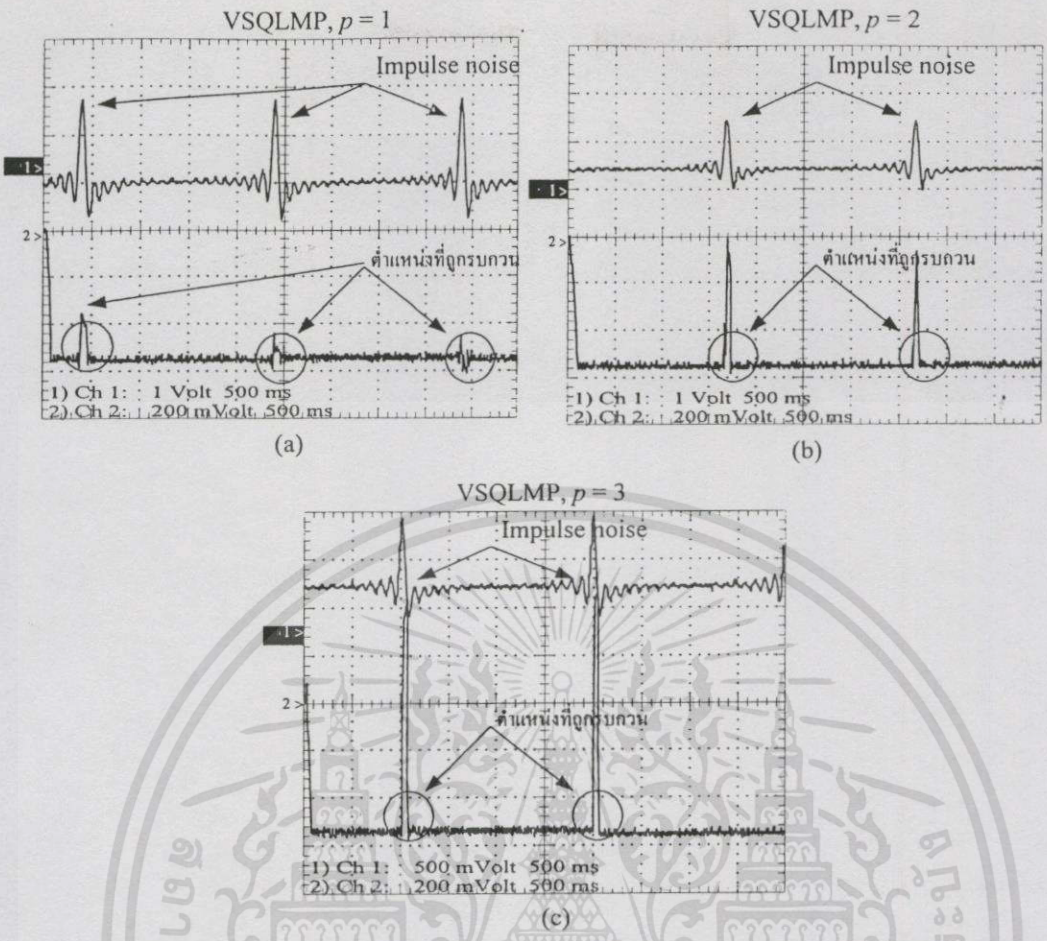
จากผลการทดลองในรูปที่ 7.22 และ 7.23 พบว่า ที่ค่า p สูง ๆ อะแดปทีฟอัลกอริทึมแบบ NQLMP จะทำงานได้แย่มาก แม้ว่าจะสามารถทนต่อสัญญาณรบกวนอิมพัลส์ได้ดีก็ตาม

สำหรับรูปที่ 7.24 และ รูปที่ 7.25 เป็นผลการทดลองของอะแดปทีฟอัลกอริทึมแบบ VSQLMP ซึ่งจากรูป พบว่า อะแดปทีฟอัลกอริทึมแบบ VSQLMP จะทนต่อสัญญาณรบกวนอิมพัลส์ได้น้อยลง เมื่อ p มีค่าสูงขึ้น

ดังนั้นจากการทดลองในหัวข้อนี้ ทำให้ได้ข้อสรุปคือ การจะนำอะแดปทีฟอัลกอริทึมแบบ NQLMP และแบบ VSQLMP ไปใช้งาน ควรจะเลือกให้ค่า p = 1 เท่านั้น จึงจะดีที่สุด ซึ่งรวมไปถึง อะแดปทีฟอัลกอริทึมแบบ QLMP ด้วย



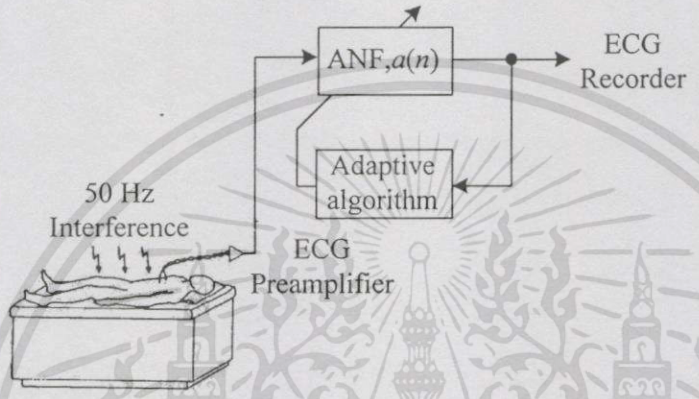
รูปที่ 7.24 ผลกระทบของสัญญาณรบกวนอิมพัลส์ต่ออะแดปทีฟอัลกอริทึมแบบ VSQLMP ที่ได้จากการจำลองการทำงาน เมื่อค่า p =1, 2 และ 3 ตามลำดับ



รูปที่ 7.25 ผลกระทบของสัญญาณรบกวนอิมพัลส์ที่เวลาจริงต่ออะแดปทีฟอัลกอริทึมแบบ VSQLMP, (a) $p = 1$, (b) $p = 2$ และ (c) $p = 3$

7.1.3 การประยุกต์ใช้งาน ANF สำหรับกำจัดสัญญาณซายน์ความถี่ 50 Hz ออกจากสัญญาณคลื่นไฟฟ้าหัวใจ (ECG)

ในหัวข้อนี้กล่าวถึง การประยุกต์ใช้งาน ANF สำหรับกำจัดสัญญาณซายน์ความถี่ 50 Hz จาก Power line 220 โวลต์ เหนี่ยวนำเข้าสู่สัญญาณคลื่นไฟฟ้าหัวใจ (ECG) ขณะที่แพทย์กำลังบันทึก ทำให้สัญญาณมีความผิดเพี้ยนไป อาจจะทำให้ได้ข้อมูลที่ผิดพลาด ซึ่งหลักการกำจัดสัญญาณความถี่ 50 Hz นี้ มีหลักการ ตามรูปที่ 7.26

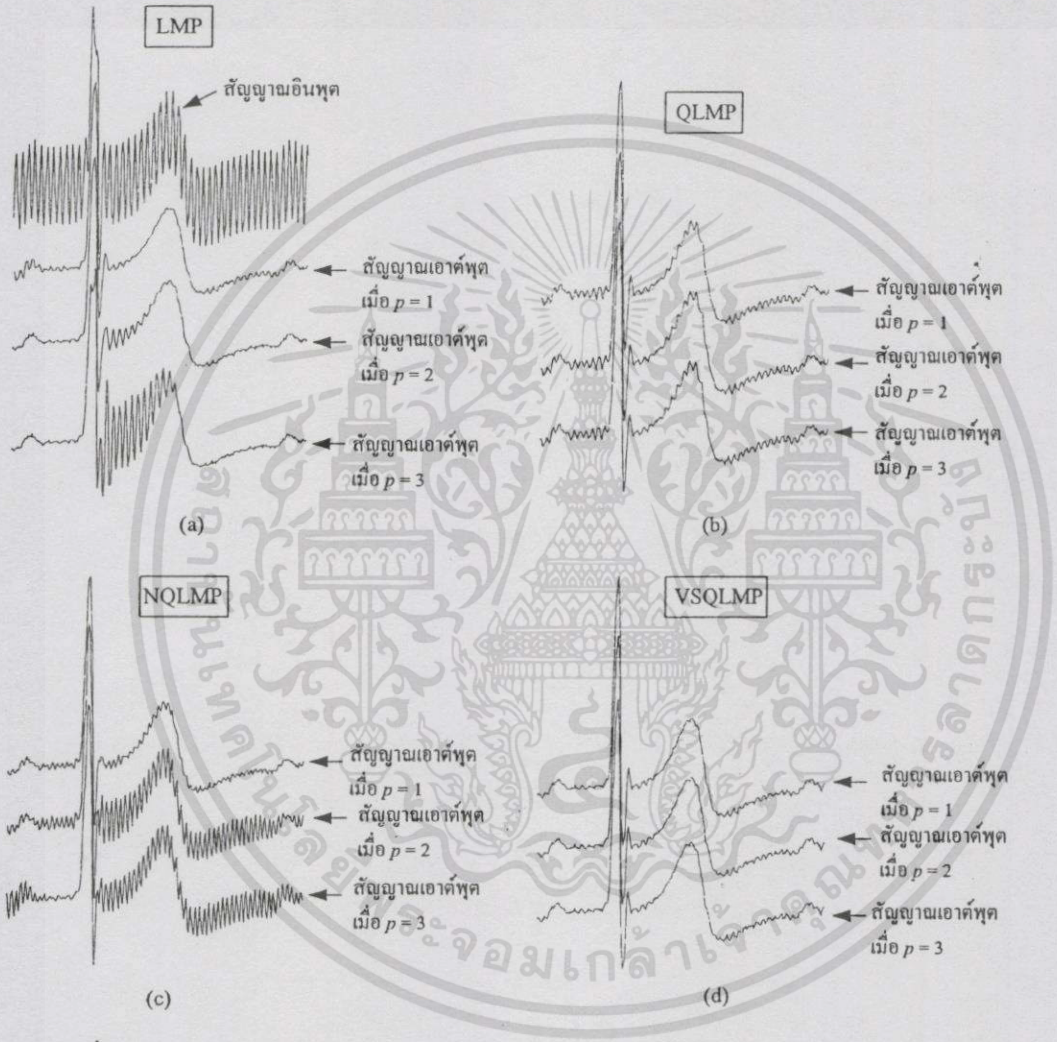


รูปที่ 7.26 การกำจัดสัญญาณรบกวน 50 Hz ออกจากคลื่น ECG โดยใช้ ANF

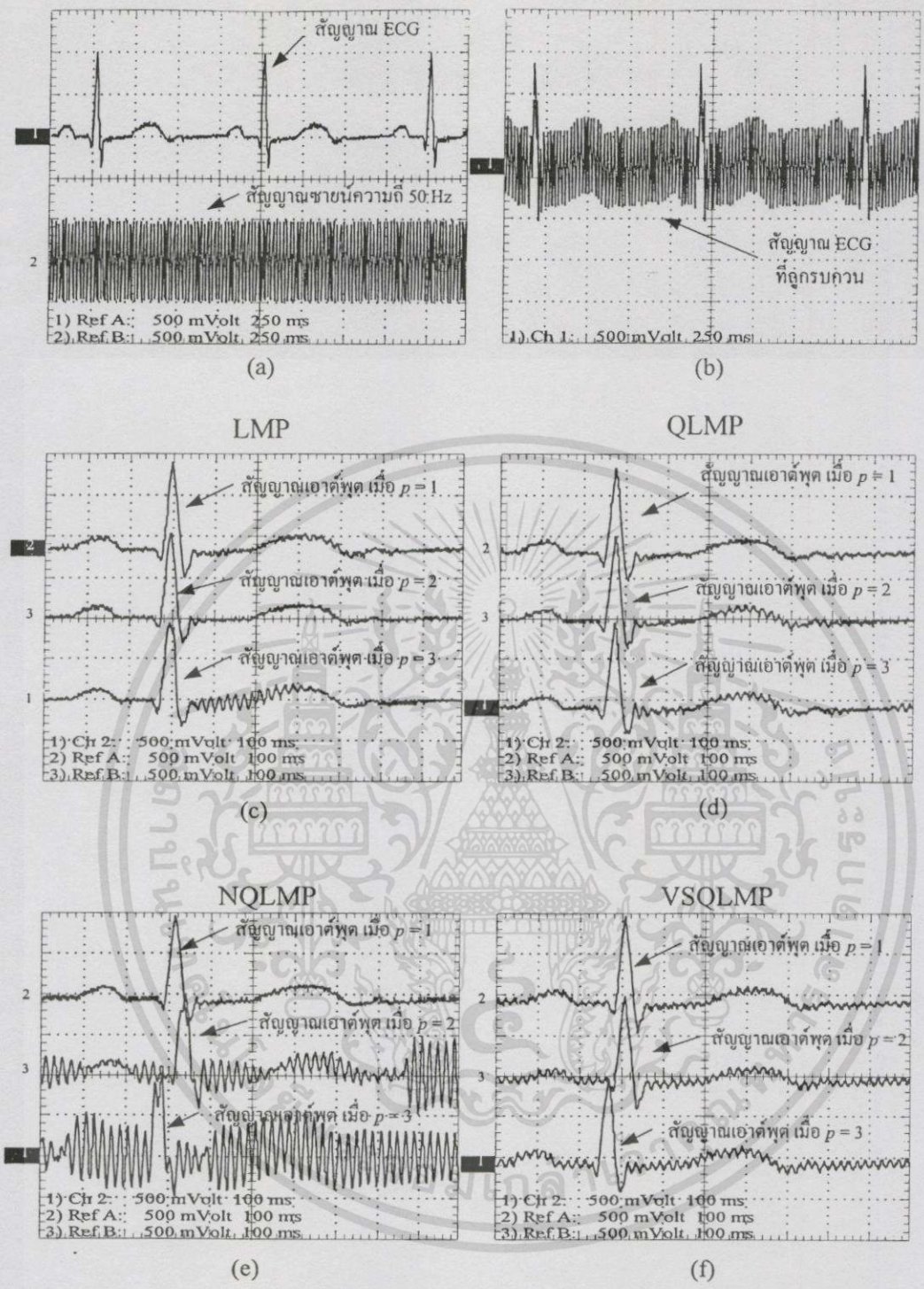
สเปกตรัมของสัญญาณ ECG จะอยู่ในช่วง [0, 100 Hz] การใช้ ANF เพื่อกำจัดสัญญาณซายน์ความถี่ 50 Hz จะทำให้ สเปกตรัมความถี่ 50 Hz ของ ECG ถูกกรองทิ้งด้วย เพื่อลดความผิดเพี้ยนที่อาจเกิดขึ้น ในการใช้งานจะกำหนดให้แบนด์วิดของ ANF แคบเท่าที่จะทำได้ ทั้งนี้เพื่อให้สัญญาณมีความผิดเพี้ยนน้อยที่สุด

7.1.3.1 ผลการกำจัดสัญญาณคลื่นไซน์ 50 Hz ออกจากสัญญาณ ECG ของอะแดปทีฟอัลกอริทึมแบบ LMP, QLMP, NQLMP และ VSQLMP

กำหนดความถี่สุ่มเป็น f_s = 400 Hz, $a(0) = -2\cos(0.25\pi)$, ส่วนพารามิเตอร์อื่น ๆ จะกำหนดให้เหมือนกับการทดลองที่ผ่านมา ผลการจำลองการทำงานแสดงได้ดังรูปที่ 7.27 และผลการทดลองที่เวลาจริงแสดงดังรูปที่ 7.28 ตามลำดับ



รูปที่ 7.27 ผลการจำลองการกำจัดสัญญาณซายน์ความถี่ 50 Hz ออกจากคลื่นไฟฟ้าหัวใจ, (a) LMP, (b) QLMP, (c) NQLMP และ (d) VSQLMP เมื่อ p = 1, 2 และ 3



รูปที่ 7.28 ผลการทดลองการกำจัดสัญญาณซายน์ความถี่ 50 Hz ออกจากคลื่นไฟฟ้าหัวใจที่เวลาจริง, (a) สัญญาณ ECG และสัญญาณซายน์ความถี่ 50 Hz, (b) สัญญาณ ECG ที่ถูกรบกวน, (c) LMP, (d) QLMP, (e) NQLMP และ (f) VSQLMP เมื่อ $p = 1, 2$ และ 3

จากรูปที่ 7.27 และ รูปที่ 7.28 จะเห็นว่ามีความสอดคล้อง ดังนั้นในหัวข้อนี้จะสามารถสรุปได้ว่า อะแดปทีฟอัลกอริทึมแบบ LMP, QLMP, NQLMP และ แบบ VSQLMP สามารถนำไปใช้สำหรับกำจัดสัญญาณคลื่นไซน์ 50 ที่รบกวนอยู่ในสัญญาณคลื่นไฟฟ้าหัวใจได้ดีพอ ๆ กันถ้าเลือกค่า $p = 1$ สำรับที่ค่า p สูง ๆ จะทำให้ประสิทธิภาพการทำงานของ LMP และ ของ NQLMP ลดลง (รูปที่ 7.25 (a), 7.25 (c), 7.26 (c) และ 7.26 (e)) แต่จะไม่มีผลกระทบต่ออะแดปทีฟอัลกอริทึม QLMP และ VSQLMP (รูปที่ 7.25 (b), 7.25 (d), 7.26 (d) และ 7.26 (f)) นอกจากนี้ การเลือกค่า $p = 1$ ยังสามารถลดการคูณลงได้ ดังแสดงไว้แล้วในตารางที่ 6.1

7.2 สรุป

บทนี้กล่าวถึงการนำ ANF ไปประยุกต์ใช้ สำหรับการประมาณค่าความถี่ที่ไม่ทราบค่าในสัญญาณรบกวน แบบ เกาส์เซียน และ สัญญาณรบกวนแบบ อิมพัลส์ และนำไปใช้สำหรับ กำจัดคลื่นซายน์ความถี่ 50 Hz ออกจากคลื่นไฟฟ้าหัวใจ ECG จากผลการจำลองการทำงาน และ ผลการทดลองที่เวลาจริงด้วยตัวประมวลผลสัญญาณดิจิตอล พบว่า ผลที่ได้มีความสอดคล้องกัน โดยที่ อะแดปทีฟที่ได้ นำเสนอ คือ QLMP, NQLMP และ VSQLMP สามารถทำงานได้เร็วกว่าแบบ LMP อะแดปทีฟอัลกอริทึมแบบ QLMP จะมีการคำนวณน้อยที่สุด และ VSQLMP จะคำนวณมากที่สุด แต่มีข้อดีที่ ทำงานได้เร็ว และ สัมประสิทธิ์ที่ประมาณได้ มีค่า variance และ bias ต่ำ อะแดปทีฟอัลกอริทึมแบบ NQLMP จะทำงานได้เร็ว แต่ สัมประสิทธิ์ที่ประมาณได้ มีค่า variance สูง อะแดปทีฟอัลกอริทึมแต่ละแบบจะมีข้อดี ข้อเสียแตกต่างกัน การเลือกนำไปใช้งาน จะขึ้นอยู่กับ วัตถุประสงค์ ของแต่ละคน แต่อย่างไรก็ตาม การจะเลือกนำไปใช้งานจะต้องคำนึงถึง เหตุผล สี่ประการดังได้กล่าวมาแล้วในบทที่ 4 หัวข้อที่ 4.4

บทที่ 8

สรุปและวิจารณ์ผลการวิจัย

วิทยานิพนธ์ได้นำเสนออะแดปทีฟอัลกอริทึมสามแบบ คือ QLMP, NQLMP และ VSQLMP โดยอะแดปทีฟอัลกอริทึมทั้งสามแบบที่ได้พัฒนาขึ้นนี้ จะเลือกใช้ $p = 1$ (p คือ ค่ายกกำลังของ cost ฟังก์ชัน) เท่านั้น ทั้งนี้เพื่อลดความซับซ้อนในการคำนวณ และค่า p ที่เพิ่มขึ้นไม่ได้ทำให้สมรรถนะการทำงานของอะแดปทีฟอัลกอริทึมที่ได้พัฒนาขึ้นทำงานได้ดีขึ้นแต่อย่างใด กลับทำให้แย่ลงอีกด้วย ซึ่งแตกต่างกับอะแดปทีฟอัลกอริทึมแบบ LMP ซึ่ง สมรรถนะของอัลกอริทึมขึ้นกับค่า p โดยต้องเลือกให้เหมาะสมกับงานที่จะไปใช้ เช่น เมื่อใช้สำหรับประมาณค่าความถี่จะต้องใช้ค่า p สูง ๆ เพื่อให้ทำงานได้เร็ว แต่ถ้าใช้สำหรับกำจัดสัญญาณคลื่นซายน์ 50 Hz ออกจากสัญญาณ ECG ต้องเลือกใช้ p ต่ำ ๆ จะเห็นว่า ไม่สะดวกเท่าใดนักเมื่อนำไปใช้งาน อะแดปทีฟอัลกอริทึมแบบ QLMP จะมีความซับซ้อนในการคำนวณน้อยที่สุด นอกจากนี้ ยังทำงานได้เร็ว และสมรรถนะของอัลกอริทึมไม่ขึ้นกับค่า p ทำให้สามารถนำไปใช้งานได้สะดวก อะแดปทีฟอัลกอริทึมแบบ NQLMP จะทำงานได้เร็วกว่าแบบ QLMP แต่คำตอบที่ประมาณได้จะมีค่า variance สูง และต้องเพิ่มการคำนวณ และอะแดปทีฟ อัลกอริทึมแบบ VSQLMP จะทำงานได้เร็วเหมือนกับ NQLMP แต่คำตอบที่ได้จะมีค่า variance ที่ต่ำกว่ามาก แต่ก็ต้องเพิ่มการคำนวณเช่นกัน จากรูปที่ 7.14 – รูปที่ 7.17 เป็นผลการจำลองการทำงานด้วยโปรแกรมคอมพิวเตอร์เพื่อหาค่า variance และ bias ของสัมประสิทธิ์ที่ประมาณได้ โดยปรับเปลี่ยนค่า *SNR* ให้มีค่าอยู่ในช่วง [-2, 20dB] พบว่าโดยรวมแล้ว อะแดปทีฟอัลกอริทึมแบบ VSQLMP จะมีค่า variance และ bias ต่ำเกือบทุกค่า *SNR* และอะแดปทีฟอัลกอริทึมแบบ NQLMP จะให้ค่า variance สูงกว่าแบบอื่น ในการทดลองได้กำหนดให้แบนด์วิดของตัวกรองคงที่ตลอดการทดลอง เพื่อง่ายในการเปรียบเทียบ และในการใช้งานจริง ก็จะต้องกำหนดไว้ค่าหนึ่งเช่นกัน โครงสร้างของตัวกรองที่ใช้จะเป็นแบบ Direct form ข้อดีของโครงสร้างนี้คือ มีความง่าย แต่มีข้อเสียคือ ทำงานได้ช้า และ มีผลจาก finite word length effect ด้วย

สำหรับการทดลองที่เวลาจริง ได้ใช้ตัวประมวลผลสัญญาณดิจิตอลเบอร์ TMS320C50 อยู่ในรูปของบอร์ดสำเร็จรูปที่เรียกกันว่า DSK (Digital Signal Processing Starter Kit) เป็นตัวประมวลผลมีขนาด 16 บิต มี A/D และ D/A มีขนาด 14 บิต ใช้ระบบตัวเลขในการคำนวณแบบ เลขจำนวนเต็ม (fixed point number system) ใช้ตัวเลขในการคำนวณแบบ เลข 2's complement ใช้จำนวนบิตสำหรับแทนทศนิยมจำนวน 12 บิต การประมวลผลแบบที่ใช้ A/D มีจำนวนบิตต่ำกว่าจำนวนบิตของตัวประมวลผลนี้ นิยมทำกันในทางปฏิบัติ การกระทำเช่นนี้ เหมือนกับการใช้สัญญาณขาเข้าที่มีการหารสี่ไปแล้ว แต่ขณะเดียวกัน สัญญาณขาออกก็รับได้เพียงแค่ 14 บิตเช่นกัน ซึ่งก็คือ ลดทอน

จากค่าสูงสุดไปสี่เท่าเช่นกัน แต่ส่วนที่พิเศษก็คือ ค่าอื่น ๆ ที่เกิดขึ้นกึ่งกลางในระบบจะสามารถถูกแทนได้ด้วย 16 บิต หรือคิดง่าย ๆ ได้ว่า ค่าอื่น ๆ สามารถมีค่าได้มากกว่าค่าที่ขาเข้า และขาออก สี่เท่า โดยไม่เกิดโอเวอร์โฟลภายใน และเป็นการเพิ่มความละเอียดในการคำนวณภายในด้วย

การเขียนโปรแกรมเพื่อสั่งให้ตัวประมวลผลทำงานจะทำได้สองวิธี คือ เขียนด้วยภาษาแอสแซมบลี และใช้คอมไพล์เลอร์แปลเป็นภาษาเครื่องป้อนให้กับตัวประมวลผล ซึ่งวิธีนี้จะทำงานได้เร็ว แต่ผู้เขียนต้องมีความชำนาญ ทั้งนี้เพราะจะต้องเข้าใจถึงโครงสร้างของตัวประมวลผลทั้งหมด วิธีที่สอง คือ เขียนด้วยโปรแกรมภาษาซี จากนั้นใช้ คอมไพล์เลอร์แปลงไปเป็นภาษาเครื่องป้อนให้กับ ตัวประมวลผล วิธีที่สองนี้ค่อนข้างจะสะดวกกว่าวิธีแรก ทั้งนี้เพราะภาษาซีเป็นภาษาขั้นสูง ทำให้สามารถทำความเข้าใจได้ง่าย และเป็นภาษาที่คุ้นเคยกันดี แต่ข้อเสียก็คือ จะไม่สามารถทราบได้เลยว่า ตัวประมวลผลมีขั้นตอนการทำงานอย่างไร เพราะกระบวนการต่าง ๆ คอมไพล์เลอร์เป็นผู้กระทำแต่เพียงผู้เดียว ทำให้เมื่อเกิดข้อผิดพลาดแล้วจะแก้ไขได้ยาก

การประมวลผลด้วยระบบตัวเลขแบบ fixed point จะมีข้อดีที่ ราคาถูก ทำงานได้เร็ว แต่มีข้อเสียหลายข้อเช่น เกิดความคลาดเคลื่อนจากการแบ่งขั้น, เกิดโอเวอร์โฟลจากการบวกได้ง่าย, เกิดสัญญาณรบกวนหลังการปัดเศษ และ เกิด limit cycles oscillation การจะไม่ให้ปัญหานี้เกิดขึ้นจะต้องหันไปใช้ระบบตัวเลขแบบ floating number แต่ก็มีราคาแพง ขนาดใหญ่ และทำงานได้ช้า

การจำลองการทำงานด้วยโปรแกรม MATLAB นั้นใช้ตัวเลขในการคำนวณแบบ floating point 64 บิตทำให้ผลที่ได้จากการจำลองการทำงานถูกต้องเกือบ 100 เปอร์เซ็น เมื่อเปรียบเทียบกับผลการทดลองด้วยตัวประมวล พบว่าผลที่ได้ มีความสอดคล้องกัน แต่ค่าที่ได้อาจผิดพลาดไป อันเนื่องจากใช้ตัวเลขในการคำนวณคนละระบบ แต่อย่างไรก็ตาม การทดลองที่เวลาจริงด้วยตัวประมวลผลสัญญาณดิจิตอลนั้น สามารถยืนยันถึงสมรรถนะในการทำงานของอะแดปทีฟอัลกอริทึมที่ได้พัฒนาขึ้นได้เป็นอย่างดี และยืนยันได้ว่า จะสามารถนำไปใช้งานจริงได้ด้วย

เนื่องจาก ทฤษฎีเกี่ยวกับตัวกรองอะแดปทีฟแบบ IIR ยังอยู่ในขั้นของการทำวิจัย ยังไม่มีสูตรสำเร็จเหมือนดังตัวกรองอะแดปทีฟแบบ FIR ทั้งนี้เพราะ ตัวกรองอะแดปทีฟแบบ IIR ทำงานในลักษณะเป็น nonlinear ซึ่งทำนายพฤติกรรมได้ยาก ต้องอาศัยการประมาณ ซึ่งจะต้องใช้ความรู้ทางด้านคณิตศาสตร์ และประสบการณ์อย่างมาก ดังนั้น ส่วนใหญ่ แนวทางที่ใช้สำหรับวิเคราะห์และทดสอบคุณสมบัติของตัวกรองอะแดปทีฟแบบ IIR จะนำเอาคอมพิวเตอร์มาช่วยในการวิเคราะห์โดยการจำลองการทำงาน จากนั้นทดสอบการทำงานจริงด้วยตัวประมวลผลสัญญาณดิจิตอล ดังที่ได้นำเสนอในวิทยานิพนธ์ ก็ใช้แนวทางนี้เช่นกัน

เอกสารอ้างอิง

[1] B. Widrow et al. "Adaptive noise canceling: Principles and applications." Proc. IEEE, vol. 63, Dec. 1975. pp. 1692-1716.

[2] L. J. Griffiths. "Rapid measurements of digital instantaneous frequency." IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-33, Aug. 1985. pp. 983-996.

[3] J. R. Treichler. "Transient and convergent behavior of the adaptive line enhancer." IEEE Trans. Acoust., Speech Signal Processing, vol. ASSP-27, Feb. 1979. pp. 53-62.

[4] T. Kwan and K. Martin. "Adaptive Detection and Enhancement of Multiple Sinusoids Using a Cascade IIR Filter." IEEE Trans. Circuits Syst., vol. CAS-36, no. 7, July 1989. pp. 937-947.

[5] K. Martin and Ming-Ting Sun. "Adaptive filters suitable for real-time spectral analysis." IEEE Trans. Circuits Syst., vol. CAS-33, Feb. 1986. pp. 218-229.

[6] D. Hush, N. Ahmed, R. David, and S. D. Stearns. "An adaptive IIR structure for sinusoidal enhancement, frequency estimation, and detection." IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-34, Dec. 1986. pp. 1380-1390.

[7] D. V. B. Rao and S. Y. Kung. "Adaptive notch filtering for the retrieval of sinusoids in noise." IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-32, Aug. 1984. pp. 791-802.

[8] D. Parikh, N. Ahmed, and S. D. Stearns. "An Adaptive Algorithm for Recursive Filters." IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-28, Feb. 1980. pp. 110-111.

[9] J. F. Chicharo and T. Sang Ng. "Gradient-Based Adaptive IIR Notch Filter for Frequency Estimation." IEEE Trans. Acoust. Speech, Signal Processing, vol. 38, no. 5, May 1990. pp. 769-777.

[10] P. A. Regalia. "An Improved Lattice-Based Adaptive IIR Notch Filter." IEEE Trans. Signal Processing, vol. 39, no. 9, Sep. 1991. pp. 2124-2128.

[11] N. I. Cho and S. U. Lee. "On the Adaptive Lattice Notch Filter for the Detection of Sinusoids." IEEE Trans. Circuits Syst. vol. 40, no. 7, July 1993. pp. 405-416.

[12] Soo-Chang Pei and chien-Cheng Tseng. "Adaptive IIR Notch Filter Based on Least Mean *p*-Power Error Criterion." IEEE Trans. Circuits syst. vol. 40, Aug. 1993. pp. 525-529.

[13] Mariane R. Petraglia, Sanjit K. Mitra and J. Szczupak. "Adaptive Sinusoid Detection Using IIR Notch Filters and Multirate Techniques." IEEE Trans. Circuits syst. Vol. 41, no. 11, Nov. 1994. pp. 709-717.

[14] Marina V. Dragosevic and Srdjan S. Stankovic. "An Adaptive Notch Filter with Improved Tracking Properties." IEEE Trans. Signal Processing, vol. 43, no. 9, Sep. 1995. pp. 2068-2075.

[15] R. Punchalard *et. al.* "Simplify Adaptive IIR Notch Filter Based on Least Mean *p*-Power Error Criterion." Proc. IEEE APCCAS, Nov. 1998. pp. 335-338.

[16] ราชู พันธ์ฉลาด และคณะ. "อะแดปทีฟ IIR นอตช์ตัวกรองบนพื้นฐานการพิจารณาค่าผิดพลาดเฉลี่ยกำลัง *p* น้อยที่สุดโดยใช้วิธีการควอนไตซ์เกรเดียนต์" การประชุมวิชาการวิศวกรรมไฟฟ้าครั้งที่ 21 ภาควิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี พฤศจิกายน 2541. หน้า 122-125.

[17] ราชู พันธ์ฉลาด และคณะ. "อะแดปทีฟ IIR นอตช์ตัวกรองสำหรับการตรวจวัดคลื่นซายน์โดยการนอร์มอลไลซ์ควอนไตซ์เกรเดียนต์" การประชุมวิชาการวิศวกรรมไฟฟ้าครั้งที่ 22 ภาควิชาวิศวกรรมไฟฟ้า มหาวิทยาลัยเกษตรศาสตร์ ธันวาคม 2542. หน้า 525-527.

[18] J. Shynk. "Adaptive IIR filtering." IEEE ASSP Magazine, April 1989. pp. 4-21.

[19] B. Widrow, Samuel D. Stearns. **Adaptive Signal Processing.** New Jersey : Prentice-Hall. 1985.

[20] Monson H. Hayes. **Statistical Digital Signal Processing and Modeling.** Canada : John Wiley & Sons. 1996.

[21] Y. Z. Ider and H. Koymen. "A New Technique for Line Interference Monitoring and Reduction in Biopotential Amplifiers." IEEE Trans. Biomed, vol. 37, No. 6, June 1990. pp. 624-631.

[22] Patrick S. Hamilton. "A Comparison of Algorithm for Ambulatory ECG Compression with Fixed Average Data Rate." IEEE Trans. Biomed, 1992. pp. 683-689.

[23] M. L. Ahlstrom and W. J. Tompkins. "Digital Filters for Real-Time ECG Signal Processing Using Microprocessors." IEEE Trans. Biomed, vol. BME-32, no. 9, Sep. 1985. pp. 708-713.

[24] Russell M. Mersereau, Mark J.T. Smith. **Digital Filtering.** John Wiley & Sons. 1994.

[25] John G. Proakis, Dimitris G. Manolakis. **Digital Signal Processing Principles, Algorithm, and Applications**. Prentice-Hall. 1996.

[26] Alan V. Oppenheim, Ronald W. Schafer. **Digital Signal Processing**. Prentice-Hall. 1975.

[27] Robert D. Strum, Donald E. Kirk. **First Principles of Discrete Systems and Digital Signal Processing**. Addison-Wesley. 1988.

[28] Lonnie C. Ludeman. **Fundamentals of Digital Signal Processing**. John wiley & Sons. 1987.

[29] R. M. Mersereau, M. J. T. Smith. **Digital Filtering A Computer Laboratory Textbook**. John Wiley & Sons. 1994. pp. 59-61.

[30] E. C. Lfeachor and B. W. Jervis. **Digital Signal Processing A Practical Approach**. Addison-Wesley Publishing Company. 1996. pp. 255-257.

[31] A. V. Oppenheim, R. W. Schafer. **Discrete-Time Signal Processing**. Prentice-Hall. 1999.

[32] S. K. Mitra. **Digital Signal Processing A computer-Based Approach**. McGRAW-HILL international edition. 1998. pp. 252-257.

[33] พรชัย ภวางษ์ศักดิ์. **การประมวลผลสัญญาณดิจิตอลเบื้องต้น**. โครงการตำราวิชาการ, มหาวิทยาลัยเทคโนโลยีมหานคร มหาวิทยาลัยเทคโนโลยีมหานคร. 2542.

[34] S. M. Kay. **Fundamentals Of Statistical Signal Processing Estimation Theory.** Prentice Hall. 1993.

[35] Dimitris G. Manolakis, Vinay K. Ingle, Stephen M. Kogon. **Statistical and Adaptive Signal Processing**. McGRAW-HILL. 2000.

[36] F. Mosteller, J. W. Tukey. **Data Analysis and Regression.** Addison-Wesley, Reading, Mass. 1977.

[37] J.M. Memdel. **Lessons in Digital Estimation Theory**. Prentice-Hall, Englewood Cliffs, N. J. 1987.

[38] S. Haykin. **Adaptive Filter Theory**. Prentice-Hall, Englewood Cliffs, N. J. 1986.

[39] B. Widrow, J. M. McCool, M. G. Larimore, and C. R. Johnson, Jr. "Stationary and nonstationary learning characteristics of the LMS adaptive filter." Porc. IEEE, vol. 64, no.8, Aug. 1976. pp. 1151-1162.

[40] I. D. Landau. **Adaptive control: The Model Reference Approach.** New York. Marcel Dekker. 1979.

[41] E. L. Jury. **Theory and Applications of the Z-Transform method**. New York. Wiley. 1964.

[42] S. A. Tretter. **Introduction to Discrete-Time Signal Processing**. New York. Wiley. 1976.

[43]ราชู พันธ์ฉลาด และ คณะ. "การลดสัญญาณรบกวนเกรเดียนต์ในควอนไตซ์เกรเดียนต์อัลกอริทึม." Mahanakorn Engineering Transaction. vol. 3, No. 1, Jan-April 2000. pp. 48-52.

[44] P. A. Regalia. **Adaptive IIR Filtering in signal Processing and Control**. Marcel Dekker, Inc. 1994.

[45] A. Anehorai. "A minimum parameter adaptive notch filter with constrained poles and zeros." IEEE Trans. Acoustics, Speech, and Signal Processing, vol. ASSP-33, no. 4, Aug. 1985. pp. 983-996.

[46] T. S. Ng. "Some aspects of an adaptive digital notch filter with constrained poles and zeros." IEEE Trans. Acoustics, Speech, and Signal Processing, vol. ASSP-35, no. 2, Feb. 1979. pp. 158-161.

[47] P. Stoica and A. Nehorai. "Performance analysis of an adaptive notch filter with constrained poles and zeros." IEEE Trans. Acoustics, Speech, and Signal Processing, vol. 36 no. 6, June 1988. pp. 911-919.

[48] C. S. Burrus and J. A. Barreto, "Least p-Power error design of FIR filters." Proc. IEEE int. Symp. Circuits and Systems, May 1992. pp. 545-548.

[49] J. Schrooeder, Rao Yarlagadda, J. Hershey. "L_p normed minimization with applications to linear predictive modeling for sinusoidal frequency estimation." Signal Processing, vol. 24, Aug. 1991. pp. 193-216.

[50] L. Ljung. "Asymptotic gain and search direction for recursive identification algorithms." Proc. 19th IEEE Conf. Dec. 1980.

[51] K. Mayyas, T. Aboulnasr. "A Robust Variable Step Size LMS-Type Algorithm: Analysis and Simulations." IEEE Trans. Signal Processing., vol. 45, no. 3, Mar. 1997. pp. 631- 639.

[52] Microprocessor Development Systems. **TMS320C5x DSP Starter Kit User's guide**. Texas Instruments. 1994.

[53] Digital Signal Processing Products. **TMS320C5x User's guide**. Texas Instruments. 1993.

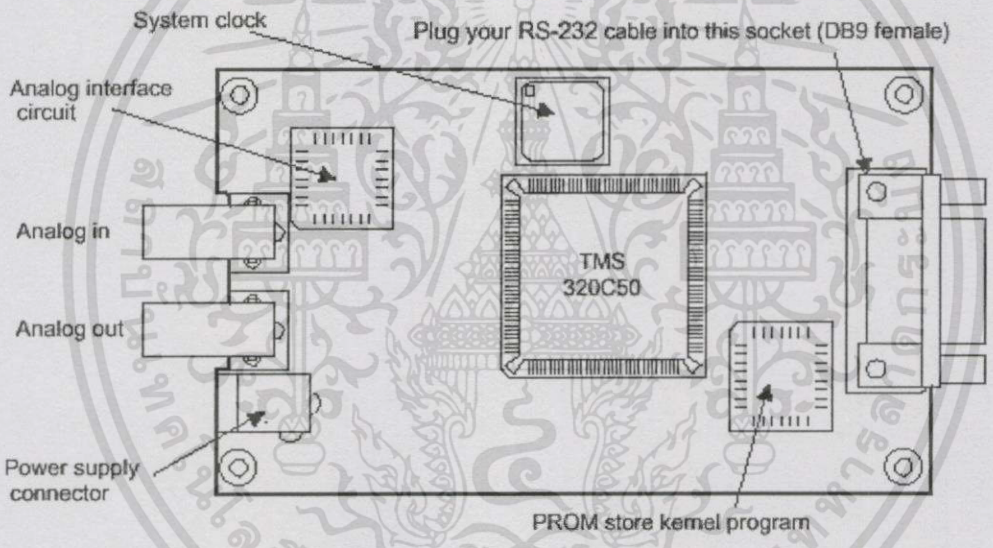
ภาคผนวก

ภาคผนวก ก.

ตัวประมวลผลสัญญาณดิจิตอล TMS320C50 (DSK)

ก.1 โครงสร้างของ TMS320C50 DSP Starter Kit(DSK)

บอร์ด TMS320C50 DSP Starter Kit(DSK) ประกอบด้วย IC ที่สำคัญ คือ เบอร์ TMS320C50 ซึ่งเป็นซีพียู (CPU) เบอร์ TLC32040C เป็น IC A/D และ D/A ซึ่งสามารถกำหนดอัตราสุ่มได้สูงสุด 19.2 kHz PROM ขนาด 32 Kbytes ในตัวเดียวกันซึ่งลักษณะของบอร์ด DSK ได้แสดงไว้ในรูปที่ ก.1 ในหัวข้อต่อไปจะกล่าวถึงรายละเอียดพื้นฐานของ TMS320C50 และ TLC32040



รูปที่ ก.1 บอร์ด TMS320C50 DSP Starter Kit(DSK)

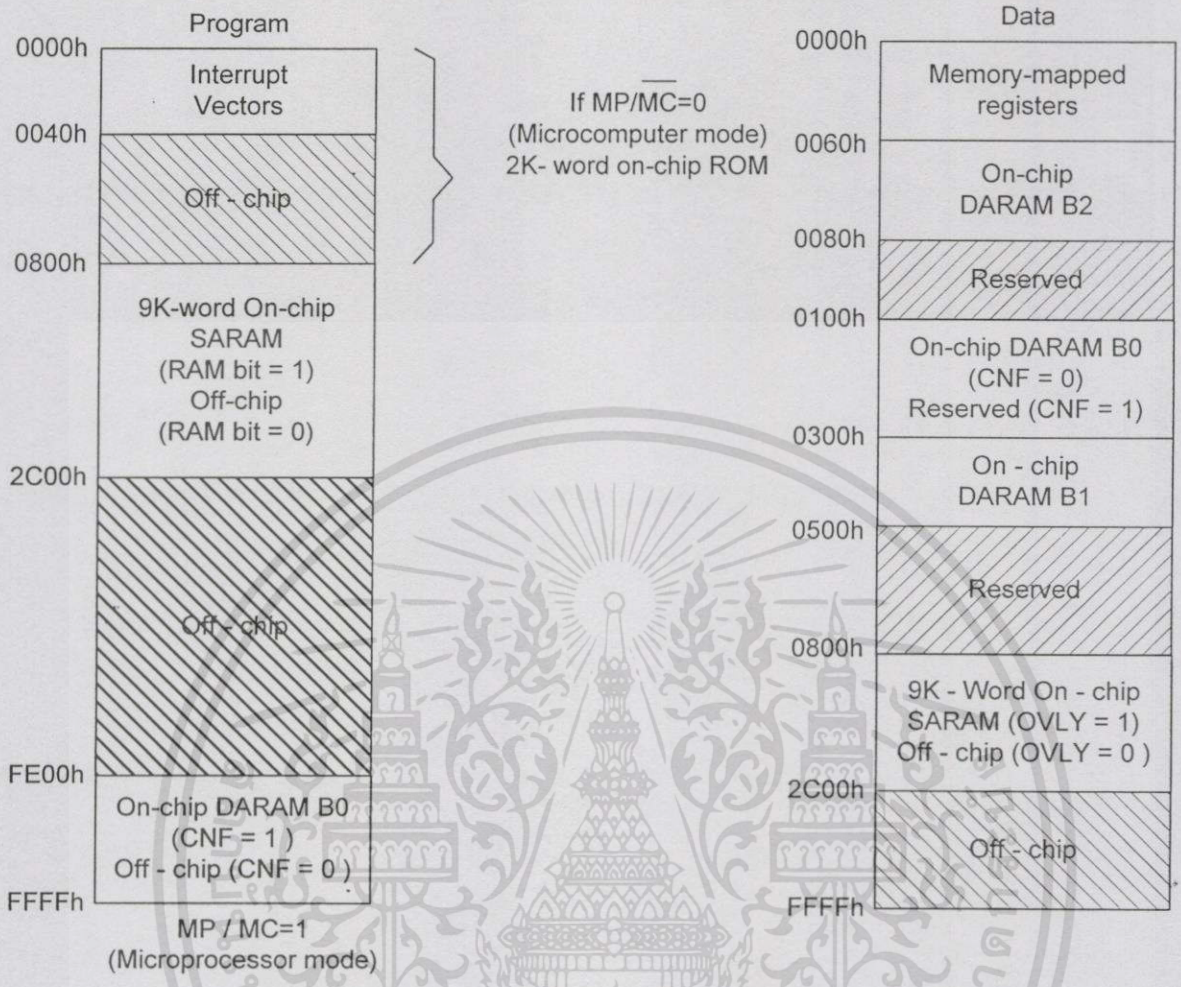
ก.1.1 โครงสร้าง IC เบอร์ TMS320C50 เป็น CPU ตระกูล 5x ของบริษัท TEXAS

มีคุณสมบัติสรุปได้ดังนี้ ส่วนรายละเอียดขาของ TMS320C50 สามารถดูได้จากคู่มือของ IC ตามที่อ้างอิงไว้ [52-53]

1. แรงดันสัญญาณอินพุต -0.3 ถึง 7 โวลต์
2. แรงดันสัญญาณเอาต์พุต -0.3 ถึง 7 โวลต์
3. อุณหภูมิในการทำงาน 0 ถึง 85 องศาเซลเซียส
4. ประกอบด้วยขาสัญญาณ 132 ขา

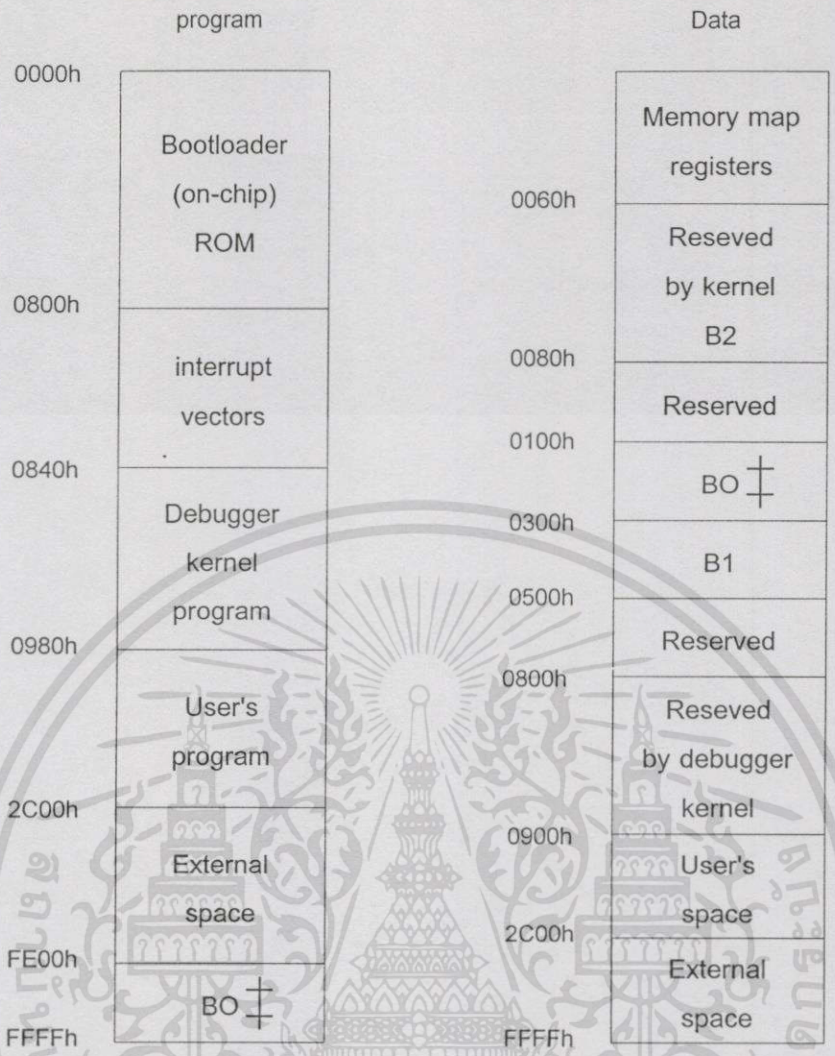
5. สัญญาณนาฬิกา 50 ns
6. I/O Port Parallel 64K Serial 2
7. Memory On Chip-RAM,DATA 1K,DATA+PROGRAM 9K -ROM PROGRAM 2K

IC เบอร์ TMS320C50 เป็นตัวประมวลผลสัญญาณดิจิตอล วงรอบการทำงานใช้เวลา 50 ns ภายในมีหน่วยความจำ Rom ในชิป 9K×16 บิต มีหน่วยความจำ Boot Rom ในชิป 2K×16 บิต พื้นที่หน่วยความจำที่สามารถขยายได้ 224K×16 บิต โดยแบ่งออกเป็น หน่วยความจำโปรแกรม 64K หน่วยความจำข้อมูล 64*K* และพื้นที่หน่วยความจำสำหรับอุปกรณ์ อินพุต / เอาต์พุต 64*K* รายละเอียดของหน่วยความจำแสดงดังรูปที่ ก.2 หน่วยความจำประมวลผลทางคณิตศาสตร์ / แอกคิวมูเลเตอร์บัฟเฟอร์ 32 บิต สามารถรองรับการคูณเลขฐานสองจำนวนขนาด 16 บิต ผลลัพธ์มีขนาด 32 บิต ใช้เวลาการคูณเพียงหนึ่งรอบคำสั่งเท่านั้น มีฮาร์ดแวร์สแต็ก 8 ระดับ มีตัวเลื่อนข้อมูลแบบขนาน (Parallel Shift) ขนาด 16 บิต มีตัวกำเนิดสัญญาณนาฬิกาภายใน ชิปรีจีสเตอร์อีอกซิลารีมากถึง 8 ตัว ใช้สำหรับเลือกลำดับรีจีสเตอร์เพื่อควบคุม CPU กับโปรแกรมย่อยการอินเตอร์รัปต์ มีพอร์ตอนุกรมเป็นแบบ Full-Duplex Syncronous สำหรับการติดต่อโดยตรงระหว่าง C5x กับไดร์เวอร์ที่ต่ออยู่ มีสัญญาณนาฬิกาที่เป็นคาบภายในและมีรีจีสเตอร์นับสำหรับหยุดเริ่มและรีเซตซอร์ฟแวร์ มีอินพุต / เอาต์พุต แบบขนานขนาด 64 Kbytes



รูปที่ ก.2 หน่วยความจำของ TMS320C50

จากรูปที่ ก.2 เป็นตำแหน่งหน่วยความจำของ TMS320C50 แต่เมื่อทำการต่อเป็น starter kit ตำแหน่งหน่วยความจำจะถูกเปลี่ยนไปเป็น ดังรูปที่ ก.3

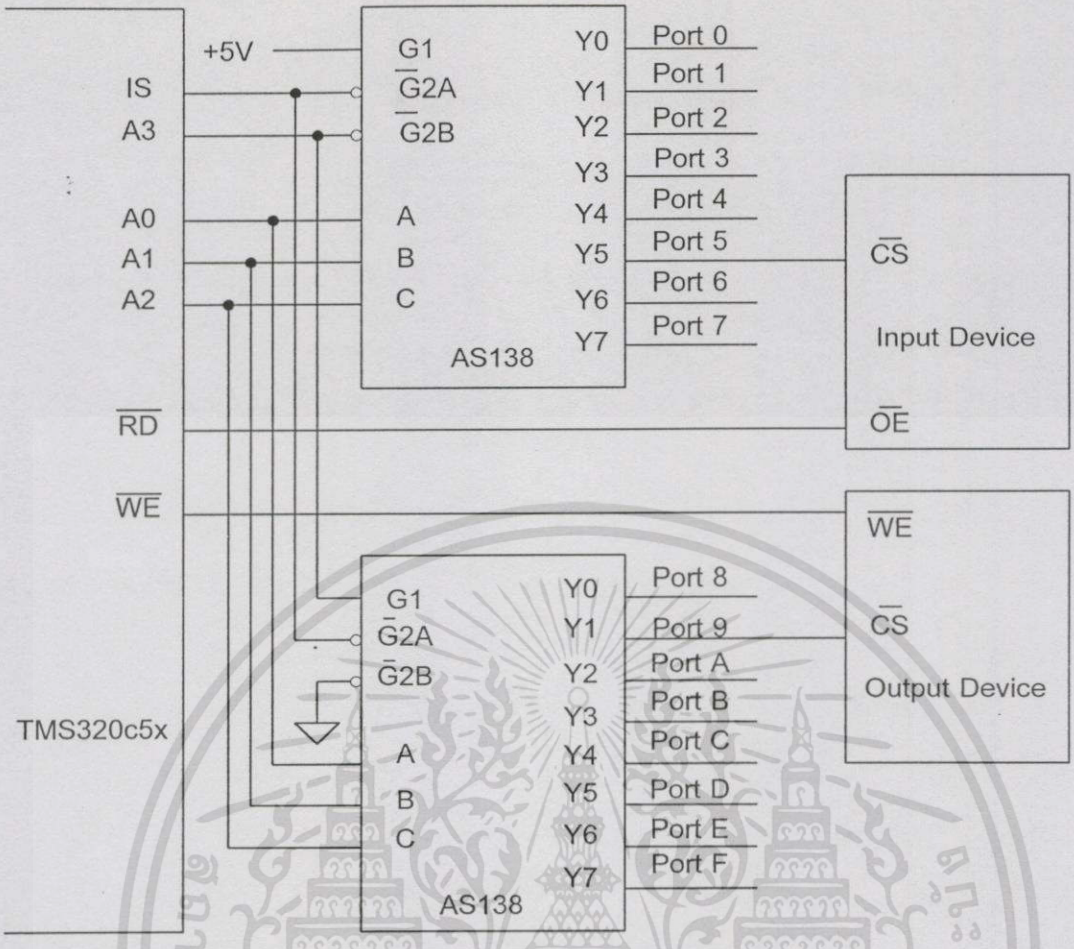


รูปที่ ก.3 หน่วยความจำของ TMS320C50 DSP STARETER KIT(DSK)

ทางเข้าออกแบบขนาน (Parallel I/O Port)

ประโยชน์หลักของพอร์ตขนานช่วยในการออกแบบการต่ออุปกรณ์ภายนอกเพิ่ม เช่น ADC DAC ซึ่ง TMS320C50 สามารถออกแบบให้รับหรือส่งข้อมูลได้ โดยขาที่ทำหน้าที่เป็นตัวควบคุมคือ RD ในสภาวะปกติเมื่อยังไม่ทำงานจะมีสถานะเป็น High เมื่อขณะทำงานจะให้สถานะที่เป็น Low ออกมาเพื่อทำให้ขา OE ของ Input Drive ทำงานเมื่อ Y5 จาก AS138 ทำงานจะทำให้สามารถส่งข้อมูลเข้ามายัง TMS320C50 ได้

WE ในสถานะปกติเมื่อยังไม่ทำงานจะมีสถานะเป็นค่า High เมื่อขณะทำงานจะให้สถานะที่เป็น Low ออกมา เพื่อทำให้ขา WE ของ Output Drive ทำงานส่งค่า Output ออกไปได้ การทำงานของ RD,WE จะทำงานพร้อมกับขา IS และ Address กับ Data โดย A0-A15 จะเป็นตัวกำหนดการรับและส่งเข้าทางพอร์ตใด



รูปที่ ก.4 แสดงการต่อ Parallel Port

อินเตอร์รัปต์(Interrupt)

วงจรประมวลผล TMS320C50 จะมีการต่ออินเตอร์รัปต์ภายนอกอยู่ 4 ขาคือ INT1-INT4 โดยการอินเตอร์รัปต์ภายในตัว TMS320C50 เกิดจากทางเข้าออกแบบอนุกรม (TINT และRINT) ตัว Timer (TINT)TDM PORT(TRNT และTXNT) โดยอินเตอร์รัปต์ที่สำคัญมากที่สุดคือ RS ดูได้จากตารางที่ ก.1

ตารางที่ ก.1 ตารางอินเตอร์รัปต์

Name	Location		Priority	Function
	Dec	Hex		
$\overline{\text{RS}}$	0	0	1(highest)	External reset signal
$\overline{\text{NMI}}$	36	24	2	Nonmaskable interrupt
$\overline{\text{INT1}}$	2	2	3	External user interrupt#1
$\overline{\text{INT2}}$	4	4	4	External user interrupt#2
$\overline{\text{INT3}}$	6	6	5	External user interrupt#3
TINT	8	8	6	Internal timer interrupt
RINT	10	A	7	Serial port receive interrupt
XINT	12	C	8	Serial port transmit interrupt
TRINT	14	E	9	TDM port receive interrupt
TXNT	16	10	10	TDM port transmit interrupt
$\overline{\text{INT4}}$	18	12	11	External user interrupt#4
-	20-23	14-21	N/A	Reserved
TRAP	34	22	N/A	Trap instruction vector
-	38-39	26-27	N/A	Reserved
-	40-63	28-3F	N/A	Solfware interrupt

ในตัวประมวลผลจะมีรีจีสเตอร์ IMR (Interrupt mask register)เป็นตัวกำหนดว่าจะให้อินเตอร์รัปต์ตัวใดทำงานบ้างโดยรีจีสเตอร์ IMR มีโครงสร้างดังนี้ เมื่อนำ TMS320C50 สร้างเป็น DSK อินเตอร์รัปต์ภายในจะถูกใช้โดย TLC 32040 คือ XINT จะถูกใช้โดย DAC และ TINT ถูกใช้โดย ADC ซึ่งจะกล่าวรายละเอียดต่อไป

ตารางที่ ก.2 ตารางรีจีสเตอร์ IMR

15 9	8	7	6	5	4	3	2	1	0
Reserved	$\overline{\text{INT4}}$	TXNT	TRNT	XIN T	RINT	TINT	$\overline{\text{INT3}}$	$\overline{\text{INT2}}$	$\overline{\text{INT}}$

ถ้าในรีจีสเตอร์ IMR กำหนดให้มีค่าเท่ากับ 12h เป็นการกำหนดให้ตัวประมวลผลรับอินเตอร์รัปต์จากภายนอกที่ 2 (INT2) และรับอินเตอร์รัปต์ภายในจากทางเข้าออกแบบอนุกรมคือ RINT (ADC) จะพบว่าภายในรีจีสเตอร์ IMR จะไม่มี RS และ NMI จึงไม่มีผลต่อการรีเซต และถ้า IMR ถูกกำหนดด้วย 22h ซึ่งจะเป็นการกำหนดอินเตอร์รัปต์จากภายนอกที่ 2 (INT2) และอินเตอร์รัปต์จากภายในคือ XINT(DAC)

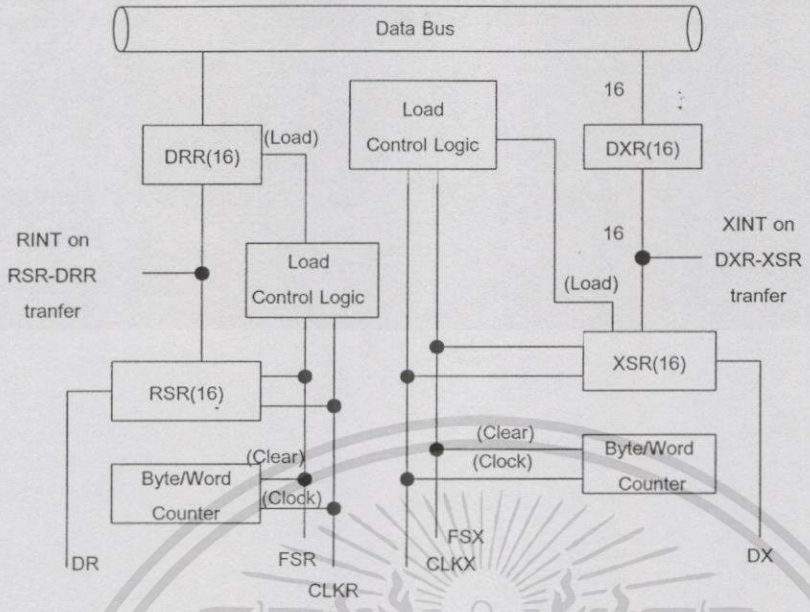
ทางเข้าออกแบบอนุกรม(Serial Port)

พอร์ตอนุกรมของ TMS320C50 นี้ถูกใช้โดย Debugger ของ Starter kit (DSK) ซึ่งจะใช้ในการติดต่อกับระบบคอมพิวเตออร์เพื่อใชในการจำลองการทำงานของ DSK พอร์ตอนุกรมจะมีรีจีสเตอร์ที่สำคัญ ดังนี้

ตารางที่ ก.3 รีจีสเตอร์ของทางเข้าออกแบบอนุกรม

Registers	Description
SPC	Serial port control register
DXR	Data transmit register
DRR	Data receive register
XSR	Transmit shift register
RSR	Receive shift register

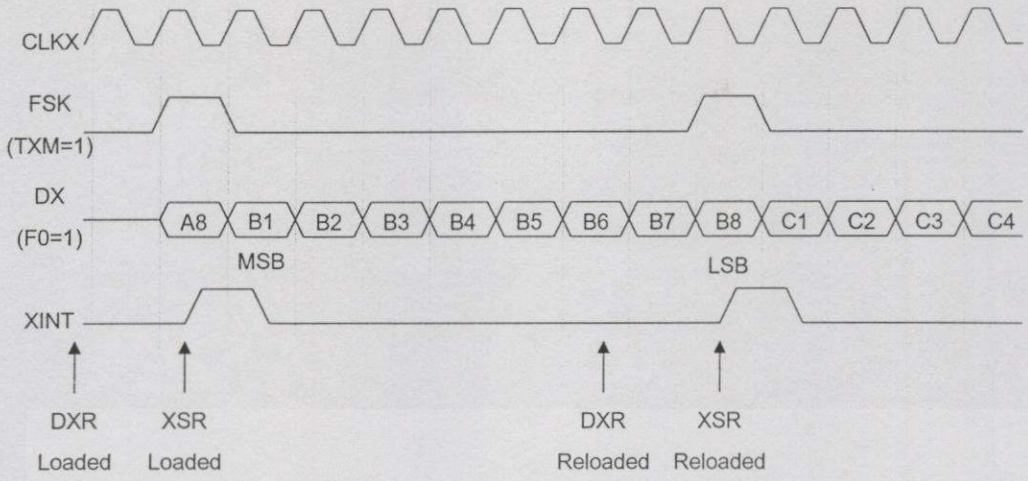
จากตารางที่ ก.3 การทำงานของทางเข้าออกแบบอนุกรมจะถูกควบคุมโดยรีจีสเตอร์ SPC การส่งข้อมูลในทางเข้าออกแบบอนุกรมจะส่งออกทางรีจีสเตอร์ DXR และการรับข้อมูลจะผ่านทางรีจีสเตอร์ DRR



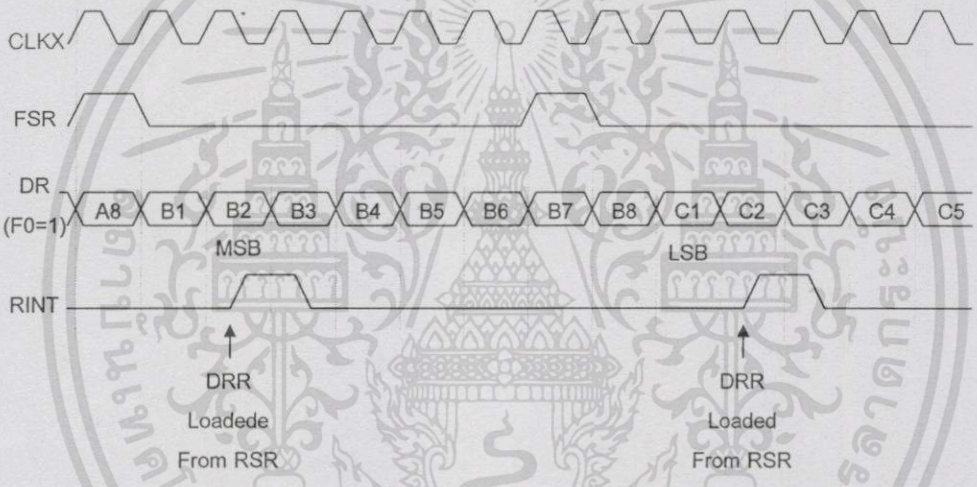
รูปที่ ก.5 บล็อกไดอะแกรมของทางเข้าออกแบบอนุกรม

การส่งข้อมูลใน DXR จะสำเร็จได้ก็ต่อเมื่อได้ทำการคัดลอกข้อมูลจาก DXR ไปยัง XSR เพื่อส่งค่าออกไปทางขา DX ในขณะนี้การทำงานของบิต XRDY ในรีจีสเตอร์ SPCและบิต XINT ในรีจีสเตอร์ IMR จะเซตค่าตัวเองให้เป็น “1” ทำให้เกิดการส่งข้อมูล

การรับข้อมูลมีการทำงานที่คล้ายกับการส่ง โดยจะรับข้อมูลทางขา DR และส่งข้อมูลเข้ารีจีสเตอร์ RSR และส่งต่อไปยัง DRR ทำให้บิต RRDY ในรีจีสเตอร์ SPC และบิต RINT ในรีจีสเตอร์ IMR เซตค่าตัวเองให้เป็น “1” เกิดการรับข้อมูล โดยแสดงการทำงานของการส่งและรับทางเข้าออกแบบอนุกรมได้ดังนี้

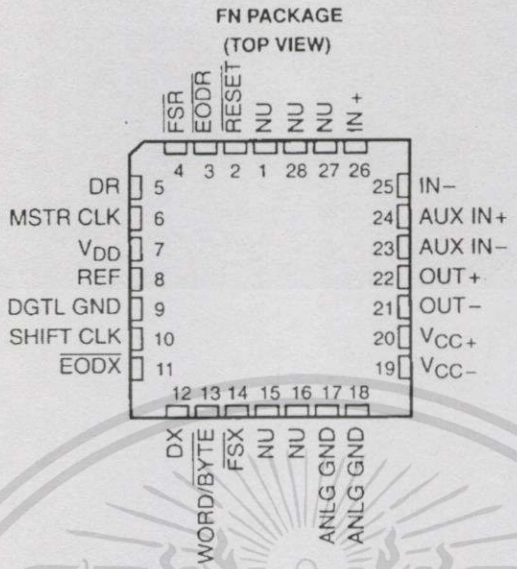


รูปที่ ก.6 การส่งข้อมูลของทางเข้าออกแบบอนุกรม



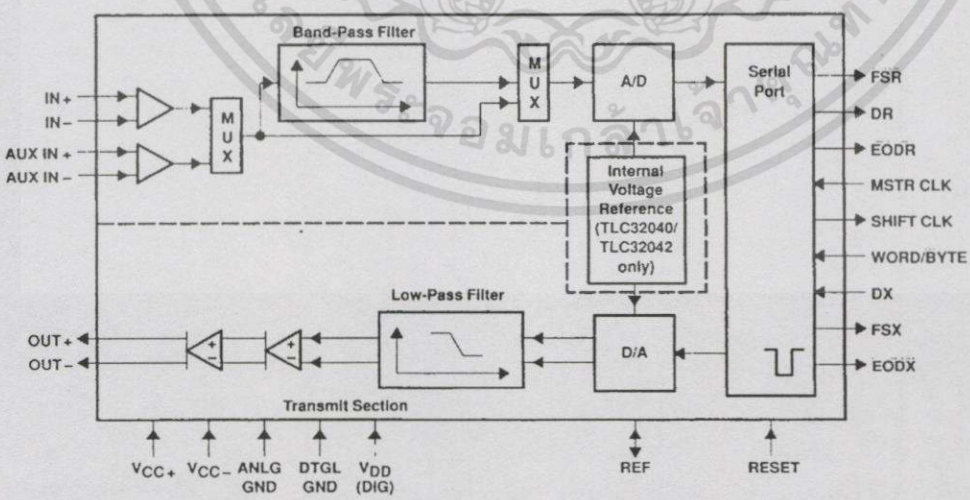
รูปที่ ก.7 การรับข้อมูลทางเข้าออกแบบอนุกรม

ก.1.2 โครงสร้าง IC เบอร์ TLC32040



รูปที่ ก.7 IC เบอร์ TLC32040

IC เบอร์ TLC32040 ทำหน้าที่แปลงผันอนาลอกเป็นดิจิตอล และแปลงผันดิจิตอลเป็นอนาลอก 14 บิต ภายในตัวเดียวกัน สามารถใช้ความถี่ในการสุ่มได้ถึง 19.2 kHz มีวงจรกรองความถี่สำหรับใช้ป้องกันการเกิด Aliasing ทางด้าน A/D และมีวงจรกรองความถี่ต่ำผ่านอยู่ทางด้าน D/A อยู่ภายในบอร์ด ซึ่งส่วนนี้ก็คือ Analog Interface Circuit (AIC) ซึ่งทำหน้าที่กำหนดความถี่ในการสุ่มของการแปลงสัญญาณอะนาลอกเป็นสัญญาณดิจิตอลหรือการกำหนดความถี่ในการสุ่มเพื่อไม่ให้เกิด Aliasing



รูปที่ ก.8 ส่วนประกอบของส่วน Analog Interface Circuit(AIC)

การ Initializing บอร์ด TMS320C50 DSP Starter Kit(DSK)

การ Initialize DSK สามารถหาโปรแกรมตัวอย่างได้จาก ftp.ti.com/pub/tms320 การ Initializing บอร์ด DSK นั้นทำได้โดยการใช้ซอฟแวร์ในการ Initializing และในการ Initializing บอร์ด นั้นจะทำ 3 อย่างด้วยกัน คือ

1. การกำหนดค่า Timer ในซิป TMS320C50
2. กำหนดค่า Serial Port ของซิป TMS320C50
3. การกำหนดส่วน Analog Interface Circuit(AIC)

ในที่นี้จะขอกล่าวถึงเฉพาะการกำหนดส่วน Analog Interface Circuit(AIC) อย่างเดียวเท่านั้น เพราะในการกำหนดในข้อ 1,2 นั้นค่อนข้างจะยุ่งยาก และจะไม่ค่อยได้เปลี่ยนแปลงบ่อยสักเท่าไร การกำหนดส่วนของ AIC ทำเพื่อให้บอร์ด DSK สามารถรับสัญญาณอินพุตเข้ามประมวลผลแล้วส่งออกทางเอาท์พุตได้อย่างถูกต้องลักษณะโครงสร้างของส่วน AIC แสดงไว้ในรูปที่ ก.8 จากรูป AIC ประกอบไปด้วยส่วนต่างๆดังนี้

-analog input สัญญาณอินพุตมีการกำหนดได้ 2 ทางคือ ทาง +IN,-IN และ AUX IN+,AUX IN- อัตราขยายของอินพุต สามารถเลือกได้เป็น 1,2 และ 4 เท่า โดยการเลือกรับสัญญาณอินพุต และอัตราขยายสามารถควบคุมและเลือกได้จากซอฟแวร์

-A/D bandpass filter แบนด์พาสฟิลเตอร์สามารถเลือกใช้หรือไม่ใช้ได้โดยการควบคุมด้วยซอฟแวร์ เช่นกันโดยความถี่คัทออฟสามารถคำนวณได้ดังที่จะได้กล่าวต่อไป

-A/D Converter วงจรแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอลจะมีอัตราการสุ่มข้อมูลและแปลงข้อมูลตามที่เราต้องการโดยการคำนวณแล้วใช้ซอฟแวร์ควบคุมการคำนวณจะได้กล่าวต่อไป

-D/A Converter วงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณอนาลอกจะมีอัตราการสุ่มข้อมูลและแปลงข้อมูลเท่ากับด้าน A/D ถ้าเรากำหนดให้โหมดการสุ่มสัญญาณเป็นแบบซิงโครนัส (Synchornous) ในบางงานอาจจำเป็นที่ต้องใช้แบบอะซิงโครนัส (Asynchornous) คือสามารถกำหนดให้อัตราในการสุ่มทั้งสองทางไม่เท่ากันได้การเลือกโหมดการสุ่มสัญญาณใช้ซอฟแวร์ควบคุมได้ ส่วนวงจรกรองความถี่ต่ำผ่าน (Low pass filter) ทำการเลือกความถี่คัทออฟได้ด้วยการคำนวณและจะสัมพันธ์กับความถี่ในการสุ่มสัญญาณด้วย

รีจิสเตอร์ที่ใช้กำหนดว่าจะเลือกส่วนต่างๆของ AIC นั้นส่วนมากแล้วจะใช้ตัวแปรชื่อ AIC_CTR โดยข้อมูลในรีจิสเตอร์จะเป็นตัวบอกว่าเลือกใช้อะไรบางซึ่งมีรายละเอียดดังต่อไปนี้

d2 = 0/1 deletes/inserts the band pass filter

d3 = 0/1 disables/enables the loopback function

d4 = 0/1 disables/enables AUX IN+ and AUX IN- pins

d5 = 0/1 Asynchronous/Synchronous transmit and receive sections

d6 = 0/1 gain control bit

d7 = 0/1 gain control bit

เช่น AIC_CTR=18h จะได้ว่า

d2 = 0 deletes the band pass filter

d3 = 0 disables the loopback function

d4 = 0 disables AUX IN+ and AUX IN- pins

d5 = 1 Synchronous transmit and receive sections

d6 = 1 gain control bit (gain =1 เท่า)

d7 = 0 gain control bit

การกำหนดความถี่ในการสุ่มข้อมูลสามารถกำหนดได้โดยการเขียนโปรแกรม ซึ่งสามารถคำนวณหาค่ารีจิสเตอร์ที่ใช้ควบคุม AIC ให้มีการสุ่มข้อมูลตามที่ต้องการ โดยเริ่มจากการคำนวณความถี่คัทออฟของวงจรกรองความความถี่ก่อน

$$f_{cLP} = \frac{f_{MCLK}(KHz)}{2 \times 288kHz \times TA} \times 3.6kHz \qquad \text{(ก.1)}$$

(3.6 kHz เป็น normalized Frequency ด้านความถี่ต่ำ)

$$f_{cHP} = \frac{f_{MCLK}(KHz)}{2 \times 288kHz \times TA} \times 300Hz \qquad \text{(ก.2)}$$

(300Hz เป็น normalized Freqency ด้านความถี่สูง)

$$f_s = \frac{f_{MCLK}}{2 \times TA \times TB} \qquad \text{(ก.3)}$$

เมื่อ

f_s = ความถี่ในการสุ่ม

f_{MCLK} = 10 MHz

f_{cLP} = ความถี่ตัด(Freqency Cutoff) ด้านความถี่ต่ำของวงจร Low Pass Filter

f_{cHP} = ความถี่ตัด(Freqency Cutoff)ด้านความถี่สูงของวงจร Band Pass Filter

TA ,TB คือ ตัวแปรที่ใช้สำหรับกำหนดความถี่(ควรกำหนดให้ $TA = RA, TB = RB$ เพื่อที่จะได้สะดวกในการแปลงสัญญาณอนาลอกเป็นดิจิตอล)

ตัวอย่าง

เช่น ต้องการความถี่ในการสุ่มสัญญาณ 8 kHz(fs=8 kHz) ,ความถี่คัทออฟของวงจรความถี่ต่ำผ่าน(f_{cLP})เท่ากับ 2.5 kHz และ f_{MCLK}=10MHz

จะได้ว่า $$TA = \frac{f_{MCLK}\ (KHz)}{2 \times 288\ kHz \times 2.5\,kHz} \times 3.6\,kHz = 25$$

และจาก $$f_s = \frac{f_{MCLK}}{2 \times TA \times TB}$$

จะได้ว่า $$TA \times TB = \frac{10MHz}{2 \times 8kHz} = 625$$

เพราะฉะนั้น $$TB = \frac{625}{25} = 25$$

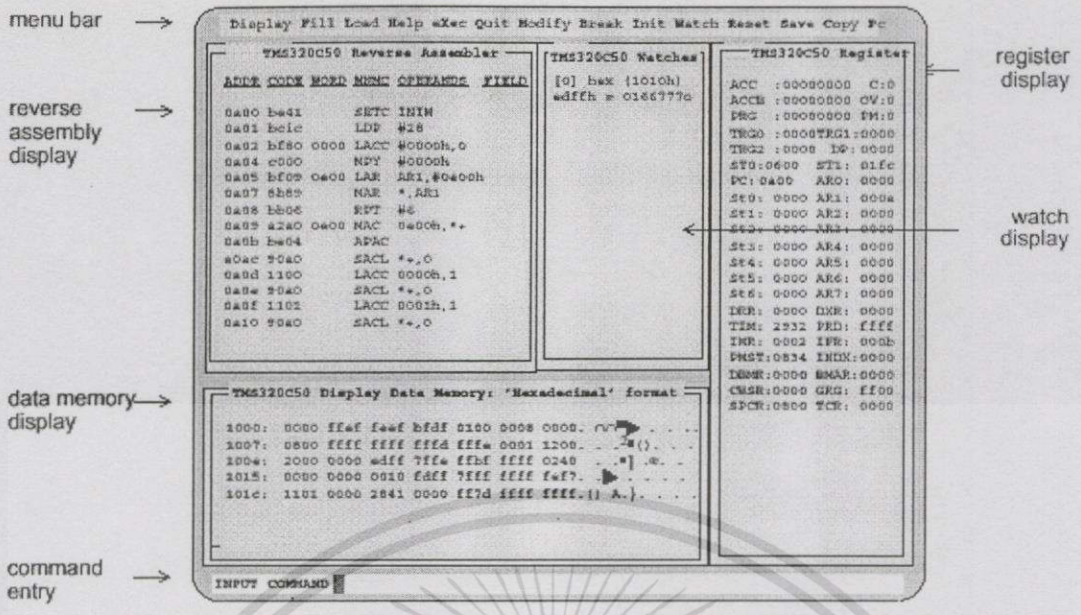
ก.1.3 การต่อบอร์ด DSK เข้ากับคอมพิวเตอร์

1. ต่อสายเคเบิ้ล RS-232 เข้ากับคอมพิวเตอร์พอร์ต COM1 หรือ COM2 แล้วต่อปลั๊ก RS-232อีกด้านเข้ากับปลั๊ก RS-232 ของบอร์ด DSK(ปลั๊ก RS-232แบบตัวเมีย)
2. ต่อแจ็ค Power Supply(9-Vac) เข้ากับบอร์ด DSK (กรณีมีแจ็ค Power Supply)
3. ต่อสายสัญญาณอินพุต และเอาต์พุตกรณีได้ใช้
4. เพื่อให้แน่ใจว่าต่อบอร์ด DSK เข้ากับคอมพิวเตอร์ถูกต้องและทำงานได้ให้ทำการทดสอบโดยการทดลองรันโปรแกรม Debugger ดูคือโปรแกรม DSK5D.EXE ที่ DOS Prompt ดังนี้

   C:\DSK5D Cx

   Cx เท่ากับ C1 กรณีต่อบอร์ด DSK กับ COM1 และเท่ากับ C2 กรณีต่อกับ COM2

ถ้าไม่มีอะไรผิดพลาดจะปรากฏหน้าต่างของโปรแกรม Debugger ของบอร์ด DSK ดังรูปที่ ก.9



รูปที่ ก.9 หน้าต่างของโปรแกรม Debugger ของบอร์ด DSK

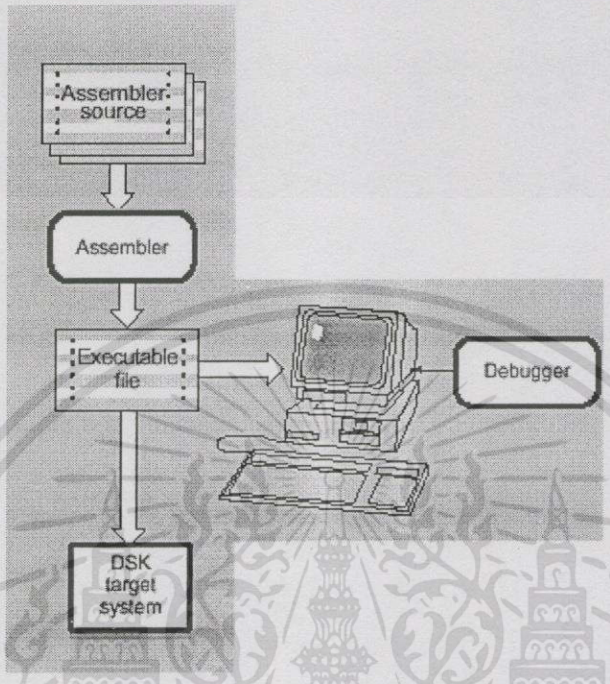
ถ้าหากยังเข้าโปรแกรมยังไม่ได้ให้กับไปตรวจสอบการเชื่อมต่ออุปกรณ์ทั้งหมดให้ถูกต้องอีกครั้ง การออกจากโปรแกรม Debuger ให้กด Q แล้ว Y

ก.2 การเขียนโปรแกรมด้วยภาษาแอสแซมบลี้

การเขียนโปรแกรมด้วยภาษาแอสแซมบลี้สามารถเขียนได้ใน editor ใดๆ เช่น q หรือ edit ภาษาแอสแซมบลี้ที่เขียนควบคุมบอร์ด DSK เป็นภาษาที่ใช้เฉพาะกับ Chip TMS320c5x ของบริษัท TI โดยเฉพาะ คำสั่งต่างๆทั้งหมดสามารถดูได้ในหนังสือ TMS320C5x User's Guide

ก.2.1 คำสั่งในการคอมไพล์ และรันโปรแกรมโดยบอร์ด DSK

สามารถแสดงได้ดังรูปที่ ก.10



รูปที่ ก.10 ลำดับขั้นการคอมไพล์ และรันโปรแกรมโดยบอร์ด DSK

assembler เป็นการแปลงแฟ้มข้อมูลที่มีนามสกุล ASM ที่เขียนไว้มาคอมไพล์(Compile) เพื่อตรวจสอบว่าไฟล์นั้นมีข้อผิดพลาดหรือไม่ ถ้าไม่มีข้อผิดพลาดจะได้ไฟล์นามสกุล DSK เพื่อที่จะนำไปเป็นโปรแกรมที่ใช้รันบอร์ด DSK ต่อไป

debugger เป็นการนำเอาไฟล์นามสกุล DSK มารันเพื่อตรวจสอบสถานะหรือค่าภายใน Register,Accumulator และค่าในตัวแปรต่างๆหรือรันโปรแกรมทีเดียวทั้งหมดเลย การ debugger ทำได้โดยการรันโปรแกรม DSK5D.EXE และให้พิมพ์คำสั่งที่ DOS Prompt ดังนี้

เมื่อเข้าสู่โปรแกรม debugger แล้วการดึงข้อมูลเข้ามารันทำได้โดยพิมพ์ LD(Load DSK) บนบรรทัด input command.ในโปรแกรม debugger จะปรากฏหน้าต่างใหม่ขึ้นมาให้พิมพ์ ชื่อไฟล์ที่จะรันลงไปแล้วกด Enter 2 ครั้งโปรแกรมที่ต้องการรันก็จะถูกโหลดเข้ามา

จากนั้นในการรันโปรแกรมจะมีด้วยกัน 3 วิธี คือ

-พิมพ์ XR (eXecute Run) หรือ XG (eXecute Go) จะเป็นการรันโปรแกรมทั้งหมดที เดียว

-พิมพ์ XS (eXecute Single) หรือกด spec bar จะเป็นการรันแบบ single step หรือที ละบรรทัดคำสั่ง

-พิมพ์ XN (eXecute Number) เป็นการรันทีละหลายๆบรรทัดคำสั่ง โดยระบุที่ number เมื่อต้องการยกเลิกการรันโปรแกรมให้กดปุ่ม Esc เพื่อยกเลิกการรันโปรแกรม

นอกจากนี้แล้วยังมีคำสั่งที่จำเป็นในการ debugger อีก คือ

-พิมพ์ Q (Quit) เพื่อออกจากโปรแกรม debugger เมื่อกด Q แล้วให้กด Y เมื่อปรากฏหน้าต่างใหม่ขึ้นมา

-พิมพ์ R (Reset) เมื่อต้องการทดลองโหลด และรันโปรแกรมใหม่อีกครั้ง

-พิมพ์ WA (Wacth Addr) ใช้เพื่อดูค่าข้อมูลให้ตำแหน่งของ Data Memory ต่างๆโดย ระบุตำแหน่งที่ต้องการดูหลังพิมพ์ WA

นอกเหนือจากนี้ถ้านักศึกษาสนใจให้ดูได้ในหนังสือ TMS320c5x DSP Starter Kit User's Guide

ก.3 การเขียนโปรแกรมด้วยภาษาซี

การเขียนโปรแกรมควบคุมบอร์ด DSK สามารถใช้ภาษาซี เพื่อช่วยให้ง่ายต่อการออกแบบและพัฒนาโปรแกรมเพราะใช้งานง่ายกว่าภาษาแอสแซมบลี้ และยังสามารถเขียนโปรแกรมแอสแซมบลี้เป็นฟังชั่นๆหนึ่งของภาษาซีได้ เพราะบางคำสั่งในแอสแซมบลี้จะทำงานได้เร็วกว่าภาษาซี เช่นในส่วนของ Initializing AIC(Analog Interface Circuit) จะใช้ภาษาแอสแซมบลี้ ไฟล์ที่จำเป็นที่ใช้รวมกับการเขียนโปรแกรมด้วยภาษาซีมีดังนี้

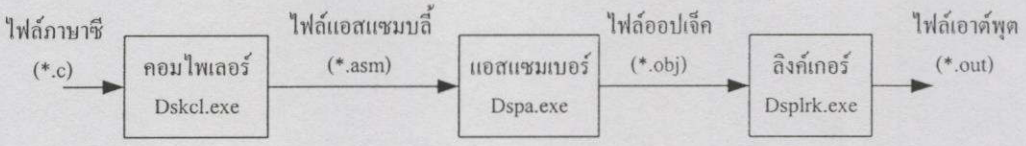
-ไฟล์ Aicinit.asm เป็นไฟล์ที่เขียนด้วยภาษาแอสแซมบลี้เพื่อใช้ในการ Initializing AIC

-ไฟล์ Vecs.asm เป็นไฟล์ที่ใช้กำหนดอินเตอร์รัปต์เวกเตอร์

-ไฟล์ Dsk.cmd เป็นไฟล์ที่ใช้กำหนดหน่วยความจำต่างๆเพื่อใช้งาน

-ไฟล์ Main ที่เป็นภาษาซี

ก.3.1 คำสั่งในการคอมไพล์ และรันโปรแกรมภาษาซีกับบอร์ด DSK



รูปที่ ก.11 ลำดับขั้นตอนการคอมไพล์โปรแกรมภาษาซี

จากรูปที่ ก.11 เป็นลำดับขั้นตอนการคอมไพล์โปรแกรมภาษาซี ซึ่งในส่วนต่างจะอธิบายได้ดังนี้

-คอมไพล์เลอร์ จะใช้โปรแกรม Dspcl.exe เป็นตัวคอมไพล์โปรแกรมภาษาซีที่เขียนมาถ้าไม่มีอะไรผิดพลาดจะได้ไฟล์นามสกุล ASM

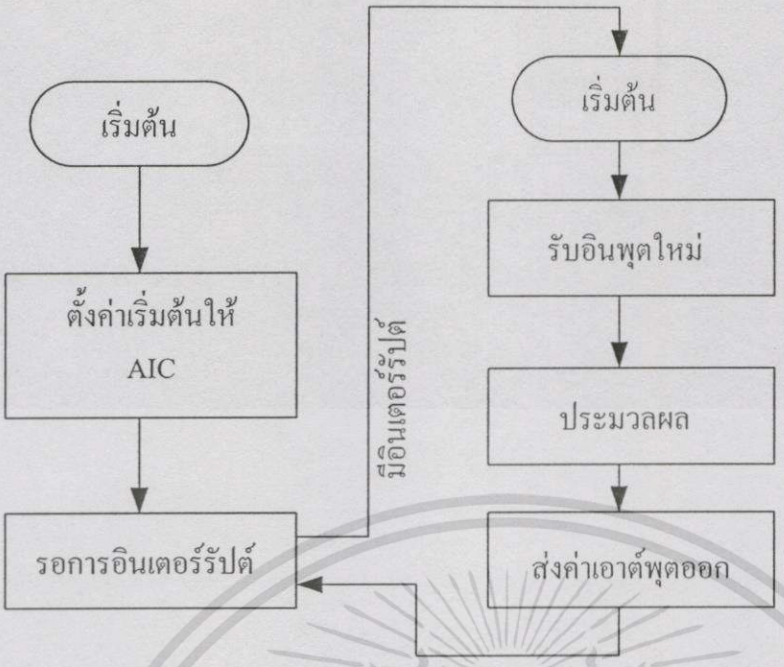
-แอสแซมเบอร์ จะใช้โปรแกรม Dspa.exe เป็นตัวแอสแซมเบอร์โปรแกรมที่คอมไพล์แล้วถ้าไม่มีอะไรผิดพลาดจะได้ไฟล์นามสกุล OBJ กรณีที่เราเขียนโปรแกรมย่อยด้วยภาษาแอสแซมบลี้แล้วการแอสแซมเบอร์ทำได้โดยการนำเอาไฟล์นามสกุล ASM มาทำได้เลย

-ลิงค์เกอร์ จะใช้โปรแกรม Dsplnk.exe ในการนำเอาไฟล์นามสกุล OBJ หรือไฟลอื่นๆที่ต้องการลิงค์มาทำการลิงค์เมื่อทำการลิงค์เสร็จจะได้ไฟล์โปรแกรมนามสกุล OUT เพื่อนำไปรันควบคุมบอร์ด DSK ต่อไป

ในการใช้งานจริงแล้วเราจะทำการเขียนคอมไพล์และลิงค์อยู่ใน batch file ทั้งหมดเพื่อที่จะทำการคอมไพเลอร์แอสแซมเบอร์ และลิงค์เกอร์โดยอัตโนมัติ

การ Debugger เมื่อการเชื่อมต่อสมบูรณ์จะทำการทดลองลงบน DSP Starter kit โดยที่ DOS promt จะใช้คำสั่ง dsk5d แล้วโปรแกรมจะมาอยู่ที่ส่วนของการ Debug ให้กดตัวอักษร L C ตามลำดับ ที่หน้าจอจะปรากฏหน้าต่างใหม่ขึ้นมาเพื่อให้พิมพ์ชื่อไฟล์ลงไป แล้ว ENTER 2 เพื่อเป็นการโปรแกรมลงบน DSP Starter Kit เมื่อเสร็จแล้วจะกลับมาสู่หน้าจอ Debug อีกครั้ง ให้กดตัวอักษร XR โปรแกรมจะเริ่มทำงานเหมือนกันการรันโปรแกรมด้วยภาษาแอสแซมบลี้

ขั้นตอนการทำงานทั้งหมดแสดงดังรูปที่ ก.12



รูปที่ ก.12 ผังการทำงานของโปรแกรม

ภาคผนวก ข.

โปรแกรมที่ใช้ในวิทยานิพนธ์

โปรแกรมที่ ข.1 อะแดปทีฟอัลกอริทึมแบบ LMP

```
function[a]=lmp(N,SNR,p,u,rho,at);
n=0:N-1;
w=acos(-at/2);
data=sin(n*w+pi/2)+sqrt(1/(2*10^(SNR/10)))*randn(1,N);
%data(1500)=data(1500)+10; (ใช้เมื่อต้องการบวกสัญญาณรบกวนอิมพัลส์)
x=zeros(1,N+2);
x(3:N+2)=data;
y=zeros(1,N+2);
a=-2*ones(1,N+2);
ey=zeros(1,N+2);
    for k=3:N+2;
        y(k)=x(k)+a(k)*x(k-1)+x(k-2)-rho*a(k)*y(k-1)-rho^2*y(k-2);
        ey(k)=x(k-1)-rho*y(k-1);
 if rem(p,2) = = 0
        update(k)=p*y(k)^(p-1)*ey(k);
    else
        update(k)=p*sign(y(k))*y(k)^(p-1)*ey(k);
 end;
        a(k+1)=a(k)-u*update(k);
end;
```

โปรแกรมที่ ข.2 อะแดปทีฟอัลกอริทึมแบบ QLMP

```
function[a]=lmp(N,SNR,p,u,rho,at);
n=0:N-1;
w=acos(-at/2);
```

```
data=sin(n*w+pi/2)+sqrt(1/(2*10^(SNR/10)))*randn(1,N);
%data(1500)=data(1500)+10; (ใช้เมื่อต้องการบวกสัญญาณรบกวนอิมพัลส์)
x=zeros(1,N+2);
x(3:N+2)=data;
y=zeros(1,N+2);
a=-2*ones(1,N+2);
ey=zeros(1,N+2);
    for k=3:N+2;
        y(k)=x(k)+a(k)*x(k-1)+x(k-2)-rho*a(k)*y(k-1)-rho^2*y(k-2);
        ey(k)=x(k-1)-rho*y(k-1);
 if rem(p,2) = = 0
        update(k)=p*y(k)^(p-1)*ey(k);
    else
        update(k)=p*sign(y(k))*y(k)^(p-1)*ey(k);
 end;
        a(k+1)=a(k)-u*sign(update(k));
end;
```

โปรแกรมที่ ข.3 อะแดปทีฟอัลกอริทึมแบบ NQLMP

```
function[a,update,Beta,mu]=nqlmp(N,SNR,p,u,rho,a,L);
n=0:N-1;g=3*1e-2;
w=acos(-a/2);
data=sin(n*w+pi/2)+sqrt(1/(2*10^(SNR/10)))*randn(1,N);
%data(1500)=data(1500)+10; (ใช้เมื่อต้องการบวกสัญญาณรบกวนอิมพัลส์)
x=zeros(1,N+2);
x(3:N+2)=data;
y=zeros(1,N+2);
a=zeros(1,N+2);
dy=zeros(1,N+2);
update=zeros(1,N+2);
Beta=zeros(1,N+2);
mu=0.002*ones(1,N+2);
```

```
for k=3:N+2;
        y(k)=x(k)+a(k)*x(k-1)+x(k-2)-rho*a(k)*y(k-1)-rho^2*y(k-2);
        dy(k)=x(k-1)-rho*y(k-1);
    if rem(p,2)==0
        update(k)=p*y(k)^(p-1)*dy(k);
    else
        update(k)=p*sign(y(k))*y(k)^(p-1)*dy(k);
end;
        Beta(k)=L*Beta(k-1)+(1-L)*update(k)^2;
  Beta(k)=Beta(k)+eps;
  mu(k)=u/(g+Beta(k));
  a(k+1)=a(k)-mu(k)*sign(update(k));
end;
```

โปรแกรมที่ ข.4 อะแดปทีฟอัลกอริทึมแบบ VSQLMP

```
function[a,u]=vsqlmp(N,SNR,p,rho,at,sigma,alpha,gamma);
n=0:N-1;
w=acos(-at/2);
data=sin(n*w+pi/2)+sqrt(1/(2*10^(SNR/10)))*randn(1,N);
%data(1500)=data(1500)+10; (ใช้เมื่อต้องการบวกสัญญาณรบกวนอิมพัลส์)
x=zeros(1,N+2);
x(3:N+2)=data;
y=zeros(1,N+2);
a=zeros(1,N+2);
u=0*ones(1,N+2);
dy=zeros(1,N+2);
P=zeros(1,N+2);
for k=3:N+2;
        y(k)=x(k)+a(k)*x(k-1)+x(k-2)-rho*a(k)*y(k-1)-rho^2*y(k-2);
        dy(k)=x(k-1)-rho*y(k-1);
    if rem(p,2)==0
            update(k)=p*y(k)^(p-1)*dy(k);
```

```
    else
            update(k)=p*sign(y(k))*y(k)^(p-1)*dy(k);
  end;
      P(k)=sigma*P(k-1)+(1-sigma)*y(k)*y(k);
      u(k+1)=alpha*u(k)+gamma*P(k)^2;
      a(k+1)=a(k)-u(k)*sign(update(k));
end;
```

โปรแกรมที่ ข.5 อะแดปทีฟอัลกอริทึมแบบ LMP สำหรับกำจัดสัญญาณซายน์ 50 Hz ออกจากสัญญาณ ECG

```
clear;
load y1.m; (โหลดไฟล์สัญญาณ ECG ที่วัดได้)
p=input('p=');
u=0.005;
rho=0.9;
N=length(y1');
n=0:N-1;
noise=cos((pi/3)*n);
data=y1(:,2)';
x=data+noise;
y=zeros(size(x));
a=-2*cos(pi/3)*ones(1,N);
dy=zeros(size(x));
    for k=3:N-1;
        y(k)=x(k)+a(k)*x(k-1)+x(k-2)-rho*a(k)*y(k-1)-rho^2*y(k-2);
        dy(k)=x(k-1)-rho*y(k-1);
    if rem(p,2)==0;
        update(k)=p*y(k)^(p-1)*dy(k);
    else
        update(k)=p*sign(y(k))*y(k)^(p-1)*dy(k);
end;
```

```
        a(k+1)=a(k)-u*update(k);
end;
```

โปรแกรมที่ ข.6 อะแดปทีฟอัลกอริทึมแบบ QLMP สำหรับกำจัดสัญญาณซายน์ 50 Hz ออกจากสัญญาณ ECG

```
clear;
load y1.m; (โหลดไฟล์สัญญาณ ECG ที่วัดได้)
p=input('p=');
u=0.005;
rho=0.9;
N=length(y1');
n=0:N-1;
noise=cos((pi/3)*n);
data=y1(:,2)';
x=data+noise;
y=zeros(size(x));
a=-2*cos(pi/3)*ones(1,N);
dy=zeros(size(x));
    for k=3:N-1;
        y(k)=x(k)+a(k)*x(k-1)+x(k-2)-rho*a(k)*y(k-1)-rho^2*y(k-2);
        dy(k)=x(k-1)-rho*y(k-1);
    if rem(p,2)==0;
        update(k)=p*y(k)^(p-1)*dy(k);
    else
        update(k)=p*sign(y(k))*y(k)^(p-1)*dy(k);
end;
        a(k+1)=a(k)-u*sign(update(k));
end;
```

โปรแกรมที่ ข.7 อะแดปทีฟอัลกอริทึมแบบ NQLMP สำหรับกำจัดสัญญาณซายน์ 50 Hz ออกจากสัญญาณ ECG

```
clear;
load y1.m;
p=input('p=');
L=0.98;
u=0.005;
rho=0.9;
N=length(y1');
n=0:N-1;
noise=cos((pi/3)*n);
data=0.5*y1(:,2)';
x=data+noise;
y=zeros(size(x));
a=-2*cos(pi/3)*ones(1,N);
dy=zeros(size(x));
g=1e-2;
Beta=zeros(1,N+2);
    for k=3:N-1;
        y(k)=x(k)+a(k)*x(k-1)+x(k-2)-rho*a(k)*y(k-1)-rho^2*y(k-2);
        dy(k)=x(k-1)-rho*y(k-1);
    if rem(p,2)==0;
        update(k)=p*y(k)^(p-1)*dy(k);
    else
        update(k)=p*sign(y(k))*y(k)^(p-1)*dy(k);
end;
        Beta(k)=L*Beta(k-1)+(1-L)*update(k)^2;
        a(k+1)=a(k)-(u/(g+Beta(k)))*sign(update(k));
end;
```

โปรแกรมที่ ข.8 อะแดปทีฟอัลกอริทึมแบบ VSQLMP สำหรับกำจัดสัญญาณซายน์ 50 Hz ออกจากสัญญาณ ECG

```
clear;
```

```
close all;
load y1.m;
p=input('p=');
sigma=0.99;
alpha=0.5;
gamma=0.001;
rho=0.9;
N=length(y1);
n=0:N-1;
noise=cos((pi/3)*n);
data=0.5*y1(:,2)';
x(3:N+2)=data+noise;
y=zeros(1,N+2);
a=-2*cos(pi/3)*ones(1,N);
dy=zeros(1,N+2);
P=zeros(1,N+2);
u=0*ones(1,N+2);
for k=3:N+2;
    y(k)=x(k)+a(k)*x(k-1)+x(k-2)-rho*a(k)*y(k-1)-rho^2*y(k-2);
        dy(k)=x(k-1)-rho*y(k-1);
    if rem(p,2)==0
            update(k)=p*y(k)^(p-1)*dy(k);
    else
          update(k)=p*sign(y(k))*y(k)^(p-1)*dy(k);
  end;
     P(k)=sigma*P(k-1)+(1-sigma)*y(k)^2;
     u(k+1)=alpha*u(k)+gamma*P(k)^2;
     a(k+1)=a(k)-u(k)*sign(update(k));
end;
```

โปรแกรมที่ ข.9 โปรแกรมคำนวณหาค่า variance และ bias

```
NN=input('NN=');
```

```
N=300;
SNR=input('SNR(dB)=');
p=input('p=');
u=0.02;
rho=0.9;
at=-1;
aa=zeros(NN,N+3);
ab=zeros(NN,N+3);
ay=zeros(NN,N+2);
    for ii=1:NN;
     [a,y]=lmp(N,SNR,p,u,rho,at); (สำหรับอะแดปทีฟอัลกิรทึมอื่น ให้เปลี่ยนเป็นเรียกใช้งานตาม
                                   ชื่อที่ตั้งไว้ และการรับค่าข้างบนให้ใส่ให้ครบตามจำนวนตัว
                                   แปรที่มีอยู่ในแต่ละโปรแกรม)
     aa(ii,:)=a;
     ab(ii,:)=a.^2;
   ay(ii,:)=y.^2;
 end;
aav=sum(aa(:,:))/NN;
aat=sum(ab(:,:))/NN;
aay=sum(ay(:,:))/NN;
var=aat-aav.^2;
bias=at-aav;
```

โปรแกรมที่ ข.10 โปรแกรมคำนวณวิธีของ Batch

```
clear;
N=2^10;
n=0:N-1;
varr=input('varr=');
x=sin(n*pi/2+pi/2)+0*sqrt(varr)*randn(1,N);
p=input('p=');
rho=0.9;
```

```
a=-1.99:0.1:1.99;
Ja=zeros(size(a));
for i=1:length(a) ;
        num=[1 a(i) 1];
        den=[ 1 rho*a(i)  rho^2];
        y=filter(num,den,x);
      z=sum((abs(y)).^p);
        Ja(i)=z/N;
        end;
     Jadb=10*log10(Ja);
     plot(a,Jadb,'k');
```

โปรแกรมที่ ข.11 โปรแกรมอะแดปทีฟอัลกอริทึมแบบ LMP ที่ $p = 1$ ด้วยภาษา C สำหรับการทดลองที่เวลาจริง

```
#include"math.h"
extern void aicinit();
int *DRR = (int *) 0x20;
int *DXR = (int *) 0x21;
signed long ACC,ACC1;
int x,x0,x1,x2,y,y1,y2,ale;
int b0=4096,b1=0,b2=4096;
int mu=12,mup=24;                    /*mup=mu*p (p=Power=2)*/
int a1=0,a2,an0,an1,dy;              /*a1=rho*an0*/
int yp,yup,py,s,error;
int rho=3686,rho2=3318;  /*rho=0.9*/
main()
{
 y1=y2=0;
 x0=x1=x2=0;
 an1=an0=dy=yp=yup=py=error=0;
 aicinit();              /*  initialize A/D, D/A  */
```

```
 for(;;);              /*  infinite loop waiting for interupt  */
}
/***** A/D (Serial Port) Receive Interupt Service Routine ******/
void c_int5(void)
{
 x=*DRR;
 x0=x;
 b1=an0;
 a2=rho2;
 ACC1=rho*an0;
 a1=ACC1>>12;
 ACC=b0*x0+b1*x1+b2*x2;
 ACC=ACC-a1*y1-a2*y2;
 y=ACC>>12;
 ACC1=rho*y1;
 py=ACC1>>12;
 ACC1=(unsigned long)(x1*4096-py*4096);
 dy=ACC1>>12;/*differential of y with respected to an0*/
 if (y>0)
   s=4096;
 else
   s=-4096;
 ACC1 =mu*s;
 yp = ACC1>>12;
 ACC1 =yp*dy;
 yup = ACC1>>12;/*update function*/
 ACC1=(unsigned long)(4096*an0-yup*4096);
 an1=ACC1>>12;
 an0 = an1 ;
 y2 = y1;
 y1 = y;
 x2 = x1;
```

```
  x1 = x0;
  ACC=(signed long)(4096*x-4096*y);
  error=ACC>>12;
  ale=x-y;
  *DXR = ale&0xfffc;        /* send output y to D/A */
}
```

โปรแกรมที่ ข.12 โปรแกรมอะแดปทีฟอัลกอริทึมแบบ QLMP ที่ $p = 1$ ด้วยภาษา C สำหรับการทดลองที่เวลาจริง

```
#include"math.h"
extern void aicinit();
int *DRR = (int *) 0x20;
int *DXR = (int *) 0x21;
signed long ACC,ACC1;
int x,x0,x1,x2,y,y1,y2,ale;
int b0=4096,b1=0,b2=4096;
int mu=10,mup=20;                /*mup=mu*p (p=Power=2)*/
int a1=0,a2,an0,an1,dy;          /*a1=rho*an0*/
int yp,yup,py,s,step;
int rho=3686,rho2=3318;  /*rho=0.9*/
main()
{
 y1=y2=0;
 x0=x1=x2=0;
 an1=an0=dy=yp=yup=py=0;
 aicinit();              /*  initialize A/D, D/A  */
 for(;;);                /*  infinite loop waiting for interupt  */
}
/***** A/D (Serial Port) Receive Interupt Service Routine ******/
void c_int5(void)
{
```

```
x=*DRR;
x0=x;
b1=an0;
a2=rho2;
ACC1=rho*an0;
a1=ACC1>>12;
ACC=b0*x0+b1*x1+b2*x2;
ACC=ACC-a1*y1-a2*y2;
y=ACC>>12;
ACC1=rho*y1;
py=ACC1>>12;
ACC1=(unsigned long)(x1*4096-py*4096);
dy=ACC1>>12;/*differential of y with respected to an0*/
if (y>0)
  s=4096;
else
  s=-4096;
ACC1 =s*dy;
yup = ACC1>>12;/*update function*/
if (yup>0)
  step=mu;
else
  step=-mu;
ACC1=(unsigned long)(an0*4096-step*4096);
an1=ACC1>>12;
an0 = an1 ;
y2 = y1;
y1 = y;
x2 = x1;
x1 = x0;
ale=x-y;
*DXR =ale&0xfffc;        /* send output y to D/A */
```

```
}
```

โปรแกรมที่ ข.13 โปรแกรมอะแดปทีฟอัลกอริทึมแบบ NQLMP ที่ $p = 1$ ด้วยภาษา C สำหรับการทดลองที่เวลาจริง

```
#include"math.h"
extern void aicinit();
int *DRR = (int *) 0x20;
int *DXR = (int *) 0x21;
signed long ACC,ACC1;
int x,x0,x1,x2,y,y1,y2;
int b0=4096,b1=0,b2=4096;
int mu=8000,co=20; /* mu=1.4648 */
int a1=0,a2,an0,an1,dy;  /*a1=rho*an0*/
int yp,yup,py,s,ss;
int rho=3686,rho2=3318;  /*rho=0.9*/
int en,en1,enn,dlamda=82,lamda=4014;/*lamda=0.98*/
int aa,bb,cc,dd,beta;
main()
{
 y1=y2=0;
 x0=x1=x2=0;
 an1=an0=dy=yp=yup=py=0;
 en=20,en1=enn=beta=0;
 aa=bb=cc=dd=0;
 aicinit();              /*  initialize A/D, D/A  */
 for(;;);                /*  infinite loop waiting for interupt  */
}
/***** A/D (Serial Port) Receive Interupt Service Routine ******/
void c_int5(void)
{
```

```
x=*DRR;
x0=x;
b1=an0;
a2=rho2;
ACC1=rho*an0;
a1=ACC1>>12;
ACC=(signed long)(b0*x0+b1*x1+b2*x2);
ACC=ACC-(signed long)(a1*y1+a2*y2);
y=ACC>>12;
ACC1=rho*y1;
py=ACC1>>12;
ACC1=(signed long)(x1*4096-py*4096);
dy=ACC1>>12;
if (y>0)
  s=4096;
else
  s=-4096;
ACC1=s*dy;
yup = ACC1>>12;/*update function*/
if (yup>0)
  ss=mu;
else
  ss=-mu;
ACC=yup*yup;
aa=ACC>>12;
ACC=dlamda*aa;
bb=ACC>>12;
ACC=lamda*en1;
cc=ACC>>12;
ACC=(signed long)(4096*cc+4096*bb);
en=ACC>>12;
ACC=(signed long)(4096*en+co*4096);
```

```
  enn=ACC>>12;
  beta = ss/enn;
  ACC1=(signed long)(4096*an0-beta*4096);
  an1=ACC1>>12;
  an0= an1 ;
  y2 = y1;
  y1 = y;
  x2 = x1;
  x1 = x0;
  en1=en;
  *DXR =an0&0xfffc;        /* send output y to D/A */
}
```

โปรแกรมที่ ข.14 โปรแกรมอะแดปทีฟอัลกอริทึมแบบ VSQLMP ที่ $p = 1$ ด้วยภาษา C สำหรับการทดลองที่เวลาจริง

```
#include"math.h"
extern void aicinit();
int *DRR = (int *) 0x20;
int *DXR = (int *) 0x21;
signed long ACC,ACC1;
int x,x0,x1,x2,y,y1,y2,p,p1;
int sigma=4055,alpha=3973,gamma=4; /*sigma=0.99,alpha=0.97,gamma=0.001*/
int dsigmma=41;                    /*(1-sigma)*/
int b0=4096,b1=0,b2=4096;
int vmu,vmu1;                  /*variable mu*/
int a1=0,a2,an0,an1,dy;          /*a1=rho*an0*/
int yp,yup,py;
int rho=3686,rho2=3318;          /*rho=0.9*/
int aa,bb,cc,dd,ee,yy,s,ss;
main()
{
```

```
 aa=bb=cc=dd=ee=p=p1=yy=0;
 y1=y2=0,vmu=287,vmu1=0;        /* initial of mu=0.07*/
 x0=x1=x2=0;
 an1=an0=dy=yp=yup=py=0;
 aicinit();                     /*  initialize A/D, D/A  */
 for(;;);                       /*  infinite loop waiting for interupt  */
}
/***** A/D (Serial Port) Receive Interupt Service Routine ******/
void c_int5(void)
{
 x=*DRR;
 x0=x;
 b1=an0;
 a2=rho2;
 ACC1=rho*an0;
 a1=ACC1>>12;
 /********************* output equation *********************/
 ACC=b0*x0+b1*x1+b2*x2;
 ACC=ACC-a1*y1-a2*y2;
 y=ACC>>12;
 ACC1=rho*y1;
 py=ACC1>>12;
 /********************* differential of y with respected to a***/
 ACC1=(unsigned long)(x1*4096-py*4096);
 dy=ACC1>>12;
 /********************* make energy of output signal to updated mu**/
 ACC1=y*y;
 yy=ACC1>>12;
 ACC1=dsigmma*yy;
 aa=ACC1>>12;
 ACC1=sigma*p1;
 bb=ACC1>>12;
```

```
ACC1=(unsigned long)(4096*bb+aa*4096);
p=ACC1>>12; /*output energy*/
ACC1=p*p;
cc=ACC1>>12;
ACC1=gamma*cc;
dd=ACC1>>12;
ACC1=alpha*vmu;
ee=ACC1>>12;
/************ time varying step size **************/
ACC1=(unsigned long)(4096*ee+4096*dd);
vmu1=ACC1>>12;
vmu=vmu1;
/************ checking the sign of output signal ****************/
if (y>0)
   s=4096;
else
   s=-4096;
/************** defined update function ************/
ACC1 =s*dy;
yup = ACC1>>12;
/************** quantizing gradient ***************/
if (yup>0)
   ss=4096;
else
   ss=-4096;
/************** coefficient adaptating ************/
ACC1=(unsigned long)(4096*an0-vmu*ss);
an1=ACC1>>12;
an0 = an1 ;
y2 = y1;
y1 = y;
x2 = x1;
```

```
x1 = x0;
p1 = p;
/******* send output y to D/A ***********/
*DXR = an0&0xfffc;
}
```

โปรแกรมที่ ข.15 โปรแกรมสำหรับ run เพื่อทำงาน

```
@echo off
REM set env. variables for library linking
set A_DIR=c:\dsk
set C_DIR=c:\dsk
REM -g to enable C debugging
REM -q for quiet mode
REM -s for listing C and assembly together in .asm file
REM -v50 for TMS320C50 device
dspcl -o3 -pk -q -s -msx -g -v50 %1.c > %1.err
type %1.err
REM -c use ROM initialization
REM -v0 to generate version 0 COFF file for using with dsk
REM dsk.cmd is command file to make the program fit into dsk memory area.
cd c:\dsk
dsplnk -c -v0 aicinit.obj vecs.obj %1.obj -o %1.out -m %1.map dsk.cmd -l rts50.lib
echo      ****************************************
echo      * Press "E" If you want return to Editor *
echo      *           OR .           *
echo      * Press "R" you want to run dsk debuger  *
echo      ****************************************
choice /c:er
if errorlevel 2 goto rundsk
if errorlevel 1 goto editf
:rundsk
```

```
dsk5d c1
goto endf
:editf
edit %1.c
:endf
```

โปรแกรมที่ ข.16 โปรแกรมสำหรับตั้งค่าเริ่มต้นให้ตัวประมวล TMS320C50

```
        .file "aicinit.asm"
        .globl _aicinit
        .mmregs
_aicinit:
        POPD   *+
        SAR    AR0,*+
        SAR    AR1,*
        LARK   AR0,1
        LAR    AR0,*0+,AR2
TA      .word  16
RA      .word  16
TB      .word  40
RB      .word  40
AIC_CTR .word  18h
        SETC   INTM
        LDP    #0
        OPL    #0834h,PMST
        LACC   #0
        SAMM   CWSR
        SAMM   PDWSR
        SETC   SXM          ; SXM MUST BE SET
        SPLK   #022h,IMR    ; This turns on receive interrupt only
        SPLK   #20h,TCR     ; To generate 10 MHz from Tout
      SPLK   #01h,PRD       ; for AIC master clock
```

```
MAR   *,AR0
LACC  #0008h        ; Non continuous mode
SACL  SPC           ; FSX as input
LACC  #00c8h        ; 16 bit words
SACL  SPC
LACC  #080h         ; Pulse AIC reset by setting it low
SACH  DXR
SACL  GREG
LAR   AR0,#0FFFFh
RPT   #10000        ; and taking it high after 10000 cycles
LACC  *,0,AR0       ; (.5ms at 50ns)
SACH  GREG
;-----------------------
LDP   #TA           ;
SETC  SXM           ;
LACC  TA,9          ; Initialize TA and RA register
ADD   RA,2          ;
CALL  AIC_2ND       ;
;-----------------------
LDP   #TB
LACC  TB,9          ; Initialize TB and RB register
ADD   RB,2          ;
ADD   #02h          ;
CALL  AIC_2ND       ;
;-----------------------
LDP   #AIC_CTR
LACC  AIC_CTR,2     ; Initialize control register
ADD   #03h          ;
CALL  AIC_2ND       ;
SPLK  #12h,IMR
CLRC  OVM
SPM   0
```

```
        CLRC   INTM
        LARP   AR1
        SBRK   2
        LAR    AR0,*-
        PSHD   *
        RET
AIC_2ND:
        LDP    #0          ; Data page point is 0  (MM regs)
        SACH   DXR         ; send ACChi 00
        CLRC   INTM        ; enable interrupts
        IDLE               ; wait for interrupt
        ADD    #6h,15      ; 0000 0000 0000 0011 XXXX XXXX XXXX XXXX b
        SACH   DXR         ; send ACChi to initiate secondary protocol
        IDLE               ; wait for interrupt
        SACL   DXR         ; send the T register data
        IDLE               ; wait for interrupt
        LACL   #0          ; clear ACClo
        SACL   DXR         ; send another to make sure 1st word got sent
        IDLE               ; wait for interrupt
        SETC   INTM
        RET                ;
        .END
```

สำหรับผู้ที่สนใจและต้องการโปรแกรมทั้งหมดสามารถติดต่อได้ที่ rachu@mut.ac.th หรือ ajrboy@hotmail.com หรือ เบอร์โทร 9883655 ต่อ 220 อาจารย์ราชู พันธ์ฉลาด เพราะโปรแกรมที่ได้นำมาลงไว้นี้เป็นเพียงโปรแกรมที่สำคัญเท่านั้น

ภาคผนวก ค.

การเผยแพร่งานวิจัย

[1] ราชู พันธ์ฉลาด และคณะ. "อะแดปทีฟอัลกอริทึมสำหรับการดีเทกต์สัญญาณไซน์คลื่นเดี่ยวด้วยวิธีการนอร์มอลไลซ์ Stochastic Gradient." Mahanakorn Engineering Transaction. vol. 1, no. 2 March-June 1998.

[2] ราชู พันธ์ฉลาด และคณะ. "อะแดปทีฟ IIR แบนด์พาสฟิลเตอร์สำหรับการตรวจวัดสัญญาณไซน์ด้วยวิธีการนอร์มอลไลซ์เกรเดียนต์." การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 21 มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี 12 – 13 พฤศจิกายน 2541.

[3] ราชู พันธ์ฉลาด และคณะ. "อะแดปทีฟ IIR นอตช์ฟิลเตอร์บนพื้นฐานการพิจารณาค่าผิดพลาดเฉลี่ยกำลัง p น้อยที่สุดโดยใช้วิธีการควอนไตซ์เกรเดียนต์." การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 21 มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี 12 – 13 พฤศจิกายน 2541.

[4] ราชู พันธ์ฉลาด และคณะ. "อะแดปทีฟ IIR แบนด์พาสฟิลเตอร์สำหรับการตรวจจับสัญญาณไซน์ที่เวลาจริงโดยใช้ TMS320C50." การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 21 มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี 12 – 13 พฤศจิกายน 2541.

[5] R. Punchalard *et. al.* "Simplify Adaptive IIR Notch Filter Based on Least Mean p-Power Error Criterion." Proc. IEEE APCCAS'98 1998.

[6] ราชู พันธ์ฉลาด และคณะ. "อะแดปทีฟ IIR นอตช์ฟิลเตอร์สำหรับการตรวจวัดสัญญาณไซน์ด้วยวิธีการควอนไตซ์นอร์มอลไลซ์เกรเดียนต์." การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 22 มหาวิทยาลัยเกษตรศาสตร์ 2-3 ธันวาคม 2542.

[7] ราชู พันธ์ฉลาด และ คณะ. "การลดสัญญาณรบกวนเกรเดียนต์ในควอนไตซ์เกรเดียนต์อัลกอริทึม." Mahanakorn Engineering Transaction. vol., 3 No. 1, Jan-April 2000. pp. 48-52.

ประวัติผู้เขียน

นายราชู พันธ์ฉลาด เกิดเมื่อวันที่ 28 พฤษภาคม 2516 ที่จังหวัดภูเก็ต สำเร็จการศึกษาอุตสาหกรรมศาสตรบัณฑิต สาขาเทคโนโลยีโทรคมนาคม จากภาควิชาเทคนิคอุตสาหกรรม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปี 2540 ปัจจุบันทำงานในตำแหน่ง อาจารย์ประจำ ที่มหาวิทยาลัยเทคโนโลยีมหานคร ภาควิชาวิศวกรรมโทรคมนาคม คณะวิศวกรรมศาสตร์