การควบคุมปรับตัวได้ในวิธีของ MRAC
Model Reference Adaptive Control

โดย

นาย จิรศักดิ์ ชัยวิริยะกุล

นาย สมพร ชาญประจักษ์วณิช

นาย เชกสรร เกียรติสุไพบูลย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
สาขาวิศวกรรมระบบควบคุม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2535

ปริญญานิพนธ์ ปีการศึกษา 2535
ภาควิชา วิศวกรรมระบบควบคุม
คณะ วิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง การควบคุมปรับตัวได้ในวิธีของ MRAC
ผู้จัดทำ

1. นาย จิรศักดิ์ ชัยวิริยะกุล 32.1059
2. นาย สมพร ชาญประจักษ์วณิช 32.1350
3. นาย เสกสรร เกียรติสุไพบูลย์ 32.1403

ผศ.ดร. จงกล งามวิวิทย์ อาจารย์ที่ปรึกษา
(.....)

รศ. วิพันธ์ ปรีชาพานิช อาจารย์ที่ปรึกษา
(.....)

รศ. สุเธียร เกียรติสุนทร อาจารย์ที่ปรึกษา
(.....)

อ. ปรเมษฐ์ ประนยานันทน์ อาจารย์ที่ปรึกษา
(.....)

การควบคุมปรับตัวได้ในวิธีของ MRAC

1. นาย จิรศักดิ์ ชัยวิริยะกุล 32.1059
2. นาย สมพร ชาญประจักษ์วณิช 32.1350
3. นาย เสกสรร เกียรติสุไพบูลย์ 32.1403

อาจารย์ที่ปรึกษา
ผศ.ดร. จงกล งามวิวิทย์
รศ. วิพันธ์ ปรีชาพานิช
รศ. สุเธียร เกียรติสุนทร
อ. ปรเมษฐ์ ประนยานันทน์

บทคัดย่อ

โครงงานนี้เริ่มต้นด้วยการศึกษาทฤษฎี และ ระเบียบวิธีการควบคุมระบบด้วยวิธี MRAC (Model Reference Adaptive Control) จากนั้นนำวิธีดังกล่าวมาทดลองโดยสร้างโปรแกรมจำลองระบบควบคุมด้วยคอมพิวเตอร์ การจำลองระบบควบคุมด้วยคอมพิวเตอร์เป็นการศึกษาทฤษฎีการควบคุมเมื่อระบบควบคุมทั้งหมดอยู่ภายใต้เงื่อนไขในทฤษฎีโดยไม่มีผลกระทบภายนอกมารบกวนระบบ หลังจากจำลองระบบควบคุมด้วยคอมพิวเตอร์แล้ว ได้นำวิธีดังกล่าวมาทดลองกับกระบวนการจริงที่มีผลกระทบภายนอก และขีดจำกัดบางอย่างมารบกวนระบบ จุดประสงค์เพื่อศึกษาความสามารถของตัวควบคุม MRAC เมื่ออยู่นอกเหนือเงื่อนไขในทฤษฎี ผลที่ได้รับแสดงให้เห็นว่าไม่สามารถควบคุมกระบวนการได้ตามทฤษฎี จึงได้มีการค้นหาสาเหตุและหาแนวทางแก้ไข การค้นหาสาเหตุและการหาแนวทางแก้ไขใช้โปรแกรมจำลองระบบควบคุมด้วยคอมพิวเตอร์ที่ได้สร้างขึ้นในการทดลองขั้นแรกมาเป็นเครื่องมือ นำแนวทางแก้ไขที่ค้นพบมาปรับปรุงตัวควบคุม ผลของการควบคุมที่ปรับปรุงแล้วแสดงให้เห็นว่าแนวทางดังกล่าวสามารถแก้ไขปัญหาที่เกิดขึ้นได้เป็นผลสำเร็จ และได้ผลการควบคุมเป็นไปตามทฤษฎี

กระบวนการจริงที่ใช้ในการโครงงานนี้ เป็นเครื่องจำลองระบบ PTS-10 ซึ่งเป็นวงจรอิเลคทรอ - นิคส์ ที่ใช้จำลองระบบเชิงเส้นซึ่งสามารถปรับพารามิเตอร์ของกระบวนการได้

Model Reference Adaptive Control

By
Mr. Jirasak Chaiviriyakul
Mr. Somporn Chanprajakwanich
Mr. Seksan Kiatsupaibul

Advisor :
Asst. Prof. Dr Jongkol Ngamwiwit
Associate Prof. Vipan Preechapanid
Associate Prof. Suthian Kiatsunthorn
Mr. Poramate Pranayanuntana

Abstract

The project is started by studying the theory and the control algorithm of MRAC (Model Reference Adaptive Control). Then simulate the control system with computer. The computer simulation experiment is done in order to study the control theory when the whole system is under the conditions based on the theory and without any effects caused by outer surrounding. Next, perform the experiment again by control process simulator PTS-10, a linear electronic process which can change some parameters of the process. The aim of This stage is to check out the ability of the MRAC control system when it is used in the real situation. The result shows that the controller controls the process in the manner different from the theory. By using the computer simultion program, we can find the deviation of process simulator from an ideal one and find the way to reduce the effects of it. Finally, improve the controller along that way. The improved controller can control the process in the same manner as one stated in the theory.

กิตติกรรมประกาศ

ขอขอบคุณ อาจารย์ จงกล งามวิวิทย์ และอาจารย์ที่ปรึกษาทุกท่านที่ให้คำแนะนำในการทำโครงงานครั้งนี้

ขอขอบคุณ อาจารย์ วิชา สำนักวิจัยและบริการคอมพิวเตอร์ ครูสอนไมโครโปรเซสเซอร์

ขอขอบคุณ องค์การความร่วมมือระหว่างประเทศของญี่ปุ่น (Japan International Cooperation Agency หรือ JICA) ที่ให้ความช่วยเหลือเครื่องมือประกอบการทดลอง process simulator PTS-10 และ เครื่องวัดสัญญาณ Servocorder SR 6221

สารบัญ

สารบัญรูป

หน้า

สัญลักษณ์

$\mathbf{x}_p$: สเตทเวคเตอร์ของกระบวนการ
y_p : เอาท์พุทของกระบวนการ
$\mathbf{x}_m$: สเตทของรูปแบบอ้างอิง
y_m : เอาท์พุทของรูปแบบอ้างอิง
$\mathbf{e}$: process-model state error
u : อินพุทของกระบวนการ
r : สัญญาณอ้างอิง (reference signal) หรือ set point
$\boldsymbol{\omega}$: signal vector
$\boldsymbol{\theta}$: พารามิเตอร์ของตัวควบคุมปฐมภูมิ
$\boldsymbol{\theta}^*$: พารามิเตอร์ของตัวควบคุมปฐมภูมิที่จะทำให้ระบบลูปปิดมีผลตอบสนองตามรูปแบบอ้างอิง
$\boldsymbol{\phi}$: parameter error vector $\boldsymbol{\theta} - \boldsymbol{\theta}^*$
$\boldsymbol{\Gamma}$: adaptation gain matrix
γ : ฟังก์ชันลีอาปูนอฟ
γ_0 : γ-modification factor

บทที่ 1
บทนำ

วิทยาการทางวิศวกรรมสาขาหนึ่งที่เรียกว่า วิศวกรรมระบบควบคุม เป็นสาขาวิชาทางวิศวกรรมที่แยกศึกษาเน้นหนักในคุณภาพการปฏิบัติงานของระบบ วิทยาการในสาขานี้เกิดขึ้นเพื่อพยายามที่จะกำหนดแนวทาง หรือวิธีการที่จะทำให้ระบบที่เรากำลังสนใจ ทำงานให้กับเราได้ตรงตามความต้องการมากที่สุด มิใช่เพียงทำงานได้เท่านั้น

ในการกำหนดระเบียบวิธีการควบคุมระบบ สิ่งจำเป็นอันดับหนึ่งคือเราจะต้องเข้าใจระบบอย่างลึกซึ้ง ความเข้าใจในระบบนี้เป็นสิ่งที่จะเปิดทางไปสู่การควบคุมในอันดับต่อไป โดยทั่วไปแล้วเราจะใช้คณิตศาสตร์เป็น เครื่องมือที่ใช้อธิบาย วิเคราะห์ระบบ และสังเคราะห์ตัวควบคุมระบบ ดังนั้นความเข้าใจระบบก็อาจจะหมายถึงเราสามารถแทนระบบนั้น ๆ ด้วยสมการทางคณิตศาสตร์

ระบบในความเป็นจริงตามธรรมชาติมีความซับซ้อนเหลือคณานับ ดังนั้นการที่จะสามารถเข้าใจถึง รายละเอียดทั้งหมดของระบบจึงเป็นเรื่องยาก คณิตศาสตร์ที่เข้ามามีบทบาทในการประมาณระบบในความเป็นจริงที่ซับซ้อนเพื่อให้ง่ายต่อการวิเคราะห์ และมีประสิทธิภาพสูง ได้แก่ สมการอนุพันธ์เชิงเส้นไม่แปรค่าตามเวลา (linear time-invariant differential equation) ซึ่งเป็นการประมาณระบบในเชิงพลศาสตร์ โดยสมมติว่า ระบบไม่มีการเปลี่ยนแปลงตามเวลา หรือมีคุณสมบัติที่คงที่แน่นอน

ในการสังเคราะห์ตัวควบคุมสำหรับระบบดังกล่าวก็จะใช้คุณสมบัติของสมการอนุพันธ์เชิงเส้นไม่แปรค่าไปตามเวลา โดยสังเคราะห์จากวิธีเส้นทางเดินราก (root locus) หรือจากแผนภาพโบด (Bode diagram) สำหรับระเบียบวิธีดั้งเดิม หรือ ใช้วิธีการป้อนกลับสเตท (state feedback) สำหรับระเบียบวิธีแนวใหม่ ดู[1]

การวิเคราะห์ระบบ และสังเคราะห์ตัวควบคุมด้วยวิธีประมาณดังกล่าวนั้นเป็นวิธีที่ยอมรับกันและให้ผลเป็นที่น่าพอใจในระดับหนึ่ง แต่จุดบกพร่องของวิธีดังกล่าวได้แก่

1) ถ้าระบบที่เราจะควบคุมเป็นระบบที่เราไม่รู้จักมาก่อน หมายความว่าเรามิได้เป็นผู้สร้างขึ้นด้วยตัวเอง เช่น ระบบในโรงงานอุตสาหกรรมที่ไม่มีตัวควบคุมมาก่อนแต่จะปรับปรุงโดยเพิ่มตัวควบคุมเข้าไป นั่นหมายความว่า เราจะไม่ทราบรูปแบบทางคณิตศาสตร์ของระบบและไม่สามารถสังเคราะห์ตัวควบคุมด้วยวิธีดังกล่าว

2) ระบบในความเป็นจริงจะเสื่อมสภาพไปตามเวลา หรืออาจมีคุณสมบัติเปลี่ยนแปลงไปตามจุดทำงาน ไม่ใช่ว่าจะมีคุณสมบัติคงที่แน่นอนไม่เปลี่ยนแปลง

3) ระบบในความเป็นจริงมีคุณสมบัติที่ไม่เป็นเชิงเส้น

ในปัญหาข้อที่ 1 นั้น ตัวควบคุมที่นิยมนำมาใช้แก้ปัญหาก็คือ PID controller ซึ่งไม่จำเป็นต้องทราบคุณสมบัติของระบบก็สามารถใช้ควบคุมได้โดยปรับพารามิเตอร์ไปตามผลตอบสนองที่ได้รับ แต่อย่างไรก็ตาม PID ก็ไม่สามารถแก้ไขปัญหาในข้อที่สองและข้อที่สามได้

การควบคุมที่ปรับตัวได้ (Adaptive control) เป็นทางหนึ่งที่ใช้แก้ปัญหาทั้งสามข้อดังกล่าว ในวิทยานิพนธ์ฉบับนี้เป็นการศึกษาและทดลองการควบคุมที่ปรับตัวได้วิธีหนึ่ง ที่ชื่อว่า MRAC (Model Reference Adaptive Control) ซึ่งเป็นการควบคุมที่ปรับตัวได้อย่างง่าย และจำกัดขอบเขตการศึกษาเพียงใช้ควบคุมระบบเชิงเส้นเท่านั้น หรือเป็นการศึกษาสำหรับปัญหาในข้อ 1 และ 2

การทำงานครั้งนี้เริ่มต้นด้วยการศึกษาทฤษฎีและระเบียบวิธีการควบคุม MRAC จากนั้นนำวิธีที่ได้ศึกษามาทดลองโดยการจำลองระบบและการควบคุมด้วยคอมพิวเตอร์ การจำลองเป็นการเปรียบเทียบวิธีป้อนกลับสเตทกับวิธี MRAC การทดลองโดยการจำลองจากคอมพิวเตอร์เป็นการศึกษาทฤษฎีการควบคุมเมื่อระบบทั้งหมดอยู่ภายใต้เงื่อนไขในทฤษฎี โดยไม่มีผลกกระทบภายนอกมารบกวน สุดท้ายนำวิธีดังกล่าวมาทดลองกับระบบจริงที่มีผลภายนอก และขีดจำกัดต่าง ๆ มารบกวนระบบทั้งหมด เพื่อศึกษาความสามารถของตัวควบคุม MRAC เมื่ออยู่นอกเหนือจากเงื่อนไขในทฤษฎี

ระบบจริงในที่นี้หมายถึง ใช้ process simulator เป็นกระบวนการ และมีคอมพิวเตอร์เป็นตัวควบคุม process simulator PTS-10 ที่ใช้ทดลองเป็นกระบวนการอิเลคทรอนิคส์ที่เราสามารถปรับทรานเฟอร์ฟังก์ชัน (transfer function) และพารามิเตอร์ต่าง ๆ ตามต้องการได้ ถึงแม้ว่า PTS-10 จะเป็นกระบวนการจริงแต่ก็ยังเป็นกระบวนการจริงที่มีลักษณะใกล้เคียงกับอุดมคติซึ่งเราสามารถเชื่อถือได้ว่ามีคุณสมบัติเป็นเชิงเส้น และมีสัญญาณรบกวน หรือ noise น้อยมาก PTS-10 จึงเหมาะกับการทดลองในเบื้องต้นในโครงงานนี้

ผู้อ่านวิทยานิพนธ์ฉบับนี้ควรมีความรู้เบื้องต้นสำหรับการควบคุมระบบที่เป็นเชิงเส้นไม่แปรค่าไปตาม เวลาทั้งแบบวิธีดั้งเดิม ซึ่งใช้การแปลงลาปลาสเป็นเครื่องมือในการวิเคราะห์ และวิธีแนวใหม่ที่ใช้เวคเตอร์ เมตริกซ์ เป็นเครื่องมือในการวิเคราะห์ (ทั้งหมดมีอยู่ใน [1])

วิทยานิพนธ์ฉบับนี้ประกอบด้วย

- เนื้อหาส่วนทฤษฎี MRAC สำหรับระบบเชิงเส้น
- ผลการทดลอง ด้วยการจำลองจากคอมพิวเตอร์และการวิเคราะห์
- ผลการทดลองจากการ ทดลองกับระบบจริงและการวิเคราะห์
- และ ภาคผนวก ซึ่งประกอบด้วยโปรแกรมสองโปรแกรม

ในวิทยานิพนธ์ฉบับนี้เน้นเฉพาะทฤษฎี และระเบียบวิธีควบคุมของ MRAC ผลของการนำไปใช้ ปัญหาที่เกิดขึ้น และแนวทางแก้ไข โดยไม่มีรายละเอียดเกี่ยวกับเครื่องมือที่ใช้ประกอบในการทดลอง

บทที่ 2
Model Reference Adaptive Control (MRAC)[2]

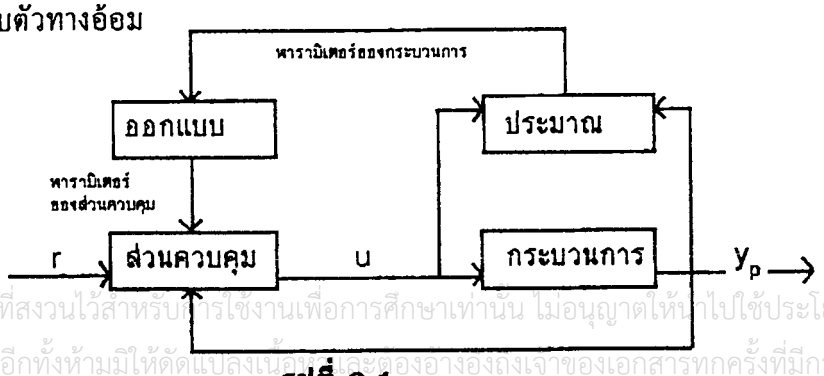
2.1 การควบคุมที่ปรับตัวได้ (Adaptive Control)

ในระหว่างการพัฒนาทฤษฎีการควบคุมแนวใหม่ จะพบว่า การควบคุมที่ถูกสังเคราะห์โดยการประมาณระบบที่ต้องการควบคุม หรือ กระบวนการ ในรูปสมการอนุพันธ์เชิงเส้นไม่แปรค่าตามเวลา และออกแบบตัวควบคุมโดยวิธีสเตทป้อนกลับ มีคุณสมบัติที่ถูกกำหนดตายตัว (ซึ่งต่อไปจะเรียกตัวควบคุมชนิดนี้ว่า การควบคุมคงที่ หรือ fixed controller) การควบคุมคงที่ดังกล่าวไม่สามารถควบคุมพฤติกรรมของระบบให้เป็นไปตามความต้องการ ได้ทุกสถานการณ์ โดยเฉพาะอย่างยิ่ง ถ้ากระบวนการที่ถูกควบคุมมีตัวแปรที่ไม่ทราบค่า หรือ แปรค่าไปตามเวลา

ในช่วงหลังของยุค 1950 ได้มีความสนใจในเรื่องการควบคุมที่ปรับตัวได้มากขึ้น โดยเริ่มจาก gain scheduling technique ซึ่งเป็นการควบคุมที่ปรับพารามิเตอร์ของตัวควบคุมไปตามสภาพแวดล้อมที่สังเกตได้ (คล้ายกับเป็นการสังเกตสภาพแวดล้อมจากนั้นเปิดตารางเพื่อหาค่าพารามิเตอร์หรือเกนที่เหมาะสม) การจะออกแบบตัวควบคุมแบบนี้สามารถทำได้เมื่อเรารู้ความสัมพันธ์ที่แน่นอนระหว่างพฤติกรรมของกระบวนการกับสภาวะภายนอกซึ่งสามารถวัดได้ เช่น ปฏิกิริยาของเครื่องบินจากคำสั่งของนักบินขึ้นโดยตรงกับระดับความสูงของเครื่องบิน เนื่องจากพฤติกรรมของเครื่องบินเป็นที่ทราบกันว่าเป็นฟังก์ชันอย่างไรกับความสูงดังนั้นจึงสามารถออกแบบตัวควบคุมที่สามารถปรับตัวไปตามระดับความสูงต่าง ๆ

คำนิยามของ **ระบบควบคุมที่ปรับตัวได้** มีอยู่หลายนิยาม ยกตัวอย่างเช่น **"ระบบที่มีการควบคุมซึ่งสามารถปรับปรุงตัวเองให้เหมาะสมตามการเปลี่ยนแปลงของกระบวนการ "** แต่นิยามที่ใช้กันมากก็คือ **"ระบบที่ประกอบด้วยการป้อนกลับปฐมภูมิ (primary feedback) ซึ่งคอยดูแลการเปลี่ยนแปลงสัญญาณของกระบวนการ และ การป้อนกลับทุติยภูมิ (secondary feedback) ซึ่งเกี่ยวข้องกับการเปลี่ยนแปลงพารามิเตอร์ของส่วนควบคุม"** จากนิยามนี้การป้อนกลับปฐมภูมิเป็นเช่นดั่งการควบคุมที่ไม่สามารถปรับปรับตัวได้โดยทั่วไป ในขณะที่การป้อนกลับทุติยภูมิจะทำให้เกิดสภาพที่สามารถปรับตัวได้ การควบคุมที่ปรับตัวได้ดังที่ได้นิยามข้างต้นต่างจากเทคนิค gain scheduling ตรงที่ gain scheduling ไม่ได้ใช้การป้อนกลับทุติยภูมิเป็นกลไกในการปรับพารามิเตอร์ แต่ใช้กลไกของการวัดสภาวะแวดล้อมภายนอก ดังนั้น เทคนิค gain scheduling อาจไม่นับว่าเป็นการควบคุมที่ปรับตัวได้ จุดประสงค์ของการเปลี่ยนแปลงพารามิเตอร์ในการป้อนกลับทุติยภูมิคือต้องการรักษาสมรรถภาพ (performance) ให้สูงแม้ว่าพารามิเตอร์ของกระบวนอาจไม่ทราบค่าหรือมีค่าที่เปลี่ยนแปลง

โครงสร้างของระบบควบคุมที่ปรับตัวได้สามารถแบ่งออกได้หลายวิธี วิธีที่ใช้มากได้แก่ การปรับตัวโดยตรง (direct adaptation) และ การปรับตัวทางอ้อม (indirect adaptation) ในรูปที่ 2.1 แสดงรูปของระบบปรับตัวทางอ้อม

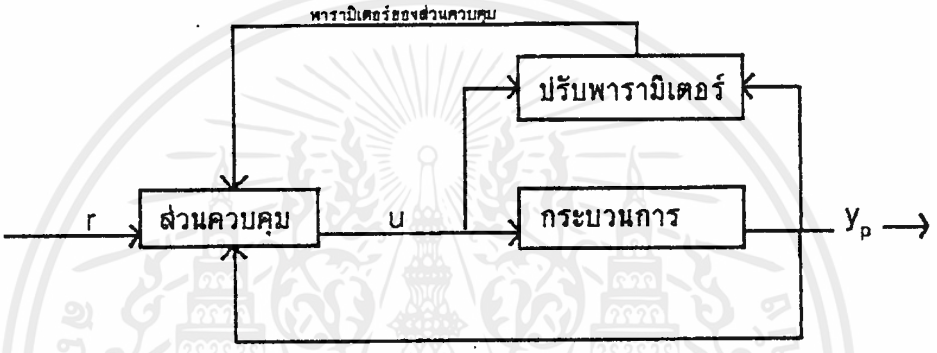


รูปที่ 2.1

จากรูปที่ 2.1 ลูปป้อนกลับปฐมภูมิหรือส่วนควบคุมปฐมภูมิ ได้แก่ ลูปล่าง ประกอบด้วย ส่วนควบคุมและกระบวนการ ซึ่งถูกป้อนกลับด้วยสัญญาณที่วัดได้จาก y_p ลูปป้อนกลับทุติยภูมิ ได้แก่ ลูปบน อันประกอบด้วย ส่วนควบคุม กระบวนการ ส่วนประมาณ และส่วนออกแบบ

ความสามารถในการปรับตัวได้ของระบบปรับตัวทางอ้อมในรูปที่ 2.1 เกิดจากลูปป้อนกลับทุติยภูมิโดยส่วนประมาณจะประมาณพารามิเตอร์ของกระบวนการและส่งต่อให้กับส่วนออกแบบเพื่อออกแบบพารามิเตอร์ให้กับส่วนควบคุม เมื่อกระบวนการมีการเปลี่ยนแปลงพารามิเตอร์ ส่วนออกแบบก็จะออกแบบพารามิเตอร์ของส่วนควบคุมให้เหมาะสม กับ กระบวนการที่เปลี่ยนไป ส่วนควบคุมที่ได้รับการปรับพารามิเตอร์ตามกระบวนการนี้ จะควบคุมพฤติกรรมของกระบวนการ ด้วยกลไกของลูปปฐมภูมิต่อไป

โครงสร้างของระบบปรับตัวโดยตรงมีรูปแบบคล้ายกับระบบปรับตัวทางอ้อมในรูปที่ 2.1 เพียงแต่ลบส่วนออกแบบทิ้ง และเปลี่ยนส่วนประมาณเป็นส่วนปรับพารามิเตอร์ของส่วนควบคุมดังรูปที่ 2.2

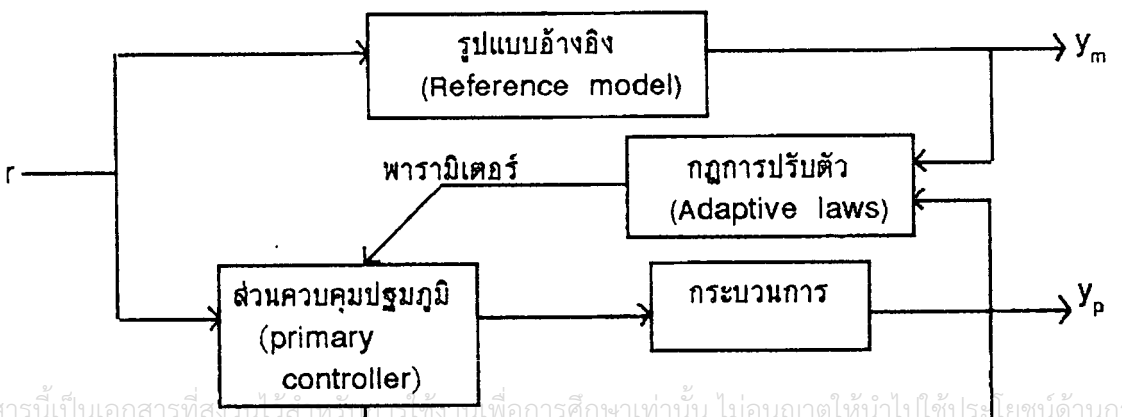


รูปที่ 2.2

ทั้งโครงสร้างของ การปรับตัวโดยตรง และทางอ้อม มีทั้งข้อดีและข้อเสีย ข้อดีแบบทางอ้อมคือความอิสระที่จะเลือกรูปแบบของส่วนออกแบบ ข้อดีข้อที่สองของแบบทางอ้อมคือ การมีส่วนประมาณพารามิเตอร์ จะทำให้ผู้ปฏิบัติการได้รู้ถึงสถานะปัจจุบันของกระบวนการและข้อผิดพลาดต่าง ๆ ของกระบวนการ อย่างไรก็ตามการปรับตัวทางอ้อมไม่อำนวยต่อการวิเคราะห์ทางทฤษฎี เพราะว่าระบบจะซับซ้อน โครงสร้างการปรับตัวโดยตรงมักจะซับซ้อนน้อยกว่า ดังนั้นเราสามารถพิสูจน์เสถียรภาพของระบบโดยรวมได้ง่ายกว่า

2.2 Model Reference Adaptive Systems

เทคนิคการควบคุมแบบ Model Reference Adaptive Control (MRAC) ถูกเสนอครั้งแรกโดย Whitacker ในปี 1958 โครงสร้างของ MRAC ที่นิยมกันมากที่สุดเป็นดังรูปที่ 2.3



รูปที่ 2.3

ในรูปที่ 2.3แสดงให้เห็นว่าส่วนควบคุมปฐมภูมิจะทำหน้าที่ควบคุมพฤติกรรมของระบบลูปปิด ในลักษณะเดียวกับระบบที่ไม่ปรับตัวโดยทั่วไป อย่างไรก็ตามในสถานการณ์ที่เราไม่ทราบค่าพารามิเตอร์ของระบบ หรือ พารามิเตอร์ของระบบแปรค่าไปตามเวลา การตั้งค่าพารามิเตอร์ของส่วนควบคุมปฐมภูมิให้ตายตัว จะไม่สามารถควบคุมพฤติกรรมของระบบลูปปิดให้เป็นไปตามต้องการได้ทุกสถานการณ์

โดยเทคนิคของ MRAC ผลตอบสนองหรือพฤติกรรมของระบบต่ออินพุท r ตามที่เราต้องการสามารถกำหนดโดยรูปแบบอ้างอิง หรือ reference model ซึ่งแสดงด้วย y_m องค์ประกอบในการปรับตัวจะติดตามเปรียบเทียบผลตอบสนองของกระบวนการ y_p กับ y_m และ ตั้งค่าพารามิเตอร์ให้กับส่วนควบคุมปฐมภูมิเพื่อที่จะนำผลตอบสนองของกระบวนการให้เข้าสู่ผลตอบสนองของรูปแบบอ้างอิง ในองค์ประกอบของส่วนปรับตัว นอกจากใช้ผลตอบสนองของกระบวนการ y_p แล้วยังอาจใช้ สเตท (state) ของกระบวนการ x_p และอินพุทของกระบวนการ u หรือ set point r ลักษณะของ MRAC ส่วนใหญ่มักจะเป็นการปรับตัวโดยตรง เพราะไม่มีส่วนของการประมาณพารามิเตอร์ของกระบวนการอย่างชัดเจน

จากรูปที่ 2.3 แสดงระบบ MRAC ซึ่งประกอบด้วยลูปป้อนกลับสองลูป ซึ่งได้กล่าวไว้ในรูป 2.1 และ 2.2 : ลูปในประกอบด้วยลูปป้อนกลับปฐมภูมิ ลูปนอกประกอบด้วยส่วนที่ทำหน้าที่ปรับตัว โดยทั่วไป ลูปป้อนกลับปฐมภูมิจะทำงานที่ความเร็วสูงกว่า ลูปปรับตัว ดังนั้นพารามิเตอร์ของกระบวนการจึงถูกสมมติว่า เปลี่ยนแปลงช้ากว่าสเตทของกระบวนการ

ในกรณีรูปที่ 2.3 แสดงให้เห็นว่ารูปแบบอ้างอิงจะขนานกับกระบวนการ แต่ไม่จำเป็นว่าระบบปรับตัว MRAC จำเป็นจะต้องมีรูปแบบอ้างอิงที่ขนานกับกระบวนการเสมอไป ระบบ MRAC บางครั้งจะนำรูปแบบอ้างอิงต่ออนุกรมกับกระบวนการ ในการต่ออนุกรมนี้ รูปแบบอ้างอิงจะทำหน้าที่เป็นตัวกำเนิดอินพุทอ้างอิง และให้ประโยชน์ต่อการควบคุมตรงที่ระบบโดยทั่วไปจะไม่สามารถตอบสนองต่ออินพุทที่เป็นขั้นบันไดได้ ดังนั้นตัวกำเนิดอินพุทอ้างอิงจะกรองสัญญาณอินพุทให้เรียบขึ้นและการตอบสนองต่อกระบวนการจะเป็นไปตามอินพุทได้ดีขึ้น ในลักษณะนี้มีผลต่อการอิ่มตัวของระบบด้วย (saturation) นอกจากรูปแบบอ้างอิงจะต่อขนาน และต่ออนุกรมกับกระบวนการแล้ว ยังสามารถนำทั้งสองแบบมารวมกัน เป็น MRAC แบบ อนุกรม-ขนานได้อีก ซึ่งมีข้อดีข้อเสียต่างกัน ในวิทยานิพนธ์ฉบับนี้ศึกษาเฉพาะ MRAC แบบที่มีรูปแบบอ้างอิงต่อขนานกับกระบวนการ (parallel MRAC)

ส่วนสำคัญที่สุดใน MRAC ได้แก่การออกแบบกฎทางการปรับตัว ซึ่งมีหลายวิธี แต่ในวิทยานิพนธ์ฉบับนี้จะศึกษา และทดลองเฉพาะวิธี เสถียรภาพของลีอาปูนอฟ (Lyapunov)

2.3 การออกแบบตัวควบคุมที่สามารถปรับตัวได้

ในหัวข้อนี้จะได้แนะนำส่วนประกอบที่สำคัญของระบบ MRAC อันได้แก่ กระบวนการ (Proceess) ตัวควบคุมปฐมภูมิ (Primary controller) รูปแบบอ้างอิง (Reference model) และที่มาของกฎการปรับตัว

กระบวนการ

แม้ว่ากระบวนการจะไม่อยู่ในส่วนของการออกแบบตัวควบคุมที่ปรับตัวได้ แต่เนื่องจากกระบวนการเป็นส่วนที่ตัวควบคุมจะต้องเกี่ยวข้องด้วย จึงสมควรที่จะได้พิจารณาในส่วนนี้ ในการออกแบบตัวควบคุมเราจะต้องสมมติกระบวนการให้เป็นไปในทิศทางบางอย่างที่อาจไม่เป็นจริงในธรรมชาติ ยกตัวอย่างเช่น ทฤษฎีการปรับตัวที่จะพิจารณาต่อไปจะต้องสมมติว่ากระบวนการควรจะเป็นเชิงเส้น ควรทราบอันดับของทรานเฟอร์ฟังก์ชัน และควรรู้ถึงเครื่องหมายของ DC gain รวมทั้งทรานเฟอร์ฟังก์ชันควรเป็น minimum phase และ กระบวนการควรจะปราศจากการรบกวน (disturbance)

ดังที่ได้กล่าวไว้แล้วว่ากระบวนการจะไม่มีทางสอดคล้องกับเงื่อนไขทั้งหมดนี้ ยกตัวอย่างเช่น อันดับของกระบวนการที่แน่นอนเราไม่สามารถรู้ได้ ซึ่งโดยมากอันดับของกระบวนการที่นำมาใช้จะมาจากการวิเคราะห์ถึงส่วนประกอบของระบบ หรือ ดูผลตอบสนองทั้งในโดเมนของเวลาหรือ โดเมนความถี่ และเป็นที่แน่นอนว่าไม่มีกระบวนการที่เป็นเชิงเส้นอย่างแท้จริง หรือปราศจากการรบกวน

อย่างไรก็ตามถึงแม้ว่าคุณสมบัติของกระบวนการจะไม่สามารถเป็นไปตามที่สมมติได้ทั้งหมด ก็ไม่ได้หมายความว่าการควบคุมที่ปรับตัวได้จะไม่สามารถเป็นไปได้ ดังนั้นสิ่งสำคัญในอันดับต่อมาก็คือ จะทำอย่างไรให้ตัวควบคุมที่ปรับตัวได้สามารถทำงานได้ภายใต้สภาวะที่ไม่สมบูรณ์ นั่นคือ สามารถทำงานได้ในกระบวนการจริง ๆ ซึ่งคือแนวทางที่ได้มีการวิจัยในเรื่องการควบคุมที่ปรับตัวได้ในช่วงสิบปีหลังนี้ และในวิทยานิพนธ์นี้ก็มีทั้งส่วนการจำลองสถานการณ์ที่อยู่ภายใต้สภาวะสมมติทุกประการ และการทดลองกับระบบภายใต้สภาวะที่ไม่สมบูรณ์ซึ่งชี้ให้เห็นถึงความแตกต่างในหลาย ๆ ด้าน

ส่วนควบคุมปฐมภูมิ (The primary controller)

ส่วนควบคุมปฐมภูมิดังรูปที่ 2.3 โดยหลักการสามารถมีโครงสร้างใด ๆ ก็ได้ตามการออกแบบตัวควบคุมเชิงเส้น (มีความหมายเดียวกับตัวควบคุมคงที่ดังที่ได้กล่าวมาและสามารถค้นคว้าได้จาก [1]) แต่อย่างไรก็ตามควรจะต้องสอดคล้องกับคุณสมบัติบางประการ

ประการแรกจะต้องสอดคล้องกับสภาวะ Perfect Model-Matching ซึ่งหมายถึงการตั้งค่าพารามิเตอร์ของตัวควบคุมจะต้องทำให้พฤติกรรมของลูปวงปิดเท่ากับรูปแบบอ้างอิงได้

ประการที่สองสำหรับการใช้การควบคุมโดยตรง สัญญาณควบคุม u จะต้องมีคุณสมบัติเป็นเชิงเส้นต่อพารามิเตอร์ โดยทั่วไปส่วนควบคุมปฐมภูมิที่ใช้ใน MRAC ประกอบด้วยส่วนที่ทำหน้าที่กำเนิดเวคเตอร์ของสัญญาณ (Signal Vector $\boldsymbol{\omega}$) ซึ่งสัญญาณควบคุม u จะอยู่ในรูป $u = \boldsymbol{\theta}^T\boldsymbol{\omega}$ โดย $\boldsymbol{\theta}$ เป็นเวคเตอร์ของพารามิเตอร์ของตัวควบคุม ซึ่งในระเบียบวิธีที่จะนำเสนอต่อไปจะได้ว่า $\boldsymbol{\omega}$ จะประกอบขึ้นจากสเตท $\mathbf{x}_p$ และสัญญาณอ้างอิง (reference signal , setpoint) r หรือถ้าไม่ทราบสเตทก็จะต้องประมาณในลักษณะคล้ายกับตัวสังเกตสเตท

รูปแบบอ้างอิง (The Reference Model)

รูปแบบอ้างอิงจะเป็นตัวบ่งชี้ถึงผลตอบสนองต่อสัญญาณอ้างอิง r ที่เราต้องการ รูปแบบอ้างอิงมักจะได้จากการจำลองภายในคอมพิวเตอร์ การกำหนดรูปแบบอ้างอิงต้องกระทำให้สมเหตุสมผลคือระบบในความเป็นจริงสามารถเป็นไปตามรูปแบบนั้น ๆ ได้ เพราะหากกำหนดรูปแบบอ้างอิงที่มีผลตอบสนองเร็วเกินไปจะทำให้เกิดสภาวะอิ่มตัว (saturation) ในส่วนหนึ่งส่วนใดของระบบได้ และจะนำมาซึ่งความไม่เสถียรของระบบดังจะได้แสดงถึงผลของการอิ่มตัวต่อเสถียรภาพของระบบต่อไปในการทดลอง

กฎการปรับตัว (Adaptive Laws)

กุญแจสำคัญในการออกแบบ MRAC คือ กฎการปรับตัวและ การพิสูจน์กฎการปรับตัวซึ่งในวิทยานิพนธ์นี้จะใช้วิธีของลีอาปูนอฟ โดยสมมติว่ากระบวนการเป็นเชิงเส้นที่มีพารามิเตอร์ไม่ทราบค่าหรือ แปรค่าตามเวลา และกระบวนการมีหนึ่งอินพุท หนึ่งเอาท์พุท และการควบคุมเป็นสเตทป้อนกลับและพารามิเตอร์ของตัวควบคุมที่จะปรับคือเกนป้อนกลับ (feedback gain)

2.4 วิธีของลีอาปูนอฟ (Lyapunov's method)

เนื่องจากคุณสมบัติไม่เป็นเชิงเส้นและแปรค่าไปตามเวลาของระบบ MRAC การวิเคราะห์ทางเชิงเส้นจึงไม่สามารถใช้ได้ เพื่อรับประกันเสถียรภาพของระบบปรับตัวได้ จึงจำเป็นต้องใช้วิธีที่รู้จักกันดีคือ Lyapunov's direct method วิธีของลีอาปูนอฟกล่าวว่าระบบจะ uniform asymtotically stable equilibrium ที่ x = 0 เมื่อมี Lyapunov function V(x) ซึ่งสอดคล้องกับเงื่อนไขต่อไปนี้

$V(x) > 0$ เมื่อ $x \neq 0$ (positive definite)
$V'(x) < 0$ เมื่อ $x \neq 0$ (negative definite)
$V(x) \rightarrow \infty$ เมื่อ $||x|| \rightarrow \infty$
$V(0) = 0$

นั่นคือ Lyapunov function V (ซึ่งมีความหมายดังเช่นพลังงานที่บรรจุในระบบ) จะต้องลดลงเรื่อย ๆ เมื่อเทียบกับเวลา การใช้ Lyapunov ในการออกแบบระบบปรับตัวได้ เสถียรภาพของระบบจะถูกจัดเข้าไปอยู่ในกฎการควบคุมตามขั้นตอนต่อไปนี้

- ขั้นแรกจะต้องหา error equation ซึ่งอธิบายถึงสมการอนุพันธ์ของ error signal **e** (ซึ่งอาจจะเป็น output error $y_p - y_m$ หรือ state error $\mathbf{x}_p - \mathbf{x}_m$)

- ขั้นที่สองเลือก Lyapunov function ซึ่งเป็นฟังก์ชันของทั้ง error signal และ parameter error ซึ่งประกอบด้วย signal error vector $\mathbf{e} = \mathbf{x}_p - \mathbf{x}_m$ และ parameter error vector $\boldsymbol{\phi} = \boldsymbol{\theta} - \boldsymbol{\theta}^*$ (ผลต่างระหว่างพารามิเตอร์ของระบบจริงกับรูปแบบจำลอง) ซึ่งปกติจะเลือก Lyapunov function ให้อยู่รูป:

$$V = \mathbf{e}^T\mathbf{P}\mathbf{e} + \boldsymbol{\phi}^T\boldsymbol{\Gamma}^{-1}\boldsymbol{\phi}$$

โดยเมตริกซ์ **P** และ $\boldsymbol{\Gamma}^{-1}$ เป็น positive definite

- ขั้นที่สามคำนวณ V' ซึ่งเป็นอนุพันธ์ของ V ต่อเวลา ซึ่ง V' มักจะอยู่ในรูป:

$$V' = -\mathbf{e}^T\mathbf{Q}\mathbf{e} + \{\text{ส่วนที่มีเทอม } \boldsymbol{\phi}\}$$

โดย **Q** เป็นเมตริกซ์ positive definite หากเราให้เทอมที่ติด $\boldsymbol{\phi}$ เป็นศูนย์ระบบจะมีเสถียรภาพ เมตริกซ์ **P** และ **Q** สามารถถูกเลือกโดยใช้ทฤษฎีลีอาปูนอฟ ซึ่งระบบที่เป็น asymtotically stable ที่ถูกกำหนดโดยเมตริกซ์ **A** ความสัมพันธ์ระหว่างเมตริกซ์ positive definite **Q** และ **P** เป็นดังนี้

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} = -\mathbf{Q}$$

สมการนี้เรียกว่าสมการลีอาปูนอฟ (Lyapunov equation)

- ขั้นที่สี่เมื่อให้ส่วนที่มีเทอม $\boldsymbol{\phi}$ เท่ากับศูนย์นอกจากจะทำให้ได้ระบบที่มีเสถียรภาพแล้ว ยังให้เกิดสมการที่ใช้ปรับพารามิเตอร์ของส่วนควบคุม ซึ่งก็คือกฎการปรับตัวนั่นเอง

ในหัวข้อต่อไปจะแสดงขั้นตอนการหากฎควบคุมดังที่ได้กล่าวมาทั้งสี่ข้อโดยละเอียด โดยศึกษาเฉพาะระบบที่เป็นเชิงเส้นและมีหนึ่งเอาท์พุทและหนึ่งอินพุท และวิเคราะห์ในแบบต่อเนื่องของเวลา

2.5 MRAC แบบเวลาต่อเนื่อง (Continuous-time MRAC)

การพัฒนา MRAC จะมีพื้นฐานอยู่บนรูปแบบเวลาต่อเนื่อง ในวิทยานิพนธ์นี้ก็ศึกษาและทดลอง MRAC ในแบบเวลาต่อเนื่องเช่นเดียวกัน โดยระเบียบวิธีที่จะกระจายต่อไปนี้มีขอบเขตอยู่ที่ ระบบต่อเนื่องเชิงเส้น ที่มีเฉพาะโพล (pole) ไม่มีซีโร่ (zero) และมีหนึ่งอินพุทและหนึ่งเอาท์พุท

MRAC เมื่อทราบสภาวะที่สมบูรณ์

ในหัวข้อที่จะกล่าวถึงต่อไปจะพิจารณาการออกแบบ MRAC สำหรับกระบวนการที่ทราบค่าเวคเตอร์ของสภาวะหรือสเตทเวคเตอร์ $\mathbf{x}_p$ โดยสมบูรณ์

2.5.1 โครงสร้างของตัวควบคุมปฐมภูมิ (Primary controller structure)

ในที่นี้จะกำหนดว่ากระบวนการเป็นเชิงเส้น และ สามารถควบคุมได้โดยสมบูรณ์ และไม่มีzero อันดับของกระบวนการคือ n สเตทป้อนกลับ $\mathbf{k}_b^T \mathbf{x}_p + k_0 r$ จะทำให้สามารถกำหนดตำแหน่งของ closed-loop pole ณ ตำแหน่งใด ๆ ก็ได้ ดังนั้นสามารถทำให้เกิดสภาพที่เข้ากับรูปแบบอ้างอิงได้โดยสมบูรณ์ (perfect model-matching condition)

$$\mathbf{x}_p' = \mathbf{A}_p\mathbf{x}_p + \mathbf{b}_p u$$

$$y_p = \mathbf{c}^T\mathbf{x}_p$$

$\mathbf{A}_p$ สมมติว่าอยู่ในรูป phase-variable form

$$\mathbf{A}_p = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ & & \vdots & & \vdots \\ -a_{p1} & -a_{p2} & -a_{p3} & \cdots & -a_{pn} \end{pmatrix}$$

$$\mathbf{b}_p = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ b_{pn} \end{pmatrix}$$

$$\mathbf{c}^T = (1, \ 0, \ \ldots, \ 0) \qquad (2.1)$$

และมีสเตทป้อนกลับ

$$u = \mathbf{k}_b^T \mathbf{x}_p + k_0 r = (k_1, \ k_2, \ \ldots, \ k_n)\,\mathbf{x}_p + k_0 r \qquad (2.2)$$

จาก (2.2) จะเขียนให้อยู่ในรูป

$$u = \boldsymbol{\theta}^T\boldsymbol{\omega} \qquad \text{โดย } \boldsymbol{\theta}^T = (k_0, \ \mathbf{k}_b^T)$$
$$\text{โดย } \boldsymbol{\omega}^T = (r, \ \mathbf{x}_p^T) \qquad (2.3)$$

สมการที่ (2.1) และ (2.2) ทำให้ได้ closed loop system

$$\mathbf{x}_p' = (\mathbf{A}_p + \mathbf{b}_p\mathbf{k}^T)\mathbf{x}_p + \mathbf{b}_p k_0 r$$

$$= \begin{pmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ & \vdots & & \vdots \\ -a_{p1}+b_{pn}k_1 & -a_{p2}+b_{pn}k_2 & \cdots & -a_{pn}+b_{pn}k_n \end{pmatrix} \mathbf{x}_p + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ b_{pn}k_0 \end{pmatrix} r$$

$$= \mathbf{A}_c\mathbf{x}_p + \mathbf{b}_c r \qquad (2.4)$$

พารามิเตอร์ของกระบวนการ a_{p1} a_{p2} ... a_{pn} และ b_{pn} สมมติว่าไม่ทราบค่าแต่มีค่าคงที่ และรู้เครื่องหมายของ b_{pn} จะสังเกตได้ว่าค่าพารามิเตอร์ของสเตทป้อนกลับ หรือ ก็คือตัวควบคุมปฐมภูมิ k_0 k_1 k_2 ... k_n จะสามารถกำหนดพารามิเตอร์ทุกตัวของระบบลูปปิดโดยรวมได้ ซึ่งมีผลในการย้ายตำแหน่งโพลไปอยู่ในตำแหน่งตามที่ต้องการ กลไกการปรับตัว ของ MRAC ในแบบที่นำเสนอนี้ก็ใช้วิธีปรับพารามิเตอร์ k ตัวนี้เพื่อที่จะปรับพารามิเตอร์ของระบบลูปปิด หากสเตทเวคเตอร์ $\mathbf{x}_p$ ยังไม่อยู่ในรูป phase variable form เราสามารถเปลี่ยนให้อยู่ในรูป phase variable form ได้

รูปแบบอ้างอิง (reference model) ต้องกำหนดให้อยู่ในรูปเดียวกับ กระบวนการ

$$\mathbf{x}_m' = \mathbf{A}_m\mathbf{x}_m + \mathbf{b}_m r$$

$$= \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ & & \vdots & & \vdots \\ -a_{m1} & -a_{m2} & -a_{m3} & \cdots & -a_{mn} \end{pmatrix} \mathbf{x}_m + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ b_{pn}k_0 \end{pmatrix} r$$

ด้วยการกำหนดโมเดลให้อยู่ในรูปเดียวกับกระบวนการ เราสามารถปรับองค์ประกอบทุกตัวของเมตริกซ์ของระบบ $\mathbf{A}_c$ และ $\mathbf{b}_c$ ให้ตรงกับเมตริกซ์ของรูปแบบอ้างอิง $\mathbf{A}_m$ และ $\mathbf{b}_m$ ได้ ซึ่งมีความหมายว่าเราสามารถทำให้ closed-loop transfer function ของกระบวนการ และ รูปแบบอ้างอิงตรงกันได้ ด้วยการเลือกพารามิเตอร์ของตัวควบคุม (k) ที่เหมาะสม ในลักษณะเช่นนี้เรียกว่าเกิดสภาพที่เข้ากับรูปแบบอ้างอิงได้โดยสมบูรณ์ (perfect model-matching condition)

หากระบบเกิดมี zero โครงสร้างของส่วนควบคุมปฐมภูมิจะต้องถูกขยายออกไปอีก

2.5.2 การสร้างกฎการปรับตัว

กฎการปรับตัว สามารถสร้างได้โดยใช้วิธีของลีอาปูนอฟสี่ขั้นตอนดังที่ได้กล่าวมาแล้วในหัวข้อที่ 2.4

การสร้างสมการ error

สำหรับอนุพันธ์เทียบกับเวลาของสัญญาณ error vector $\mathbf{e} = \mathbf{x}_p - \mathbf{x}_m$ จะมีรูปสมการดังนี้

$$\begin{aligned} \mathbf{e}' &= \mathbf{x}_p' - \mathbf{x}_m' \\ &= \mathbf{A}_c\mathbf{x}_p + \mathbf{b}_c r - \mathbf{A}_m\mathbf{x}_m - \mathbf{b}_m r \\ &= \mathbf{A}_m(\mathbf{x}_p - \mathbf{x}_m) + (\mathbf{A}_c - \mathbf{A}_m)\mathbf{x}_p + (\mathbf{b}_c - \mathbf{b}_m)r \\ &= \mathbf{A}_m\mathbf{e} + \mathbf{A}\mathbf{x}_p - \mathbf{b}r \end{aligned} \tag{2.5}$$

เมื่อ

$$\begin{aligned} \mathbf{A} &= \mathbf{A}_c - \mathbf{A}_m \\ \mathbf{b} &= \mathbf{b}_c - \mathbf{b}_m \end{aligned} \tag{2.6}$$

กำหนด parameter error vector ϕ

$$\phi^T = (b_{pn}k_0 - b_{mn},\ -a_{p1} + b_{pn}k_1 - a_{m1},\ \ldots,\ -a_{pn} + b_{pn}k_n + a_{mn}) \tag{2.7}$$

ใช้สมการ (2.7) และ (2.3) แทนในสมการที่ (2.5)

$$\begin{aligned} \mathbf{e}' &= \mathbf{A}_m\mathbf{e} + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} \phi^T\omega \\ &= \mathbf{A}_m\mathbf{e} + \mathbf{b}_I\phi^T\omega \end{aligned} \tag{2.8}$$

สมการ (2.8)แสดงสมการ error ของระบบซึ่งประกอบด้วยส่วนที่เป็น เชิงเส้น $\mathbf{A}_m$ และ $\mathbf{b}_I$ ซึ่งถูกควบคุมโดยผลคูณที่ไม่เป็นเชิงเส้น $\phi^T\omega$

Lyapunov function

ขั้นต่อไปในการออกแบบคือการเลือก Lyapunov function ดังได้กล่าวไว้แล้วในหัวข้อที่ 2.4 โดยทั่วไป Lyapunov function จะเป็น quadratic function ของทั้ง signal error vector $\mathbf{e}$ และ parameter error vector ϕ

$$V = \mathbf{e}^T\mathbf{P}\mathbf{e} + \phi^T\Gamma^{-1}\phi \tag{2.9}$$

adaptive gain matrix (Γ) จะต้องเป็น positive definite ดังนั้นจึงเลือกให้เป็นเมตริกเส้นแทยงมุม และเป็นผลให้ Γ^{-1} เป็น positive definite ด้วย และเมตริกซ์ $\mathbf{P}$ จะต้องเป็น positive definite

หาอนุพันธ์ของ V และการสร้างกฎการปรับตัว

เพื่อให้ได้ระบบที่ asymtotically stable adaptive system, V' จะต้องเป็น negative definite อนุพันธ์ของ V ได้แก่

$$V' = \mathbf{e}^T(\mathbf{A}_m{}^T\mathbf{P} + \mathbf{P}\mathbf{A}_m)\mathbf{e} + 2\mathbf{e}^T\mathbf{P}\mathbf{b}_I\boldsymbol{\phi}^T\boldsymbol{\omega} + 2\boldsymbol{\phi}^T\boldsymbol{\Gamma}^{-1}\boldsymbol{\phi}' \quad (2.10)$$

โดยใช้สมการลีอาปูนอฟ เมตริกซ์ **P** และ **Q** จะสามารถหาค่าได้
เทอมแรกด้านขวาของสมการ (2.10) สอดคล้องกับ

$$\mathbf{e}^T(\mathbf{A}_m{}^T\mathbf{P} + \mathbf{P}\mathbf{A}_m)\mathbf{e} = -\mathbf{e}^T\mathbf{Q}\mathbf{e}$$

เพื่อที่จะประกันว่าระบบลูปปิดจะเสถียร กำหนดให้สองเทอมหลังด้านขวาของสมการ (2.10) เท่ากับศูนย์

$$\begin{aligned} 2\mathbf{e}^T\mathbf{P}\mathbf{b}_I\boldsymbol{\phi}^T\boldsymbol{\omega} + 2\boldsymbol{\phi}^T\boldsymbol{\Gamma}^{-1}\boldsymbol{\phi}' &= 0 \\ \rightarrow \quad \boldsymbol{\phi}' &= -\boldsymbol{\Gamma}\mathbf{e}^T\mathbf{P}\mathbf{b}_I\boldsymbol{\omega} \\ &= -\boldsymbol{\Gamma}(\mathbf{p}^T\mathbf{e})\boldsymbol{\omega} \end{aligned} \quad (2.11)$$

ผลคูณ $\mathbf{P}\mathbf{b}_I$ ในสมการ (2.11) เป็นเวคเตอร์ที่ประกอบด้วยหลักที่ n ของ **P** และผลคูณของเวคเตอร์นี้กับ signal error vector : $\mathbf{p}^T\mathbf{e}$ เรียกว่า 'compensated error' compensated error จะถูกนำมาใช้ในกฎการปรับตัว เพื่อคำนวณหาอัตราการเปลี่ยนแปลงของพารามิเตอร์ หรือ $\boldsymbol{\phi}'$

ขณะที่พารามิเตอร์ของรูปแบบอ้างอิงและกระบวนการถูกสมมติให้คงที่ จากนิยามของ $\boldsymbol{\phi}$ (สมการ 2.6) จะได้ว่า

$$\begin{aligned} b_{pn}k_0' &= -\gamma_{00}(p^Te)r \\ b_{pn}k_i' &= -\gamma_{ii}(p^Te)x_{pi} \qquad \text{เมื่อ } i = 1, 2, ..., n \end{aligned}$$

โดย $\Gamma = \text{diag}(\gamma_{00}, \gamma_{11}, ..., \gamma_{nn})$ ดังนั้น

$$\begin{aligned} k_0' &= -\gamma'_{00}(p^Te)r \\ k_i' &= -\gamma'_{ii}(p^Te)x_{pi} \end{aligned} \quad (2.12)$$

หรือในรูปทั่วไป

$$\boldsymbol{\theta}' = -\boldsymbol{\Gamma}'(\mathbf{p}^T\mathbf{e})\boldsymbol{\omega} \quad (2.13)$$

เมื่อ

$$\boldsymbol{\Gamma}' = (1/b_{pn})\boldsymbol{\Gamma}$$

บทที่ 3
การจำลองระบบควบคุม MRAC ด้วยคอมพิวเตอร์

3.1 จุดประสงค์ในการจำลองระบบด้วยคอมพิวเตอร์

จุดประสงค์ในการจำลองระบบด้วยคอมพิวเตอร์มีสองประการได้แก่

1. ต้องการศึกษาทฤษฎีของ MRAC ว่าจะให้ผลในลักษณะเช่นไร ในขณะที่ส่วนประกอบทุกส่วนเป็นอุดมคติหรืออยู่ภายใต้เงื่อนไขในทฤษฎี

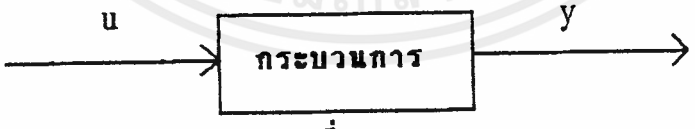
2. แสดงให้เห็นถึงความแตกต่างระหว่างการควบคุมกระบวนการที่มีพารามิเตอร์ไม่แน่นอนด้วยตัวควบคุมคงที่ กับ ตัวควบคุมที่ปรับตัวได้ เมื่อกระบวนการเป็นอุดมคติ

ขอบเขตการจำลองระบบในครั้งนี้ต้องสอดคล้องกับขอบเขตทางทฤษฎีที่กำลังศึกษาที่ได้กล่าวไว้แล้วในบทที่ 2 ซึ่งมีดังนี้ กระบวนการที่จะใช้จำลองมีคุณสมบัติเป็นเชิงเส้น ไม่มีซีโร่ มีหนึ่งอินพุท หนึ่งเอาท์พุท โดยในการศึกษาครั้งนี้จะระบุอันดับของกระบวนการเป็นอันดับสอง

ในหัวข้อที่จะกล่าวถึงต่อไป จะเริ่มด้วยสมมติกระบวนการที่จะถูกควบคุมซึ่งมีผลตอบสนองต่อสัญญาณอ้างอิงไม่ตรงกับความต้องการ และมีพารามิเตอร์เปลี่ยนไปตามจุดทำงานและเราไม่อาจทราบค่า จากนั้นระบุรูปแบบอ้างอิงที่มีผลตอบสนองตรงกับความต้องการ ขั้นต่อไปคือหาการควบคุมคงที่ให้กับกระบวนการ และหาการควบคุมที่ปรับตัวได้ให้กับกระบวนการ โดยทุกขั้นตอนแทนด้วยสมการทางคณิตศาสตร์ นำผลที่ได้ทั้งหมดจำลองด้วยคอมพิวเตอร์ และนำมาแสดงรวมทั้งวิเคราะห์ผล ผลที่แสดงโดยคอมพิวเตอร์จะมีสองหน้าต่าง หน้าต่างบนแสดงผลตอบสนองของกระบวนการที่ถูกควบคุมด้วยตัวควบคุมที่ปรับตัวได้ ส่วนหน้าต่างล่างแสดงผลตอบสนองของกระบวนการที่ถูกควบคุมด้วยตัวควบคุมคงที่

3.2 กระบวนการที่ใช้ศึกษา

กระบวนการที่ใช้ศึกษาแสดงดังแผนภาพข้างล่าง



รูปที่ 3.1

โดยทรานเฟอร์ฟังก์ชัน (transfer function) ระหว่างเอาท์พุท y ต่ออินพุท u ถูกสมมติให้มีสมการดังนี้

$$\frac{y}{u} = \frac{q}{s^2 + 2qs} \qquad (3.1)$$

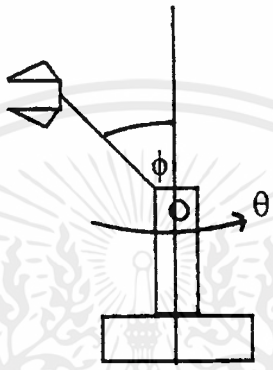
หรือเขียนเป็นสมการเสตทในรูป phase variable form ได้ดังนี้

$$\mathbf{x}' = \begin{pmatrix} 0 & 1 \\ 0 & -2q \end{pmatrix}\mathbf{x} + \begin{pmatrix} 0 \\ q \end{pmatrix}\mathbf{u}$$

$$y = \begin{pmatrix} 1 & 0 \end{pmatrix}\mathbf{x} \qquad (3.2)$$

สมการที่ (3.1) และ (3.2) แสดงคุณลักษณะของระบบ ตัวแปร q เป็นตัวแปรไม่ทราบค่าซึ่งจะเปลี่ยนแปลงไปตามจุดทำงาน

ตัวอย่างระบบจริงที่แสดงได้ด้วยสมการ (3.1) และ (3.2) ได้แก่ ระบบเซอร์โวมอเตอร์ เช่น มอเตอร์ที่ใช้หมุนแขนกลอินพุทเป็นโวลเตจ v เพื่อที่จะทำให้แขนกลหมุนไปเป็นมุม θ ดังรูปที่ 3.2



รูปที่ 3.2

ซึ่งสามารถประมาณทรานเฟอร์ฟังก์ชันได้ดังสมการ (ดูใน [3] และ [4])

$$\frac{\theta}{v} = \frac{K/J}{s^2 + B/J\, s} \qquad (3.3)$$

เป็นรูปเดียวกับสมการที่ (3.1) เมื่อมุมของแขนกล ϕ ยกขึ้นลง จะทำให้โมเมนต์ของความเฉื่อย (moment of inertia) J ใน (3.3) มีค่าเปลี่ยนไป ซึ่งถ้าเปรียบเทียบกับค่า q นั้นคือเราจะไม่ทราบค่า q ใน (3.1) ที่แน่นอน โดยค่า q จะเปลี่ยนไปตามจุดทำงาน ซึ่งในที่นี้เราจะกำหนดให้ที่จุดทำงานปกติค่า q จะเท่ากับ 1 และเปลี่ยนแปลงอยู่ระหว่าง 0.25 ถึง 4

3.3 รูปแบบอ้างอิง

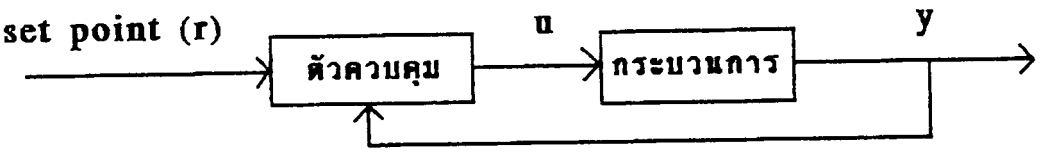
กำหนดให้รูปแบบอ้างอิงมี pole อยู่ที่ $-2 \pm 3.464j$ นั่นคือมีสมการสเตท

$$\mathbf{x}' = \begin{pmatrix} 0 & 1 \\ -16 & -4 \end{pmatrix}\mathbf{x} + \begin{pmatrix} 0 \\ 16 \end{pmatrix}r$$

$$y = \begin{pmatrix} 1 & 0 \end{pmatrix}\mathbf{x} \qquad (3.4)$$

3.4 ออกแบบตัวควบคุมคงที่

ตัวควบคุมที่จะนำมาควบคุมกระบวนการจะอยู่ในรูปแบบดังรูปที่ 3.3



รูปที่ 3.3

ตัวควบคุมคงที่ในที่นี้จะใช้วิธีการควบคุมแบบป้อนกลับสเตท

จะเห็นได้ว่ากระบวนการที่เราสมมติขึ้นในสมการ (3.1) มีตัวอินทิเกรตอยู่หนึ่งตัว ดังนั้นการควบคุมให้ระบบลูปปิดมีโพลตามตำแหน่งที่ต้องการและมีเอาท์พุทตาม set point ซึ่งกำหนดให้เป็น unit step สามารถใช้วิธี 'type 1 servo system when plant has an integrator' ใน [1] ซึ่งมีระเบียบวิธีดังนี้

กระบวนการที่สมมติขึ้นมี หนึ่งอินพุท หนึ่งเอาท์พุท ซึ่งสามารถเขียนให้อยู่ในรูป

$$\begin{aligned} \mathbf{x}' &= \mathbf{A}\mathbf{x} + \mathbf{B}u \\ y &= \mathbf{C}\mathbf{x} \end{aligned} \qquad (3.5)$$

โดยในที่นี้ $\mathbf{x}$ เป็นเวคเตอร์ 2 มิติ $\mathbf{A}$ เป็นเมตริกซ์ 2*2 $\mathbf{B}$ เป็นเมตริกซ์ 2*1 $\mathbf{C}$ เป็นเมตริกซ์ 1*2 และ u และ y เป็นสเกลาร์

ตัวควบคุมจะเป็นสเตทป้อนกลับ

$$\begin{aligned} u &= -\begin{pmatrix} 0 & k_2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + k_1(r - x) \\ &= -\mathbf{K}\mathbf{x} + k_1 r \end{aligned} \qquad (3.6)$$

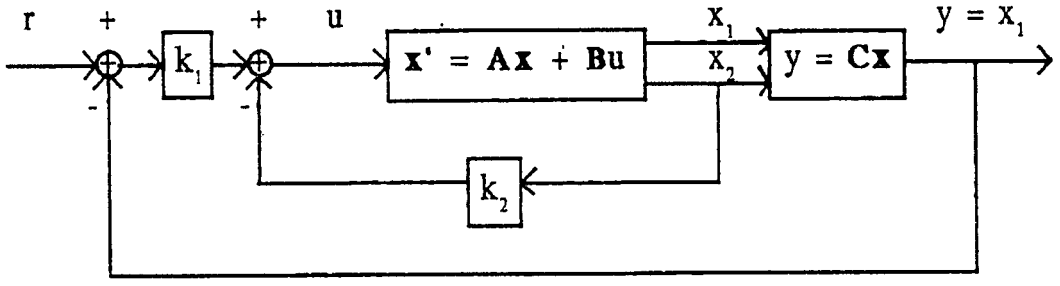
โดย

$$\mathbf{K} = (k_1 \quad k_2)$$

เพราะฉะนั้น

$$\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{B}u = (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x} + \mathbf{B}k_1 r \qquad (3.7)$$

เราสามารถกำหนด pole ของระบบได้ตามต้องการโดยกำหนด เกนป้อนกลับ $\mathbf{K}$ และเนื่องจากกระบวนการมีอินทิเกรเตอร์หนึ่งตัว หรือเป็น type 1 ดังนั้นที่สภาวะอยู่ตัวความแตกต่างของ set point r กับ u จะเท่ากับ 0 แผนภาพระบบลูปปิดสามารถแสดงในรูปที่ 3.4



รูปที่ 3.4

นำระเบียบวิธีในสมการที่ (3.5) (3.6) และ (3.7) มาใช้กับกระบวนการในสมการที่ (3.2) แต่เนื่องจากในสมการที่ (3.2) มีตัวแปร q ที่มีค่าไม่แน่นอน ในขณะที่ สมการที่ (3.5) (3.6) (3.7) ใช้กับกระบวนการที่เป็นเชิงเส้นไม่แปรเปลี่ยนตามเวลา และรู้ค่าแน่นอน ดังนั้นการออกแบบจึงควรออกแบบที่จุดทำงานปกติที่ได้กำหนดไว้ในตอนต้น คือให้ค่า q เป็นค่า 1 จาก (3.2) จะได้

$$\mathbf{x}' = \begin{pmatrix} 0 & 1 \\ 0 & -2 \end{pmatrix}\mathbf{x} + \begin{pmatrix} 0 \\ 1 \end{pmatrix}u$$

$$y = \begin{pmatrix} 1 & 0 \end{pmatrix}\mathbf{x} \qquad (3.8)$$

เมื่อนำสเตทมาป้อนกลับจะได้ระบบลูปปิดเป็น

$$\mathbf{x}' = \left[\begin{pmatrix} 0 & 1 \\ 0 & -2 \end{pmatrix} - \begin{pmatrix} 0 \\ 1 \end{pmatrix}\begin{pmatrix} k_1 & k_2 \end{pmatrix}\right]\mathbf{x} + \begin{pmatrix} 0 \\ 1 \end{pmatrix}k_1 r \qquad (3.9)$$

ค่า k_1 และ k_2 เป็นเกนป้อนกลับซึ่งจะย้ายโพลของกระบวนการ ให้เป็นโพลของลูปปิดตามที่ต้องการ ในที่นี้โพลที่เราต้องการได้กำหนดไว้ในรูปแบบอ้างอิงแล้วคือ $-2 \pm 3.464j$ ซึ่งมีรูปสมการสเตทดังสมการที่ (3.4) ดังนั้นเกนป้อนกลับจึงต้องมีค่า

$$\mathbf{K} = (16 \quad 2)$$

$$k_1 = 16, \quad k_2 = 2$$

เพื่อที่จะให้ได้รูปสมการเสตทเป็นดังสมการที่ (3.4)

จะสังเกตว่าการใช้ตัวควบคุมคงที่ โดยวิธีสเตทป้อนกลับจะสามารถย้ายโพลมาอยู่ในตำแหน่งที่ต้องการและที่สภาวะอยู่ตัวจะมีค่าเท่ากับสัญญาณอ้างอิง หรือ set point แต่อย่างไรก็ตามก็สามารถควบคุมระบบให้อยู่ในสภาพที่ต้องการได้เฉพาะเพียงที่จุดทำงานปกติหรือค่า q = 1 เท่านั้นแต่ในที่นี้เราสมมติว่าค่า q จะเปลี่ยนไปตามจุดทำงานโดยมีค่าอยู่ระหว่าง 0.25 ถึง 4 ดังนั้นการควบคุมจึงไม่ครอบคลุมทุกกรณี ซึ่งจะแสดงในผลการควบคุมอีกครั้งหนึ่ง

3.5 การออกแบบตัวควบคุม MRAC

ทฤษฎีเกี่ยวกับตัวควบคุม MRAC ได้กล่าวไว้แล้วในหัวข้อ 2.5 สามารถสรุปว่าการควบคุมประกอบด้วยส่วนการควบคุมปฐมภูมิ และส่วนปรับตัว การควบคุมปฐมภูมิเป็นสเตทป้อนกลับในรูปแบบเดียวกับตัวควบคุมคงที่ โดยส่วนปรับตัวจะทำหน้าที่ปรับพารามิเตอร์ของตัวควบคุมปฐมภูมิอีกทีหนึ่ง พารามิเตอร์ที่ถูกปรับก็คือเกนป้อนกลับนั่นเอง

ในหัวข้อนี้จะทำการสร้างสมการสำหรับการปรับตัวของพารามิเตอร์ในตัวควบคุม MRAC

จากสมการกระบวนการที่ (3.2)

$$\mathbf{x}' = \begin{pmatrix} 0 & 1 \\ 0 & -2q \end{pmatrix}\mathbf{x} + \begin{pmatrix} 0 \\ q \end{pmatrix}u$$

$$y = \begin{pmatrix} 1 & 0 \end{pmatrix}\mathbf{x} \qquad (3.2)$$

ตัวควบคุมปฐมภูมิเมื่อใช้ full state feedback

$$u = k_0 r + k_1 x_{p1} + k_2 x_{p2} \qquad (3.10)$$

ดังนั้นสมการต่าง ๆ ที่ต้องนำมาใช้ในการพิสูจน์กฎการปรับตัว ซึ่งได้กำหนดไว้ในสมการที่ (2.2) ถึง (2.4) สำหรับกระบวนการนี้จะเป็นดังนี้

$$\mathbf{A}_p = \begin{pmatrix} 0 & 1 \\ 0 & -2q \end{pmatrix}$$

$$\mathbf{b}_p = \begin{pmatrix} 0 \\ q \end{pmatrix}$$

$$\mathbf{A}_c = \mathbf{A}_p + \mathbf{b}_p\mathbf{K}^T = \begin{pmatrix} 0 & 1 \\ qk_1 & -2q+qk_2 \end{pmatrix}$$

$$\mathbf{b}_c = \begin{pmatrix} 0 \\ qk_0 \end{pmatrix}$$

$$\mathbf{A}_p = \begin{pmatrix} 0 & 1 \\ -16 & -4 \end{pmatrix}$$

$$\mathbf{b}_p = \begin{pmatrix} 0 \\ 16 \end{pmatrix} \qquad (3.11)$$

จากสมการ (3.11) สามารถหา error metrix ได้ดังนี้

$$\mathbf{A} = \mathbf{A}_c - \mathbf{A}_m = \begin{pmatrix} 0 & 0 \\ qk_1+16 & -2q+qk_2+4 \end{pmatrix}$$

$$\mathbf{b} = \mathbf{b}_c - \mathbf{b}_m = \begin{pmatrix} 0 \\ qk_0-16 \end{pmatrix} \tag{3.12}$$

จากนิยาม

$$\boldsymbol{\phi}^T = (qk_0-16 \quad qk_1+16 \quad -2q+qk_2+4)$$

$$\boldsymbol{\omega}^T = (r \quad x_{p1} \quad x_{p2}) \tag{3.13}$$

จะได้ error equation

$$\mathbf{e}' = \mathbf{A}_m\mathbf{e} + \begin{pmatrix} 0 \\ 1 \end{pmatrix}\boldsymbol{\phi}^T\boldsymbol{\omega} \tag{3.14}$$

ในกรณีนี้ $\mathbf{e}^T = (e \quad e')$ เมื่อ $e = x_{p1} - x_{m1}$ และ $e' = x_{p2} - x_{m2}$

การเลือกฟังก์ชันลีอาปูนอฟจะได้ V' ดังสมการ (2.10) ในส่วนแรก V' สามารถหาค่า positive definite symmetric metrix **P** โดยใช้สมการของลีอาปูนอฟ ซึ่งในตัวอย่างนี้จะกำหนดให้ Q เป็นเมตริกซ์เอกลักษณ์

$$\mathbf{A}_m{}^T\mathbf{P} + \mathbf{P}\,\mathbf{A} = \mathbf{Q}$$

จะได้

$$\mathbf{P} = \begin{pmatrix} 288 & 4 \\ 4 & 17 \end{pmatrix} \tag{3.15}$$

จากกฎการปรับตัวตามสมการที่ 3.11

$$\boldsymbol{\phi}' = -\boldsymbol{\Gamma}\mathbf{e}^T\mathbf{P}\mathbf{b}_I\boldsymbol{\omega} = -\begin{pmatrix} \gamma_0 & 0 & 0 \\ 0 & \gamma_1 & 0 \\ 0 & 0 & \gamma_2 \end{pmatrix}(e \quad e')\begin{pmatrix} 4 \\ 17 \end{pmatrix}\begin{pmatrix} r \\ x_{p1} \\ x_{p2} \end{pmatrix}$$

$$= -(4e + 17e')\begin{pmatrix} \gamma_0 r \\ \gamma_1 x_{p1} \\ \gamma_2 x_{p2} \end{pmatrix}$$

จะได้กฎการปรับตัวอยู่ในรูปอัตราการเปลี่ยนแปลงของพารามิเตอร์ของส่วนควบคุมปฐมภูมิ

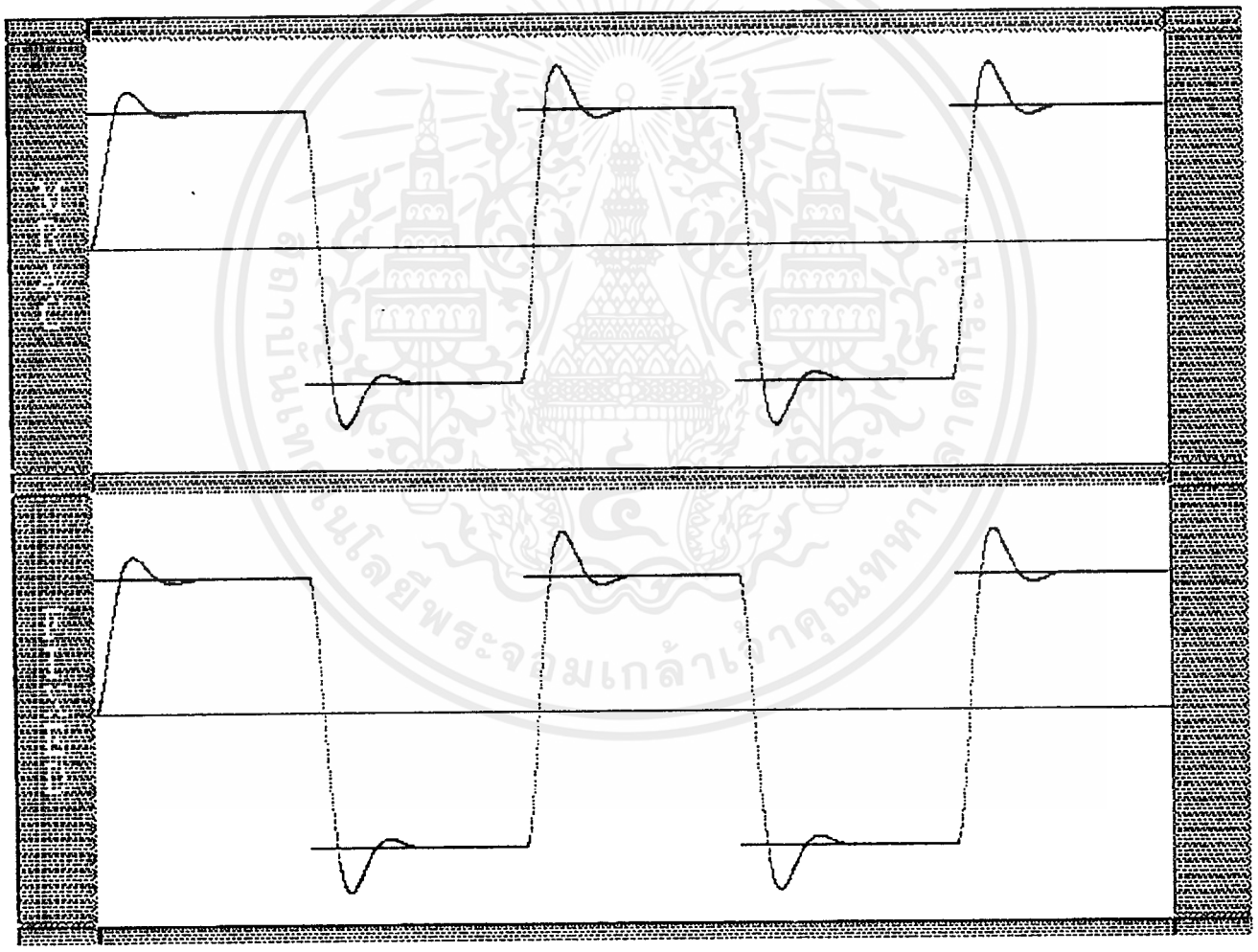
$$\begin{aligned} k_0' &= -\gamma'_0(4e + 17e')r \\ k_1' &= -\gamma'_1(4e + 17e')x_{p1} \\ k_2' &= -\gamma'_2(4e + 17e')x_{p2} \end{aligned} \tag{3.16}$$

จากรูปสมการค่า γ'_i เสมือนเป็นอัตราเร็วในการปรับตัวเราจะเลือกค่าใดก็ได้ในที่นี้จะให้เป็น 10

3.6 ผลการจำลองระบบด้วยคอมพิวเตอร์ และการวิเคราะห์ผล

การจำลองระบบด้วยคอมพิวเตอร์ในหัวข้อนี้ที่ได้นำผลการวิเคราะห์จากหัวข้อที่แล้วมาแปลงเป็นโปรแกรมจำลอง และ นำผลการจำลองมาแสดงในหัวข้อนี้ (ดูโปรแกรมในภาคผนวก ก)

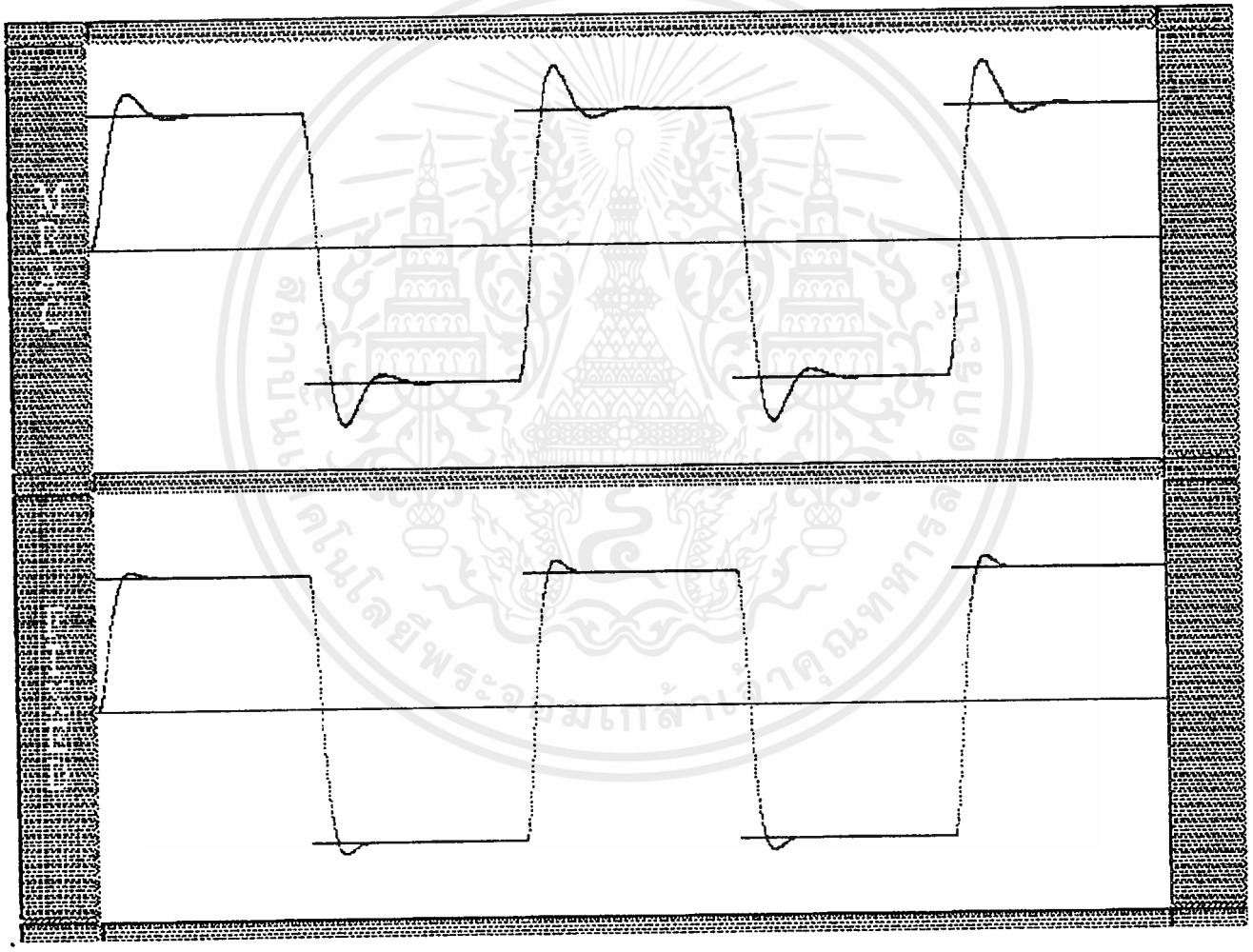
เริ่มแรก จากสมการกระบวนการที่ (3.1) หรือ (3.2) กำหนดให้ค่า q เท่ากับ 1 หรือจุดทำงานปกติ กำหนดค่าเกนป้อนกลับสำหรับตัวควบคุมคงที่ ตามสมการที่ (3.9) เพื่อให้ได้ผลตอบสนองตามรูปแบบอ้างอิง และกำหนดค่าพารามิเตอร์ให้กับตัวควบคุมปฐมภูมิของการควบคุมแบบ MRAC เช่นเดียวกับที่กำหนดให้กับตัวควบคุมคงที่ (**หมายเหตุ** การควบคุมปฐมภูมิของ MRAC กับการควบคุมคงที่จะมีลักษณะเดียวกันคือเป็นการป้อนกลับสเตท ดังนั้นหากควบคุมกระบวนการคงที่ด้วยสองวิธีโดยตั้งค่าเดียวกันก็จะได้ผลตอบสนองเช่นเดียวกัน) การควบคุมทั้งสองแบบจะได้ผลเหมือนกันคือได้ผลตอบสนองตรงกับรูปแบบอ้างอิง คือได้โพลของระบบลูปปิดอยู่ที่ $-2 \pm 3.464j$ ดังรูป



รูปที่ 3.5 ผลตอบสนองเมื่อ q ของกระบวนการเท่ากับ 1

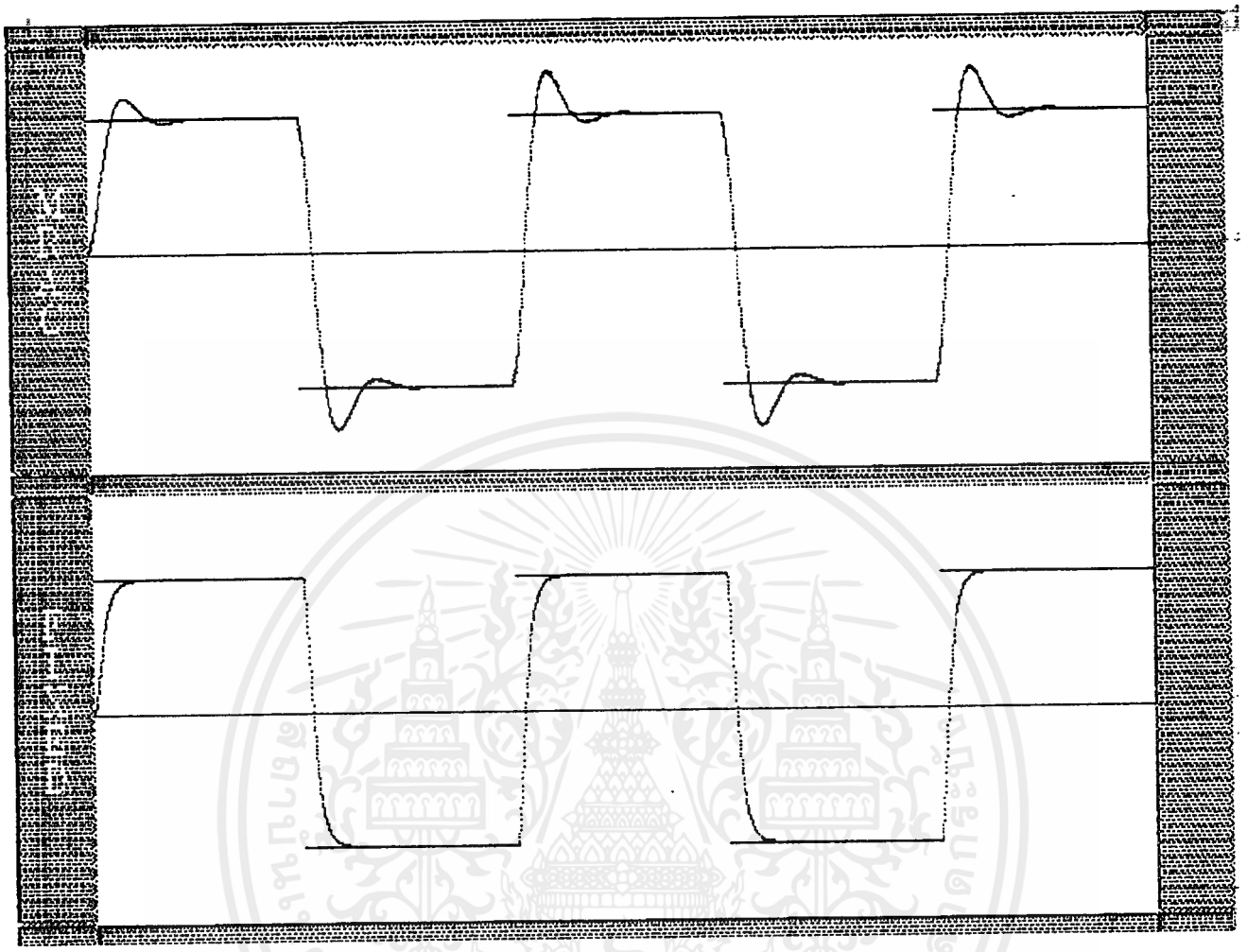
ผลการจำลองพิมพ์ออกมาจากคอมพิวเตอร์ หน้าต่างบนแสดงระบบที่ควบคุมด้วย MRAC หน้าต่างล่างแสดงระบบที่ควบคุมด้วยตัวควบคุมคงที่ แกนนอนเป็นแกนเวลาแกนตั้งเป็นขนาดของผลตอบสนอง หน้าต่างบนมีกราฟสองเส้นเส้นหนึ่งเป็นเส้นประ อีกเส้นหนึ่งเป็นเส้นทึบ เส้นประแสดงรูปแบบอ้างอิงที่ต้องการ เส้นทึบแสดงผลตอบสนองของระบบลูปปิดหากผลตอบสนองของระบบลูปปิดเหมือนรูปแบบอ้างอิงแล้วเส้นทึบจะทับเส้นประทำให้เห็นเป็นเส้นเดียว หน้าต่างล่างมีเส้นทึบเส้นเดียวแสดงผลตอบสนองของระบบลูปปิดเพียงอย่างเดียว อินพุทของระบบลูปปิด หรือ set point ของทั้งสองระบบ เป็น square wave ขนาดเท่ากับหนึ่ง คาบเวลา(ขึ้น ลงหนึ่งรอบ) เท่ากับ 10 วินาที แสดงด้วยเส้นทึบตรง ผลที่นำมาแสดงในที่นี้อาจเห็นไม่ชัดเจนเนื่องจากข้อจำกัดของการพิมพ์ สามารถใช้โปรแกรมในภาคผนวก ก ดูอย่างละเอียดอีกครั้ง โดยเส้นประที่นำมาแสดงในที่นี้จะถูกแทนด้วย เส้นสีแดง และ เส้นทึบแทนด้วยเส้นสีขาว

ต่อมาเปลี่ยนค่า q ของกระบวนการจาก 1 เป็น 2 แสดงผลดังรูป 3.6



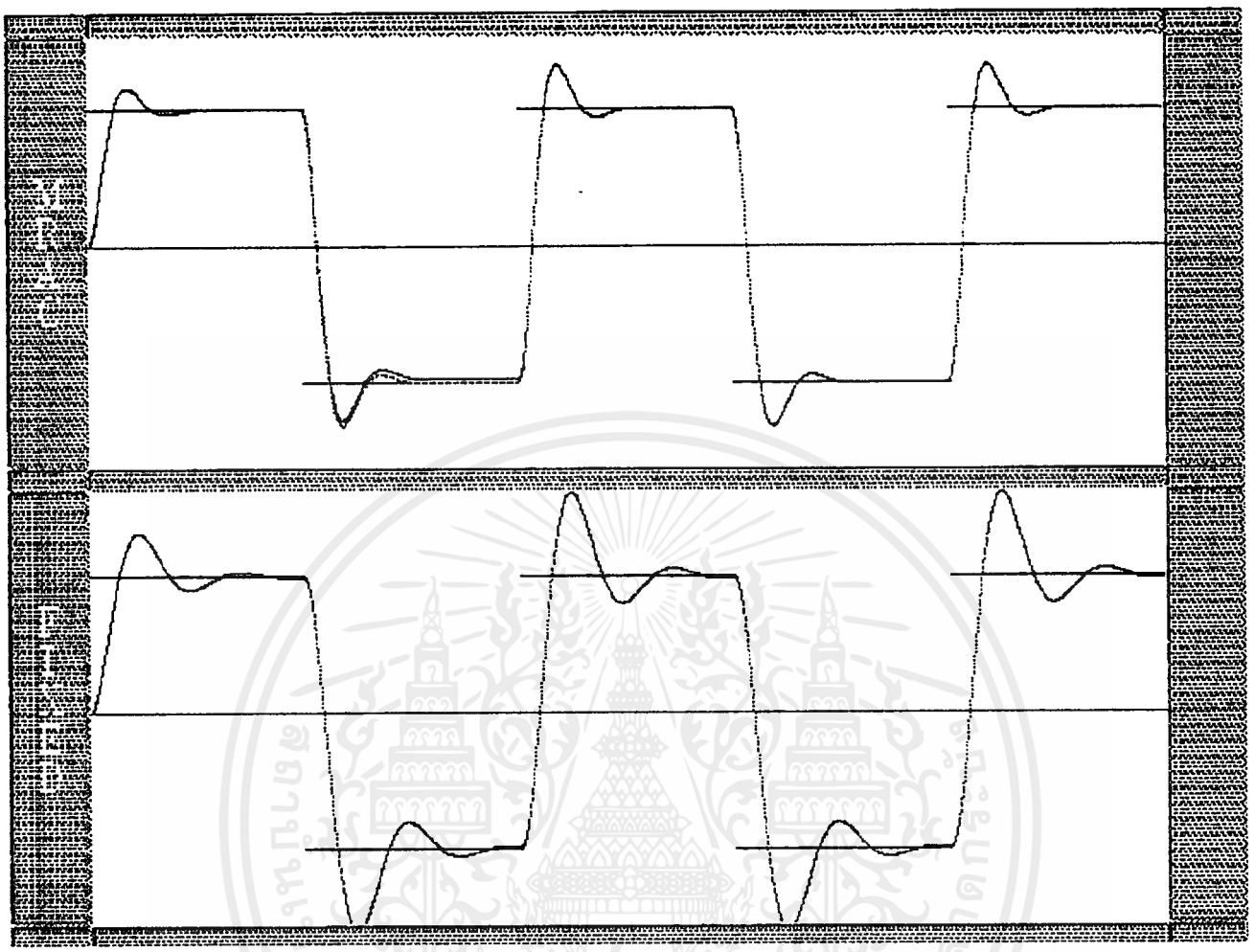
รูปที่ 3.6 ผลตอบสนองเมื่อเปลี่ยน q จาก 1 เป็น 2

เปลี่ยน q ของกระบวนการจาก 1 เป็น 4 ผลแสดงดังรูปที่ 3.7



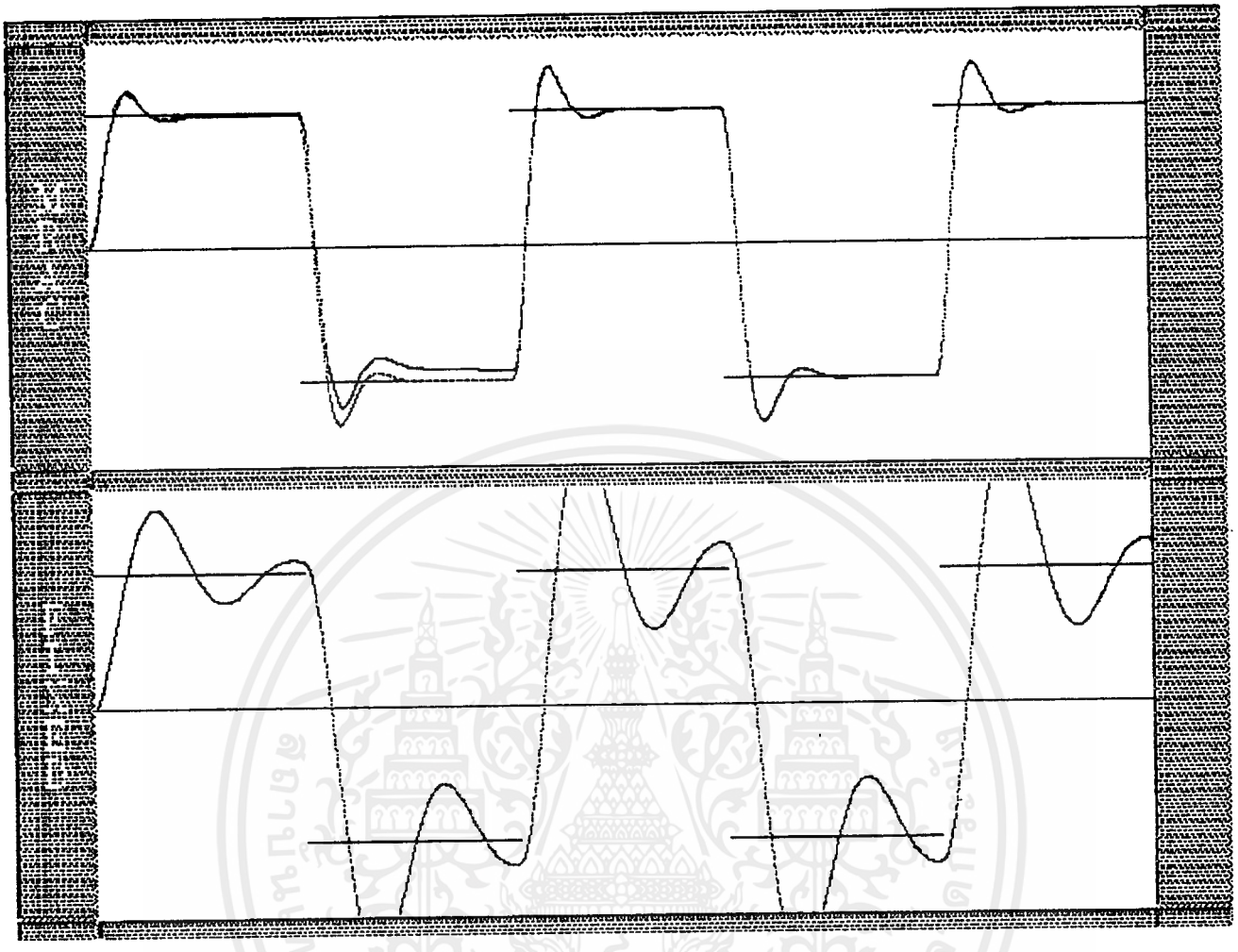
รูปที่ 3.7 ผลตอบสนองเมื่อเปลี่ยน q จาก 1 เป็น 4

เปลี่ยน q ของกระบวนการจาก 1 เป็น 0.5 ผลแสดงดังรูปที่ 3.8



รูปที่ 3.8 ผลตอบสนองเมื่อเปลี่ยน q จาก 1 เป็น 0.5

เปลี่ยน q ของกระบวนการจาก 1 เป็น 0.25 ผลแสดงดังรูปที่ 3.9

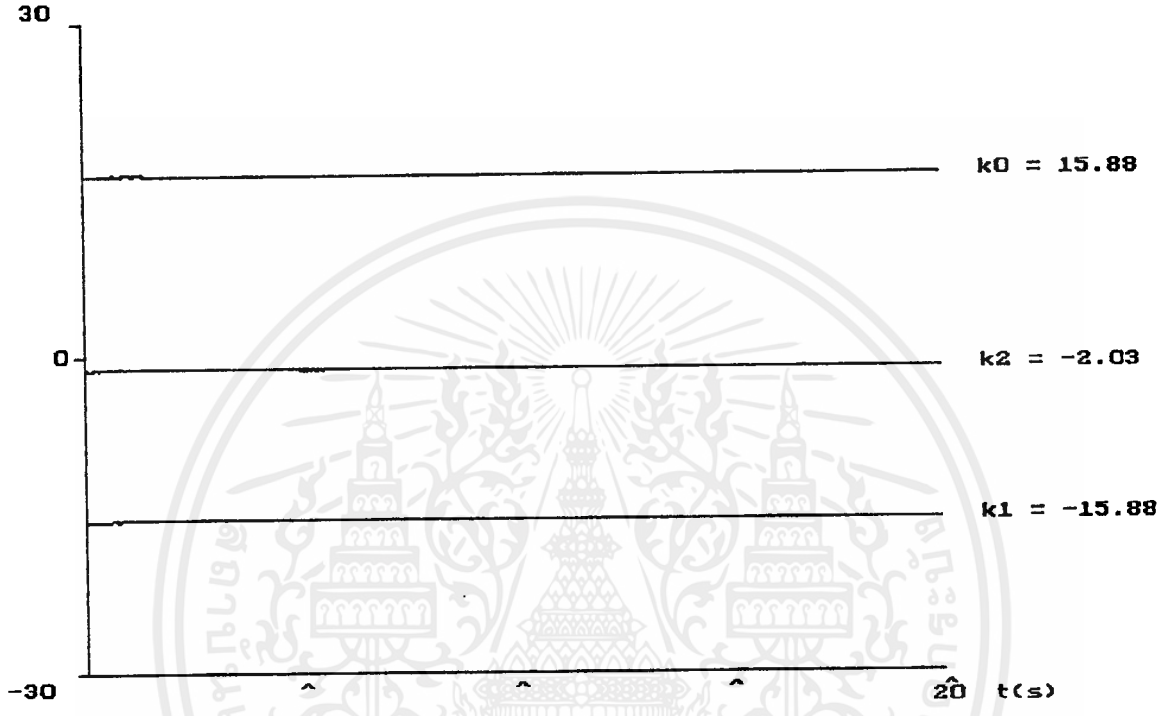


รูปที่ 3.9 ผลตอบสนองเมื่อเปลี่ยน q จาก 1 เป็น 0.25

จากผลการจำลองจะเห็นว่าเมื่อตัวกระบวนการมีการเปลี่ยนแปลงไป ในตัวควบคุมคงที่ซึ่งได้ออกแบบพารามิเตอร์ไว้ตายตัวจะไม่สามารถเปลี่ยนแปลงตัวเองได้ทำให้ตำแหน่งของโพลลูปปิดย้ายไปจากตำแหน่งที่เราต้องการ ซึ่งทำให้การตอบสนองของระบบเปลี่ยนแปลงไป สำหรับตัวควบคุม MRAC แม้ว่ากระบวนการจะมีการเปลี่ยนแปลงไป ตัวควบคุมก็ยังสามารถเปลี่ยนแปลงตัวเองให้ระบบลูปปิดยังคงมีโพลอยู่ที่ตำแหน่ง $-2 \pm 3.464j$ ได้ ซึ่งทำให้การตอบสนองของระบบลูปปิดยังคงเป็นไปตามที่ต้องการ

การที่ MRAC ยังคงสามารถควบคุมให้ได้สมรรถภาพตามที่ต้องการได้ ในขณะที่ตัวควบคุมคงที่ไม่สามารถทำได้ เนื่องจากการที่ตัวควบคุม MRAC สามารถเปลี่ยนแปลงค่า k_i ได้โดยอาศัยกฎการปรับตัวตามสมการ (3.16) ในขณะที่ตัวควบคุมคงที่ไม่สามารถเปลี่ยนแปลงค่า k_i ได้ ดังจะได้มีการแสดงการเปลี่ยนแปลงของค่า k_i ดังต่อไปนี้ (สอดคล้องกับรูปที่ 3.5 3.7 และ 3.9)

เมื่อ q = 1

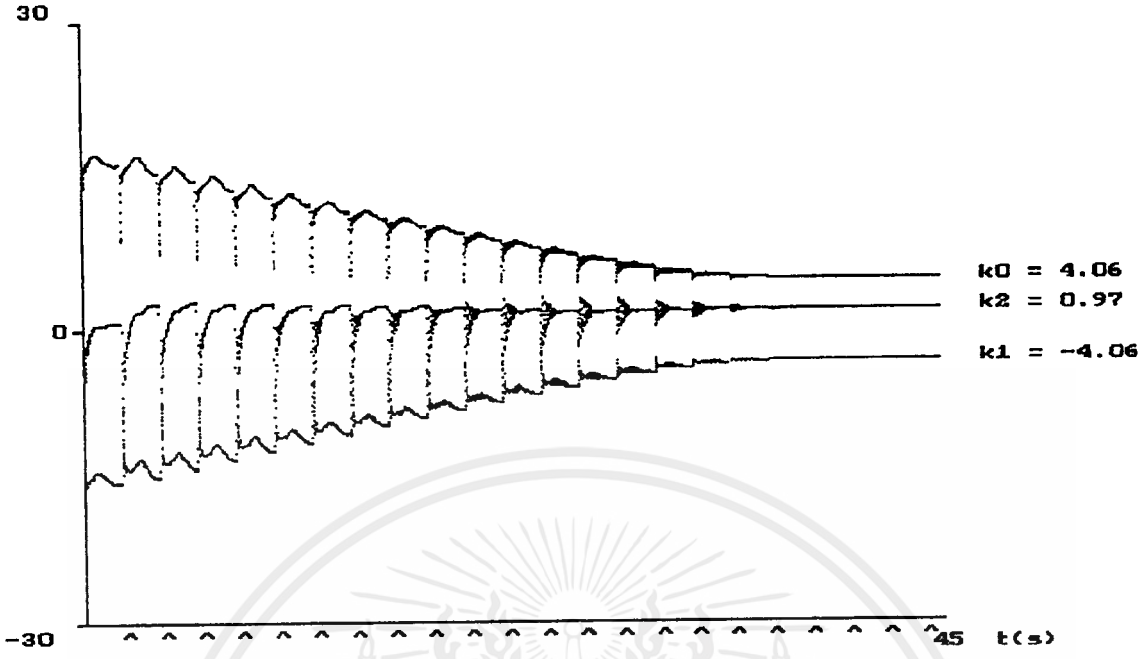


รูปที่ 3.10 ค่า k_i ของตัวควบคุม MRAC เมื่อ q มีค่าคงที่เท่ากับ 1

ในรูปแกนตั้งเป็นค่า k_i แกนนอนเป็นเวลา จะพบว่าค่า k_i ของ MRAC จะไม่มีการเปลี่ยนแปลงเนื่องจากตอนเริ่มต้นได้มีการตั้งค่า k ที่เหมาะสมไว้ แล้วและจะเห็นว่าถ้าค่า k จะไม่หนีจากค่าที่เหมาะสมนั้น ในรูปที่แกนเวลาตรงที่มีหัวลูกศรเป็นจุดที่เปลี่ยนแปลง set point คาบของหัวลูกศรเป็นครึ่งหนึ่งของ square wave ในรูปที่ 3.5 ถึง 3.9 คือช่วงห่างของหัวลูกศรเป็นเวลาเท่ากับ 5 วินาที

โปรแกรมแสดงค่า k ไม่ได้เสนอไว้ในภาคผนวก แต่สามารถนำโปรแกรมในภาคผนวก ก มาดัดแปลงได้

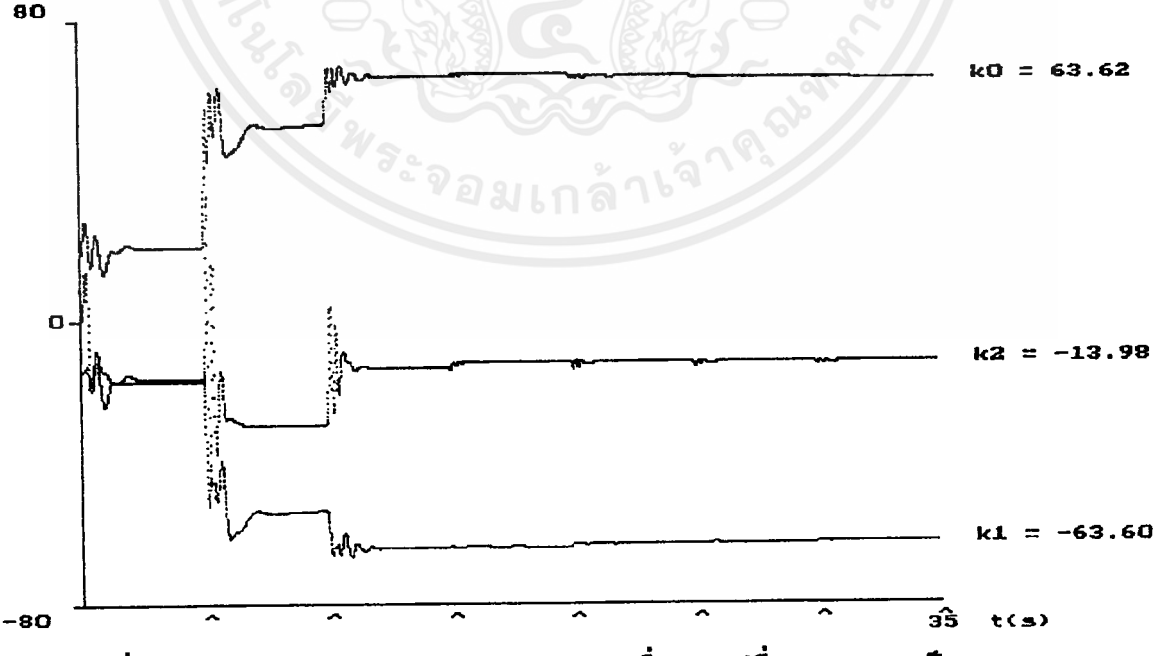
เมื่อ q เปลี่ยนจาก 1 เป็น 4



รูปที่ 3.11 ค่า k_i ของตัวควบคุม MRAC เมื่อ q เปลี่ยนจาก 1 เป็น 4

จะพบว่าค่า k_i ของ MRAC จะมีการเปลี่ยนแปลงจากค่าเริ่มต้น เข้าสู่ค่า k ที่เหมาะสมกับกระบวนการที่มีค่า q = 4 โดยการทำงานของกฎการปรับตัวตามสมการ (3.16) (คาบของหัวลูกศรมีค่าเท่ากับ 2 วินาทีซึ่งต่างจากรูปที่ 3.7 ซึ่งเป็นรูปที่สอดคล้องกัน การเปลี่ยนคาบของหัวลูกศรที่ควรเป็น 5 วินาทีเช่นเดียวกับรูป 3.7 มาเป็น 2 วินาทีเนื่องจากค่า k เปลี่ยนช้าเกินไป จึงเลือกคาบการเปลี่ยนระดับสัญญาณให้เร็วขึ้น)

เมื่อ q เปลี่ยนจาก 1 เป็น 0.25



รูปที่ 3.12 ค่า k_i ของตัวควบคุม MRAC เมื่อ q เปลี่ยนจาก 1 เป็น 0.25

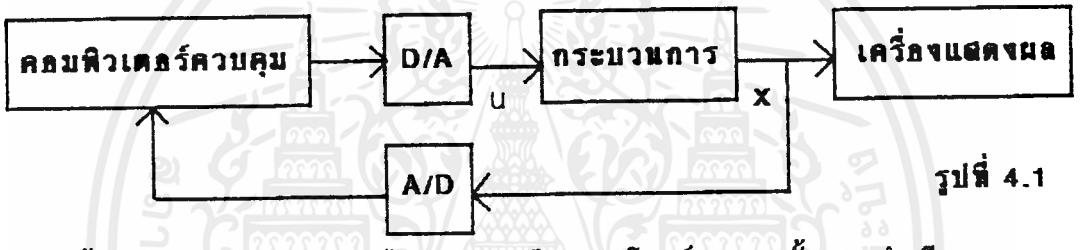
จะพบว่าค่า k_i ของ MRAC จะมีการเปลี่ยนแปลงจากค่าเริ่มต้น เข้าสู่ค่า k ที่เหมาะสมกับกระบวนการที่มีค่า q = 0.25 โดยการทำงานของกฎการปรับตัวตามสมการ (3.16)

บทที่ 4
การควบคุมระบบจริงด้วย MRAC

ในบทที่แล้วได้แสดงการจำลองการควบคุมระบบด้วยคอมพิวเตอร์ ในวิธี MRAC ซึ่งเป็นการควบคุมภายใต้เงื่อนไขที่ทุกองค์ประกอบของระบบเป็นอุดมคติตามทฤษฎี สำหรับในบทที่ 4 นี้เป็นการแสดงการทดลองการควบคุมระบบจริงด้วยวิธี MRAC จุดประสงค์ก็เพื่อศึกษาผลของการควบคุมในทางปฏิบัติ ภายใต้ องค์ประกอบที่ไม่เป็นอุดมคติซึ่งจะแสดงถึงความมั่นคงของทฤษฎีต่อสภาพแวดล้อม (robustness) ในการทดลองได้มีการทดลองเปรียบเทียบกับการควบคุมคงที่เช่นเดียวกับบทที่แล้วด้วย

4.1 องค์ประกอบในการทดลอง

องค์ประกอบในการทดลองแสดงได้ดังรูปที่ 4.1



รูปที่ 4.1

สัญญาณอนาลอกทั้งหมดในระบบอยู่ในช่วงไม่เกิน 12 ถึง -12 โวลท์ ระบบทั้งหมดทำเลียนแบบระบบที่ได้จำลอง ในบทที่ 3

คอมพิวเตอร์ควบคุม

การควบคุมกับระบบจริงเราใช้คอมพิวเตอร์ทำหน้าที่เป็นตัวควบคุม การควบคุมจึงเป็นแบบไม่ต่อเนื่อง แต่เราใช้ความถี่ในการควบคุมสูงพอสมควรเมื่อเทียบกับความถี่ของระบบดังนั้นจึงนำระเบียบวิธีของแบบต่อเนื่องที่กล่าวในบทที่แล้วมาใช้

การควบคุมใช้อินเทอร์รัพของ timer ของคอมพิวเตอร์ (interrupt 1C) มาใช้ซึ่งจะทริกด้วยความถี่ 18.2044 Hz หมายความว่าคาบเวลาในการสุ่มสัญญาณจะมีค่าประมาณ 0.055 วินาที และช่วงเวลาในการคำนวณค่าสัญญาณควบคุมตามวิธี MRAC มีค่าประมาณ 1/94 เท่าของช่วงการสุ่มสัญญาณ หรือ ประมาณ 1 เปอร์เซ็นต์ของช่วงการสุ่มสัญญาณ (ใช้เครื่องคอมพิวเตอร์ IBM compatible, 80386, math co-processor) โปรแกรมที่ใช้ควบคุมดูในภาคผนวก ข

A/D และ D/A

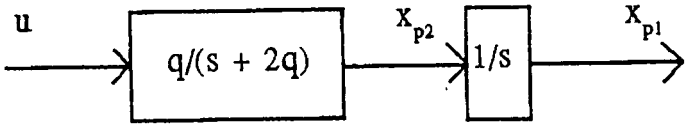
เนื่องจากตัวควบคุมใช้คอมพิวเตอร์ดังนั้นจึงต้องมีการแปลงสัญญาณเข้าซึ่งเป็นเสตท x และ x' จากอนาลอกให้กลายเป็นดิจิตอล (A/D) และแปลงสัญญาณออกซึ่งเป็นสัญญาณควบคุม u (ที่ได้จากการคำนวณด้วยคอมพิวเตอร์) จากดิจิตอลให้เป็นอนาลอก (D/A)

ตัวแปลงสัญญาณ A/D ใช้การ์ด PCL 726, Lab card ส่วนตัวแปลงสัญญาณ D/A ใช้การ์ด PCL 812, Lab card สัญญาณทางอนาลอกอยู่ในช่วงไม่เกิน ± 10 โวลท์

กระบวนการ

กระบวนการจริงที่ใช้ควบคุมได้แก่ process simulator PTS-10 ของบริษัท Pacific ซึ่งเป็นกระบวนการอิเลคทรอนิคที่ไว้ทดลองระบบเชิงเส้นสามารถปรับทรานเฟอร์ฟังก์ชันและพารามิเตอร์ได้

ทรานเฟอร์ฟังก์ชันของกระบวนการเป็นเช่นเดียวกับที่ได้จำลองในบทที่ 3 อันได้แก่สมการ 3.1 แต่ได้แปลงรูปให้สามารถใช้กับ PTS-10 ได้ดังบล็อกไดอะแกรมข้างล่าง



สัญญาณเข้าออก และ สัญญาณในกระบวนการเป็นสัญญาณไฟฟ้าอยู่ในช่วงไม่เกิน ± 12 โวลท์

หมายเหตุ PTS-10 คุณสมบัติใกล้เคียงกับอุดมคติมาก คือ เป็นเชิงเส้นและมีสัญญาณรบกวนน้อยมาก

เครื่องวัดแสดงผล

เป็นเครื่อง Servocorder SR 6221 ของบริษัท Graphtec ซึ่งใช้วัดสัญญาณไฟฟ้าและเขียนออกมาด้วยปากกา ดังนั้นผลที่แสดงในบทนี้จะมาจากเครื่องวัดนี้ โดยสัญญาณแต่ละเส้นจะเป็นสัญญาณที่มีอัตราส่วนเดียวกันซึ่งอยู่ระหว่าง ±10 โวลท์

ผลการทดลองของกระบวนการจริงที่นำมาแสดงในที่นี้เป็นผลที่ได้จากการบันทึกของ Servocorder กระดาษกราฟมีอัตราส่วนในแนวตั้ง 42 ช่องต่อ 10 โวลต์ หรือประมาณ 1ช่องต่อ 0.25 โวลต์ สำหรับอัตราส่วนในแนวนอนประมาณ 1 ช่องต่อ 2 วินาที

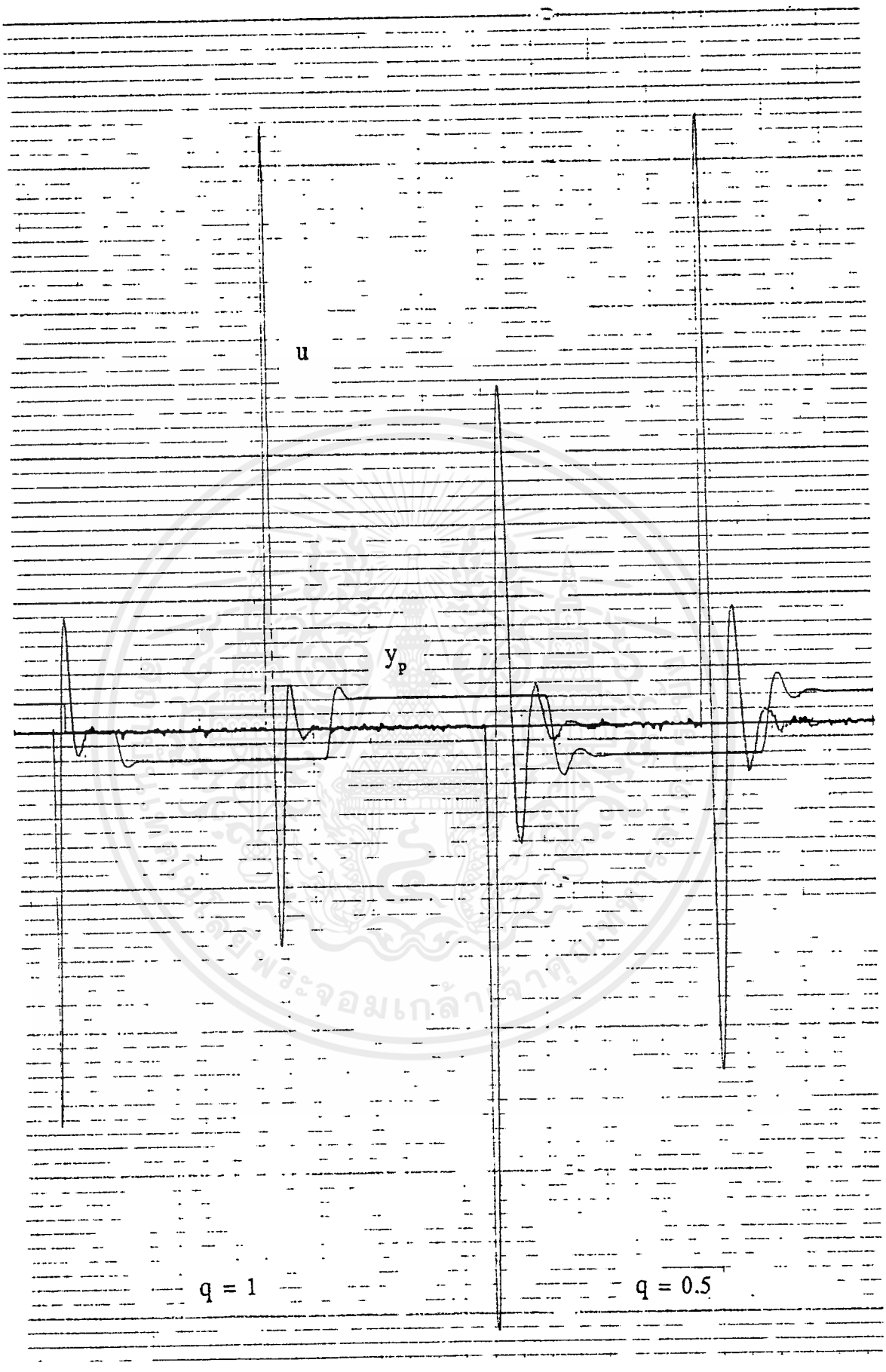
รูปแบบของสัญญาณอ้างอิง (set point) ที่ใช้ทดลอง

ดังได้กล่าวไว้แล้วว่า การทดลองกับกระบวนจริงเลียนแบบลักษณะทุกอย่างจากการจำลองด้วยคอมพิวเตอร์ในบทที่ 3 ดังนั้นสัญญาณอ้างอิงหรือ set point จึงเป็น square wave เช่นเดียวกันแต่มีส่วนที่ต่างกันคือคาบของ set point ที่เป็น square wave เปลี่ยนจาก 10 วินาทีเป็น 30 วินาที สำหรับแอมปลิจูดของ set point จะเปลี่ยนไปตามการทดลองในแต่ละขั้น

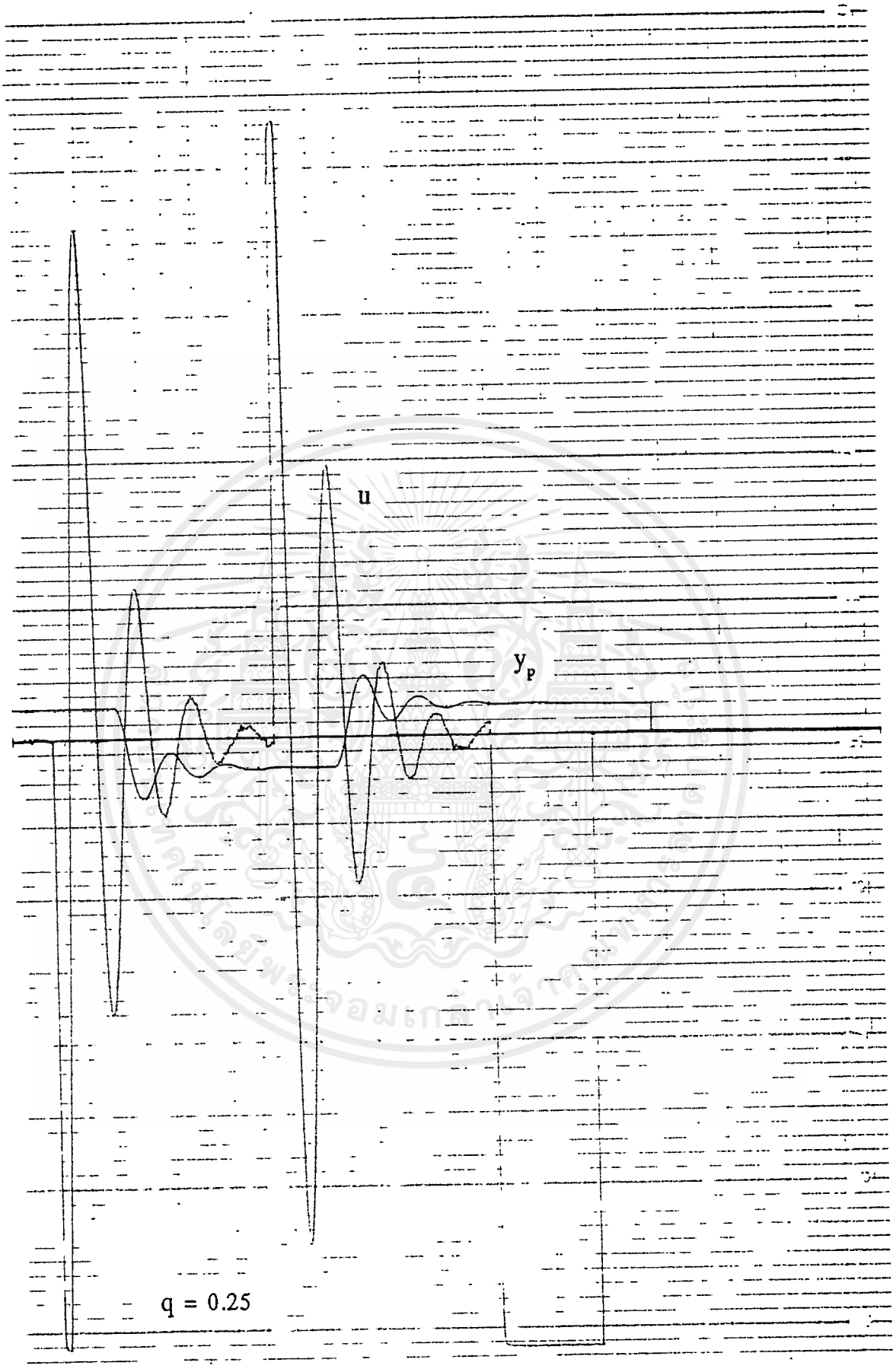
4.2 วิเคราะห์ผลการควบคุมระบบจริงโดยใช้ตัวควบคุมคงที่ หรือ ตัวควบคุมสเตทปีอนกลับ

ผลที่ได้เป็นเช่นเดียวกับที่ได้จากการจำลองด้วยคอมพิวเตอร์ แสดงดังรูปที่ 4.2

จากรูปที่ 4.2 จะพบว่าการใช้ตัวควบคุมคงที่ จะสามารถควบคุมระบบได้ตามต้องการเมื่อกระบวนการยังไม่มีการเปลี่ยนแปลง แต่เมื่อระบบมีการเปลี่ยนแปลง โดยการเปลี่ยนแปลงค่า q จะพบว่าโพลของลูปปิดมีการเปลี่ยนตำแหน่ง ทำให้พฤติกรรมของระบบเกิดการแกว่งมากขึ้น



รูปที่ 4.2 ผลการทดลองกับระบบจริงเมื่อตัวควบคุมเป็นการป้อนกลับสเตท



รูปที่ 4.2 ผลการทดลองกับระบบจริงเมื่อตัวควบคุมเป็นการป้อนกลับสเตท

4.3 การวิเคราะห์การควบคุมระบบจริงโดยใช้ตัวควบคุม MRAC

เมื่อทำการติดตั้งระบบครบถ้วนได้มีการบันทึกผลการทดลองต่าง ๆ ดังนี้

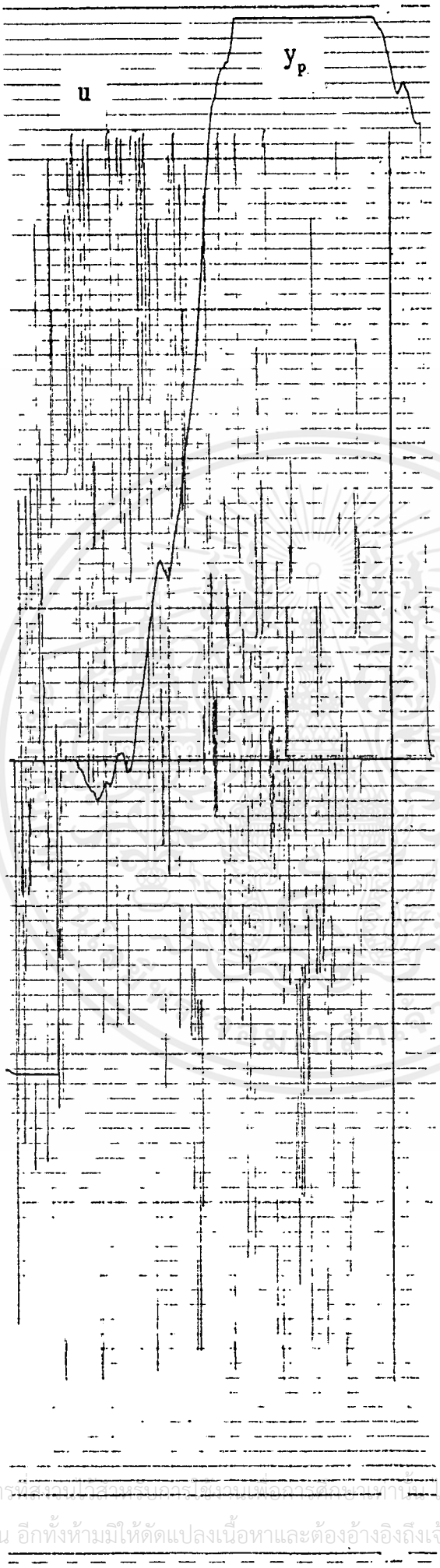
ขั้นที่ 1

เมื่อตั้ง set point เท่ากับ 5.0 โวลต์ และ γ ทุกตัวเท่ากับ 1 (ในสมการ 16) พบว่าได้ผลดังรูปที่ 4.3

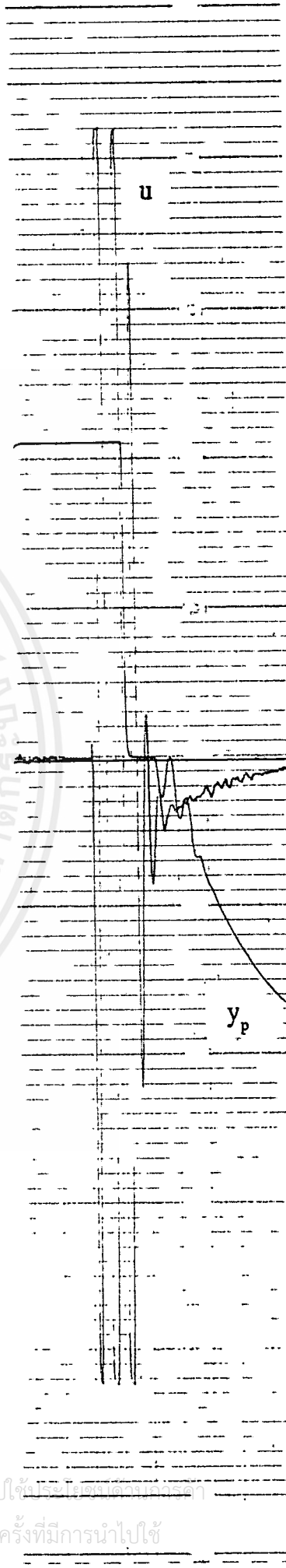
และ เมื่อตั้ง set point เท่ากับ 5.0 โวลต์ และ ลดค่า γ ทุกตัวให้เป็น 0.1 ผลที่ได้เป็นดังรูปที่ 4.4

วิเคราะห์ผลการทดลองขั้นที่ 1

จากผลทั้งสองจะเห็นว่าเกิดการแกว่งของสัญญาณขึ้น ซึ่งเป็นผลจากการที่การปรับตัวเร็วเกินไป และ เนื่องจากมีการตั้ง set point ที่สูงจนสัญญาณ u ที่ออกจากคอมพิวเตอร์ไปยังระบบเกิดการอิ่มตัว (สัญญาณ u จะต้องมีค่าอยู่ภายใน $\pm$ 10 โวลต์)



รูปที่ 4.3 ผลการทดลองกับระบบจริงเมื่อใช้ตัวควบคุม MRAC , sp = 5.0, γ = 1



รูปที่ 4.4 ผลการทดลองกับระบบจริงเมื่อใช้ตัวควบคุม MRAC , sp = 5.0, γ = 0.1

ขั้นที่ 2

ตั้ง set point เท่ากับ 0.5 โวลต์ และ γ เท่ากับ 1 เพื่อลดผลที่เกิดจากการอิ่มตัวของ u จะได้ผลว่าระบบสามารถทำงานได้ดังรูปที่ 4.5

จากผลการทดลองพบว่า MRAC สามารถทำงานได้ แต่จะเห็นว่าเกิดผลคือค่า k จะมีการเปลี่ยนแปลงสูงขึ้นเรื่อย ๆ ซึ่งทำให้พฤติกรรมของเอาท์พุทมีการเปลี่ยนแปลง ไม่ได้เข้าสู่รูปแบบอ้างอิง

และถ้าปล่อยให้ระบบดำเนินการไปได้ช่วงเวลาหนึ่งจะทำให้ค่า k มีค่าเพิ่มขึ้นจนมีค่าสูงมาก ซึ่งแสดงผลดังรูปที่ 4.6

ซึ่งถ้าค่า k มีค่าสูงมากเกินไปจะทำให้ระบบมีการแกว่งของสัญญาณ

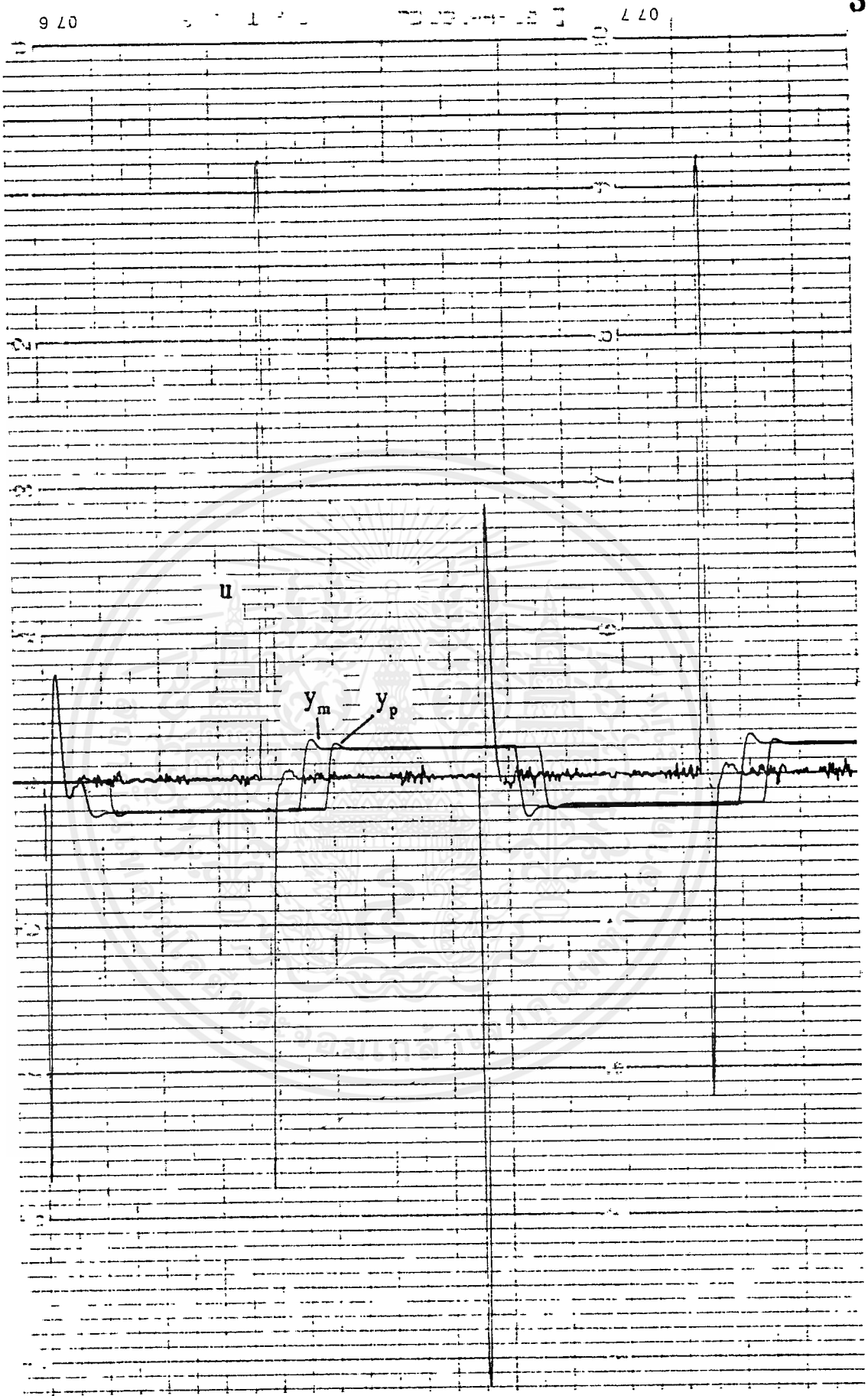
ในขั้นนี้แม้ว่าพฤติกรรมของระบบจะไม่เข้าสู่รูปแบบอ้างอิงแต่เมื่อมีการเปลี่ยนแปลงค่า q ของกระบวนการจะ ได้ผลดังรูปที่ 4.7

ผลที่เกิดขึ้นจะเห็นว่า ผลตอบสนองของระบบก็ยังดีกว่า ผลตอบสนองของตัวควบคุมคงที่ (เปรียบเทียบรูป 4.2 และ 4.7 ซึ่งเป็นการควบคุมภายใต้เงื่อนไขเดียวกัน)

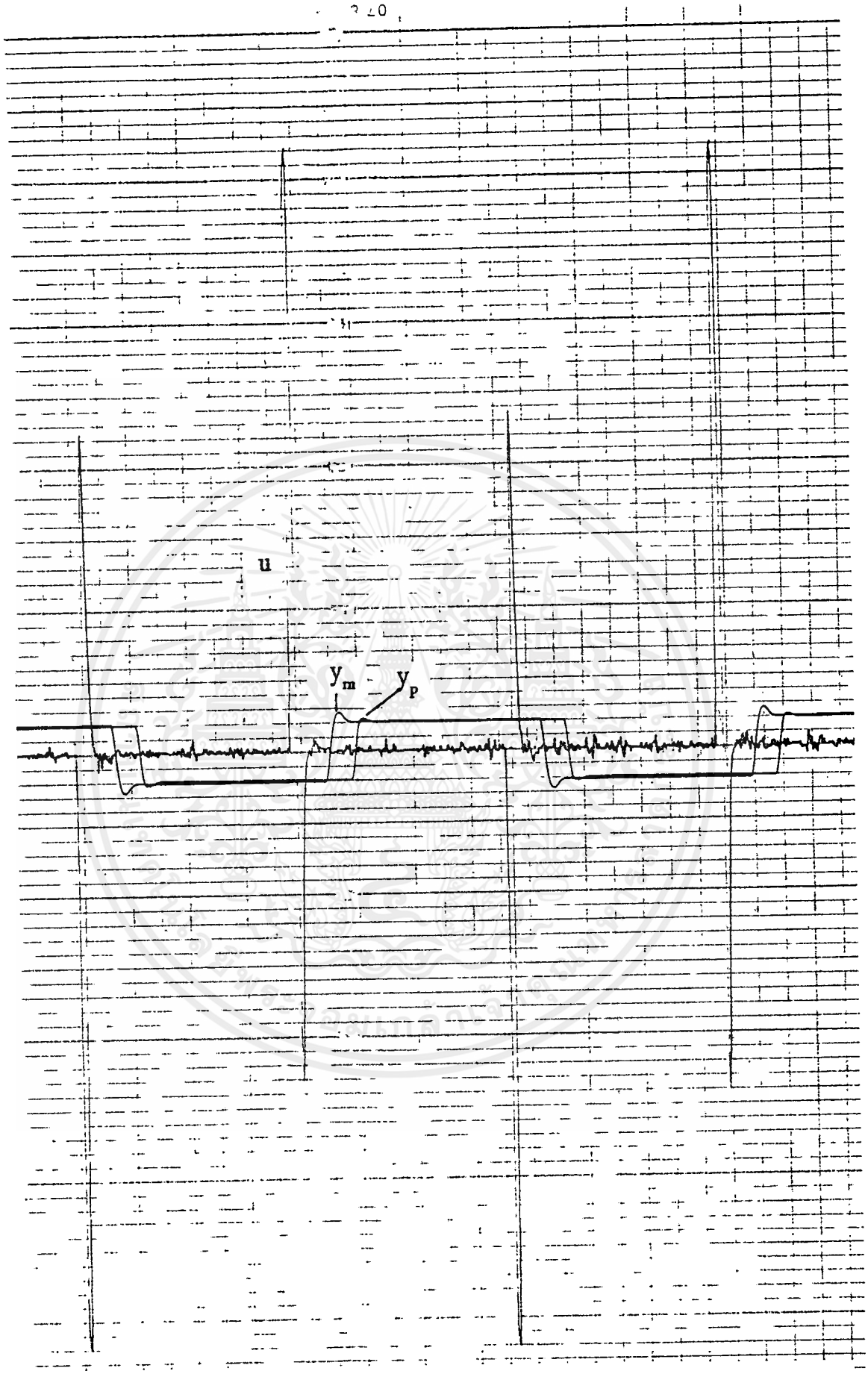
การที่ผลตอบสนองของกระบวนการจริงไม่เป็นไปตามทฤษฎีนั้นสามารถแยกสาเหตุที่สำคัญได้ดังนี้

- สัญญาณมีการอิ่มตัว
- ระบบโดยรวมอาจไม่ใช่อันดับสองจริง ถึงแม้ว่า PTS-10 จะมีคุณสมบัติเป็นอุดมคติ แต่อุปกรณ์อินเตอร์เฟสอาจทำให้เกิดโพลอื่นที่อยู่ไกลซึ่งมองไม่เห็นอย่างชัดเจน
- อาจมีสัญญาณรบกวน

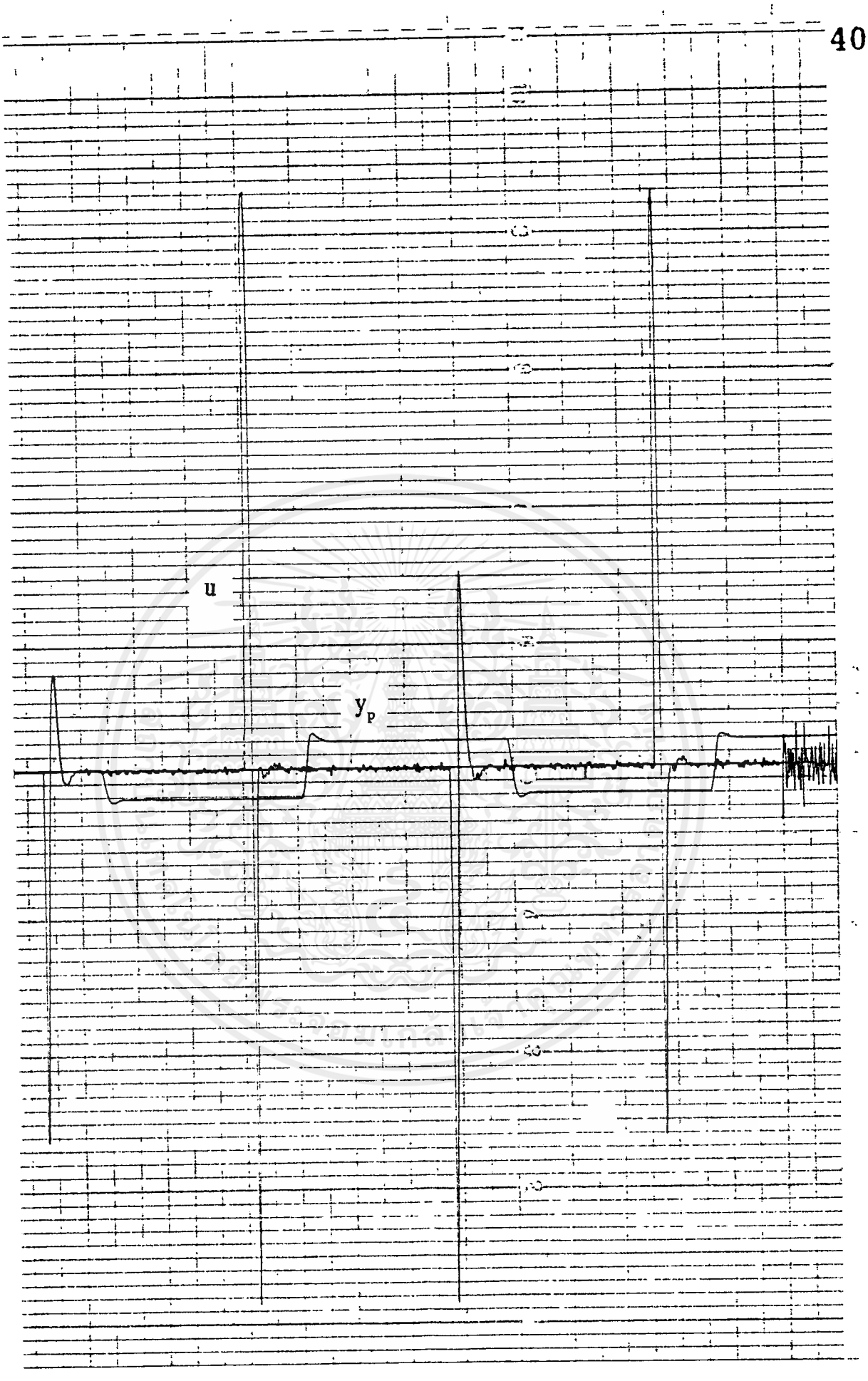
เนื่องจาก PTS-10 มีคุณสมบัติใกล้เคียงกับอุดมคติดังนั้นสัญญาณรบกวนควรจะมีผลน้อยในที่นี้ (แต่ในสถานการณ์อื่นอาจมีความสำคัญอย่างยิ่งยวด) ดังนั้นในหัวข้อต่อไปจะใช้สาเหตุที่สมมติฐานไว้ในสองข้อแรกเป็นพื้นฐาน และใช้โปรแกรมจำลองระบบในบทที่ 3 หรือในภาคผนวก ก เป็นเครื่องมือในการศึกษาถึงผลตอบสนองของระบบควบคุมต่อสถานการณ์ทั้งสอง



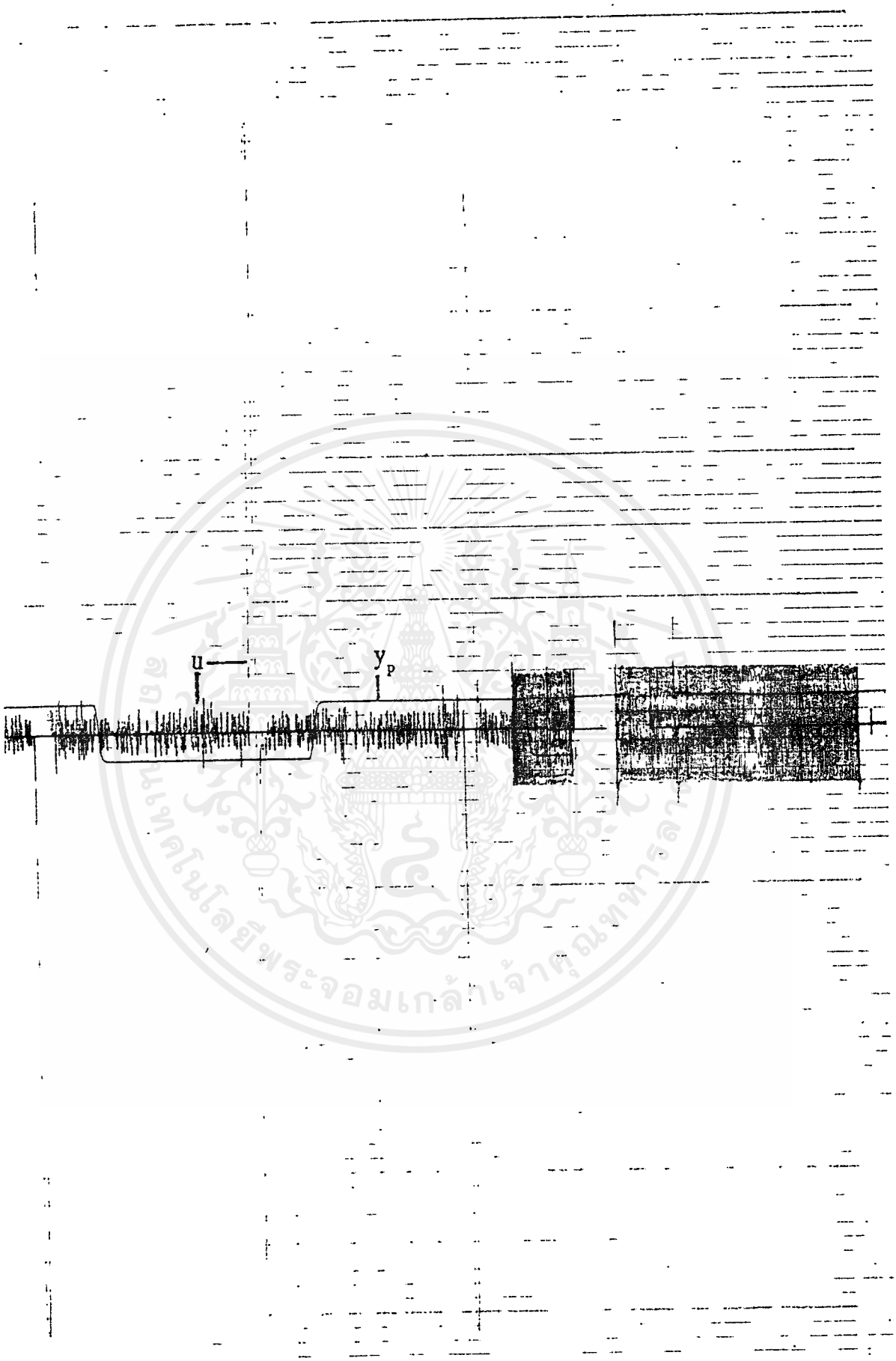
รูปที่ 4.5 ผลการทดลองกับระบบจริงเมื่อใช้ตัวควบคุม MRAC , sp = 0.5, γ = 1



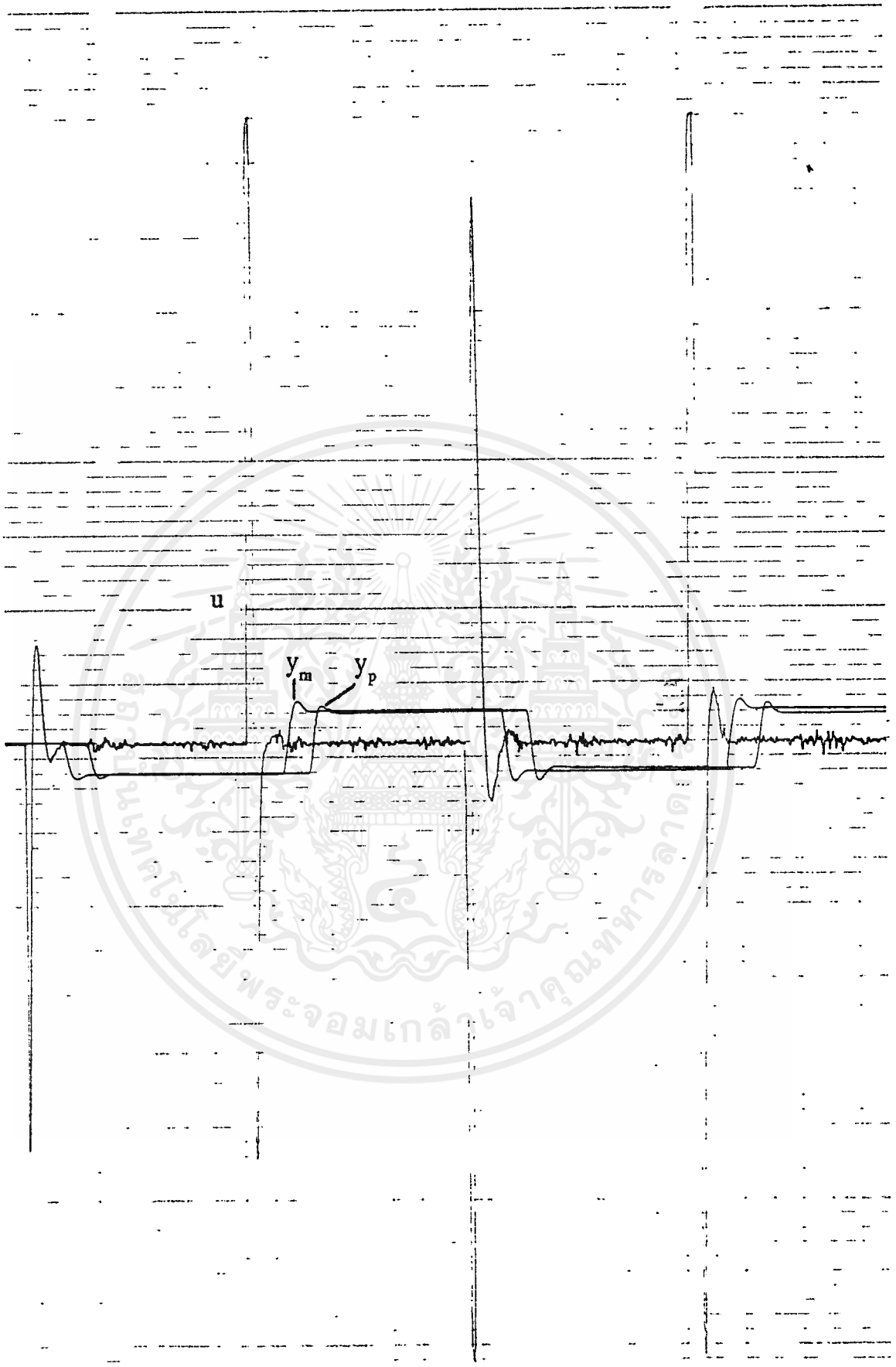
รูปที่ 4.5 ผลการทดลองกับระบบจริงเมื่อใช้ตัวควบคุม MRAC , sp = 0.5, γ = 1



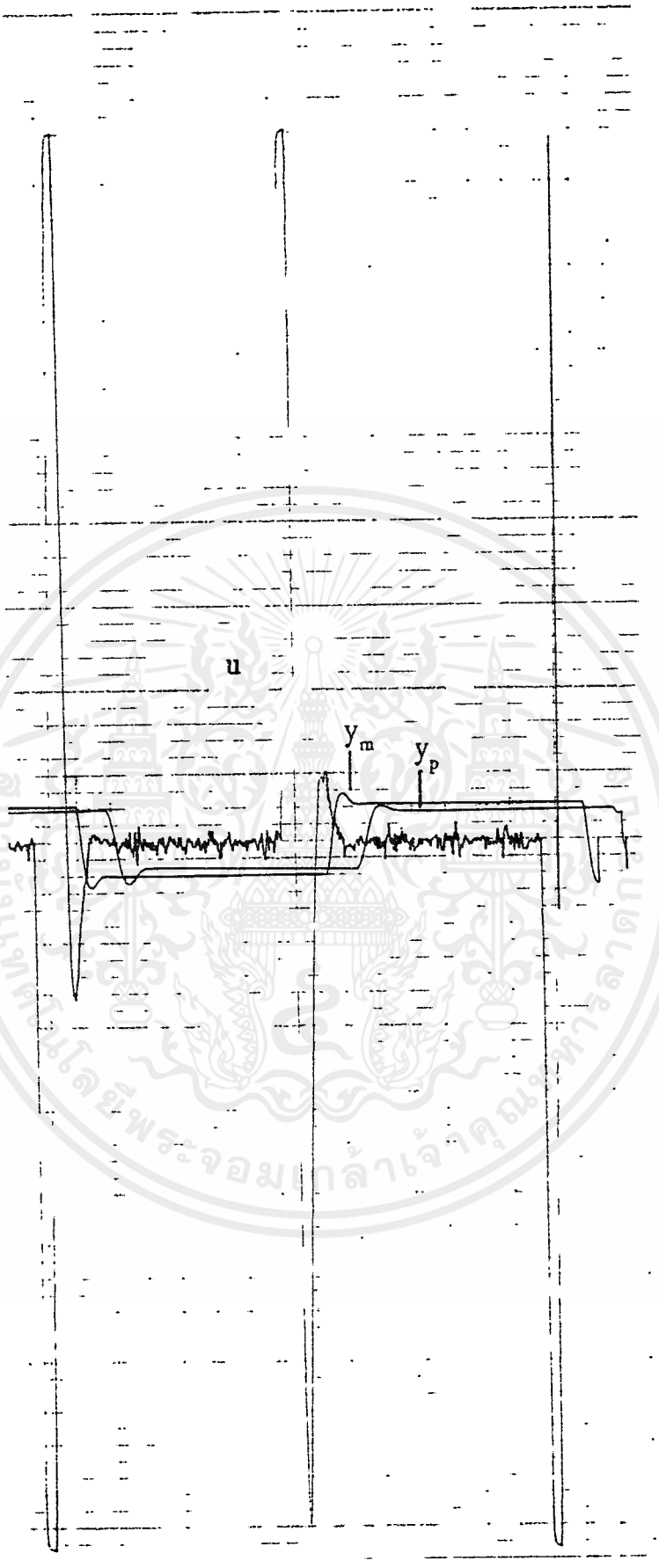
รูปที่ 4.6 ผลการทดลองกับระบบจริงเมื่อใช้ตัวควบคุม MRAC , sp = 0.5, γ = 1 และเวลาผ่านไปนาน



รูปที่ 4.6 ผลการทดลองกับระบบจริงเมื่อใช้ตัวควบคุม MRAC , sp = 0.5, γ = 1 และเวลาผ่านไปนาน



รูปที่ 4.7 ผลการทดลองกับระบบจริงเมื่อใช้ตัวควบคุม MRAC , sp = 0.5, γ = 1 และเปลี่ยนค่า q
q = 1 และ q = 0.5



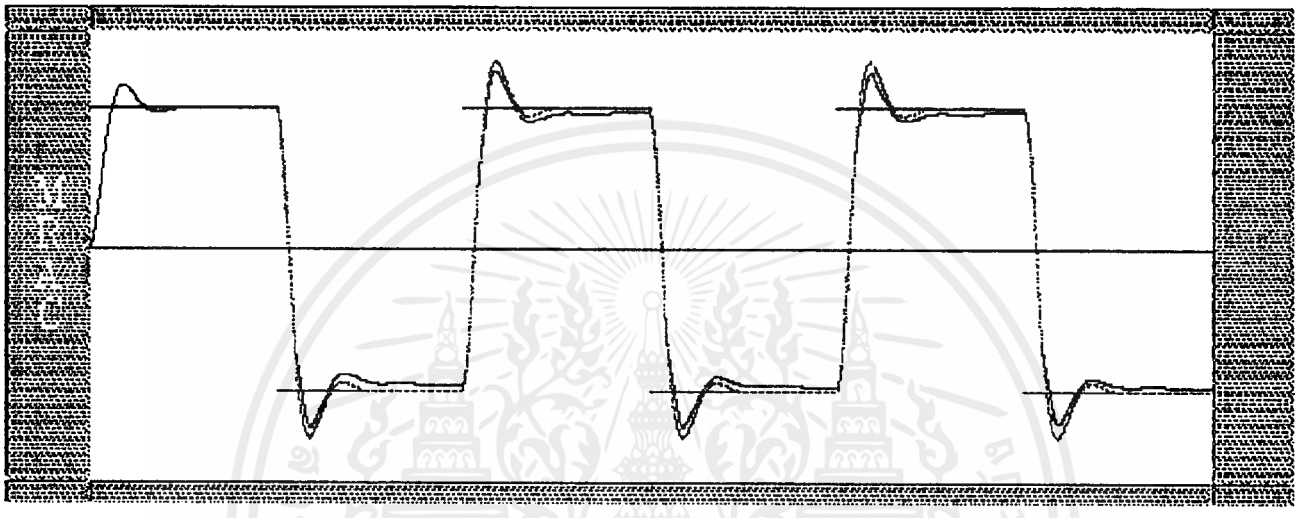
รูปที่ 4.7 ผลการทดลองกับระบบจริงเมื่อใช้ตัวควบคุม MRAC , sp = 0.5, γ = 1 และเปลี่ยนค่า q
q = 0.25

วิเคราะห์ผลการทดลองขั้นที่ 2

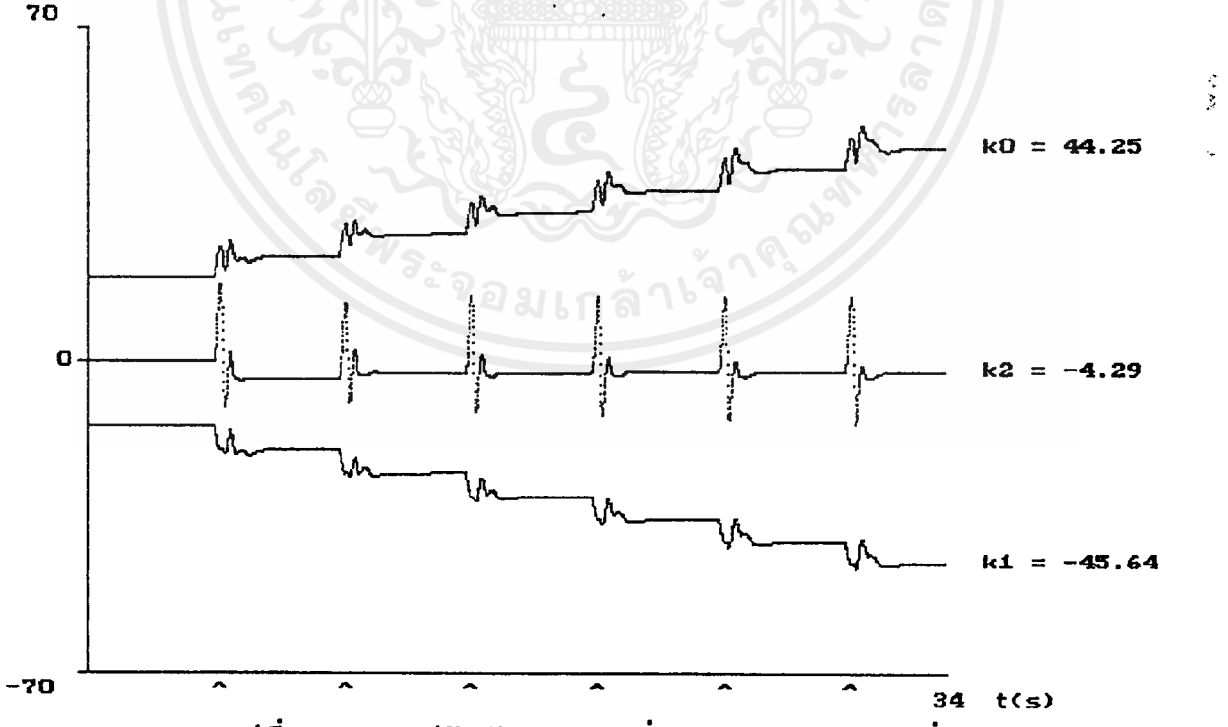
ในหัวข้อนี้จะทำการค้นหาสาเหตุที่ทำให้ค่า k มีค่าสูงขึ้นเรื่อย ๆ ไม่เข้าสู่จุดสมดุลย์ โดยการจำลองสถานะการด้วยคอมพิวเตอร์

2.1 ทำการศึกษาผลที่เกิดจากการอิ่มตัว (saturation) ของสัญญาณควบคุม u

ได้ทำการใช้โปรแกรมในภาคผนวก ก มาดัดแปลง โดยทำการกำหนดของเขตของสัญญาณควบคุมให้มีของเขตอยู่ในช่วง ±10 โวลต์ เพื่อให้เกิดการอิ่มตัวขึ้น ซึ่งได้ผลดังรูปที่4.8 และ 4.9



รูปที่ 4.8 ผอตอบสนองของ MRAS เมื่อมีเงื่อนไขของการอิ่มตัวของ u



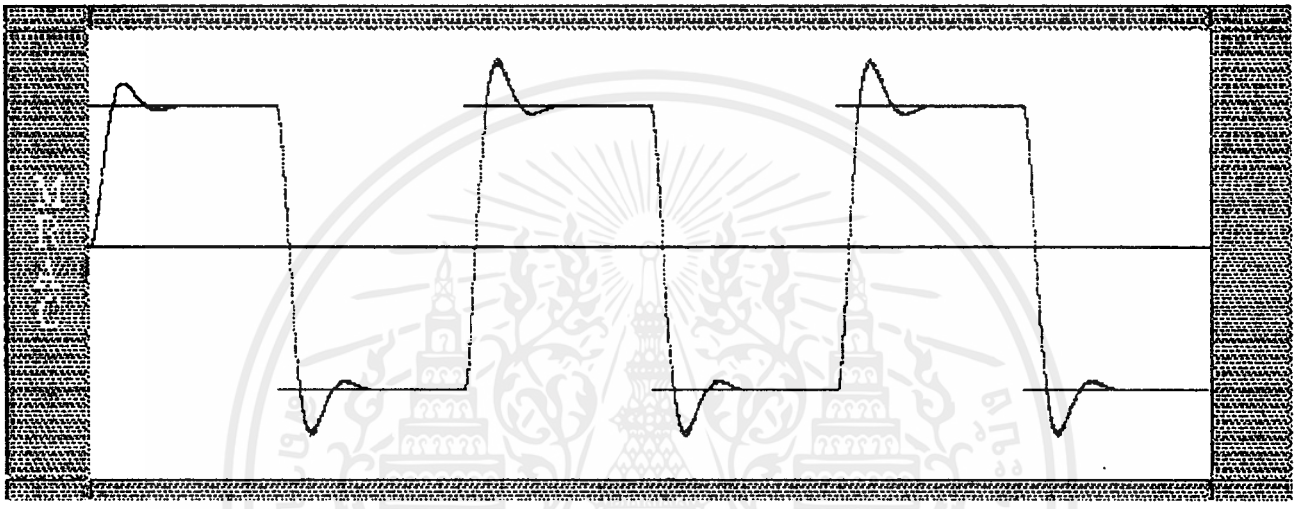
รูปที่ 4.9 การปรับตัวของ k ที่สอดคล้องกับผลรูปที่ 4.8

ผลการจำลองมีลักษณะเช่นเดียวกับในบทที่สาม ผลตอบสนองเส้นประแสดงรูปแบบอ้างอิง เส้นทึบแสดงผลตอบสนองของกระบวนการที่ถูกควบคุมแล้ว เส้นทึบตรงเป็น set point แต่ขนาดเปลี่ยนเป็น 0.5 และคาบเปลี่ยนเป็น 30 วินาที เพื่อให้สอดคล้องกับการทดลองกับกระบวนการจริง

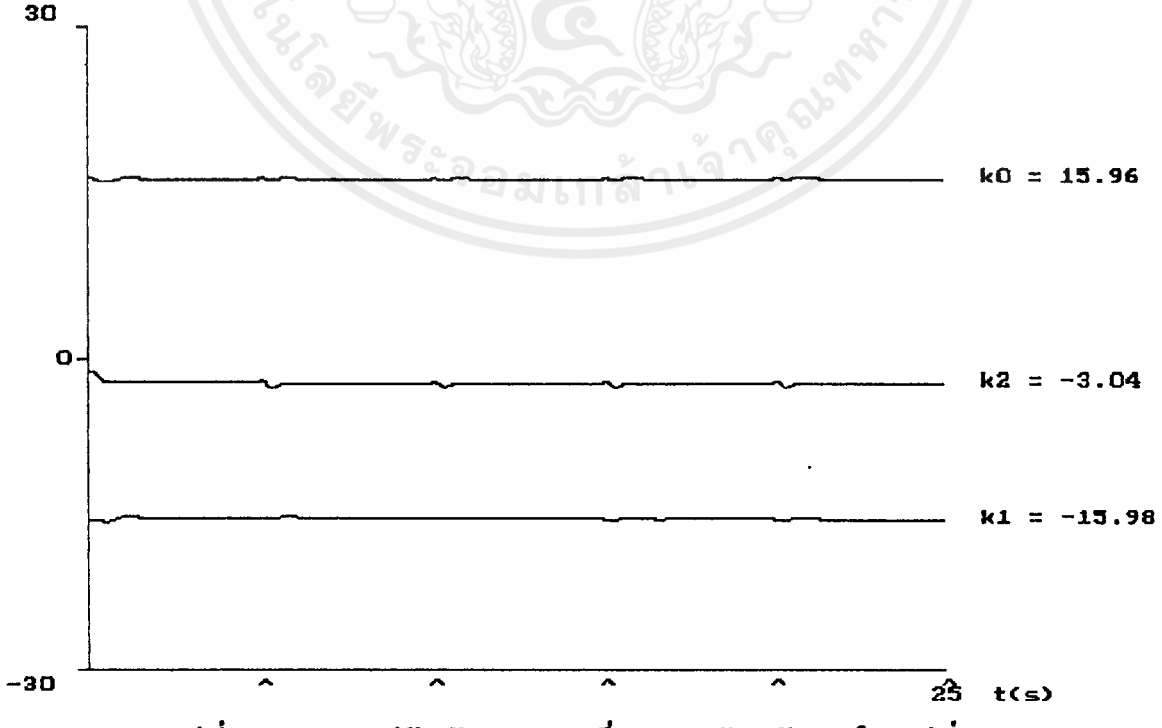
จะพบว่าเนื่องจากการที่สัญญาณ u ที่ออกจากการ์ด D/A ของคอมพิวเตอร์เข้าสู่ตัวระบบมีการอิ่มตัว จะส่งผลให้ค่า k ถูกปรับไปเรื่อย ๆ ไม่เข้าสู่ค่าสมดุล ซึ่งทำให้เกิดปรากฏการณ์ที่เกิดขึ้นกับระบบจริงดังรูป 4.7

2.2 ทำการศึกษาผลที่เกิดปรากฏการณ์เข้ากันไม่ได้ของระบบ กับ รูปแบบอ้างอิง (model mismatch)

เมื่อนำโปรแกรมจำลองระบบด้วยคอมพิวเตอร์ในภาคผนวก ก มาดัดแปลงโดยการเปลี่ยนกระบวนการจาก 1/s(s+2) เป็น 100/s(s+2)(s+100) ซึ่งเป็นระบบอันดับสามเป็นการเพิ่มโพลที่อยู่ไกลจากจุดกำเนิดมากและมีผลต่อกระบวนการน้อย และใช้รูปแบบอ้างอิงเดิมที่เป็นอันดับสอง พบว่าได้ผลดังรูปที่ 4.10 และได้สังเกตการเปลี่ยนแปลงของค่า k ดังรูปที่ 4.11



รูปที่ 4.10 ผลตอบสนองของ MRAS เมื่อเกิดปรากฏการณ์ model mismatch

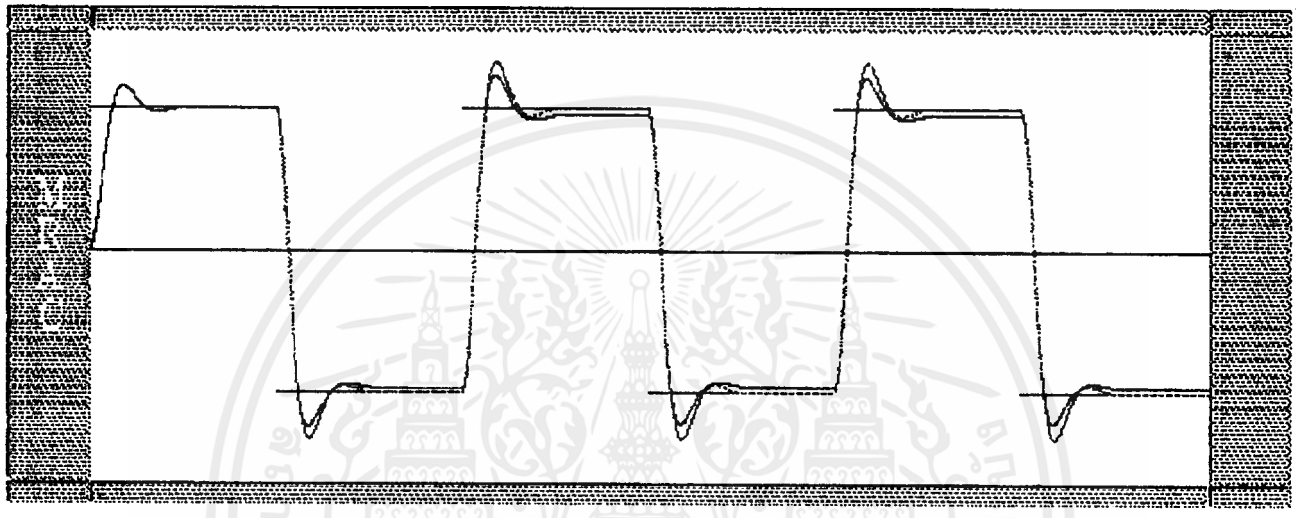


รูปที่ 4.11 การปรับตัวของ k ซึ่งสอดคล้องกับผลในรูปที่ 4.10

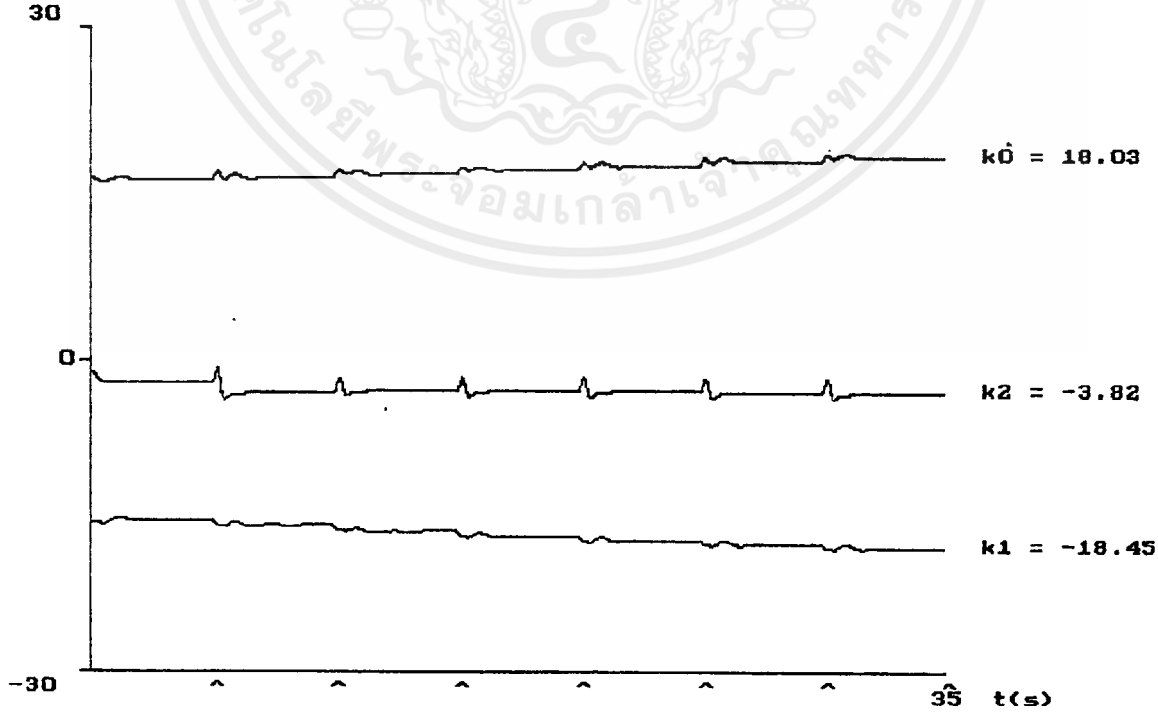
จะพบว่าเมื่อกระบวนการมีอันดับสูงขึ้น โดยการเพิ่มโพลที่ห่างจากจุดกำเนิดไปทางซ้ายมือมาก MRAC ก็ยังสามารถปรับค่า k ที่ทำให้ผลตอบสนองของระบบเข้าสู่รูปแบบอ้างอิงได้ แต่ค่า k ที่สถานะคงที่แล้วจะแตกต่างจากค่า k เมื่อไม่มีผลจากโพลที่เพิ่มขึ้น

2.3 ทำการศึกษาผลที่เกิดจากการอิ่มตัวของสัญญาณควบคุม u บวกกับปรากฏการณ์ model mismatch

โดยการจำลองให้เกิดเหตุการณ์เช่นเดียวกับกรณี 2.1 บวกกับ 2.2 ซึ่งได้ผลดังรูปที่ 4.12 และ 4.13



รูปที่ 4.12 ผลตอบสองเมื่อเกิดการอิ่มตัวบวกกับปรากฏการณ์ model mismatch



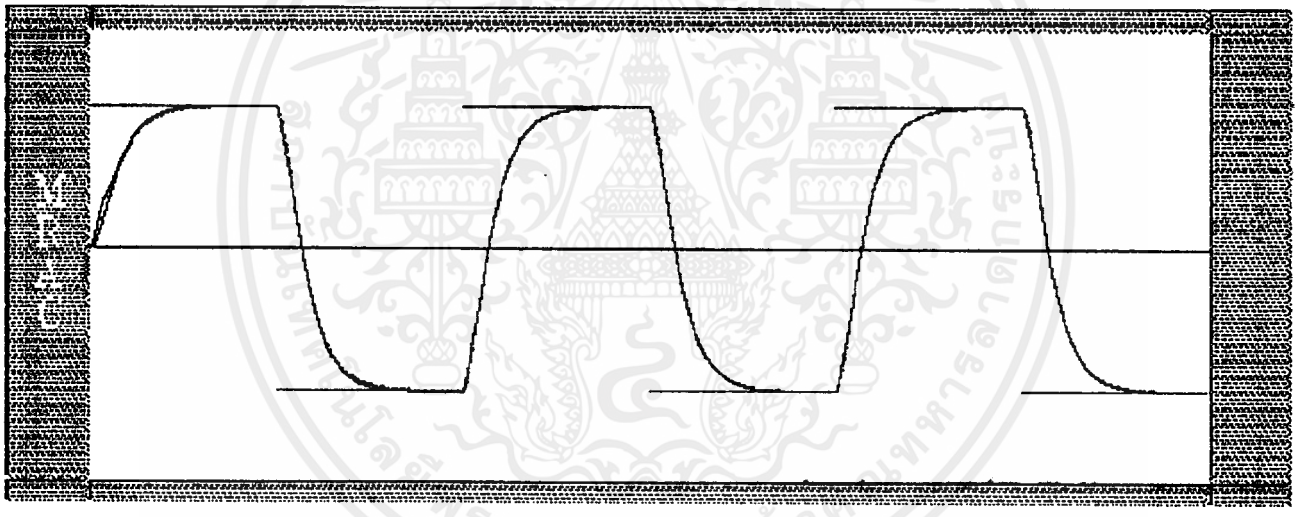
รูปที่ 4.13 การปรับตัวของ k ซึ่งสองคล้องกับผลในรูปที่ 4.12

จะพบว่าผลตอบสนองของระบบและการเปลี่ยนแปลงของค่า k จะมีลักษณะใกล้เคียงกับรูป 2.1 ซึ่งเป็นผลจากการอิ่มตัวซึ่งจะทำให้ค่า k ไม่เข้าสู่ค่าคงที่ แต่จะมีค่าเพิ่มขึ้นเรื่อย ๆ ในทางเดียว จึงทำให้ผลตอบสนองของกระบวนการไม่เข้าสู่รูปแบบอ้างอิง

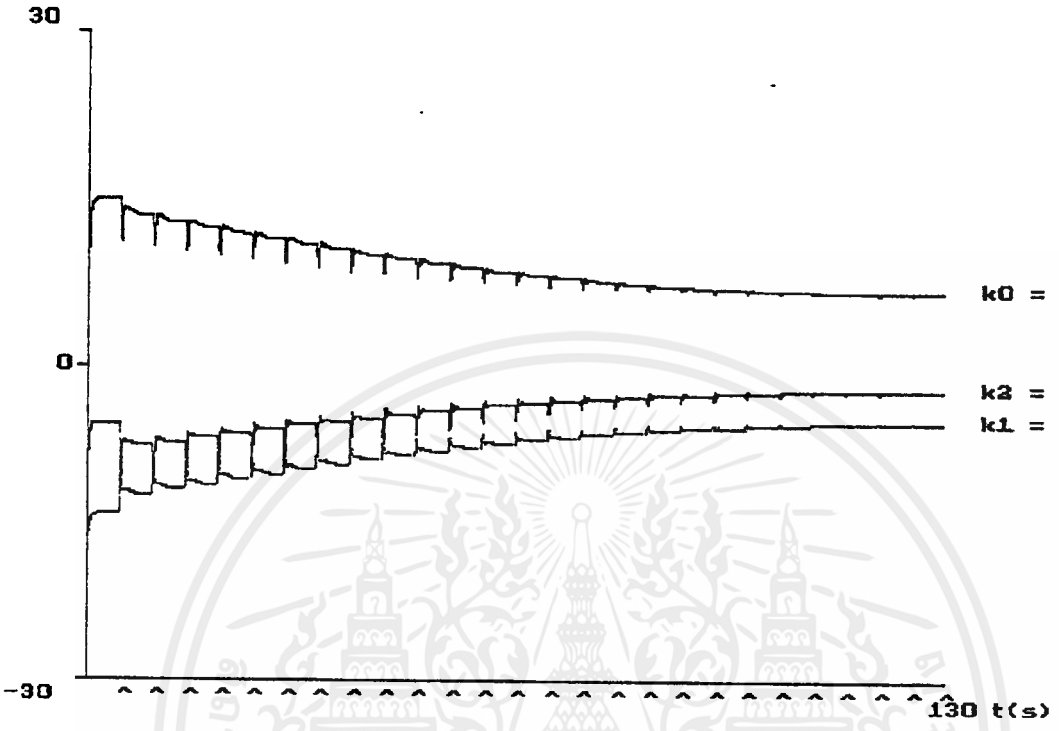
แต่สิ่งที่แตกต่างจากหัวข้อ 2.1 คือในหัวข้อนี้มีผลจากการไม่เข้ากันของระบบกับรูปแบบอ้างอิง โดยมีการเพิ่มโพลที่ห่างจากจุดกำเนิดมาก ๆ ทำให้เป็นกระบวนการอันดับสาม จึงพบว่า ในช่วงที่มีการเปลี่ยนแปลง set point ค่า k จะมีการเปลี่ยนแปลงที่เรียบกว่าในหัวข้อ 2.1 ซึ่งมีผลมาจากโพลที่เพิ่มเข้ามา อาจทำให้ระบบถูกหน่วง และ การแกว่งของค่า k จะน้อยลง

<u>2.4 ทำการศึกษาระบบที่เกิดการอิ่มตัวของสัญญาณควบคุม u แต่บวกการเปลี่ยนแปลงรูปแบบอ้างอิงเพื่อลดผลของการอิ่มตัว</u>

ในหัวข้อนี้ได้ทำการเปลี่ยนแปลงรูปแบบอ้างอิงจาก $16/s^2+4s+16$ เป็น $6/s^2+5s+6$ เพื่อให้โพลเปลี่ยนจาก $-2\pm3.464j$ เป็น -2 และ -3 ซึ่งจะเป็น overdamped หรือ ระบบอ้างอิงจะถูกหน่วงให้ช้าลงเพื่อลดผลของการอิ่มตัวของ u ได้ผลดังรูปที่ 4.14 และค่า k ที่ปรับตัวดังรูปที่ 4.15



รูปที่ 4.14 ผลตอบสนองของ MRAS กรณีเกิดการอิ่มตัวแต่หน่วงรูปแบบอ้างอิง

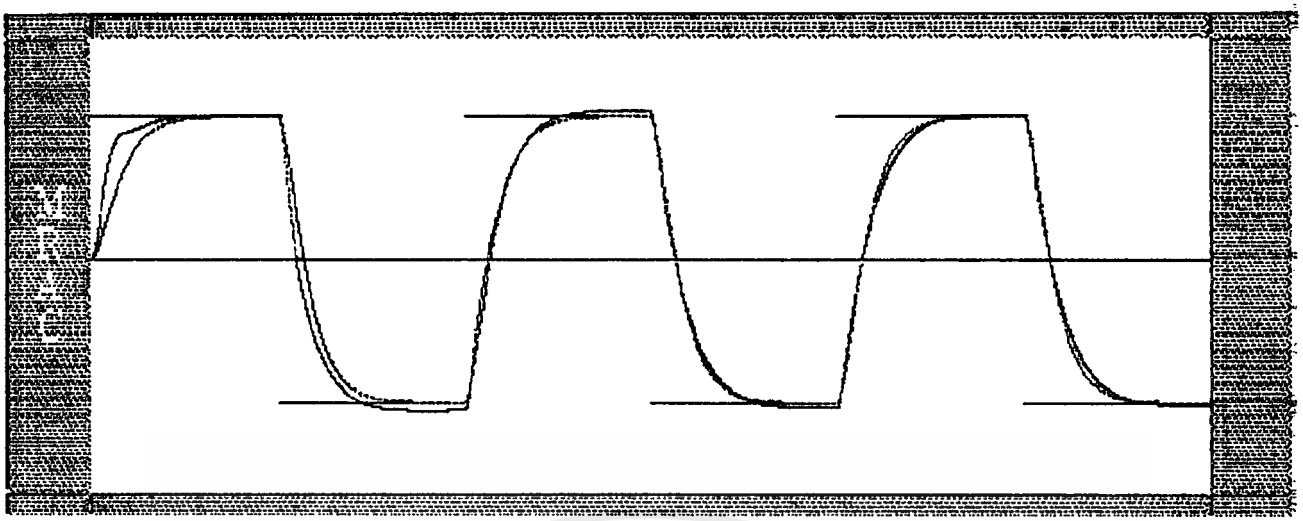


รูปที่ 4.15 การปรับตัวของค่า k ที่สอดคล้องกับผลในรูปที่ 4.14

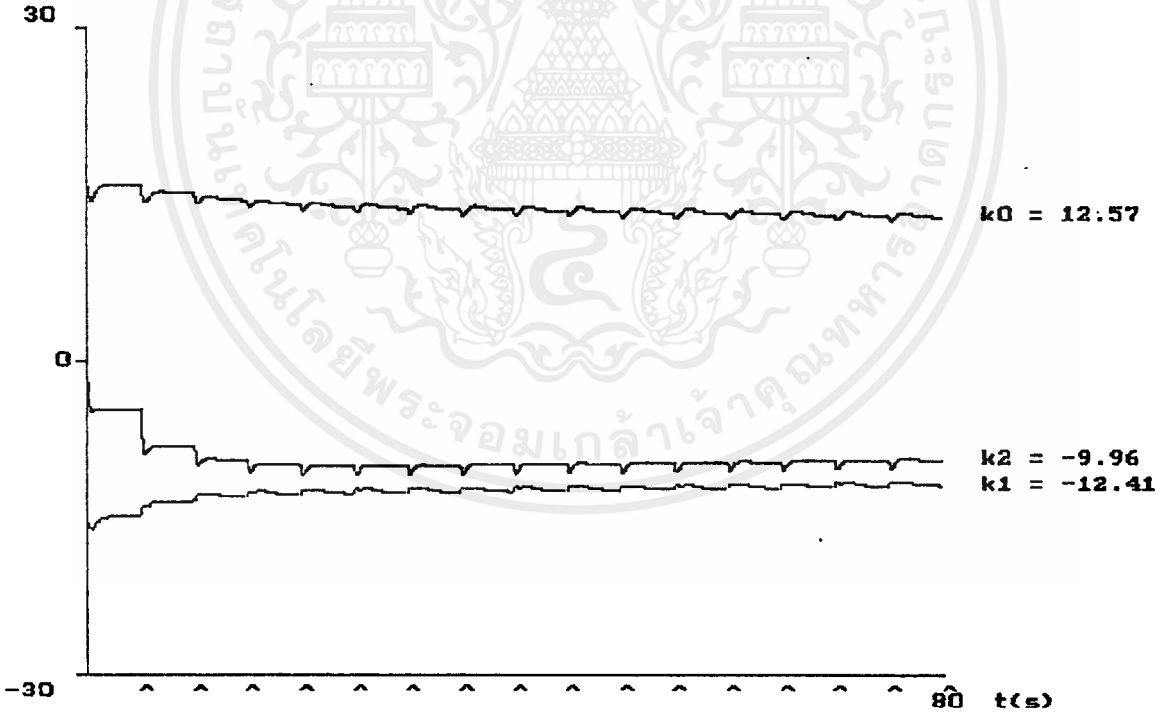
จาก 2.1 พบว่าต้นเหตุที่ทำให้ค่า k ไม่เข้าสู่สถานะคงที่คือผลจากการอิ่มตัวของสัญญาณควบคุม แต่เมื่อลดผลจากการอิ่มตัว โดยการเปลี่ยนแปลงรูปแบบอ้างอิงในลักษณะที่ผลตอบสนองช้าลงจะทำให้ไม่เกิด u ที่มีค่าสูงเกินไป ซึ่งจะเป็นการลดผลจากการอิ่มตัว จึงทำให้ MRAC สามารถปรับค่า k ให้เข้าสู่ค่า k ที่เหมาะสมได้ และ ค่า k เมื่อคงที่แล้วก็มีค่าที่สอดคล้องกับค่า k ที่ได้จากการคำนวณ เมื่อไม่มีการอิ่มตัว

2.5 ทำการศึกษาระบบที่มีการอิ่มตัว บวกกับปรากฏการ model mismatch แต่เปลี่ยนแปลงรูปแบบอ้างอิงให้มีความหน่วงมากขึ้นเพื่อลดผลของการอิ่มตัว

โดยใช้ระบบดังหัวข้อ 2.4 แต่ได้เปลี่นกระบวนการจาก 1/s(s +2) เป็น 100/s(s+2)(s+100) ซึ่งเป็นระบบอันดับ 3 ในขณะที่รูปแบบอ้างอิงยังเป็นระบบอันดับสองที่ย้ายโพลจาก -2 ±3.464j มาเป็น -2 และ -3 เพื่อศึกษาผลของ model mismatch ได้ผลดังรูปที่ 4.16 และ 4.17



รูปที่ 4.16 ผลตอบสนองเมื่อเกิดการอิ่มตัวของ u + model mismatch + หน่วงรูปแบบอ้างอิง



รูปที่ 4.17 การปรับตัวของค่า k ที่สอดคล้องกับผลในรูปที่ 4.16

จะพบว่า MRAC สามารถทำให้ผลตอบสนองของระบบเข้าสู่รูปแบบอ้างอิงได้เช่นเดียวกับหัวข้อ 2.4 แต่สิ่งที่แตกต่างกันคือค่า k เมื่อเข้าสู่สภาวะคงที่แล้วจะเป็นคนละค่ากับในหัวข้อ 2.4 (เป็นผลจาก model mismatch เหมือนกับในหัวข้อ 2.2 ซึ่งแสดงผลของการ mismatch)

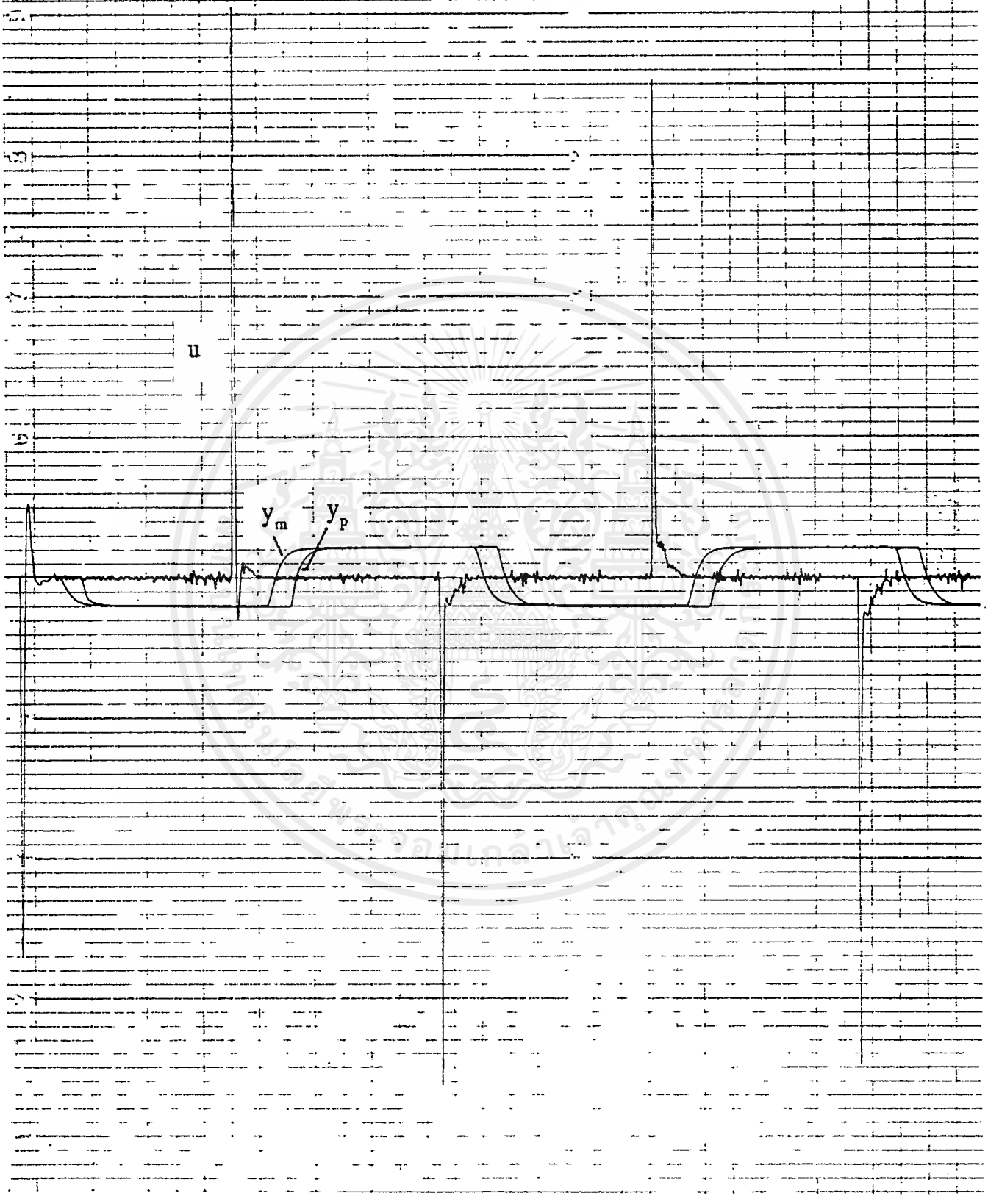
ขั้นที่ 3

ทำการทดลองกับระบบจริงโดยการเปลี่ยนแปลงรูปแบบอ้างอิงดังได้กล่าวไว้ในขั้นตอนที่ 2.4 ถึง 2.6 ซึ่งมีผลตอบสนองเป็น overdamped ซึ่งถูกหน่วงให้ช้าลง เพื่อที่จะลดผลจากการอิ่มตัว ซึ่งได้ผลดังรูปที่ 4.18

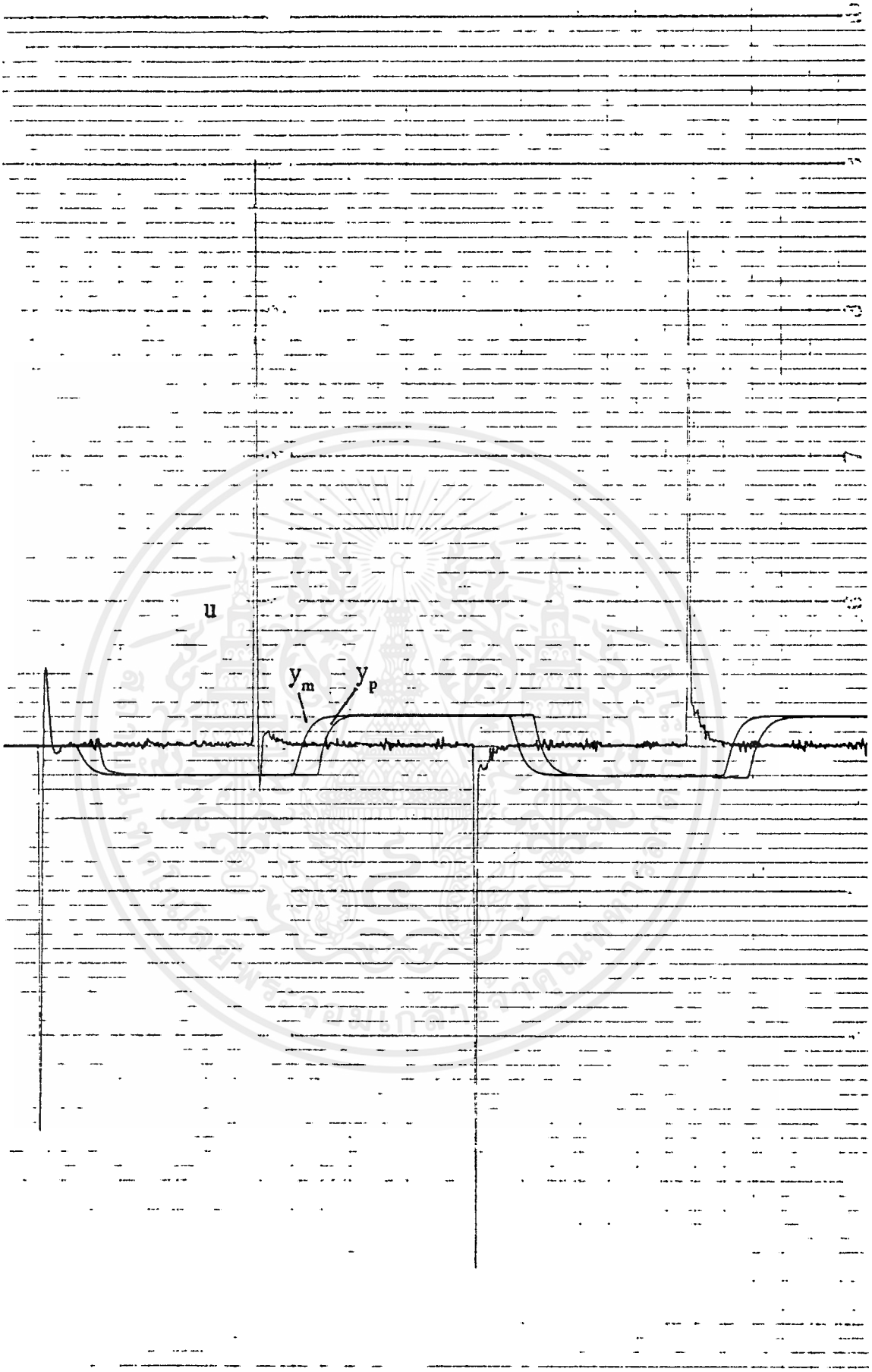
จากรูปจะพบว่าเมื่อเวลาผ่านไป MRAC จะทำให้ผลตอบสนองเข้าสู่ reference model ได้ตามต้องการ โดยค่า k จะเข้าสู่ค่าที่เหมาะสม

และเมื่อมีการเปลี่ยนแปลงพารามิเตอร์ในกระบวนการ (q) ได้ผลดังรูปที่ 4.19

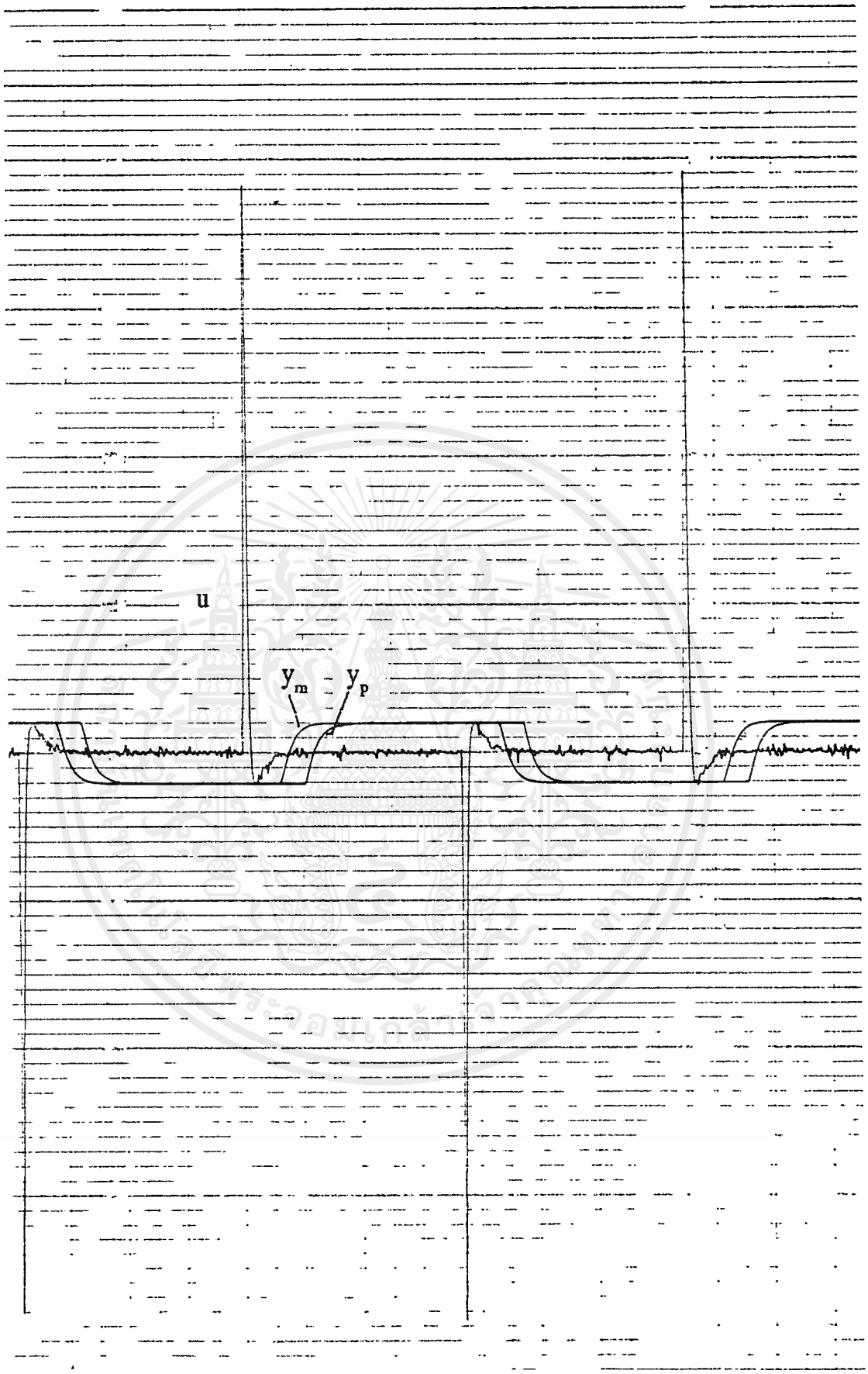
จะเห็นว่า MRAC สามารถควบคุมให้ผลตอบสนองเข้าสู่รูปแบบอ้างอิงได้ตามต้องการแม้ว่ากระบวนการจะเปลี่ยนแปลง



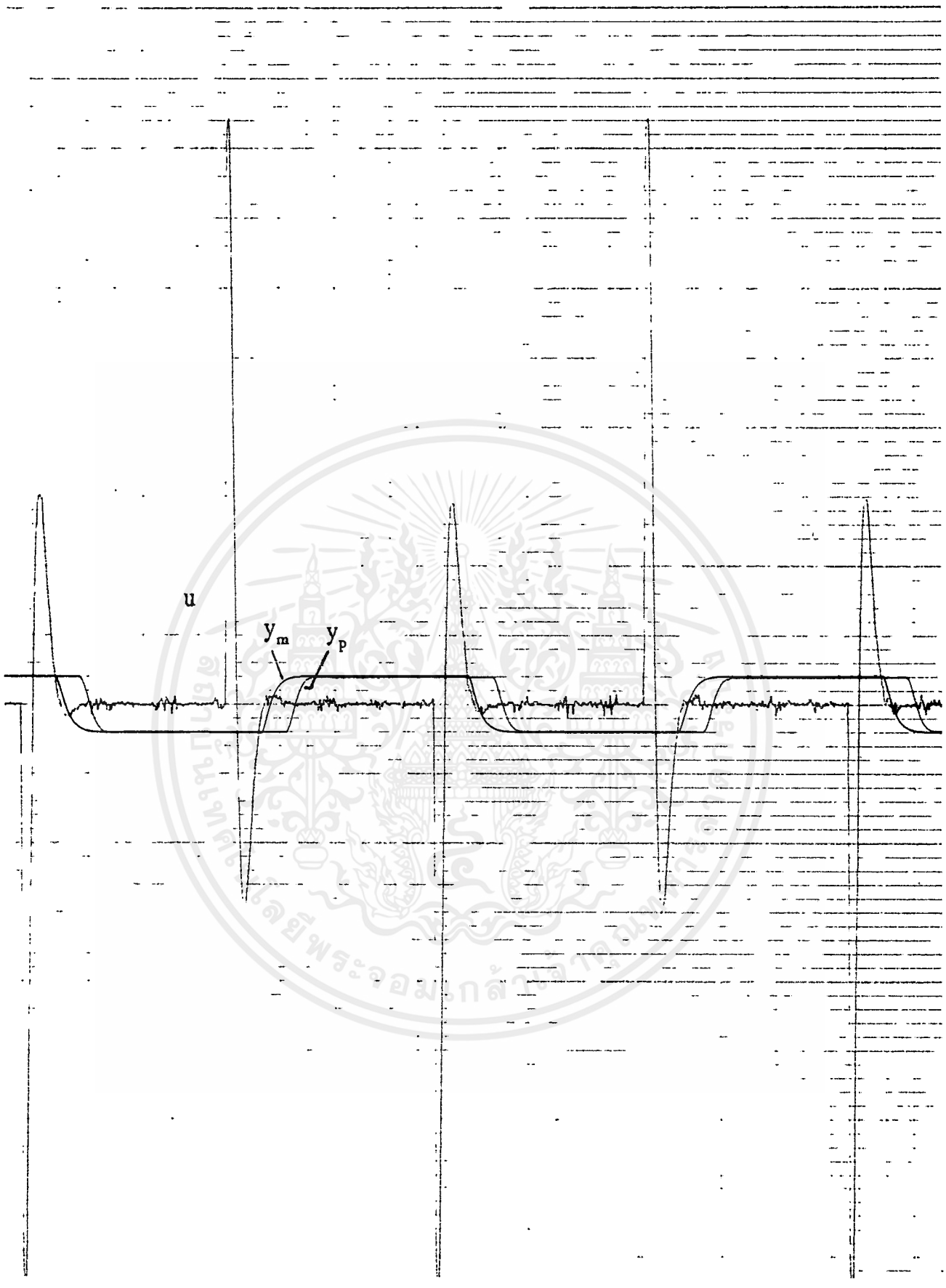
รูปที่ 4.18 ผลการทดลองกับระบบจริงเมื่อลดผลการอิ่มตัวโดยหน่วงรูปแบบอ้างอิง q = 1



รูปที่ 4.19 ผลการทดลองกับระบบจริงเมื่อลดผลการอิ่มตัวโดยหน่วงรูปแบบอ้างอิง
q = 1



รูปที่ 4.19 ผลการทดลองกับระบบจริงเมื่อลดผลการอิ่มตัวโดยหน่วงรูปแบบอ้างอิง และลองแปรค่า q
q = 0.5



รูปที่ 4.19 ผลการทดลองกับระบบจริงเมื่อลดผลการอิ่มตัวโดยหน่วงรูปแบบอ้างอิง และลองแปรค่า q
q = 0.25

สรุปผลของโครงงาน

จากโครงงานนี้พบว่าทฤษฎี MRAC สามารถทำการควบคุม และ ปรับตัวตามกระบวนการได้จริง ดังผลที่ได้จากการจำลอง ซึ่งจำกัดปัจจัยภายนอกอื่น ๆ ออกไป และเมื่อได้นำ MRAC มาใช้กับกระบวนการจริง พบว่าข้อสังเกตหนึ่งข้อ และมีปัจจัยสำคัญสองประการที่มีผลต่อการควบคุมของ MRAC

ข้อสังเกตคือ การที่จะให้ได้ผลตอบสนองที่รวดเร็วเราจะต้องใช้สัญญาณควบคุมที่มีค่าสูงกว่าค่าที่ set point ที่ต้องการใช้จริงถึง 10 เท่าหรืออาจจะมากกว่านั้น นั่นแสดงให้เห็นถึงข้อจำกัดของธรรมชาติที่มีต่อระบบ ไม่เฉพาะกับระบบควบคุม MRAC หรือ MRAS (Model Reference Adaptive System) เท่านั้น แต่เกิดขึ้นกับระบบควบคุมทุกแบบ จึงควรระวังในเรื่องนี้

และปัจจัยที่มีความสำคัญสองประการได้แก่

1. การอิ่มตัวของสัญญาณควบคุม u ที่ออกจากตัวควบคุม (คอมพิวเตอร์) ไปสู่กระบวนการ

เนื่องจากสัญญาณ u นี้จะถูกสร้างจากการ์ด D/A จึงทำให้ค่า u ถูกจำกัดอยู่ระหว่าง ±10 โวลต์ ซึ่งการแก้ปัญหานี้อาจแก้ได้โดยการเพิ่มตัวขยายสัญญาณให้สูงขึ้น แต่ก็จะยังคงมีข้อจำกัดทางกายภาพของกระบวนการที่จะรับสัญญาณได้ไม่เกิน ±12 โวลต์ ซึ่งการอิ่มตัวนี้จะมีผลต่อ MRAC ทำให้การปรับค่า K ไม่เข้าสู่สภาวะคงที่ และถ้าค่า k สูงขึ้นเรื่อย ๆ อาจไปถึงจุดที่ระบบไม่สามารถรับได้และเกิดการแกว่งหรือ oscillation ได้ จึงได้ทำการลดผลของการอิ่มตัวลงโดยการเปลี่ยนรูปแบบของรูปแบบอ้างอิง โดยทำให้มีผลตอบสนองที่ช้าลง ซึ่งจะใช้ u ที่ต่ำลง ทำให้ MRAC สามารถทำงานได้ปกติตามทฤษฎี

2. การไม่เข้ากันระหว่างกระบวนการ กับ รูปแบบอ้างอิง

ในการใช้ MRAC กับกระบวนการจริง อันดับของระบบรวมอาจมีค่าสูงกว่าที่ได้กำหนดไว้ในรูปแบบอ้างอิงซึ่งอาจเนื่องมาจากผลของส่วน interface และส่วนประกอบอื่น ๆ ในระบบ ซึ่งถ้ากระบวนการรวม มีอันดับที่สูงขึ้น เนื่องจากโพลที่อยู่ห่างจากจุดกำเนิดมาก จะมีผลน้อย ซึ่ง MRAC ยังสามารถควบคุมได้ แต่ผลที่แตกต่างคือ ค่า k ในสภาวะคงที่อาจมีค่าสูงขึ้นเนื่องจากต้องมีการใช้สัญญาณควบคุมที่สูงขึ้นเพื่อชดเชยการหน่วงของผลตอบสนองของระบบอันเนื่องจากโพลที่อยู่ห่างจากจุดกำเนิด

และนอกจากปัจจัยสองข้อดังกล่าวแล้ว ยังอาจมีปัจจัยอื่น ๆ อีกที่มีผลต่อ MRAC ยกตัวอย่างเช่น

- สัญญาณรบกวนในระบบ
- สัญญาณรบกวนในการวัด
- dead time ใน transfer function ของระบบ
- ความไม่เป็นเชิงเส้นของระบบ

แต่สำหรับระบบในโครงงานนี้มีผลจากปัจจัยเหล่านี้น้อยเนื่องจากส่วนประกอบของระบบในโครงงานนี้ประกอบขึ้นเพื่อให้ได้ใกล้เคียงกับอุดมคติ จึงไม่ได้วิเคราะห์ผลดังกล่าวไว้ในที่นี้

ภาคผนวก

ภาคผนวก ก
โปรแกรมจำลอง MRAC

รายละเอียดของโปรแกรม MRACSIM

- เป็นโปรแกรมเปรียบเทียบการทำงานของระบบลูปปิดอันดับสองที่มีตัวควบคุม MRAC กับที่มีตัวควบคุมสเตทป้อนกลับดังที่ได้ โดยสมการทุกอย่างเสนอไว้ในบทที่ 3
- การจำลองใช้ระเบียบวิธีของ Runge-Kutta
- ค่า q ของกระบวนการในสมการที่ 3.1 และ 3.2 สามารถเปลี่ยนได้ 5 ระดับ ตั้งแต่ q เท่ากับ 0.25, 0.5, 1, 2 และ 4 กดปุ่มลูกศรขึ้นจะเพิ่มค่า q กดปุ่มลูกศรลงจะลดค่า q โดยตอนเริ่มต้นโปรแกรมจะตั้งค่า q เท่ากับ 1 ซึ่งเป็นจุดทำงานปกติ และค่าพารามิเตอร์ของตัวควบคุมจะตั้งค่าเริ่มต้นให้ระบบลูปปิดมีผลตอบสนองตรงกับรูปแบบอ้างอิง
- กดปุ่มใด ๆ เพื่อหยุดการจำลองระบบและกดอีกครั้งเพื่อออกจากโปรแกรม

โปรแกรมย่อย

```
int  f_to_i(double x)
        เปลี่ยนเลขทศนิยมเป็นเลขจำนวนเต็ม โดยปัดเศษ

void  s_x_v(double scalar, double vector[], double vector_r[])
        vector_r = scalar*vector

void  s_x_m(double scalar, double matrix[][N], double matrix_r[][N])
        matrix_r = scalar*matrix

void  v_mul_v(double vector_1[], double vector_2[], double matrix_r[][N])
        matrix_r = vector_1*vector_2^T
        เป็นการคูณของหลักกับแถว

void  v_x_v(double vector_1[], double vector_2[], double *scalar)
        scalar = vector_1^T*vector_2
        เป็นการคูณของแถวกับหลัก

void m_x_v(double matrix[][N], double vector[], double vector_r[])
        matrix_r = matrix*vector

void m_add_m(double matrix_1[][N], double matrix_2[][N],
             double matrix_r[][N])
        matrix_r = matrix_1 + matrix_2

void v_add_v(double vector_1[], double vector_2[], double vector_r[])
        vector_r = vector_1*vector_2
```

void move_graph(void)
เลื่อนหน้าจอเพื่อทำให้กราฟผลตอบสนองเคลื่อนไปเรื่อย ๆ

void plot(void)
นำผลที่ได้จากการคำนวณผลตอบสนองมาจุดลงไปบนหน้าจอ

void sys(double t, double un, double xn[], double xn_1[])
นำเสตทเวคเตอร์ xn และ un มาเข้าสมการสเตทของกระบวนการ เพื่อให้ได้อนุพันธ์ของเสตทเวคเตอร์ xn

void pid_ctr_sys(void)
จำลองผลตอบสนองต่อไปของระบบควบคุมคงที่ ที่ควบคุมโดยวิธีป้อนกลับสเตท โดยวิธีของ Runge-Kutta

void adp_ctr_sys(void)
จำลองผลตอบสนองต่อไปของระบบควบคุมปรับตัวได้ ที่ควบคุมโดยวิธี MRAC ด้วยสมการลีอาปูนอฟ การจำลองใช้วิธีของ Runge-Kutta

void ref_sys(void)
นำเสตทเวคเตอร์ xn และ un ของรูปแบบอ้างอิงมาเข้าสมการสเตทของรูปแบบอ้างอิง เพื่อให้ได้อนุพันธ์ของเสตทเวคเตอร์ xn ของรูปแบบอ้างอิง

void set_parameter(void)
ตั้งค่าพารามิเตอร์ของกระบวนการโดยขึ้นอยู่กับค่า q

void initialize(void)
ตั้งค่าเริ่มต้นให้กับทุกอย่างของระบบ

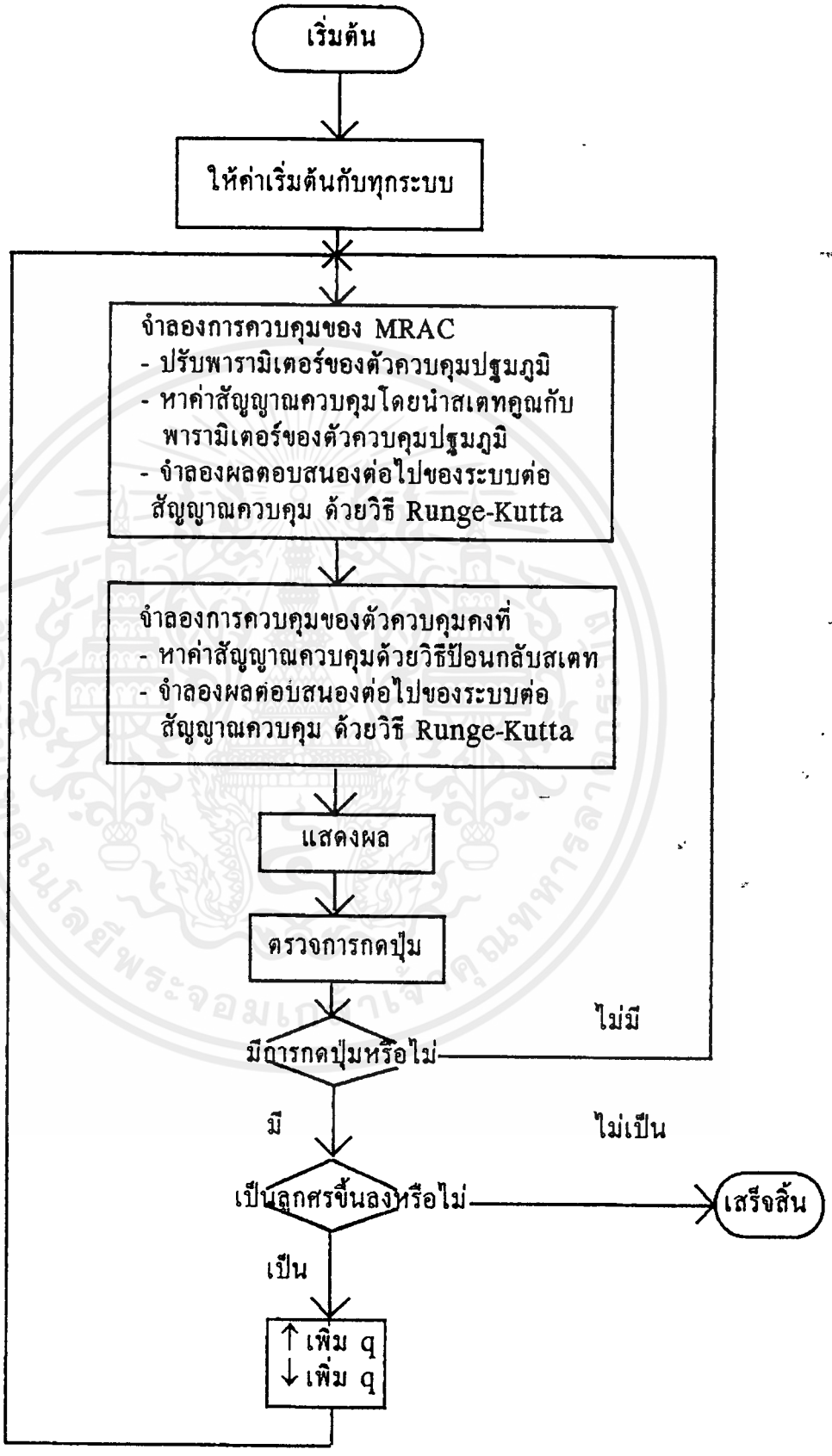
void init_image(void)
วาดหน้าจอ

void readme(void)
ออกข้อความบอกคุณลักษณะของโปรแกรมโดยย่อ

ตัวแปร

pid_x, pid_y, pid_u, pid_e	เป็นตัวแปรของระบบควบคุมคงที่
adp_x, adp_y, adp_u, adp_e, adp_de	เป็นตัวแปรของระบบควบคุมปรับตัวได้
pid_k	พารามิเตอร์ของตัวควบคุมคงที่หรือเกนป้อนกลับเสตท
adp_k, adp_dk	พารามิเตอร์ของตัวควบคุมปฐมภูมิของตัวควบคุมปรับตัว
sys_A, sys_B, sys_C, sys_d	เป็นพารามิเตอร์ของสมการสเตทของกระบวนการ
ref_A, ref_B, ref_C, ref_d	เป็นพารามิเตอร์ของสมการสเตทของรูปแบบอ้างอิง
k_1, k_2, k_3, _k4	ค่าคงที่ใช้ในระเบียบวิธี Runge-Kutta
g	ค่าคงที่ γ ในสมการ (3.16) ในกฎการปรับตัว
h	คาบเวลาในการจำลองแต่ละครั้ง
sp	set point

Flow Chart



```
/*                    M R A C SIMULATION  ( 29/9/92 )                    */
/* MRACSIM.C : This program is written to simulate MRAS.                 */

#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <process.h>
#include <math.h>
#include <dos.h>
#include <alloc.h>

#define N 2

int f_to_i(double x);
void s_x_v(double scalar, double vector[], double vector_r[]);
void s_x_m(double scalar, double matrix[][N], double matrix_r[][N]);
void v_mul_v(double vector_1[], double vector_2[], double matrix_r[][N]);
void v_x_v(double vector_1[], double vector_2[], double *scalar);
void m_x_v(double matrix[][N], double vector[], double vector_r[]);
void m_add_m(double matrix_1[][N], double matrix_2[][N],
             double matrix_r[][N]);
void v_add_v(double vector_1[], double vector_2[], double vector_r[]);
void move_graph(void);
void plot(void);
void sys(double t, double un, double xn[], double xn_1[]);
void pid_ctr_sys(void);
void adp_ctr_sys(void);
void ref_sys(double t, double un, double xn[], double xn_1[]);
void ref_mod_sys(void);
void set_parameter(void);
void initialize(void);
void init_image(void);
void readme(void);

char mixcolor[8] = {0xff,0x00,0xff,0x00,0xff,0x00,0xff,0x00};
char *bmp;

double ratio_x, ratio_y, init_x, init_y_pid, init_y_adp;
double q;
double sp, t, h;
double sys_a[N+1], sys_b[N+1];
double sys_A[N][N], sys_B[N], sys_C[N], sys_d;
double pid_x[N], pid_e, pid_u, pid_y;
double pid_k[N];
double adp_x[N], adp_e, adp_de, adp_u, adp_y;
double adp_k[N], adp_dk[N], adp_k_sp, adp_dk_sp;
double g[N+1];
double ref_A[N][N], ref_B[N], ref_C[N], ref_d;
double ref_x[N], ref_e, ref_u, ref_y;
double k1[N], k2[N], k3[N], k4[N];
double temp1[N], temp2[N], temp3[N], temp4[N];
```

```
int f_to_i(double x)
{
        double y;

        if(modf(x,&y) >= 0.5)
                return(y + 1);
        else if(modf(x,&y) <= -0.5)
                return(y - 1);
        else
                return(y);
}

void s_x_v(double scalar, double vector[], double vector_r[])
{
        int i;

        for(i = 0; i < N; i++)
                vector_r[i] = scalar*vector[i];
}

void s_x_m(double scalar, double matrix[][N], double matrix_r[][N])
{
        int i,j;

        for(i = 0; i < N; i++)
                for(j = 0; j < N; j++)
                        matrix_r[i][j] = scalar * matrix[i][j];
}

void v_mul_v(double vector_1[], double vector_2[], double matrix_r[][N])
{
        int i,j;

        for(i = 0; i < N; i++)
                for(j = 0; j < N; j++)
                        matrix_r[i][j] = vector_1[i] * vector_2[j];
}

void v_x_v(double vector_1[], double vector_2[], double *scalar)
{
        int i;
        double temp;

        temp = 0.0;
        for(i = 0; i < N; i++)
                temp += vector_1[i]*vector_2[i];
        *scalar = temp;
}

void m_x_v(double matrix[][N], double vector[], double vector_r[])
{
        int i, j;
        double temp;

        for(i = 0; i < N; i++){
                temp = 0.0;
                for(j = 0; j < N; j++){
                        temp += matrix[i][j]*vector[j];
                }
                vector_r[i] = temp;
        }
}
```

```
void m_add_m(double matrix_1[][N], double matrix_2[][N],
             double matrix_r[][N])
{
        int i,j;

        for(i = 0; i < N; i++)
                for(j = 0; j < N; j++)
                        matrix_r[i][j] = matrix_1[i][j] + matrix_2[i][j];
}

void v_add_v(double vector_1[], double vector_2[], double vector_r[])
{
        int i;

        for(i = 0; i < N; i++)
                vector_r[i] = vector_1[i] + vector_2[i];
}

void move_graph(void)
{
        /*move pid viewport*/
        getimage(43,getmaxy()/2+6,getmaxx()-41,getmaxy()-11,bmp);
        putimage(42,getmaxy()/2+6,bmp,0);
        setcolor(0);
        line(getmaxx()-41,getmaxy()/2+6,getmaxx()-41,getmaxy()-11);
        putpixel(getmaxx()-41,3*getmaxy()/4,15);

        /*move adaptive viewport*/
        getimage(43,12,getmaxx()-41,getmaxy()/2-6,bmp);
        putimage(42,12,bmp,0);
        setcolor(0);
        line(getmaxx()-41,12,getmaxx()-41,getmaxy()/2+6);
        putpixel(getmaxx()-41,getmaxy()/4,15);
}

void plot(void)
{
        static int absc = 0;
        static int ordi_pid = 0, ordi_ref = 0, ordi_adp = 0;

        ordi_pid = f_to_i(init_y_pid - pid_y*ratio_y);
        ordi_ref = f_to_i(init_y_adp - ref_y*ratio_y);
        ordi_adp = f_to_i(init_y_adp - adp_y*ratio_y);
        if(f_to_i(init_x + t*ratio_x) <= getmaxx()-41){
                absc = f_to_i(init_x + t*ratio_x);
                putpixel(absc, f_to_i(init_y_pid - sp*ratio_y), 11);
                putpixel(absc, f_to_i(init_y_adp - sp*ratio_y), 11);
                if((ordi_pid > getmaxy()/2+6)&&(ordi_pid < getmaxy()-11))
                        putpixel(absc, ordi_pid, 15);
                else
                        ;
                putpixel(absc, ordi_ref, 12);
                if((ordi_adp > 12)&&(ordi_adp < getmaxy()/2-6))
                        putpixel(absc, ordi_adp, 15);
                else
                        ;
        }
```

```
        else{
                if( f_to_i(init_x + t*ratio_x) > absc ){
                        absc =  f_to_i(init_x + t*ratio_x);
                        move_graph();
                }
                else
                        ;
                putpixel(getmaxx()-41, f_to_i(init_y_pid - sp*ratio_y), 11);
                putpixel(getmaxx()-41, f_to_i(init_y_adp - sp*ratio_y), 11);
                if((ordi_pid > getmaxy()/2+6)&&(ordi_pid < getmaxy()-11))
                        putpixel(getmaxx()-41, ordi_pid, 15);
                else
                        ;
                putpixel(getmaxx()-41, ordi_ref, 12);
                if((ordi_adp > 12)&&(ordi_adp < getmaxy()/2-6))
                        putpixel(getmaxx()-41, ordi_adp, 15);
                else
                        ;
        }
}

void sys(double t, double un, double xn[], double xn_1[])
{
        double Ax[N], Bu[N];

        m_x_v(sys_A, xn, Ax);
        s_x_v(un, sys_B, Bu);
        v_add_v(Ax, Bu, xn_1);
}

void pid_ctr_sys(void)
{
                /*Calculate state feedback control signal.*/
                v_x_v(pid_k,pid_x,&pid_u);
                pid_u = -1.0*pid_u + pid_k[0]*sp;

                /*Calculate pid_x(k+1) with runge-kutta method.*/
                sys(t, pid_u, pid_x, temp1);
                s_x_v(h, temp1, k1);
                s_x_v(0.5, k1, temp1);
                v_add_v(pid_x, temp1, temp1);
                sys(t + h/2, pid_u, temp1, temp2);
                s_x_v(h, temp2, k2);
                s_x_v(0.5, k2, temp2);
                v_add_v(pid_x, temp2, temp2);
                sys(t + h/2, pid_u, temp2, temp3);
                s_x_v(h, temp3, k3);
                v_add_v(pid_x, k3, temp3);
                sys(t + h, pid_u, temp3, temp4);
                s_x_v(h, temp4, k4);
                s_x_v(2.0, k2, temp1);
                s_x_v(2.0, k3, temp2);
                v_add_v(k1, temp1, k2);
                v_add_v(k2, temp2, k3);
                v_add_v(k3, k4, temp3);
                s_x_v(1.0/6.0, temp3, temp4);
                v_add_v(pid_x, temp4, pid_x);
                v_x_v(sys_C, pid_x, &pid_y);
}
```

```
void adp_ctr_sys(void)
{
                /*Adaptive Laws.*/
                adp_e = adp_x[0] - ref_x[0];
                adp_de = adp_x[1] - ref_x[1];

                adp_dk_sp =  -1.0*g[0]*(4*adp_e + 17*adp_de)*sp*h;
                adp_dk[0] =  -1.0*g[1]*(4*adp_e + 17*adp_de)*adp_x[0]*h;
                adp_dk[1] =  -1.0*g[2]*(4*adp_e + 17*adp_de)*adp_x[1]*h;

                adp_k_sp += adp_dk_sp;
                adp_k[0] += adp_dk[0];
                adp_k[1] += adp_dk[1];

                /*Calculate adaptive control signal.*/
                v_x_v(adp_k,adp_x,&adp_u);
                adp_u = adp_u + adp_k_sp*sp;

                /*Calculate adp_x(k+1) with runge-kutta method.*/
                sys(t, adp_u, adp_x, temp1);
                s_x_v(h, temp1, k1);
                s_x_v(0.5, k1, temp1);
                v_add_v(adp_x, temp1, temp1);
                sys(t + h/2, adp_u, temp1, temp2);
                s_x_v(h, temp2, k2);
                s_x_v(0.5, k2, temp2);
                v_add_v(adp_x, temp2, temp2);
                sys(t + h/2, adp_u, temp2, temp3);
                s_x_v(h, temp3, k3);
                v_add_v(adp_x, k3, temp3);
                sys(t + h, adp_u, temp3, temp4);
                s_x_v(h, temp4, k4);
                s_x_v(2.0, k2, temp1);
                s_x_v(2.0, k3, temp2);
                v_add_v(k1, temp1, k2);
                v_add_v(k2, temp2, k3);
                v_add_v(k3, k4, temp3);
                s_x_v(1.0/6.0, temp3, temp4);
                v_add_v(adp_x, temp4, adp_x);
                v_x_v(sys_C, adp_x, &adp_y);
}

void ref_sys(double t, double un, double xn[], double xn_1[])
{
        double Ax[N], Bu[N];

        m_x_v(ref_A, xn, Ax);
        s_x_v(un, ref_B, Bu);
        v_add_v(Ax, Bu, xn_1);
}

void ref_mod_sys(void)
{
                /*Calculate model reference x(k+1) with runge-kutta method.*/
                ref_sys(t, ref_u, ref_x, temp1);
                s_x_v(h, temp1, k1);
                s_x_v(0.5, k1, temp1);
                v_add_v(ref_x, temp1, temp1);
                ref_sys(t + h/2, ref_u, temp1, temp2);
                s_x_v(h, temp2, k2);
                s_x_v(0.5, k2, temp2);
```

```
                v_add_v(ref_x, temp2, temp2);
                ref_sys(t + h/2, ref_u, temp2, temp3);
                s_x_v(h, temp3, k3);
                v_add_v(ref_x, k3, temp3);
                ref_sys(t + h, ref_u, temp3, temp4);
                s_x_v(h, temp4, k4);
                s_x_v(2.0, k2, temp1);
                s_x_v(2.0, k3, temp2);
                v_add_v(k1, temp1, k2);
                v_add_v(k2, temp2, k3);
                v_add_v(k3, k4, temp3);
                s_x_v(1.0/6.0, temp3, temp4);
                v_add_v(ref_x, temp4, ref_x);
                v_x_v(sys_C, ref_x, &ref_y);
}

void set_parameter(void)
{
        sys_A[0][1] = 1.0;
        sys_A[1][1] = -2.0*q;

        sys_B[0] = 0.0;
        sys_B[1] = 1.0*q;

        sys_C[0] = 1.0;
        sys_C[1] = 0.0;

        sys_d = 0.0;
}

void initialize(void)
{
        q = 1.0;

        /*initialize system*/
        set_parameter();

        sp = 1.0;
        t = 0;
        h = 1.0/128.0;

        /*initialize pid control*/
        pid_k[0] = 16.0;
        pid_k[1] = 2.0;
        pid_y = 0.0;

        /*initialize adaptive control*/
        adp_k[0] = -16.0;
        adp_k[1] = -2.0;
        adp_k_sp = 16.0;
        adp_y = 0.0;
        g[0] = 10.0;
        g[1] = 10.0;
        g[2] = 10.0;

        /*initialize reference model*/
        ref_A[0][1] = 1.0;
        ref_A[1][0] = -16.0;
        ref_A[1][1] = -4.0;
```

```
        ref_B[0] = 0.0;
        ref_B[1] = 16.0;

        ref_C[0] = 1.0;
        ref_C[1] = 0.0;

        ref_d = 0.0;

        ref_u = sp;
        ref_y = 0.0;
}

void init_image(void)
{
        int i;
        unsigned size;

        cleardevice();
        setfillpattern(mixcolor,3);
        setfillstyle(12,3);
        bar(0,0,getmaxx(),getmaxy());

        setcolor(7);
        line(0,0,0,getmaxy());
        line(1,0,1,getmaxy());
        line(0,0,getmaxx(),0);
        line(0,getmaxy()-10,getmaxx(),getmaxy()-10);
        line(getmaxx()-39,0,getmaxx()-39,getmaxy());
        line(getmaxx()-40,0,getmaxx()-40,getmaxy());
        line(0,12,41,12);
        line(42,0,42,10);
        line(43,0,43,10);
        line(42,getmaxy(),42,getmaxy()-10);
        line(43,getmaxy(),43,getmaxy()-10);
        line(getmaxx(),12,getmaxx()-39,12);
        line(0,getmaxy()/2-5,getmaxx(),getmaxy()/2-5);
        line(42,getmaxy()/2-5,42,getmaxy()/2+5);
        line(0,getmaxy()/2+6,41,getmaxy()/2+6);
        line(getmaxx()-40,getmaxy()/2+6,getmaxx(),getmaxy()/2+6);

        setcolor(8);
        line(1,10,getmaxx(),10);
        line(0,11,getmaxx(),11);
        line(40,1,40,getmaxy());
        line(41,0,41,getmaxy());
        line(getmaxx(),0,getmaxx(),getmaxy());
        line(getmaxx()-1,1,getmaxx()-1,getmaxy());
        line(0,getmaxy(),getmaxx(),getmaxy());
        line(1,getmaxy(),getmaxx(),getmaxy());
        line(1,getmaxy()-11,41,getmaxy()-11);
        line(0,getmaxy()-12,41,getmaxy()-12);
        line(getmaxx(),getmaxy()-11,getmaxx()-39,getmaxy()-11);
        line(getmaxx(),getmaxy()-12,getmaxx()-40,getmaxy()-12);
        line(getmaxx()-41,0,getmaxx()-41,11);
        line(getmaxx()-42,1,getmaxx()-42,11);
        line(getmaxx()-41,getmaxy(),getmaxx()-41,getmaxy()-10);
        line(getmaxx()-42,getmaxy(),getmaxx()-42,getmaxy()-9);
        line(0,getmaxy()/2+5,getmaxx(),getmaxy()/2+5);
        line(getmaxx()-41,getmaxy()/2-5,getmaxx()-41,getmaxy()/2+5);
        line(getmaxx()-42,getmaxy()/2-4,getmaxx()-42,getmaxy()/2+5);
        line(1,getmaxy()/2-6,41,getmaxy()/2-6);
```

```
        line(0,getmaxy()/2-7,41,getmaxy()/2-7);
        line(getmaxx()-39,getmaxy()/2-7,getmaxx(),getmaxy()/2-7);
        line(getmaxx()-40,getmaxy()/2-6,getmaxx(),getmaxy()/2-6);

        for(i=0; i<2; i++){
                putpixel(41+i,12,0);
                putpixel(41+i,getmaxy()/2-6,0);
                putpixel(41+i,getmaxy()/2+6,0);
                putpixel(41+i,getmaxy()-11,0);
        }
        putpixel(getmaxx()-41,13,0);
        putpixel(getmaxx()-41,getmaxy()/2+6,0);
        putpixel(getmaxx()-41,getmaxy()-11,0);
        putpixel(getmaxx()-41,getmaxy()/2-6,0);
        putpixel(0,12,0);
        putpixel(0,getmaxy()/2+6,0);
        putpixel(0,getmaxy()-11,0);
        putpixel(0,getmaxy()/2-6,0);
        putpixel(getmaxx(),13,0);
        putpixel(getmaxx(),getmaxy()/2+6,0);
        putpixel(getmaxx(),getmaxy()-11,0);
        putpixel(getmaxx(),getmaxy()/2-6,0);

        setfillstyle(0,0);
        bar(42,12,getmaxx()-41,getmaxy()/2-6);
        bar(42,getmaxy()/2+6,getmaxx()-41,getmaxy()-11);

        setcolor(0);
        settextstyle(1,0,2);
        outtextxy(15,80,"M");
        outtextxy(15,100,"R");
        outtextxy(15,120,"A");
        outtextxy(17,140,"C");
        outtextxy(15,300,"F");
        outtextxy(17,320,"I");
        outtextxy(15,340,"X");
        outtextxy(15,360,"E");
        outtextxy(15,380,"D");

        init_x = 42;
        init_y_adp = getmaxy()/4;
        init_y_pid = 3*getmaxy()/4;

        ratio_x = (getmaxx()-82.0)/25.0;
        ratio_y = getmaxy()/6.0 - 10;

        setcolor(15);
        line(41,getmaxy()/4,getmaxx()-41,getmaxy()/4);
        line(41,3*getmaxy()/4,getmaxx()-41,3*getmaxy()/4);

        /*initgraphviewport*/
        size = imagesize(42,getmaxy()/2+6,getmaxx()-41,getmaxy()-11);
        bmp = (char *)malloc(size);
}
```

```
void readme(void)
{
        clrscr();
        printf("(18/3/93)\n");
        printf("This program is written to simulate MRAS.\n");
        printf("Press arrow key up and down to change the process.\n");
        printf("Look for detail in the thesis.\n");
        printf("Press any key to start simulation and ");
        printf("press another key to quit the program.\n");
        getch();
}

main()
{
        int G_DRIVER, G_MODE, G_ERROR, i, ex;
        double test_num;
        union inkey{
                char ch[2];
                int i;
        } c;
        char ch;

        readme();

        /*Initgraph*/
        detectgraph (&G_DRIVER, &G_MODE);
        initgraph(&G_DRIVER, &G_MODE, "");

        /*Initialize mornitor*/
        init_image();

        /*Initialize control system*/
        initialize();

        for(ex = 0; ex == 0; ){
                adp_ctr_sys();
                ref_mod_sys();
                pid_ctr_sys();
                plot();
                t += h;
                if(fmod(t,5.0) == 0.0){
                        sp *= -1;
                        ref_u = sp;
                }
                if(bioskey(1)){
                        c.i = bioskey(0);
                        if(c.ch[0] == 0){
                                ch = c.ch[1];
                                if(ch == 72){
                                        q *= 2.0;
                                        if(q > 4)
                                                q = 4;
                                        else
                                                ;
                                }
                                if(ch == 80){
                                        q /= 2.0;
                                        if(q < .25)
                                                q = 0.25;
                                        else
                                                ;
                                }
```

```
                        if(ch == 71){
                                q = 1;
                        }
                        else
                                ;
                        set_parameter();
                }
                else
                        ex = 1;
        }
    }
    getch();
    closegraph();
}
```

ภาคผนวก ข
โปรแกรมตัวควบคุม MRAC

รายละเอียดของโปรแกรม MRAC.C

- เป็นโปรแกรมที่ใช้ควบคุมกับระบบจริงอันดับสองเพื่อให้ได้ผลตอบสนองตามที่ได้กำหนดไว้ในบทที่ 3
- โปรแกรมนี้ใช้ยืนยันผลของโปรแกรมจำลองในภาคผนวก ก และหาผลกระทบของข้อจำกัดของระบบที่ไม่เป็นไปตามทฤษฎี
- วิธีคำนวณค่าสัญญาณอ้างอิง ใช้วิธีที่กล่าวไว้ในบทที่ 2 โดยถือว่าความถี่การสุ่มสัญญาณสูงกว่าความถี่ของระบบมาก และใช้การคำนวณแบบเวลาต่อเนื่อง
- โปรแกรมนี้ใช้กับเครื่อง PTS-10 ของบริษัท Pacific และต้องมีการ์ด A/D PCL 812 และ D/A PCL 726 ต่อกับคอมพิวเตอร์ที่พอร์ตเบอร์ 220 และ 2C0 ตามลำดับ และวัดสัญญาณด้วยเครื่อง servocorder ของบริษัท Graphtec
- ต่อเอาท์พุทช่องที่ 0 ของ PCL 726 กับอินพุทของ PTS-10 ต่อเอาท์พุท y ของ PTS-10 เข้ากับอินพุทช่องที่ 0 ของ PCL 812 และ y' เข้ากับช่องที่ 1
- ต่อสัญญาณเริ่มทำงานของ PTS-10 เข้ากับอินพุทช่องที่ 2 ของ PCL 812 ต่อเอาท์พุทช่องที่ 1 ของ PCL 726 (y_m) และต่อเอาท์พุท y และ y' เข้ากับเครื่อง servocorder เพื่อวัดสัญญาณ
- เปิดเครื่อง PTS-10 ก่อน จากนั้นรันโปรแกรม เมื่อเริ่มให้ระบบทำงานกดปุ่ม start บน PTS-10 โปรแกรมจะตรวจจับสัญญาณเริ่มทำงานของ PTS-10 และทำงานไปพร้อมกันทันที เมื่อหยุดการทำงานของ PTS-10 ให้กดปุ่มเดิมกับตอนเริ่มทำงาน และโปรแกรมจะหยุดทำงานในขณะเดียวกันทันที
- การควบคุมจะใช้อินเทอร์รัพของไทม์เมอร์ของ PC ที่ตำแหน่ง 1C เป็นตัวกระตุ้นให้โปรแกรมทำงานเพื่อให้ได้ความถี่การสุ่มสัญญาณและการควบคุมคงที่ (18.2044 Hz) ดังนั้นเครื่องคอมพิวเตอร์ที่ใช้ทำงานควรทำงานเร็วมากพอ ผลที่นำออกแสดงในบทที่ 4 เวลาในการคำนวณประมาณ 1เปอร์เซ็นของคาบการสุ่มสัญญาณ หรือคาบของการอินเทอร์รัพ
- การแสดงผลจะแสดง เกนของตัวควบคุมปฐมภูมิที่ปรับตัว AK[0], [1], [2] และ ผลตอบสนองของกระบวนการ กับ ผลตอบสนองของรูปแบบอ้างอิงที่สอดคล้องกัน IN[0], MX[0] ตามลำดับ

โปรแกรมย่อย

void interrupt mrac(void)
โปรแกรม interrupt service routine (ISR) ซึ่งบรรจุโปรแกรมที่ทำหน้าที่เป็นตัวควบคุม MRAC

void installint(void)
เก็บค่า address ของอินเทอร์รัพ 1C เดิม และ นำ address ของ mrac() เข้าไปแทนที่

void model(void)
ให้ค่าเริ่มต้นกับรูปแบบอ้างอิง

void readme(void)
แสดงข้อความบอกคุณสมบัติโดยย่อของโปรแกรม

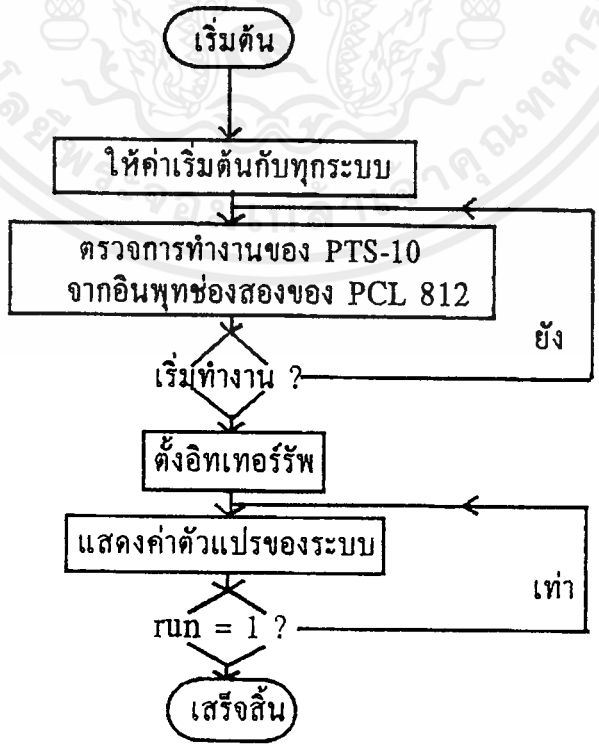
ตัวแปร

HIIN, LOIN, HIOUT, LOOUT	ค่าเลขฐานสองที่ใช้ส่งเข้า และ ออกจาก A/D และ D/A
IN	ค่าสัญญาณที่วัดเข้ามาแปลงจากตัวแปรข้างบนให้กลายเป็นค่าจริง ซึ่งก็คือสเตทของระบบ
OUT	ค่าสัญญาณควบคุมที่คำนวณได้
CT	สัญญาณควบคุมที่คำนวณได้และแปลงเป็นเลขฐานสองพร้อมที่จะส่งออกไป
MR	ค่าผลตอบสนองอ้างอิงที่แปลงเป็นเลขฐานสองพร้อมที่จะส่งไปยังเครื่องบันทึกผล
MG, MH, MX	เมตริกซ์พารามิเตอร์ และ สเตทของรูปแบบอ้างอิง
RK	ตัวแปรใช้ในการคำนวณ สเตทของรูปแบบอ้างอิงด้วยวิธี Runge-Kutta
AE, AG, AKD, AK	พารามิเตอร์ของตัวควบคุม MRAC
t, T	เวลาและคาบการสุ่มสัญญาณตามลำดับ
SET_P	สัญญาณอ้างอิงหรือ set point
run	ตัวแปรบอกถึงอยู่ในขณะทำงาน
start	ตัวแปรบอกถึงรอบแรกของการควบคุม
BASEIN	เบอร์พอร์ทของ PCL 812
BASEOUT	เบอร์พอร์ทของ PCL 726

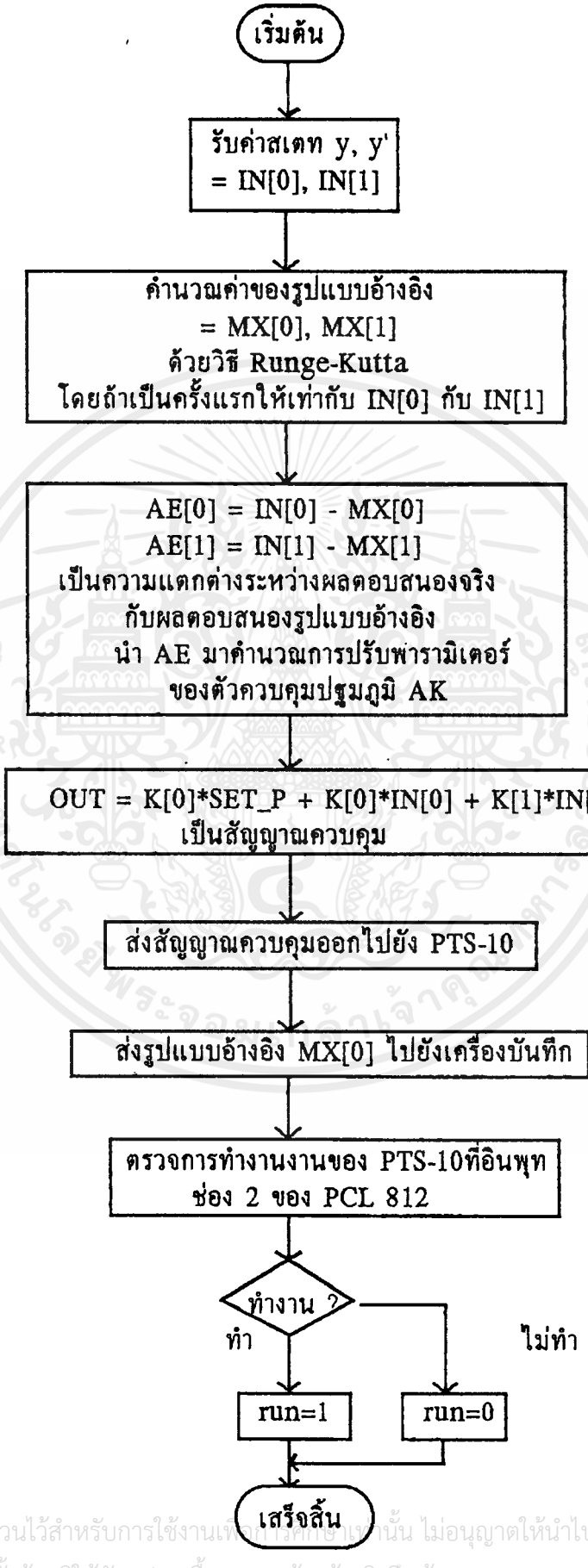
Flow Chart

การทำงานของโปรแกรมแบ่งเป็นสองส่วนคือ โปรแกรมหลัก และโปรแกรมอินเทอร์รัพ

โปรแกรมหลัก



โปรแกรมอินเทอร์รัพ (controller)



```
/*                        M R A C Controller (6/3/93)                    */
/* MRAC.C : This program is MRAC controller.                             */

#include<stdio.h>
#include<dos.h>
#include<math.h>
#include<conio.h>

#define ZERO 2048

void interrupt (*oldvec)();
void interrupt mrac(void);
void installint(void);
void model(void);
void readme(void);

volatile int start=1;
volatile int BASEIN = 0x220;
volatile int BASEOUT = 0x2C0;
volatile int CT, MR, run;
volatile unsigned char HIIN, LOIN, HIOUT, LOOUT;
volatile double t=0,T;
volatile double MG[4], MH[2], MX[2], AE[2], AG[3], AKD[3], AK[3], RK[2][4];
volatile double IN[2], OUT, SET_P, temp;

void interrupt mrac(void)
{
        /*Measure state variable of process X.*/
        outportb(BASEIN+10,0x0);
        outportb(BASEIN+11,0x1);
        outportb(BASEIN+12,0xff);
        do{
        HIIN = inportb(BASEIN+5);
        }while(HIIN>=16);
        LOIN = inportb(BASEIN+4);
        IN[0] = (HIIN*256 + LOIN - ZERO)*20.0/4095.0;

        /*Measure state variable of process X'.*/
        outportb(BASEIN+10,0x1);
        outportb(BASEIN+11,0x1);
        outportb(BASEIN+12,0xff);
        do{
        HIIN = inportb(BASEIN+5);
        }while(HIIN>=16);
        LOIN = inportb(BASEIN+4);
        IN[1] = (HIIN*256 + LOIN - ZERO)*20.0/4095.0;

        /*Calculate model reference state.*/
        if(start){
                MX[0] = IN[0];
                MX[1] = IN[1];
                start--;
        }
        else{
                RK[0][0] = T*MX[1];
                RK[1][0] = T*(MG[2]*MX[0] + MG[3]*MX[1] + MH[1]*SET_P);

                RK[0][1] = T*(MX[1] + RK[1][0]/2.0);
                RK[1][1] = T*(MG[2]*(MX[0]+RK[0][0]/2.0) + MG[3]*(MX[1]+RK[1][0]/2.0)
                        + MH[1]*SET_P);
```

```
        RK[0][2] = T*(MX[1] + RK[1][1]/2.0);
        RK[1][2] = T*(MG[2]*(MX[0]+RK[0][1]/2.0) + MG[3]*(MX[1]+RK[1][1]/2.0)
                   + MH[1]*SET_P);

        RK[0][3] = T*(MX[1] + RK[1][2]);
        RK[1][3] = T*(MG[2]*(MX[0]+RK[0][2]) + MG[3]*(MX[1]+RK[1][2])
                   + MH[1]*SET_P);

        MX[0] += (RK[0][0] + 2*RK[0][1] + 2*RK[0][2] + RK[0][3])/6.0;
        MX[1] += (RK[1][0] + 2*RK[1][1] + 2*RK[1][2] + RK[1][3])/6.0;
}

/*Adaptive Laws.*/
AE[0] = IN[0] - MX[0];
AE[1] = IN[1] - MX[1];

AE[0] = ((int)(AE[0]*100.0))/100.0;
AE[1] = ((int)(AE[1]*100.0))/100.0;

AKD[0] = -1.0*AG[0]*((4*AE[0]) + (17*AE[1]))*SET_P*T;
AKD[1] = -1.0*AG[1]*((4*AE[0]) + (17*AE[1]))*IN[0]*T;
AKD[2] = -1.0*AG[2]*((4*AE[0]) + (17*AE[1]))*IN[1]*T;

AK[0] += AKD[0];
AK[1] += AKD[1];
AK[2] += AKD[2];

/*Calculate MRAC control signal*/
OUT = AK[0]*SET_P + AK[1]*IN[0] + AK[2]*IN[1];
CT = (int)(4095*(OUT+10.0)/20.0);
if(CT > 4095)
        CT = 4095;
else if(CT < 0)
        CT = 0;
else
        ;

/*Out control signal (D/A).*/
HIOUT = CT / 256;
LOOUT = CT % 256;
outportb(BASEOUT+0, HIOUT);
outportb(BASEOUT+1, LOOUT);

/*Check for change level of set point*/
t += T;
if(t > 15){
        SET_P *= -1.0;
        t = 0;
}

/*Out model reference state to servocorder.*/
MR = (int)(4095*(MX[0]+10.0)/20.0);
HIOUT = MR / 256;
LOOUT = MR % 256;
outportb(BASEOUT+2, HIOUT);
outportb(BASEOUT+3, LOOUT);
```

```
        /*Check for end program.*/
        outportb(BASEIN+10,0x2);
        outportb(BASEIN+11,0x1);
        outportb(BASEIN+12,0xff);
        do{
                HIIN = inportb(BASEIN+5);
        }while(HIIN>=16);
        LOIN = inportb(BASEIN+4);
        run = HIIN*256 + LOIN;
}

void installint(void)
{
        oldvec = getvect(0x1C);
        setvect(0x1C, mrac);
}

void model(void)
{
        start = 1;
        SET_P = 0.5;

        T = (3600.0/65535.0);  /*Sampling time.*/
        MG[0] = 0;
        MG[1] = 1.0;
        MG[2] = -16.0;
        MG[3] = -4;

        MH[0] = 0;
        MH[1] = 16;

        AG[0] = 1;
        AG[1] = 1;
        AG[2] = 1;

        AK[0] = 16.0;
        AK[1] = -16.0;
        AK[2] = -2.0;
}

void readme(void)
{
        clrscr();
        printf("This program is MRAC controller for ");
        printf("PTS-10 process simulator (second order).\n");
        printf("Using with PCL 812 (A/D) and PCL 726 (D/A) card.\n");
        printf("There will be adaptation parameter showing on screen.\n");
        printf("(k0 , k1, k2, process variable, medel reference variable)\n");
        printf("Look for detail in the thesis.\n");
        printf("Open PTS-10 then press any key to start program.\n");
        getch();
}
```

```
main()
{
        readme();

        clrscr();

        /*Initialize parameter.*/
        model();

        /*Clear D/A buffer.*/
        outportb(BASEOUT+0, 0x08);
        outportb(BASEOUT+1, 0x00);

        printf("Press startkey (green button) of PTS-10 to start");
        printf("control system.\n");
        printf("Then, to stop program, press the same button.\n");

        /*Wait for start button signal.*/
        do{
                outportb(BASEIN+10,0x2);
                outportb(BASEIN+11,0x1);
                outportb(BASEIN+12,0xff);
                do{
                        HIIN = inportb(BASEIN+5);
                }while(HIIN>=16);
                LOIN = inportb(BASEIN+4);
                run = HIIN*256 + LOIN;
        }while(run >= 2060);

        installint();

        /*Print adaptation parameter.*/
        while(run < 2060)
                printf("%f, %f, %f, %f, %f\n", AK[0], AK[1], AK[2], IN[0], MX[0]);

        /*Clear D/A buffer.*/
        outportb(BASEOUT+2, 0x08);
        outportb(BASEOUT+3, 0x00);

        setvect(0x1C, oldvec);
}
```

เอกสารอ้างอิง

[1] Katsuhiko ogata, Modern Control Engineering, Prentice-Hall, Englewood Cliffs, 1990.

[2] Hans Butler, Model Reference Adaptive Control, Prentice-Hall, Marylands Avenue, 1992.

[3] Karl J Astrom & Bjorn W., Adaptive Control, Addison-Wesley, 1989

[4] Joseph J. Distefano, Theory and ProBlems of Feedback and Control systems, McGraw-Hill Book Co., Singapore, 1976