การประยุกต์ใช้การประมวลผลสัญญาณแบบปรับตัวเองในการลดสัญญาณรบกวน

NOISE REDUCTION BY METHOD OF ADAPTIVE SIGNAL PROCESSING

โดย

นายบุญโชค วงศ์ดีเลิศ 35104236

นายปรีชา ศรีรัตนพงษ์ 35104261

นางสาวอรการ หาญภพอมร 35104368

อาจารย์ที่ปรึกษา

อาจารย์ ชินภัทร นันทจิวากรชัย

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

ปริญญานิพนธ์ ปีการศึกษา 2538

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การประยุกต์ใช้การประมวลผลสัญญาณแบบปรับตัวเองในการลดสัญญาณรบกวน

ผู้จัดทำ

1. นาย บุญโชค	วงศ์ดีเลิศ	35104236
2. นาย ปรีชา	ศรีรัตนพงษ์	35104261
3. นางสาว อรการ	หาญภพอมร	35104368

..... อาจารย์ที่ปรึกษา

(อาจารย์ ชินภัทร นันทจิวากรชัย)

การประยุกต์ใช้การประมวลผลสัญญาณแบบปรับตัวเองในการลดสัญญาณรบกวน

บุญโชค วงศ์ดีเลิศ
ปรีชา ศรีรัตนพงษ์
อรการ หาญภพอมร
อ.ชินภัทร นันทจิวากรชัย อาจารย์ที่ปรึกษา
ปีการศึกษา 2538

บทคัดย่อ

บทความนี้เป็นการกล่าวถึงการลดเสียงรบกวนที่มีลักษณะเป็นคาบโดยใช้การ์ด TMS320C50 ทำการประมวลผลสัญญาณเชิงเลข โดยอาจกล่าวได้ว่า การ์ดนี้จะสร้างเสียงที่มีลักษณะของสัญญาณตรงข้าม โดยใช้การป้อนกลับไปหักล้างและนำผลของความผิดพลาดไปปรับตัวเองอยู่ตลอดเวลา จนผลการหักล้างนี้ ทำให้เสียงรบกวนเงียบลงได้

NOISE REEDUCTION BY METHOD OF ADAPTIVE SIGNAL PROCESSING

Boonchoke Wongdeelert
Precha Sriratanapong
Orakarn Hanpobamorn
Chinnapat Nantajiwakornchai Advisor
1995

ABSTRACT

This article describe about periodic noise reduction by using TMS 320C50 digital signal processing card.This technique can make the surrounding to be calm by generating the contradictory noise to cancel with the real noise.The generated comes from a method of adaptive signal processing.Finally the result is quiet.

สารบัญ

เนื้อเรื่อง หน้า

สารบัญรูป

สารบัญตาราง

สัญญลักษณ์

r	- อัตราส่วนของการลู่เข้าในการหาเกรย์เดียนท์
s	- สัญญาณอินพุท
v'	- ค่าถ่วงน้ำหนักบนระบบแกนหลัก
w	- ค่าถ่วงน้ำหนัก
x	- สัญญาณอินพุท
y	- สัญญาณเอาท์พุท
z	- ค่าในการเปลี่ยนรูปแบบ z
z^{-1}	- ค่าอินเวอร์สของ z (การเลื่อนเวลา 1 หน่วย)
$\mathbf{I}$	- เมตริกซ์สัญญลักษณ์
L	- ค่าถ่วงน้ำหนักของตัวกรองสุดท้าย, w_L
$\mathbf{P}$	- เวคเตอร์คอร์รีเรชั่นของสัญญาณอินพุทและสัญญาณที่ต้องการ
$\mathbf{Q}$	- เมตริกซ์เวคเตอร์ไอเกนของ R
$\mathbf{R}$	- เมตริกซ์คอร์รีเรชั่นของสัญญาณอินพุท, x
$\mathbf{T}$	- ทรานซ์โพสของเวคเตอร์และเมตริกซ์
$\mathbf{V}$	- เวคเตอร์ค่าถ่วงน้ำหนักที่แปลงมา, $\mathbf{w}-\mathbf{w}^*$
$\mathbf{v}'$	- เวคเตอร์ค่าถ่วงน้ำหนักบนระบบแกนหลัก
$\mathbf{w}$	- เวคเตอร์ค่าถ่วงน้ำหนัก
$\mathbf{x}$	- เวคเตอร์ของสัญญาณอินพุท
λ	- ค่าไอเกน
Λ	- เมตริกซ์ของไอเกน
ξ	- ค่าเฉลี่ยยกกำลังสองของค่าผิดพลาด
$*$	- ตัวแสดงให้เห็นว่าเป็นค่าสูงสุด เช่น w^*
∇	- เวคเตอร์ของเกรย์เดียนท์

บทที่1 บทนำ

เสียงรบกวนเป็นสิ่งที่คนส่วนใหญ่ไม่ต้องการ เพราะมันมีผลต่อโสตประสาทของผู้รับฟัง และอาจเป็นอันตรายได้ หากได้รับฟังเป็นเวลานาน เช่น เสียงจากเครื่องจักรกลในโรงงานที่มีเสียงค่อนข้างดัง ดังนั้นจึงควรจะมีการจัดการเสียงรบกวนเหล่านี้ให้ดี โดยในปัจจุบันมีวิธีการกำจัดเสียงรบกวนเหล่านี้อยู่หลายวิธี ซึ่งมีทั้งวิธีทางด้านแพสซีฟ (Passive),แอคทีฟ (Active) หรือวิธีควบคุมสัญญาณรบกวน (Noise Control) โดยแต่ละแบบก็จะมีวิธีปฏิบัติต่าง ๆ กัน

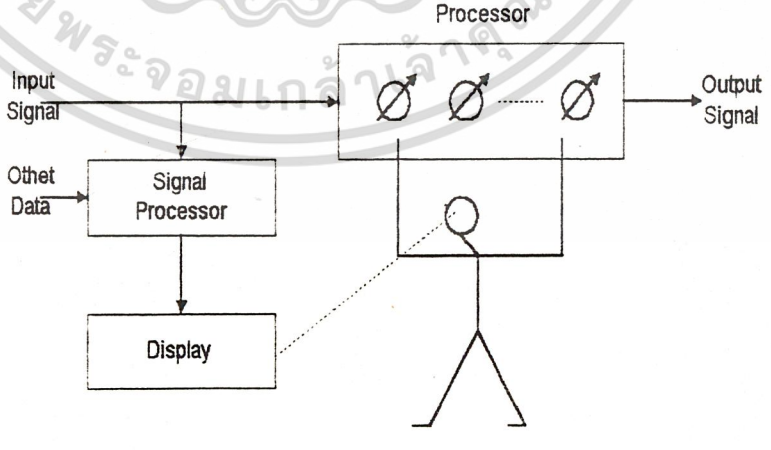
ในโครงงานนี้ จะเป็นการศึกษาการลดทอนเสียงรบกวนด้วยวิธีทางแอคทีฟ และได้นำความรู้ทางทฤษฎีการประมวลผลสัญญาณแบบปรับตัวเอง (Adaptive Signal Processing) มาประยุกต์ใช้โดยจะสร้างสัญญาณลักษณะตรงข้ามมาหักล้างกับสัญญาณเดิม เพื่อให้ได้เสียงเงียบตามที่ต้องการตามหลักการนี้สามารถที่จะนำไปประยุกต์ใช้ในการลดสัญญาณรบกวนต่างๆที่มีลักษณะเป็นคาบซ้ำเดิม เช่น เสียงรบการจากแอร์, เสียงจากเครื่องกลต่างๆ ฯลฯ ซึ่งคาดหวังว่าจากการศึกษานี้จะสามารถนำไปพัฒนาจนสามารถนำไปกำจัดเสียงรบกวนได้จริง

บทที่2 ระบบปรับตัวเอง (Adaptive System)

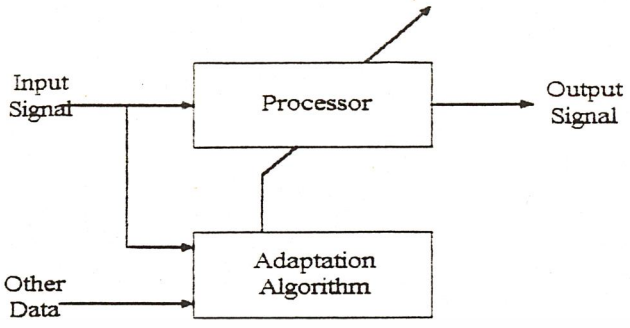
2.1 การปรับตัวแบบลูปเปิดและลูปปิด (Open and Closed-Loop Adaptation)

การปรับตัวแบบลูปเปิด (Opened-Loop) นั้นเป็นการปรับโดยการวัดค่าของอินพุท ที่เข้าระบบ จากนั้นก็จะคำนวณค่าของอินพุทนี้ด้วยสูตร หรือ การคำนวณทางอัลกอร์ลิทึ่ม ซึ่งผลที่ได้จะนำไปใช้ในการปรับตัวของระบบ ส่วนการปรับตัวแบบลูปปิด (Closed-Loop) จะเป็นในทางตรงข้าม คือ จะเป็นการศึกษาเอาท์พุทแล้วจึงนำเอาท์พุทนี้มาเป็นข้อมูลในการปรับตัว เพื่อที่จะได้เกิดการปรับตัวที่เข้าใกล้ระบบที่ต้องการให้มากที่สุด

หลักการของการปรับตัวแบบลูปเปิดและลูปปิดอธิบายได้ ดังรูป 2.1 และ 2.2 ซึ่งจะเห็นว่าในการปรับตัวของระบบนั้น จะคล้ายกับการมีผู้ควบคุมคอยปรับอยู่ ซึ่งพิจารณาดูรูป2.1 (a) จะเป็นว่าการปรับตัวแบบลูปเปิดนั้น ผู้ควบคุมจะปรับตัวตามค่าของอินพุทบนส่วนแสดงผล ที่แสดงส่วนการปรับตัวแบบลูปปิดนั้น พิจารณารูป2.2 (a) จะเห็นว่าผู้ควบคุมจะพิจารณาเอาท์พุทที่ออกมาเพื่อที่จะนำไปปรับตัว ซึ่งจะต่างจากแบบลูปเปิดจะเห็นได้ว่าทั้งการปรับตัวแบบลูปเปิดและลูปปิดนั้นผู้ควบคุมไม่ต้องลงมือจัดการสัญญาณอินพุทที่เข้ามาเอง เป็นเพียงควบคุมการปรับตัวของ ตัวประมวลผล ให้เป็นไปตามรูปแบบที่ต้องการ เราเรียกผู้ควบคุมในระบบนี้ว่า “อัลกอร์ลิทึ่มในการปรับตัว” (Adaptive Algorithms) พิจารณาได้ดังรูป 2.1(b) และ 2.2(b) ในส่วนของ “Other data” นั้น อาจจะเป็นข้อมูลแวดล้อมของระบบปรับตัวเอง หรือ ถ้าเป็นในระบบปิด อาจจะเป็นสัญญาณเอาท์พุทที่เราต้องการที่จะให้มันปรับตัวไป

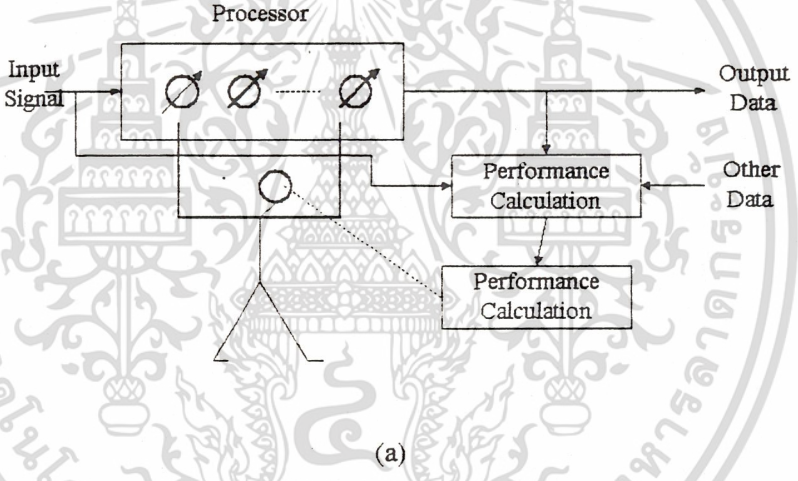


(a)

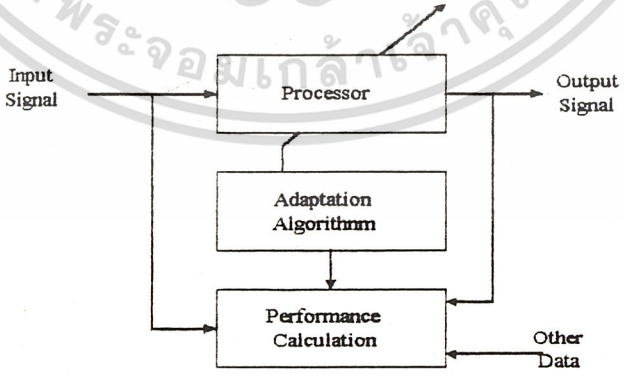


(b)

รูป 2.1 การปรับตัวแบบลูปเปิด (a) แนวความคิด (b) รูปแบบระบบ



(a)



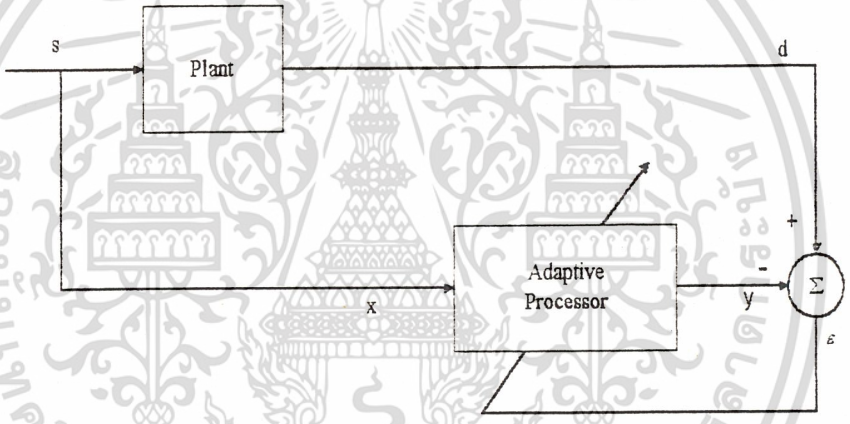
(b)

รูป 2.2 การปรับตัวแบบลูปปิด (a) แนวความ (b) รูปแบบระบบ

2.2 การประยุกต์ใช้งานระบบปรับตัวเองแบบลูปปิด (Applications of Closed-Loop Adaptation)

เราจะพิจารณาการประยุกต์ใช้งานในระบบปรับตัวเองแบบลูปปิด โดยตัวอย่างในรูป 2.3 ที่เรายกมาศึกษาเป็นตัวอย่างที่เข้าใจได้ง่าย และสามารถนำไปเป็นพื้นฐานในการศึกษาต่อไปได้ง่าย ระบบนี้เรียกว่า ระบบจำลองแบบ (System Modeling)

จากรูปสัญญาณ s เป็นอินพุทของทั้งแพล้นท์ (plant) และ ส่วนประมวลผลแบบปรับตัวเอง (Adaptive Processor) ลักษณะการทำงานจะเป็นการจำลองแบบของแพล้นท์ โดยพยายามลดค่าผิดพลาด ε (Error) แล้วนำค่าผิดพลาดนี้ไปปรับปรุงตัวเอง โดยส่วนประมวลผลแบบปรับตัวเอง จนกระทั่งค่าผิดพลาดที่ได้มีค่าเข้าใกล้ศูนย์ จะได้สัญญาณ y ที่จำลองแบบจากแพล้นท์

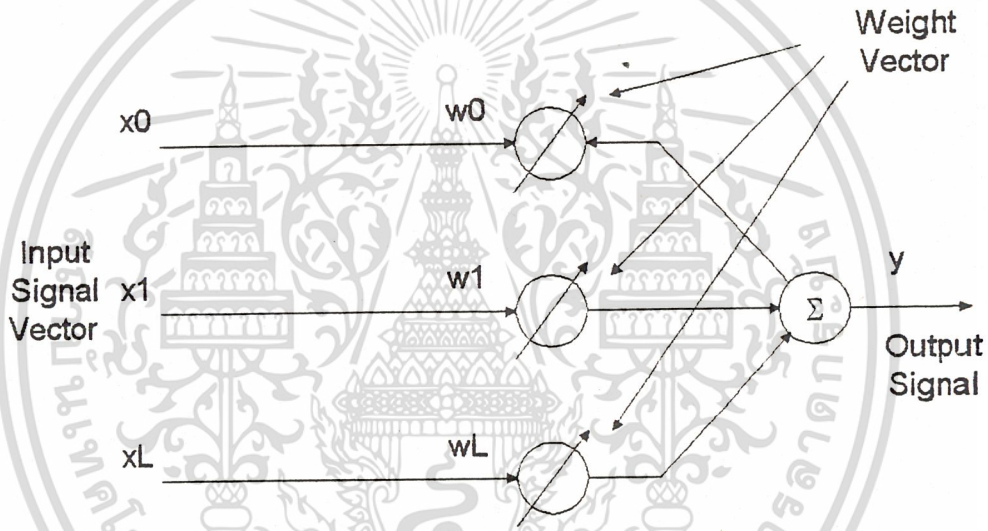


รูป 2.3 ระบบจำลองแบบ

บทที่ 3 ตัวรวมในการปรับตัวเองเชิงเส้น (The Adaptive Linear Combiner)

ตัวรวมในการปรับตัวเองเชิงเส้นนั้น เป็นพื้นฐานของวิธีการทางการประมวลผลสัญญาณแบบปรับตัวเอง (Adaptive Signal Processing) ซึ่งง่ายต่อการวิเคราะห์และการเข้าใจ

พิจารณารูป 3.1 เป็นรูปแบบทั่วไปของการรวมในการปรับตัวเองเชิงเส้น มีสัญญาณอินพุทเป็น $x_0, x_1,...,x_L$ มีเวคเตอร์ค่าถ่วงน้ำหนัก (Weight Vector) ด้วยค่า $w_0,w_1,...,w_L$ มีหน่วยรวมและสัญญาณเอาท์พุทเพียงสัญญาณเดียว ในกรณีถ้าเป็นการรวมสัญญาณดังรูป 3.1 โดยที่ค่าถ่วงน้ำหนักคงที่แล้ว สัญญาณเอาท์พุท ที่ออกมาจะเป็นเชิงเส้นกับสัญญาณอินพุทที่เข้ามา แต่ในกรณีที่ค่าถ่วงน้ำหนักมีการปรับตัวแล้ว สัญญาณเอาท์พุทก็จะไม่เป็นเชิงเส้นกับสัญญาณอินพุทอีกต่อไป



รูป 3.1 รูปแบบของการปรับตัวเองเชิงเส้น

3.1 สัญญาณอินพุทและค่าถ่วงน้ำหนัก (Input Signal and Weight Vectors)

ในการพิจารณารูปแบบทั่วไปของการรวมในการปรับตัวเองเชิงเส้น ดังรูป 3.1 จะเห็นว่าอินพุทที่เข้าจะเหมือนมี L+1 แหล่ง แต่ในทางปฏิบัตินั้น อาจจะเป็นการสุ่มตัวอย่าง สัญญาณจากแหล่งกำเนิดเพียงแหล่งเดียว แต่สุ่มตัวอย่างมา L+1 ครั้งแทน เป็น x_0 ถึง x_L

เราจะแบ่งลักษณะสัญญาณอินพุทที่เข้ามาเป็นแบบหลายอินพุท(Multiple-input) และแบบอินพุทเดียว (Single-input) ซึ่งแสดงได้เป็น

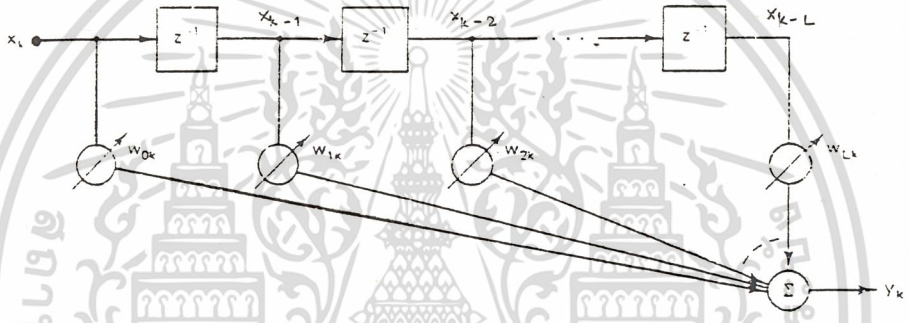
$$\text{แบบหลายอินพุท: } X_k = [\, x_{0k} \;\; x_{1k} \;\ldots\; x_{lk} \,]^T \tag{3.1}$$

$$\text{แบบอินพุทเดียว : } X_k = [\, x_k \;\; x_{k-1} \;\ldots\; x_{k-L} \,]^T \tag{3.2}$$

ในที่นี้ T คือ ทรานส์โพส (Transpone) ของเมตริกซ์ จึงทำให้ X_k เป็นเมตริกซ์คอลัมน์ และ k คือ จำนวนครั้งในการสุ่มตัวอย่าง ดังนั้นแบบหลายอินพุท จะเป็นการสุ่มตัวอย่างในครั้งที่

k เท่านั้น ส่วนแบบอินพุทเดียว จะเป็นการสุ่มตัวอย่างที่จุด k,k-1,..., ตามจำนวนการสุ่มตัวอย่างทั้งหมด

ในกรณีอินพุทเดียว เราอาจจะมีการเพิ่มเติมด้วยตัวรวมที่ปรับตัวเชิงเส้น และอุปกรณ์ดีเลย์หนึ่งหน่วย เข้าไปได้ ดังรูป 3.2 ซึ่งเรียกว่า ตัวกรองปรับตัวเองแบบทรานซ์เวอร์ซอล (Adaptive Transversal Filter) ซึ่งตัวกรองลักษณะนี้เอาไปประยุกต์ใช้ในการจำลองแบบโดยการปรับตัวเอง (Adaptive Modeling) หรือ การประมวลผลสัญญาณแบบปรับตัวเอง



รูป 3.2 รูปแบบการปรับตัวเองเชิงเส้นแบบตัวกรองทรานซ์เวอร์ซอลที่มีอินพุทเดียว

เราจะเขียนความสัมพันธ์ของอินพุทและเอาท์พุทได้โดย พิจารณารูป 3.1 และ 3.3 ดังนี้

แบบอินพุทเดียว : $$y_k = \sum_{l=0}^{L} w_{lk} x_{k-l} \tag{3.3}$$

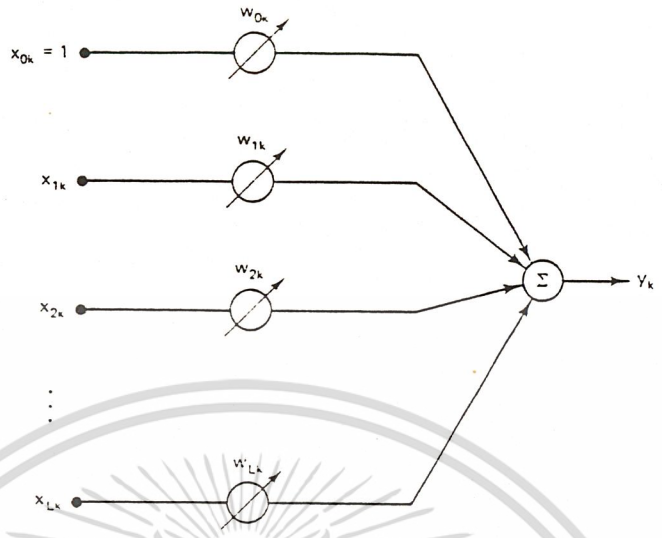
แบบหลายอินพุท : $$y_k = \sum_{l=0}^{L} w_{lk} x_{lk} \tag{3.4}$$

และเราจะได้ค่าถ่วงน้ำหนักดังนี้

$$\mathbf{W}_k = [w_{0k}\ w_{1k}\ \ldots\ w_{Lk}]^T \tag{3.5}$$

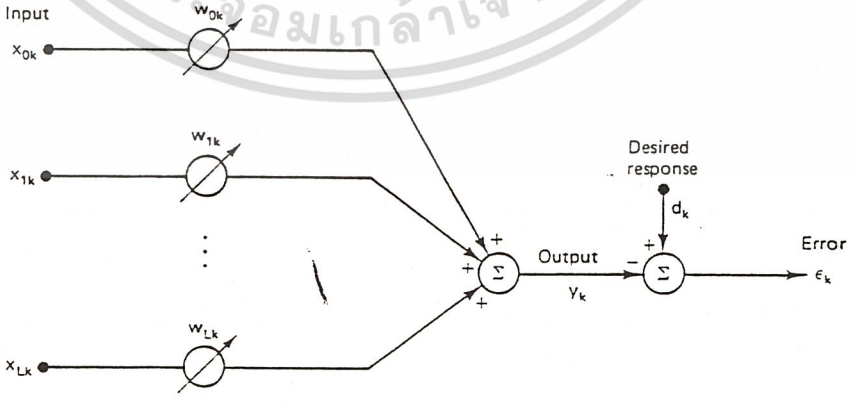
จากสมการ (3.3) และสมการ (3.4) จะได้ความสัมพันธ์

$$y_k = \mathbf{X}_k^T \mathbf{W}_k = \mathbf{W}_k^T \mathbf{X}_k \tag{3.6}$$



รูป 3.3 รูปแบบการปรับตัวเองเชิงเส้นแบบหลายอินพุท

พิจารณารูป 3.4 จะเห็นรูปแบบของการปรับตัวเอง จนได้ลักษณะของผลตอบสนองที่ต้องการ จะเห็นว่า สัญญาณเอาท์พุท y_k นั้นจะขึ้นอยู่กับสัญญาณอินพุทและค่าถ่วงน้ำหนัก เมื่อมีการรวมกันของสัญญาณ d_k และ y_k แล้วจะเกิดค่าผิดพลาด ε_k ขึ้น ถ้าสัญญาณเอาท์พุท y_k ยังไม่เท่ากับ d_k ซึ่งการปรับตัวเองของระบบจะขึ้นอยู่กับค่าความผิดพลาด ที่มีผลต่อค่าถ่วงน้ำหนัก ดังนั้นก็คือ จะเกิดการปรับค่าถ่วงน้ำหนักตามค่าผิดพลาดที่เกิดขึ้น จนกระทั่งค่าผิดพลาดที่เกิดขึ้นเป็นศูนย์



รูป 3.4 ระบบปรับตัวเองเชิงเส้นแบบหลายอินพุทกับผลตอบสนองที่ต้องการและค่าผิดพลาด

3.2 ฟังก์ชั่นเพอร์ฟอร์แมนซ์ (The Performance Function)

ค่าผิดพลาดของสัญญาณ จากรูป 3.4

$$\varepsilon_k = d_k - y_k \tag{3.7}$$

$$\varepsilon_k = d_k - \mathbf{X}_k^T\mathbf{W} = d_k - \mathbf{W}^T\mathbf{X}_k \tag{3.8}$$

ε_k ยกกำลังสอง

$$\varepsilon_k^2 = d_k^2 + \mathbf{W}^T\mathbf{X}_k\mathbf{X}_k^T\mathbf{W} - 2d_k\mathbf{X}_k^T\mathbf{W} \tag{3.9}$$

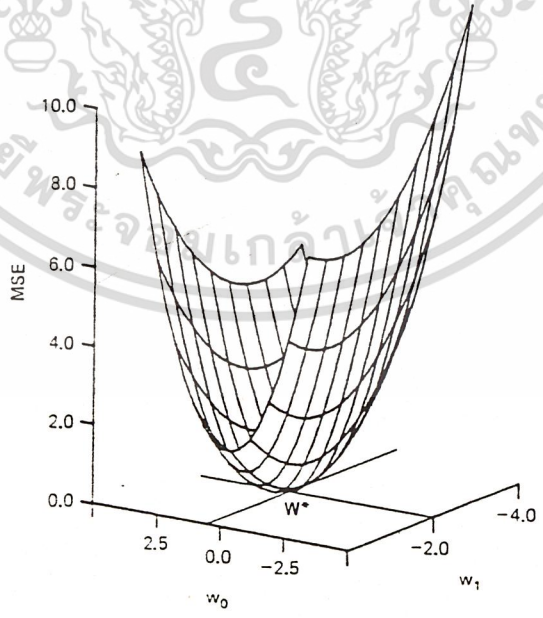
ดังนั้นค่าที่ว่าไว้คือ

$$E[\varepsilon_k^2] = E[d_k^2] + \mathbf{W}^T E[\mathbf{X}_k\mathbf{X}_k^T]\mathbf{W} - 2E[d_k\mathbf{X}_k^T]\mathbf{W} \tag{3.10}$$

นิยามให้ $\mathbf{R} = E[\mathbf{X}_k\mathbf{X}_k^T]$

$$= E\begin{bmatrix} X_{0k}^2 & X_{0k}X_{1k} & X_{0k}X_2 & \cdots & X_{0k}X_L \\ X_1X_{0k} & X_{1k}^2 & X_1X_2 & \cdots & X_1X_L \\ \vdots & \vdots & \vdots & & \vdots \\ X_LX_{0k} & X_LX_{1k} & X_LX_2 & \cdots & X_L^2 \end{bmatrix} \tag{3.11}$$

$$\mathbf{P} = E[d_k\mathbf{X}_k] = E[d_kX_{0k} \quad d_kX_{1k} \quad \cdots \quad d_kX_{Lk}]^T \tag{3.12}$$



รูป 3.5 รูปพื้นผิวค่าผิดพลาด 2 มิติ

เราจะได้ว่า ค่าผิดพลาดเฉลี่ยยกกำลังสองในสมการ (3.10) ให้สัญลักษณ์เป็น ξ และเขียนได้เป็น

$$MSE \cong \xi = E[\varepsilon_k^2] = E[d_k^2] + \mathbf{W}^T\mathbf{R}\mathbf{W} - 2\mathbf{P}^T\mathbf{W} \quad (3.13)$$

เราสามารนำสมการ (3.13) ไปเขียนเป็นรูปพื้นผิวของค่าผิดพลาด ได้ดังรูป 3.5 ซึ่งจะเห็นเป็นรูปร่างถ้วยพาราโบลอยด์ ที่มีก้นถ้วยเป็นจุดที่ค่าถ่วงน้ำหนัก ทำให้เกิดค่าความผิดพลาดต่ำที่สุด (จุด $\mathbf{W}^*$)

3.3 เกรเดียนท์และค่าผิดพลาดเฉลี่ยยกกำลังสองต่ำที่สุด (Gradient and Minimum Mean-Square Error)

ในการหาค่าต่ำที่สุดของพื้นผิวค่าผิดพลาดนั้น เราใช้วิธีการของเกรเดียนท์ สัญลักษณ์ของเกรเดียนท์ คือ

$$\nabla \cong \frac{\partial \xi}{\partial \mathbf{W}} = 2\mathbf{R}\mathbf{W} - 2\mathbf{P} \quad (3.14)$$

จะได้จุดที่มีค่าผิดพลาดเฉลี่ยยกกำลังสองต่ำที่สุด ก็คือจุดที่มีค่าถ่วงน้ำหนักเป็น $\mathbf{W}^*$ ซึ่งก็คือ จุดที่มีค่าเกรเดียนท์เป็นศูนย์

$$\nabla = 0 = 2\mathbf{R}\mathbf{W}^* - 2\mathbf{P} \quad (3.15)$$

$$\mathbf{W}^* = \mathbf{R}^{-1}\mathbf{P} \quad (3.16)$$

แทนลงในสมการ (3.13) จะได้ค่าผิดพลาดเฉลี่ยยกกำลังสองต่ำที่สุด คือ

$$\xi_{min} = E[d_k^2] + \mathbf{W}^{*T}\mathbf{R}\mathbf{W}^* - 2\mathbf{P}^T\mathbf{W}^*$$

$$= E[d_k^2] + [\mathbf{R}^{-1}\mathbf{P}]^T\mathbf{R}\mathbf{R}^{-1}\mathbf{P} - 2\mathbf{P}^T\mathbf{R}^{-1}\mathbf{P} \quad (3.17)$$

3.4 การแสดงโดยใช้เกรเดียนท์ (Expression of Gradient)

เนื่องจากค่าผิดพลาดเฉลี่ยยกกำลังสองต่ำที่สุดนั้นเป็นรูปแบบทางควอดราติก (Quadratic) ของ $\mathbf{W}$ ซึ่งจะทำให้มีค่าต่ำที่สุดเมื่อ $\mathbf{W}$ เท่ากับ $\mathbf{W}^*$ ดังนั้นจึงแสดงได้ว่า

$$\xi = \xi_{min} + (\mathbf{W}-\mathbf{W}^*)^T\mathbf{R}(\mathbf{W}-\mathbf{W}^*) \quad (3.18)$$

เรากำหนดให้

$$\mathbf{V} = \mathbf{W} - \mathbf{W}^* = [v_0 \;\; v_1 \;\; \ldots \;\; v_L]^T \quad (3.19)$$

จะได้

$$\xi = \xi_{min} + \mathbf{V}^T\mathbf{R}\mathbf{V} \quad (3.20)$$

เกรเดียนท์ของค่าเฉลี่ยยกกำลังสองน้อยที่สุด ในความสัมพันธ์กับ V จะได้

$$\frac{\partial \xi}{\partial \mathbf{v}} = \left[\frac{\partial \xi}{\partial v_0} \;\; \frac{\partial \xi}{\partial v_1} \;\; \ldots \;\; \frac{\partial \xi}{\partial v_L}\right] = 2\mathbf{R}\mathbf{V} \quad (3.21)$$

เกรเดียนท์ที่ได้จะเหมือนกับในสมการ (3.14) เพราะผลต่างของ **W** และ **V** เป็นค่าคงที่ ดังนั้น

$$\nabla = \frac{\partial \xi}{\partial w} = \frac{\partial \xi}{\partial v} = 2\mathbf{RV} = 2(\mathbf{RW}-\mathbf{P}) \qquad (3.22)$$

สมการ (3.22) นี้ เราจะนำไปใช้ในการวิเคราะห์อัลกอร์ลิทึ่ม ในการปรับตัวเองได้ต่อไป

บทที่ 4 คุณสมบัติของพื้นผิวค่าผิดพลาดที่เป็นควอดราติก (Properties of The Quadratic Performance Surface)

4.1 คุณสมบัติของพื้นผิวค่าผิดพลาด

คุณสมบัติของพื้นผิวค่าผิดพลาด เราพิจารณาได้ด้วยสมการค่าผิดพลาดเฉลี่ยยกกำลังสองต่ำที่สุด ในสมการ (3.20) ซึ่งจะแสดงในเทอมของ เมตริกซ์ **R**

$$\xi = \xi_{min} + (\mathbf{W} - \mathbf{W}^*)^T \mathbf{R}(\mathbf{W} - \mathbf{W}^*)$$
$$= \xi_{min} + \mathbf{V}^T\mathbf{R}\mathbf{V} \qquad (3.20)$$

จะเห็นได้ว่าเราศึกษาลักษณะของพื้นผิวได้จาก **R** ในเทอมของ ค่าไอเกน(Eigenvalues) และ เวคเตอร์ไอเกน(Eigenvectors)

พิจารณา ค่าไอเกนของเมตริกซ์ **R** จากสมการ

$$[\mathbf{R} - \lambda\mathbf{I}]Q_n = 0 \qquad (4.1)$$

λ : ค่าสเกลาร์ที่เปลี่ยนแปลงได้

Q_n : เวคเตอร์หลัก

I : เมตริกซ์เอกลักษณ์

เราจะหาค่า **R** และ Q_n ก็โดยการใช้ดีเทอร์มิแนนท์

$$\det[\mathbf{R} - \lambda\mathbf{I}] = 0 \qquad (4.2)$$

เมื่อแก้สมการออกมาแล้ว **R** ซึ่งมีดีกรีอยู่ L+1 จะได้ค่า ออกมาเป็น เรียกว่าค่าไอเกนของ **R**

เราสามารถหาค่า Q_n ได้จากแต่ละค่าของค่าไอเกนดังนี้

$$\mathbf{R}Q_n = \lambda_n Q_n \qquad (4.3)$$

ซึ่งเรียก Q_n ว่าเวคเตอร์ไอเกนที่ n ของ **R**

$$\mathbf{R}[\,Q_0\;\; Q_1 \cdots Q_L\,] = [\,Q_0\;\; Q_1 \cdots Q_L\,]\begin{bmatrix} \lambda_0 & 0 & \cdots & 0 \\ 0 & \lambda_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_L \end{bmatrix} \qquad (4.4)$$

$$\mathbf{R}Q = Q\Lambda \quad \text{หรือ} \quad \mathbf{R} = Q\Lambda Q^{-1} \qquad (4.5)$$

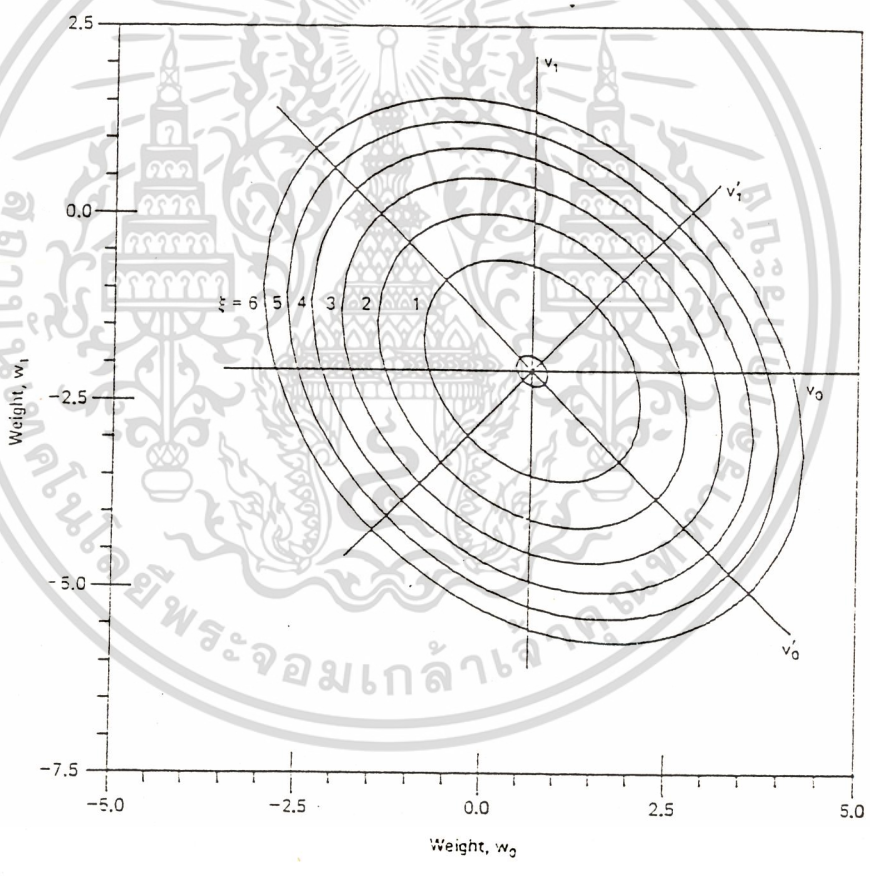
Λ เรียกว่า เมตริกซ์ของค่าไอเกน (Eigenvalue matrix)

Q เรียกว่า เมตริกซ์ของเวคเตอร์ไอเกน (Eigenvector matrix)

4.2 ความสำคัญของค่าไอเกน และเวคเตอร์ไอเกน (Significance of Eigenvalues and Eigenvectors)

ค่าไอเกน และ เวคเตอร์ไอเกน มีความสัมพันธ์ต่อคุณสมบัติของพื้นผิวค่าผิดพลาดเป็นอย่างมาก ค่าผิดพลาด ξ และค่าถ่วงน้ำหนัก $\mathbf{W}$ ที่มี L+1 ตัว เราจะนำไปพล๊อตได้ใน L+2 มิติ ซึ่งแต่ละแกน ก็คือ $\xi, w_0, w_1, \ldots, w_L$ อย่างเช่นพล๊อตรูปพื้นผิวค่าผิดพลาด ที่พิจารณาค่าถ่วงน้ำหนัก 2 ตัว ก็จะมี 3 แกน คือ ξ, w_0, w_1 ดังรูป 3.5

กรณีค่าถ่วงน้ำหนัก 2 ตัว จะได้เป็นพาราโบลอยด์ของค่าผิดพลาด ดังรูป 3.5 แต่เมื่อเราตัดพาราโบลอยด์ขนานกับระนาบ $w_0 w_1$ เราจะได้รูปวงรี ดังรูป 4.1



รูป 4.1 วงรีที่เกิดจากการพิจารณาค่าผิดพลาดบนระนาบ w_0-w_1

จากสมการ (3.13) จะได้สมการของค่าผิดพลาดที่เป็นรูปวงรีบนระนาบ $w_0 w_1$

$$\mathbf{W}^T\mathbf{R}\mathbf{W} - 2\mathbf{P}^T\mathbf{W} = \text{ค่าคงที่} \qquad (4.6)$$

สมการ (4.6) นี้จะพิจารณาบนระนาบ $\mathbf{W}_0\mathbf{W}_1$ เรายังแปลงตัวแปรจาก $\mathbf{W}$ ไปเป็น $\mathbf{V}$ ได้ โดยพิจารณาร่วมกับสมการ (3.16)

$$\mathbf{V} = \mathbf{W} - \mathbf{R}^{-1}\mathbf{P} = \mathbf{W} - \mathbf{W}^{*} \tag{4.7}$$

ดังนั้นสมการ (4.6) จะเปลี่ยนเป็น

$$\mathbf{V}^{T}\mathbf{R}\mathbf{V} = \text{ค่าคงที่} \tag{4.8}$$

จะได้เป็นรูปวงรี ที่มีจุดเริ่มต้นบนระนาบ $\mathbf{V}_0\mathbf{V}_1$ โดยที่มีแกน $\mathbf{V}_0$ และ $\mathbf{V}_1$ เป็นแกนหลัก เราจะคิดว่า วงรี มีสมการตาม $F(\mathbf{V}) = \mathbf{V}^{T}\mathbf{R}\mathbf{V}$ และหากเกรเดียนท์ของ F

$$\nabla = \left[\frac{\partial F}{\partial v_0} \quad \frac{\partial F}{\partial v_1} \quad \cdots \quad \frac{\partial F}{\partial v_L} \right]^{T}$$

$$= 2\mathbf{R}\mathbf{V} \tag{4.9}$$

เวคเตอร์ที่ผ่านจุดเริ่มต้น $\mathbf{V} = 0$ เราจะได้เป็น $\mu\mathbf{V}$ แต่แกนหลักที่ผ่านจุดเริ่มต้น จะอยู่ในฟังก์ชั่น F(v) ซึ่งจะได้ว่า

$$2\mathbf{R}\mathbf{V}' = \mu\mathbf{V}'$$

$$\left[\mathbf{R} - \frac{\mu}{2}\mathbf{I} \right]\mathbf{V}' = 0 \tag{4.10}$$

ซึ่ง V เราให้เป็นแกนหลัก จากผลในสมการ (4.10) เมื่อเทียบกับสมการ (4.1) จะได้ว่า $\mathbf{V}$ จะเป็น เวคเตอร์ไอเกนของเมตริกซ์ $\mathbf{R}$ ดังนั้นเราจึงสรุปได้ว่า

"*เวคเตอร์ไอเกนของ เมตริกซ์อินพุท จะเป็นแกนหลักของพื้นผิวค่าผิดพลาด*"

ดังนั้นเราจะได้ค่าจากการพิจารณาสมการ (3.18), (3.20) และ(4.5)

$$\xi = \xi_{min} + (\mathbf{W}-\mathbf{W}^{*})^{T}\mathbf{R}(\mathbf{W}-\mathbf{W}^{*}) \tag{4.11}$$

$$= \xi_{min} + \mathbf{V}^{T}\mathbf{R}\mathbf{V} \tag{4.12}$$

$$= \xi_{min} + \mathbf{V}^{T}(\mathbf{Q}\,\Lambda\mathbf{Q}^{T})\mathbf{V}$$

$$= \xi_{min} + (\mathbf{Q}^{T}\mathbf{V})\Lambda\nabla(\mathbf{Q}^{T}\mathbf{V})$$

$$= \xi_{min} + \mathbf{V}'^{T}\,\Lambda\,\mathbf{V}' \tag{4.13}$$

จากสมการ (4.13) เมื่อเราคิดเกรเดียนท์ได้

$$\nabla = 2\Lambda\mathbf{V}'$$

$$= 2\,[\lambda_0 v_0' \quad \lambda_1 v_1' \quad \ldots \quad \lambda_L v_L']^{T} \tag{4.14}$$

การเปลี่ยนรูปในสมการ (4.12) และ (4.13)

การแปลง(Translation) : $\mathbf{V} = \mathbf{W} - \mathbf{W}^{*}$ (4.15)

การหมุน(Rotation) : $\mathbf{V}' = \mathbf{Q}^{T}\mathbf{V} = \mathbf{Q}^{-1}\mathbf{V}$ (4.16)

เกรเดียนท์ของ E เมื่อเทียบกับแกนหลัก

$$\frac{\partial \xi}{\partial v'_n} = 2\lambda_n v_n' \tag{4.17}$$

และจะได้

$$\frac{\partial^2 \xi}{\partial v'^2_n} = 2\lambda_n \quad ; \quad n = 0,1,\ldots,L \tag{4.18}$$

จะเห็นว่าการทำการเกรเดียนท์สองครั้งของค่าผิดพลาดเทียบกับค่าแกนหลัก v_n' จะได้ค่าสองเท่าของค่าไอเกน ซึ่งเราสรุปได้ว่า

"ค่าไอเกนของอินพุทเมตริกซ์ ก็คือ การหาเกรเดียนท์ของค่าผิดพลาดของพื้นผิวสองครั้งเมื่อคิดเทียบกับแกนหลักของ ξ"

บทที่ 5 การหาพื้นผิวค่าผิดพลาด (Searching the Performance Surface)

พื้นผิวของค่าผิดพลาดดังในรูปที่แสดงให้เห็นมีประโยชน์เป็นอย่างมากในการพิจารณาค่าผิดพลาดที่เกิดขึ้น จากค่าถ่วงน้ำหนักที่ใช้ เราจึงสามารถที่หาค่าถ่วงน้ำหนักที่จะทำให้ค่าผิดพลาดที่เกิดขึ้นมีค่าน้อยที่สุด ซึ่งก็คือจุดยอดของรูปพาราโบลอยด์ ต่อไปก็จะเป็นพื้นฐานและหลักการในการหาพื้นผิวค่าผิดพลาดด้วยวิธีการต่าง ๆ

5.1 วิธีในการหาพื้นผิวค่าผิดพลาด (Methods of Searching The Performance Surface)

วิธีการที่ใช้ในการหาพื้นผิวค่าผิดพลาด ที่นิยมใช้มีอยู่ 2 วิธี คือ วิธีของนิวตัน (Newton 's Method) และวิธีการลดความชัน (The Methods of Steepest Descent)

วิธีการของนิวตัน จะมีพื้นฐานทางคณิตศาสตร์เป็นอย่างมาก โดยจะใช้วิธีทางเกรเดียนท์เพื่อหาค่าถ่วงน้ำหนักที่เปลี่ยนแปลงไปในแต่ละขั้นตอน หรือแต่ละรอบ ลักษณะการเปลี่ยนจะเป็นในทิศทางที่จะลดพื้นผิวค่าผิดพลาดให้น้อยที่สุด วิธีการของนิวตันนี้จะค่อนข้างยากในการนำไปใช้งาน

วิธีการลดค่าความชัน เป็นวิธีการที่นำไปประยุกต์ใช้ได้อย่างมาก ในขั้นตอนการหาทางเกรเดียนท์เพื่อหาค่าถ่วงน้ำหนัก จะทำเป็นขั้นตอน หรือเป็นรอบซ้ำ ๆ กัน โดยที่การเปลี่ยนแปลงค่าถ่วงน้ำหนักจะมีทิศทางเกรเดียนท์ตรงข้ามของพื้นผิวค่าผิดพลาด ซึ่งไม่จำเป็นต้องเป็นทิศทางที่ให้ค่าพื้นผิวต่ำที่สุด

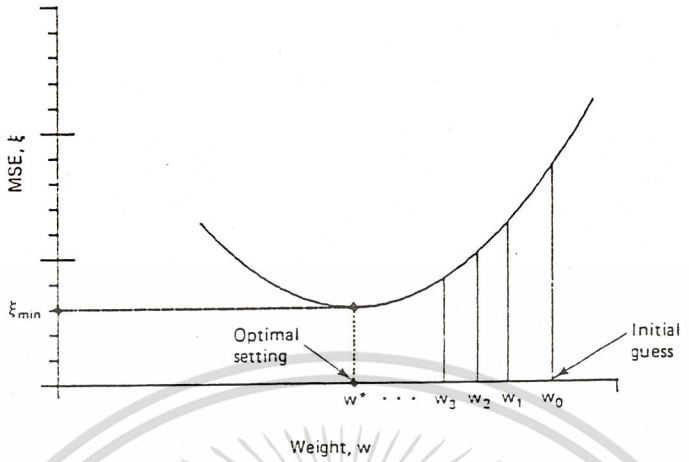
5.2 พื้นฐานของแนวความคิดในการหาโดยใช้วิธีทางเกรเดียนท์ (Basic Ideas of Gradient Search Methods)

แนวความคิดพื้นฐานของการหาโดยใช้วิธีทางเกรเดียนท์ เราจะพิจารณากรณีที่ง่ายที่สุด ก็คือ การใช้ค่าถ่วงน้ำหนักเพียงตัวเดียว โดยที่เราสามารถเห็นรูปร่างของพื้นผิวค่าผิดพลาดจากการใช้ค่าถ่วงน้ำหนักเพียงค่าเดียว ดังรูป 5.1 ซึ่งเป็นรูปพาราโบล่า ดังที่มีสมการเป็นดังนี้

$$\xi = \xi_{min} + \lambda(w - w^*)^2 \qquad (5.1)$$

$$\frac{d\xi}{d} = 2\lambda(w - w^*) \qquad (5.2)$$

$$\frac{d^2\xi}{d^2} = 2\lambda \qquad (5.3)$$



รูป 5.1 รูปพาลาโบลาที่เกิดจากการพิจารณาพื้นผิวค่าผิดพลาดเมื่อใช้ค่าถ่วงน้ำหนักเพียงค่าเดียว

ปัญหาก็คือเราจะหา w^* ซึ่งก็คือ ค่าถ่วงน้ำหนักที่ทำให้ค่าผิดพลาดเฉลี่ยยกกำลังสองมีค่าต่ำที่สุด เนื่องจากเราไม่ทราบว่าลักษณะพื้นผิวเป็นอย่างไร เราจะมีวิธีการในการหา โดยการเริ่มด้วยการกำหนดว่า w_0 ก่อน เริ่มที่ค่าใดก็ได้ให้เป็นค่าเริ่มต้น จากนั้นก็วัดค่าความชันที่จุด w_0 เราก็เลือกค่า w_1 ใหม่ ซึ่งจะมีค่าเท่ากับ w_0 บวกกับค่าที่เพิ่มขึ้นตามสัดส่วนของค่าความชันที่ลดลง ต่อไปในการหาจุด w_2 ก็ทำเหมือนกัน โดยทำการหาค่าความชันที่จุด w_1 ก่อน แล้วก็จะได้ w_2 คือ ค่า w_1 บวกกับค่าที่เพิ่มขึ้นตามสัดส่วนของค่าความชันที่ลดลง ขั้นตอนการหาก็จะทำซ้ำไปจนไปถึงค่า w^* ซึ่งเป็นค่าที่เราต้องการหา

ค่าที่ได้เกิดจากการวัดความชันของเส้นโค้งพื้นผิว ณ ที่จุด w_0, w1, w_2, ... เรียกว่า วิธีการประมาณโดยใช้เกรเดียนท์

5.3 อัลกอร์ลิทึ่มในการหาเกรเดียนท์และการแก้ปัญหา (A Gradient Search Algorithm and its Solution)

สำหรับค่าถ่วงน้ำหนักตัวเดียวในการหาเกรเดียนท์แบบซ้ำ เราอธิบายได้โดยใช้สมการ

$$w_{k+1} = w_k + \mu (-\nabla_k) \qquad (5.4)$$

k คือ จำนวนรอบในการทำซ้ำ

w_k คือ ค่าปัจจุบันที่จะปรับค่า

w_{k+1} คือ ค่าใหม่ที่ปรับค่าแล้ว

μ คือ ค่าคงที่ที่จะเสถียรเมื่ออยู่ในอัตราที่เหมาะสม ซึ่งจะกล่าวต่อไป

∇_k คือ เกรเดียนท์ที่ใช้ปรับค่า

เกรเดียนท์ ∇_k สำหรับค่าถ่วงน้ำหนักค่าเดียวจาก (5.2) คือ

$$\nabla_k = \left.\frac{d\xi}{d}\right|_{w=w_k} = 2\lambda(w - w^*) \tag{5.5}$$

จากสมการ (5.4) และ (5.5)

$$w_{k+1} = w_k - 2\mu\lambda(w_k - w^*) \tag{5.6}$$

$$w_{k+1} = (1 - 2\mu\lambda)w_k + 2\mu\lambda w_k \tag{5.7}$$

สมการ (5.7) จะเป็นสมการเชิงเส้นซึ่งเมื่อทำซ้ำโดยเริ่มที่ w_0 จะได้

$$w_1 = (1 - 2\mu\lambda)w_0 + 2\mu\lambda w^* \tag{5.8}$$

$$w_2 = (1 - 2\mu\lambda)^2 w_0 + 2\mu\lambda w^* [(1 - 2\mu\lambda) + 1] \tag{5.9}$$

$$w_3 = (1 - 2\mu\lambda)^3 w_0 + 2\mu\lambda w^* [(1 - 2\mu\lambda^2) + (1 - 2\mu\lambda) + 1] \tag{5.10}$$

ผลที่ได้ทำให้เราเขียนเป็นสมการทั่วไปได้เป็น

$$w_k = (1 - 2\mu\lambda)^k w_0 + 2\mu\lambda w^* \sum_{n=0}^{k-1} (1 - 2\mu\lambda)^n \tag{5.11}$$

$$= (1 - 2\mu\lambda)^k w_0 + 2\mu\lambda w^* \frac{1-(-2\mu\lambda)^k}{1-(-2\mu\lambda)} \tag{5.12}$$

$$= w^* + (1 - 2\mu\lambda)^k (w - w^*) \tag{5.13}$$

ซึ่งก็จะเป็นสมการหาค่าถ่วงน้ำหนักที่จุดต่างๆ ที่ต้องการหา

5.4 ความเสถียรและขอบเขตของการลู่เข้า (Stability and Rate of Convergence)

จากสมการ (5.13) เราให้ $r = 1-2\mu\lambda$ ซึ่ง (5.13) จะเสถียรได้ก็ต่อเมื่อ

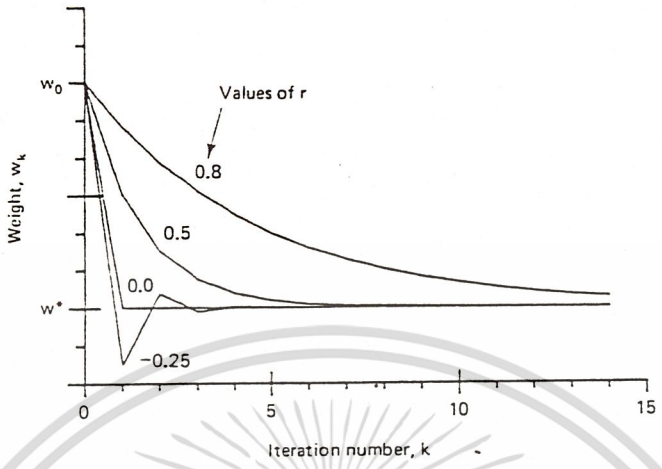
$$| r | = | 1-2\mu\lambda | < 1 \tag{5.14}$$

หรือ

$$\frac{1}{\lambda} > \mu > 0 \tag{5.15}$$

ซึ่งก็คือขอบเขตของค่า μ ที่จะทำให้ ค่า w_k มีความเสถียร

นอกจากนี้นในการปรับค่าถ่วงน้ำหนัก อาจจะทำให้เกิดผลที่ออกมาต่างกัน ถ้าค่า r ไม่เหมือนกัน ดังในรูป 5.2 จะเห็นได้ว่า ผลที่ออกมาไม่เหมือนกัน ดังนั้น ขอบเขตการลู่เข้าก็จะต่างกัน ซึ่งแสดงให้เห็นดังตาราง 5.1



รูป 5.2 ความแตกต่างในการปรับตัวของค่าถ่วงน้ำหนักเมื่อใช้ค่า r ต่างๆกัน

เสถียร (ลู่เข้า)	$0<\mu<\frac{1}{\lambda}$	$\|r\|<1$
โอเวอร์แดมพ์	$0<\mu<\frac{1}{2\lambda}$	$1>r>0$
คริติคอลแดมพ์	$\mu=\frac{1}{2\lambda}$	$r=0$
อันเดอร์แดมพ์	$\frac{1}{2\lambda}<\mu<\frac{1}{\lambda}$	$0>r>-1$
ไม่เสถียร(ไม่ลู่เข้า)	$\mu\geq\frac{1}{\lambda}$ and $\mu\leq 0$	$\|r\|>1$

ตาราง 5.1 แสดงผลของค่า μ ที่มีต่อการลู่เข้าของค่าถ่วงน้ำหนัก

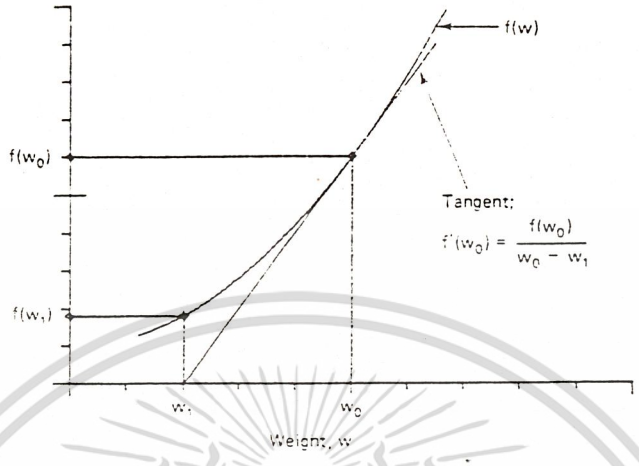
5.5 การหาเกรเดียนท์โดยวิธีของนิวตัน (Gradient Search by Newton's Method)

วิธีของนิวตันเป็นวิธีการหาซีโร่ของฟังก์ชัน ดังนั้นจึงเป็นการหาคำตอบของสมการ $f(w) = 0$ ซึ่งวิธีการ คือ เริ่มจากการเดาค่าถ่วงน้ำหนักเริ่มต้น w_0 จากนั้น ก็หา $f(w_0)$ แล้วเราก็จะหาค่า w_1 ได้ดังที่แสดงในรูป 5.3 w_1 ก็จะจุดที่เส้นแทนเจนท์(tangent) ของ $f(w_0)$ ตัดกับแกน w

$$f'(w_0) = \frac{f(w_0)}{w_0 - w_1}$$

หรือ

$$w_1 = w_0 - \frac{f(w_0)}{f'(w_0)} \qquad (5.16)$$



รูป 5.3 แสดงการใช้วิธีของนิวตันในการหาซีโร่ของ f(w)

และการหาจุด w_2 ก็จะทำการหาโดยใช้ w_1 เป็นเหมือนค่าเริ่มต้น และทำซ้ำเหมือนเดิม

$$w_{k+1} = w_k - \frac{f(w_k)}{f'(w_k)} \quad ; \quad k=0,1,... \tag{5.17}$$

สมการ (5.22) เป็นสมการต่อเนื่อง ดังนั้น ในการหา $f'(w)$ จะต้องเป็นการประมาณ ดังนี้

$$f'(w_k) \approx \frac{f(w_k) - f(w_{k-1})}{w_k - w_{k-1}} \tag{5.18}$$

จากสมการ (5.17) และ (5.18)

$$w_{k+1} = w_k - \frac{f(w_k)(w_k - w_{k-1})}{(f(w_k) - f(w_{k-1})} \quad ; \quad k=0,1,... \tag{5.19}$$

การประยุกต์ใช้วิธีของนิวตันในการหาพื้นผิวค่าผิดพลาด เราจะเริ่มที่สมการ $f(w) = 0$ ซึ่งจากสมการนี้ทำให้ได้ว่า $f(w) = 0$ หรือ $\nabla = 0$ ดังในสมการ (2.15) เนื่องจากเราต้องการหาค่า $\xi(w)$ ที่ต่ำที่สุด ดังนั้นเราจึงได้

$$f(w) = \xi'(w) \tag{5.20}$$

จากสมการ (5.17) จะได้

$$w_{k+1} = w_k - \frac{\xi'(w_k)}{\xi''(w_k)} \quad ; \quad k=0,1,... \tag{5.21}$$

การประยุกต์ใช้วิธีของนิวตันแบบคิดขั้นตอนเดียว ดังรูป 5.2 ใช้สมการ (5.2) ในอัลกอร์ลิทึมสมการ (5.21) และให้ $k = 0$

$$w_1 = w_0 - \frac{2\lambda(w_0 - w^*)}{2\lambda} = w^* \qquad (5.22)$$

จะเห็นได้ว่าการใช้วิธีของนิวตัน กรณีตัวแปรเดียว จะใช้ได้กับทุกค่าของ w ก่อเมื่อ ต้องเป็นฟังก์ชั่นควอดราติก

วิธีของนิวตันในกรณีตัวแปรเดียวนี้ จะมีวิธีที่ซับซ้อน ถ้าเป็นในตามกรณีนี้ คือ อย่างแรก เราไม่ทราบค่าอย่างแท้จริง เพราะเราจะต้องหา ξ' และ ξ'' และอย่างที่สอง คือ พื้นผิวไม่เป็นฟังก์ชันควอดราติก ซึ่งเมื่อเป็นตาม 2 กรณีนี้แล้ว จะทำให้การคิดเป็นไปได้ยาก

5.6 วิธีของนิวตันในกรณีหลายมิติ (Newton's Method In Multidimensional space)

ที่ผ่านมาคือ การหาค่าถ่วงน้ำหนัก w^* ภายในครั้งเดียว โดยที่มีตัวถ่วงน้ำหนักเพียงตัวเดียว หรือ มิติเดียว และพื้นผิวจะต้องเป็นควอดราติกด้วย ต่อไปจะเป็นการหาโดยใช้วิธีของนิวตันในหลายมิติ หรือ ตัวถ่วงน้ำหนักหลายตัว ซึ่งก็จะคล้ายกับวิธีของตัวถ่วงน้ำหนักตัวเดียว

จากสมการ (2.16)

$$\mathbf{W}^* = \mathbf{R}^{-1}\mathbf{P} \qquad (5.23)$$

หาเกรเดียนท์จากสมการ (2.13)

$$\nabla = 2\mathbf{R}\mathbf{W} - 2\mathbf{P} \qquad (5.24)$$

คูณเข้ากับสมการ (5.24)

$$\mathbf{W}^* = \mathbf{W} - \frac{1}{2}\mathbf{R}^{-1}\nabla \qquad (5.25)$$

เปลี่ยนให้อยู่ในรูปอัลกอร์ลิทึมของการปรับตัวเอง

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \frac{1}{2}\mathbf{R}^{-1}\nabla_k \qquad (5.26)$$

สมการ (5.26) จะเห็นว่าเป็นกรณีหลายตัวแปร เมื่อพื้นผิวค่าผิดพลาดเป็นควอดราติกแล้ว ขั้นตอนในการหา $\mathbf{W}^*$ จะใช้เพียงขั้นตอนเดียว ดังสมการ (5.25) รูป 5.4 แสดงให้เห็นรูปหน้าตัดของพื้นผิวที่มีตัวถ่วงน้ำหนัก 2 ตัว จะเห็นได้ว่า ค่าถ่วงน้ำหนักจาก $\mathbf{W}_0 = (w_{00}, w_{10})$ จะหา $\mathbf{W}^* = (w_0^*, w_1^*)$ ได้ในขั้นตอนเดียว

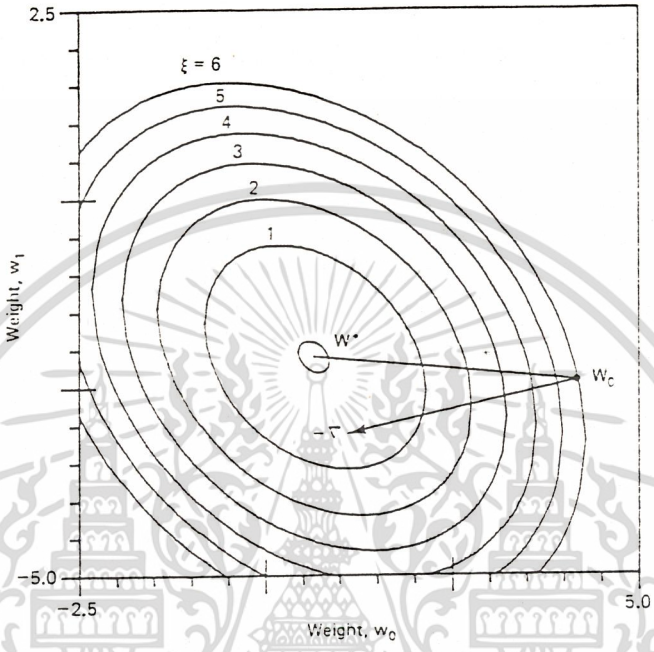
จากรูป 5.4 จะเห็นได้ว่าขั้นตอนวิธีของนิวตันจะไม่เป็นในทิศทางของเกรเดียนท์ แต่ทิศทางของค่าถ่วงน้ำหนักจะตั้งฉากกับเส้นรอบวงของพื้นผิว

พิจารณาสมการ (5.26) เราจะแทนค่าคงที่ด้วย μ ดัง (5.4) ด้วยอัตราของการลู่เข้าได้ดังนี้

$$\mathbf{W}_{k+1} = \mathbf{W}_k - \mu\mathbf{R}^{-1}\nabla_k \qquad (5.27)$$

ในที่นี้ $\mu = \frac{1}{2}$ แต่ขอบเขตการลู่เข้าควรเป็น

$$0 < \mu < 1 \tag{5.28}$$



รูป 5.4 แสดงการใช้วิธีของนิวตันโดยมีการใช้ตัวถ่วงน้ำหนัก 2 ตัว และ $\mu = 1$

แต่ในกรณีต้องการแบบโอเวอร์แคมพ์ μ ควรน้อยกว่า $\frac{1}{2}$

จากสมการ (5.27),(5.24) และ (5.23) จะได้

$$\mathbf{W}_{k+1} = (1 - 2\mu)\mathbf{W}_k + 2\mu\mathbf{W}^* \tag{5.29}$$

และจะได้

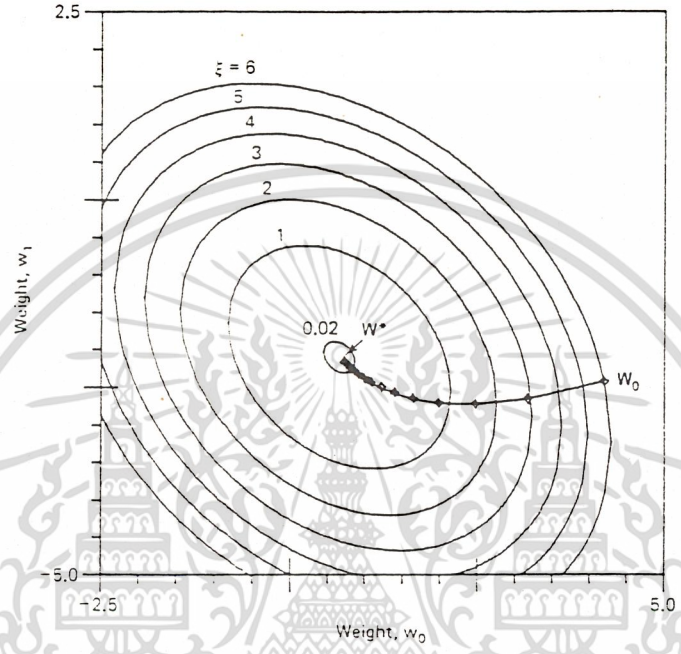
$$\mathbf{W}_k = \mathbf{W}^* + (1 - 2\mu)^k(\mathbf{W}_0 - \mathbf{W}^*) \tag{5.30}$$

5.7 การหาเกรเดียนท์โดยใช้วิธีการลดค่าความชัน (Gradient Search by The Method of Steepest Descent)

การหาโดยวิธีนี้จะต่างจากวิธีของนิวตัน โดยที่ทิศทางการเปลี่ยนค่าถ่วงน้ำหนักนี้ จะเปลี่ยนไปในทิศทางของเกรเดียนท์ในทุกขั้นตอน ดังในรูป 5.5 ซึ่งเป็นพื้นผิวควอดราติกเหมือนในรูป 5.4 เช่นกัน จะเห็นได้ว่า ในรูป 5.5 ซึ่งเป็นแบบการลดค่าความชันนั้น จะเป็นขั้นเล็กๆ ค่อยๆ เปลี่ยนแปลง แต่วิธีของนิวตันจะเห็นการเปลี่ยนแค่ขั้นเดียว หรือ เปลี่ยนแปลงอย่างรวดเร็ว

การลู่เข้าในขั้นตอนเดียว ด้วยวิธีของนิวตันนั้น เป็นวิธีการหยาบและเร็วเกินไป ไม่เป็นที่นิยม ในการวิเคราะห์ทางตัวเลขฟังก์ชันเป็นสิ่งที่ต้องรู้ แต่ในทางปฏิบัติ บางที่เราก็ไม่ทราบ จึง

ต้องหาโดยวิธีการวัดหรือประมาณข้อมูลที่เข้ามา ซึ่งการปรับตัวเองอย่างช้า จะเป็นประโยชน์อย่างมาก จึงทำให้วิธีการลดค่าความชันนิยมนำไปประยุกต์ใช้เป็นอย่างมาก



รูป 5.5 แสดงการใช้วิธีการลดค่าความชันโดยมีการตัวถ่วงน้ำหนัก 2 ตัว และ μ=1

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \mu\ (-\nabla_k) \tag{5.31}$$

จะเห็นได้ว่าสมการ (5.31) เหมือน (5.4) แต่จะเห็นหลายมิติ และพื้นผิวจะต้องเป็นควอดราติกเช่นเดิม เมื่อแทนสมการ (5.24) และ (5.23) จะได้

$$\mathbf{W}_{k+1} = \mathbf{W}_k - 2\mu\ \mathbf{R}\mathbf{V}_k$$

$$= \mathbf{W}_k + 2\mu\ \mathbf{R}(\mathbf{W}^* - \mathbf{W}_k) \tag{5.32}$$

$$\mathbf{W}_{k+1} = (\mathbf{I} - 2\mu\ \mathbf{R})\mathbf{W}_k + 2\mu\ \mathbf{R}\mathbf{W}^* \tag{5.33}$$

ใช้คุณสมบัติ $\mathbf{V} = \mathbf{W} - \mathbf{W}^*$

$$\mathbf{V}_{k+1} = (\mathbf{I} - 2\mu)\mathbf{V}_k \tag{5.34}$$

ด้วยคุณสมบัติ $\mathbf{V} = \mathbf{Q}\mathbf{V}'$ ดังนั้น

$$\mathbf{Q}\mathbf{V}_{k+1}' = (\mathbf{I} - 2\mu\mathbf{R})\mathbf{Q}\mathbf{V}_k' \tag{5.35}$$

คูณ $\mathbf{Q}^{-1}$

$$\mathbf{V}_{k+1}' = \mathbf{Q}^{-1}(\mathbf{I} - 2\mu\mathbf{R})\mathbf{Q}\mathbf{V}_k'$$

$$= (\mathbf{Q}^{-1}\mathbf{I}\mathbf{Q} - 2\mu\mathbf{Q}^{-1}\mathbf{R}\mathbf{Q})\mathbf{V}_k'$$

$$\mathbf{v}_{k+1}' = (\mathbf{I} - 2\mu\Lambda)\mathbf{v}_k' \tag{5.36}$$

เนื่องจากเมตริกซ์ของค่าไอเกน เป็นเมตริกซ์ทแยงมุม

$$\mathbf{v}_k' = (\mathbf{I} - 2\mu\Lambda)^k \mathbf{v}_0' \tag{5.37}$$

ซึ่งสมการ (5.37) แสดงให้เห็นว่า วิธีการลดความชัน จะเสถียรและลู่เข้าเมื่อ

$$\lim_{k\to\infty} (\mathbf{I} - 2\mu\Lambda)^k = 0 \tag{5.38}$$

จึงทำให้ขอบเขตของการลู่เข้า เป็นดังนี้

$$0 < \mu < \frac{1}{\lambda_{max}} \tag{5.39}$$

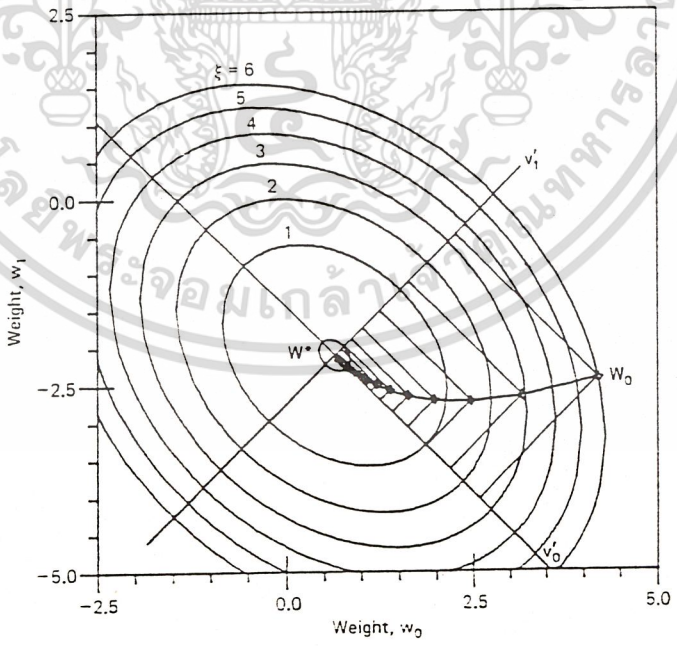
คือ ค่าไอเกนที่ใหญ่ที่สุดของ $\mathbf{R}$ ค่า μ นี้มีความสำคัญต่อการลู่เข้าเป็นอย่างมาก ถ้าสภาวะเป็นดังนี้จริง จะได้ว่า

$$\lim_{k\to\infty} \mathbf{V}_k' = 0 \tag{5.40}$$

จาก $\mathbf{V}' = \mathbf{Q}^{-1}\mathbf{V} = \mathbf{Q}^{-1}(\mathbf{W} - \mathbf{W}^*)$ จะได้ว่า

$$\lim_{k\to\infty} \mathbf{W}_k = \mathbf{W}^* \tag{5.41}$$

ซึ่งจะเห็นได้ว่าวิธีการลดค่าความชันนั้นจะเสถียรและลู่เข้า ถ้าค่าของ μ เป็นไปตามสมการ (5.39) เท่านั้น



รูป 5.6 แสดงการใช้วิธีการลดค่าความชันเมื่อมีค่าการใช้ตัวถ่วงน้ำหนัก 2 ตัว จะทำให้อัตราส่วนของค่า v_0' กับ v_1' มีค่าคงที่

เมื่อคูณสมการ (5.37) ด้วย Q

$$Q\mathbf{v}_k' = (\mathbf{I} - 2\mu\Lambda)^k \mathbf{v}_0' \tag{5.42}$$

ใช้คุณสมบัติ $\mathbf{v}_k' = Q^{-1}(\mathbf{w}_0 - \mathbf{w}^*)$ ซึ่งจะได้

$$\mathbf{w}_k = \mathbf{w}^* + Q(\mathbf{I} - 2\mu\Lambda)^k Q^{-1}(\mathbf{w}_0 - \mathbf{w}^*) \tag{5.43}$$

เมื่อแก้สมการแล้วจะได้

$$\mathbf{w}_k = \mathbf{w}^* + (\mathbf{I} - 2\mu\mathbf{R})^k (\mathbf{w}_0 - \mathbf{w}^*) \tag{5.44}$$

เราจะเห็นได้ว่าอัลกอร์ลิทึมของวิธีลดค่าความชัน จะเป็นไปในทิศทางของเกรเดียนท์ของพื้นผิวค่าผิดพลาด ดังรูป 5.6

บทที่ 6 อัลกอร์ลิทึ่มแบบค่าเฉลี่ยกำลังสองน้อยที่สุด (The LMS Algorithm)

เราได้รู้จัดกับอัลกอร์ลิทึ่ม 2 แบบ ได้แก่ วิธีของนิวต้น และ วิธีการลดค่าความชัน ซึ่งวิธีทั้ง 2 นั้น จะเป็นการหาโดยการประมาณค่าเกรเดียนท์ในแต่ละครั้ง แต่เราจะมีอีกอัลกอร์ลิทึ่ม ที่ใช้วิธีหาค่าความแตกต่างระหว่างค่า ξ ซึ่งวิธีนี้เรียกว่า อัลกอร์ลิทึ่มค่าเฉลี่ยกำลังสองน้อยที่สุด

อัลกอร์ลิทึ่ม แบบค่าเฉลี่ยกำลังสองน้อยที่สุดนี้ จะง่ายต่อการคำนวณ ซึ่งถ้าระบบปรับตัวเอง ประกอบด้วยตัวรวมที่ปรับตัวเองเชิงเส้น, เวคเตอร์อินพุท $\mathbf{X}_k$ และ ผลตอบสนองที่ต้องการแล้ว อัลกอร์ลิทึ่มแบบค่าเฉลี่ยแบบกำลังสองน้อยที่สุดนี้จะเป็นวิธีทีดีที่สุดในการประยุกต์ใช้

6.1 ที่มาของอัลกอร์ลิทึ่มแบบค่าเฉลี่ยกำลังสองน้อยที่สุด (Derivation of The LMS Algorithm)

จากพื้นฐานของตัวรวมที่ปรับตัวเองเชิงเส้นที่ผ่านมา พิจารณารูป 6.1 ซึ่งเราจะได้

$$\varepsilon_k = d_k - \mathbf{X}_k^T \mathbf{W}_k \tag{6.1}$$

$\mathbf{X}_k$ เป็นเวคเตอร์ของอินพุทที่สุ่มสัญญาณเข้ามาในการปรับตัวเองนั้น จะใช้การประมาณเกรเดียนท์ของ $\xi = E[\varepsilon_k^2]$ ดังนั้น การประมาณเกรเดียนท์

จากสมการ (5.31)

$$\hat{\nabla}_k = \begin{bmatrix} \frac{\partial \varepsilon_k^2}{\partial w_0} \\ \vdots \\ \frac{\partial \varepsilon_k^2}{\partial w_L} \end{bmatrix} = 2\varepsilon_k \begin{bmatrix} \frac{\partial \varepsilon_k}{\partial w_0} \\ \vdots \\ \frac{\partial \varepsilon_k}{\partial w_L} \end{bmatrix} = 2\varepsilon_k \mathbf{X}_k \tag{6.2}$$

นำไปใช้ในการหาค่าถ่วงน้ำหนัก

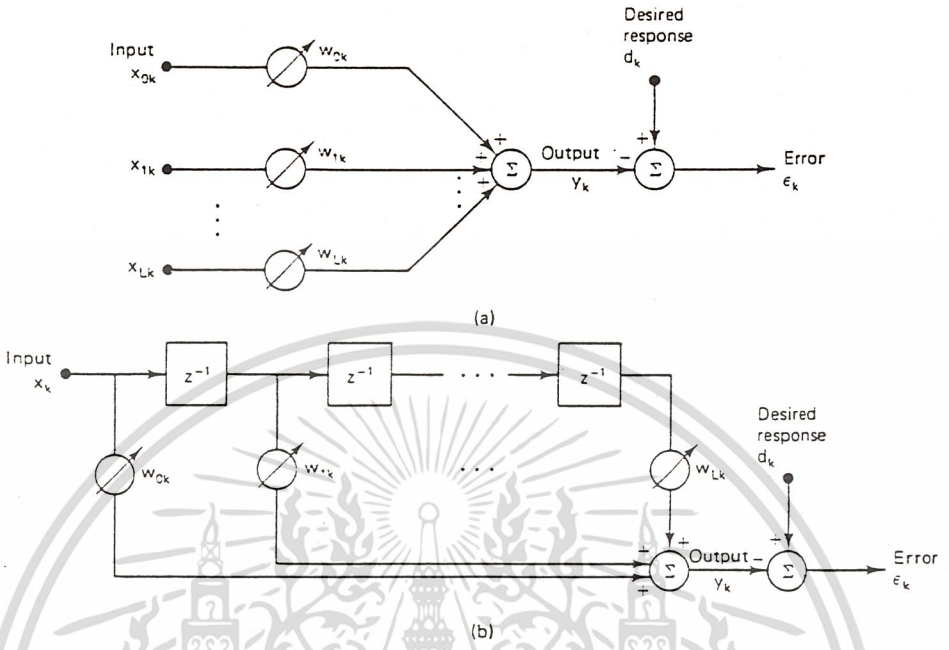
$$\begin{aligned} \mathbf{W}_{k+1} &= \mathbf{W}_k - \mu \hat{\nabla}_k \\ &= \mathbf{W}_k + 2\mu \varepsilon_k \mathbf{X}_k \end{aligned} \tag{6.3}$$

6.2 การลู่เข้าของเวคเตอร์ค่าถ่วงน้ำหนัก (Convergence of The Weight Vector)

อัลกอร์ลิทึ่มแบบค่าเฉลี่ยกำลังสองน้อยที่สุดนี้ ค่าเป็นค่าคงที่ที่จะบอกความเร็วและเสถียรภาพในการปรับตัวเอง ดังนั้น เราจะกำหนดขอบเขตการลู่เข้า จากค่า

$$\frac{1}{\lambda_{max}} > \mu > 0 \tag{6.4}$$

λ_{max} เป็นค่าไอเกนที่ใหญ่ที่สุด



รูป 6.1 ระบบปรับตัวเองเชิงเส้น (a) รูปแบบทั่วไป (b) แบบตัวกรองทรานซ์เวอร์ซอล

6.2 การลู่เข้าของเวคเตอร์ค่าถ่วงน้ำหนัก (Convergence of The Weight Vector)

อัลกอร์ลิที่มแบบค่าเฉลี่ยกำลังสองน้อยที่สุดนี้ ค่าเป็นค่าคงที่ที่จะบอกความเร็วและเสถียรภาพในการปรับตัวเอง ดังนั้น เราจะกำหนดขอบเขตการลู่เข้า จากค่า

$$\frac{1}{\lambda_{\max}} > \mu > 0 \qquad (6.4)$$

λ_{max} เป็นค่าไอเกนที่ใหญ่ที่สุด

นอกจากนี้เรายังจะกำหนดขอบเขตการลู่เข้าเพื่อสะดวกในการพิจารณาได้อีก เนื่องจากจะไม่สามารถที่จะมีค่ามากกว่า ผลรวมเส้นทแยงมุมของเวคเตอร์ **R** ได้ ดังนั้น

$$\lambda_{max} \leq \text{tr}[\Lambda] = \Sigma \text{ (ส่วนประกอบเส้นทะแยงมุมหลักของ } \Lambda)$$

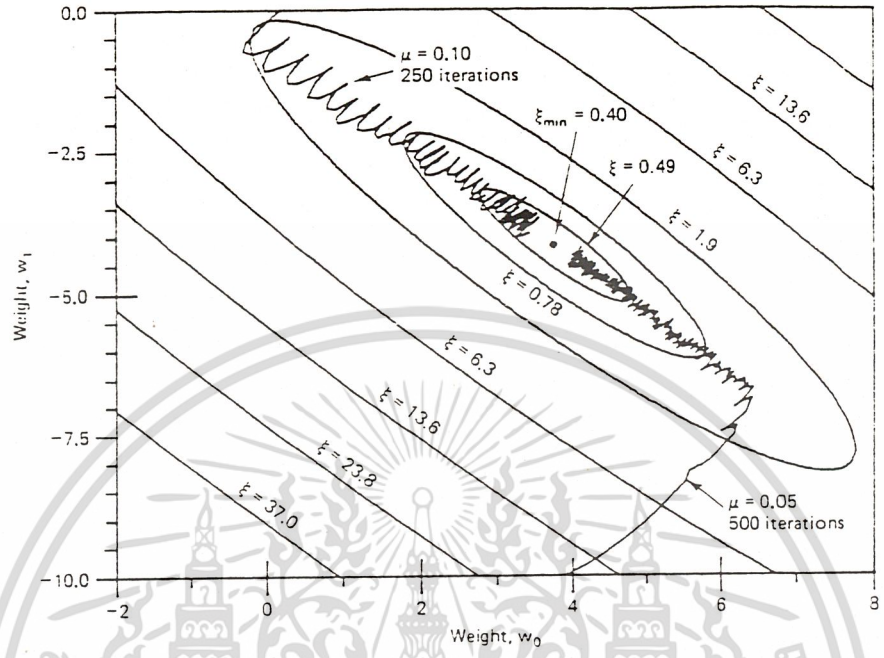
$$= \Sigma \text{ (ส่วนประกอบเส้นทะแยงมุมหลักของ } \mathbf{R}) = \text{tr}[\mathbf{R}] \qquad (6.5)$$

นอกจากนั้น ตัวกรองทรานซ์เวอร์ซอลที่ปรับตัวเองได้ เราสามารถแทนด้วยค่า (L+1) คูณกับกำลังของสัญญาณอินพุท

$$\text{แบบทั่วไป} : 0 < \mu < 1/\text{tr}[R] \qquad (6.5)$$

$$\text{ตัวกรองแบบทรานซ์เวอร์ซอล} : 0 < \mu < 1/(L+1)(\text{พลังงานของสัญญาณ}) \qquad (6.6)$$

จะเห็นได้ว่าขอบเขตจะจำกัดกว่าสมการ (6.4) แต่การประยุกต์ใช้และการคำนวณจะง่ายกว่าการหาค่าไอเกนของ **R**



รูป 6.2 แสดงเส้นทางพื้นผิวค่าผิดพลาดและการปรับตัวของค่าถ่วงน้ำหนัก
เมื่อใช้วิธีค่าเฉลี่ยยกกำลังสองน้อยที่สุด (N=16 ,$E[r_k^2] = 0.01$)

ลองพิจารณารูป 6.2 จะเห็นการเปรียบเทียบการใช้ค่า μ ที่ต่างกันจะมีผลอย่างไรต่อการปรับตัวเอง ซึ่งเราจะเห็นได้ว่า

กรณีแรก ใช้ค่า μ = 0.10 ค่าถ่วงน้ำหนักเริ่มต้น $w_0 = 0$, $w_1 = 0$จะใช้จำนวนรอบในการวนซ้ำ 250 รอบ

กรณีที่สอง ใช้ค่า μ = 0.50 ค่าถ่วงน้ำหนักเริ่มต้น $w_0 = 4$, $w_1 = -10$ จะใช้จำนวนรอบในการวนซ้ำถึง 500 รอบ

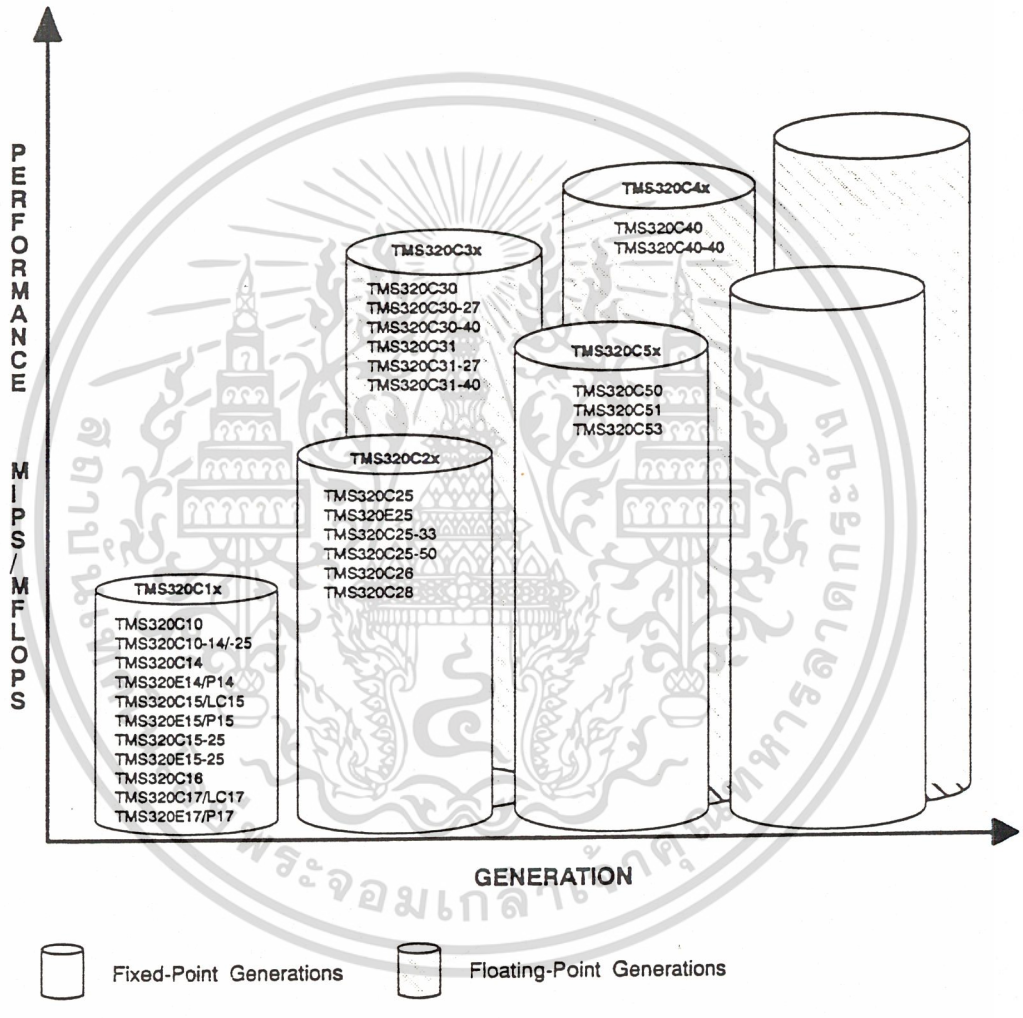
ซึ่งจะเห็นได้ว่า ค่า μ มีผลต่อการปรับตัวเอง ถ้าเราสังเกตุจากรูป 6.2 จะเห็นว่า ที่ มีค่ามาก จะมีการปรับตัวอย่างรวดเร็วและแต่ละครั้งจะเปลี่ยนแปลงมาก แต่ μ น้อย จะค่อยๆ เปลี่ยนแปลง

ดังนั้นการเลือกค่า μ จึงมีผลต่อการปรับตัวเองเป็นอย่างมาก

บทที่ 7 การ์ด TMS320c5X

7.1 แนะนำการ์ดประมวลสัญญาณเชิงเลข

ภาพรวมของ IC ตระกูล TMS 320c5X



รูปที่ 7.1 ไอซีตระกูล TMS320c5X

บทนี้จะกล่าวถึง IC TMS 320c50 DSP พร้อม การ์ดในตระกูล TMS 320 มี CPU ภายในพัฒนาจาก 'c25 IC CPU รุ่นนี้มีความสามารถ ประมวลผลเร็วเป็น 2 เท่าของ 'c2X และ โค๊ดโปรแกรม(Source code) ใช้ได้กับ 'c1X และ 'c2X มีการขยายฟิกซ์พ้อยท์(fixed pointed) ได้เพื่อให้เหมาะกับการประมวลผลสัญญาณทางดิจิตอลในอนาคต

TMS 320c50 ประกอบด้วยอุปกรณ์ต่อไปนี้

- ซีมอสแบบสแตติก(Static CMOS DSP) ที่ประกอบด้วย แรม(RAM) บนชิพขนาด10 K
รอม(ROM) บนชิพขนาด 2 K

รายละเอียดของ IC TMS 320c50 ประกอบด้วยฟิกซ์พ้อยท์ 16 บิต และ โฟลทติ้งพอยท์ 32 บิต บนชิพ

- มีความสามารถทางการคำนวณทางคณิตศาสตร์สำหรับ ผู้ใช้ทั่วไป
- มีคำสั่งที่ยืดหยุ่น
- การทำงานที่ซ้ำซ้อนอยู่แล้วภายใน
- ความเร็วสูง ในการประมวลผล
- การทำงานแบบขนานของสถาปัตยกรรมรุ่นใหม่
- ราคาถูก

7.1.1 การประยุกต์ใช้งานการ์ด

ด้วยความคล่องตัวของการ์ดรุ่นนี้จึงสามารถนำมาทำงานแบบเวลาจริง(real time), ตัวอย่าง TMS 320c50 สามารถใช้ในการประมวลผลสัญญาณ เช่น ใช้ในการกรอง และยิ่งไปกว่านั้น ยังสามารถใช้กับการทำงานที่ซับซ้อนหลายอย่างพร้อมๆ กัน

ตาราง 7.1 การทำงานทั่วไปของการ์ด

เครื่องจักร	ผู้บริโภค	การควบคุม
-การปรับตัวเองในการขับขี่ -การเดินเรือ -การสั่งคำสั่งทางเสียง	-วิทยุและโทรทัศน์แบบดิจิตอล -หุ่นยนต์เพื่อการศึกษา -เครื่องวิเคราะห์เสียงดนตรี	-ควบคุมดิสก์ไดรว์ -ควบคุมเครื่องจักร -ควบคุมเซอร์โว
ใช้งานทั่วไป	ภาพและกราฟฟิก	การอุตสาหกรรม
-ตัวกรองแบบปรับตัวเองได้ -การคอนวอร์ลูชั่น -การคอร์รีเรชั่น	-การหมุนภาพ 3 มิติ -แผนที่ดิจิตอลและภาพเคลื่อนไหว -การมองเห็นของหุ่นยนต์	-การควบคุมทางตัวเลข -ควบคุมการส่งพลังงาน -หุ่นยนต์ใช้งาน
เครื่องวัด	เครื่องมือแพทย์	การทหาร
-ตัวกรองแบบดิจิตอล -วิเคราะห์สเปรคตรัม -วิเคราะห์ทรานเซี้ยน	-เครื่องมือวินิจฉัยโรค -วิเคราะห์การทำงานของหัวใจ -เครื่องช่วยฟัง	-การประมวลผลทางภาพ -กำหนดทิศทางขีปนาวุธ -เรดาร์

ดังที่กล่าวแล้วจึงนำการ์ดมาใช้กับอัลกอริทึมการลดเสียงรบกวน

7.1.2 รายละเอียดทั่วไปของการ์ด

TMS320'c5X ประกอบด้วย 'c50,'c51,'c53 ชิพ ทำจาก ซีมอสแบบสแตติก มีพื้นฐานจาก 'c25 โดยใช้สถาปัตยกรรมของฮาวาด(Harvard Architecture) รวมลงในชิพ ทำให้

- หน่วยความจำ (Memory) ติดต่อได้เร็ว
- ใช้ภาษาคำสั่งชั้นสูงได้ (High Language Instruction)
- โค๊ดโปรแกรมสามารถใช้ร่วมกับรุ่นก่อน
- การออกแบบใหม่ทำให้ชิพกินไฟน้อยลง

ตาราง 7.2 แสดงแรมและรอม

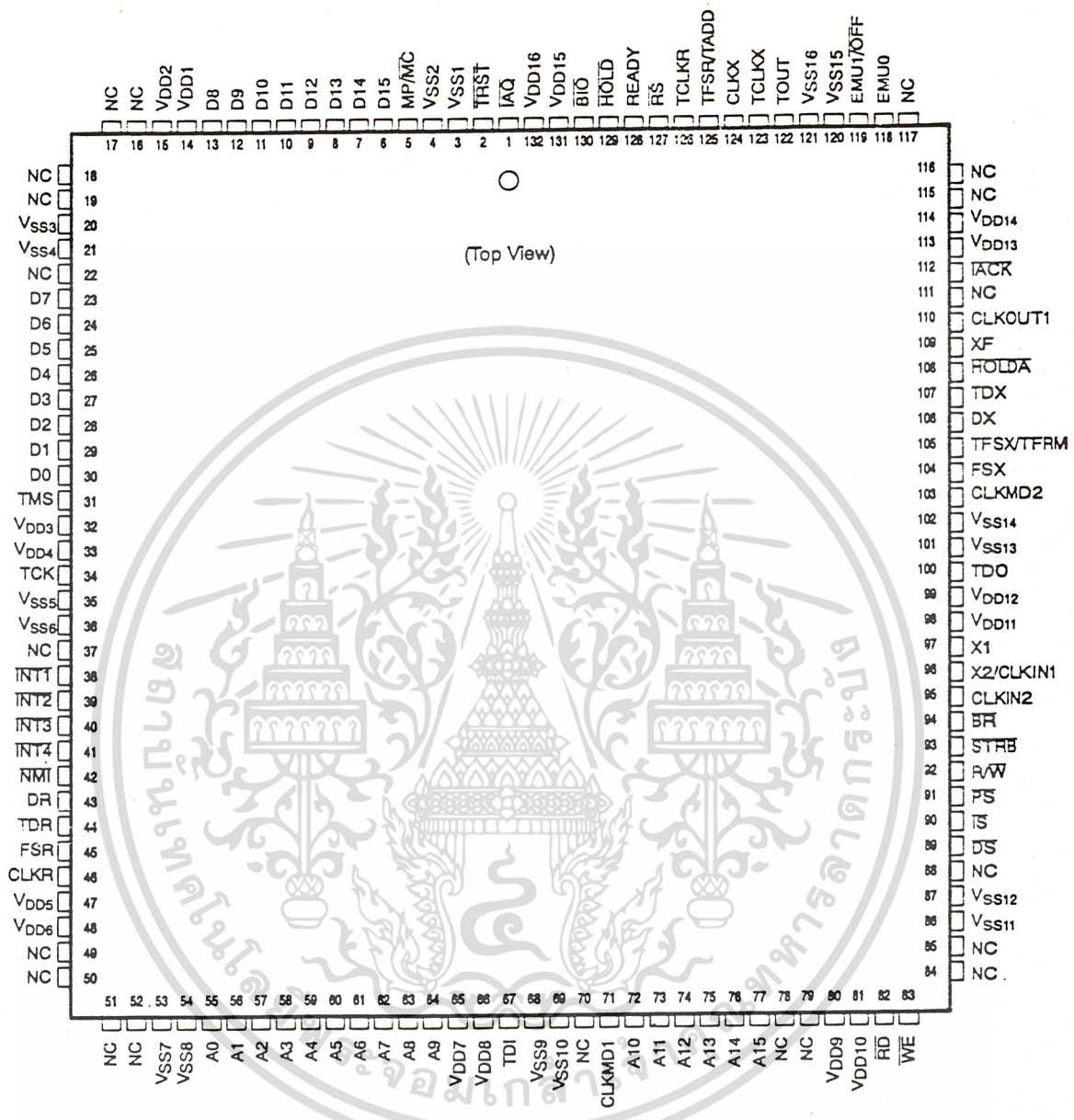
TMS 320 อุปกรณ์	หน่วยความจำ			I/O พอร์ท		เวลาใน 1 รอบ (ns)	ชนิดของแพคเกจ QFP
	รอม		แรม				
	ข้อมูล	ข้อมูล+โปรแกรม	โปรแกรม	อนุกรม	ขนาน		
TMS320c50	1K	9K	2K	2	64K	50/35	132-พิน

ตาราง 7.3 รายละเอียดขาสัญญาณต่างๆที่สำคัญ

จำนวน	ชื่อ	รายละเอียด
16	A0-A15	Parallel Address bus external data program memory
16	D0-D15	Parallel Data bus external data program memory
1	R/W	ขาสัญญาณ Read / Write
1	Strobe	ขาสัญญาณ Signal Strobe
1	BR	ขอใช้บัส (Bus Request)
1	IAR	Instruction acquisition
1	IACK	Interrupt Acknowledge - receive interrupt and program counter point to interrupt pointer
4	INT4-INT1	สัญญาณอินเทอร์รัปท์จากภายนอก
1	Serial	Port Signal
1	ClkXX TclkXX	Transmit Clock

7.1.3 รายละเอียดของขาและสัญญาณ

ชิพ มีขาทั้งหมด 132 ขาตามรูป 7.2



Note: NC = No connect. (These pins are reserved.)

รูป 7.2 รายละเอียดของขาสัญญาณบนชิพ

7.1.4 สถาปัตยกรรมของ IC และการ์ด

- หน่วยประมวลผล(CPU) มีการประมวลผลจากส่วนกลาง

เป็นสถาปัตยกรรมที่มีหน่วยประมวลผลทางตรรก(ALU) 32 บิต และ บัฟเฟอร์(Buffer) สำหรับการคูณและการบวกกัน และหน่วยประมวลผลตัวหลักมีการประมวลผลแบบขนาน (Parallel Logic Unit)

- การติดต่อกับอุปกรณ์ภายนอก (External Interface)

สามารถใช้พอร์ตที่มีอยู่ใช้ต่อเพิ่มได้เพื่อเป็น อินพุตหรือเอาท์พุต หรือการอินเทอร์รัปท์ นอกจากนี้ยังสามารถต่อหน่วยความจำโปรแกรมภายนอกและหน่วยความจำข้อมูลภายนอกได้

- หน่วยความจำ

หน่วยความจำภายในจัดการให้รีจิสเตอร์ ถูกจับคู่กับหน่วยความจำ (memory map) หน้า (page) 0 ตามตาราง 7.4

ตาราง 7.4 แสดงรีจิสเตอร์ที่จับคู่หน่วยความจำหน้าศูนย์

ชื่อ	ตำแหน่ง		รายละเอียด
	'C50 Dec	'C50 Hex	
-	0-3	0-3	Reserved
IMR	4	4	Interrupt mask Reg.
GREG	5	5	Global memory allocation register
IFR	6	6	Interrupt flag register
PMST	7	7	Processor mode status register
RPTC	8	8	Repeat counter register
BRCR	9	9	block repeat counter register
PASR	10	A	block repeat PC address start
PAER	11	B	block repeat PC end register
TREG0	12	C	Temporary register for Multiplicand
TREG1	13	D	Temporary register for dynamic shift
TREG2	14	E	Temporary register used as bit pointer
DBMR	15	F	Dynamic bit manipulation register
AR0-7	16-23	10-17	Auxiliary register zero to seven
INDX	24	18	Index register
ARCR	25	19	Auxiliary register compare register
CBSR1	26	1A	Circular buffer 1 start address register
CBER1	27	1B	Circular buffer 1 end address register
CBSR2	28	1C	Circular buffer 2 start address register
CBER2	29	1D	Circular buffer 2 end address register
CBCR	30	1E	Circular buffer control register
BMAR	31	1F	Block move address register

7.2 โหมดการติดต่อกับหน่วยความจำ

'c50 สามารถจัดการกับหน่วยความจำ ตั้งแต่ 0 ถึง 64K หน่วยความจำเก็บข้อมูลจึงมีการใช้วิธีการติดต่อหลายแบบดังนี้

- การติดต่อแบบโดยตรง (Direct Address Mode)

ใช้รีจิสเตอร์ชี้หน้าข้อมูล (Register DP) (รีจิสเตอร์ชี้หน้าข้อมูลขนาด 9 บิต) จะชี้ไปยังหน้าที่ต้องการ โดยแอดเดรส 9 บิตบน จะมาจากรีจิสเตอร์ชี้หน้าข้อมูล และ แอดเดรส 7 บิตล่างมาจาก ค่าตำแหน่งที่ต้องการในคำสั่ง โดยคำสั่งจะต้องมีบิตที่ 8 เป็น 0 แสดงว่า เป็นการอ้างอิงแบบโดยตรง

- การติดต่อแบบอ้างอิงหน่วยความจำ (Memory map Address Mode)

เหมือนกับการอ้างอิงแบบโดยตรง แต่รีจิสเตอร์ชี้หน้าข้อมูล จะเป็นศูนย์หมด เพราะฉะนั้นจึงสามารถทำการอ้างอิงกับ หน่วยความจำได้หน้าเดียวคือหน้าศูนย์

- การติดต่อแบบโดยอ้อม (Indirect Address Mode)

มีการเลือกรีจิสเตอร์ช่วย (Auxilary register) ก่อน โดยใช้ตัวชี้รีจิสเตอร์ช่วยขนาด 32 บิต ARP (Auxiliary register pointer 32bits) ชี้ AR0-7 แล้วนำค่าในรีจิสเตอร์ AR0-7 (16 bits) มาใช้ชี้ตำแหน่งข้อมูล

- การติดต่อแบบทันทีแบบสั้น (short immediate mode)

ในคำสั่งจะมีโอเปอแรนด์ (operand) อยู่แล้วเพราะฉะนั้นคำสั่งนี้จึงเสมือนกับไม่ได้อ้างอิงหน่วยความจำ แต่หลังจากทำคำสั่งเสร็จแล้วมีการนำค่าไปเก็บที่หน่วยความจำ จึงนำคำสั่งมารวมไว้ที่นี่ด้วย

- การติดต่อทันทีแบบยาว (long immediate mode)

ลักษณะการทำงานจะเหมือนกับการอ้างอิงแบบทันทีแบบสั้น แต่วิธีนี้จะมีค่าโอเปอแรนด์ที่ตามมายาว 16 บิต

- การติดต่อโดยใช้กลุ่มรีจิสเตอร์กลุ่ม (Register block Memory address mode)

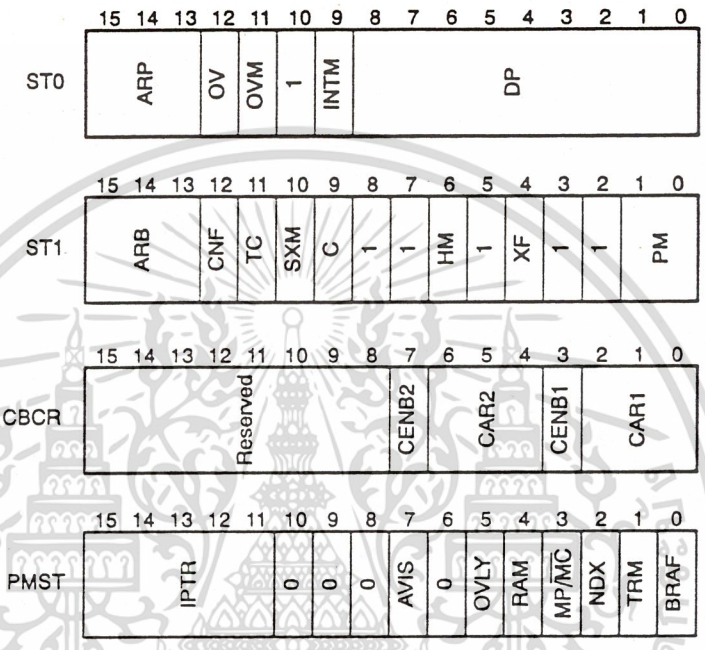
เหมือนโหมดการอ้างอิงทันทีแบบยาว แต่ค่าที่ใส่โปรแกรมเคาท์เตอร์ (PC) มาจากรีจิสเตอร์บีเอ็มพีอาร์ (BMPR register)

ภายในชิพ มีรีจิสเตอร์ช่วย 8 ตัวสามารถที่นำมาใช้ในการทำการอ้างอิงโดยอ้อม ซึ่งชี้โดยตัวชี้รีจิสเตอร์ช่วย การอ้างอิงค่าในตำแหน่ง สามารถนำค่าในรีจิสเตอร์ช่วย มารวมกับค่าในรีจิสเตอร์ดัชนี (Index Register) แล้วชี้ไปยังตำแหน่งได้ส่วนของการควบคุมระบบ ควบคุมโดย

- สแตก(hardware stack)
- โปรแกรมเคาท์เตอร์
- การรีเซตภายนอก(external reset)
- อินเทอร์รัปท์(interrupt)

ตำแหน่งการทำงานใช้โปรแกรมเคาท์เตอร์เป็นตัวชี้ตำแหน่ง การทำงานและมีสแตกลึก 8 ตำแหน่งสำหรับการอินเทอร์รัปท์และการเรียกโปรแกรมย่อย

7.2.1 รีจิสเตอร์ควบคุมและแสดงสถานะ



รูป 7.3 รีจิสเตอร์ควบคุมและแสดงสถานะ

รีจิสเตอร์ควบคุมและแสดงสถานะ มีรีจิสเตอร์ 4 ตัว (16 บิต)ตามรูป 7.3 ใช้กับงานนี้มี ST0,ST1 เก็บสถานะและโหมดการทำงานต่างๆ PMST, CBCR เก็บสถานะพิเศษ ค่าในรีจิสเตอร์ 4 ตัว สามารถเก็บในหน่วยความจำข้อมูล และเรียกออกมาได้ทำให้สะดวกในการทำงานเรียกโปรแกรมย่อย ST0, ST1, PMST ติดต่อกับ สแตก ขั้นที่ 1 เมื่อมีการอินเทอร์รัปท์ และคืนค่าจากสแตก เมื่อมีการคืนกลับมาจากอินเทอร์รัปท์ (RETI & RETE)

<u>สังเกต</u> XF ค่านี้ใน ST1 จะไม่ถูกบันทึกไว้

PMST, CBCR อยู่ในแผนผังหน่วยความจำหน้าที่ศูนย์ เพราะฉะนั้นสามารถแสดงค่าได้โดยตรงกับ CALU & PLU เพราะฉะนั้นสามารถเก็บเข้าสู่หน่วยความจำ ได้เหมือนการเก็บทั่วๆ ไป

ตาราง 7.5 หน้าที่ของบิตต่างๆ

Field	Function
ARB	Auxiliary Register Pointer จะทำการเก็บค่าเก่าของ ARP เดิม
ARP	Auxiliary Register Pointer เลือกใช้ชี้ AR register
AVIS	เป็นบิทแสดงค่าตำแหน่งโปรแกรมภายในไปที่ขา IC
BRAF	เป็นบิทแสดงว่ามีการทำคำสั่งเดิมซ้ำลูป
C	บิทตัวทด (Carry bit)
CNEB1,2	บิทเลือกการทำงานของบัฟเฟอร์หมุนเวียน (circular buffer) 1,2
CNF	การเลือกรูปแบบของแรมภายใน
DP	9 บิทเลือกหน้าของหน่วยความจำภายใน
INTM	บิทอนุญาตให้มีการอินเทอร์รัปท์
IPTR	5 บิทแสดงตำแหน่งการทำงานหลังอินเทอร์รัปท์
MP/MC	เลือกโหมดการทำงานระหว่างไมโครโปรเซสเซอร์หรือไมโครคอมพิวเตอร์
0V	บิทแสดงค่าการโอเวอร์โฟลว์(overflow)
OVLY	บิทที่อนุญาตให้นำค่าในหน่วยความจำโปรแกรมมาซ้อนกับค่าในหน่วยความจำข้อมูล
RAM	บิทที่อนุญาตให้นำค่าในหน่วยความจำข้อมูลมาซ้อนกับค่าในหน่วยความจำในโปรแกรม
SXM	บิทแสดงเครื่องหมาย

7.3 การอินเทอร์รัปท์ (Interrupt)

TMS320c50 รองรับ 16 อินเทอร์รัปท์ (INT16-INT1) โดยสัญญาณอินเทอร์รัปท์สร้างโดยการสื่อสารแบบอนุกรม (RINT, XINT, TRNT, TXNT),ไทม์เมอร์(timer)(TINT), ซอฟแวร์อินเทอร์รัป (Software interrupt),รีเซตและอินเทอร์รัปท์จากผู้ใช้ (RESET (RS) & USER Interrupt)

- RESET

1.เป็นการทำงานแบบนอนมาสค์เอเบิลอินเทอร์รัปท์ (nonmaskable interrupt)

2.โหลดค่าให้ PC มาชี้ที่ตำแหน่ง 0

3.BIT CNF จะถูกเคลียร์เป็น 0

7.3.1 การควบคุมภายนอก (Peripheral Control)

วงจรภายนอกถูกควบคุมโดยส่วนควบคุมแผนผังหน่วยความจำ การทำงานของพอร์ทอนุกรม (serial port) และไทม์เมอร์จะทำงานด้วยกัน เมื่อเกิดอินเทอร์รัปต์การเซตและการเคลียร์บิทสามารถที่จะจัดการระบบภายนอกได้ ข้อมูลสามารถโอนย้ายไปมาระหว่าง แผนผังหน่วยความจำรีจิสเตอร์ เมื่อระบบภายนอกไม่ใช้งานคล็อกภายในจะไม่ถูกต่อกับระบบภายนอก ทำให้ประหยัดพลังงานทั้งเมื่อระบบอยู่ในสภาวะทำงานและไม่ทำงาน

รีจิสเตอร์แผนผังหน่วยความจำและอินพุตและเอาท์พุตพอร์ท (memory map register & I/O port)

รีจิสเตอร์ทั้ง 28 ตัว จะถูกจัดอยู่ใน ช่องหน่วยความจำข้อมูลตามลำดับ นอกจากนั้นยังมีรีจิสเตอร์อีก 15 ตัว และ พอร์ทอินพุทเอาท์พุท 16 พอร์ท ที่ถูกใส่ในช่องหน่วยความจำข้อมูล

ตาราง 7.6 แสดงลำดับแผนผังหน่วยความจำของรีจิสเตอร์และอินพุทเอาท์พุตพอร์ท

Memory-Mapped Core Processor Registers			
Name	Address		Description
	Dec	Hex	
—	0–3	0–3	Reserved
IMR	4	4	Interrupt Mask Register
GREG	5	5	Global Memory Allocation Register
IFR	6	6	Interrupt Flag Register
PMST	7	7	Processor Mode Status Register
RPTC	8	8	Repeat Counter Register
BRCR	9	9	Block Repeat Counter Register
PASR	10	A	Block Repeat Program Address Start Register
PAER	11	B	Block Repeat Program Address End Register
TREG0	12	C	Temporary Register Used for Multiplicand
TREG1	13	D	Temporary Register Used for Dynamic Shift Count (5 bits only)
TREG2	14	E	Temporary Register Used as Bit Pointer in Dynamic Bit Test (4 bits only)
DBMR	15	F	Dynamic Bit Manipulation Register
AR0	16	10	Auxiliary Register Zero
AR1	17	11	Auxiliary Register One
AR2	18	12	Auxiliary Register Two

ตาราง 7.6 แสดงลำดับแผนผังหน่วยความจำของรีจิสเตอร์และอินพุทเอาท์พุตพอร์ท(ต่อ)

Memory-Mapped Core Processor Registers (Concluded)			
Name	**Address**		**Description**
	Dec	**Hex**	
AR3	19	13	Auxiliary Register Three
AR4	20	14	Auxiliary Register Four
AR5	21	15	Auxiliary Register Five
AR6	22	16	Auxiliary Register Six
AR7	23	17	Auxiliary Register Seven
INDX	24	18	Index Register
ARCR	25	19	Auxiliary Register Compare Register
CBSR1	26	1A	Circular Buffer 1 Start Register
CBER1	27	1B	Circular Buffer 1 End Register
CBSR2	28	1C	Circular Buffer 2 Start Register
CBER2	29	1D	Circular Buffer 2 End Register
CBCR	30	1E	Circular Buffer Control Register
BMAR	31	1F	Block Move Address Register
Memory-Mapped Peripheral Registers			
DRR	32	20	Data Receive Register
DXR	33	21	Data Transmit Register
SPC	34	22	Serial Port Control Register
—	35	23	Reserved
TIM	36	24	Timer Register
PRD	37	25	Period Register
TCR	38	26	Timer Control Register
—	39	27	Reserved
PDWSR	40	28	Program/Data S/W Wait-State Register
IOWSR	41	29	I/O S/W Wait-State Register
CWSR	42	2A	S/W Wait-State Control Register
—	43–47	2B–2F	Reserved
TRCV	48	30	TDM Data Receive Register
TDXR	49	31	TDM Transmit Data Register
TSPC	50	32	TDM Serial Port Control Register
TCSR	51	33	TDM Channel Select Register
TRTA	52	34	TDM Receive/Transmit Address Register
TRAD	53	35	TDM Received Address Register

ตาราง 7.6 แสดงลำดับแผนผังหน่วยความจำของรีจิสเตอร์และอินพุทเอาท์พุตพอร์ท(ต่อ)

Name	Address		Description
	Dec	Hex	
—	54–79	36–4F	Reserved
Memory-Mapped I/O Ports†			
PA0	80	50	I/O Port 50h
PA1	81	51	I/O Port 51h
PA2	82	52	I/O Port 52h
PA3	83	53	I/O Port 53h
PA4	84	54	I/O Port 54h
PA5	85	55	I/O Port 55h
PA6	86	56	I/O Port 56h
PA7	87	57	I/O Port 57h
PA8	88	58	I/O Port 58h
PA9	89	59	I/O Port 59h
PA10	90	5A	I/O Port 5Ah
PA11	91	5B	I/O Port 5Bh
PA12	92	5C	I/O Port 5Ch
PA13	93	5D	I/O Port 5Dh
PA14	94	5E	I/O Port 5Eh
PA15	95	5F	I/O Port 5Fh

ชิพตัวนี้ มีอินเทอรรัปท์ภายนอก แบบมาสค์เอเบิล (maskable) (interrupt INT1-4) 4 ตัว ซึ่งอุปกรณ์ภายนอกสามารถใช้ อินเทอรรัปท์ นี้ได้ มี อินเทอร์รัปท์ ภายนอก 1 ตัว (NMI) อินเทอร์รัปท์ภายใน จะกำเนิดโดย พอร์ทอนุกรม (RINT , XINT, Timer GINT) พอร์ท TDM, TRNT, TXNT และ คำสั่งซอฟแวร์อินเทอร์รัปท์ (TINT ,NMI และ INTR) การจัดลำดับความสำคัญถูกกำหนดไว้ดังนี้ รีเซต มีลำดับความสำคัญสูงสุด และ INT 4 มีลำดับความสำคัญต่ำที่สุด (NMI มีลำดับความสำคัญเป็นลำดับที่ 2)

7.3.2 ลำดับความสำคัญและตำแหน่งของอินเทอร์รัปท์ภายในและนอก

TRAP (ใช้เป็นอินเทอร์รัปท์ ซอฟแวร์) ไม่ถูกจัดอยู่ในระดับความสำคัญแต่รวมอยู่ในตารางเพราะมีตำแหน่งอินเทอร์รัปท์เวคเตอร์ (Interrupt vector) จึงนำมาเขียนรวมกันไว้ด้วย

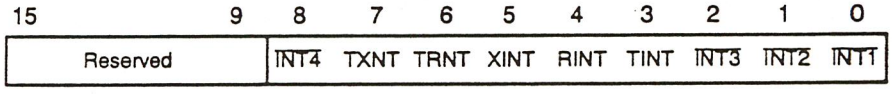
ตำแหน่งของอินเทอร์รัปท์เวคเตอร์ จะจัดโดย IPTR ซึ่งมี 5 บิท และอยู่ในส่วนของ PMST ค่าตำแหน่งแสดงในตาราง 7.7 ส่วน IPTR ถูกตั้งค่าเป็นศูนย์ขณะรีเซต เพราะฉะนั้นผลของการรีเซตอินเทอร์รัปท์เวคเตอร์จะมีค่าเท่ากับ 0000H ในหน่วยความจำโปรแกรม เวคเตอร์ของหน่วยความจำนี้สามารถเลือกชี้ไปที่อื่นในตำแหน่งใดๆก็ได้ภายใน 32K เวิร์ด ของหน่วยความจำ

โปรแกรมแรก ซึ่งทำโดยการโหลดค่าลงใน IPTR (มี 5 บิต) ตัวอย่างเช่น เวคเตอร์เลื่อนจุดเริ่มต้นไปสู่หน่วยความจำโปรแกรมโดย 1. เมื่อเกิดอินเทอร์รัปท์แทรพ(Interrupt Trap) ค่าของ IPTR จะถูกโหลดลงใน 5 ตำแหน่งที่มีนัยสำคัญสูงสุดของพีซี และตำแหน่งอ้างอิงสัมพัทธ์ที่เกิดจากอินเทอร์รัปท์แทรพจะถูกใส่ลงใน 6 บิตที่มีนัยสำคัญต่ำสุด(การทำงานเช่นนี้เกิดกับทุกอินเทอร์รัปท์ยกเว้นอินเทอร์รัปท์ที่เกิดจากการรีเซต เพราะ IPTR มีค่าเท่ากับศูนย์ขณะรีเซต

ตาราง 7.7 แสดงลำดับความสำคัญและตำแหน่งของอินเทอร์รัปท์ภายในและนอก

Name	Location		Priority	Function
	Dec	Hex		
RS	0	0	1 (highest)	External reset signal
NMI	36	24	2	Nonmaskable interrupt
INT1	2	2	3	External user interrupt #1
INT2	4	4	4	External user interrupt #2
INT3	6	6	5	External user interrupt #3
TINT	8	8	6	Internal timer interrupt
RINT	10	A	7	Serial port receive interrupt
XINT	12	C	8	Serial port transmit interrupt
TRNT	14	E	9	TDM port receive interrupt
TXNT	16	10	10	TDM port transmit interrupt
INT4	18	12	11	External user interrupt #4
——	20–33	14–21	N/A	Reserved
TRAP	34	22	N/A	Trap instruction vector
——	38–39	26–27	N/A	Reserved
——	40–63	28–3F	N/A	Software interrupts

อินเทอร์รัปท์แฟล็กรีจิสเตอร์อยู่ตำแหน่งที่6 ภายในช่องหน่วยความจำข้อมูล สามารถอ่านค่าที่ทำงานอยู่และสามารถเขียนเพื่อเคลียร์การทำงานอินเทอร์รัปท์ได้



รูป 7.4 แสดงอินเทอร์รัปท์แฟล็กรีจิสเตอร์

ยกตัวอย่างค่า IFR อ่านค่าได้เป็น 0005H ดังนั้น IP3 และ INT1 จะทำงานได้

อินเทอร์รัปท์ที่เกี่ยวข้องจะถูกเคลียร์โดยอัตโนมัติขณะที่เกิดการอินเทอร์รัปท์แทรพเมื่อซีพียูได้รับอินเทอร์รัปท์และมีการอ่านคำสั่งที่ตำแหน่งของอินเทอร์รัปท์เวคเตอร์ มันจะสร้างสัญญาณรับรู้อินเทอร์รัปท์ (Interrupt Acknowledge) ซึ่งจะไปเคลียร์แฟล็คของอินเทอร์รัปท์ที่เกี่ยวข้อง การรีเซตของวงจรก็สามารถเคลียร์อินเทอร์รัปท์ที่เกี่ยวข้องได้เหมือนกัน

TMS320c50 มีแผนผังหน่วยความจำรีจิสเตอร์อินเทอร์รัปท์มาสค์ (Interrupt Mask Register) สำหรับป้องกันอินเทอร์รัปท์ภายในและอินเทอร์รัปท์ภายนอก

รูป 7.5 แสดงอินเทอร์รัปท์มาสค์รีจิสเตอร์

แต่ละบิตใน IMR สามารถกระทำได้เช่นเดียวกับใน IFR คือเซตและเคลียร์ได้ แต่ IMR ไม่มีผลต่อการเกิดนอนมาสค์เอเบิลอินเทอร์รัปท์

7.3.3 การรีเซต

ขณะรีเซต TMS320c50 มีการปฏิบัติตามที่บรรยายดังนี้คือ เมื่อเกิดสัญญาณรีเซต(เป็นนอนมาสค์เอเบิลอินเทอร์รัปท์) ขณะกำลังทำงานทำให้ขาสัญญาณ RS เป็นสถานะต่ำ (low) ส่งผลทำให้ TMS320c50 ถูกสั่งให้จบการทำงานและโปรแกรมเคาน์เตอร์ถูกเคลียร์เป็นศูนย์ นอกจากนี้ยังมีรีจิสเตอร์และแฟล็คอื่นๆที่มีผลด้วย ขณะทำการทำงานที่สภาวะใดๆของโปรเซสเซอร์สัญญาณรีเซตสามารถเข้ามาที่คล็อกจังหวะใดๆ ทำให้สัญญาณถูกดึงลงต่ำ ชิพมีการเก็บและอ่านค่ารีเซตแล้วตำแหน่งอินเทอร์รัปท์เวคเตอร์เท่ากับ 0000H และจะมีการทำงานดังต่อไปนี้

0-> CNF (บิท 1 ใน ST1) , PC=0 ,ค่าของอินเทอร์รัปท์ทั้งหมดถูกเซตเป็น 0 , 0-> OV , 1-> XF, 0-> SXM , 0-> PM, 0-> OVM, 0->CENB2, 1-> HM, 0-> BRAF, 0->TRM, 1->NDX, 0->CENB1, 0->CENB2, 1->IPTR, 1->OVLY, 0->AVIS,. 1->RAM , 0->BIG, 1-> CNF, 1-> INTM, MP/MC(Pin)-> PMST(MP/MC), and 0-> C

ค่าของหน่ายความจำต่างๆ ใน หน่วยความจำข้อมูลจะถูกเคลียร์ไปยังตำแหน่งที่เดิม

รีจิสเตอร์ที่ใช้ในการนับรอบก็จะถูกเคลียร์เป็น 0 ด้วย

การรับรู้สัญญาณ (IACK) สัญญาณเกิดในลักษณะเดียวกันกับมาสค์เอเบิลอินเทอร์รัปท์

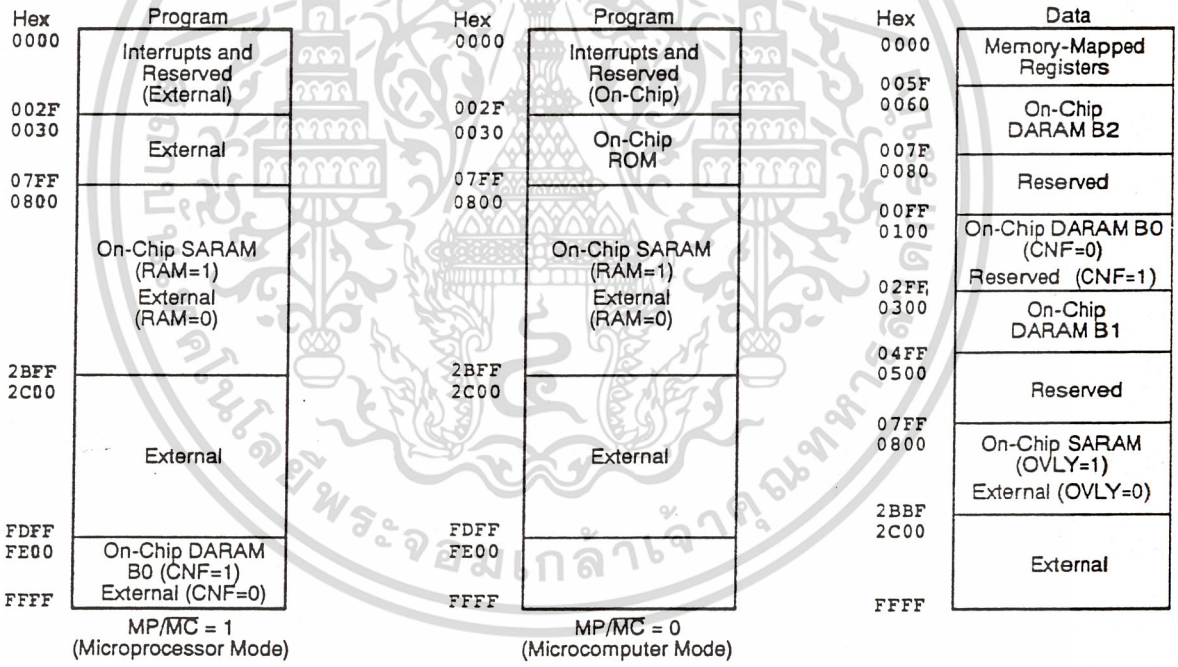
7.4 การจัดการหน่วยความจำภายในของการ์ด TMS320c50

การ์ดมีหน่วยความจำภายในทั้งหมด 9Kx16 บิท จากจำนวนดังกล่าวเป็นหน่วยความจำโปรแกรมและหน่วยความจำข้อมูล โดยสามารถกำหนดได้ว่าช่วงใดจะเป็นหน่วยความจำโปรแกรมหรือหน่วยความจำข้อมูลก็ได้

7.4.1 หน่วยความจำข้อมูล

9Kx16 บิต บนชิพถูกแบ่งออกเป็น 3 บล็อก (B0, B1 และB2) ดังรูป 7.6

บล็อกB0 512 เวิร์ด สามารถกำหนดให้เป็นหน่วยความจำโปรแกรมหรือหน่วยความจำข้อมูลได้โดยใช้คำสั่งกำหนดตามแต่ละจุดประสงค์ของการใช้งาน ส่วนบล็อกบB1 และ B2 640 เวิร์ด เป็นหน่วยความจำข้อมูลอย่างเดียว แบ่งเป็นการอ้างอิงแบบโดยตรง และแบบอ้างอิงแบบโดยอ้อม



รูป 7.6 แผนผังหน่วยความจำ

TMS320c50 สามารถอ้างอิงหน่วยความจำได้ทั้งหมด 64K ถ้าเราใช้หน่วยความจำข้อมูลบนชิพด้วยตำแหน่งของมันจะถูกจับคู่ลงในตำแหน่งที่ต่ำกว่า 1K เวิร์ดของ ช่องว่างหน่วยความจำข้อมูล (Data Memory Space) และหน่วยความจำข้อมูลสามารถขยายเพิ่มขึ้นโดยตรงจนมีขนาด 64 K เวิร์ด ขณะที่ยังคงทำงานด้วยความเร็วเต็มที่

7.4.2 หน่วยความจำโปรแกรม

โปรแกรมแรม,รอมบนชิพ เป็นหน่วยความจำภายในที่มีประสิทธิภาพสูงเพราะไม่มีสภาวะรอ (Wait State) ,มีราคาถูกกว่าหน่วยความจำภายนอก และใช้พลังงานน้อยกว่าหน่วยความจำภายนอก และหากมีหน่วยความจำภายนอกจะทำให้การติดต่อช้า เพราะต้องรอสัญญาณพร้อม (Ready)

7.4.3 แผนผังหน่วยความจำ

TMS320c50 มีตำแหน่งว่างของหน่วยความจำโปรแกรม หน่วยความจำข้อมูลอินพุทและเอาท์พุทแยกกัน

สังเกต เนื้อที่ตำแหน่งภายนอก (External) ใช้สำหรับต่อเพิ่มได้ทั้งหน่วยความจำโปรแกรมและหน่วยความจำข้อมูล

หน่วยความจำจะถูกอ้างอิงในตำแหน่ง 0800H - 2BBFH หน่วยความจำโปรแกรมถูกอ้างอิงในตำแหน่ง 0800H - 2BBFH แต่อาจถูกเปลี่ยนแปลงได้โดยใช้ซอฟแวร์ (การรีเซตเป็นการกำหนดให้ช่อง B0 เป็นหน่วยความจำข้อมูล)

7.4.4 รีจิสเตอร์ช่วย (Auxilary Register)

ภายใน TMS320C50 มีรีจิสเตอร์ช่วยอยู่ 8 ตัว(AR0-AR7) ใช้ในการอ้างอิงหน่วยความจำภายนอกหรือใช้ในการเก็บข้อมูลชั่วคราว โดยที่จะมีรีจิสเตอร์อีกตัวคอยชี้รีจิสเตอร์ช่วยอีกทีหนึ่ง (ARP)

7.5 หน่วยประมวลผลกลางทางตรรกศาสตร์ (Central Arithmetics Logic Unit)

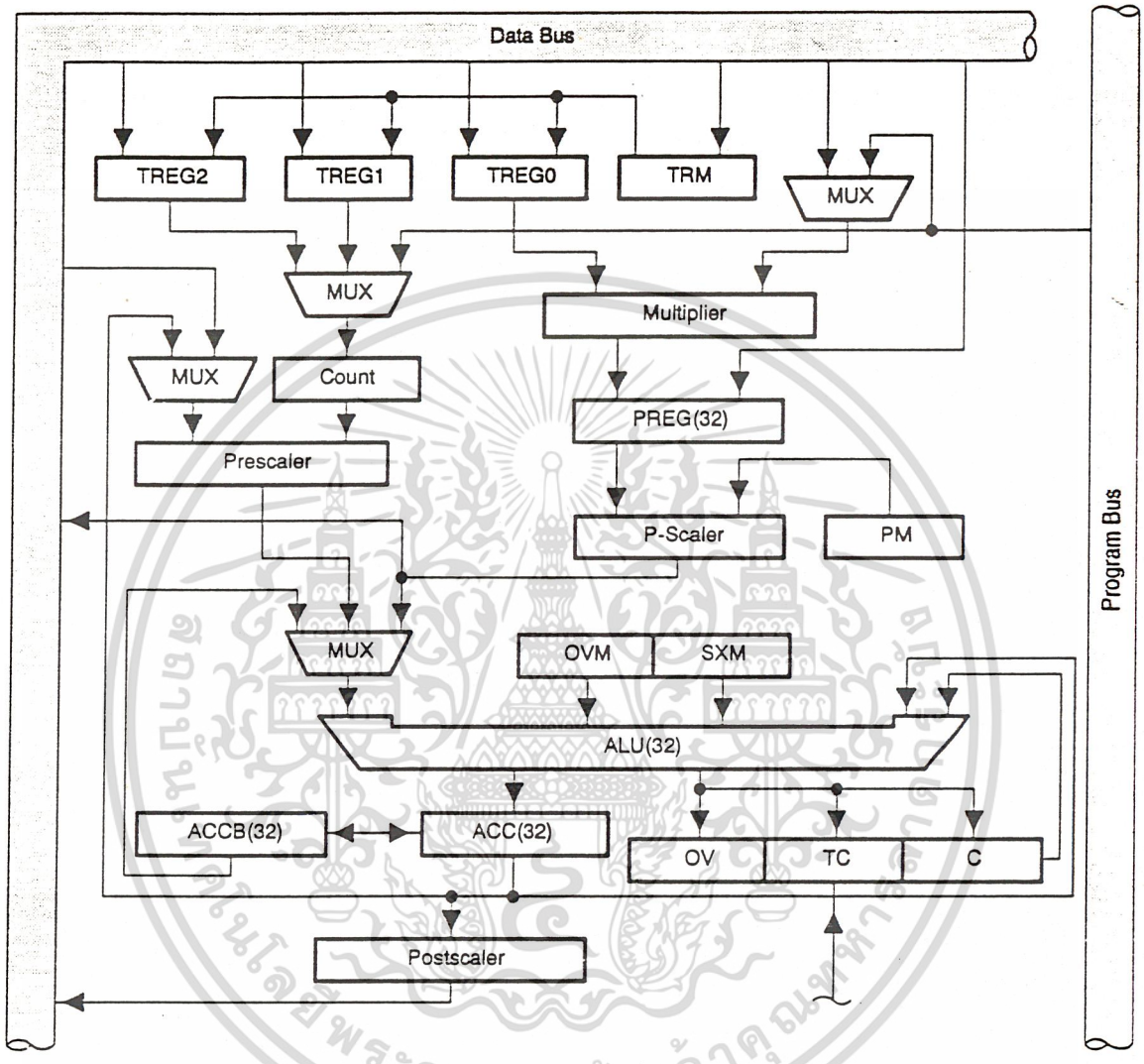
ประกอบด้วยอุปกรณ์ต่างๆตามรูป 7.7

- 16 บิต สเกลลิ่งชิฟเตอร์(Scaling Shifter)
- 32 บิต หน่วยประมวลผลทางตรรกศาสตร์(Arithmetics Logic Unit)
- 32 บิต แอคคิวมูเลเตอร์
- 16x16 บิต ตัวคูณ(Multiplier) และ 32 บิต รีจิสเตอร์เก็บผลคูณ(Product Register)

7.6 ระบบควบคุม(Control System)

7.6.1 การกำเนิดตำแหน่งโปรแกรมและการควบคุม

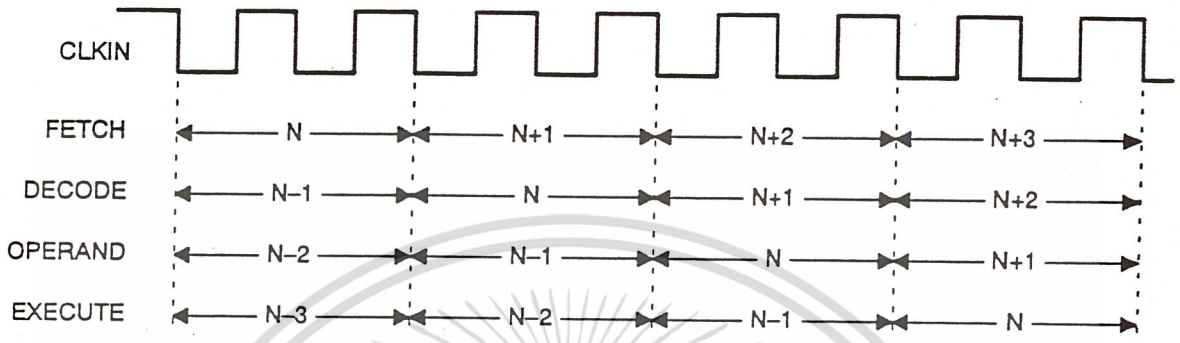
TMS320C50 มีสแตกลึก 8 ระดับ โดยที่ตำแหน่งหน่วยความจำอาจอยู่ภายในหรืออยู่ภายนอกก็ได้ และเราสามารถเซตได้โดยใช้ซอฟท์แวร์ รวมทั้งยังสามารถย้ายหน่วยความจำโปรแกรมไปใส่ไว้ในช่องว่างของหน่วยความจำข้อมูล(ในที่นี้ถ้าใช้คำสั่ง MAC,MACD,MADDและMADS) ยังคงไปอ่านตำแหน่งที่หน่วยความจำเดิมอยู่



รูป 7.7 แสดงบล็อกไดอะแกรมของส่วนประกอบของ CALU

7.6.2 การทำงานเป็นจังหวะ(Pipeline Operation)

คำสั่งที่อยู่ภายในชิพตัวนี้ประกอบด้วยการทำงานเป็นลำดับ ระหว่างปฏิบัติ(Execution)คำสั่ง การทำงานแบบเป็นจังหวะมีส่วนของขั้นตอนดังนี้ เฟท(Fetch),ดีโค้ด (Decode),อ่านข้อมูล (Fetch Operand),ปฏิบัติ(Execute) ซึ่งแต่ละขั้นตอนในช่วงการทำงานแยกกันทำงานอย่างอิสระ(ไม่จำเป็นต้องรอให้คำสั่งแรกทำงานเสร็จจึงจะทำคำสั่งที่สอง)



รูป 7.8 การทำงานแบบเป็นจังหวะ

บทที่ 8 การประยุกต์ใช้งาน (Adaptive Signal Processing Applicltion)

การประยุกต์ใช้งานของโครงงานนี้เราจะนำความรู้ทั้ง 2 ส่วน คือส่วนของทฤษฎีการประมวลผลสัญญาณแบบปรับตัวเองได้ และการใช้งานการ์ดประมวลผล TMS320c50

-ส่วนทฤษฎีการประมวลผลสัญญาณแบบปรับตัวเองนั้น เราจะเอาส่วนของรูปแบบการหักล้างสัญญาณรบกวน (Noise Cancellation) โดยที่มีการปรับค่าถ่วงน้ำหนักตามวิธีค่าเฉลี่ยกำลังสองน้อยที่สุด ซึ่งสมการปรับค่าถ่วงน้ำหนักจะเป็นไปตามสมการ

$$W_{k+1} = W_k + 2\mu\varepsilon_k X_k$$

-ส่วนการใช้งานการ์ดประมวลผล TMS320c50 เราจะนำความรู้จากการศึกษาฮาร์ดแวร์ของการ์ดและการใช้คำสั่งใช้งาน มาประยุกต์ใช้ในการเขียนโปรแกรมภาษาเครื่อง

ในการประยุกต์ใช้งานของทฤษฎีการประมวลผลสัญญาณแบบปรับตัวเองได้ เข้ากับการลดสัญญาณรบกวนนั้น เราจะใช้รูปแบบตามรูป 8.1 ไปประยุกต์ใช้ ซึ่งเป็นการลดสัญญาณรบกวน (Noise)ที่มีความถี่เดียว เช่น สัญญาณฮัม(Hum) 50 Hz ที่รวมมากับสัญญาณ(Signal) โดยการใช้สัญญาณที่มีการปรับตัวเองจนมีลักษณะตรงข้ามกับสัญญาณรบกวนไปหักล้าง จนทำให้สัญญาณรบกวนหมดไป

การทำงานตามรูป 8.1 นั้นจะเป็นการลดสัญญาณรบกวนที่รวมมากับสัญญาณ โดยเราจะใช้รูปแบบของสัญญาณรบกวนที่มารบกวนนั้นเป็นสัญญาณพื้นฐานในการปรับตัวเองเพื่อนำไปหักล้างกับสัญญาณที่มีสัญญาณรบกวนนี้ปนอยู่

การทำงานของระบบนี้เริ่มจากการสุ่มสัญญาณที่มีสัญญาณรบกวนปนอยู่ และสัญญาณรบกวนที่จะใช้เป็นสัญญาณพื้นฐาน เราจะนำสัญญาณรบกวนนี้ไปผ่านกระบวนการปรับตัวเองซึ่งเราจะนำไปประมวลผลต่อในส่วนประมวลผลซึ่งเราจะการ์ด TMS320c50 ทำตัวเป็นส่วนประมวลผล โดยเกิดจากการเขียนโปรแกรมเป็นภาษาเครื่องเพื่อให้การ์ดสามารถประมวลผลตามรูปแบบที่เราต้องการได้

การทำงานของส่วนประมวลผลนั้น จะมีสภาวะเริ่มต้น คือ ค่าถ่วงน้ำหนัก ทั้ง w_{0k} และ w_{1k} จะเป็นศูนย์ และสุ่มตัวอย่างจากสัญญาณรบกวนเข้ามา 2 แบบ คือสัญญาณที่สุ่มมา (x_{0k}) และสัญญาณสุ่มที่เลื่อนสัญญาณสุ่มออกไป 90 องศา (x_{1k}) ซึ่งเราจะได้ y_k ออกมา

จาก

$$y_k = x_{0k}w_{0k} + x_{1k}w_{1k}$$

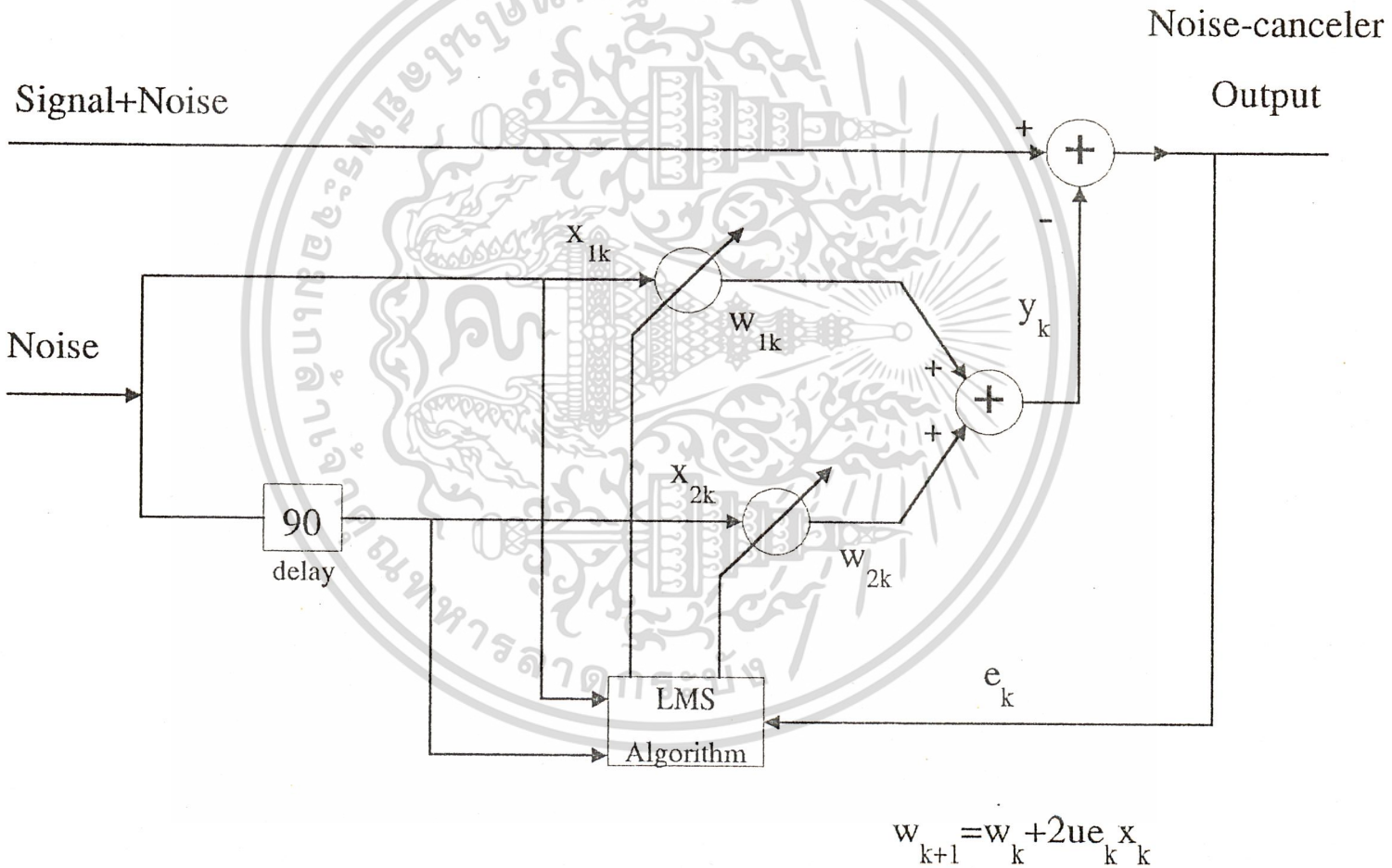
y_k ที่ได้จะส่งไปหักล้างกับสัญญาณที่รวมกับสัญญาณรบกวน ผลต่างที่ได้เราจะพิจารณาเป็นค่าความผิดพลาด ε_k ที่เราจะนำไปปรับค่าถ่วงน้ำหนัก ตามสมการของค่าเฉลี่ยกำลังสองน้อยที่สุด

$$W_{k+1} = W_k + 2\mu\varepsilon_k X_k$$

นิยามว่า $$W_k = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$$

$$X_k = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$$

ในการที่เรารับค่าความผิดพลาดไปปรับค่าถ่วงน้ำหนักแล้วนั้น ก็เป็นอันครบ 1 รอบของการสุ่มสัญญาณเข้ามา ซึ่งเราจะต้องสุ่มสัญญาณต่อไปเข้ามาเป็น x_{0k} และ x_{1k} ใหม่ แล้วก็จะมีการทำตามขั้นตอนเหล่านี้วนเวียนไปเรื่อย ๆ จนในที่สุดค่าความผิดพลาดที่ออกมาจะมีลักษณะใกล้เคียงกับสัญญาณจริง นั่นก็คือ เราได้สร้างสัญญาณที่มีลักษณะตรงข้ามกับ ออกไปหักล้างได้สำเร็จ ซึ่งทำให้ผลที่ได้ ก็คือ สัญญาณจริงที่ถูกกำจัดสัญญาณรบกวนออกไป ซึ่งเป็นไปตามความต้องการ



รูป 8.1 รูปที่ใช้การทดลอง

บทที่ 9 ผลการทดลองและสรุปผลการทดลอง

วิธีทดลอง

การเขียนโปรแกรมลงในการ์ด TMS320c50 ทำโดยการใช้โปรแกรมอิดิตเตอร์ทั่วไปแล้วใช้โปรแกรม DSK54D.EXE แปลเป็นภาษาเครื่อง แล้วโหลดลงการ์ด ส่วนอัลกอร์ลิทึ่มของโปรแกรมเป็นไปตามดังรูปกราฟไหลที่ได้ออกแบบไว้แล้ว

ผลการทดลอง

ในการทดลองครั้งแรกๆ ผลปรากฏว่าไม่มีสัญญาณออกมา เนื่องมาจากลำดับของการประมวลสัญญาณและการกำหนดช่วงเวลาในการประมวลผลยังไม่ถูกต้อง(ใช้คำสั่งโปรแกรมที่ถูกต้องแต่คำสั่งที่ใช้นั้นไม่อยู่ในช่วงเวลาที่เหมาะสม) จึงทำให้ผลที่ออกมาคือไม่มีสัญญาณออกมาตามที่ต้องการ

แต่เมื่อใช้คำสั่งถูกลำดับเวลาแล้ว สัญญาณที่ออกมาก็ยังไม่ถูกต้อง เพราะการประมวลสัญญาณจำเป็นที่จะต้องมี การเช็คเครื่องหมาย ขณะที่มีการบวกลบเสมอ ส่งผลให้ สัญญาณที่ออกมา มีค่า แรงดันที่ขลิบและวนกลับจากด้านที่เกินจากด้านบวก เพราะฉะนั้นจึงจำเป็นที่จะต้องเช็คเครื่องหมายอยู่ตลอดเวลาที่ประมวล

ต่อจากนั้นสัญญาณที่ออกมา ก็มีเครื่องหมายถูกต้องแต่ก็ยังมีข้อผิดพลาดอยู่เพราะ ในบางครั้งถึงแม้จะมีการเช็คเครื่องหมายแต่ การบวกกันของสัญญาณ ก็สามารถทำให้เกิดการล้น ทำให้ผลบอกของ จำนวนบวก 2 จำนวน มีค่าเป็นค่าลบได้ จึงต้องมีการเช็คว่าเกินค่ามากที่สุดหรือไม่

หลังจากที่ได้แก้ไขปัญหาดังที่ได้กล่าวมาก็ ได้เอาท์พุทที่ถูกต้องตามต้องการ แต่ผลของอัลกอริทึ่ม ที่ใช้ประมวลสัญญาณก็ ยังควรต้องมีการปรับปรุงต่อไป เพราะการประมวลผลในการทดลองนี้เป็นแบบเวลาจริง(Real time processing) เพราะฉะนั้นการประหยัดเวลาในการประมวลสัญญาณจึงจำเป็นมากในการทำงาน และจำทำให้เราสามารถเพิ่มความถี่สุ่มได้ สูงขึ้นและเข้าถึงสัญญาณจริงได้ใกล้เคียงขึ้น

ผลการทดลองที่ได้จากการลดสัญญาณรบกวนที่รวมมากับสัญญาณตามรูปแบบระบบที่ 8.1 นั้นได้ผลดังรูป 9.4 ซึ่งจะแสดงให้เห็นถึงลักษณะการเปลี่ยนแปลงสัญญาณในการลดสัญญาณรบกวน จะเห็นได้ว่าสัญญาณที่มีสัญญาณรบกวนรวมด้วยนั้นจะค่อยๆปรับตัวเองจนมีลักษณะเข้าใกล้สัญญาณที่แท้จริง ทำให้สามารถลดสัญญาณรบกวนได้

สรุปผลการทดลอง

การทำงานของระบบยังไม่มีเสถียรภาพที่ดีเท่าที่ควร เพราะสถานะของระบบยังต้องพึ่งพาผลตอบสนองของวงจรให้ทันเวลา และการประมวลผลต้องการวิธีที่ทำงานที่รวบรัดมากขึ้น โดย

การพัฒนาต่อไปนั้นทำได้โดยการเพิ่มตัววัดผลของสัญญาณ (sensor) ให้มากข้อน ปรับปรุงวิธีการประมวลใหม่ให้สามารถรองรับการทำงานที่ความถี่สูงขึ้นได้

การทำงานของโปรแกรมคำสั่งภาษาแอสเซมบลี้

(.set) ส่วนต้นของโปรแกรมเป็นการ define ค่าตัวแปรต่าง ๆ ในโปรแกรม

(.mmregs) เป็นการให้ Assembler เข้าใจถึงตัวแปรให้ตรงกับ memory map register

(.ps) program start

-(.ps) ส่วนแรกเป็น Interrupt vector

-(.ps) ส่วนที่สองเป็น save run

Routine 1 Initial Card เพื่อ set wait state ของ Ram ภายในและภายนอก (ถ้ามี) ที่ Blank ต่าง ๆ ว่าจะให้มี wait state อย่างไรบ้าง ใน card นี้มีแต่ RAM ภายใน wait state = 0

Routine 2 ใส่ค่าของลำดับ $x_n \rightarrow x_0$ ให้มีค่าเป็น 0 ก่อนเริ่ม

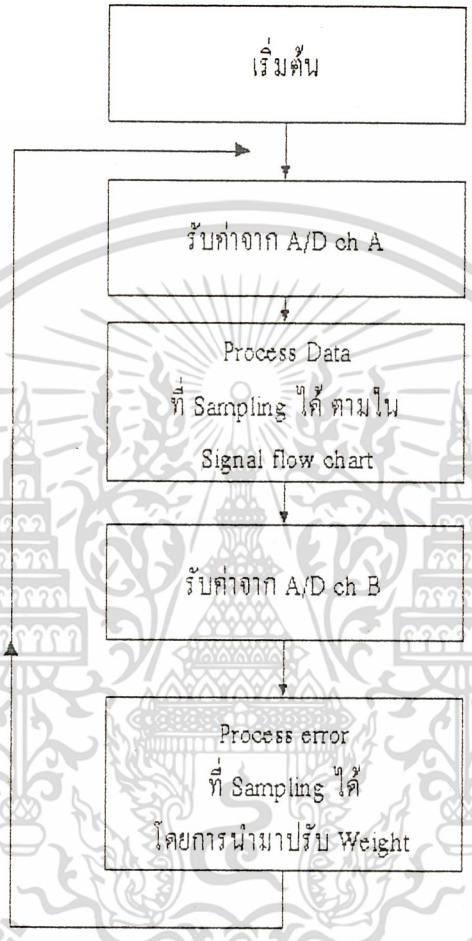
Routine 2 Set Main input

Routine 3 Set Auxin input

Routine 4 Process Data โดยที่คำสั่งหลักใน Routine นี้คือ macd (multiply accumulator & data move) จะมีการนำค่าใน pma ชี้กับ dma ชี้มาคูณกันตาม label และหลังจากคูณกันแล้วก็จะมีการ Shift ค่าใน dma

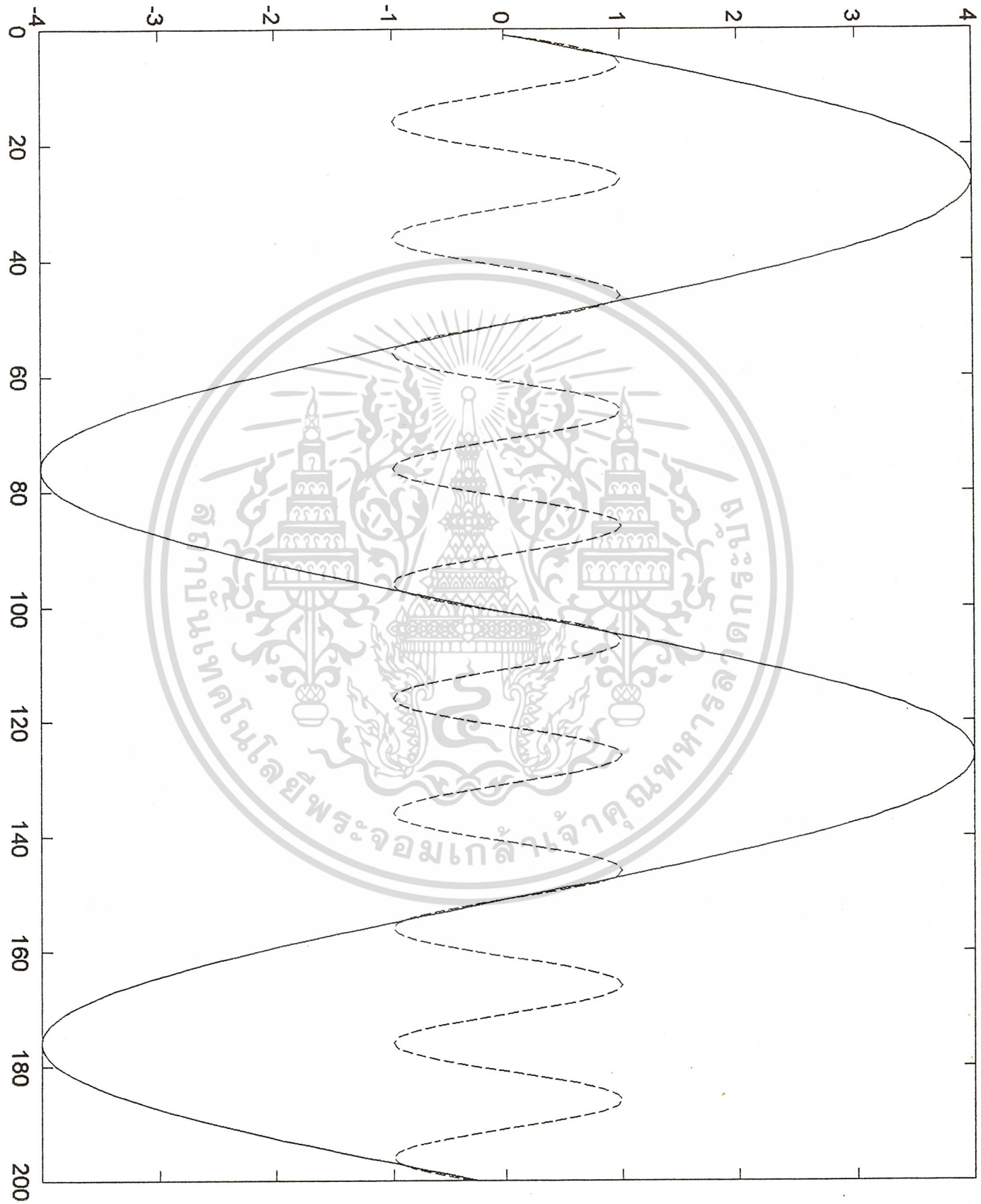
Routine 5 หลังจากที่ Sampling error ได้ จะมีการนำค่ามาคูณกับ x และบวกกับ w เดิมตามสมการที่ 8.1 แต่การทำงานของ ACC ไม่มีการเช็คเครื่องหมายหลังการบวก ทำให้ต้องมีการเช็คเครื่องหมายแล้วไปทำงานที่ตำแหน่งต่าง ๆ

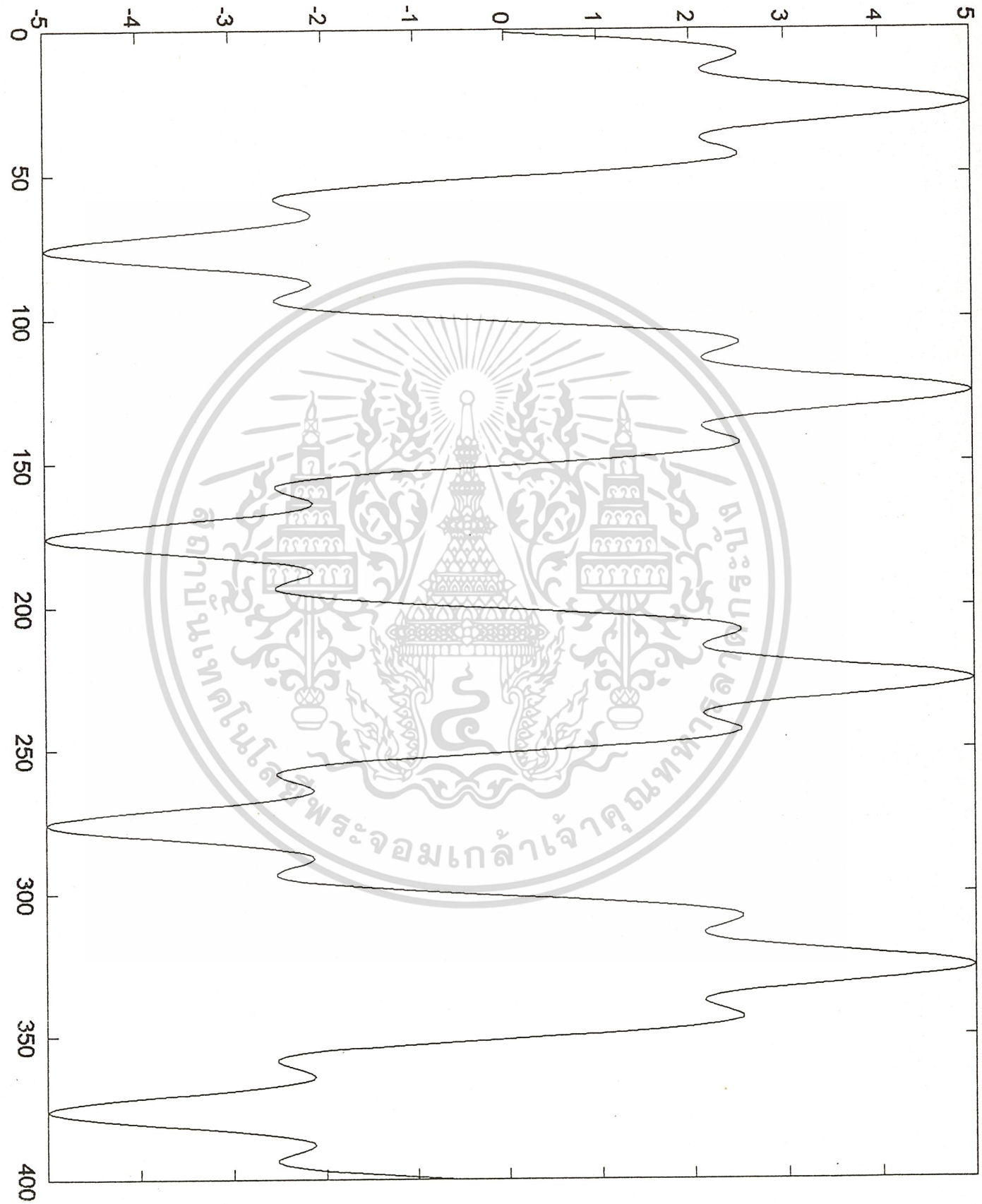
แผนภูมิไหล

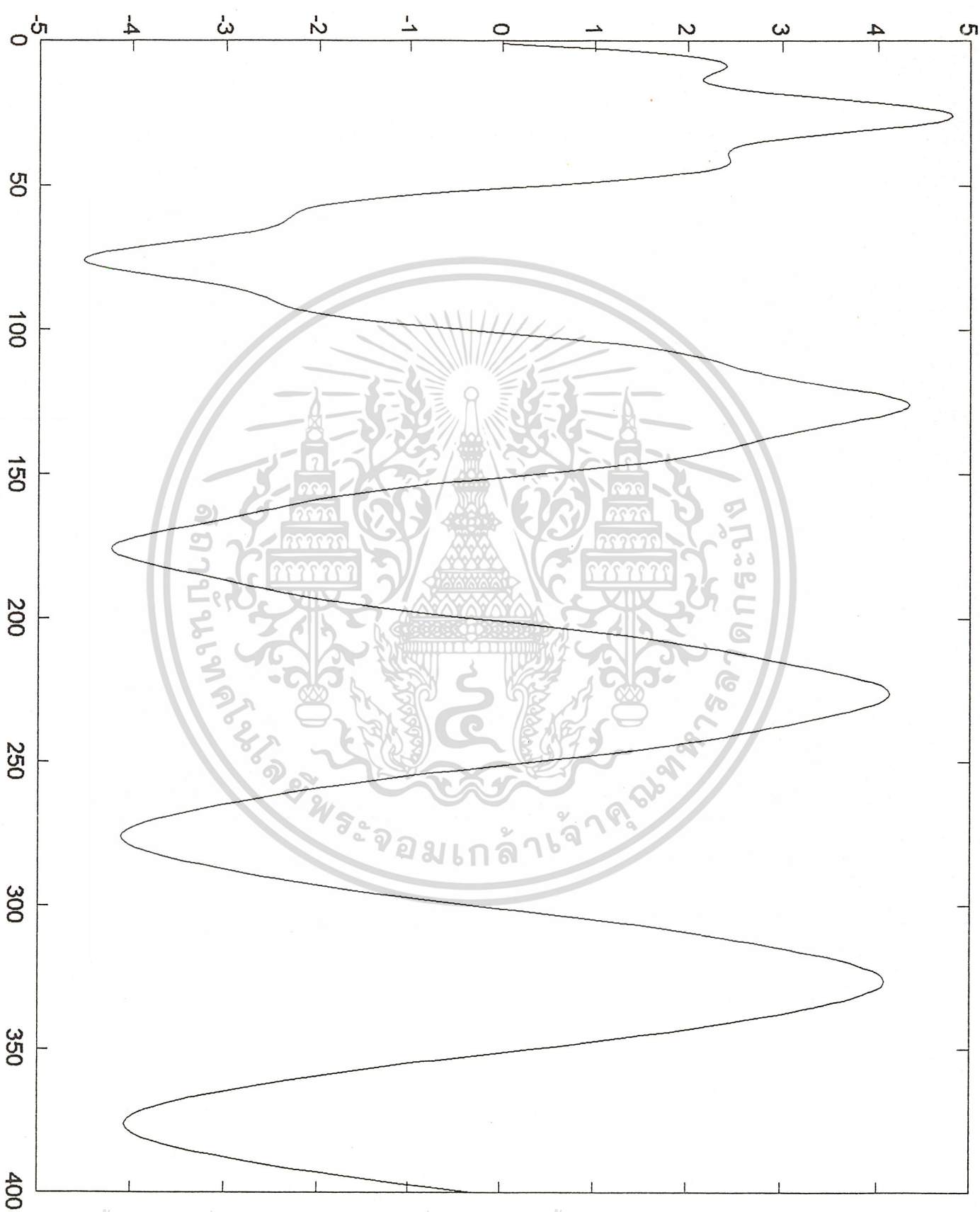


A/D ch A คือ สัญญาณ Signal + Noise

A/D ch B คือ สัญญาณ Signal + Noise







```
*************************************************************************
*       This is the source code for the DSK digital lowpass filter      *
*        Assemble using the DSK5A assembler and execute the DSK         *
*          loader to load and run the application on your PC.           *
*************************************************************************
*       Init The value of the valiable in this program                  *
              .mmregs
TA            .set   31          ;      Auxin -----+  +----- Loopback
RA            .set   31          ;       Synch --+  |  |  +-- BP Filter
TAp           .set   1           ;                  |  |  |  |
RAp           .set   1           ;     +------------+-----------+
TB            .set   18          ;     |00 00 G1 G0 | SY AX LB BP|
RB            .set   18          ;     +------------+-----------+
AIC_CMD       .set   00h         ;              GAIN
yn            .set   107
xn            .set   108

              .ps    080ah
              B      RINT        ;0A; Serial port receive interrupt RINT
              B      XINT        ;0C; Serial port transmit interrupt XINT

*
* TMS32C05X INITIALIZATION
* This routine initializes the C5x registers, internal RAM and
* external RAM from xxxx to FFFF
*
              .ps    0a00h
              .entry
start         ldp    #0          ; Set data page pointer
              splk   #830h,PMST  ; 9K on-chip RAM as Data, No ROM
              lacl   #0          ; Set Wait State Control Regester
              samm   CWSR        ; for 0 waits in pgm & data memory
```

```
                samm    PDWSR           ;

*
* initalize and reset serial port
*
errdp           .set    106
yn              .set    107
xn              .set    108

first_xo        .set    04ech
last_xn         .set    04f5h
;******  OK   ***************
first_wn        .set    0e00h
last_wo         .set    0e01h

                .ps     080ah
                B       RINT            ;0A; Serial port receive interrupt RINT
                B       XINT            ;0C; Serial port transmit interrupt XINT

                .ps     0a00h
                .entry
                ldp #0

                call init               ; the routine initial card TMS 320c50
                call put                ; the routine put initial data for start up
loop2:          call sam_in             ; the routine set A/D sampling signal
                call filter             ; the routine filter and calculate data output
                call aux_in             ; the routine set A/D sampling error signal
                call adaptive           ; the routine adapt all weight in filtering section
                b   loop2

;******* Routine intial Card TMS320C50 ****
```

```
init:       ldp    #0              ; Set data page pointer
            splk   #830h,PMST      ; 9K on-chip RAM as Data, No ROM

            lacl   #0              ; Set Wait State Control Regester
            samm   CWSR            ; for 0 waits in pgm & data memory
            samm   PDWSR           ;

            splk   #20h,TCR
            splk   #1,PRD
            lacl   #08h            ; set FSM bit for FSX/FSR per frame
            samm   spc             ; Configure for 16 bit mode with
            lacl   #0C8h           ; external CLKX, reset tx and rx
            samm   spc
            lamm   DRR             ; clear first int
            lacc   #0080h
            sach   DXR             ; clear first int
            sacl   GREG            ; Pulse AIC reset by setting it low

            lar    AR0,#0FFFFFh
            rpt    #10000          ; and taking it high after 1000 cycles
            lacc   *,0,AR0         ; (.5ms at 50ns)
            sach   GREG
            setc   SXM
            setc   OVM
            call   AIC_SET         ; DO NOT CHANGE DP WITHOUT RESTORING IT!
            lacl   #12h            ; RINT
            samm   IMR             ;
            ret
;************ end _init


;******* start put data  in Xo-Xn to 0001h
```

```
put:    LAR     AR7,#first_xo
        ldp     #0
        lacc    #star,0,ar7
        RPT  #9
             SACL    *+,0,ar7
        ret

;*******  Set A/D  sampling Main Analog port
sam_in:      lacl #20h
             samm IMR
             lacc #sam_cmd,2
             add #03h
             call AIC_2nd
             ret

;*******  Set A/D  sampling Auxi Analog port
aux_in:      lacl #20h
             samm IMR
             lacc #aux_cmd,2
             add #03h
             call AIC_2nd
             ret

;*******  Sigma-Multiple-Inverse Xo-Xn with Wo-Wn
filter:  ;   ldp #9
             mar *,ar0
             ldp #0
        splk #06fffh,brcr
        rptb end_1
             ldp #9
             lamm    DRR
             sacl    xn
```

```
                zpr
                lar     AR0, #last_xn
                rpt     #9
                macd    coeff, *-
                apac
                sach    yn, 1
                lacc    yn, 3
               neg          ; inverse data
                ldp #0
end_1:                samm    DXR
              ret

;*******  Change a wage by using (Beta.Err.X + W_old)
adaptive: lamm DRR
          mar  *,ar3
          lar  ar3,#first_xo        ; ar3 point to  Xo
          lar  ar2,#last_wo
          ldp  #9
          sacl errdp
          lt   errdp
          LDP #0
          splk #n,BRCR
          rptb end_block2
          mpy  *+,ar2     ; now Ar point to Wo
          pac
          bcnd pl,geq

ne:  setc SXM
     rpt  #17
     sfr
     bit *,0        ; check  W is +/-
     bcnd nepl,ntc
```

```
nene:     add *
          bcnd end_add,lt
          lacc #8000h,15
          rol
          b end_add

nepl:     add *
          b end_add

pl:  clrc SXM
     rpt #17
     sfr
     bit *,0        ; check  W is +/-
     bcnd plpl,ntc

plne:     add *
          b end_add

plpl:     add *
          bcnd end_add,geq
          lacc #7fffh,15
          rol
          b end_add

end_add:  sacl *-,0,ar3
end_block2:    NOP

               ret

AIC_SET:       call sam_in
```

```
                lacc   #TB,9
                add    #RB,2
                add    #02h
                call   AIC_2nd
                ;-----------------------
                lacc   #TA,9
                add    #RA,2
                call   AIC_2nd
                ;-----------------------
                ret
AIC_2nd:        sach   DXR
                clrc   INTM
                idle
                add    #6,15
                sach   DXR
                idle                ;ACCU_hi requests 2nd XMIT
                samm   DXR
                idle                ;ACCU_lo sets up registers
                lacl   #0
                samm   DXR          ;make sure the word got sent
                idle
                setc   INTM
                ret
*
* Serial port interrupt
*

XINT:   rete
RINT:   rete
                .data
coeff           .int    0180h, 0180h, 0180h, 0180h, 0180h
                .int    0180h, 0180h, 0180h, 0180h, 0180h
```

กิตติกรรมประกาศ

สำหรับโครงงานในครั้งนี้สามารถลุล่วงไปได้ด้วยดี ก็ด้วยความกรุณาของบุคคลหลายท่าน ซึ่งผู้จัดทำขอขอบคุณไว้ ณ โอกาสนี้ ได้แก่ อ.ชินภัทร นันทจิวากรชัย อาจารย์ทุกๆท่าน รุ่นพี่ และเพื่อนๆทุกคน ที่ได้ให้ความช่วยเหลือ

หนังสืออ้างอิง

1.Bernard Widrow,"Adaptive Signal Processing",Prentice-Hall,Inc.474 p.,1985

2.Texas Instrument,"TMS320C5x User's Guide Digital Signal Processing Product",Texas Instrument,763 p. , 1993.

3.Texas Instrument,"TMS320C5x User's Guide DSP Starter Kit",Texas Instrument,253 p. , 1994