ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ

เรื่อง การลดทอนสัญญาณรบกวนบนคลื่นไฟฟ้าหัวใจโดยใช้ตัวกรองเชิงเลขแบบปรับตัวได้

โดย นายชลิต จิตต์สวัสดิ์ไทย

ได้รับอนุมัติให้นับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิทยาศาสตรมหาบัณฑิต สาขาวิชาอุปกรณ์การแพทย์

_____คณบดีบัณฑิตวิทยาลัย

(อาจารย์ ดร.มงคล หวังสถิตย์วงษ์)

21 พฤษภาคม 2550

คณะกรรมการสอบวิทยานิพนธ์

_____ประธานกรรมการ

(รองศาสตราจารย์สุรพันธ์ ยิ้มมั่น)

_____กรรมการ

(รองศาสตราจารย์จิระศักดิ์ ชาญวุฒิธรรม)

_____กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.พิพัฒน์ พรหมมี)

การลดทอนสัญญาณรบกวนบนคลื่นไฟฟ้าหัวใจโดยใช้ตัวกรองเชิงเลขแบบปรับตัวได้

นายชลิต จิตต์สวัสดิ์ไทย

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิทยาศาสตรมหาบัณฑิต
สาขาวิชาอุปกรณ์การแพทย์ ภาควิชาฟิสิกส์อุตสาหกรรมและอุปกรณ์การแพทย์
บัณฑิตวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ
ปีการศึกษา 2549

ชื่อ : นายชลิต จิตต์สวัสดิ์ไทย
ชื่อวิทยานิพนธ์ : การลดทอนสัญญาณรบกวนบนคลื่นไฟฟ้าหัวใจโดยใช้ตัวกรองเชิงเลขแบบปรับตัวได้
สาขาวิชา : อุปกรณ์การแพทย์
สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ
ที่ปรึกษาวิทยานิพนธ์ : รองศาสตราจารย์สุรพันธ์ ยิ้มมั่น
ปีการศึกษา : 2549

บทคัดย่อ

ในปัจจุบันระบบการสื่อสารโทรคมนาคม และระบบควบคุมได้มีการพัฒนาไปอย่างมากทั้งงานด้านอุตสาหกรรมและงานด้านการแพทย์แต่ในการพัฒนายังมีข้อผิดพลาดที่จะต้องแก้ไขโดยเฉพาะงานด้านการแพทย์ เช่น การแสดงผลของเครื่องวัดคลื่นไฟฟ้าหัวใจ (ECG) ได้มีสัญญาณรบกวนเข้ามา ดังนั้นวิทยานิพนธ์ฉบับนี้จึงได้ศึกษาเกี่ยวกับการลดทอนคลื่นไฟฟ้าหัวใจ โดยอาศัยโปรแกรมตัวกรองเชิงเลขแบบปรับตัวได้ซึ่งเป็นตัวลดทอนสัญญาณรบกวนแบบปรับตัวเองจะใช้สมการอัลกอริซึมของลิสมันสแควร์เออเรอร์ (LMS)ในการลดทอนจากโครงสร้างแบบ FIR โดยจะลดทอนคลื่นสัญญาณรบกวนในคลื่นไฟฟ้าหัวใจจำลองในย่านความถี่ 50 Hz แสดงผลผ่านบอร์ด TMS3210C3x อาศัยตัวโปรแกรมภาษาซีเป็นตัวประมาณผล

(วิทยานิพนธ์นี้มีจำนวนทั้งสิ้น 113 หน้า)

คำสำคัญ : ตัวกรองเชิงเลขแบบปรับตัวได้, สัญญาณคลื่นไฟฟ้าหัวใจ

_____อาจารย์ที่ปรึกษาวิทยานิพนธ์

Name : Mr.Chalit Jittsawadthai

Thesis Title : Noise Eliminate in ECG using Adaptive Filter

Major Field : Biomedical Instrument

King Mongkut's Institute of Technology North of Bangkok

Major Thesis Advisor : Associatc Professor Surapan Yimman

Academic Year : 2006

Abstract

According to the communication and controlling systems have been developed rapidly both of industrial and medical parts. There are some noises and errors that need to improve especially ECG measuring equipment, medical equipment and other. This thesis presents the studied of noise reduction by Digital Adaptive Filter program. Algorithm of Least Mean Square is Used based on FIR structure for 50 Hz noise reduction from electrocardiogram (ECG). The experimental results have been carried out by TMS320C3X board using C Programing language.

(Total 113 Pages)

Key Words : Adaptive Filter, ECG

_____Advisor

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้เกิดจากองค์ความรู้ที่ได้ศึกษามาตั้งแต่หลักสูตรปริญญาวิทยาศาสตร์บัณฑิตจนถึงระดับปริญญาวิทยาศาสตร์มหาบัณฑิตและที่สำคัญผู้ที่อยู่เบื้องหลังแห่งความสำเร็จหลายๆท่านซึ่งจะต้องกล่าวขอบคุณและจารึกไว้ในวิทยานิพนธ์เล่มนี้คือบิดานาวาอากาศเอกชูศักดิ์ จิตต์สวัสดิ์ไทย มารดานางศฤงคาร จิตต์สวัสดิ์ไทย พี่สาวนางศิริฉัตร กลิ่นคล้าย พี่ชายพันจ่าอากาศเอกชีละ จิตต์สวัสดิ์ไทย ซึ่งเป็นบุคลภายในครอบครัวผู้ให้การสนับสนุนทางด้านทุนทรัพย์และการพิมพ์เอกสารส่วนบุคลอื่นที่จะต้องกล่าวขอบคุณเป็นอย่างมากคือรองศาสตราจารย์สุรพันธ์ ยิ้มมั่น อาจารย์พยุง เดชอยู่ และบุคลอีกท่านหนึ่งที่จะไม่กล่าวถึง ณ ที่นี้ไม่ได้เลยคืออาจารย์สุกัญญา แพรสมบูรณ์(น้องสุ)ผู้เสียสละเวลาเป็นธุระช่วยเหลืองานในด้านต่างๆทำให้ผู้เขียนวิทยานิพนธ์ทำงานสะดวกยิ่งขึ้นและห้องทดลองปฏิบัติการเชิงเลขที่สนับสนุนเครื่องมือ, การพิมพ์เอกสารซึ่งเป็นอุปกรณ์ของภาควิชาฟิสิกส์อุตสาหกรรมและอุปกรณ์การแพทย์

สุดท้ายนี้ผู้เขียนวิทยานิพนธ์ใคร่ขอกราบขอบพระคุณบุคคลต่างๆที่ได้กล่าวมา ณ ที่นี้และขออำนาจบารมีคุณพระศรีรัตนตรัยจนดลบันดาลให้บุคคลเหล่านี้จงมีชีวิตที่มีความสุขทั้งในด้านหน้าที่การงาน, ชีวิตครอบครัว, ชีวิตส่วนตัวและมีอนาคตที่ก้าวไกล หลังจากยื่นวิทยานิพนธ์ฉบับนี้ต่อทางสถาบันแล้วผู้เขียนวิทยานิพนธ์ก็คงเก็บความรู้สึกปราถนาดีนี้เอาไว้ตลอด

ชลิต จิตต์สวัสดิ์ไทย

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

บทที่ 1
บทนำ

1. รายละเอียดโครงงาน

1.1 ความสำคัญและที่มาของโครงงาน

ระบบการสื่อสารโทรคมนาคมได้มีการพัฒนาไปอย่างก้าวไกลแต่การพัฒนายังมีข้อบกพร่อง
ที่ต้องแก้ไขในทางการแพทย์เครื่องวัดคลื่นไฟฟ้าหัวใจมักมีสัญญาณปะปน 50 Hz ที่เกิดจากไฟ 220 โวลต์และความถี่อื่นๆที่อยู่ในสภาวะแวดล้อมการทำงานในที่นั้นๆ ดังนั้นปริญญานิพนธ์ฉบับนี้จึงได้ทำ
การทดลองลดทอนคลื่นสัญญาณปะปนจากคลื่นไฟฟ้าหัวใจจำลองโดยใช้ตัวกรองสัญญาณเชิงตัวเลข แบบปรับตัวเอง และใช้การประมวลผลผ่านบอร์ด TMS320C3X อาศัยภาษา C ในการประมวลผล

1.2 วัตถุประสงค์โครงงาน

1.2.1 ศึกษาคุณสมบัติโครงสร้างของสมการลีสท์มีนแสควร์เออเรอร์ (LMS)

1.2.2 ศึกษาคุณสมบัติรูปแบบของสมการ ตัวกรองสัญญาณเชิงเลขแบบปรับตัวเอง (Adaptive Digital Filter)

1.2.3 รู้จักถึงตัวประมวลผลสัญญาณดิจิตอล TMS320C3X

1.3 ขอบเขตของโครงงาน

1.3.1 ออกแบบตัวกรองสัญญาณเชิงตัวเลขแบบปรับตัวเอง โดยใช้โปรแกรม MatLab

1.3.2 จำลองการทดลองโดยใช้โปรแกรม MatLab

1.3.3 ทดลองจริงโดยใช้คลื่นไฟฟ้าหัวใจจำลอง ทดลองผ่านบอร์ด TMS320C3X ประมวลผลโดยใช้ภาษาซี

1.4 ระยะเวลาการดำเนินการ

ระยะเวลาตั้งแต่ เดือนกันยายน 2548 ถึง เมษายน 2549

1.5 วิธีการดำเนินโครงการ

การดำเนินโครงงานแบ่งเป็นขั้นตอนดังนี้

1.5.1 ศึกษาและรวบรวมข้อมูลเกี่ยวกับตัวกรองสัญญาณเชิงตัวเลขแบบปรับตัวเอง

1.5.2 ศึกษาวิธีการ และภาษาใช้ในการเขียนโปรแกรม

1.5.3 เขียนโปรแกรม และข้อมูลต่างๆ

1.5.4 ทดลองโปรแกรมและปรับปรุงแก้ไข

1.5.5 สรุปโครงงาน

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1.6.1 สามารถนำรูปแบบของสมการ ตัวกรองสัญญาณเชิงเลขแบบปรับตัวเอง (Adaptive Digital Filter) มาประยุกต์ใช้งานกับเครื่องวัดคลื่นไฟฟ้าหัวใจได้

1.6.2 สามารถเขียนโปรแกรมภาษาซีนำมาประยุกต์ได้

1.6.3 สามารถประยุกต์ใช้กับสัญญาณปะปนในรูปแบบต่างๆ ได้

บทที่ 2
ตัวกรองสัญญาณเชิงตัวเลขแบบปรับตัวเอง
(Adaptive Digital Filter)

ในการศึกษาและวิจัยเราจะพิจารณาตัวกรองสัญญาณแบบเอฟไอออาร์เท่านั้น แต่เราจะกล่าวถึงโครงสร้างของไอไอออาร์ด้วยพอสังเขป ในการพิจารณาเฉพาะเอฟไอออาร์นั้น เพื่อต้องการหลีกเลี่ยงปัญหาเกี่ยวกับความเสถียรภาพของตัวกรองสัญญาณ เนื่องจากตัวกรองสัญญาณแบบเอฟไอออาร์ มีเฉพาะค่าซีโร่ที่สามารถปรับค่าได้ อย่างไรก็ตามเสถียรภาพของตัวกรองสัญญาณยังขึ้นอยู่กับอัลกอริธึมซึ่งใช้ในการปรับ ค่าสัมประสิทธิ์ของตัวกรองสัญญาณด้วยโดยจะกล่าวต่อไป

2.1 อัลกอริธึมปรับตัวเองแบบลีสท์มีนแสควร์ (Least Mean Square)

อัลกอริธึมปรับตัวเองแบบลีสท์มีนแสควร์ หรือ แอลเอ็มเอส (LMS) เป็นอัลกอริธึมซึ่งใช้โครงสร้างแบบไดเรกท์ และใช้สัญญาณผิดพลาด e(n) มาคำนวณเพื่อปรับค่าสัมประสิทธิ์ของตัวกรองสัญญาณโดยใช้เงื่อนไขการลดค่ามีนแสควร์เออเรอร์ (Mean Square Error) หรือ MSE ให้มีค่าต่ำสุดแสดงได้ดังนี้

$$\beta = E[e^2(n)] \tag{2-1}$$

E [ ] แสดงถึงเอ็กเปกเตชั่นอ็อปเปอร์เรเตอร์(ExpectationOperator) เป็นค่าความคาดหวัง และ e(n) คือค่าผลต่างของสัญญาณที่ต้องการ คือ d(n) กับสัญญาณเอาท์พุทของตัวกรอง คือ y(n) ดังนี้

$$e(n) = d(n) - y(n) \tag{2-2}$$

หรือ $$e(n) = d(n) - w^t(n)x(n) \tag{2-3}$$

ซึ่งค่าของ w(n) คือสัมประสิทธิ์ตัวกรองที่เวลาใด ๆ เรียกว่า เวทเวกเตอร์ (Weight Vector)

$$w = \begin{bmatrix} w(0) \\ w(1) \\ \vdots \\ w(L-1) \end{bmatrix}$$

และค่าของ

$$x(n) = \begin{bmatrix} Xn \\ Xn-1 \\ \vdots \\ Xn-(L-1) \end{bmatrix}$$

ซึ่งค่าของ $w^t(n)$ คือการทรานโพสท์ของเวกเตอร์ w(n)

จากสมการ 2-3 สามารถเขียนสมการ 2-1 ใหม่ได้ดังสมการ 2-4

จัดรูปสมการ

$$e^2(n) = (d(n) - y(n))^2$$

$$e^2(n) = (d(n) - w^t(n)x(n))^2$$

$$e^2(n) = d^2(n) - 2d(n)\, w^t(n)x(n) + \{w^t(n)x(n)\}^2$$

ดังนั้นจะได้

$$\beta = E[e^2(n)]$$

$$E[e^2(n)] = E[d^2(n)] - 2E[d(n) - w^t(n)x(n)] + [\{w^t(n)x(n)\}]^2$$

เมื่อจัดรูปของสมการใหม่จะได้

$$\beta = E\left[d^2(n)\right] - 2\underline{w}^t(n)\underline{P} + \underline{w}^t(n)R\underline{w}(n) \qquad (2\text{-}4)$$

โดยที่ $R = E\left[\underline{x}(n)\underline{x}^t(n)\right]$ เป็น Autocorretation Matrix ขนาด $N \times N$ แสดงความสัมพันธ์ของแต่ละค่าสัญญาณสุ่ม (Sample) ของอินพุท x(n) และ $\underline{P} = E\left[d(n)x(n)\right]$ เป็น Crosscorrelation Vector ขนาด $N \times 1$ แสดงความสัมพันธ์กันระหว่างสัญญาณที่ต้องการ d(n) กับสัญญาณอินพุท x(n)

ในการหาสัมประสิทธิ์ที่เป็นคำตอบของสมการ 2-4 คือ

$\underline{w} = \left[w_0\ w_1\ \ldots w_{n-1}\right]^t$ จะพิจารณาโดยการกำหนดค่าแกรเดี้ยนของ MSE ให้มีค่าต่ำสุด แล้วทำการแก้สมการดังต่อไปนี้

$$\nabla(\varepsilon) = \frac{\partial\varepsilon}{\partial\underline{w}(n)} = 0 \qquad (2\text{-}5)$$

ในสมการ 2-5 เรียกว่าสมการพื้นผิวของค่าแกรเดี้ยน (Gradient of MSE Performance Surface) ซึ่งสามารถจัดสมการในรูปของค่าสัมประสิทธิ์ คำตอบ w ได้มีชื่อเรียกว่าสมการ นอร์มอล (Normal Equation) ดังนี้

$$RW = P \qquad (2\text{-}6)$$

จากสมการ 2-3 สามารถหาค่าสัมประสิทธิ์ $\underline{w}^*$ ได้ใน 2 ลักษณะ คือ แบบทีละชุด (Block-By-Block) โดยการประมาณค่า R และ $\underline{P}$ จากอินพุทซึ่งแบ่งเป็นส่วนย่อย ๆ แล้วจึงหาคำตอบต่อไปนี้

$$W = R^{-1}\underline{P} \qquad (2\text{-}7)$$

และแบบทีละค่า (Sample-By-Sample) ซึ่งนิยมใช้กันในทางปฏิบัติมีลักษณะการทำงานเป็นแบบตามเวลาจริง (Real Time) หลีกเลี่ยงการคำนวณที่ยุ่งยากในการหา R^{-1} และ $\underline{P}$

โดยการทำให้อยู่ในรูปแบบของการทำซ้ำ (Iteration) ซึ่งอาศัยหลักการประมาณค่าแกรเดี้ยน เช่นเดียวกับวิธีสติฟเดชเซ้นท์ (Steepest Descent) ดังสมการต่อไปนี้

$$W(n+1) = W(n) - \mu\nabla(n) \qquad (2\text{-}8)$$

สังเกตได้ว่าค่า $\underline{W}(n+1)$ จะปรับค่าเป็นสัดส่วนตามค่าลบของแกรเดี้ยนของ MsE, ค่า μ ในสมการ คือขนาดขั้น (Stop Size) เป็นค่าคงที่มีผลต่อเสถียรภาพ และอัตราการลู่เข้าของอัลกอริธึมในการประมาณค่าแกรเดี้ยน $\underline{\nabla}(n)$ สำหรับอัลกอริธึมแบบแอลเอ็มเอส ประมาณโดยการกำหนดให้ค่า MsE, (ε) มีค่าเท่ากับ $e^2(n)$ ซึ่งแสดงค่าประมาณแกรเดี้ยนได้ดังนี้

$$\nabla(n) = \frac{\partial\left[e^2(n)\right]}{\partial \underline{W}(n)} = -2e(n)X(n) \qquad (2\text{-}9)$$

และเมื่อแทนค่าประมาณแกรเดี้ยนที่เวลาใดๆในสมการ(2.8) จะได้อัลกอริธึม Widrow – Hoft LMS Algorithm ดังนี้

$$W(n+1) = W(n) + 2\mu e(n)X(n) \qquad (2\text{-}10)$$

ในการกำหนดค่าเริ่มต้นสำหรับสัมประสิทธิ์เรทเวกเตอร์ (Weight Vector) W(0) สามารถกำหนดเป็นค่าใด ๆ ส่วนค่าที่ต้องกำหนดให้เหมาะสมคือ ค่าขนาดขั้น (Step Size) μ เนื่องจากมีผลต่อความเสถียรและความเร็วในการลู่เข้าซึ่งมีผลต่อความสามารถในการติดตามสัญญาณของตัวกรองด้วย จากความสัมพันธ์ตามสมการจะสามารถกำหนดค่าขนาดขั้น μ ได้ดังนี้

$$0 < \mu < \frac{1}{\lambda_{max}} \qquad (2\text{-}11)$$

โดยที่ λ_{max} เป็นค่าเอ็งเก็น (Eigen Value) ที่ใหญ่สุดของแมทริกซ์ R และค่า λmax สามารถหาขอบเขตได้จาก

$$\lambda\text{max} < \text{Tr}\,[R] = \Sigma(\text{Diagonal Elements of R}) \qquad (2\text{-}12)$$

โดยที่ Tr [ ] คือ เทรส (trace) ของแมทริกซ์ใด ๆ

ความเร็วในการลู่เข้า (Speed of Convergence) ของสัมประสิทธิ์ตัวกรองสัญญาณ (Weight Vector) ซึ่งเป็นตัวกำหนดความสามารถในการติดตามสัญญาณของตัวกรองสัญญาณ โดยตัวกรองสัญญาณจะสามารถลู่เข้าได้เมื่อค่าความเร็วในการลู่เข้าที่ช้าที่สุดสามารถทำให้ตัวกรองสัญญาณลู่เข้าได้เท่านั้น ค่าคงที่เวลา (Time Constant) กรณีที่ช้าที่สุดแสดงได้ดังนี้

$$t = \frac{1}{\mu \lambda \min} \tag{2-13}$$

จากสมการข้างต้น แสดงให้เห็นว่าค่าคงที่เวลาสำหรับการลู่เข้าแปรผกผันกับค่าขนาดขั้น μ และค่าเอ็งเก็น (Eigen Valu)ของ Autocorrelation แมทริกซ์ซึ่งเวลาที่ช้าที่สุดจะถูกกำหนดโดยค่า λmin

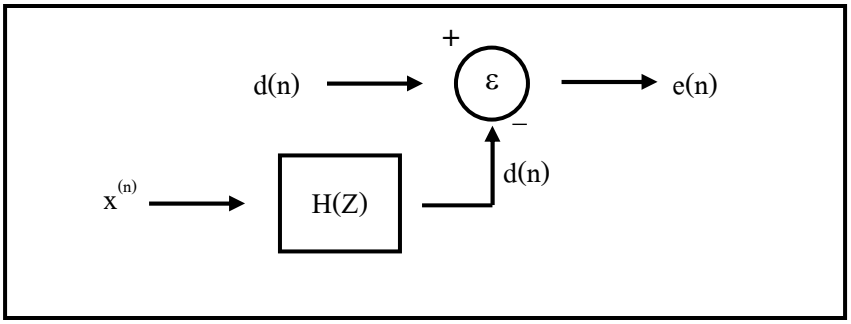
ผลการทำงานของการปรับตัวเองโดยใช้การประมาณค่าแกรเดี้ยน (Gradient) จะเกิดค่าผิดพลาด เนื่องจากการแปรเปลี่ยนในลักษณะสุ่มของสัมประสิทธิ์ตัวกรองสัญญาณจากค่า Optimum ในสภาวะคงตัว (Steady State) ความถูกต้องของสัมประสิทธิ์ในสภาวะคงตัวสามารถวัดได้ในรูปของค่าผิดพลาดซึ่งเกินจากค่าเฉลี่ยหรือ Excess Mean Square Error

(Excess MSE = E[ε - εmin]) และ กรณีอัลกอริธึมแลลแอลเอ็มเอส ค่า Excess MSE แสดงได้ดังนี้

$$\text{Excess MSE} = \mu \, \text{Tr}[R] \, \varepsilon\text{min} \tag{2-14}$$

โดยที่ εmin คือค่า MSE ต่ำสุดในสภาวะคงตัว จากสมการ (2.13) และ (2.14) สามารถใช้เป็นพื้นฐานในการเลือกขนาดขั้น μ ของอัลกอริธึมแบบแอลเอ็มเอส เพื่อกำหนดคุณสมบัติของอัลกอริธึมเกี่ยวกับอัตราการลู่เข้าและความเที่ยงตรง ของสัมประสิทธิ์เทียบกับค่าที่เหมาะ สมที่สุด (Optimum) ในสภาวะคงตัว

โดยที่เราสามารถสรุปขั้นตอนการทำงานของ LMS (Least Mean Square) ได้ดังนี้



ภาพที่ 2-1 แสดงรูปแบบขั้นตอนการทำงานของ LMS (Least Mean Square)

ขั้นที่ 1 สุ่มเลือกตัวเลขต่าง ๆ หาค่าของ $y(n) = w^t(n)x(n)$

ขั้นที่ 2 คำนวณตามรูปแบบสมการ Adaptive Filter $e(n) = d(n) - w^t(n)x(n)$

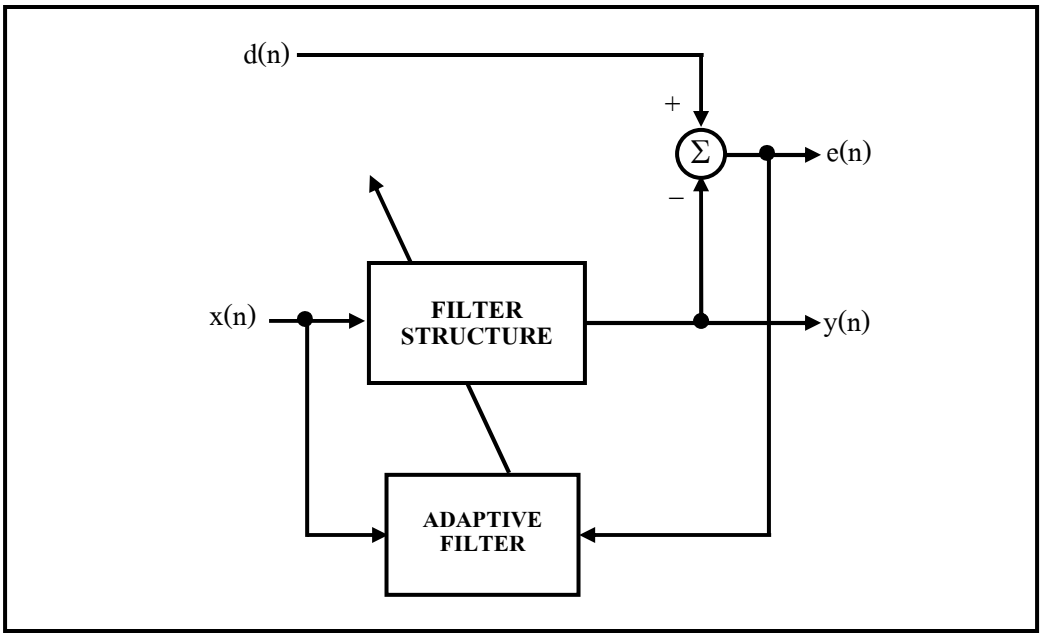
ขั้นที่ 3 Update ค่า $w(n+1) = w(n) + 2\mu e(n)x(n)$

ขั้นที่ 4 ไปทำซ้ำขั้นที่ 2

รูปแบบการทำงานจะวนลูปอย่างนี้ไปเรื่อย ๆ

2.2 การประยุกต์ใช้งานตัวกรองสัญญาณแบบปรับตัวเอง

คุณสมบัติสำคัญของตัวกรองสัญญาณแบบปรับตัวเอง คือ สามารถทำงานได้อย่างมีประสิทธิผลในสภาพแวดล้อมที่ไม่อาจคาดเดาได้ และการติดตามสัญญาณอินพุทซึ่งมีลักษณะสมบัติแปรเปลี่ยนตามเวลาได้ ระบบต่าง ๆ ซึ่งตัวกรองสัญญาณแบบปรับตัวเองประสบผลสำเร็จดีในการนำไปประยุกต์ใช้งาน ได้แก่ ระบบสื่อสาร, เรดาร์ (Radar) โซนาร์ (Sonar), ระบบควบคุมและการประมวลผลภาพรูปแบบ โดยทั่วไปของตัวกรองสัญญาณแบบปรับตัวเองแสดงดังภาพที่ 2-2 ประกอบด้วยสัญญาณอินพุท x(n) และ d(n) สัญญาณเอาท์พุท y(n) และสัญญาณค่าผิดพลาด e(n) ซึ่งเป็นผลต่างของสัญญาณที่ต้องการ d(n) และสัญญาณเอาท์พุท y(n) ตัวกรองสัญญาณแบบปรับตัวเองสามารถประยุกต์ใช้งานได้หลายแบบชิ้นกับการจัดรูปแบบของอินพุทและเอาท์พุท ดังตัวอย่างแบบย่อ ๆ ต่อไปนี้

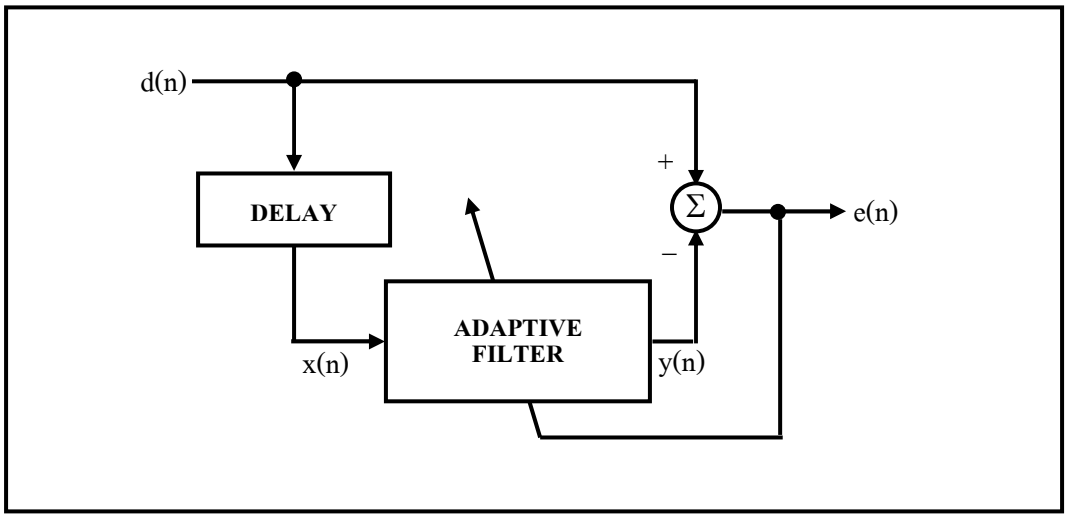


ภาพที่ 2-2 แสดงรูปแบบและส่วนประกอบของตัวกรองสัญญาณแบบปรับตัวเอง

2.3 การคาดคะเนแบบปรับตัวเอง (Adaptive Prediction)

ตัวคาดคะเนแบบปรับตัวเองแสดงรูปแบบดังรูปที่ 11 ประกอบด้วยสัญญาณที่เกี่ยวข้อง คือ d(n) เป็นสัญญาณอินพุท, x(n) เป็นสัญญาณอินพุทที่ถูกหน่วงเวลา, y(n) เป็นสัญญาณที่ถูกสร้างโดยการคาดคะเนของตัวกรองสัญญาณแบบปรับตัวเองและ e(n) คือสัญญาณค่าผิดพลาดของการคาดคะเน

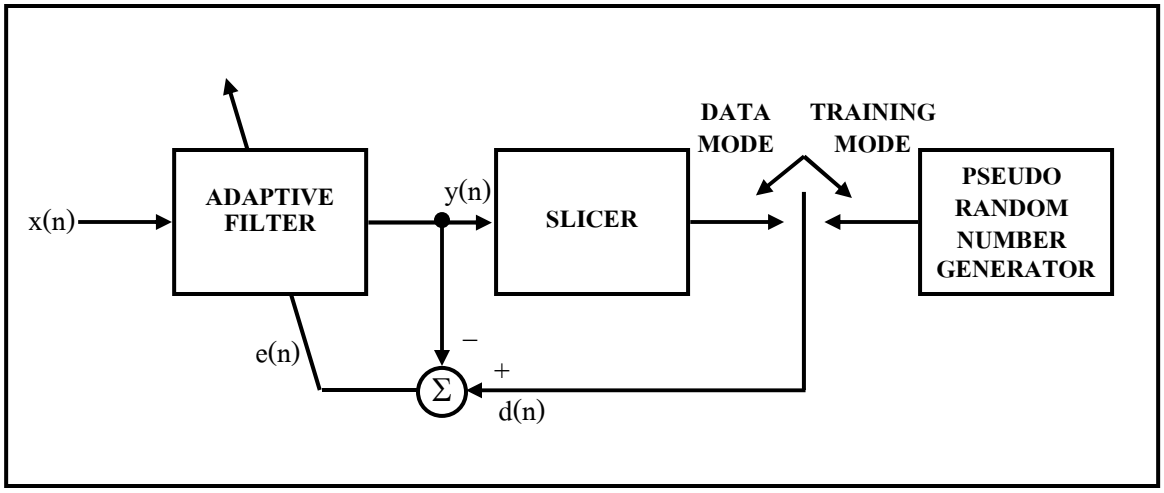
การใช้งานส่วนใหญ่ของการคาดคะเนแบบปรับตัวเอง คือการเข้ารหัสสัญญาณพูด ตัวกรองสัญญาณแบบปรับตัวเองถูกออกแบบให้ใช้ประโยชน์ของการมีความสัมพันธ์กันของค่าสัญญาณสุ่ม(Sample)ใกล้เคียงกันของสัญญาณพูดซึ่งจะได้ขนาดของสัญญาณค่าผิดพลาดจากการคาด คะเนนี้จะถูกคัดแยกระดับ(Quantized)และถูกส่งไปยังตัวรับสัญญาณเพื่อลดจำนวนบิทให้เหมาะสมตามความต้องการของระบบส่งสัญญาณ แบบของการเข้ารหัสนี้ เรียกว่า เอดีพีซีเอ็ม (ADPCM ซึ่งย่อจาก Adaptive “Differential Pulse-Code Modulation) ตัวคาดคะเนแบบปรับตัวเองยังสามารถใช้สำหรับตรวจสอบและเพิ่มระดับสัญญาณแถบความถี่ แคบ ๆ ซึ่งผสมรวมอยู่ในสัญญาณปะปนความถี่กว้างได้



ภาพที่ 2-3 แสดงตัวคาดคะเนแบบปรับตัวเอง

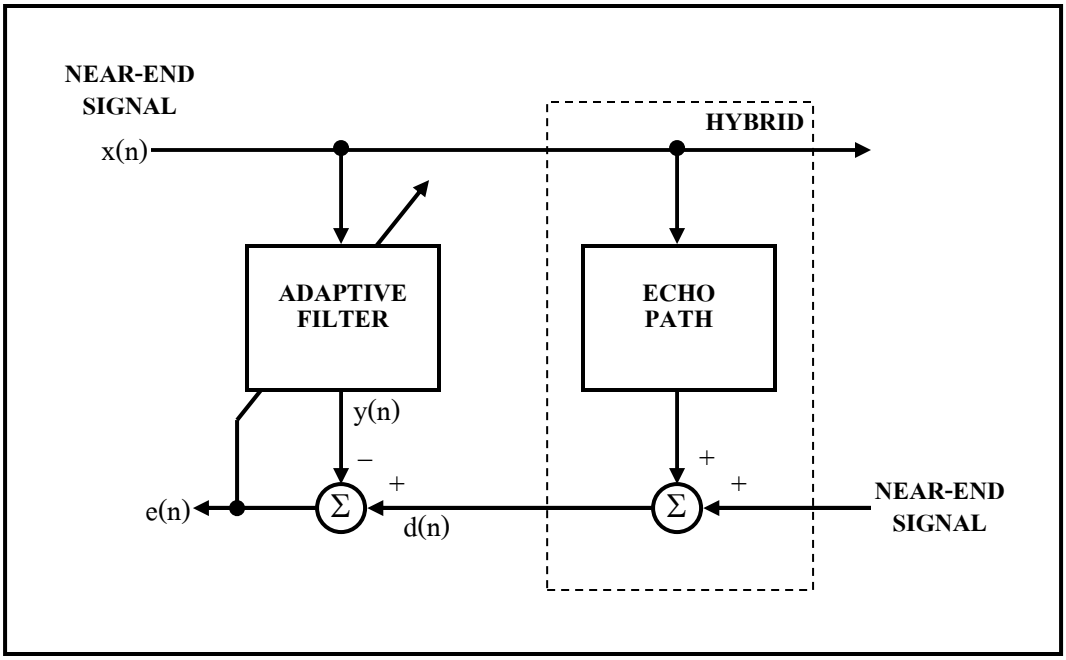
2.4 การปรับระดับสัญญาณแบบปรับตัวเอง (Adaptive Equalization)

ตัวปรับระดับสัญญาณแบบปรับตัวเอง ประกอบด้วย x(n) เป็นสัญญาณอินพุทที่รับได้ ซึ่งมีทั้งสัญญาณที่ต้องการรวมกับสัญญาณปะปนในช่องสัญญาณ, d(n) จะเป็นสัญญาณที่ผ่านการคัดแยกแล้วกรณีรับข้อมูล (Data Mode) หรือเป็นสัญญาณปะปนเทียม (Pseudo Random Number) กรณีปรับ สปส. ของตัวกรองสัญญาณ (Training Mode), y(n) เป็นสัญญาณซึ่งผ่านการปรับระดับสัญญาณแล้วใช้เป็นอินพุทของการคัดแยกข้อมูลที่รับได้ และ e(n) เป็นสัญญาณค่าผิดพลาดที่ยังคงเหลือจากการมีการแทรกสอดของสัญลักษณ์รวมกับสัญญาณปะปนองค์ประกอบของตัวปรับระดับสัญญาณแบบปรับตัวเองแสดงได้ดังภาพที่ 2-4



ภาพที่ 2-4 แสดงตัวปรับระดับสัญญาณแบบปรับตัวเอง

การปรับระดับสัญญาณแบบปรับตัวเองเป็นกระบวนการที่ใช้สำหรับกำจัดการบิดเบี้ยวของแอมพิจูดและเฟสเมื่อมีการส่งสัญญาณผ่านช่องสัญญาณสื่อสารซึ่งถือ เป็นการประยุกต์ใช้งานอย่างแรกของตัวกรองสัญญาณแบบปรับตัวเองในด้านโทรคมนาคมเมื่อมีการส่งสัญลักษณ์ผ่าน ช่องสัญญาณ จะเกิดการกระจายออกตามเวลา(TimeDispersive)ให้ผลของสัญลักษณ์ที่รับได้ไม่ชัดเจน ทั้งนี้เนื่องจากช่องสัญญาณส่วนใหญ่มีลักษณะสมบัติแปรเปลี่ยนตามเวลาและไม่สามารถรู้ ล่วงหน้าได้ดังนั้นตัวปรับระดับสัญญาณของช่องสัญญาณแบบปรับตัวเอง(AdaptiveChannelEqualizer)จึงถูกออกแบบขึ้นเพื่อจัดการกับปัญหาการแทรกสอดของสัญลักษณ์ (IntersymbolInterference) และมีการใช้กันอย่างกว้างขวางสำหรับเพิ่มประสิทธิผลของช่องสัญญาณ เช่น ในสายโทรศัพท์และช่องสัญญาณวิทยุซึ่งมีแถบความถี่จำกัด



ภาพที่ 2-5 แสดงตัวกำจัดสัญญาณสะท้อนแบบปรับตัวเอง

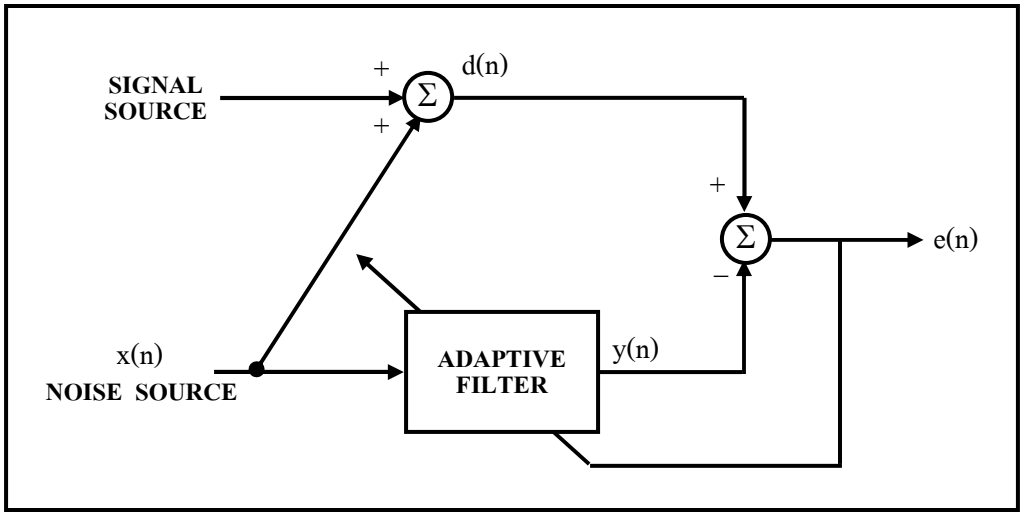
2.5 การกำจัดสัญญาณสะท้อนแบบปรับตัวเอง (Adaptive Echo Cancellation)

ตัวกำจัดสัญญาณสะท้อนแบบปรับตัวเอง ประกอบด้วย x(n) เป็นสัญญาณที่รับได้จากระยะไกล, d(n) เป็นสัญญาณที่เกิดการรวมของสัญญาณสะท้อน กับสัญญาณที่ต้องการส่งออก, y(n) เป็นสัญญาณเลียนแบบสัญญาณสะท้อน และ e(n) เป็นสัญญาณที่ต้องการส่งออกบวกกับสัญญาณสะท้อนที่เหลือ รูปแบบของตัวกำจัดสัญญาณสะท้อนแบบปรับตัวเองแสดงดังภาพที่ 2-5 ในการใช้งานได้มีการนำไปใช้ในการกำจัดสัญญาณสะท้อน ในการสื่อสารของโทรศัพท์ผ่านระยะทางไกล ๆ หรือในโมเดม (MODEM) ที่เป็นระบบฟูลดูเฟล็กซ์ (Full-Duplex) ย่านความถี่เสียง ในการใช้งานตัวกำจัดสัญญาณสะท้อนจะต้องติดตั้งทั้งสองด้านของระบบ

2.6 การกำจัดสัญญาณปะปนแบบปรับตัวเอง (Adaptive Noise Cancellation

รูปแบบของตัวกำจัดสัญญาณปะปนแบบปรับตัวเองแสดงได้ดังภาพที่ 2-6 ประกอบด้วย d(n) เป็นสัญญาณอินพุทหลัก(PrimaryInput)ประกอบด้วยสัญญาณที่ต้องการรวมกับสัญญาณปะปน, x(n) เป็นสัญญาณอินพุทอ้างอิง (Reference Input) ตัวกรองสัญญาณแบบปรับตัวเองจะประเมินสัญญาณปะปนในอินพุทหลัก d(n) ได้เป็นสัญญาณ y(n) นำไปลบกับสัญญาณอินพุทหลักจะได้สัญญาณเอาท์พุทซึ่งกำจัดสัญญาณปะปนออกแล้ว ในที่นี้คือ สัญญาณค่าผิดพลาด

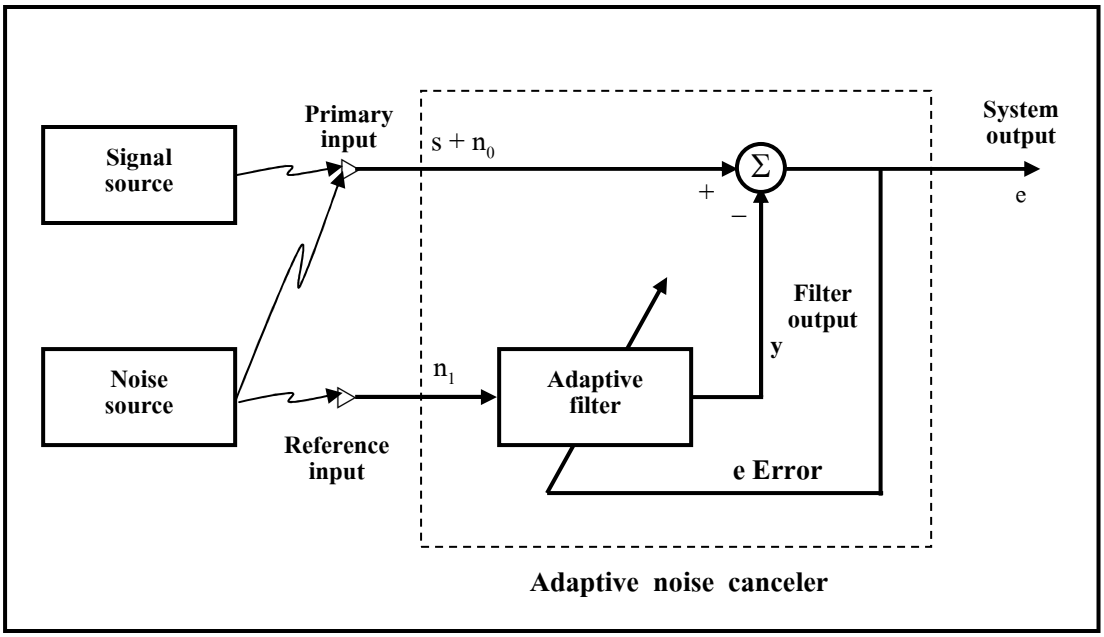
ตัวกำจัดสัญญาณปะปนแบบปรับตัวเองได้มีการนำไปใช้กับการกำจัดสัญญาณปะปนในเครื่องตรวจวัดการเต้นของหัวใจ (Electrocardiography), สัญญาณปะปนในเสียงพูด, การแทรกสอดของคลื่นไซด์โลบ(Sidelobe)ในจานสายอากาศส่วนของรายละเอียดเกี่ยวกับตัวกำจัดสัญญาณปะปนแบบปรับตัวเอง จะกล่าวถึงในหัวข้อต่อไป



ภาพที่ 2-6 แสดงตัวกำจัดสัญญาณปะปนแบบปรับตัวเอง

2.7 ตัวกำจัดสัญญาณปะปนแบบปรับตัวเอง

ตัวกำจัดสัญญาณปะปนที่จะพิจารณาต่อไปจะใช้กับกรณีที่สัญญาณปะปนเป็นแบบเสริม (Additive Noise) [3] เท่านั้น องค์ประกอบที่สำคัญของตัวกำจัดสัญญาณปะปนคือ ตัวกรองสัญญาณ ซึ่งต้องการการออกแบบที่เหมาะสมเพราะหากใช้ตัวกรองสัญญาณที่ไม่เหมาะสมอาจทำให้เกิดสัญญาณปะปนเพิ่มขึ้น หรือเกิดการบิดเบี้ยวของสัญญาณได้การออกแบบตัวกรองสัญญาณสามารถทำได้ 2 แบบ คือ แบบตรึงสัมประสิทธิ์ (Fixed Filter) และแบบปรับตัวเอง (Adaptive Filter) แต่เนื่องจากการออกแบบโดยใช้ตัวกรองแบบตรึงสัมประสิทธิ์จะต้องรู้ลักษณะสมบัติพื้นฐานของช่องสัญญาณที่สัญญาณผ่าน และยังอาจเปลี่ยนแปลงได้ตามสภาพแวดล้อม การออกแบบจึงทำได้ยาก ในขณะที่ตัวกรองแบบปรับตัวเองสามารถปรับสัมประสิทธิ์เองได้ การพิจารณาจะเลือกเฉพาะกรณีใช้ตัวกรองแบบปรับตัวเอง



ภาพที่ 2-7 แสดงหลักการของตัวกำจัดสัญญาณปะปนแบบปรับตัวเอง

หลักการทำงานโดยทั่วไปของตัวกำจัดสัญญาณปะปน (Noise Canceler) คือ ใช้ตัวรับสัญญาณหนึ่งตัวสำหรับรับสัญญาณปะปน เพื่อใช้เป็นอินพุทอ้างอิง (Reference Input) ซึ่งจะผ่านเข้าตัวกรองสัญญาณ แล้วนำสัญญาณที่ได้ไปหักล้างกับสัญญาณปะปนในอินพุทหลัก (Primary Input) ซึ่งจะได้สัญญาณที่ต้องการโดยปราศจากสัญญาณปะปน

กรณีเป็นตัวกำจัดสัญญาณปะปนแบบปรับตัวเอง (Adaptive Noise Canceler) หลักการทำงานแสดงในภาพที่ 2-7 สัญญาณที่ต้องการ S ถูกส่งผ่านช่องสัญญาณไปยังตัวรับสัญญาณซึ่งจะรับเอาสัญญาณปะปน n_o เข้ามาด้วยแต่ไม่เกี่ยวพันกัน ผลรวมของสัญญาณทั้งสองได้เป็น $s + n_o$ เข้าที่อินพุทหลักของตัวกำจัดสัญญาณปะปน ส่วนตัวรับสัญญาณตัวที่ 2 จะรับเอาสัญญาณปะปน n_1 ซึ่งเกี่ยวพันกับสัญญาณปะปน n_o แต่ไม่เกี่ยวพันกับสัญญาณ s ตัวรับสัญญาณตัวนี้จะเป็นอินพุทอ้างอิงของตัวกำจัดสัญญาณปะปน โดยสัญญาณปะปน n_1 จะผ่านตัวกรองสัญญาณแบบปรับตัวเองและให้เอาท์พุทเป็นสัญญาณ y ซึ่งจะถอดแบบสัญญาณปะปน n_o อย่างใกล้เคียง จากนั้น เอาท์พุทนี้จะถูกเอาไปลบกับสัญญาณจากอินพุทหลัก $s + n_o$ และให้เอาท์พุทของระบบคือ $s + n_o - y$

ในภาพที่ 2-7 จะเห็นว่าสัญญาณจากอินพุทอ้างอิงจะถูกประมวลผลโดยตัวกรองสัญญาณแบบปรับตัวเอง (Adaptive Filter) ซึ่งจะมีการปรับค่าสัมประสิทธิ์ (Impulse Response) ของตัวเองโดยอัตโนมัติผ่านอัลกอริธึมแบบลีสท์แสควร์(LeastSquare)ได้แก่LMSซึ่งจะตอบสนองต่อสัญญาณผิดพลาด(Error)จากเอาท์พุทของตัวกรองดังนั้นด้วยอัลกอริธึมที่เหมาะสมตัวกรองจะสามารถทำงานภายใต้สภาพการที่เปลี่ยนแปลงได้และสามารถปรับตัวเองได้อย่างต่อเนื่องเพื่อคงสัญญาณผิดพลาดให้มีค่าต่ำที่สุด

ในกระบวนการปรับตัวเอง (Adaptive Process)จะนำเอาสัญญาณผิดพลาดไปประยุกต์ใช้งานต่างกันขึ้นกับธรรมชาติของระบบ ในกรณีของตัวกำจัดสัญญาณปะปนแบบปรับตัวเอง จุดประสงค์คือต้องการให้เอาท์พุทของระบบ $s + n_o - y$ ให้ผลใกล้เคียงที่สุดกับสัญญาณ s ในแง่ของลีสท์แสควร์ (Least Square) ซึ่งสามารถทำให้สำเร็จได้โดยการป้อนเอาท์พุทของระบบกลับไปยังตัวกรองสัญญาณแบบปรับตัวเอง ซึ่งใช้อัลกอริธึมแบบปรับตัวเอง (Adaptive Algorithm) เพื่อลดกำลังเอาท์พุททั้งหมดของระบบให้ต่ำที่สุดดังนั้นในกรณีตัวกำจัดสัญญาณปะปน สัญญาณเอาท์พุทของระบบจะหมายถึงสัญญาณผิดพลาดของกระบวนการปรับตัวเอง

ข้อควรคำนึงก่อนจะออกแบบตัวกรองสัญญาณคือ จะต้องรู้ว่าสัญญาณเป็นแบบใดระหว่าง Statistical หรือ Deterministic เพื่อสามารถสร้างสัญญาณกำจัดสัญญาณปะปนได้ ในกรณีนี้สมมติว่า s, n_o, n_1 และ y เป็นสัญญาณแบบ Statistically Stationary และมีค่าเฉลี่ยเป็นศูนย์ และสมมติว่า s ไม่เกี่ยวพันกับ n_o และ n_1 แต่ n_1 มีความเกี่ยวพันกับ n_o เอาท์พุทของระบบจะเป็นดังนี้

$$e = s + n_o \tag{2-15}$$

และค่ากำลังสองของสมการ คือ

$$e^2 = s^2 + (n_o - y)^2 + 2s(n_o - y) \tag{2-16}$$

เมื่อใส่เอ็กเป็กเตชั่น (Expectation) ทั้งสองข้างของ (4.9) และอาศัยความสัมพันธ์ที่กำหนดในตอนแรก คือ s ไม่เกี่ยวพันกับ n_o และ y จะได้ค่าเอ็กเป็กเตชั่น (Expectation) ของ e^2 ดังนี้

$$E[e^2] = E[s^2] + E[(n_o - y)^2] \tag{2-17}$$

จะเห็นว่าค่ากำลังสองของ $E[s^2]$ ไม่มีผลต่อการปรับตัวเพื่อลดค่าของ $E[s^2]$ ซึ่งค่ากำลังของเอาท์พุทจะต่ำสุดเมื่อ

$$E_{min}[e^2] = E[s^2] + E_{min}[(n_o - y)^2] \quad (2\text{-}18)$$

เมื่อตัวกรองสัญญาณปรับตัวเอง เพื่อลดค่าของ $E[s^2]$ ให้ต่ำสุด ค่า $E[(n_o - y)^2]$ จะลดลงด้วยซึ่งเอาท์พุทของตัวกรอง y ในขณะนั้นจะใกล้เคียงกับสัญญาณปะปน n_o ที่อินพุทหลักมากที่สุด นอกจากนี้เมื่อค่า $E[(n_o - y)^2]$ ลดลงต่ำสุด ค่าของ $E[(e - s)^2]$ จะลดลงต่ำสุดด้วย เพราะเมื่อพิจารณาตามสมการ 2-15 จะได้ว่า

$$e - s = n_o - y \quad (2\text{-}19)$$

ทั้งนี้การปรับตัวเองของตัวกรองสัญญาณเพื่อลดค่ากำลังของเอาท์พุทให้ต่ำสุดนั้นจะได้ผลดีเพียงใดขึ้นอยู่กับโครงสร้างที่กำหนด และความสามารถในการปรับตัวเองของ ตัวกรองสัญญาณแบบปรับตัวเองที่เลือกใช้ ตลอดจนอินพุทอ้างอิงที่กำหนด

สัญญาณเอาท์พุท e โดยทั่วไปจะประกอบด้วยสัญญาณ s บวกกับสัญญาณปะปนบางส่วนซึ่งถูกกำหนดโดย $n_o - y$ เพราะว่าการลดค่าของ $E[e^2]$ จะเท่ากับการลดค่า $E[n_o - y)^2]$ ฉะนั้นการลดค่ากำลังของเอาท์พุท จะเท่ากับเป็นการลดค่ากำลังของสัญญาณปะปนเอาท์พุทเพราะสัญญาณ s ในเอาท์พุทจะยังคงที่อยู่ หรือ สรุปได้ว่าการลดค่ากำลังของเอาท์พุทจะเป็นการเพิ่มค่า อัตราส่วนของสัญญาณต่อสัญญาณปะปน (S/N ratio) จากสมการ (5.0) จะเห็นว่าค่าที่เล็กที่สุดซึ่งเป็นไปได้ของค่ากำลังเอาท์พุทคือ $E_{min}[e^2] = E[s^2]$ เมื่อเราสามารถทำให้ค่า $E[n_o - y)^2] = 0$ ซึ่งก็คือ $y - n_o$ และ $e = s$ ในกรณีนี้การลดค่ากำลังของเอาท์พุท จะทำให้สัญญาณเอาท์พุทปราศจากสัญญาณปะปน แต่ในทางกลับกันหากว่าสัญญาณที่รับเข้าทางอินพุทอ้างอิงไม่เกี่ยวพันกันเลยกับ สัญญาณที่รับเข้าทางอินพุทหลักจะทำให้ตัวกรองสัญญาณหยุดตัวเองและไม่เพิ่มสัญญาณปะปนที่ เอาท์พุท และ ในกรณีนี้สัญญาณเอาท์พุท y จากตัวกรองสัญญาณก็จะไม่เกี่ยวพันกับสัญญาณ
อินพุทหลัก ฉะนั้นค่ากำลังเอาท์พุทจะเป็นดังนี้

$$E[e^2] = E[(s + n_o)^2] + 2E[-y(s + n_o)] + E[y^2] \quad (2\text{-}20)$$

ในกรณีนี้ตัวกรองสัญญาณแบบปรับตัวเองจะพยายามลดค่ากำลังของเอาท์พุทโดยกำหนดให้ค่าสัม $= E[(s + n_o)^2] + E[y^2]$

ประสิทธิ์ของตัวกรองทั้งหมดเป็นศูนย์ ซึ่งจะทำให้ $E[y^2]$ เป็นศูนย์

2.8 ตัวกรองสัญญาณแบบเอฟไอาร์ (FIR)

เอฟไอาร์มีคุณสมบัติดังสมการ

$$y(n) = \sum_{k=0}^{n-1} h(k)x(n-k) \tag{2-21}$$

เขียนให้อยู่ในรูปแซดโดเมน (Z – domain) ดังสมการ

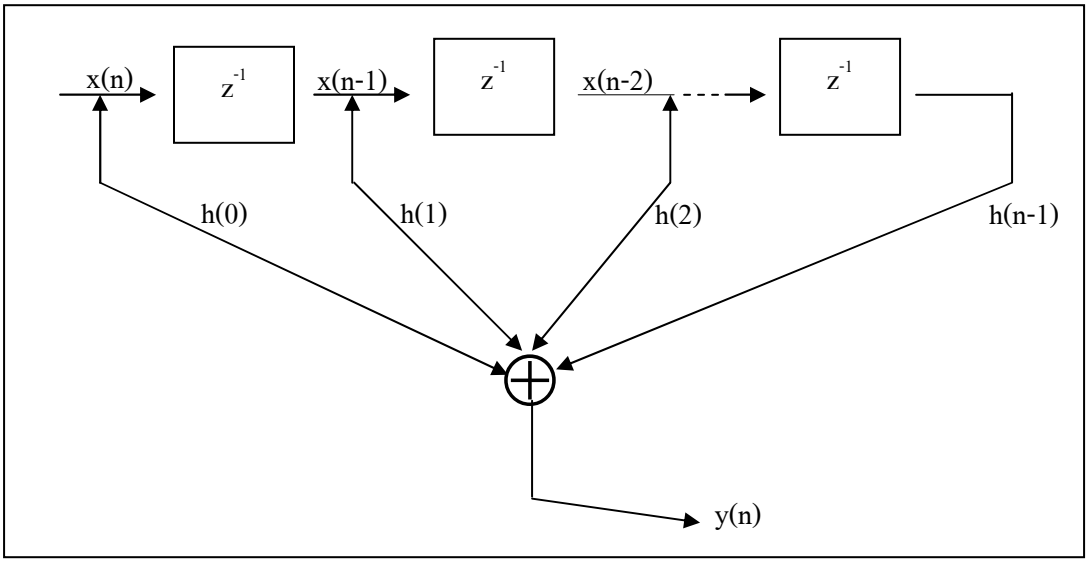
$$y(z) = \sum_{k=0}^{n-1} h(k)x(z)z^{-k} \tag{2-22}$$

หรือสามารถเขียนให้อยู่ในรูปของฟังก์ชั่นถ่ายโอนจะได้

$$H(z) = \sum_{k=0}^{n-1} h(k)z^{-k}$$

$$y(z) = h(0)x(z) + h(1)x(z)z^{-1} + h(z)x(z)z^{-2} + ...h(k)x(z)z^{-k} \tag{2-23}$$

จากสมการที่ 2-23 สามารถนำไปเขียนเป็นโครงสร้างได้ดังภาพที่ 2.8



ภาพที่ 2-8 แสดงโครงสร้างของเอฟไอาร์

ตัวกรองแบบเอฟไอออาร์จัดเป็นตัวกรองสัญญาณแบบนอนรีเคอร์ซีฟ (Non Recursive) เนื่องจากไม่มีการป้อนกลับจากทางด้านเอาท์พุท และนอกจากนี้ตัวกรองสัญญาณแบบเอฟไอออาร์ยังมีคุณสมบัติอื่น ๆ อีก คือ

2.8.1 สามารถสร้างได้ง่ายเมื่อเทียบกับตักรองแบบไอไอออาร์

2.8.2 มีคุณสมบัติการตอบสนองทางเฟสเป็นแบบเชิงเส้น (Linear Phase)

2.8.3 สัมประสิทธิ์ที่เกิดจากการคำนวณจะมีค่าไม่เกินหนึ่งทำให้สามารถสร้างและทำงานได้ดีบนตัวประมวลผลแบบจุดทศนิยมคงที่ (Fix-point)

2.8.4 ตัวกรองแบบเอฟไอออาร์จะเสถียรภาพ (Stable) แน่นอน เนื่องจากว่ามีโพลอยู่ที่จุดกำเนิด (Origin) บนระนาบแซด

พิจารณาจากสมการ

$$y(n) = x(n-k) \tag{2-24}$$

เมื่อทำการแปลงฟูเรียร์ ในสมการ (2.24) จะได้

$$y(j\varpi) = e^{-j\varpi kt} x(j\varpi) \tag{2-25}$$

ย้ายข้างจะได้

$$\frac{y(j\varpi)}{x(j\varpi)} = H(j\varpi) = e^{-j\varpi kt} \tag{2-26}$$

จากสมการ 2-26 จะมีขนาด (Magnitude) เท่ากับ 1 และมีเฟสดังสมการ

$$\theta(\varpi) = -\varpi kt \tag{2-27}$$

และจากสมการ (2.27) ซึ่งเป็นสมการของเฟสสามารถนำไปหาค่าหน่วงกลุ่ม (Group Delay) ได้ โดยทำการหาอนุพันธ์สมการ (2.27) เทียบกับ w จะได้สมการของค่าหน่วงกลุ่มดังสมการ

$$_{d}\theta = \frac{d(-\varpi kt)}{d\varpi}$$

$$\theta = -kt \qquad (2\text{-}28)$$

จากสมการที่ (2.28) จะเห็นได้ว่าตัวกรองสัญญาณแบบเอฟไออาร์มีผลตอบสนองทางเฟสเป็นเชิงเส้น ดังนั้น ตัวกรองแบบเอฟไออาร์ จึงถูกนำไปใช้งานอย่างกว้างขวาง และถ้ากำหนดสมการผลตอบสนองทางเฟสใหม่จะได้

$$KT = \alpha \qquad (2\text{-}29)$$

จะได้

$$\theta(w) = -\alpha w \qquad (2\text{-}30)$$

หรือถ้าค่าของผลตอบสนองทางเฟสเท่ากับ

$$\theta(w) = \beta - \alpha w \qquad (2\text{-}31)$$

β เป็นค่าคงที่

ถ้าตัวกรองมีทั้งผลการตอบสนองทางเฟสและค่าหน่วยกลุ่มเป็นแบบเชิงเส้น ตามสมการที่ 2-30 จะให้ค่าผลตอบสนองอิมพัลส์ของตัวกรองเป็นแบบสมมาตรบวก (Positive Symmetry) ซึ่งผลตอบสนองทางเฟสจะเป็นฟังก์ชั่นของความยาวตัวกรอง (Filter Length)

$h(n) = h(N-n-1)$ ที่ $n = 0, 1,\ldots,(N-1)/2$ และ n เป็นจำนวนคี่

$$\alpha = \left(\frac{N-1}{2}\right)$$ ที่ $n = 0, 1,\ldots,(N-1)/2$ และ n เป็นจำนวนคี่

และ n เป็นจำนวนคู่

และถ้าฟิลเตอร์มีผลตอบสนองทางเฟสดังสมการที่ (2.31) จะได้ผลตอบสนองอิมพัลส์ของตัวกรองเป็นแบบสมมาตรลบ (Negative Symmetry) ดังสมการ

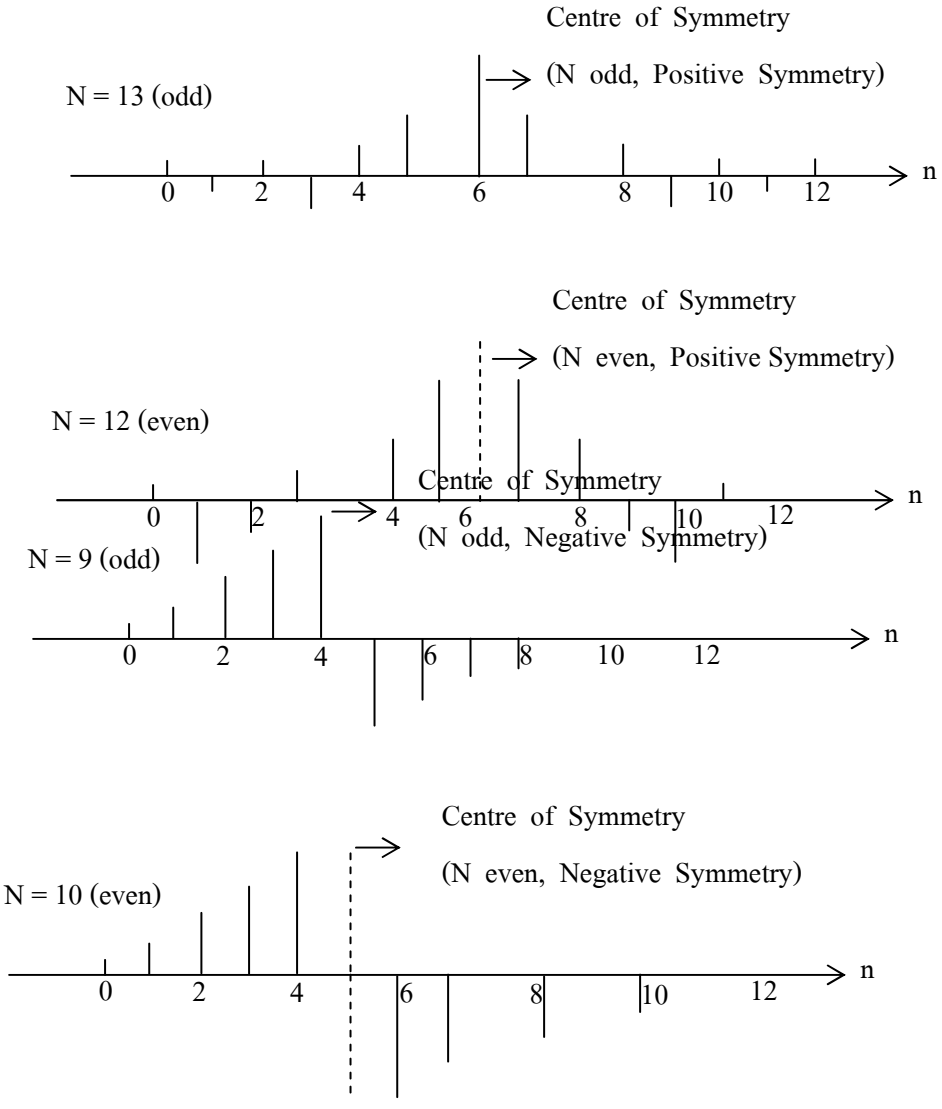
$$h(n) = -h(N-n-1)$$

$$\alpha = \left(\frac{N-1}{2}\right)$$

โดยที่ค่า N เป็นจำนวนลำดับของตัวกรองและ

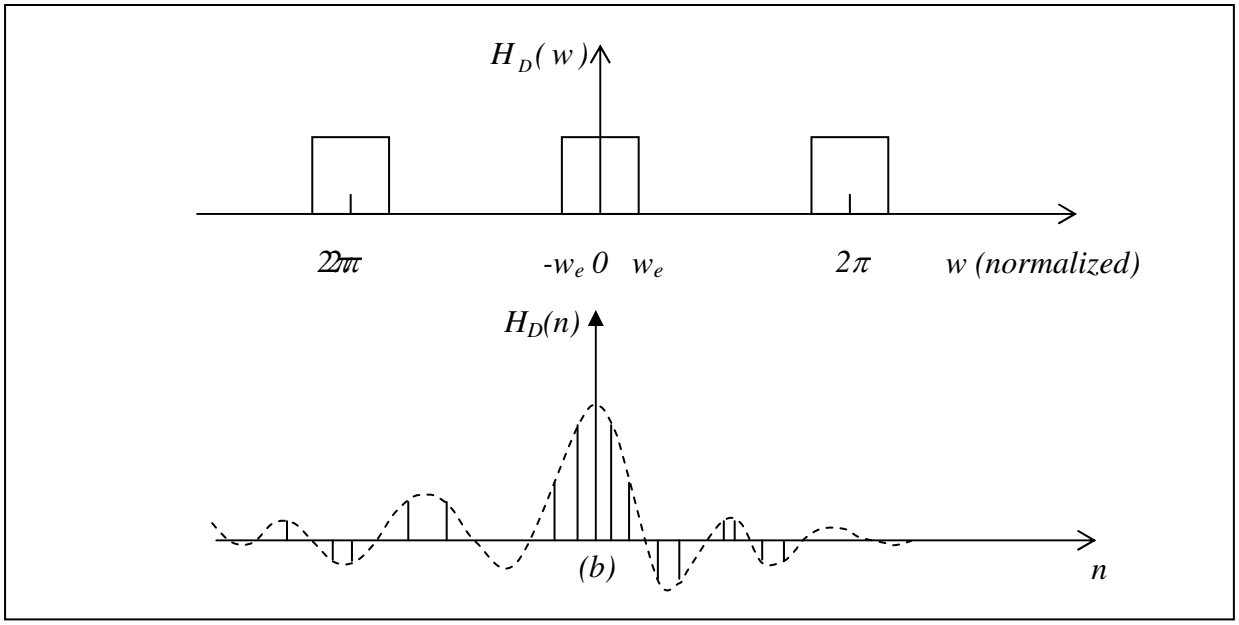
$0 \le n \le n-1$ ดังนั้น ถ้า N = 7 จะได้ h(0) = h(6), h(1) = h(5), h(2) = h(4)

และถ้า N = 8 จะได้ h(0) = h(7), h(1) = h(6), h(2) = h(5), h(3) = h(4) จากสมการข้างต้นสามารถนำไปเขียนเป็นกราฟได้ ดังนี้



ภาพที่ 2-9 แสดงผลตอบสนองอิมพัลส์ของเฟสเชิงเส้นของตัวกรอง 4 ชนิด

การออกแบบตัวกรองดิจิตอลแบบเอฟไออาร์ จะใช้วิธีฟูเรียร์ซีรีส์ (Fourier Series)มาทำการออกแบบพิจารณาภาพที่ 2-10



ภาพที่ 2-10 แสดงผลตอบสนองความถี่และผลตอบสนองอิมพัลส์ในทางอุดมคติ

(a) แสดงการตอบสนองความถี่ของตัวกรองความถี่ต่ำผ่านในทางอุดมคติ

(b) แสดงผลตอบสนองอิมพัลส์ของตัวกรองความถี่ต่ำผ่านในทางอุดมคติ

ตัวกรองที่มีผลตอบสนองทางความถี่ $H_D(\varpi)$ สามารถหาค่าผลตอบสนองอิมพัลส์ $N_D(n)$ ได้จากความสัมพันธ์ของการแปลงกลับของฟูเรียร์ ดังสมการ

$$h_d(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_D(w)e^{jwn}dw \qquad (2\text{-}32)$$

พิจารณาจากภาพที่ 2-9 ซึ่งเป็นกราฟแสดงการตอบสนองความถี่ของตัวกรองความถี่ต่ำผ่านมีความคัท-ออฟ คือ W_c โดย W_c นี้จะเป็นความถี่นอร์มอลไลซ์ (Normalized Frequencey) ซึ่งมีค่า

$$w_c = 2\pi f$$

$$f = \frac{f_c}{f_s}$$

โดย f คือ ความถี่คัท-ออฟนอร์มอลไลซ์ (Normalized Cut off Frequency)

f_c คือ ความถี่คัท-ออฟ (Cut off Frequency (H_z))

f_s คือ ความถี่สุ่ม (Sampling Frequency (H_z))

จากกราฟภาพที่ 2-10 จะเห็นได้ว่าค่า $H_D(\omega)$ จะมีค่าเท่ากับ 1 ในช่วงตั้งแต่ $-(\omega)_c$ จนถึง $+(\omega)_c$ ดังนั้นค่าของผลตอบสนองอิมพัลส์ของภาพที่ 2-10 (a) จะหาได้จากสมการ

$$h_D(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} 1 \times e^{j\varpi n} dw = \frac{1}{2\pi}\int_{-\varpi c}^{\varpi c} e^{j\varpi n} dw \quad (2\text{-}33)$$

$$h_D(n) = \frac{\sin(2\pi fn)}{n\pi} \quad (2\text{-}34)$$

$$h_D(n) = \frac{2 \times \sin(2\pi fn)}{2\pi fn} \quad (2\text{-}35)$$

เนื่องจากค่าผลตอบสนองอิมพัลส์นั้นมีความสมมาตรกัน ดังนั้นในการหาค่าจะทำการหาเพียงครึ่งหนึ่งก็พอ เช่น ถ้าต้องการหาผลตอบสนองอิมพัลส์ n = 53 จะทำการหาค่า n ตั้งแต่ 0 จนถึง 26 ส่วนค่า -1 จนถึง -26 นั้น ไม่จำเป็นต้องหาเนื่องจากความเป็นสมมาตรนั่นเอง

2.9 ตัวกรองสัญญาณแบบไอไออาร์ (IIR)

คุณสมบัติสำคัญของตัวกรองแบบไอไออาร์ คือให้ผลตอบสนองทางความถี่ได้คม (Sharp-Cutoff Frequency) ซึ่งดีกว่าตัวกรองแบบเอฟไออาร์ เนื่องจากสมการทรานเฟอร์ฟังก์ชั่นเป็นแบบแลทชั่นแนล (Rational Function) มีทั้งโพล (Pole) และ ซีโร่ (Zero) สามารถปรับเลือกค่าได้ ในการออกแบบจึงไม่ต้องใช้ออร์เดอร์ (Order) สูง แต่จะได้ตัวกรองเป็นแบบริเคอชิฟ (Recursive) คุณสมบัติไม่เป็นเชิงเส้น และเสถียรภาพไม่ดี การออกแบบทำได้ยาก ฟังก์ชั่นถ่ายโอน (Transfer Function) ของตัวกรองแบบไอไออาร์เขียนเป็นสมการ คือ

$$H(z) = \frac{\sum_{k=0}^{n} b_k z^{-k}}{1 + \sum_{k=1}^{n-1} a_k z^{-k}} \quad (2\text{-}36)$$

หรือจัดรูปแบบสมการ เขียนใหม่ได้คือ

$$H(z) = \frac{b_0 + b_1 z^{-1} + ... b_n z^{-n}}{1 + a.z^{-1} + ... a_m z^{-m}} \qquad (2\text{-}37)$$

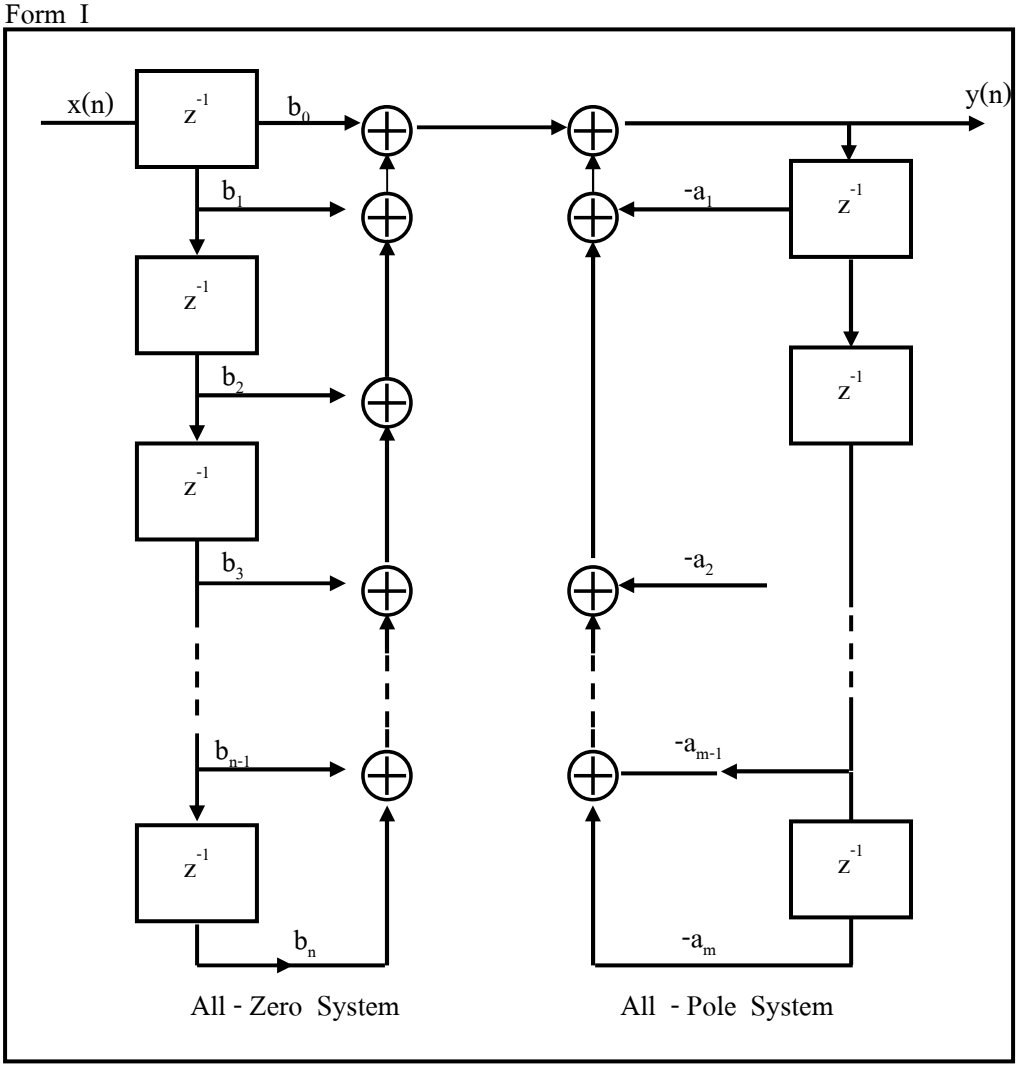
นำไปจัดเทอมต่าง ๆ ใหม่และเขียนอยู่ในรูปสมการผลต่าง (Difference Equation)

$$Y(z) = \sum_{k=0}^{n} b_k x_{(z)} z^{-k} - \sum_{k=1}^{m} a_k y_{(z)} z^{-k}$$

$$Y(z) = b_0 x_{(z)} + b_1 x_{(z)} z^{-1} + ... + b_k x_{(z)} z^{-k} - a_1 y_{(z)} z^{-1} - ... - a_k y_{(z)} z^{-k}$$

$$Y(n) = \sum_{k=0}^{n} b_k x^{(n-k)} - \sum_{k=1}^{m} a_k y^{(n-k)}$$

จากสมการที่ 2-37 สามารถนำมาเขียนเป็นโครงสร้างในภาพที่ 2-11 เป็นโครงสร้างแบบ Direct Form I



ภาพที่ 2-11 แสดงโครงสร้างแบบตรง I

ถ้านำ $\frac{W_{(z)}}{W_{(z)}}$ **คูณตลอดสมการของ 2-37 จะได้**

$$H_{(z)} = \frac{W_{(z)}}{W_{(z)}} \times \frac{b_0 + b_1 z^{-1} + ... + b_n z^{-n}}{1 + a_1 z^{-1} + ... a_m z^{-m}}$$

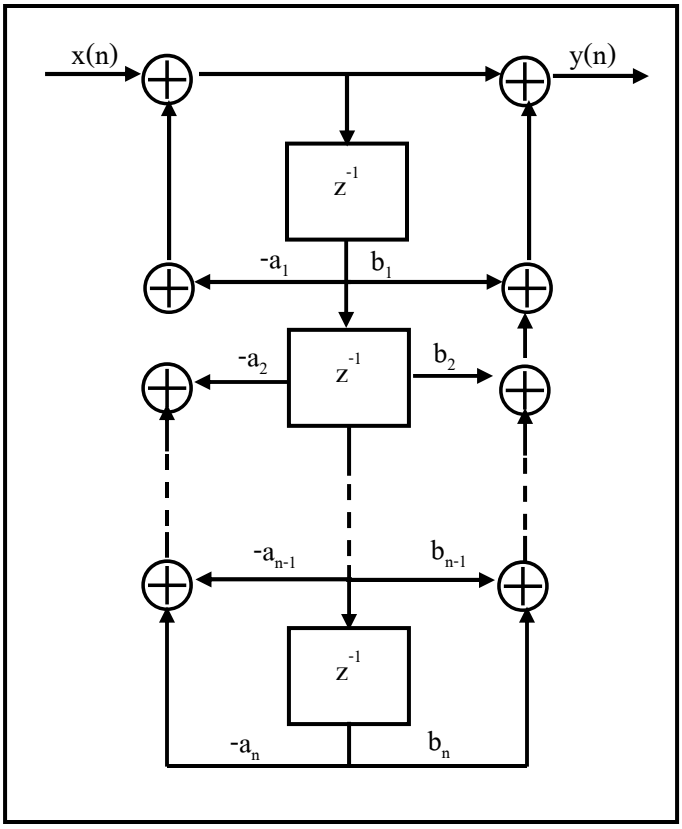
$$H_{(z)} = \frac{Y_{(z)}}{X_{(z)}} = \frac{W_{(z)}}{W_{(z)}} \times \frac{b_0 + b_1 z^{-1} + ... + b_n z^{-n}}{1 + a_1 z^{-1} + ... + a_m z^{-m}}$$

$$X_{(z)} = X_{(z)} + a_1 w_{(z)} Z^{-1} + ... + a_m w_{(z)} z^{-m} \qquad (2\text{ -}38)$$

$$W_{(z)} = X_{(z)} + a_1 w_{(z)} z^{-1} - ... - a_m w_{(z)} z^{-m} \qquad (2\text{-}39)$$

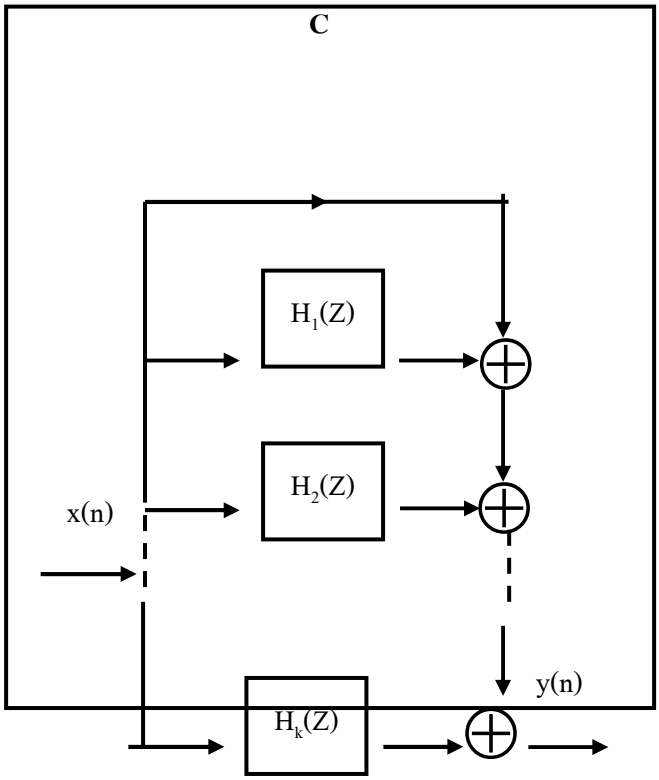
$$Y_{(z)} = b_0 w_{(z)} + b_1 w_{(z)} z^{-1} + ... + b_n w_{(z)} z^{-n} \qquad (2\text{-}40)$$

จากสมการของ 2-38 และ 2-40 สามารถนำมาเขียนเป็นโครงสร้างได้ดังภาพที่ 2-12 เรียกว่าโครงสร้างแบบ Direct Form II



ภาพที่ 2-12 แสดงโครงสร้างแบบตรง II

แต่เนื่องจากตัวกรองดิจิตอลที่มีโครงสร้างแบบ Direct Form I เมื่อมีจำนวนลำดับเพิ่มขึ้น จะมีค่าย่านไดนามิค (Dynamic Range) ของสัมประสิทธิ์มากขึ้น กล่าวคือค่าสัมประสิทธิ์ในเทอม a จะมีค่ามากและสัมประสิทธิ์ในเทอม b จะมีค่าน้อยทำให้เป็นปัญหาอย่างมาก โดยเฉพาะอย่างยิ่งถ้าใช้ตัวประมวลผลแบบจุดทศนิยมคงที่ (Fixed Point) ดังนั้น ในการใช้งานจริง ๆ จึงสามารถปรับโครงสร้างของสมการใหม่ได้เป็น Direct Form II

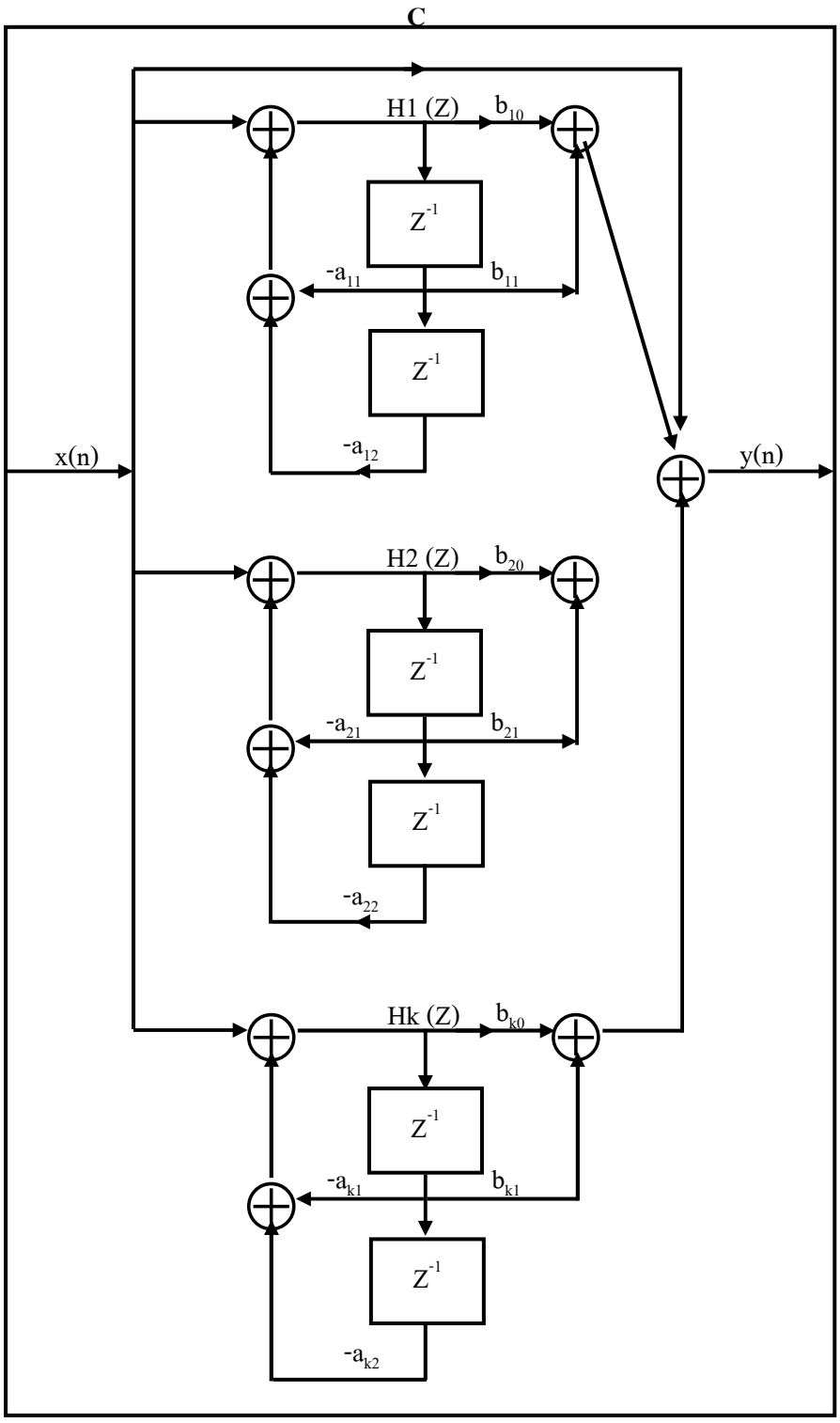


ภาพที่ 2-13 ไอไออาร์ที่มีโครงสร้างในแบบขนาน

ในภาพที่ 2-13 จะเห็นได้ว่าตัวกรองสัญญาณแบบไอไออาร์ที่มีโครงสร้างเป็นแบบขนานนี้จะประกอบด้วย $H_1(z)$, $H_2(z)$, $H_k(z)$, $H(z)$ แต่ละตัวจะมีโครงสร้างในภาพที่ 2-14 การจัดโครงสร้างตัวกรองสัญญาณแบบไอไออาร์ เป็นแบบขนานจะให้ผลดีคือ

2.9.1 ย่านไดนามิคของสัมประสิทธิ์มีค่าน้อยทำให้ไม่เป็นปัญหาเมื่อใช้กับตัวประมวลผลแบบจุดทศนิยมคงที่

2.9.2 เป็นอัลกอริธึมที่เหมาะที่จะใช้กับระบบประมวลผลสัญญาณดิจิตอลแบบขนานอีกด้วย



ภาพที่ 2-14 โครงสร้างย่อยของตัวกรองสัญญาณแบบขนาน

การออกแบบตัวกรองดิจิตอลแบบไอไออาร์

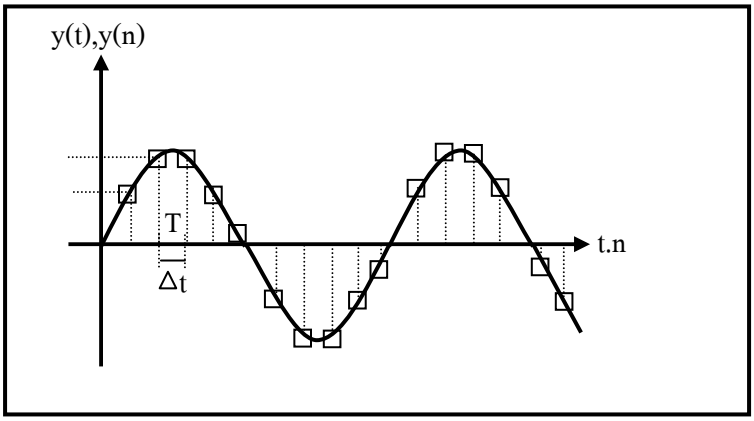
สามารถกระทำได้ 4 วิธี

1. แบ่งวางโพล-ซีโร (Pole-zero placement)
2. แบบประมาณค่าเบี่ยงเบน (Approximation of derivatives)
3. แบบอิมพัลส์อินวาเรนซ์ (Impulse invariance)
4. แบบการแปลงเชิงเส้นคู่ (Bilinear transform) [4]

สำหรับในตำรานี้จะกล่าวเพียง 2 วิธี คือ วิธีที่ 2 และวิธีที่ 4 เท่านั้น วิธีที่ 1 และวิธีที่ 3 จะไม่กล่าวถึง

2.10 การออกแบบตัวกรองดิจิตอลแบบไอไออาร์ด้วยวิธีประมาณค่าเบี่ยงเบน

เป็นวิธีการหนึ่งที่ใช้แปลงเอสโดเมนไปสู่แซดโดเมน พิจารณาภาพที่ 2-15



ภาพที่ 2-15 แสดงความสัมพันธ์ระหว่างสัญญาณที่เป็นช่วง (Discrete Signal) กับสัญญาณที่ต่อเนื่อง

จากภาพที่ 2-15 จะพบว่า

$$\frac{dy(t)}{dt} = \frac{\Delta y(t)}{\Delta t} \tag{2-41}$$

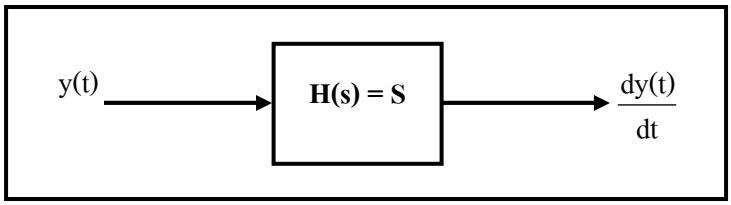
กำหนดให้ระยะห่างระหว่าง n กับ n + 1 เท่ากับ T และเท่ากับ Δt ดังนั้น ณ เวลาใด ๆ จะได้ $t = nT$

$$\frac{dy(t)}{dt} = \frac{y(nT) - y(nT - T)}{T} \qquad (2\text{-}42)$$

T คืออัตราสุ่ม (Sampling Rate) หรืออาจเขียนให้อยู่ในรูปความถี่สุ่ม, $f_S = \frac{1}{T}$ ก็ได้ ซึ่งค่า T นี้จะมีค่าเท่ากันทุกช่วง

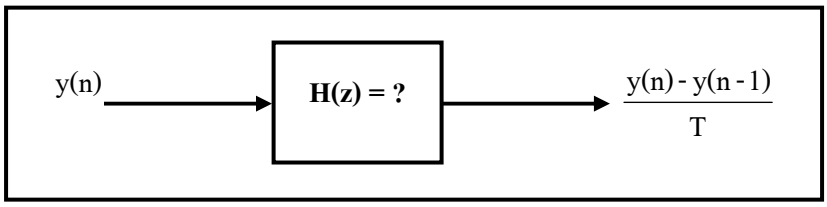
$$\frac{dy(t)}{dt} = \frac{y(n) - y(n - 1)}{T} \qquad (2\text{-}43)$$

กำหนดให้ $\frac{dy(t)}{dt}$ เป็นเอาท์พุทของระบบอนาลอกที่มีฟังก์ชั่นถ่ายโอน $H(s) = S$ โดยมีอินพุทเป็น $y(t)$



ภาพที่ 2-16 แสดงระบบอนาลอกที่มีฟังก์ชั่นถ่ายโอนเป็น H(s)

และจากสมการที่ 2-43 กำหนดให้ $\frac{y(n) - y(n - 1)}{T}$ เป็นเอาท์พุทของระบบไม่ต่อเนื่อง (Discrete) โดยมีอินพุทเป็น y(n) และมีฟังก์ชั่นถ่ายโอนเป็น H(z) จะได้



ภาพที่ 2-17 แสดงระบบไม่ต่อเนื่องที่มีฟังก์ชั่นถ่ายโอนเท่ากับ H(z)

จากภาพที่ 2-17 พบว่าเอาท์พุทของระบบไม่ต่อเนื่องคือ

$$y(z)\cdot H(z) = \frac{y(z) - y(z)Z^{-1}}{T}$$

$$y(z)\cdot H(z) = \frac{(1 - Z^{-1})y(z)}{T}$$

$$H(z) = \frac{(1 - Z^{-1})}{T} \quad (2\text{-}44)$$

และจากภาพที่ 2-16 เอาท์พุทของระบบอนาลอกจะมีค่าเท่ากับ

$$\frac{dy(t)}{dt} = H(s)\cdot y(t) \quad (2\text{-}45)$$

ณ จุดใด ๆ บนกราฟในภาพที่ 2-15 y(n) จะมีค่าเท่ากับ y(t) จะได้

$$\frac{dy(t)}{dt} = \frac{y(n) - y(t)}{T}$$

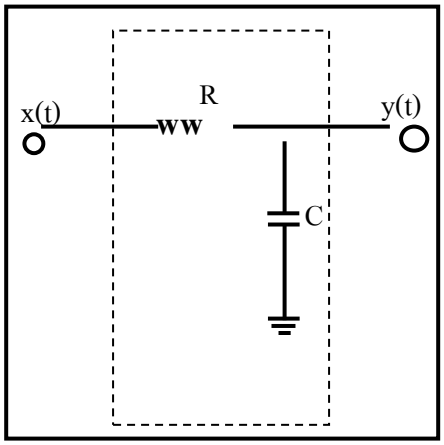
$$H(s)\cdot y(t) = H(z)\cdot y(n)$$

$$H(s) = H(z) \quad (2\text{-}46)$$

$$H(s) = S, H(z) = \frac{(1 - z^{-1})}{T} \quad (2\text{-}47)$$

$$Z = \frac{1}{1 - ST} \quad (2\text{-}48)$$

พิจารณาวงจรดังภาพที่2.18



ภาพที่ 2-18 แสดงวงจรกรองความถี่ต่ำผ่าน (RC Lowpass Filter)

จากภาพที่ 2-18 วงจรจะมีฟังก์ชั่นถ่ายโอนดังสมการ

$$H(s) = \frac{\frac{1}{RC}}{S+\frac{1}{RC}} \tag{2-49}$$

กำหนดให้ จะได้

$$a = \frac{1}{RC}$$

$$H(s) = \frac{a}{S+a} \tag{2-50}$$

จากสมการที่ 2-44 จะได้ H(z) มีค่าดังสมการที่ 2-50

$$H(z) = \frac{aT}{1 + aT - Z^{-1}} \tag{2-51}$$

$$Vo(z) + aTVo(z) - Vo(z)Z^{-1} = aTVi(z)$$

$$Vo(z) + aTVo(z) = aTVi(z) + Vo(z)Z^{-1}$$

$$Vo(z)(1 + aT) = aTVi(z) + Vo(z)Z^{-1}$$

$$Vo(z) = \frac{aTVi(z)}{(1 + aT)} + \frac{Vo(z)Z^{-1}}{(1 + aT)} \tag{2-52}$$

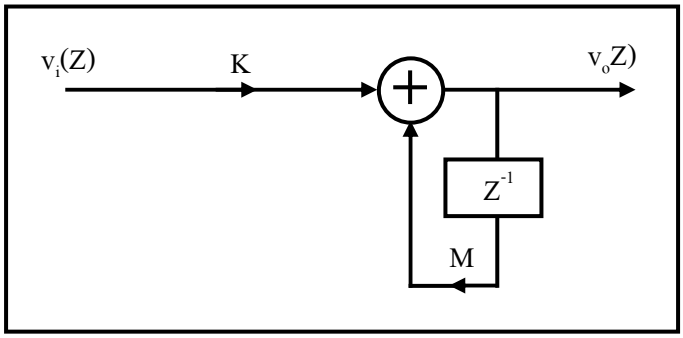
กำหนดให้

$$K = \frac{aT}{(1 + aT)}, \quad M = \frac{1}{(1 + aT)}$$

$$Vo(n) = KVi(n) + MVo(n - 1) \tag{2-53}$$

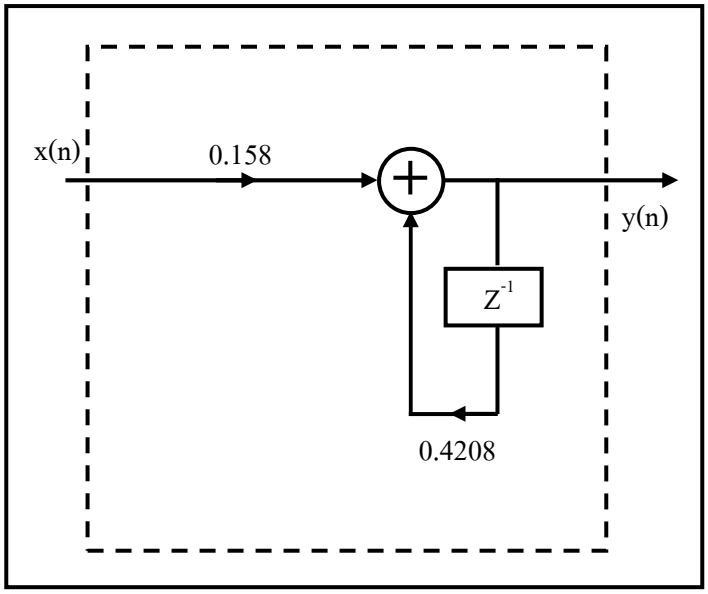
$$Vo(z) = KVi(z) + MVo(z)Z^{-1} \tag{2-54}$$

และจากสมการที่ 2-54 นำไปเขียนเป็นโครงสร้างได้ดังภาพที่ 2-19



ภาพที่ 2-19 แสดงโครงสร้างตัวกรองดิจิตอลความถี่ต่ำผ่าน 1 ลำดับ

และจากสมการที่ 2-54, Vo(z) นำไปเขียนเป็นโครงสร้างได้ดังภาพที่ 2-20



ภาพที่ 2-20 แสดงโครงสร้างของตัวกรองดิจิตอลแบบไอไออาร์ 1 ลำดับที่ถูกสร้างขึ้นด้วยวิธีประมาณค่าเบี่ยงเบน

2.11 การออกแบบตัวกรองดิจิตอลแบบไอไออาร์ด้วยวิธีการแปลงเชิงเส้นคู่

การแปลงฟังก์ชั่นในเอสโดเมนไปสู่แซดโดเมนด้วยวิธีเชิงเส้นคู่สามารถกระทำได้โดยใช้กฎของแท็บปิซอยด์ดอล (Trapezoidal) พิจารณาตัวกรองแบบอนาลอก ที่มีฟังก์ชั่นถ่ายโอน H(s) ดังสมการที่ 2-55

$$H(s) = \frac{b}{S+a} \tag{2-55}$$

จากฟังก์ชั่นถ่ายโอนตามสมการที่ 2-55 นี้สามารถจัดให้อยู่ในรูปสมการเชิงอนุพันธ์ได้ดังสมการที่ 2.56

$$\frac{dy(t)}{dt} + ay(t) = bx(t) \tag{2-56}$$

แทนอนุพันธ์ในสมการ 2-56 และประมาณค่าด้วยแท็บปิซอยด์ดอลได้ดังสมการที่ 2-57

$$y(t) = \int_{t0}^{t} y^{t}(\tau)d\tau + y(t0) \tag{2-57}$$

เมื่อ y(t) แทนอนุพันธ์ของ y(t) การประมาณค่าของการอินทิกรัล ในสมการที่ 2-56 ด้วยกฎของแท็บปิซอยด์ดอลที่ $t = nT$ และ $t_0 = nT\text{-}T$ จะได้

$$y(nT) = \frac{T}{2} \times [y^t(nT) + y^t(nT - T)] + y(nT - T) \tag{2.-58}$$

ดังนั้นถ้าแทน $t = nT$ ในสมการเชิงอนุพันธ์ที่ 2-56 จะได้

$$y^t(nT) = -ay(nT) + bx(nT) \tag{2-59}$$

นำสมการที่ 2-56 แทนลงในสมการที่ 2-58 และแทน $y(n) = y(nT), x(n) = x(nT)$ จะได้

$$(1 + \frac{aT}{2})y(n) - (1 - \frac{aT}{2})y(n-1) = \frac{bT}{2}[\times(n) + x(n-1)] \tag{2-60}$$

ใช้การแปลงแซด (Z-Transform) เปลี่ยนสมการผลต่าง (difference equation) ในสมการที่ 2-59 จะได้

$$(+\frac{aT}{2})Y(z) - (1 - \frac{aT}{2})Z^{-1}Y(z) = \frac{bT}{2}(1 + Z^{-1})X(z)$$

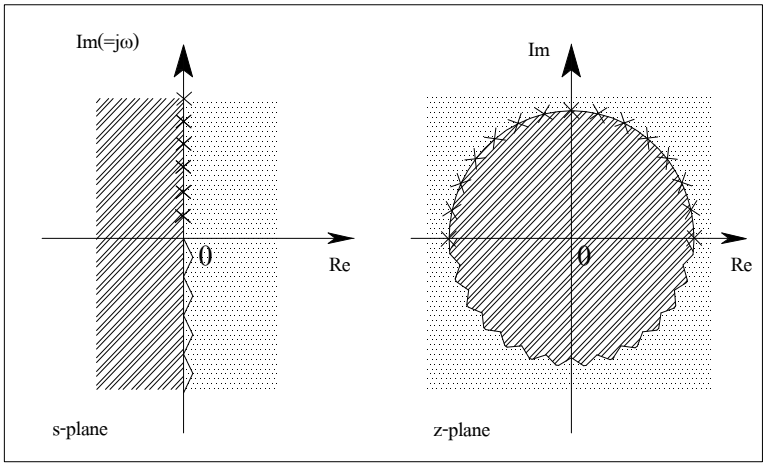
$$H(z) = \frac{Y(z)}{X(z)} = \frac{(bT/2)(1 + Z^{-1})}{1 + (aT/2) - (1 - aT/2)Z^{-1}}$$

$$H(z) = \frac{b}{\frac{2}{T} \times \frac{\left(1 - Z^{-1}\right)}{\left(1 + Z^{-1}\right)} + a} \tag{2-61}$$

เทียบสัมประสิทธิสมการที่ 2-55 กับสมการที่ 2-61 จะได้

$$S = \frac{2(1 - Z^{-1})}{T(1 + Z^{-1})} \tag{2-62}$$

ในการแมพจากระนาบเอสไปสู่ระนาบแซดด้วยวิธีแปลงเชิงเส้นคู่ จะพบว่าค่าที่อยู่ทางขวาของระนาบเอสจะไปอยู่นอกวงกลมหนึ่งหน่วยบนระนาบแซดค่าที่อยู่ทางซ้ายของระนาบเอสจะไปอยู่ภายในวงกลมหนึ่งหน่วยและค่าที่อยู่บนแกนจินตภาพ จะไปอยู่บนเส้นรอบวงของวงกลมหนึ่งหน่วยบนระนาบแซด ดังแสดงดังภาพที่ 2-21



ภาพที่ 2-21 แสดงความสัมพันธ์ระหว่างค่าที่อยู่บนระนาบเอสกับค่าที่อยู่บนระนาบแซดและจากสมการ

$$Z = re^{jwT} \tag{2-63}$$

$$Z = re^{jwT}$$
$$S = \sigma + j\Omega \tag{2-64}$$

$$S = \frac{2}{T} \times \frac{\left(1 - Z^{-1}\right)}{\left(1 + Z^{-1}\right)}$$

$$S = \frac{2}{T} \times \frac{\left(1 - Z\right)}{\left(1 + Z\right)} \tag{2-65}$$

$$S = \frac{2}{T} \times \frac{\left(re^{jwT} - 1\right)}{\left(re^{jwT} + 1\right)} \tag{2-66}$$

$$S = \frac{2}{T} \times \left(\frac{r^2 - 1}{1 + r^2 + 2r\cos wT} + j\frac{2r\sin wT}{1 + r^2 + 2r\cos wT} \right) \tag{2-67}$$

เมื่อเทียบสัมประสิทธิ์สมการที่ 2-63 กับสมการที่ 2-67 จะได้

$$\sigma = \frac{2}{T} \times \frac{r^2 - 1}{1 + r^2 + 2r\cos wT} \quad (2\text{-}68)$$

$$\Omega = \frac{2}{T} \times \frac{2r\sin wT}{1 + r^2 + 2r\cos wT} \quad (2\text{-}69)$$

จากสมการที่ 2-68 และสมการที่ 2-69 จะพบว่าถ้า $r < 1$, $\sigma < 0$ และถ้า $r > 1$, $\sigma > 0$ นั่นก็หมายความว่าค่าที่อยู่ทางซ้ายของระนาบเอส จะไปอยู่ภายในวงกลมหนึ่งหน่วยในระนาบเซต และค่าที่อยู่ทางขวาของระนาบเอส จะอยู่นอกวงกลมหนึ่งหน่วยในระนาบแซด และถ้า $r = 1$, $\sigma = 0$ ค่าที่อยู่บนแกนจินตภาพบนระนาบเอสจะไปอยู่บนเส้นรอบวงของวงกลมรัศมีหนึ่งหน่วยบนระนาบแซด และเมื่อ $r = 1$ สมการที่ 2-67 จะได้ Ω มีค่าดังสมการที่ 2-70 และ 2-71

$$\Omega = \frac{2}{T} \times \frac{\sin wT}{1 + \cos wT} \quad (2\text{-}70)$$

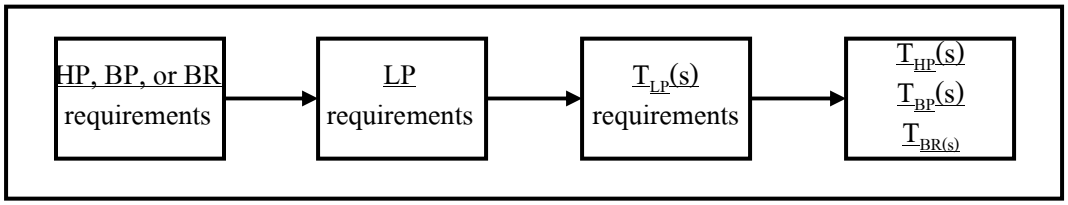
$$\Omega = \frac{2}{T} \tan \frac{wT}{2} \quad (2\text{-}71)$$

สรุป ขั้นตอนการออกแบบตัวกรองดิจิตอบแบบไอไออาร์ด้วยวิธีแปลงเชิงเส้นแบบคู่

1. ออกแบบตัวกรองความถี่แบบอนาลอกโดยการหาฟังก์ชั่นถ่ายโอน H(s)
2. หาความถี่คัท-ออฟหรือความถี่ขอบแถบผ่านของตัวกรองความถี่แบบดิจิตอล
3. หาค่า $\Omega = \frac{2}{T} \times \tan \frac{wT}{2}$
4. ทำสเกลความถี่ (Frequency scaling) H(s) โดยแทนค่า $S = \frac{S}{\Omega}$
5. หาค่า H(z) โดยแทนค่า S ตามสมการที่ 2-61 โดยเปรียบเทียบดูค่า S จากสมการที่ 2-62

2.12 การแปลงความถี่ (Frequency Transformations)

ในการสร้างตัวกรองชนิดอื่น ๆ เช่น ตัวกรองความถี่สูงผ่าน, ตัวกรองช่วงความถี่ผ่าน, ตัวกรองช่วงความถี่ไม่ผ่าน สามารถกระทำได้ตามไดอะแกรม ดังภาพที่2-22



ภาพที่2-22 แสดงขั้นตอนการออกแบบตัวกรองความถี่สูงผ่าน,ตัวกรองความถี่ผ่านและตัวกรองช่วงความถี่หยุดด้วยการแปลงความถี่

การแปลงความถี่ของตัวกรองความถี่สูงผ่าน (High-pass Filter) จะแทนค่า $S = \frac{\Omega_c}{S}$ ลงใน H(s) ของตัวกรองความถี่ต่ำผ่าน (Low-pass Filter) การแปลงความถี่ของตัวกรองความถี่ต่ำผ่าน การแปลงความถี่ของตัวกรองช่วงความถี่หยุด (Brand-stop Filter)

จะแทนค่า $S = \frac{\Omega_B S}{S^2 + \Omega_o^2}$

ลงใน H(s) ของตัวกรองความถี่ต่ำผ่าน เมื่อ $\Omega_o = \Omega_L \Omega_H$ และ $\Omega_B = \Omega_H \Omega_L$

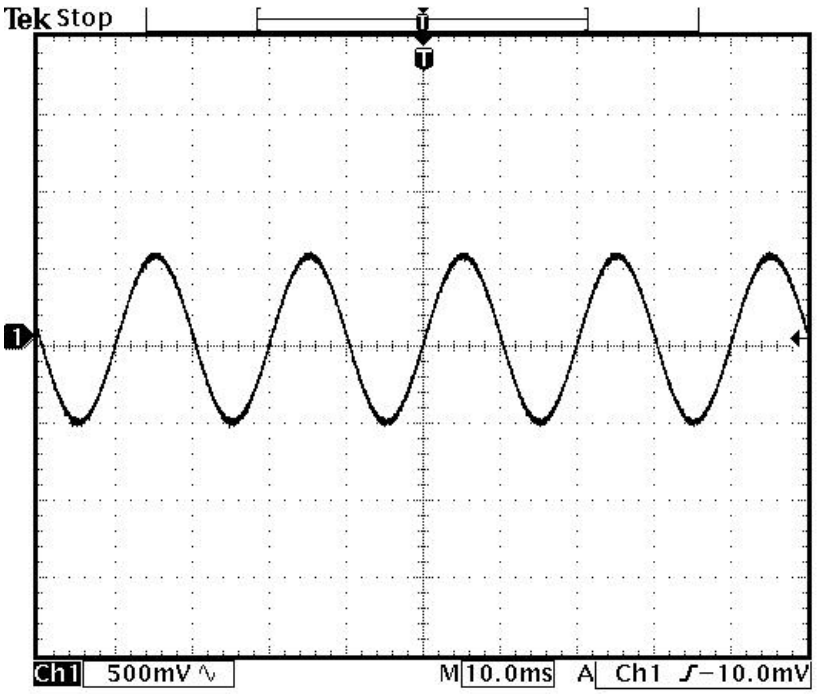
บทที่ 3
อุปกรณ์และวิธีการทดลอง

3.1 การสร้างสัญญาณรบกวนแบบปรับตัวเอง

การสร้างสัญญาณรบกวนแบบปรับตัวเองเราจะใช้รูปแบบของสมการไซน์(sine) ซึ่งจะมีโครงสร้างของสมการสามารถหาค่าสัมประสิทธิ์ได้ดังนี้

```
W = 2*pi*Fsin;
A = 2*cos((w*p));
B = 1.0;
C = sin((w*p));
Y = A*y1-B*y2+C*imp;
y2 = y1;
y1 = y;
```

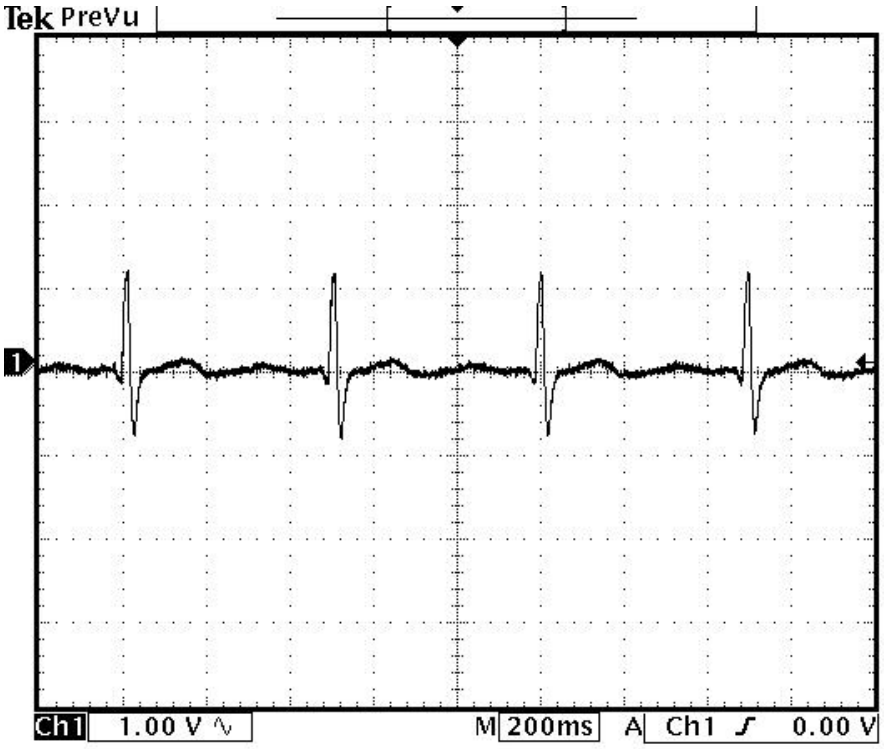
เมื่อกำหนดโปรแกรมลงบนภาษาซีแสดงผลผ่านบอร์ด TMS320C3X จะได้คลื่นสัญญาณอยู่ในรูปของไซน์ (sine)



ภาพที่ 3–1 สัญญาณรบกวน

3.2 การสร้างคลื่นไฟฟ้าหัวใจจากเครื่องวัดคลื่นไฟฟ้าหัวใจจำลอง

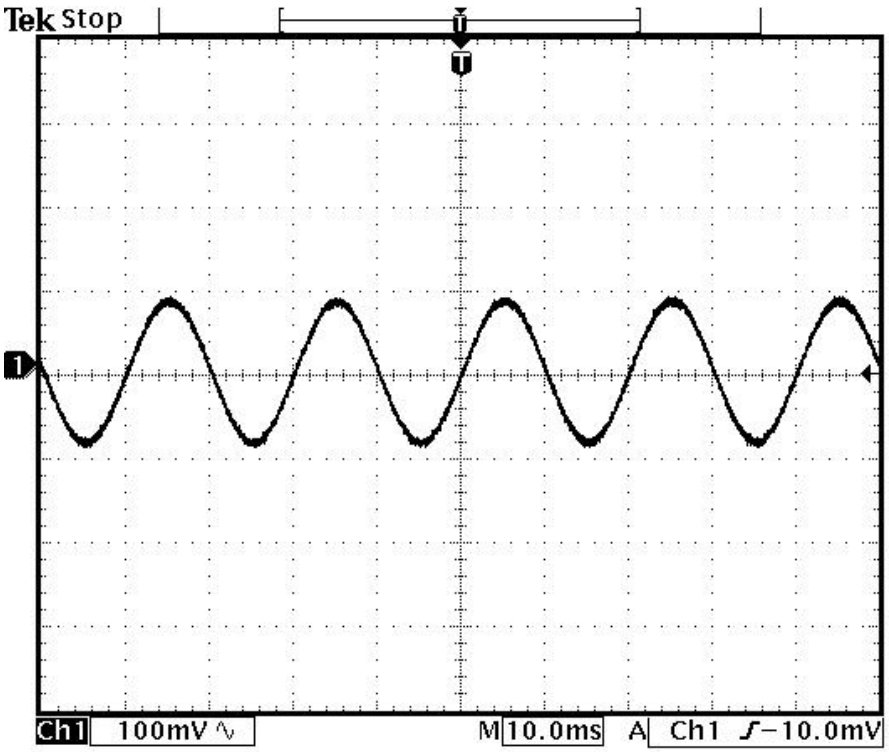
เราจะสร้างคลื่นไฟฟ้าหัวใจจากเครื่องวัดคลื่นไฟฟ้าหัวใจจำลอง โดยกำหนดให้อัตราการเต้นของหัวใจเท่ากับ 60 ครั้ง/นาที จะได้สัญญาณแสดงผลดังรูป



ภาพที่ 3–2 แสดงผลการวัดคลื่นไฟฟ้าหัวใจจำลอง

3.3 การสร้างสัญญาณรบกวนจากเครื่องกำเนิดสัญญาณ

เราจะสร้างสัญญาณรบกวนจากเครื่องกำเนิดสัญญาณในรูปแบบสัญญาณไซน์ (sine) ซึ่งจะนำสัญญาณรบกวนนี้ไปรวมกับคลื่นไฟฟ้าหัวใจจำลอง โดยใช้การต่อวงจรแบบ Summing จะได้สัญญาณดังรูป



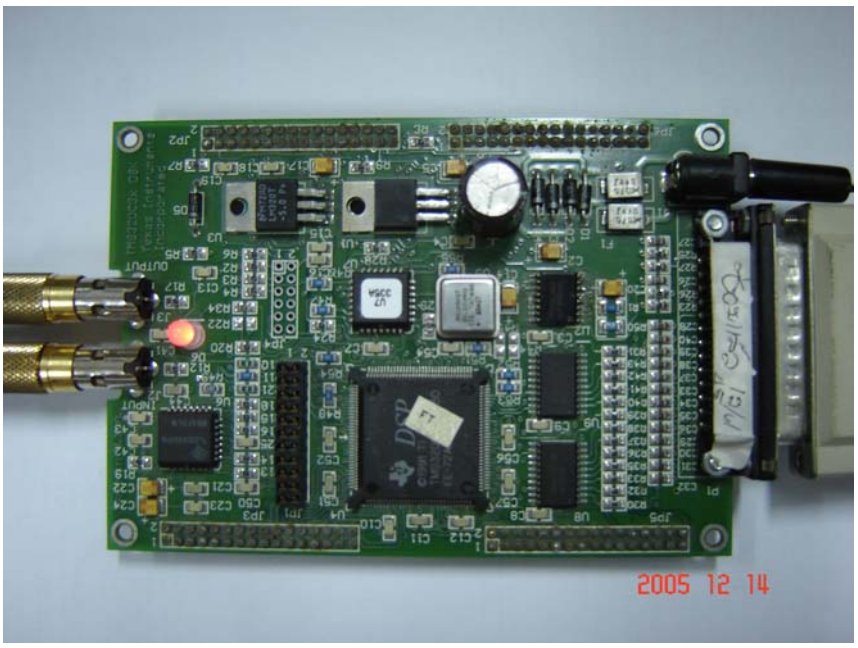
ภาพที่ 3–3 สัญญาณรบกวนจากเครื่องกำเนินสัญญาณ

3.4 สรุปรูปแบบของการทดลอง

1. สร้างสัญญาณรบกวนจากเครื่องกำเนิดสัญญาณ
2. สร้างคลื่นไฟฟ้าหัวใจจากเครื่องกำเนิดคลื่นไฟฟ้าหัวใจจำลอง
3. ต่อสัญญาณจากข้อ 1 และข้อ 2 เข้าสู่วงจร summing
4. ต่อสัญญาณจากวงจร summing เข้าสู่ input ของบอร์ด TMS3210C3X
5. สร้างสัญญาณ nose referenc ในโปรแกรมภาษาซี
6. ใช้รูปแบบของสมการ Adaptive Filter อาศัยอัลกอริธึมแบบลีสมีนแสควร์ โดยนำค่า error มาปรับเวทเวกเตอร์เพื่อจะไปปรับ nose referenc

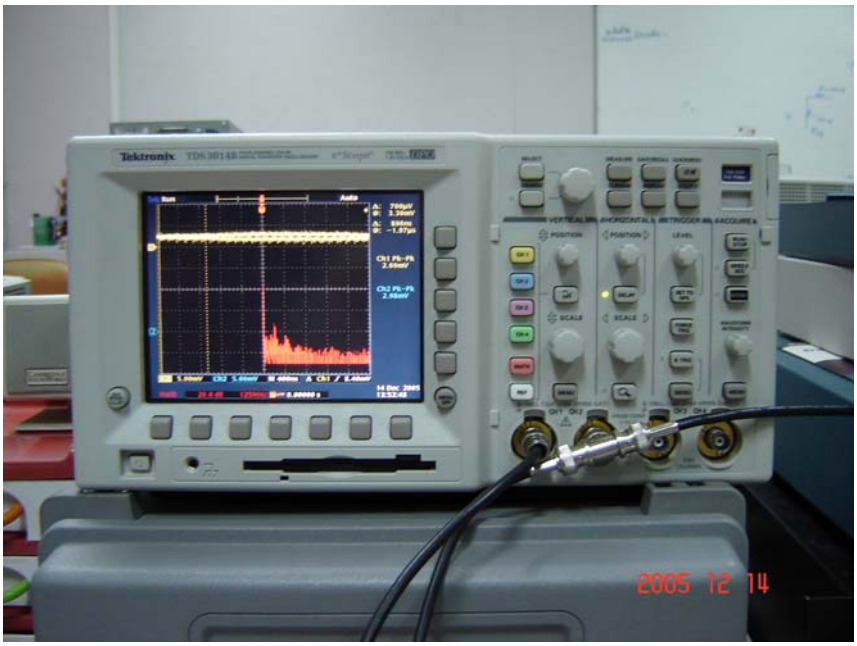
3.5 อุปกรณ์การทดลอง

อุปกรณ์ที่ใช้ในการทดลองแสดงได้ดังรูปและมีรายละเอียดตามรายการดังต่อไปนี้

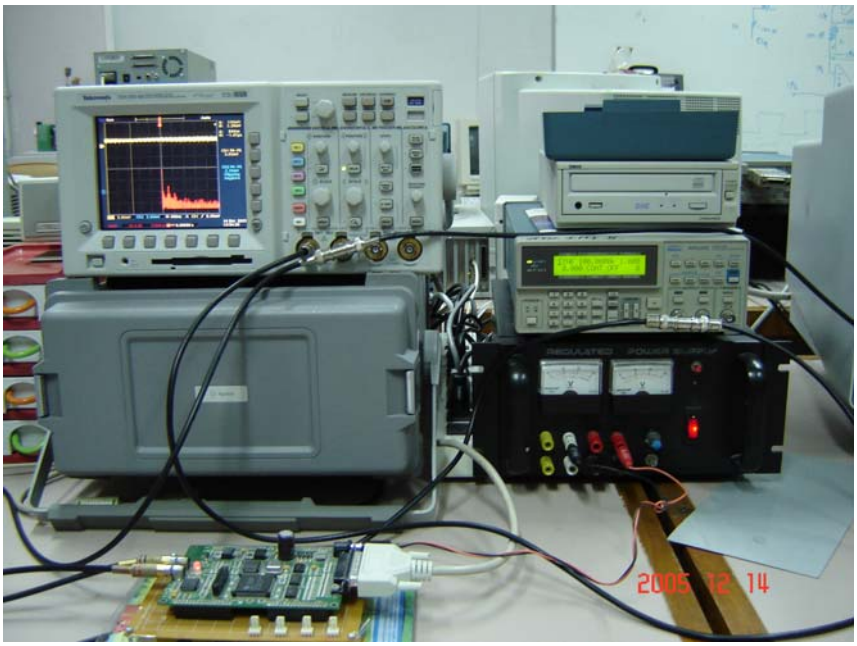


ภาพที่ 3-4 บอร์ด TMS320C3X

ภาพที่ 3–5 แหล่งจ่ายไฟ (Power Supply)กับเครื่องกำเนิดสัญญาณ 1 เครื่อง



ภาพที่ 3–6 ออสซิลโลสโคป (Oscilloscope)



ภาพที่ 3-7 การต่ออุปกรณ์ทดลอง

สรุป อุปกรณ์ที่ใช้ในการทำการทดลอง

1. คอมพิวเตอร์ตั้งโต๊ะ 2 เครื่อง
2. เครื่องกำเนิดสัญญาณ 1 เครื่อง
3. ออสซิลโลสโคป (Oscilloscope)
4. บอร์ด TMS320C3X
5. แหล่งจ่ายไฟ (Power Supply)
6. สายสัญญาณ 6 เส้น
7. วงจร Summing 1 ชุด
8. เครื่องวัดคลื่นไฟฟ้าหัวใจจำลอง

บทที่ 4

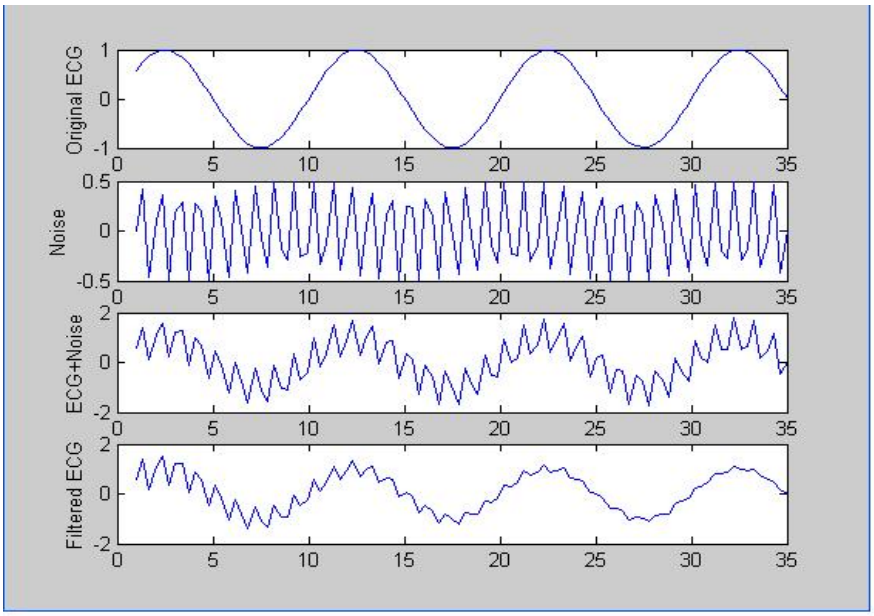
ผลการทดลอง

ในการทดลองได้ทำการออกแบบและสร้างโปรแกรมการลดทอนสัญญาณรบกวนแบบปรับตัวเอง โดยมีค่าความถี่ที่แตกต่างกันออกไป และความแตกต่างของค่าขนาดชั้น μ (Step Size) จะมีผลต่อเสถียรภาพและความเร็วในการลู่เข้า ซึ่งมีผลต่อความสามารถในการติดตามสัญญาณของตัวกรองด้วยการทดลองได้แบ่งออกเป็น 2 ส่วน

1. การออกแบบและสร้างตัวกรองสัญญาณเชิงตัวเลขแบบปรับตัวเองโดยโปรแกรม Matlab

2. การออกแบบและสร้างตัวกรองสัญญาณเชิงตัวเลขแบบปรับตัวเองโดยอาศัยโปรแกรมภาษาซีประมวลผลผ่านบอร์ด TMS320C3X

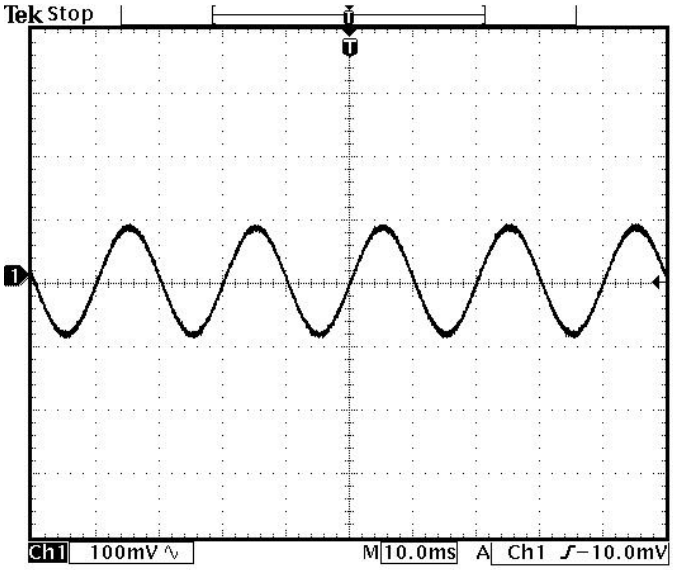
การออกแบบที่ 1 ใช้การแสดงผลผ่านโปรแกรม Matlab กำหนดสัญญาณที่ต้องการ(Signal) 50 Hz และให้สัญญาณที่รบกวนมีความถี่สูง(Noise), $\mu = 0$



ภาพที่ 4-1 การแสดงผลผ่านโปรแกรม Matlab

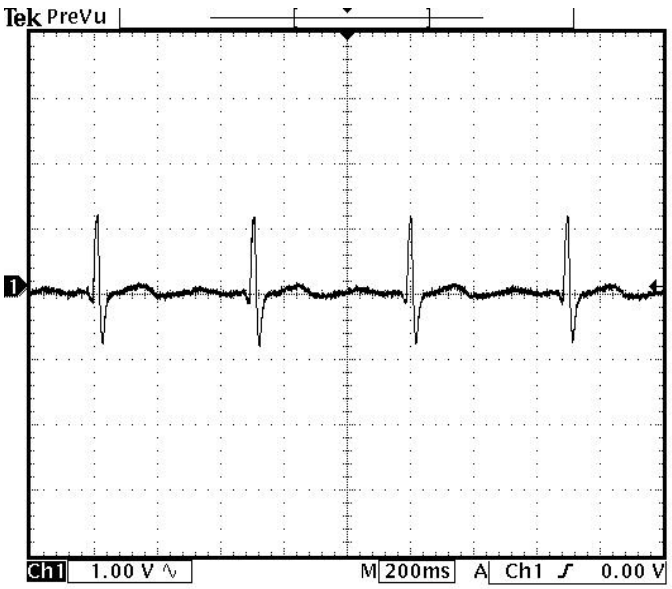
เมื่อเราดูผลการทดลองที่ 1 สัญญาณที่ต้องการมีความถี่ 50 Hz (กราฟ 1) ถูกสัญญาณรบกวนที่มีความถี่สูง (กราฟ 2) เข้ามารบกวนปะปนกลายเป็นสัญญาณที่ต้องการรวมกับสัญญาณรบกวน (กราฟ 3) แต่เมื่อถูกการกรองด้วยวิธีการของตัวกรองสัญญาณเชิงตัวเลขแบบปรับตัวเองสัญญาณที่ต้องการจะค่อยๆเพิ่มความคมชัดจนเข้าใกล้รูปสัญญาณที่เราต้องการ

ในการทดลองต่อไปเราจะให้สัญญาณรบกวนเป็นความถี่ในย่านความถี่ต่ำ (Low Frequency : Hz) และสัญญาณที่เราต้องการเป็นสัญญาณคลื่นไฟฟ้าหัวใจ (ECG)ดังตัวอย่างสัญญาณรบกวนที่ใช้ความถี่ 50 Hz



ภาพที่ 4-2 สัญญาณรบกวน 50Hz

สัญญาณที่ต้องการคลื่นไฟฟ้าหัวใจ

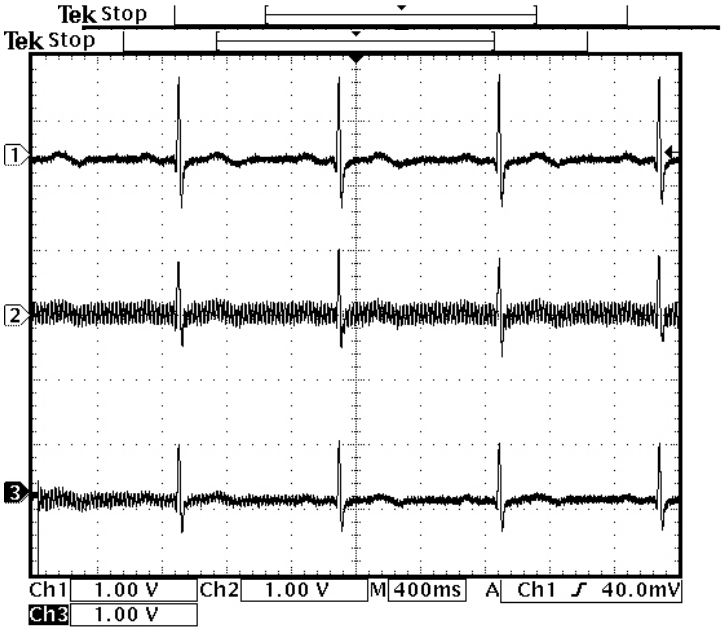


ภาพที่ 4-3 สัญญาณคลื่นไฟฟ้าหัวใจ (ECG)

การดูกราฟจากผลการทดลองต่อไปคือแบบที่ 2 และแบบที่ 3 โดยลักษณะของกราฟบน (หมายเลข 1)จะเป็นกราฟคลื่นไฟฟ้าหัวใจ, กราฟส่วนกลาง(หมายเลข 2)จะเป็นกราฟที่เกิดจากการรบกวนโดยในที่นี้เราใช้สัญญาณรบกวนความถี่ต่ำไปกระทำต่อสัญญาณคลื่นไฟฟ้าหัวใจ, กราฟล่าง (หมายเลข 3)เป็นกราฟที่เกิดจากการลดทอนด้วยตัวกรองเชิงเลขแบบปรับตัวเองผ่านบอร์ด TMS320C3X

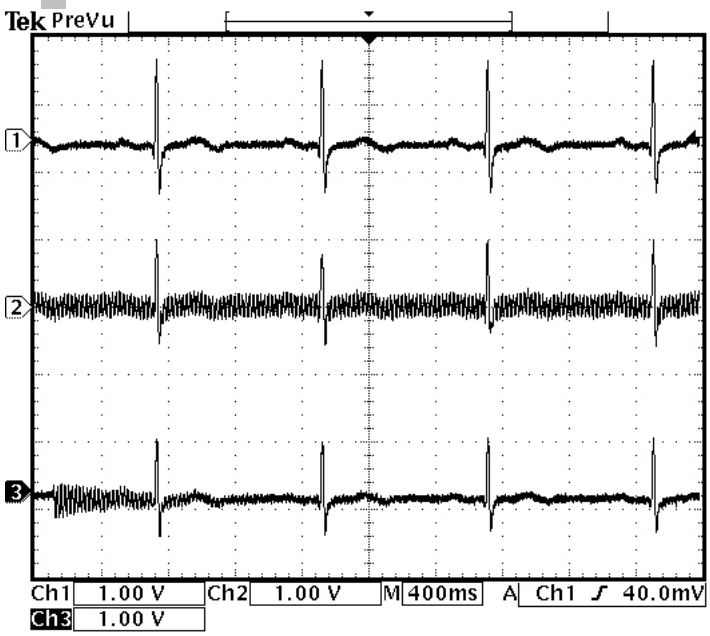
การออกแบบที่ 2 ใช้การแสดงผลผ่านโปรแกรม Matlab กำหนดสัญญาณรบกวน 45 **Hz** ให้ μมีค่าต่าง ๆ

μ = 0.001 เวลาในการลู่เข้า (Time) = 1.79 s



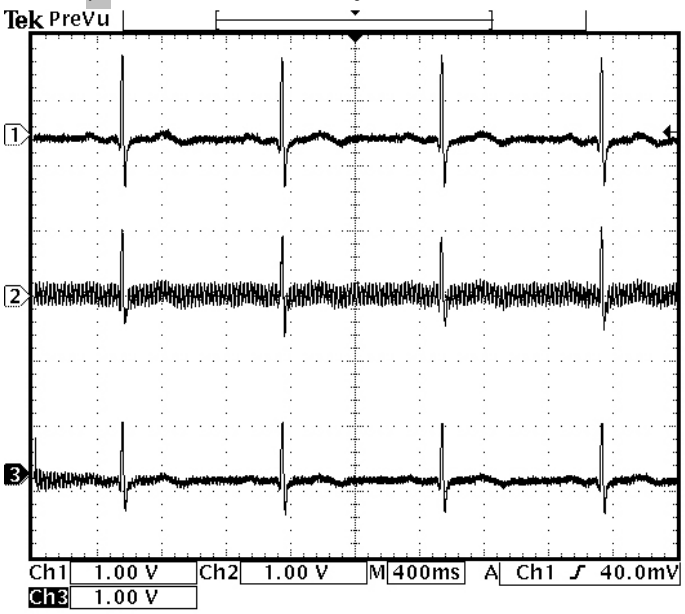
ภาพที่ 4-4 การแสดงผลผ่านบอร์ด TMS320C3X ความถี่ 45 Hz μ = 0.001

μ= 0.002 เวลาในการลู่เข้า (Time) = 1.32 s



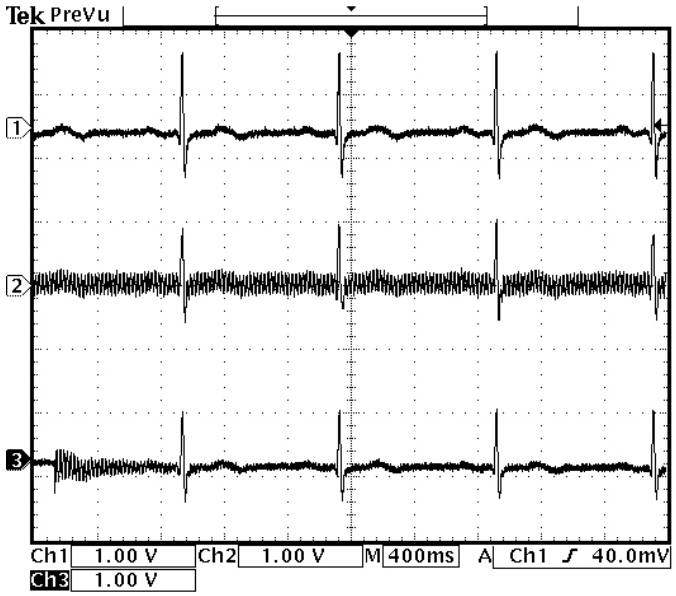
ภาพที่ 4-5 การแสดงผลผ่านบอร์ด TMS320C3X ความถี่ 45 Hz μ = 0.002

μ= 0.003 เวลาในการลู่เข้า (Time) = 1.02 s



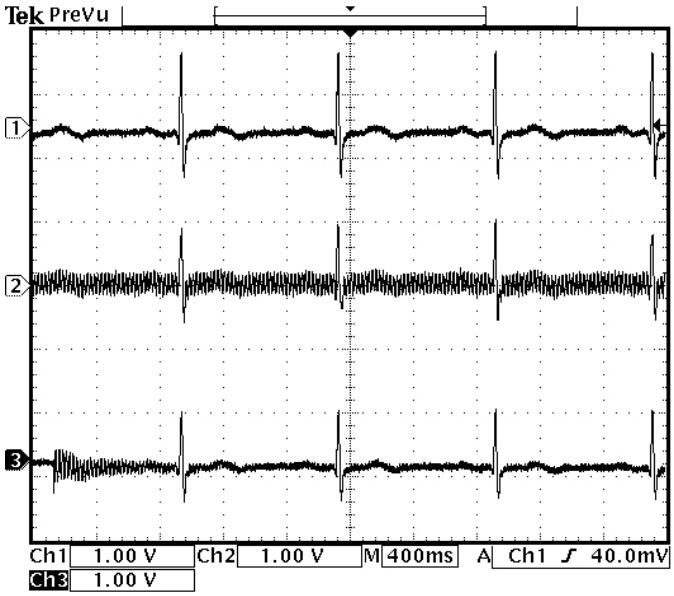
ภาพที่ 4-6 การแสดงผลผ่านบอร์ด TMS320C3X ความถี่ 45 Hz μ = 0.003

μ= 0.004 เวลาในการลู่เข้า (Time) = 856



ภาพที่ 4-7 การแสดงผลผ่านบอร์ด TMS320C3X ความถี่ 45 Hz **μ = 0.004**

μ = 0.005 เวลาในการลู่เข้า (Time) = 720 ms



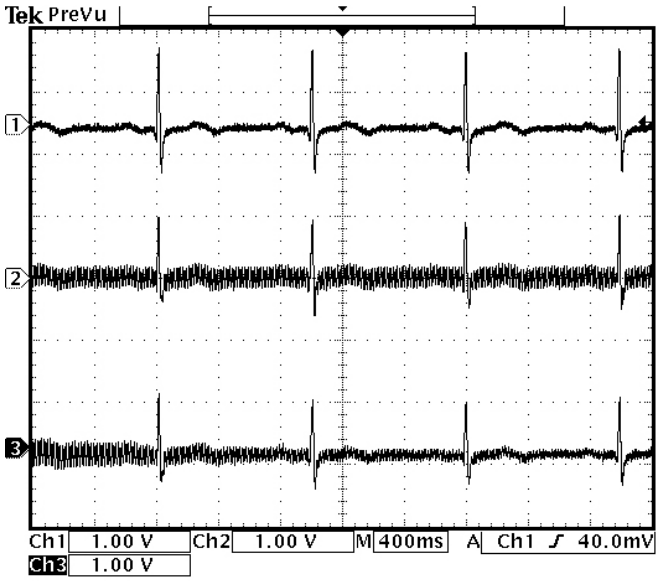
ภาพที่ 4-8 การแสดงผลผ่านบอร์ด TMS320C3X ความถี่ 45 Hz μ = 0.005

เมื่อดูจากผลการทดลองค่าของ μ (Step Size) มีค่าเพิ่มขึ้นจะทำให้อัตราการลู่เข้ามีค่าลดลงเป็นไปตามทฤษฎี $t = \frac{1}{\mu\lambda \min}$

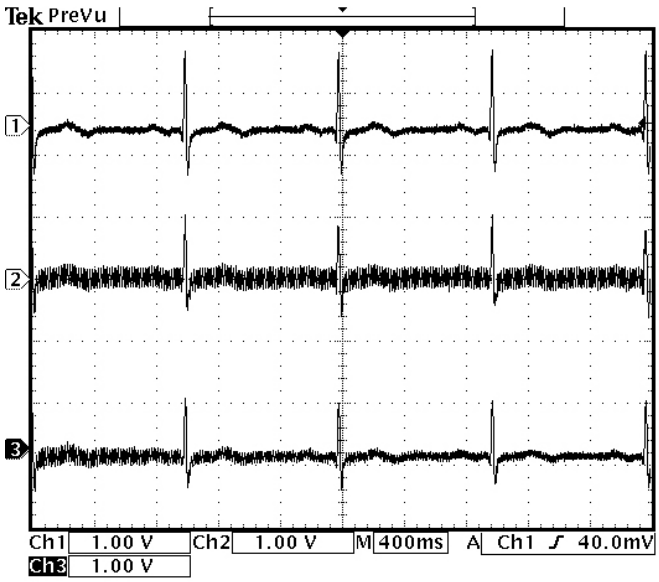
ตารางที่ 4-1 การแสดงผลการเปรียบเทียบการออกแบบที่ 2 ความถี่ (f) 45 Hz แต่ค่าขนาดขั้น (μ) มีค่าต่าง ๆ กัน ดูเวลาการลู่เข้า

ความถี่ ,f (Hz)	ขนาดขั้น (μ)	เวลาการลู่เข้า
45	0.001	1.79 s
	0.002	1.32 s
	0.003	1.02 s
	0.004	856 ms
	0.005	720 ms

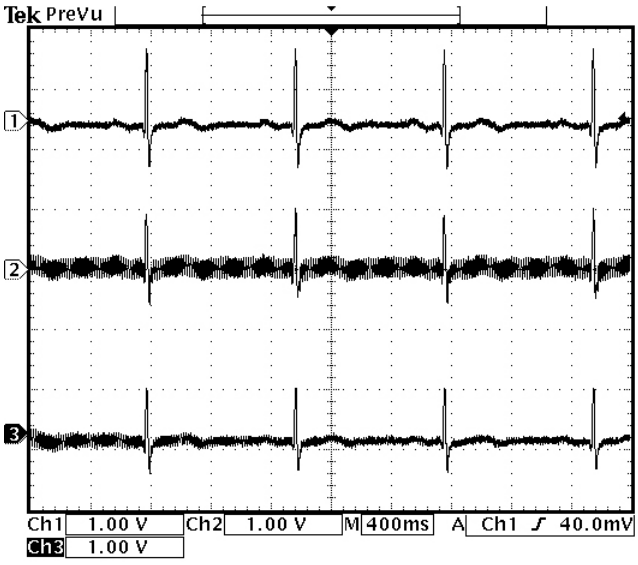
การออกแบบที่ 3 ใช้การแสดงผลผ่านโปรแกรม Matlab บอร์ด TMS320C3X ให้ความถี่(F)มีค่าต่าง ๆ กัน โดยไม่มีการเปลี่ยนแปลงค่าขนาดขั้น (μ) ให้ค่า μ = 0.001 ทุก ๆ ค่าความถี่ของแต่ละการทดลองเพื่อดูผลการลดทอนและเวลาในการลู่เข้า



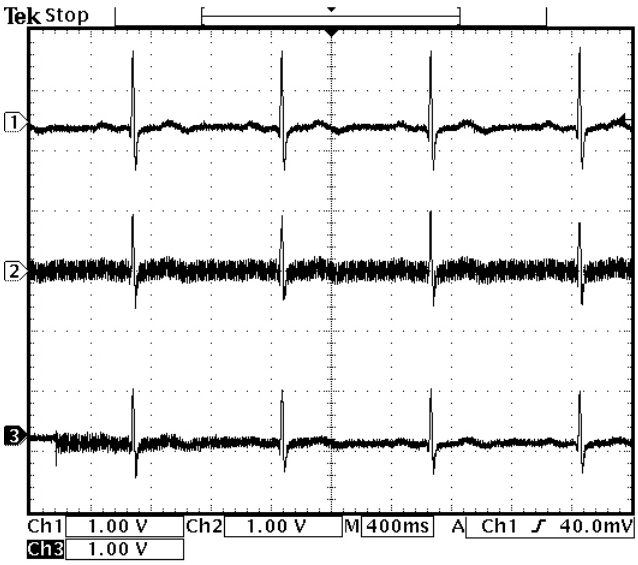
ภาพที่ 4-9 การแสดงผ่านบอร์ด TMS320C3X ความถี่ 50 Hz, μ = 0.001 เวลาในการลู่เข้า 2.78s



ภาพที่ 4-10 การแสดงผ่านบอร์ด TMS320C3X ความถี่ 55 Hz, μ = 0.001 เวลาในการลู่เข้า 2.50s



ภาพที่ 4-11 การแสดงผ่านบอร์ด TMS320C3X ความถี่ 60 Hz, μ = 0.001 เวลาในการลู่เข้า 2.54s



ภาพที่ 4-12 การแสดงผ่านบอร์ด TMS320C3X ความถี่ 70 Hz, μ = 0.001 เวลาในการลู่เข้า 2.56s

ตารางที่ 4-2 การแสดงผลการเปรียบเทียบการออกแบบที่ 3 ค่าความถี่ (f) ที่มีค่าต่าง ๆ กัน แต่ค่าขนาดขั้น (μ) มีค่าเดียวกัน

ความถี่,f (Hz)	ขนาดขั้น (μ)	เวลาการลู่เข้า
50	0.001	2.78 s
55	0.001	2.50 s
60	0.001	2.54 s
70	0.001	2.56s

สรุป เมื่อเราดูตารางผลการทดลองจากการทดลองแบบที่ 2 และแบบที่ 3 เราจะเห็นได้ว่าตัวกรองสัญญาณเชิงตัวเลขแบบปรับตัวเองจะค่อยๆทำการลดทอนสัญญาณรบกวนจนเข้าใกล้หรือเหมือนสัญญาณที่เราต้องการ โดยการลดทอนสัญญาณเร็วหรือช้า(เวลาในการลู่เข้า)ขึ้นอยู่กับค่าสัมประสิทธิ์, μ โดยถ้าค่า μ มีค่ามากเวลาในการลู่เข้าจะมีค่าน้อยนั่นคือในการลดทอนจะทำได้เร็ว และถ้าค่า μ มีค่าน้อยเวลาในการลู่เข้าจะมีค่ามากนั่นคือในการลดทอนจะทำได้ช้า

บทที่ 5
สรุปผลการทดลอง

5.1 สรุปผลการทดลอง

จากผลการทดลองการลดทอนคลื่นไฟฟ้าหัวใจจำลองด้วยตัวกรองสัญญาณเชิงตัวเลขแบบปรับตัวเอง พบว่าอัลกอริธึมของสมการลีสท์มีนแสควร์โครงสร้างแบบตรงมีอัตราการลู่เข้าช้าที่สุด เมื่อเปรียบเทียบกับอัลกอริธึมของรีเคอชีฟลีสท์แสควร์ (RLS) และฟาสท์รีเคอชัฟลีสท์แสควร์ (FRLS) แต่เสถียรภาพของอัลกอริธึมลีสท์มีนแสควร์จะดีที่สุดซึ่งอัตราการลู่เข้าจะขึ้นอยู่กับค่าของขนาดขั้น (Step Size) μ ดังที่กล่าวมาแล้ว นอกจากนี้ยังสามารถนำรูปแบบของสมการ Adaptive Filter ไปประยุกต์ใช้กับการลดทอนสัญญาณในรูปแบบต่างๆ ได้

5.2 ข้อเสนอแนะ

ข้อจำกัดในการทดลองการลดทอนคลื่นไฟฟ้าหัวใจทดลองด้วยตัวกรองสัญญาณเชิงตัวเลขแบบปรับตัวเองของบอร์ด TMS320C3X มีหน่วยความจำ RAM และจำนวนรีจิสเตอร์น้อย จึงทำให้ผลการทดลองมีความยุ่งยากและอุปกรณ์ที่ใช้มีความเสถียรน้อยทำให้ผลการทดลองออกมา คาดเคลื่อน ดังนั้นถ้าอุปกรณ์ดังกล่าวมีความสามารถประสิทธิภาพผลการทดลองจะออกมาได้เด่นชัดยิ่งขึ้น

เอกสารอ้างอิง

1. Parks T.W. and Burrus C.S.. Digital Filter Design. USA : JOHN WILLY & SOPNG, 1987.
2. Proakis G. John, and Manolakis G. Dimitris. Digital Signal Processing. USA : McGraw-hill, 1986.
3. Haykin Simon. Adaptive Filter Theory. USA : Prentice – hall, 1986.
4. TMS320C3X User's Guide. USA : Texas Instrument Corporate, 1994.
5. Sawitree K. and Sukanya P. Implementation of Multi – Narrow Band Digital Filter. THAI, 2002.
6. Rulph C. DSP Applications Using C and the TMS320C6X DSP. USA : JOHN WILLY & SOPNG, 2002.

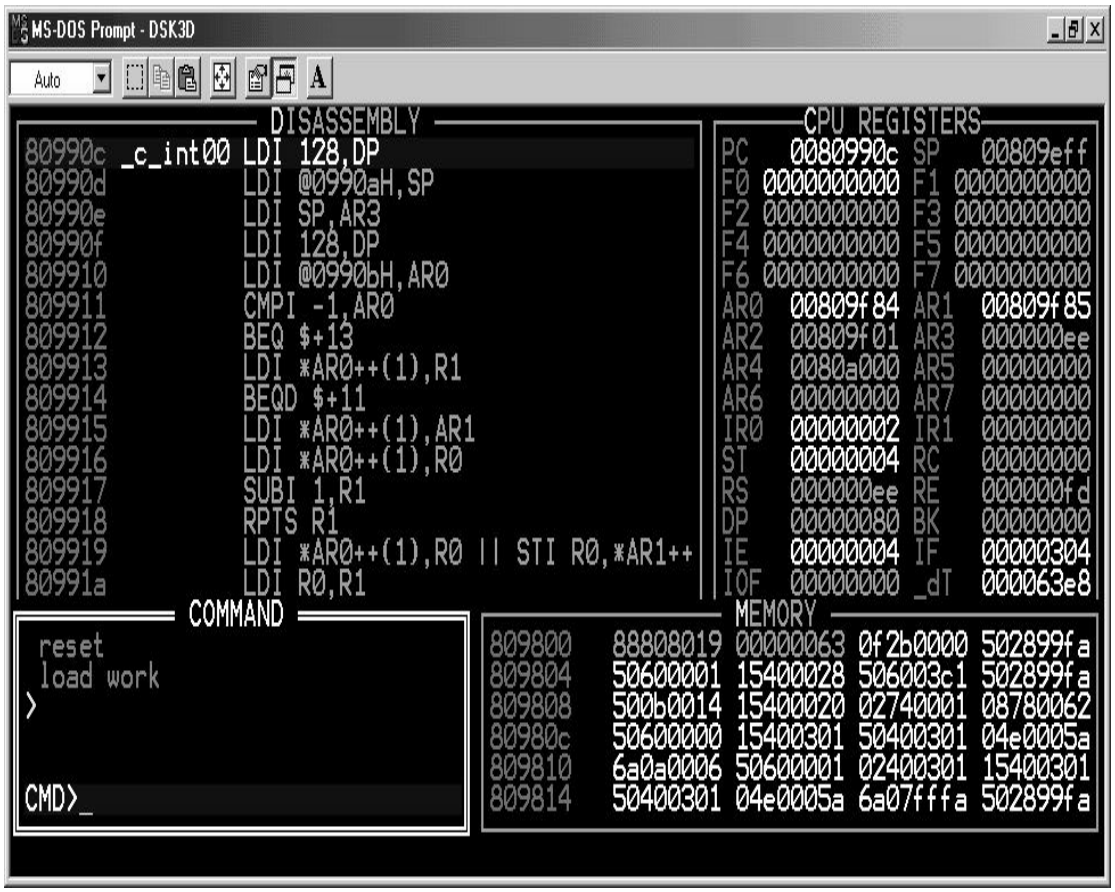
ภาคผนวก ก

การใช้งานบอร์ด TMS320C31 DSP STARTER KIT

การใช้งานบอร์ด TMS320C31 DSP STARTER KIT

ก-1 ขั้นตอนการใช้งานบอร์ด TMS320C31 DSP STARTER KIT มีดังนี้

1. ต่อบอร์ด TMS320C31 DSK เข้ากับPort Printerของเครื่องไมโครคอมพิวเตอร์
2. จ่ายไฟ 10-12 Vdc เข้ากับบอร์ด TMS320C31 DSK
3. สร้าง Sub-Directory ด้วยคำสั่ง md c:\c3xtools
4. Copy โปรแกรมของ TMS320C31 DSK ด้วยคำสั่ง copy d:*.* c:\c3xtools
5. Rum คำสั่ง c:\c3xtools\dsk3d ถ้าไม่มีข้อผิดพลาดบนจอภาพของเครื่องไมโครคอมพิวเตอร์จะแสดงดังภาพที่ 1



ภาพที่ ก-1 แสดงการ RUN โปรแกรม DSK3D เป็นไปอย่างถูกต้อง

หน้าต่างแสดง Debugger ที่พร้อมทำงาน

ภาพที่ ก-2 แสดงหน้าต่าง Debugger ที่พร้อมทำงาน

ภาพที่ ก-3 หน้าต่างแสดงการโหลด DSK3d

ก-2 การเรียกใช้หน้าต่าง Debugger

คำสั่งที่เรียกใช้หน้าต่าง Debugger

Dsk3d [option]

Dsk3d เป็นคำสั่งที่เรียกใช้ หน้าต่าง Debugger

Option เป็นการเพิ่มรายละเอียดของ Debugger

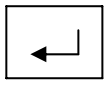
ตารางที่ 1 แสดงรายการ Option ของ Debugger โดยในตารางจะอธิบาย Option บาง Option ที่เรียกใช้บ่อย

ตารางที่ ก-1 แสดง Optiont ของ Debugger

Option	อธิบายสรุป
? หรือ HELP	การแสดงรายการของ opion
AUTO	ค้นหาอัตโนมัติถ้า พอร์ตขนานสนับสนุนโมด 8 บิต หรือ 4 บิต
BW=4,Nibbie unindirectional	การสื่อสารใช้พอร์ตขนานแบบมาตรฐาน 4 บิต mode
BW=5, Byte unindirectional	การสื่อสารใช้พอร์ตขนานแบบมาตรฐาน 8 บิต mode
LPTx, LPT=x	เลือกพอร์ตพรินเตอร์ขนาน (LPT1 เป็น defauft)
PORT=Ox378	เลือก address ของ พอร์ตต่างๆ
RESET	รีเซต DSK (cold boot)
TEST	ค้นหาอัตโนมัติทั้ง LPT1 , LPT2 และLPT3 เพื่อติดต่อกับ DSK
T=xx	เพิ่มบัส xx I/O ไซเคิลพิเศษในการส่งแต่ละครั้งสำหรับสายที่ยาวหรือมี nois
WIN=1	การจัดการหน้าต่าง Time Slice ให้ทำงาน
WIN=0	การจัดการหน้าต่าง Time Slice ให้หยุดการทำงาน และ เคลียร์อินเตอร์รัฟท์

การแสดงรายการของ option (? หรือ HETP option)

สามารถที่จะดู option ที่แสดงในตารางที่ 1 โดยใช้ ? หรือ HETP ตัวอย่าง

DSK3D ? 

การเลือกพอร์ตพรินเตอร์ขนาน (LET = 3 หรือ LET#option)

LPT Option จะเลือกพอร์ตพรินเตอร์ขนานจากการติดต่อของ DSK กับ host

พอร์ตพรินเตอร์ขนาน	หน้าที่
LPT1 หรือ LPT = 1	คัดเลือกพอร์ตพรินเตอร์ที่ I/O address 0x378
LPT2 หรือ LPT = 2	คัดเลือกพอร์ตพรินเตอร์ที่ I/O address 0x278
LPT3 หรือ LPT = 3	คัดเลือกพอร์ตพรินเตอร์ที่ I/O address 0x3BC

หมายเหตุ สำหรับเครื่อง ESSA และ IBM PS/2s ใช้ LPTx ที่แตกต่างกัน

AT Convertion	EISA และ PS/2	I/O Address
LPT1	LPT2	Ox378
LPT2	LPT3	Ox278
LPT3	LPT1	Ox3BC

การเลือกพอร์ตพรินเตอร์ที่จะใช้ (พอร์ต option)

พอร์ต option เลือกพอร์ตพรินเตอร์ขนานใน Address ที่มีให้ดังตัวอย่าง

Port = Ox378

หมายความว่า เลือกพอร์ตพรินเตอร์ขนานของ host ใน Address Ox378

การค้นหาพอร์ตพรินเตอร์อัตโนมัติ (TEST option)

ใช้ test option ไปค้นหาในระบบเพื่อหาพอร์ตขนานที่จะติดต่อกับ DSK

หมายเหตุ ถ้ามีพอร์ตพรินเตอร์หรือการติดต่ออื่นๆ ไปยัง PC คุณจะต้องหยุดการทำงานก่อนใช้ test option

ก-3 การใช้หน้าต่าง Debugger

ก-3.1 หน้าต่าง DISASSEMBLY แสดงการจองพื้นที่หน่วยความจำของ assembly ในหน้าต่างนี้จะแสดงบรรทัดที่มากมายของ code ซึ่งแต่ละบรรทัดจะแสดงให้เห็นถึงโครงสร้างของตำแหน่ง, โครงสร้าง opcode, ชื่อ และโครงสร้างของ mnemonic บรรทัดที่แสดงแถบสว่างและเป็นโครงสร้างต่อไปที่ถูกดำเนินการ

ในการเลือกหน้าต่าง DISASSEMBLY โดยกด ALT+D ขณะที่ในหน้าต่าง DISASSEMBLY คุณสามารถใช้เคอร์เซอร์เลือกบรรทัดและใช้ function key เพื่อตั้งหรือเคลียร์ break point

ก-3.2 หน้าต่าง CPU รีจิสเตอร์แสดงให้เห็นรีจิสเตอร์ที่มีอยู่ใน CPU ดังที่เห็นในภาพที่ 7 ตามปกติแล้วค่าที่บรรจุในรีจิสเตอร์จะเป็นเลขฐาน 16 สามารถ F3 ค่ารีจิสเตอร์จะเป็นทแบบ Floating-point-decimail ถ้ากด F2 ค่าภายในรีจิสเตอร์จะเป็นแบบ 40 บิต (hexadeadimal)

การเปลี่ยนแปลงค่ารีจิสเตอร์ทำโดยกด ALT+C สามารถเขียนค่าที่แถบข้อมูลที่เป็นแถบสีและกด ENTER เพื่อยอมรับการเปลี่ยนแปลง เมื่อแน่ใจและใช้ key ดังนี้เพื่อคลิก data ที่ต้องการแก้ไข

ก-3.3 หน้าต่าง MEMORY แสดงให้เห็นช่วงของ memory ดังในภาพที่ 8 ในหน้าต่างจะมี 2 ส่วน คือ

- Address ในคอลัมแรกจะเป็นเลขของตำแหน่งแรกในหน้าต่างของคอลัมข้อมูล ตำแหน่งที่คอลัมข้อมูลในหน้าจอจะมีตำแหน่งเดียวกับตำแหน่งของคอลัม แต่ละตำแหน่งในคอลัมจะแสดงถึงลักษณะของข้อมูลที่ถูกต้อง
- Data ค่าที่มีอยู่ในคอลัมจะแสดงค่าให้เห็นในตำแหน่งนั้น

ตัวอย่างหน้าต่าง memory มีข้อมูล 4 คอลัม ดังนั้นในแต่ละตำแหน่งเริ่มต้นค่าจะเพิ่มขึ้นทีละ 4 ถึงแม้หน้าต่างจะแสดงข้อมูลเพียง 4 คอลัมแต่มันยังคงเป็นเพียงตำแหน่งของคอลัมเดียว เช่นที่ตำแหน่ง 0x00809800 มีค่าเท่ากับ , ที่ตำแหน่ง 0x00809801 มีค่าเท่ากับ 0xFFFFFFFC ที่ตำแหน่ง 0x00809804 (ค่าแรกในแถวที่ 2) มีค่าเท่ากับ 0x0080982C , ที่ตำแหน่ง 0x00809805มีค่าเท่ากับ 0x00809839 เป็นต้น

การเปลี่ยแปลงค่าของหน้าต่าง memory สามารถทำได้โดยกด ALT+M จึงจะสามารถเขียนข้อมูลทับได้ การเลือก cell สามารถใช้ key

ก-3.4 หน้าต่าง COMMAND จะมีพื้นที่สำหรับส่งผ่านคำสั่ง ตอบรับคำสั่งที่ผิดพลาดและข้อความ Error หน้าต่าง COMMAND มี 2 ส่วนคือ

- Command line ส่วนนี้มีไว้ป้อนคำสั่ง เมื่อต้องการส่งผ่านคำสั่ง

- ส่วนแสดงผล เป็นส่วนตอบรับคำสั่งที่ป้อนเข้ามาหรือแสดง output อื่นๆ จากคำสั่ง และแสดงข้อความ Error

ตารางที่ ก-2 การแก้ไขคำสั่ง

สิ่งที่ได้	ใช้ Command ต่อไปนี้
เลื่อนไปตลอดคำสั่ง	← →
แทรก และเขียนทับ	INS
ลบตัวอักษรที่ตำแหน่งเคอร์เซอร์	DEL
เลื่อนไปยังบรรทัดเริ่มต้น	HOME
เลื่อนไปยังบรรทัดสุดท้าย	END
เคลียร์คำสั่ง	ESC
เลือกคำสั่งจาก Buffer	↑ ↓

ก-4 การใช้เมนูช่วยเหลือ

สามารถกด F1 หรือปุ่ม H เพื่อให้หน้าต่างช่วยเหลือเปิดขึ้นมาแสดงให้เห็นดังภาพที่ 10 เลือกจากรายการเมนูข้างล่างในการหารายละเอียดเพิ่มเติม

การเลื่อนไปยังหน้าต่าง Help สามารถใช้

- PGUT เลื่อนไปยังรายการด้านบน
- PGDN เลื่อนกลับไปยังรายการเดิม

- END ไปยังรายการสุดท้ายของ Help Menu
- S เซฟ Help Text เป็นไฟล์
- ESC ออกจาก Help Menu และกลับไปยัง Debugger
- H เพื่อผ่านไปยัง Help อันดับที่ 2 ส่วนใหญ่จะเกี่ยวกับฮาร์ดแวร์และการใช้คำสั่งเฉพาะของ Debugger

ค่ำสั่ง Debugger

ในตารางจะเป็นการสรุปปุ่ม Function และคำสั่งของ Debugger

ตารางที่ ก-3 คำสั่งแก้ไขในบรรทัด

สิ่งที่ได้	ใช้คำสั่งต่อไป
เลื่อนเคอร์เซอร์ไปยังจุดเริ่มต้นของ Command	HOME

line	
เลื่อนเคอร์เซอร์ไปยังจุดสุดท้ายของ Command line	END
สิ่งที่ได้	ใช้คำสั่งต่อไปนี้
ลบตัวอักษรด้านซ้ายของเคอร์เซอร์	DEL
ลบตัวอักษรด้านขวาของเคอร์เซอร์	SHIFT+END
เลื่อนเคอร์เซอร์ไปด้านซ้าย	←
เลื่อนเคอร์เซอร์ไปด้านขวา	→

ตารางที่ ก-4 Command-Line Buffer Manipulation

สิ่งที่ได้	ใช้คำสั่งต่อไปนี้
เรียกคำสั่งสุดท้าย	PAGE UP หรือ ↑
เรียกคำสั่งแรกใน Command Line Buffer อีกด้วย	PAGE DOWN หรือ↓
ปฏิบัติในคำสั่งสุดท้ายอีกครั้ง	TAM

ตารางที่ ก-5 การสั่งงานโปรแกรม

สิ่งที่ได้	ใช้คำสั่งต่อไปนี้
ปฏิบัติทีละโครงสร้างใน Single-step	SS
ปฏิบัติทีละ n โครงสร้าง	XN n
ปฏิบัติจนถึงโครงสร้างที่ถูกกำหนดใน addr ใน Single-step	XG addr
ปฏิบัติโปรแกรมจนถึง breakpoint	RUN
ปฏิบัติโปรแกรมและละเลย breakpoint run-free	RUNG

ตารางที่ ก-6 การแสดงผลและการเปลี่ยนแปลงข้อมูล

สิ่งที่ได้	ใช้คำสั่งต่อไปนี้
การแสดงค่าที่อยู่สในหน่วยความจำที่ addr ในหน้าต่าง memory	MEM addr
การเปลี่ยนแปลงค่าในหน่วยความจำที่ addr	MM addr
ใส่ค่า lang ไปในหน่วย่ความจำโดยเริ่มต้นที่ addr ด้วย val เป็นค่า floating-poin เฉพาะจะถูกเปลี่ยนแปลงไปเป็น ค่า floating-poin ของ TMS320	MM addr leng val
แสดงภาษา Assembly ที่ addr ในหน้าต่าง DISASSEMBLY	DASM addr
แสดงรีจิสเตอร์ extended-precision แบบ 40 บิต ในหน้าต่าง Register	REG40
แสดงรีจิสเตอร์ extended-precision แบบ floating-poin ในหน้าต่าง Register	FLOAT
การเปลี่ยนแปลง reg ในหน้าต่าง CPU REGISTER กับค่าจาก expression ดังตัวอย่าง PC=0X809800 RO=1.34	reg = expression

ตารางที่ ก-7 การจัดการ Breakpoint

สิ่งที่ได้	ให้คำสั่งต่อไปนี้
เซต Breakpoint ที่ addr	SB addr
เคลียร์ Breakpoint ที่ addr	CB addr
เคลียร์ทุก Breakpoint	CB
แสดงรายการ Breakpoint ทั้งหมดที่ถูกเซต	DB

ตารางที่ ก-8 การโหลดโปรแกรม

สิ่งที่ได้	ใช้คำสั่งต่อไปนี้
โหลดไฟล์	LOAD filename
โหลดสัญลักษณ์	SLOAD filename
โหลดไบนารี	BLOAD filename
เคลียร์สัญลักษณ์	SCLEAR

ตารางที่ ก-9 Performing System Tasks

สิ่งที่ได้	ใช้คำต่อไปนี้
รีเซท DSK	RESET
ออกจาก Debugger	QUIT , EXIT
ออกไป DOS และปฏิบัติตามคำสั่ง, และพิมพ์ EXIT เพื่อกลับมาที่ Denugger	DOS (Expression to run)
ออกไปยัง DOS และทำการแก้ไขไฟล์ (ถ้าไม่มีไฟล์ให้แก้ไขจะโหลดไฟล์ปัจจุบันมาใช้)	EDIT filename
ออกไปยัง DOS และแปลง DSK แอสเซมเบอไปเป็นไฟล์แอสเซมเบอ	Dsk3a filename.asm

Quick Referance Guide

ตารางที่ ก-10 แสดงปุ่ม Shortcuts ฟังก์ชันสำหรับหน้าต่าง DISASSEMBLY

ปุ่มฟังก์ชั่น	คำอธิบาย
F1	หน้าจอ Hetp
F2	เซต Breakpoint ที่เคอร์เซอร์
F3	เคลียร์ Breakpoint ที่เคอร์เซอร์
F4	Run ที่เคอร์เซอร์
F5	Run
F6	หน้าจอ Breakpoint
F7	เคลียร์ Breakpoint ทั้งหมด
F8	Run โปรแกรมทีละขั้น

F9	Grow wimdows
F10	Step over
SHIFT+F9	เลือกหน้าต่าง DISASSEMBLY
ESC หรือ ENTER	Ecsape

ตารางที่ ก-11 แสดงปุ่ม Shortcuts ฟังก์ชั่นสำหรับหน้าต่าง CPU

ปุ่มฟังก์ชั่น	คำอธิบาย
F1	หน้าจอ Help
ESC	ออกจากหน้าต่าง CPU
HOME	เลื่อนขึ้นด้านบน
END	เลื่อนลงด้านล่าง
	เลื่อน cell ในแนวตั้ง
TAM	เลื่อน cell ในแนวนอน

ตารางที่ ก-12 แสดงปุ่ม Shortcuts ฟังก์ชั่นสำหรับหน้าต่าง MEMORY

ปุ่มฟังก์ชั่น	คำอธิบาย
F1	หน้าจอ Help
F9	Toggle windows size
ESC	ออกจากหน้าต่าง MEMORY
HOME	เลื่อนขึ้นด้านบน
END	เลื่อนลงด้านล่าง
PAGE UP , PAGE DOWN	เลื่อนหน้าขึ้นหรือลง
↑ , ↓	เลื่อน cell ในแนวตั้ง
TAP	เลื่อน cell ในแนวนอน

ตารางที่ ก-13 แสดงปุ่ม Shortcuts ฟังก์ชั่นสำหรับหน้าต่าง COMMAND

ปุ่มฟังก์ชั่น	คำอธิบาย
F1	แสดงรายการคำสั่ง
F2	รีจิสเตอร์ extended-precistion ฐาน 16 แบบ 40 บิต
F3	รีจิสเตอร์ extended-precistion ฐาน 10 แบบ floating-point
F4	Toggle ระหว่างการแสดงไฟล์ source และหน่วยความจำ DISASSEMBLY
F5	Run โปรแกรมจนถึง Breakpoint ต่อไป
F6	แสดง Breakpoint ทั้งหมด
F7	เคลียร์ Breakpoint ทั้งหมด
F8	Run โปรแกรมทีละขั้น
F9	Toggle the DISASSEMBLY windows size
F10	Run โปรแกรมทีละขั้นและกลับไปขั้นตอนที่ผ่านมาด้วย
ALT+D	เลือกหน้าต่าง DISASSEMBLY
ALT+M	เลือกหน้าต่าง MEMORY
ALT+C	เลือกหน้าต่าง COMMAND
ESC	ออกจากหน้าต่างที่ทำงาน

ภาคผนวก ข

โปรแกรมภาษาซี

โปรแกรมภาษาซี

ในการทดลองเมื่อเราได้ทดสอบรูปแบบของสมการจาก Matlab แล้วจะใช้ภาษาซี ในการทดลองโดยต้องแสดวงผลออกทางบอร์ด TMS3201C3X ดังนั้น เนื้อหาของตัวโปรแกรมภาษาซี จะอธิบายในส่วนของคำสั่งและวิธีการทำงานในการแสดงผลการทดลองนี้

ข-1 ส่วนประกอบของโปรแกรมภาษาซี

ภายในโปรแกรมภาษา C แบ่งออกเป็น 2 ส่วน ประกอบด้วยกันคือ ส่วนหัวของโปรแกรม (Head) และส่วนของตัวโปรแกรม (Body) โดยรายละเอียดของแต่ละส่วนประกอบ

ข-1.1 ส่วนหัวของโปรแกรมจะเริ่มตั้งแต่บรรทัดแรกของโปรแกรมจนมาสิ้นสุดที่บรรทัดก่อน main () ส่วนหัวของโปรแกรมมีไว้เพื่อเขียนคำสั่งพิเศษบางอย่างที่ต้องการให้ทำงานก่อนที่จะเข้าสู่การทำงานของตัวโปรแกรม นอกจากนี้ยังเป็นส่วนที่ใช้ในการสร้างและกำหนดข้อมูลที่จะนำไปใช้ในตัวโปรแกรมอีกด้วย ดังนั้นภายในส่วนหัวของโปรแกรมจะประกอบด้วยพรีโปรเซสเซอร์ไดเร็คทีฟ (Preprocessor directive) และส่วนของการกำหนดข้อมูล

ข-1.2 พรีโปรเซสเซอร์ไดเร็คทีฟ (Preprocessor directive)พรีโปรเซสเซอร์ไดเร็คทีฟ คือคำสั่งรูปแบบหนึ่งของภาษา C ที่มีความพิเศษ โดยในขั้นตอนการแปลความหมายโปรแกรม ถ้าตัวแปลภาษา C ตรวจพบว่ามีการใช้พรีโปรเซสเซอร์ภายในโปรแกรม พรีโปรเซสเซอร์ไดเร็คทีฟเหล่านั้นจะถูกแปลความหมายเป็นลำดับแรกก่อนคำสั่งประเภทอื่น ๆ

รูปแบบของการเขียนพรีโปรเซสเซอร์ไดเร็คทีฟ จะต้องขึ้นต้นด้วยเครื่องหมาย # แต่ไม่ต้องลงท้ายด้วยเครื่องหมาย : เหมือนคำสั่งอื่นทั่วไป โดยคำสั่งที่จัดอยู่ในกลุ่มของพรีโปรเซสเซอร์ไดเร็คทีฟ แสดงได้ดังนี้

#include	#define	#error	#if	#endif
#elif	#else	#ifdef	#ifndef	#lundef
#line	#pragma			

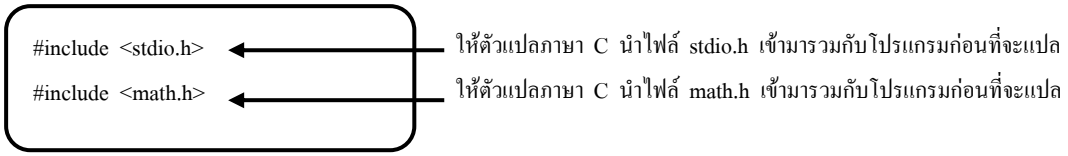
พรีโปรเซสเซอร์ไดเร็คทีฟบางคำสั่ง ไม่จำเป็นต้องเขียนไว้ที่ส่วนหัวของโปรแกรม อย่างเช่น #elf, #undef ซึ่งขึ้นอยู่กับการทำงานของพรีโปรเซสเซอร์ไดเร็คทีฟนั้น และความต้องการของผู้เขียนโปรแกรมว่าจะเรียกใช้พรีโปรเซสเซอร์ไดเร็คทีฟในขั้นตอนใด

#include

พรีโปรเซสเซอร์ไดเร็คทีฟ #include จะใช้สำหรับสั่งให้ตัวแปลภาษา C นำไฟล์ที่กำหนดชื่อไว้ต่อจาก #include เข้ามารวมกับโปรแกรมก่อนที่จะทำการแปลโปรแกรม เนื่องจากในบางกรณีที่มีการเรียกใช้คำสั่งจากไฟล์อื่น ดังนั้นเราจึงต้องเขียน #include ไว้ที่ส่วนหัวของโปรแกรมเสมอ

การเขียน พรีโปรเซสเซอร์ไดเร็คทีฟ #include สามารถทำได้ 2 รูปแบบ ดังนี้

แบบที่ 1 #include <ชื่อไฟล์>



การใช้เครื่องหมาย < > ระบุชื่อไฟล์ เพื่อให้ตัวแปลภาษา C เริ่มค้นหาไฟล์จากไดเร็คทอรีที่กำหนดไว้ก่อน (สำหรับ Turbo C++ จะเป็นไดเร็คทอรี include) ถ้าไม่พบจะกลับมาค้นหาต่อที่ไดเร็คทอรีปัจจุบัน (ซึ่งก็คือไดเร็คทอรีเดียวกับที่บันทึกไฟล์โปรแกรมไว้)

แบบที่ 2 #include “ชื่อไฟล์”

```
#include “stdio.h”
#include “math.h”
```

ปัจจุบันก่อน ถ้าไม่พบจะไปค้นหาต่อในไดเร็คทอรีที่กำหนดไว้

#define

ก่อนที่จะพูดถึงหน้าที่ของพรีโปรเซสเซอร์ไดเร็คทีฟ #define เราจำเป็นจะต้องรู้จักกับคำว่า มาโคร (Macro) ในภาษา C เสียก่อน

ข-1.3 มาโคร (Macro)มาโครเป็นชื่อที่เราสร้างขึ้นมาภายในโปรแกรม พร้อมกับทำการกำหนดค่าหรือความหมายให้กับมาโครนั้น โดยมีข้อกำหนดว่าชื่อของมาโครต้องเป็นตัวอักษรตัวใหญ่ และค่าที่กำหนดให้กับมาโครสามารถเป็นได้ตั้งแต่ตัวเลข ข้อความ หรือคำสั่งที่ได้ผลลัพธ์ออกมาแน่นอน (เช่น การคำนวณ) ยกตัวอย่างเช่น สร้างมาโครชื่อ NUMBER กำหนดค่าให้เป็น 15 หรือมาโครชื่อ NAME กำหนดค่าให้เป็นข้อความ “Boonserm”

เมื่อมีการเรียกใช้มาโครที่สร้างขึ้นที่ตำแหน่งใดก็ตามภายในโปรแกรม ตัวแปลภาษา C จะนำค่าของมาโครไปแทนที่ตำแหน่งนั้น ประโยชน์ของการใช้งานมาโครที่เห็นได้ชัดเจนก็คือ ในกรณีที่เรามีข้อมูลซึ่งต้องเรียกใช้บ่อยครั้งในโปรแกรม การสร้างมาโครสำหรับข้อมูลนั้นขึ้นมาใช้งานจะเหมาะสมและสะดวกกว่าการที่ต้องเขียนข้อมูลนั้นหลาย ๆ ครั้ง ซึ่งอาจจะเกิดความผิดพลาดขึ้นได้

#define ใช้สำหรับสร้างมาโครและกำหนดค่าให้กับมาโครนั้น รูปแบบการเขียนพรีโปรเซสเซอร์ไดเร็คทีฟ #define

โดยส่วนใหญ่พรีโปรเซสเซอร์ไดเร็คทีฟ #define จะเขียนไว้ที่ส่วนหัวของโปรแกรม เพื่อให้มาโครเหล่านั้นสามารถนำไปใช้ได้ตลอดทั้งโปรแกรม

ข-2 ตัวแปรและหน้าที่ของตัวแปร

เมื่อเราเตรียมข้อมูลไว้พร้อมแล้ว การจะนำข้อมูลเข้ามาใช้ในโปรแกรม เราต้องทำให้ตัวแปลภาษา C รู้จักข้อมูลเหล่านั้นเสียก่อนจึงจะใช้งานได้ ซึ่งวิธีการก็คือ การสร้างตัวแปรสำหรับข้อมูลเหล่านั้นขึ้นมา

ตัวแปร (Variable) ก็คือ การจองที่เก็บข้อมูลในหน่วยความจำหลัก (RAM) ของเครื่องคอมพิวเตอร์ พร้อมกับกำหนดชื่อเรียกแทนหน่วยความจำในตำแหน่งนั้น เปรียบเทียบได้กับการจองห้องที่มีเลขที่ประจำห้องให้กับข้อมูล เวลาจะใช้งานข้อมูลใดก็ให้เรียกผ่านชื่อของตัวแปร อย่างเช่น ถ้าเราสร้างตัวแปรขึ้นมา 1 ตัวโดยใช้ชื่อว่า num สำหรับเก็บค่าตัวเลข 16 เมื่อต้องการนำจำนวน 16 มาใช้งานเราก็เพียงแต่เรียกชื่อ num ซึ่งตัวแปลภาษา C จะแปลความหมายได้ถูกต้องว่า num คือการนำค่าตัวเลข 16 ที่เก็บไว้ในหน่วยความจำมาใช้งาน

ข-3 ชนิดของตัวแปรในภาษา C

ตัวแปรในภาษา C สามารถแบ่งได้เป็น 2 ประเภทใหญ่ ๆ คือ ตัวแปรพื้นฐาน (Scalar) ซึ่งหมายถึงตัวแปรที่เก็บข้อมูลได้เพียงค่าเดียว และตัวแปรชุด (Array) ซึ่งก็คือตัวแปรที่สามารถเก็บข้อมูลไว้ได้หลายค่าภายในตัวแปรตัวเดียว ในบทนี้จะพูดถึงเฉพาะตัวแปรพื้นฐาน

ตัวแปรพื้นฐานในภาษา C ตามมาตรฐาน ANSI

ชนิดของตัวแปร	ขนาด (bits)	ขอบเขต	ความหมาย
char	8	-128 ถึง 127 (อักขระ ASCII)	เก็บข้อมูลชนิดอักขระ โดยใช้พื้นที่หน่วยความจำในการจัดเก็บ 8 bits (1 byte)
unsigned char	8	0 ถึง 255	เก็บข้อมูลชนิดอักขระแบบไม่คิดเครื่องหมาย
int	16	-32,768 ถึง 32,767	เก็บข้อมูลชนิดตัวเลขจำนวนเต็ม ใช้พื้นที่ในหน่วยความจำ 16 bits (2 bytes)
unsigned int	16	0 ถึง 65,535	เก็บข้อมูลชนิดตัวเลขจำนวนเต็ม แบบไม่คิดเครื่องหมาย
short	8	-128 ถึง 127	เก็บข้อมูลชนิดตัวเลขจำนวนเต็มแบบสั้น ใช้พื้นที่หน่วยความจำ 8 bits (1 byte)
unsigned short	8	0 ถึง 255	เก็บข้อมูลชนิดตัวเลขจำนวนเต็มแบบสั้น โดยไม่คิดเครื่องหมาย
long	32	-2,147,483,648 ถึง 2,147,483,649	เก็บข้อมูลชนิดตัวเลขจำนวนเต็มแบบยาว ใช้พื้นที่หน่วยความจำในการจัดเก็บ 32 bits (4 bytes)
unsigned long	32	0 ถึง 4,294,967,296	เก็บข้อมูลชนิดตัวเลขจำนวนเต็มแบบยาว และไม่คิดเครื่องหมาย
float	32	$3.4*10^{-38}$ ถึง $3.4*10^{38}$	เก็บข้อมูลชนิดตัวเลขทศนิยม ใช้พื้นที่ในหน่วยความจำ 32 bits (4

			bytes) โดยเก็บค่าทศนิยมประมาณ 6 ตัว
double	64	$3.4*10^{-308}$ ถึง $3.4*10^{308}$	เก็บข้อมูลชนิดตัวเลขทศนิยม ใช้พื้นที่ในหน่วยความจำ 64 bits (8 bytes) เก็บค่าทศนิยมประมาณ 12 ตัว
long double	128	$3.4*10^{-4032}$ ถึง $1.1*104032^{4032}$	เก็บข้อมูลชนิดตัวเลขทศนิยม ใช้พื้นที่ในหน่วยความจำ 128 bits (16 bytes) เก็บค่าทศนิยมประมาณ 24 ตัว

ข-3.1 เครื่องหมายและการดำเนินการในภาษาซีเมื่อเตรียมข้อมูลพร้อมทั้งสร้างตัวแปรสำหรับเก็บข้อมูลขึ้นมาแล้ว ต่อไปเราจะนำข้อมูลเหล่านั้นมาดำเนินการเพื่อเขียนโปรแกรมให้ทำงานออกมาอย่างที่ต้องการ โดยการดำเนินงานดังกล่าวอาจจะหมายถึง การนำข้อมูลมาคำนวณทางคณิตศาสตร์ การคำนวณทางตรรกศาสตร์ หรือทำการเปรียบเทียบเพื่อให้ได้ผลลัพธ์ออกมา ซึ่งทั้งหมดเป็นสิ่งที่เราจะเรียนรู้กันในบทนี้

ข-3.2 เครื่องหมายการคำนวณทางคณิตศาสตร์การดำเนินการพื้นฐานที่สุดทั้งในชีวิตประจำวันและในการเขียนโปรแกรมก็คือ การคำนวณทางคณิตศาสตร์ ซึ่งถือได้ว่าเป็นการดำเนินการที่กระทำบ่อยครั้ง โดยเครื่องหมายที่ใช้ในการคำนวณทางคณิตศาสตร์ในภาษาซี แสดงได้ดังนี้

เครื่องหมาย	**การดำเนินการ**	**ตัวอย่างการใช้งาน**	**ความหมาย**
+	บวก	z = x + y	บวกค่าในตัวแปร x เข้ากับค่าในตัวแปร y ผลลัพธ์เก็บไว้ที่ตัวแปร z
–	ลบ	z = x – y	ลบค่าในตัวแปร x ด้วยค่าในตัวแปร y ผลลัพธ์เก็บไว้ที่ตัวแปร z
*	คูณ	z = x * y	คูณค่าในตัวแปร x กับค่าในตัวแปร y ผลลัพธ์เก็บไว้ที่ตัวแปร z

/	หาร	z = x / y	หารค่าในตัวแปร x ด้วยค่าในตัวแปร y ผลลัพธ์เก็บไว้ที่ตัวแปร z
%	หารเอาเศษ (Modulo)	z = x % y	หารค่าในตัวแปร x ด้วยค่าในตัวแปร y ผลลัพธ์คือเศษที่ได้จากการหาร โดยเก็บไว้ที่ตัวแปร z

เครื่องหมายการคำนวณทางคณิตศาสตร์ประเภทของการเพิ่มหรือลดค่าทีละหนึ่ง แสดงได้ดังนี้

เครื่องหมาย	การดำเนินการ	ตัวอย่างการใช้งาน	ความหมาย
++	เพิ่มค่าทีละหนึ่ง (Increment)	y = ++x	บวกค่าในตัวแปร x เพิ่มขึ้น 1 ก่อนที่จะกำหนดค่า x ให้กับตัวแปร y
		y = x++	กำหนดค่าในตัวแปร x ให้กับตัวแปร y ก่อนที่จะบวกค่า x เพิ่มขึ้น 1
– –	ลดค่าทีละหนึ่ง (Decrement)	y = – –x	ลบค่าในตัวแปร x ลง 1 ก่อนที่จะกำหนดค่า x ให้กับตัวแปร y
		y = x– –	กำหนดค่าในตัวแปร x ให้กับตัวแปร y ก่อนที่จะลดค่า x ลง 1

นอกจากนี้การคำนวณทางคณิตศาสตร์ในภาษา C ยังมีเครื่องหมายประเภทที่เรียกว่าลดรูป ดังแสดงต่อไปนี้

เครื่องหมาย	ตัวอย่างการใช้งาน	ความหมาย
+=	y + = x	บวกค่าในตัวแปร y ด้วยค่าในตัวแปร x ผลลัพธ์ที่ได้กำหนดกลับไปให้ y
–=	y – = x	ลบค่าในตัวแปร y ด้วยค่าในตัวแปร x ผลลัพธ์ที่ได้กำหนดกลับไปให้ y
*=	y * = x	คูณค่าในตัวแปร y ด้วยค่าในตัวแปร x ผลลัพธ์ที่ได้กำหนดกลับไปให้ y
/=	y / = x	หารค่าในตัวแปร y ด้วยค่าในตัวแปร x ผลลัพธ์ที่ได้กำหนดกลับไปให้ y

%=	Y% = x	หารค่าในตัวแปร y ด้วยค่าในตัวแปร x เศษจากการหารเป็นผลลัพธ์กำหนดกลับไปให้ y

ข-3.3 เครื่องหมายการเปรียบเทียบส่วนใหญ่แล้วการดำเนินการเปรียบเทียบจะทำงานร่วมกับการดำเนินการอื่น ๆ เช่น เปรียบเทียบผลจากการคำนวณทางคณิตศาสตร์หรือตรรกศาสตร์ ซึ่งทำให้ในหลายครั้งเราเห็นภาพของการดำเนินการเปรียบเทียบได้ไม่ชัดเจน ทั้งที่การเปรียบเทียบเป็นการดำเนินการที่มีความสำคัญไม่น้อยไปกว่าการดำเนินการรูปแบบอื่น สำหรับเครื่องหมายที่ใช้ในการเปรียบเทียบแสดงได้ดังนี้

เครื่องหมาย	**การเปรียบเทียบ**	**ตัวอย่างการใช้งาน**	**ความหมาย**
==	เท่ากับ	x == y	ผลลัพธ์จะเป็นจริง ถ้าค่าในตัวแปร x เท่ากับค่าในตัวแปร y
|=	ไม่เท่ากับ	x | =y	ผลลัพธ์จะเป็นจริง ถ้าค่าในตัวแปร x ไม่เท่ากับค่าในตัวแปร y
<	น้อยกว่า	x < y	ผลลัพธ์จะเป็นจริง ถ้าค่าในตัวแปร x น้อยกว่าค่าในตัวแปร y
<=	น้อยกว่าหรือเท่ากับ	x < = y	ผลลัพธ์จะเป็นจริง ถ้าค่าในตัวแปร x น้อยกว่าหรือเท่ากับค่าในตัวแปร y
>	มากกว่า	x > y	ผลลัพธ์จะเป็นจริง ถ้าค่าในตัวแปร x มากกว่าค่าในตัวแปร y
>=	มากกว่าหรือเท่ากับ	x >= y	ผลลัพธ์จะเป็นจริง ถ้าค่าในตัวแปร x มากกว่าหรือเท่ากับค่าในตัวแปร y

ข-3.4 การควบคุมทิศทางการทำงานของโปรแกรมในทางปฏิบัตินั้นสภาพของปัญหาที่เราต้องเขียนโปรแกรมขึ้นมาเพื่อแก้ไขนั้นมีความซับซ้อน ซึ่งคงจะไม่ใช่โปรแกรมที่ทำงานเรียงกันไปตั้งแต่ต้นจนจบโปรแกรม แต่ควรจะเป็นโปรแกรมทีควบคุมทิศทางการทำงานได้ อย่างเช่น ถ้าข้อมูลที่รับเข้ามาเป็นเลขคู่ให้ทำงานอย่างหนึ่ง แต่ถ้าเป็นเลขคี่ให้ทำงานอีกอย่างหนึ่ง หรือกำหนดให้ทำงานซ้ำคำสั่งเดิม ซึ่งจะทำให้โปรแกรมของเราทำงานได้อย่างมีประสิทธิภาพมากขึ้น ในภาษาซี แบ่งลักษณะการควบคุมทิศทางของโปรแกรมออกเป็น 2 ประเภทหลักคือ การควบคุม

ทิศทางแบบเลือกทำและแบบวนรอบ ซึ่งแต่ละประเภทก็จะมีคำสั่งที่ภาษาซี กำหนดไว้ให้เพื่อนำไปใช้งาน

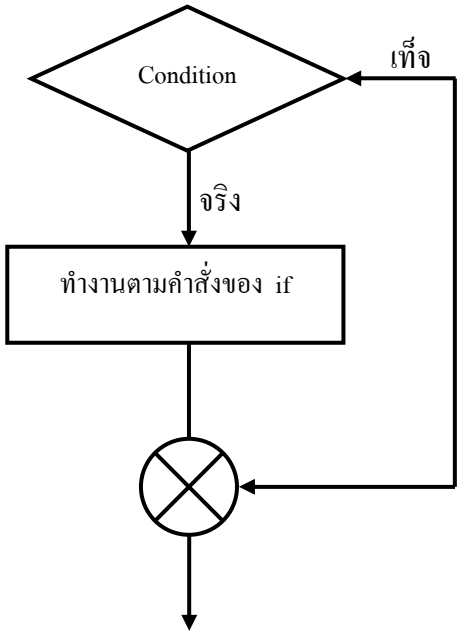
คำสั่ง if

การเลือกทำโดยใช้คำสั่ง if จะใช้ในกรณีที่มีทางเลือกให้ทำงานอยู่เพียงทางเลือกเดียว ผลจากการตรวจสอบเงื่อนไขคือ ทำกับไม่ทำตามคำสั่งนั้น รูปแบบการเรียกใช้งานคำสั่ง if แสดงได้ดังนี้

```
if (condition) statement ;
```

condition : เงื่อนไขที่กำหนดขึ้นเพื่อใช้พิจารณาว่าจะทำหรือไม่ทำตามคำสั่ง โดยจะต้องเขียนไว้ภายในเครื่องหมาย () ซึ่งเงื่อนไขอาจจะอยู่ในรูปของนิพจน์การคำนวณและเปรียบเทียบหรือเป็นค่าของตัวแปรก็ได้

statement : คำสั่งที่จะให้ทำงานถ้าผลการตรวจสอบเงื่อนไขเป็นจริง โดยอาจจะมีมากกว่าหนึ่งคำสั่งได้ แต่ต้องใช้เครื่องหมายปีกกา { } ครอบคำสั่งเหล่านั้นเอาไว้ด้วย ดังนี้ แผนภาพแสดงการทำงานของคำสั่ง if แสดงได้ดังรูปต่อไปนี้

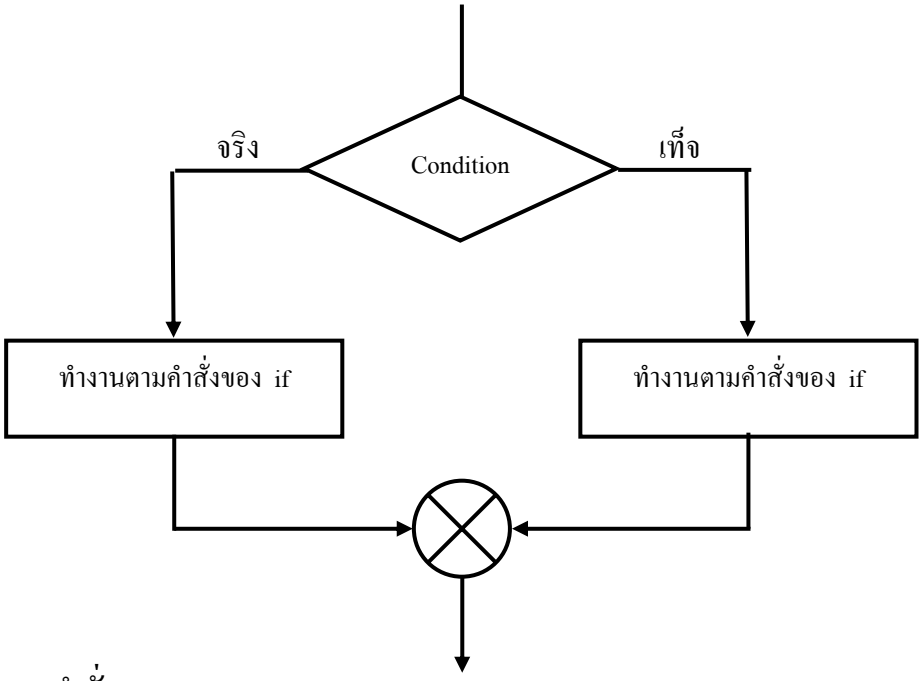


คำสั่ง if-else

คำสั่ง if-else จะใช้ในกรณีที่มีทางเลือกให้ทำงาน 2 ทางเลือก โดยรูปแบบของการเรียกใช้งานคำสั่ง if-else แสดงได้ดังนี้

การทำงานของคำสั่ง if-else จะเริ่มจากการตรวจสอบเงื่อนไข ถ้าผลออกมาเป็นจริง ตัวแปลภาษา C จะทำงานตามคำสั่งของ if แต่ถ้าผลออกมาเป็นเท็จ คำสั่งของ else จะถูกทำงาน

แผนภาพแสดงการทำงานของ คำสั่ง if-else แสดงได้ดังนี้

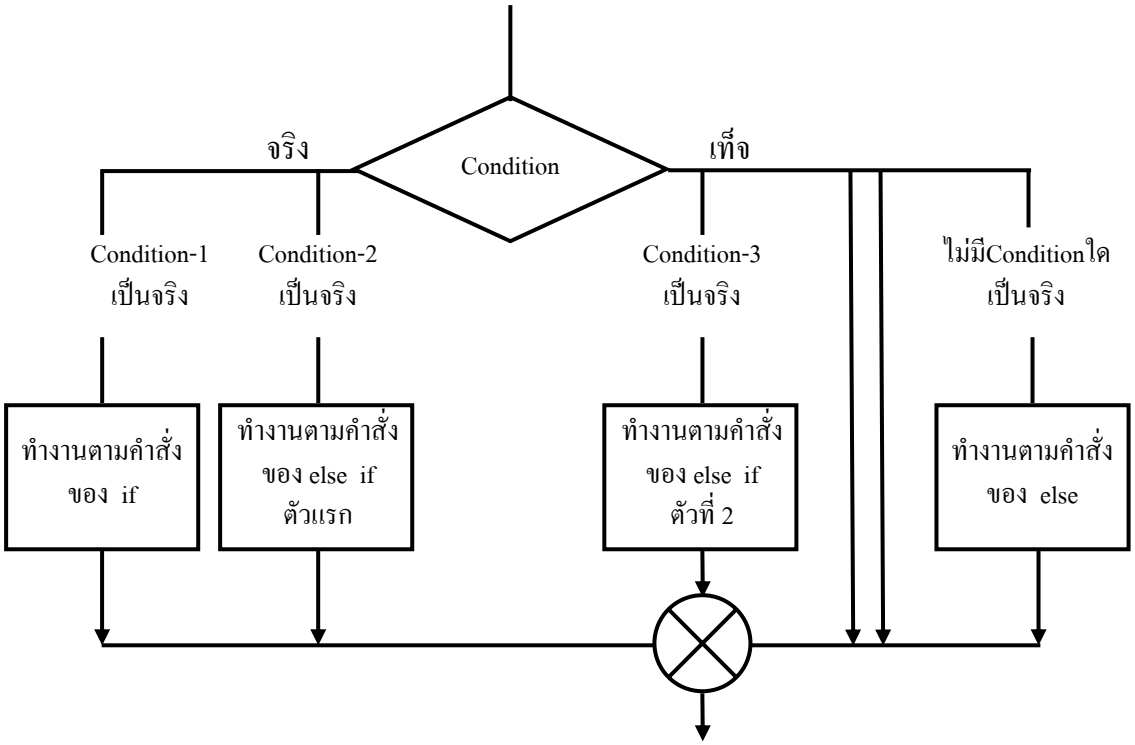


คำสั่ง if-else if

คำสั่ง if-else if จะใช้ในกรณีที่มีทางเลือกให้ทำงานมากกว่า 2 ทางเลือก โดยแต่ละทางเลือกมีเงื่อนไขต่างกัน ดังนั้นเราจึงต้องใช้เรียกคำสั่ง if หลายครั้งเพื่อกำหนดเงื่อนไขสำหรับแต่ละทางเลือก รูปแบบการเรียกใช้งานคำสั่ง if-else if แสดงได้ดังนี้

การทำงานจะเริ่มตั้งแต่ตัวแปลภาษา C ทำการตรวจสอบเงื่อนไขแรก ถ้าผลออกมาเป็นจริงก็จะทำงานตามคำสั่งของ if แต่ถ้าผลออกมาไม่จริง ตัวแปลภาษา C จะทำการตรวจสอบเงื่อนไขที่ 2 ซึ่งถ้าผลเป็นจริงก็จะทำงานตามคำสั่งของ else if นั้น ถ้าไม่จริง ตัวแปลภาษา C จะทำการตรวจสอบเงื่อนไขอื่นเรียงตามลำดับต่อไปจนเมื่อครบทุกเงื่อนไขแล้วถ้าผลยังคงไม่จริง ตัวแปลภาษา C จะทำงานตามคำสั่งที่กำหนดไว้ที่ else

แผนภาพแสดงการทำงานของคำสั่ง if-else if แสดงได้ดังนี้

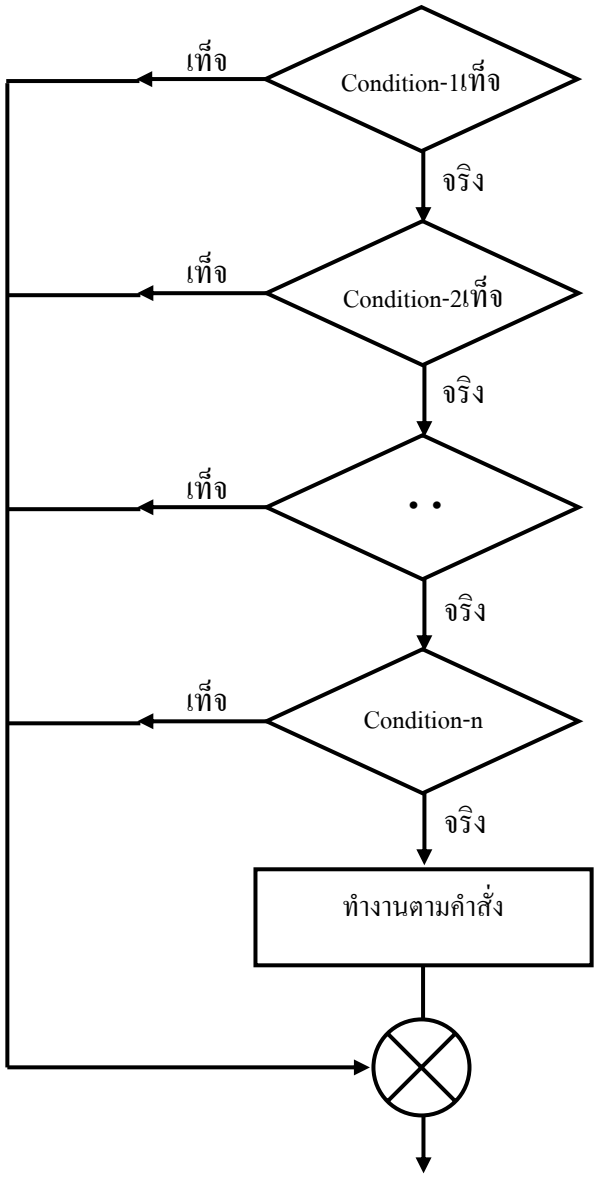


คำสั่ง if ซ้อน if

คำสั่ง if ซ้อน if จะใช้กับปัญหาที่มีเงื่อนไขซับซ้อน อย่างเช่น กำหนดว่าต้องเป็นจำนวนคู่และมีค่าไม่เกิน 50 หรือเงื่อนไขต้องเป็นเพศชายอายุไม่เกิน 20 ปี เป็นต้น ซึ่งคำสั่ง if ซ้อน if ก็คือคำสั่ง if ตามปกติ แต่เรานำมาเขียนซ้อนกันมากกว่าหนึ่งชั้น

ในการทำงานตัวแปลภาษา C จะเริ่มจากการตรวจสอบเงื่อนไขแรก ถ้าเป็นจริงก็จะทำงานต่อโดยการตรวจสอบเงื่อนไขที่ 2 และเงื่อนไขอื่น ๆ ต่อไปจนครบ โดยถ้าทุกเงื่อนไขเป็นจริงจึงจะทำงานตามคำสั่ง แต่ถ้าเงื่อนไขใดก็ตามไม่เป็นจริง ตัวแปลภาษา C จะออกจากการทำงานของคำสั่ง if ซ้อน if ทันที

แผนภาพแสดงการทำงานของคำสั่ง if ซ้อน if แสดงได้ดังนี้



การควบคุมทิศทางแบบวนรอบ หรือที่เรียกกันว่าการทำงานแบบวนลูป (loop) ก็คือ การที่เราเขียนโปรแกรมให้วนรอบทำงานซ้ำคำสั่งเดิม โดยมีการกำหนดเงื่อนไขเพื่อให้โปรแกรมวนรอบทำงาน คำสั่งในภาษา C ที่ใช้สำหรับควบคุมทิศทางแบบวนรอบ ได้แก่ while, do-while และ for ซึ่งแสดงรายละเอียดได้ดังนี้

คำสั่ง while

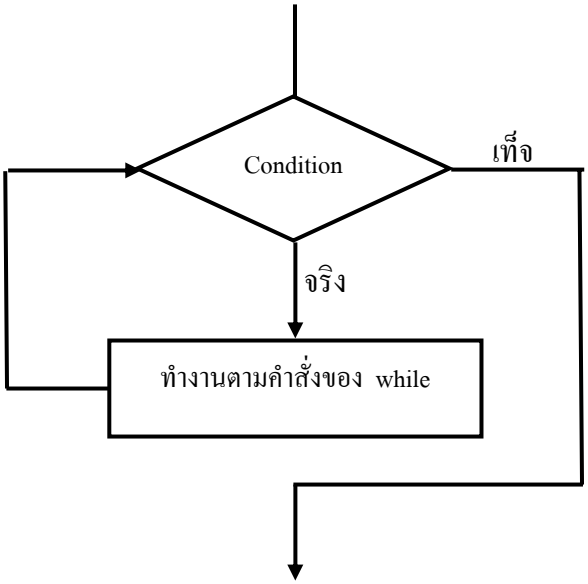
ก่อนที่จะอธิบายการทำงาน เพื่อความเข้าใจให้ดูรูปแบบการเรียกใช้งานคำสั่ง while ดังนี้

```
while(condition)
        statement;
```

```
while(condition)
(
        statement-1;
        statement-2;
        …
        statement-n;
)
```

วนรอบแบบ while จะเริ่มต้นทำงานจากการตรวจสอบเงื่อนไข (condition) ถ้าเป็นจริง จะทำงานตามคำสั่งของ whil เมื่อเสร็จแล้วก็จะวนกลับไปตรวจสอบเงื่อนไขใหม่ เป็นเช่นนี้ไปเรื่อย ๆ จนกว่าเงื่อนไขจะไม่เป็นจริงจึงจะหลุดออกจากการทำงาน

แผนภาพแสดงการทำงานของวนรอบแบบ while แสดงได้ดังนี้



คำสั่ง for

คำสั่ง for ใช้สำหรับการควบคุมทิศทางของโปรแกรมให้ทำงานแบบวนรอบเช่นเดียวกับ while และ do-while แต่คำสั่ง for มีรูปแบบการเรียกใช้งานที่ต่างไปจากคำสั่งอื่น ดังนี้

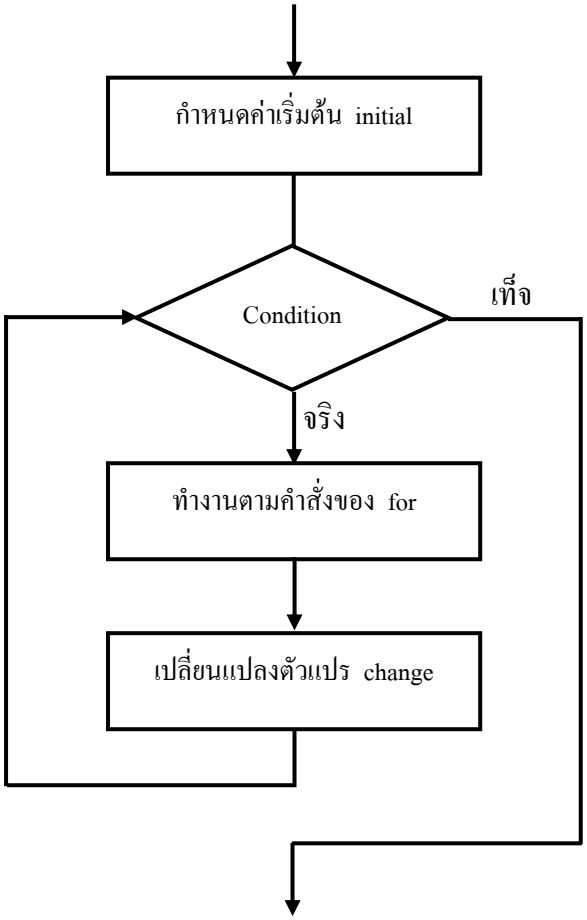
initial : เป็นส่วนที่ใช้กำหนดค่าเริ่มต้นให้กับตัวแปร ซึ่งตัวแปรนี้จะนำมาใช้กำหนดเป็นเงื่อนไขด้วย

condition : เงื่อนไขที่กำหนดขึ้นเพื่อให้โปรแกรมวนรอบทำงาน

change : ส่วนที่ทำการเปลี่ยนแปลงค่าของตัวแปร ซึ่งอาจจะเป็นการเพิ่มหรือลดค่าของตัวแปรทีละหนึ่ง (Increment/Decrement) หรือมากกว่านั้น เพื่อที่จะนำค่าของตัวแปรนั้นไปตรวจสอบเงื่อนไขในรอบถัดไป

statement : คำสั่งที่จะให้วนรอบทำงาน ในกรณีที่ผลการตรวจสอบเงื่อนไขออกมาเป็นจริง

แผนภาพการทำงานของคำสั่ง for แสดงได้ดังนี้



ภาคผนวก ค

โปรแกรม Matlab

โปรแกรม MATLAB

ในการศึกษาเกี่ยวกับ Adaptive Filter เราใช้โปรแกรม Matlab เข้ามาช่วยในการแสดงผล โดยจะอธิบายในส่วนของคำสั่งและการแสดงผล ดังนี้

ค-1 เมตริกซ์และเวคเตอร์

การป้อนค่าเมตริกซ์ใน Matlab สามารถกระทำได้ง่ายมากเช่น ถ้าต้องการสร้างตัวแปร a ให้เก็บเมตริกซ์ $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ สามารถสั่งได้ ดังนี้

```
>>a = [1 2 3;4 5 6; 7 8 9];
```

หรือ a = [1 2 3;4 5 6; 7 8 9] ก็ได้

สรุปว่า เครื่องหมายคอมม่า (,) หรือ วรรค ใช้แบ่งระหว่างคอลัมน์ และ เครื่องหมายเซมิโคลอน (;) ใช้แบ่งระหว่างแถว (สังเกตว่าการเติมวรรคมากน้อยไม่มีผลต่อ Matlab)

ส่วนเวคเตอร์ก็คือ เมตริกซ์ที่มีแถวเดียว (เรียกว่า เวคเตอร์แถว) หรือเมตริกซ์ที่มีคอลัมน์เดียว (เรียกว่า เวคเตอร์คอลัมน์) ซึ่งสามารถป้อนค่าได้โดยวิธีเดียวกันเช่น

```
>>a = [1 2 3 4 5 6 7];          เวคเตอร์แถว
>>a = [1; 2; 3;4; 5; 6; 7];     เวคเตอร์คอลัมน์
```

สำหรับการสร้างเวคเตอร์แถวที่มีค่าเป็นเชิงเส้น สามารถทำได้อย่างสะดวกอีกวิธีหนึ่งโดยใช้เครื่องหมายโคลอน (:) เช่น

```
>>a = 1:9        (หรือ a = [1:9])      จะให้ค่า a = [1 2 3 4 5 6 7 8 9]
>>a = 1:0.5 : 5                        จะให้ค่า a = [1  1.5  2  2.5  3  3.5  4  4.5  5]
>>a = 3 : -2 : -10                     จะให้ค่า a = [3  1  -1  -3  -5  -7  -9]
```

สรุปว่า รูปแบบการใช้คือ [ค่าแรก : ค่าที่เพิ่มขึ้น : ค่าสุดท้าย] โดยสามารถละ [] ได้ และถ้าละค่ากลางก็จะถือว่า ค่าที่เพิ่มขึ้นเป็น 1

ค-2 การกระทำทางเมตริกซ์ (Matrix operations)

การบวกลบคูณหาร และยกกำลังของเมตริกซ์ สามารถกระทำได้โดยใช้สัญลักษณ์เหมือนการกระทำกับค่าสเกลาปกติ ข้อควรระวัง คือ ขนาดของเมตริกซ์ที่มากระทำกันต้องถูกต้องตามกฎของการกระทำนั้น ๆ เช่น ถ้าเอา a*b โดย a และ b เป็นเมตริกซ์ จะได้ว่าจำนวนแถวของ a จะต้องเท่ากับจำนวนคอลัมน์ของ b เสมอ และถ้าเอา a^2 จะต้องใช้ a ที่เป็นเมตริกซ์จัตุรัส (square) เสมอ เป็นต้น

การกระทำที่จะขอแนะนำเพิ่มเติมในที่นี้คือ transpose ซึ่งทำได้โดยใช้สัญลักษณ์เครื่องหมายคำพูดขีดเดียว (Ə) ดังนี้

>>b=aƏ ให้เมตริกซ์ b เป็น transpose ของเมตริกซ์ a

หมายเหตุ ถ้าสมาชิกใน a เป็นจำนวนเชิงซ้อน การกระทำนี้จะเป็น conjugate transpose คือ นอกจาก transpose แล้ว ยังแปลงสมาชิกทุกตัวใน a เป็นค่าคอนจูเกตด้วย (เปลี่ยนเครื่องหมายของค่าจินตภาพ) ถ้าต้องการ transpose ธรรมดาโดยไม่มีคอนจูเกตให้ใช้เครื่องหมาย .Ə แทนดังนี้

>>b=a.Ə

การกระทำที่เข้าถึงสมาชิกทุกตัวในเมตริกซ์ (Array operations)

สมมติมีเวคเตอร์ (หรือเมตริกซ์) ที่ขนาดเท่ากัน ดังนี้

```
>> a = [1 2 3];
>> b = [4 5 6];
```

และต้องการหาเวคเตอร์ ที่มีสมาชิกเป็นผลคูณของสมาชิกแต่ละตัวที่ตำแหน่งตรงกันของ a กับ b ทำได้โดยใช้เครื่องหมาย .* ดังนี้

```
>> a.*bƏ
ans = [4  10  18]
```

การกระทำกับสมาชิกต่าง ๆ ได้นี้มีประโยชน์มากในการประมวลผลสัญญาณ ซึ่งนอกจาก .* แล้ว ยังมีการกระทำในทำนองนี้อีก คือ

```
>> a./b        เอาสมาชิกแต่ละตัวของ a กับ b มาหารกัน
>> a.^b        เอาสมาชิกแต่ละตัวของ a ยกกำลังด้วยสมาชิกแต่ละตัวของ b
>> a.^3        เอาสมาชิกแต่ละตัวของ a ยกกำลังด้วย 3
>> a+3         เอาสมาชิกแต่ละตัวของ a บวกด้วย 3 (ลบก็ทำได้เช่นเดียวกัน)
```

สังเกตว่า ไม่มีการกระทำ .+ และ .- เพราะ การบวกลบเมตริกซ์เป็นการกระทำกับสมาชิกแต่ละตัวอยู่แล้ว

สำหรับฟังก์ชั่นของค่าสเกลาต่าง ๆ เมื่อนำมากระทำกับเมตริกซ์ ก็จะมีผลเป็นการกระทำกับสมาชิกแต่ละตัวในเมตริกซ์ และให้ผลลัพธ์เป็นเมตริกซ์ขนาดเท่าเดิม เช่น

>> sin(n) หาค่า sin ของสมาชิกแต่ละตัวของ a

การอ้างถึงสมาชิกภายในเมตริกซ์ (หรือเวคเตอร์)

ถ้ามีเมตริกซ์ a = $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

เราสามารถอ้างถึงสมาชิกแต่ละตัวใน a ได้โดยใช้รูปแบบ a (แถว), คอลัมน์) เช่น a(2,3) จะได้ค่าเป็น 6 เป็นต้น ที่พิเศษไปกว่านั้นก็คือ Matlab อนุญาตให้เราอ้างค่าในเมตริกซ์ได้หลาย ๆ ค่าพร้อม ๆ กัน เพื่อสร้างเป็นเมตริกซ์ หรือเวคเตอร์ใหม่ได้ เช่น

>> a([1 2] , [2 3]) หรือ a(1:2 , 2:3) อ้างถึงแถวที่ 1 กับ 2 และคอลัมน์ที่ 2 กับ 3

ans = $\begin{bmatrix} 2 & 3 \\ 5 & 6 \end{bmatrix}$

>> a(2 , [13]) อ้างถึงแถวที่ 2 และคอลัมน์ที่ 1 กับ 3

ans = [4 6]

>> a(2 , :) อ้างถึงแถวที่ 2 ทั้งแถว

ans = [4 5 6]

>> a(2 , :) อ้างถึงคอลัมน์ที่ 2 ทั้งคอลัมน์ (สังเกตการใช้ :)

ans = $\begin{bmatrix} 2 \\ \\ 8 \end{bmatrix}$

โดยสรุปก็คือเราสามารถใช้ “เวคเตอร์เป็นตัวชี้” ได้ แทนที่จะใช้สเกลเฉย ๆ สำหรับการอ้างค่าในเวคเตอร์ก็สามารถทำได้ในทำนองเดียวกัน เพียงแต่ใช้ตัวชี้เพียงตัวเดียว เช่น

```
>> a=[1 2 3 4 5 6 7 8 9];
>> a(3)                                  อ้างถึงค่าที่ 3
   ans = 3
>> a(2:5) หรือ a([2:5])                  อ้างถึงค่าที่ 2 ถึง 5
   ans = [2 3 4 5]
>> a(1:2:9)                              อ้างถึงค่าที่ 1, 3, 5, 7 และ 9
   ans = [1 3 5 7 9]
```

ค-2.1 ฟังก์ชั่นภายในMatlab มีฟังก์ชั่นพื้นฐานอยู่มากมาย และก็มีวิธีใช้บอกไว้ด้วยในซอฟท์แวร์ เราสามารถดูว่ามีฟังก์ชั่นชื่ออะไรอยู่บ้าง และใช้ทำอะไร โดยใช้เมาส์เลือก Help ที่เมนู หรือพิมพ์ help ก็ได้ โดยฟังก์ชั่นต่าง ๆ ได้จัดไว้เป็นหมวดหมู่อย่างดี ในกรณีที่เรารู้ชื่อฟังก์ชั่นแต่จำวิธีใช้ไม่ได้ ก็สามารถเรียกดูวิธีใช้ได้โดยพิมพ์ help แล้วตามด้วยชื่อฟังก์ชั่น เช่น

```
>> help plot                             ขอดูวิธีใช้ฟังก์ชั่น plot
```

ในที่นี้ขอสรุปฟังก์ชั่นที่สำคัญบางส่วนไว้ ดังต่อไปนี้ (ให้ดูวิธีใช้โดยละเอียดจาก help)

<u>ฟังก์ชั่นเกี่ยวกับการสร้างเมตริกซ์</u>

```
>> zeros(n,m)                           ให้เมตริกซ์ที่มีสมาชิกเป็น 0 หมด ขนาด n × m
>> ones(n,m)                            ให้เมตริกซ์ที่มีสมาชิกเป็น 1 หมด ขนาด n × m
>> eye(n)                               ให้เมตริกซ์ identity ขนาด n × n
>> rand(n,m)                            ให้เมตริกซ์มีค่าแรนดอม 0 ถึง 1 ขนาด n × m
>> randn(n,m)                           ให้เมตริกซ์มีค่าแรนดอมแบบเกาส์เซียน ขนาด n × m
```

<u>ฟังก์ชั่นของสเกลา</u>

```
sin, cos, tan, asin, acos, atan, atan2      ฟังก์ชั่นตรีโกณมิติ
sinh, cosh, tanh, asinh, acosh, atanh       ฟังก์ชั่นไฮเปอร์โบลิก
exp (เอกโปเนนเชียล), log (ลอกการิธึมฐาน e), log10 (ลอกการิธึมฐานสิบ)
abs (หาขนาด),          sign (หาเครื่องหมาย),    rem (หาเศษจากการหาร)
round (ปัดทศนิยม),     floor (ปัดทศนิยมทิ้ง),   ceil (ปัดทศนิยมขึ้น)
sqrt (หารากที่สอง)
```

ฟังก์ชั่นของเวคเตอร์

max (หาค่ามากที่สุด), min (หาค่าน้อยที่สุด), length (หาจำนวนสมาชิก)

mean (หาค่าเฉลี่ย), median (หาค่ากลาง), std (หาค่าเบี่ยงเบนมาตรฐาน)

sum (หาผลรวม), sort (เรียงลำดับค่าจากน้อยไปมาก)

roots (หาค่ารากของโพลิโนเมียล)

ฟังก์ชั่นของเมตริกซ์

Size (หาขนาด), eig (หาค่า eigen), det (หาค่า determinant)

ค-2.2 คำสั่งเกี่ยวกับการวนลูป และเปรียบเทียบMatlab มีคำสั่งสำหรับการวนลูป และเปรียบเทียบเช่นเดียวกับภาษาสูงทั่ว ๆ ไป คำสั่งเหล่านี้มีประโยชน์มากในการเขียนในโปรแกรมสคริปต์ ซึ่งได้แก่

คำสั่ง if มีรูปแบบการใช้ คือ

```
If <เงื่อนไข>
    <คำสั่ง>
elseif <เงื่อนไข>          (elseif จะมีหลายครั้งก็ได้)
    <คำสั่ง>
else
    <คำสั่ง>
end
```

<คำสั่ง> ประกอบด้วย คำสั่ง หรือ การเรียกฟังก์ชั่น หลาย ๆ คำสั่งก็ได้

<เงื่อนไข> คือ ประโยคที่เปรียบเทียบเพื่อให้ค่า 0 ถ้าเป็นเท็จ และให้ค่า 1 ถ้าเป็นจริง เช่น

```
if a == b & ok          ถ้า a เท่ากับ b และ ok เท่ากับ 1
   a = a+1;             ให้เพิ่มค่า a ขึ้นหนึ่ง
else                    ถ้าไม่เช่นนั้น
   a = a–1;             ให้ลดค่า a ลงหนึ่ง
end
```

การกระทำที่ใช้สำหรับเปรียบเทียบมีอยู่หลายตัว สรุปได้ดังนี้

==	เท่ากับ	~=	ไม่เท่ากับ

<	น้อยกว่า	>	มากกว่า
<=	น้อยกว่าหรือเท่ากับ	>=	มากกว่าหรือเท่ากับ

และการกระทำที่ใช้เชื่อมเงื่อนไข ได้แก่

&	และ (and)	|	หรือ (or)
~	ไม่ (not)		

การกระทำทั้งสองหมวดนี้จริง ๆ แล้วยังมีวิธีการใช้เหมือนกับการใช้การกระทำบวก/ลบที่ได้กล่าวมาแล้ว เพียงแต่จะให้ผลลัพธ์เป็น 0 กับ 1 เท่านั้น เช่น

การเปรียบเทียบเมตริกซ์กับค่าคงที่

```
>>a = [1  2  3  4] > 2
   a = [0  0  1  1]
```

จะเอาค่าสมาชิกทุกตัวเปรียบเทียบว่ามากกว่า 2 หรือไม่ สร้างเป็นเมตริกซ์ใหม่ซึ่งเป็นผลลัพธ์ของการเปรียบเทียบ (ถ้าการเปรียบเทียบเป็นจริงให้ค่าเท่ากับหนึ่ง ถ้าเป็นเท็จให้ค่าเท่าศูนย์)

หรือ การเปรียบเทียบเมตริกซ์ กับเมตริกซ์

```
>>a = [1  2  3  4] == [1  4  3  5]  เปรียบเทียบสมาชิกที่ตำแหน่งตรงกันว่าเท่ากันหรือไม่
   a = [1  0  1  0]
```

คำสั่ง while มีรูปแบบการใช้คือ

```
while <เงื่อนไข>
  <คำสั่ง>
  end
```

เช่น

```
i = 10; a=1;
While  i >1
   a = a*i;
   i = i-1;
end
```

โปรแกรมนี้จะหาค่าสุดท้ายของ a เป็น 10! (10 แฟคทอเรียล) ซึ่งเท่ากับ 10*9*8 1

คำสั่ง for มีรูปแบบการใช้คือ

```
for  ตัวแปรลูป = ค่าต้น: ค่าที่เพิ่ม: ค่าสุดท้าย
```

<คำสั่ง>

end

เช่น a=1;

For i = 1:10

a = a*i;

end

เช่นเดียวกัน โปรแกรมนี้จะหาค่า 10! เช่นกัน ขอสังเกตความแตกต่างจากการใช้ while โดยปกติถ้าเรารู้ค่าเริ่มต้น และค่าสิ้นสุดของการวนลูป การใช้ for จะสะดวกกว่า

<u>เทคนิค</u> ถ้าต้องการหยุดการทำงานของลูป หรือของโปรแกรมขณะที่มันกำลังทำงานอยู่ ให้กด ctrl-C

ค-3 การวาดกราฟ

กราฟกล่าวได้ว่าเป็นสิ่งที่สำคัญที่สุดในการวิเคราะห์ และแสดงผล Matlab สามารถวาดกราฟได้หลายชนิดทั้งสองมิติและสามมิติ กราฟที่แสดงในหน้าปกของหนังสือนี้ก็วาดจาก Matlab ในที่นี้จะขอแนะนำเฉพาะการวาดกราฟสองมิติเท่านั้น

สมมติว่าเรามีเวคเตอร์ t, x, y ซึ่งเกิดจากฟังก์ชั่น gensine ที่เขียนไว้ในหัวข้อเรื่องโปรแกรมฟังก์ชั่น ดังนี้

```
>> [t,s] = gensine(50,1000);
>> [t,y] = gensine(20,1000);
```

เราต้องการวาดกราฟของ x และ y ในรูปเดียวกันโดยมี t เป็นแกนนอน สามารถทำได้ดังนี้

>> plot(t,x)	วาดโดยใช้ t เป็นแกนนอน และ x เป็นแกนตั้ง
>> hold on	ค้างรูปเอาไว้ (การวาดครั้งต่อไป จะวาดซ้อนบนรูปเดิม)
>> plot(t, y, Эb-.Э)	วาด y โดยคราวนี้ใช้สีน้ำเงิน และเป็นเส้นประขีดสลับจุด
>> gird	วาดเส้นกริด
>> axis([0 0.1 -1.5 1.5])	ปรับช่วงของการแสดงผลให้แกนนอนอยู่ระหว่าง 0 ถึง 0.1 และแกนตั้งอยู่ระหว่างค่า -1.5 ถึง 1.5
>> xlabel(Эtime(sec)Э)	เขียนคำอธิบายแกนนอน
>> ylabel(Эsignal(V)Э)	เขียนคำอธิบายแกนตั้ง

>> title(ƎSamples of Sine WavesƎ) เขียนคำอธิบายบนหัวรูป

>> hold off ยกเลิกการค้างรูป (การวาดครั้งต่อไปจะลบรูปเก่าทิ้งก่อน)

ภาพที่ ก-1 แสดงผลลัพธ์ของรูปที่ได้จากคำสั่งเหล่านี้

สรุป ว่าคำสั่ง plot ต้องการตัวแปรเข้า 3 ตัว คือ

Plot (เวคเตอร์ของแกนนอน, เวคเตอร์ของแกนตั้ง, รูปแบบของสีและลายเส้น)

โดยอย่างน้อยเราต้องใส่ค่าเวคเตอร์ของแกนตั้งเสมอ เช่น ลองสั่ง plot(x) ดู จะพบว่า Matlab ใช้คำว่า 1, 2, 3, เป็นแกนนอน สำหรับรูปแบบของสีและลายเส้นให้ใส่เป็นข้อความภายในเครื่องหมาย ƎƎ โดยอักษรตัวแรกเป็นตัวกำหนดสี และตัวต่อไปกำหนดลายเส้น โดยมีความหมายคือ

b สีน้ำเงิน, r สีแดง, k สีดำ, w สีขาว, y สีเหลือง, m สีม่วง, c สีฟ้า, g สีเขียว

- เส้นปกติ, : เส้นประไข่ปลา, -- เส้นประขีด, -. เส้นประขีดสลับจุด,

. จุด, * จุดดอกจัน, + จุดกากบาท, x จุดกากบาท, o จุดวงกลม

การวาดกราฟหลาย ๆ เส้นในรูปเดียวกันนอกจากทำโดยใช้คำสั่ง hold แล้ว ยังสามารถสั่งให้ plot ทีเดียวหลาย ๆ เส้นได้เลย โดยคำสั่งต่อไปนี้ซึ่งมีผลเหมือนข้างต้น

คำสั่งอื่นที่เกี่ยวกับการวาดกราฟที่สำคัญได้แก่

whitebg เปลี่ยนสีพื้นของรูปเป็นสีขาว มีประโยชน์มากถ้าจะพิมพ์รูปออกเครื่องพิมพ์ คำสั่งนี้ครั้งเดียวตอนเปิด Matlab ซึ่งเราอาจใส่ไว้ใน \matlab\bin\startup.m เลยก็ได้

stem ใช้แทน plot สำหรับวาดรูปเป็นจุด และมีขีดแกนตั้งให้ด้วย

semilogx และ **semilogy** ใช้แทน plot สำหรับวาดรูปที่แกนนอน หรือตั้งเป็นสเกลล็อก

figure เปิดรูปใหม่ (ในหน้าต่างใหม่)

elf ลบรูปปัจจุบัน

zoom ขยายรูปที่แสดงผลอยู่ โดยหลังจากใช้คำสั่ง zoom แล้ว ให้กดปุ่มซ้ายของเม้าส์ค้างไว้แล้วลากเม้าส์เพื่อตีกรอบภายในบริเวณของรูปกราฟ เมื่อปล่อยปุ่มเม้าส์ รูปในกรอบก็จะถูกขยาย ถ้าต้องการหดรูปเหมือนเดิมให้กดปุ่มขวาของเม้าส์ที่บริเวณรูปนั้น

Subplot(n,m,i) หรือ **subplot(nmi)** แบ่งรูปย่อยในหน้าต่างเดียวกัน ให้มี n × m รูปย่อยและชี้ที่รูปที่ i ตัวอย่างเช่น subplot(231) แบ่งเป็นรูปย่อย 6 รูป และชี้ที่รูปที่ 1 ดังแสดงในรูปที่ ก.2 รูปย่อยแต่ละรูปมีการตั้งค่าต่าง ๆ แยกอิสระต่อกันเหมือนเป็นคนละรูป ซึ่งเราต้องใช้คำสั่ง subplot ระบุ หรือใช้เมาส์คลิกที่รูปย่อยหนึ่ง ๆ เพื่อย้ายไปกระทำกับรูปย่อยนั้น

DSP Toolbox หรือ Signal Processing Toolbox เป็นชุดฟังก์ชั่นพิเศษซึ่งเพิ่มเติมเข้ามาใน Matlab (ซึ่งผู้ใช้ต้องซื้อเพิ่มเติมเอง) ฟังก์ชั่นเหล่านี้เกี่ยวข้องกับการประมวลผลสัญญาณทั้งในการวิเคราะห์ และออกแบบ ในที่นี้จะขอสรุปฟังก์ชั่นที่สำคัญซึ่งสำหรับใช้ในการประมวลผลสัญญาณขั้นพื้นฐานเท่านั้น โดยอ้างอิงกับ DSP Toolbox เวอร์ชั่น 3.0

ฟังก์ชั่นสำหรับคำนวณ และวิเคราะหตัวกรอง

Sinc	ฟังก์ชั่นซิงค์
conv	คอนโวลูชัน
fiftfilt	คอนโวลูชันแบบเร็วโดยวิธี overlap-add
filter	ตัวกรองดิจิตอล
freqz	วาดผลตอบสนองเชิงความถี่ของระบบ
grpdelay	หาความเร็วกลุ่มของระบบ
impz	หาผลตอบสนองต่ออิมพัลส์ของระบบ
unwrap	ปรับเวคเตอร์ที่เป็นค่าเฟสที่ถูกจำกัดอยู่ใน $-\pi$ ถึง π ให้ขยายออกนอกย่านนี้ได้
zplane	วาดแผนภาพโพล/ศูนย์

ฟังก์ชั่นที่ใช้จัดรูปแบบของฟังก์ชั่นถ่ายโอน

residuez	กระจายฟังก์ชั่นเป็นเศษส่วนย่อย
sos2tf	แปลงจากรูปแบบผลคูณของเทอมอันดับสอง เป็นรูปแบบเศษส่วนรวม
sos2zp	แปลงจากรูปแบบผลคูณของเทอมอันดับสอง เป็นรูปแบบกระจายค่าโพลและศูนย์
tf2zp	แปลงจากรูปแบบเศษส่วนรวม เป็นรูปแบบผลคูณของเทอมอันดับสอง
zp2sos	แปลงจากรูปแบบกระจายค่าโพลและศูนย์ เป็นรูปแบบผลคูณของเทอมอันดับสอง
zp2tf	แปลงจากรูปแบบกระจายค่าโพลและศูนย์ เป็นรูปแบบเศษส่วนรวม

ฟังก์ชั่นสำหรับออกแบบตัวกรองแบบ IIR

butter	ออกแบบตัวกรอง Butterworth

cheby1	ออกแบบตัวกรอง Chebyshev type I
cheby2	ออกแบบตัวกรอง Chebyshev type II
ellip	ออกแบบตัวกรอง Elliptic
yulewalk	ออกแบบตัวกรอง Yule-Walker
buttord	หาอันดับของตัวกรอง Butterworth
cheb1ord	หาอันดับของตัวกรอง Chebyshev type I
cheb2 ord	หาอันดับของตัวกรอง Chebyshev type II filter
ellipord	หาอันดับของตัวกรอง Elliptic filter

ฟังก์ชั่นสำหรับออกแบบตัวกรองแบบ FIR

Fir1	ออกแบบโดยวิธีหน้าต่าง
Fir2	ออกแบบโดยวิธีสุ่มความถี่
remez	ออกแบบโดยวิธี Parks-McClellan
remezord	หาอันดับของตัวกรองโดยวิธี Parks-McClellan

ฟังก์ชั่นการแปลง

fit	การแปลง FFT
fitshift	สลับผลตอบครึ่งบนของ FFT มาเป็นครึ่งล่าง
hilbert	การแปลง Hilbert
ifft	การแปลง FFT ผกผัน
psd	หา Power Spectral Density
xcorr	หา Cross-correlation
specgram	หา Spectrogram
bilinear	การแปลง Bilinear

ฟังก์ชั่นหน้าต่าง

bartlett	หน้าต่าง Bartlett
blackman	หน้าต่าง Blackman
boxcar	หน้าต่างสี่เหลี่ยม
chebwin	หน้าต่าง Chebyshev

hamming	หน้าต่าง Hamming
hanning	หน้าต่าง Hanning
kaiser	หน้าต่าง Kaiser
triang	หน้าต่าง Triangular

ฟังก์ชั่นการแปลงอัตราการสุ่ม

decimate	decimator
interp	interpolator
resample	แปลงอัตราการสุ่มด้วยอัตรา I/D เท่า

ฟังก์ชั่นเพื่อหาตัวกรองแอนะลอกต้นแบบ

besselap	ตัวกรองต้นแบบ Bessel
buttap	ตัวกรองต้นแบบ Butterworth
cheb1ap	ตัวกรองต้นแบบ Chebyshev type I
cheb2ap	ตัวกรองต้นแบบ Chebyshev type II
ellipap	ตัวกรองต้นแบบ Elliptic
Ip2bp	การแปลง LPF เป็น BPF
Ip2bs	การแปลง LPF เป็น BSF
Ip2ph	การแปลง LPF เป็น HPF
Ip2lp	การแปลง LPF เป็น LPF

ฟังก์ชั่นพิเศษเพื่อแสดงการใช้งาน

filtdemo	แสดงการออกแบบตัวกรองดิจิตอล
sosdemo	แสดงการลักษณะของฟังก์ชั่นถ่ายโอนย่อยอันดับสอง

ภาคผนวก ง

โปรแกรมที่ใช้ทดลอง

โปรแกรมที่ใช้ทดลอง

ง-1 การออกแบบที่ 1 การจำลองโดยโปรแกรม Matlab

```
clc
close all;
clear  all;

N = 0;
%Create the signal
%Original  signal  -this would  be the ECG  signal  as seen at the
%electrodes.
%Hear ,it will be represented as a simuusoid


t = linspace(1,35,100);
%t = 1:1/300:15;
%E0103_1;

Heartbeat = sin(2*pi*.1*t); %+(1/3)*sin(2*pi*0.3*t)+(1/5)*sin(2*pi*0.5*t);

%Noise signal  -a 50Hz  sinusoid


Noise = 0.8*sin(2*pi*100*t);


%Shifted  and  scaled  noise signal  this is done to show the convergence
%of the  algolithm  is not  dependent  on  magnitude  or phase  matching
%between  the reference  noise and  the  EGG signal
```

```
reference  = 0.5*sin(2*pi*100.*t);% + pi/2);

%Signal  pluse  Noise

primary  = Heartbeat + Noise;

%Perform adaptive filtering  using  a LMS  algorithm  Set the filter order
%to  2,and  the step size to 0.1



%output  = anc(primary,reference,2,0.1);
  afOrder = 2;
  mu = 0.1;
   N   =   length(primary);
   we   =   zeros(1,afOrder);
   adjustment =  zeros(1,afOrder);
   we1 = 0;

for k = 1:N
  we   =   reference(k);
  y    =   we * adjustment';
  output   =   primary(k) - y;

  output1  = output';
  output1 = output1*[1;0];
  output2(k) = output1;

  adjustment1   =   adjustment' + 2*mu*output.* we;
  adjustment    =   adjustment1';
  we(2:afOrder)  =   we(1:afOrder-1);
```

```
end

%Display  all signals

figure(1);
subplot(4,1,1);
plot(t,Heartbeat);
ylabel('Original ECG');

subplot(4,1,2);
plot(t,reference);
ylabel('Noise');

subplot(4,1,3);
plot(t,primary);
ylabel('ECG+Noise');

subplot(4,1,4);
plot(t,output2);
ylabel('Filtered ECG');
```

ง-2 การออกแบบที่ 2 การสร้างจริงโดยใช้ภาษาซี

จะยกตัวอย่างโปรแกรมแสดงผลที่ ความถี่ 50 Hz โดยค่า γ จะเปลี่ยนจาก 0.001 – 0.005 ตามลำดับ

ง-2.1 ให้ γ = 0.001 ความถี่ 50 Hz

```
#include "aiccomc.c"              /*AIC comm routines     */
#include "math.h"                 /*math library function */
#define beta    0.001    //9E-5 //0.00001 //1E-5 it OK  1.5E-8
```

```
#define N 36                /* Coefficients*/
#define samp  4984.05//4509.4          /*sample frequency      */
#define Fosc 50             /*desired frequency     */
#define pi 3.14159265                /*constant pi           */
//int AICSEC[4] = {0x162C,0x1,0x7EFE,0x63}; /*AIC Config Data*/
int AICSEC[4] = {0x162C,0x1,0x72e6,0x63}; /*AIC config data*/
//#define samp 8000 //7891.41          /*sample frequency      */
//#define Fosc 50.0             /*desired frequency     */
//#define pi 3.14159265                /*constant pi           */
//int AICSEC[4] = {0x162C,0x1,0x4892,0x67}; /*AIC Config Data*/
//int AICSEC[4] = {0x0E1C,0x1,0x3872,0x67}; /*AIC config data*/
   main()
{
 float Fs, Fsin, w, P, A, B, C,DPLUSN,Z,E,output;
float W[N+1];
float Delay[N+1];
float REFNOISE;
int n = 0, result,imp,input,refinput,refno,T,I;
 float y1=1.0,y2=1.0,y;
 AICSET();                    /*initialize AIC          */
 Fs = samp;
 Fsin = Fosc;
 P = 1/Fs;
 w = 2*pi*Fsin;
 A = 2 * cos((w * P));
 B = 1.0;
 C = sin((w * P));

for (T=0; T<N; T++)
            {
```

```
            W[T] = 0.0;
            Delay[T] = 0.0;
            }

while(1)
 {
  TWAIT;
  if(n==0) imp=1;
  else imp=0;
  y = A*y1 - B*y2 + C*imp;
  REFNOISE = y;
  y2=y1;   y1=y;
  n=1;
  Delay[0] = REFNOISE;    //New noise sample

  input=UPDATE_SAMPLE(result);
  DPLUSN = input;
  Z=0;                        //filter output set tp zero

            for (I=0; I<N; I++)
                    Z += (W[I]*Delay[I]);
                    E = DPLUSN-Z;
                    for (I = N; I>0; I--)
                    {
                        W[I] = W[I] + (beta*E*Delay[I]);
                        if (I != 0)
                        Delay[I] = Delay[I-1];
                    }
                    output =E;
                    result = (int)(output);  /* remove *1000 */
```

```
                DPLUSN=DPLUSN;
        }
}
```

ง-2.2 ให้ γ = 0.002 ความถี่ 50 Hz

```
#include "aiccomc.c"            /*AIC comm routines     */
#include "math.h"               /*math library function */
#define beta    0.002    //9E-5 //0.00001 //1E-5 it OK  1.5E-8
#define N 36              /* Coefficients*/
#define samp  4984.05//4509.4         /*sample frequency      */
#define Fosc 50           /*desired frequency     */
#define pi 3.14159265             /*constant pi           */
//int AICSEC[4] = {0x162C,0x1,0x7EFE,0x63}; /*AIC Config Data*/
int AICSEC[4] = {0x162C,0x1,0x72e6,0x63}; /*AIC config data*/
//#define samp 8000 //7891.41         /*sample frequency      */
//#define Fosc 50.0           /*desired frequency     */
//#define pi 3.14159265             /*constant pi           */
//int AICSEC[4] = {0x162C,0x1,0x4892,0x67}; /*AIC Config Data*/
//int AICSEC[4] = {0x0E1C,0x1,0x3872,0x67}; /*AIC config data*/
  main()
{
 float Fs, Fsin, w, P, A, B, C,DPLUSN,Z,E,output;
float W[N+1];
float Delay[N+1];
float REFNOISE;
int n = 0, result,imp,input,refinput,refno,T,I;
 float y1=1.0,y2=1.0,y;
 AICSET();                  /*initialize AIC         */
 Fs = samp;
 Fsin = Fosc;
 P = 1/Fs;
```

```
w = 2*pi*Fsin;
A = 2 * cos((w * P));
B = 1.0;
C = sin((w * P));

for (T=0; T<N; T++)
        {
        W[T] = 0.0;
        Delay[T] = 0.0;
        }

while(1)
 {
  TWAIT;
  if(n==0) imp=1;
  else imp=0;
  y = A*y1 - B*y2 + C*imp;
  REFNOISE = y;
  y2=y1;   y1=y;
  n=1;
  Delay[0] = REFNOISE;    //New noise sample

  input=UPDATE_SAMPLE(result);
  DPLUSN = input;
  Z=0;                    //filter output set tp zero

        for (I=0; I<N; I++)
                Z += (W[I]*Delay[I]);
                E = DPLUSN-Z;
                for (I = N; I>0; I--)
```

```
                {
                    W[I] = W[I] + (beta*E*Delay[I]);
                    if (I != 0)
                    Delay[I] = Delay[I-1];
                }
                output =E;
                result = (int)(output);  /* remove *1000 */
                DPLUSN=DPLUSN;
            }
}
```

ง-2.3 ให้ γ = 0.003 ความถี่ 50 Hz

```
#include "aiccomc.c"              /*AIC comm routines     */
#include "math.h"                 /*math library function */
#define beta    0.003    //9E-5 //0.00001 //1E-5 it OK  1.5E-8
#define N 36              /* Coefficients*/
#define samp  4984.05//4509.4          /*sample frequency      */
#define Fosc 50            /*desired frequency     */
#define pi 3.14159265               /*constant pi          */
//int AICSEC[4] = {0x162C,0x1,0x7EFE,0x63}; /*AIC Config Data*/
int AICSEC[4] = {0x162C,0x1,0x72e6,0x63}; /*AIC config data*/
//#define samp 8000 //7891.41          /*sample frequency      */
//#define Fosc 50.0            /*desired frequency     */
//#define pi 3.14159265               /*constant pi          */
//int AICSEC[4] = {0x162C,0x1,0x4892,0x67}; /*AIC Config Data*/
//int AICSEC[4] = {0x0E1C,0x1,0x3872,0x67}; /*AIC config data*/
  main()
{
 float Fs, Fsin, w, P, A, B, C,DPLUSN,Z,E,output;
float W[N+1];
```

```
float Delay[N+1];
float REFNOISE;
int n = 0, result,imp,input,refinput,refno,T,I;
 float y1=1.0,y2=1.0,y;
 AICSET();                    /*initialize AIC          */
 Fs = samp;
 Fsin = Fosc;
 P = 1/Fs;
 w = 2*pi*Fsin;
 A = 2 * cos((w * P));
 B = 1.0;
 C = sin((w * P));

for (T=0; T<N; T++)
            {
            W[T] = 0.0;
            Delay[T] = 0.0;
            }

 while(1)
  {
   TWAIT;
   if(n==0) imp=1;
   else imp=0;
   y = A*y1 - B*y2 + C*imp;
   REFNOISE = y;
   y2=y1;   y1=y;
   n=1;
   Delay[0] = REFNOISE;    //New noise sample
```

```
  input=UPDATE_SAMPLE(result);
  DPLUSN = input;
  Z=0;                    //filter output set tp zero

        for (I=0; I<N; I++)
            Z += (W[I]*Delay[I]);
            E = DPLUSN-Z;
            for (I = N; I>0; I--)
            {
                W[I] = W[I] + (beta*E*Delay[I]);
                if (I != 0)
                Delay[I] = Delay[I-1];
            }
            output =E;
            result = (int)(output);  /* remove *1000 */
            DPLUSN=DPLUSN;
        }
}
```

ง-2.4 ให้ γ = 0.004 ความถี่ 50 Hz

```
#include "aiccomc.c"            /*AIC comm routines     */
#include "math.h"               /*math library function */
#define beta    0.004    //9E-5 //0.00001 //1E-5 it OK  1.5E-8
#define N 36            /* Coefficients*/
#define samp  4984.05//4509.4       /*sample frequency      */
#define Fosc 50          /*desired frequency     */
#define pi 3.14159265            /*constant pi          */
//int AICSEC[4] = {0x162C,0x1,0x7EFE,0x63}; /*AIC Config Data*/
int AICSEC[4] = {0x162C,0x1,0x72e6,0x63}; /*AIC config data*/
//#define samp 8000 //7891.41       /*sample frequency      */
```

```
//#define Fosc 50.0             /*desired frequency     */
//#define pi 3.14159265                /*constant pi          */
//int AICSEC[4] = {0x162C,0x1,0x4892,0x67}; /*AIC Config Data*/
//int AICSEC[4] = {0x0E1C,0x1,0x3872,0x67}; /*AIC config data*/
  main()
{
 float Fs, Fsin, w, P, A, B, C,DPLUSN,Z,E,output;
float W[N+1];
float Delay[N+1];
float REFNOISE;
int n = 0, result,imp,input,refinput,refno,T,I;
 float y1=1.0,y2=1.0,y;
 AICSET();                    /*initialize AIC          */
 Fs = samp;
 Fsin = Fosc;
 P = 1/Fs;
 w = 2*pi*Fsin;
 A = 2 * cos((w * P));
 B = 1.0;
 C = sin((w * P));

for (T=0; T<N; T++)
            {
            W[T] = 0.0;
            Delay[T] = 0.0;
            }

 while(1)
  {
   TWAIT;
```

```
if(n==0) imp=1;
else imp=0;
y = A*y1 - B*y2 + C*imp;
REFNOISE = y;
y2=y1;   y1=y;
n=1;
Delay[0] = REFNOISE;    //New noise sample

input=UPDATE_SAMPLE(result);
DPLUSN = input;
Z=0;                    //filter output set tp zero

        for (I=0; I<N; I++)
                Z += (W[I]*Delay[I]);
                E = DPLUSN-Z;
                for (I = N; I>0; I--)
                {
                    W[I] = W[I] + (beta*E*Delay[I]);
                    if (I != 0)
                    Delay[I] = Delay[I-1];
                }
                output =E;
                result = (int)(output);  /* remove *1000 */
                DPLUSN=DPLUSN;
        }
}
```

ง-2.5 ให้ γ = 0.005 ความถี่ 50

```
#include "aiccomc.c"              /*AIC comm routines     */
#include "math.h"                 /*math library function */
#define beta    0.005    //9E-5 //0.00001 //1E-5 it OK  1.5E-8
#define N 36              /* Coefficients*/
#define samp  4984.05//4509.4         /*sample frequency      */
#define Fosc 50            /*desired frequency     */
#define pi 3.14159265               /*constant pi           */
//int AICSEC[4] = {0x162C,0x1,0x7EFE,0x63}; /*AIC Config Data*/
int AICSEC[4] = {0x162C,0x1,0x72e6,0x63}; /*AIC config data*/
//#define samp 8000 //7891.41         /*sample frequency      */
//#define Fosc 50.0            /*desired frequency     */
//#define pi 3.14159265               /*constant pi           */
//int AICSEC[4] = {0x162C,0x1,0x4892,0x67}; /*AIC Config Data*/
//int AICSEC[4] = {0x0E1C,0x1,0x3872,0x67}; /*AIC config data*/
main()
{
 float Fs, Fsin, w, P, A, B, C,DPLUSN,Z,E,output;
float W[N+1];
float Delay[N+1];
float REFNOISE;
int n = 0, result,imp,input,refinput,refno,T,I;
 float y1=1.0,y2=1.0,y;
 AICSET();                    /*initialize AIC         */
 Fs = samp;
 Fsin = Fosc;
 P = 1/Fs;
 w = 2*pi*Fsin;
 A = 2 * cos((w * P));
 B = 1.0;
```

```
 C = sin((w * P));
for (T=0; T<N; T++)
            {
            W[T] = 0.0;
            Delay[T] = 0.0;
            }
 while(1)
  {
   TWAIT;
   if(n==0) imp=1;
   else imp=0;
   y = A*y1 - B*y2 + C*imp;
   REFNOISE = y;
   y2=y1;   y1=y;
   n=1;
   Delay[0] = REFNOISE;    //New noise sample
     input=UPDATE_SAMPLE(result);
   DPLUSN = input;
   Z=0;                        //filter output set tp zero
              for (I=0; I<N; I++)
                        Z += (W[I]*Delay[I]);
                        E = DPLUSN-Z;
                        for (I = N; I>0; I--)
                        {
                            W[I] = W[I] + (beta*E*Delay[I]);
                            if (I != 0)
                            Delay[I] = Delay[I-1];
                        }
                        output =E;
                        result = (int)(output);  /* remove *1000 */
```

```
                DPLUSN=DPLUSN;
        }
}
```

ประวัติผู้วิจัย

ชื่อ : นายชลิต จิตต์สวัสดิ์ไทย

ชื่อวิทยานิพนธ์ : การลดทอนสัญญาณรบกวนบนคลื่นไฟฟ้าหัวใจโดยใช้ตัวกรองเชิงเลขแบบปรับตัวได้

สาขาวิชา : อุปกรณ์การแพทย์

ประวัติ

ประวัติส่วนตัว เกิดเมื่อวันที่ 29 มิถุนายน 2518 ที่จังหวัดพระนครศรีอยุธยา

ประวัติการศึกษา จบมัธยมศึกษาปีที่ 6 จากโรงเรียนอยุธยาวิทยาลัย, จบโรงเรียนจ่าอากาศเหล่าต้นหน จำพวกบังคับการบิน(ตห.บบ) นจอ.รุ่น36, จบการศึกษาระดับอนุปริญญา สาขาวิชาอิเล็กทรอนิกส์จากมหาวิทยาลัยราชภัฏนครปฐม, จบการศึกษาระดับปริญญาตรี สาขาเทคโนโลยีอุตสาหกรรม(ไฟฟ้า)จากมหาวิทยาลัยราชภัฏพระนครศรีอยุธยา, จบการศึกษาระดับปริญญาตรี สาขาเทคโนโลยีอุตสาหกรรม(อิเล็กทรอนิกส์)จากมหาวิทยาลัยราชภัฏพระนคร

ประวัติการทำงาน รับราชการอยู่ที่โรงเรียนการบิน อำเภอกำแพงแสน จังหวัดนครปฐม ตำแหน่ง เจ้าหน้าที่ควบคุมอากาศยาน เข้า-ออก (พ.ศ.2538-2542), รับราชการที่ศูนย์บริการการบินดอนเมือง ตำแหน่ง เจ้าหน้าที่ข่าวสารการบิน (พ.ศ.2542-2548)และรับราชอยู่ที่สำนักป้องกันและบรรเทาสาธารณภัย สังกัดกรุงเทพมหานคร