ปีการศึกษา 2530 ภาคการเรียนที่ 2

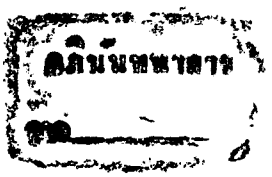
ตัวควบคุมแบบจูนปรับตัวเอง
(Self-tuning controller)

โดย

ธัญญวัฒน์ อภิรัตน์วงศา
พรเทพ ปรางค์ปรักมะ

อาจารย์ที่ปรึกษา

อ.วันชัย ริ้วรุจา



ปริญญานิพนธ์ปีการศึกษา 2530 ภาคเรียนที่ 2
ภาควิชา วิศวกรรมระบบควบคุม
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง ตัวควบคุมแบบจูนปรับตัวเอง
(Self-tuning controller)

ผู้จัดทำ
1. นายธัญญวัฒน์ อภิรัตน์วงศา
2. นายพรเทพ ปรางค์ปรักมะ

.....วันชัย ริ้วรุจา.....อาจารย์ที่ปรึกษา
(อ.วันชัย ริ้วรุจา)

ตัวควบคุมแบบจูนปรับตัวเอง

ธัญญวัฒน์ อภิรัตน์วงศา
พรเทพ ปรางค์ปรักมะ
อ.วันชัย ริ้วรุจา อาจารย์ที่ปรึกษา
ปีการศึกษา 2530

บทคัดย่อ

สมรรถนะของการควบคุมกระบวนการที่ใช้ในทางปฏิบัติจะเลวลงเมื่อกระบวนการมีการเปลี่ยนแปลงพารามิเตอร์หรือมีสิ่งรบกวนเกิดขึ้นภายในกระบวนการปริญญานิพนธ์นี้เป็นการพัฒนาตัวควบคุมโดยใช้การควบคุมแบบจูนปรับตัวเองกับกระบวนการดังกล่าวข้างต้นเพื่อให้การควบคุมมีสมรรถนะดีขึ้น การควบคุมระบบใช้ไมโครคอมพิวเตอร์เป็นตัวควบคุมโดยอาศัยการประมาณค่าพารามิเตอร์แบบรีเคอร์ซีฟลีสสแควร์ และอัลกอลิทึมการควบคุมแบบกำหนดโพล.

SELF TUNING CONTROL

Ponthep Prankprakma
Thanyawat Abhiratanawongsa
Wanchai Ruiruja Advisor
1987

Abstract

The performance of a self-tuning controller is studied, when the controller is used to control a system with unknown constant or slowly varying parameters. The thesis describes the development of a self-tuning control of the system to achieve better performance. Implementation of recursive least-squares parameter estimation and control with pole-placement approach on 16-bit microcomputer to on-line control of a model plant is illustrated.

สารบัญ

สารบัญรูปภาพ

หน้า

บทที่ 1
บทนำ

ในบทนี้จะกล่าวถึงปัญหาของระบบควบคุมกับการควบคุมแบบจูนปรับตัวเอง จุดประสงค์และขอบเขตของปริญญานิพนธ์ ตลอดจนวิธีดำเนินงานของปริญญานิพนธ์

1.1 ปัญหาของระบบควบคุม

ในวงการอุตสาหกรรมได้ให้ความสนใจกับตัวควบคุมที่สามารถปรับค่าได้เองอย่างอัตโนมัติมาเป็นเวลานานแล้ว เนื่องจากการปรับค่าของพารามิเตอร์ของตัวควบคุมโดยใช้ความสามารถของคนทำได้ช้าและไม่แน่นอน ซึ่งจะทำให้ประสิทธิภาพของการควบคุมลดลง โดยปกติในระบบที่มีพลวัติ (Dynamic) ซับซ้อน ตัวอย่างเช่น ระบบที่พารามิเตอร์(Parameter) เปลี่ยนแปลงตามเวลา หรือมีค่าที่ไม่ทราบ (Unknown) หรือระบบมีสิ่งรบกวน (Disturbance) เกิดขึ้น เป็นต้น การออกแบบตัวควบคุมเพื่อใช้ในการควบคุมระบบดังกล่าวให้เป็นไปตามข้อกำหนด (Specification) ทำได้ไม่สะดวกนักเพราะค่าพารามิเตอร์ของระบบจะเปลี่ยนแปลงตลอดเวลาทำให้ไม่สามารถทราบค่าที่แน่นอนได้ การออกแบบระบบควบคุมโดยใช้การควบคุมแบบจูนปรับตัเอง (Self-tuning control) เป็นวิธีหนึ่งที่ใช้ได้ดีกับระบบที่ไม่แน่นอน (Uncertainty) เหล่านี้ สามารถปรับค่าของพารามิเตอร์ของต้วควบคุมได้อย่างอัตโนมัติ และยังสามารถที่จะเพิ่มสมรรถนะ (Performance) ของการควบคุมระบบอีกด้วย ตัวควบคุมแบบจูนปรับตัวเอง (Self-tuning controller) เป็นตัวควบคุมแบบปรับตัวเอง (Adaptive controller) ที่ง่ายที่สุด อาศัยการบอกเอกลักษณ์ของพารามิเตอร์ (Parameter identification) กับกฎการควบคุมแบบป้อนกลับ (Feedback control law) ทำให้อัลกอริทึมของการจูนปรับตัวเอง (Self-tuning algorithm) ง่ายและสะดวกต่อการใช้งาน ถึงแม้การควบคุมแบบจูนปรับตัวเองจะไม่ได้ค่าเล็งเลิศ (Optimal) แต่สามารถใช้ควบคุมกระบวนการที่มีการเปลี่ยนแปลงเกิดขึ้นได้อย่างมีประสิทธิภาพและให้สมรรถนะที่เหนือกว่าการควบคุมแบบดั้งเดิม (Conventional control)

การนำตัวควบคุมแบบจูนปรับตัวเองมาประยุกต์ใช้งานนั้น ในบางกรณีอาจจะไม่สามารถใช้ควบคุมระบบได้ เนื่องจากยังไม่มีทฤษฎีรับรองว่าอัลกอริทึมของการจูนปรับตัวเองจะลู่เข้าเสมอ (Converge) มีทฤษฎีที่รับรองเพียงว่าเมื่ออัลกอริทึมทั้งหมดของการควบคุมแบบจูนปรับตัวเองสามารถลู่เข้าแล้วผลตอบของตัวควบคุม (Control action) ก็จะลู่เข้าด้วยความไม่สมบูรณ์ของการควบคุม

แบบจูนปรับตัวเองนั้นมีอีกหลายประการ เช่น อัตราการลู่เข้า (Convergence rate) ที่ไม่แน่นอนทำให้ไม่ทราบว่าระบบจะลู่เข้าเมื่อไร หรือแม้ว่าการควบคุมจะอยู่ภายใต้เงื่อนไขที่เหมาะสมแล้ว แต่การควบคุมก็อาจจะลู่ออก (Diverge) ก็ได้ การควบคุมแบบจูนปรับตัวเองส่วนใหญ่ใช้ได้ดีกับระบบเชิงเส้น (Linear system) เนื่องจากใช้การประมาณพารามิเตอร์แบบเชิงเส้น แต่ก็สามารถใช้กับระบบที่ไม่เป็นเชิงเส้นได้ถ้ามีความไม่เป็นเชิงเส้น (Non-linearity) ที่ไม่ยุ่งยากนักได้

แนวเหตุผลที่การควบคุมแบบจูนปรับตัวเองได้รับความนิยมเนื่องจากใช้ได้ดีกับระบบสโตแคสติกแบบเวลาไม่ต่อเนื่อง (Discrete time stochastic system) ซึ่งสามารถนำไปประยุกต์ใช้กับไมโครคอมพิวเตอร์ได้สะดวก นอกจากนี้แล้วการควบคุมแบบจูนปรับตัวเองยังช่วยลดความยุ่งยากในการออกแบบระบบควบคุมที่ซับซ้อนต่างๆ ซึ่งทำไม่ได้ด้วยการออกแบบระบบควบคุมโดยทั่วไป และช่วยเพิ่มประสิทธิภาพในการปรับค่าพารามิเตอร์ได้ดีกว่าการปรับโดยวิธีที่ใช้กันในทางปฏิบัติทั่วๆไป

1.2 <u>จุดประสงค์ของปริญญานิพนธ์</u>

สำหรับในปริญญานิพนธ์นี้จะศึกษาปัญหาของการควบคุมและพัฒนาการควบคุมแบบจูนปรับตัวเองในระบบที่มีการเปลี่ยนแปลงพารามิเตอร์อย่างช้าๆ ตามเวลา หรือมีสิ่งรบกวนเกิดขึ้น ขณะที่ระบบอยู่ในสภาวะคงตัว (Steady state) เพราะเมื่อระบบเป็นเช่นนั้นแล้วย่อมทำให้สมรรถนะของระบบเลวลง ซึ่งการปรับปรุงแก้ไขสมรรถนะให้ดีขึ้นจะต้องปรับค่าพารามิเตอร์ของตัวควบคุมใหม่ให้เหมาะสมกับการเปลี่ยนแปลง การนำตัวควบคุมแบบจูนปรับตัวเองมาประยุกต์ใช้กับระบบที่มีปัญหาข้างต้น จะสามารถแก้ไขและปรับปรุงสมรรถนะของการควบคุมได้อย่างมีประสิทธิภาพ โดยคุณสมบัติของการควบคุมแบบจูนปรับตัวเองเมื่อระบบเกิดการเปลี่ยนแปลงขึ้น ตัวควบคุมแบบจูนปรับตัวเองจะปรับค่าพารามิเตอร์ของตัวควบคุมอย่างอัตโนมัติเพื่อรักษาผลตอบของระบบให้มีค่าตามที่ตั้งไว้ (Set point) และรักษาสมรรถนะของระบบตามกฎเกณฑ์ของการควบคุม (Criteria)

สำหรับปริญญานิพนธ์นี้จะใช้ตัวควบคุมแบบจูนปรับตัวเองแบบกำหนดโพล (Self-tuning controller using pole placement) ซึ่งอาศัยวิธีการกำหนดโพลทำให้สามารถกำหนดลักษณะของผลตอบของการควบคุมได้ตามต้องการ ถึงแม้ผลที่ได้จะไม่เป็นค่าที่เล็งเลิศ แต่การควบคุมด้วยวิธีนี้สามารถใช้กับระบบที่ไม่มีเฟสน้อยสุด (Non-minimum phase system) หรือระบบที่มีเวลา

หน่วง (Delay time) เปลี่ยนแปลงตามเวลาซึ่งการควบคุมแบบเล็งเลิศไม่สามารถควบคุมได้ วิธีนี้จึงครอบคลุมปัญหาในการควบคุมได้มากกว่าเป็นวิธีที่น่าสนใจ

1.3 ขอบเขตของปริญญานิพนธ์

1. พัฒนาตัวควบคุมแบบจูนปรับตัวเองที่ใช้การกำหนดโพล เพื่อควบคุมระบบที่มีการเปลี่ยนแปลงพารามิเตอร์ช้า ๆ ตามเวลา หรือมีสิ่งรบกวนเกิดขึ้นขณะระบบอยู่ในภาวะคงตัว
2. ทดสอบตัวควบคุมที่ออกแบบกับระบบจริง

1.4 วิธีดำเนินงานของปริญญานิพนธ์

สำหรับลำดับขั้นตอนของปริญญานิพนธ์เป็นดังนี้

บทที่ 2 กล่าวถึงทฤษฎีและหลักการของการควบคุมแบบจูนปรับตัวเองที่ใช้การกำหนดโพล

บทที่ 3 การออกแบบและการสร้าง

บทที่ 4 การทดลองใช้ตัวควบคุมที่ออกแบบขึ้นกับระบบจริง

บทที่ 5 บทวิจารณ์และสรุป

บทที่ 2

ทฤษฎีและหลักการ

การควบคุมแบบจูนปรับตัวเองเป็นการควบคุมแบบจูนอัตโนมัติ สามารถประยุกต์ใช้ได้กับการควบคุมชนิดต่าง ๆ ได้โดยง่าย และใช้ได้กับระบบหลายอินพุตหลายเอาต์พุต (Multi-input multi-output) ในบทนี้จะกล่าวถึงลักษณะของการควบคุมแบบจูนปรับตัวเองที่ใช้การกำหนดโพลซึ่งใช้กับระบบหนึ่งอินพุตหนึ่งเอาต์พุต (Single-input single-output)

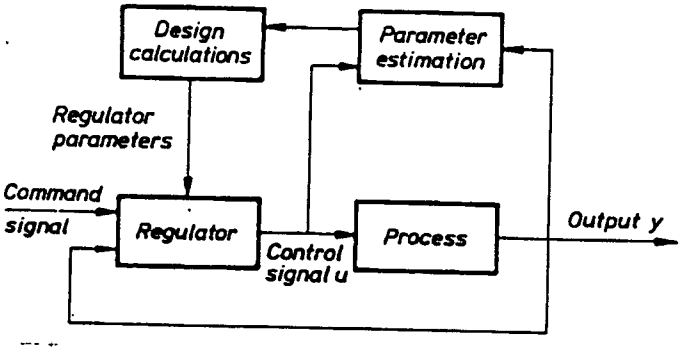
2.1 ลักษณะของการควบคุมแบบจูนปรับตัวเอง

ในการควบคุมระบบที่มีความไม่แน่นอน เนื่องจากมีการเปลี่ยนแปลงค่าพารามิเตอร์อย่างช้า ๆ ตามเวลาหรือมีสิ่งรบกวนเกิดขึ้น ให้มีสมรรถนะและข้อกำหนดเป็นไปตามที่ต้องการนั้น วิธีการควบคุมแบบจูนปรับตัวเองเป็นวิธีหนึ่งที่ใช้ได้ดีสำหรับระบบดังกล่าวและสามารถใช้กับระบบที่มีลักษณะดังนี้

1. ระบบที่มีพลวัติ (Dynamic) ยุ่งยาก เช่นระบบที่มีเฟสตามหลังมาก ๆ
2. ระบบที่มีภาวะไม่เชิงเส้น
3. ระบบที่มีพารามิเตอร์เปลี่ยนแปลงตามเวลา
4. มีการกระทำต่อกัน (Interaction) ระหว่างลูปในระบบ
5. มีสิ่งรบกวนจากสภาวะแวดล้อม (Environmental disturbance) มาเกี่ยวข้อง

จุดประสงค์ของการควบคุมแบบจูนปรับตัวเองก็เพื่อต้องการควบคุมผลตอบของระบบที่มีลักษณะข้างต้นให้เป็นไปตามข้อกำหนดที่วางไว้ โดยการปรับค่าพารามิเตอร์ของตัวควบคุมให้เหมาะสมกับการเปลี่ยนแปลงที่เกิดขึ้นอย่างอัตโนมัติตามกฎเกณฑ์ของการควบคุม ในการควบคุมจะต้องพิจารณาเสถียรภาพ สมรรถนะ และการลู่เข้าเป็นหลัก โดยทั่วไปแล้วการควบคุมแบบจูนปรับตัวเองจะมีโครงสร้างดังรูป 2.1

จากรูปจะเห็นได้ว่าการควบคุมแบบจูนปรับตัวเองประกอบด้วย 2 ลูป ลูปภายในประกอบด้วยตัวกระบวนการ (Process) และตัวควบคุมป้อนกลับเชิงเส้น (Linear-feedback regulator) ส่วนลูปนอกประกอบด้วยขั้นตอนการประมาณค่าพารามิเตอร์ และการคำนวณออกแบบ (Design calculations) ซึ่งจะทำการปรับค่าพารามิเตอร์ของตัวควบคุม



รูปที่ 2.1 โครงสร้างของการควบคุมแบบจูนปรับตัวเอง

การควบคุมแบบจูนปรับตัวเองสามารถแบ่งได้เป็นสองแบบคือ แบบเอ็กพลิสิต (Explicit) กับแบบอิมพลิสิต (Implicit) โดยที่แบบเอ็กพลิสิตจะประมาณค่าของพารามิเตอร์ของระบบแล้วแทนในขั้นตอนการคำนวณเพื่อหาค่าสัมประสิทธิ์ที่ใช้ในกฎการควบคุม ส่วนแบบอิมพลิสิต จะข้ามช่วงการคำนวณออกแบบไปโดยจัดสมการเสียใหม่แล้วทำการประมาณค่าสัมประสิทธิ์ของตัวควบคุมโดยตรง ทำให้รวดเร็วขึ้นแต่ทว่าความแม่นยำ (Accuracy) และความมั่นคง (Robust) ของแบบเอ็กพลิสิต ดีกว่า

ในการควบคุมแบบจูนปรับตัวเองจำเป็นต้องอาศัยค่าจากการบอกเอกลักษณ์หรือการประมาณค่าพารามิเตอร์เป็นหลัก ดังนั้นการเลือกใช้อัลกอริทึมในการประมาณ (Estimation algorithm) จะต้องพิจารณารูปแบบของระบบ (System model) และขั้นตอนการออกแบบ (Design procedure) ถ้าใช้อัลกอริทึมที่ขาดความมั่นคง จะทำให้การควบคุมแบบจูนปรับตัวเองล้มเหลวได้ การประมาณค่าพารามิเตอร์ที่ใช้ในอัลกอริทึมของการควบคุมแบบจูนปรับตัวเองเป็นแบบเชิงเส้น (Linear estimation) ดังนั้นการควบคุมแบบจูนปรับตัวเองไม่สามารถใช้กับกระบวนการที่มีความไม่เป็นเชิงเส้นยุ่งยากมาก ๆได้

2.2 รูปแบบและปัญหาของการควบคุมแบบจูนปรับตัวเอง

โดยทั่วไปรูปแบบของระบบสามารถแทนได้ในรูปของเวลาไม่ต่อเนื่อง (Discrete time) ได้ดังนี้

$$A(z^{-1})y(k) = z^{(-d)}B(z^{-1})u(k) + x(k) \quad \ldots(2.2.1)$$

โดยที่

$u(k), y(k)$ เป็น อินพุตและเอาต์พุตของระบบที่ช่วงเวลาการสุ่มที่ k

$A(z^{-1}), B(z^{-1})$ เป็นโพลีโนเมียลที่สอดคล้องกับอินพุตและเอาต์พุตของระบบ

$x(k)$ เป็นสิ่งรบกวนในระบบที่ช่วงเวลาการสุ่มที่ k

การแทนรูปแบบของระบบนี้ x(k) เป็นสิ่งรบกวนที่เกิดขึ้นภายในระบบจึงสามารถพิจารณาได้หลายรูปแบบ ตัวอย่างเช่น

1. เป็นค่าคงตัว (Constant)
2. เป็นสิ่งรบกวนเนื่องจากโหลด (Load disturbance)
3. สิ่งรบกวนที่สามารถวัดได้ (Measurement disturbance)
4. สิ่งรบกวนที่มีค่าเฉลี่ยไม่เป็น 0 (Non zero mean disturbance)
5. สิ่งรบกวนที่มีค่าเฉลี่ยเป็น 0 (Zero mean disturbance)

การแทนรูปแบบของระบบให้ใกล้เคียงความเป็นจริงนั้น จะต้องเลือกเงื่อนไขของ x(k) ให้เหมาะสมกับลักษณะของกระบวนการมากที่สุด ตัวอย่างเช่น

$$x(k) = c + (C(z^{-1}) / A(z^{-1}))e(k)$$

โดยที่

c - ค่าคงที่

e(k) - สัญญาณรบกวนของตัวแปรอิสระที่มีค่าเฉลี่ยเป็น 0

แต่ปกติมักจะแทนรูปแบบของระบบด้วย

$$A(z^{-1})y(k) = z^{-d} B(z^{-1})u(k) + c(z^{-1})e(k) \quad ...(2.2.2)$$

ปัญหาการแทนรูปแบบของระบบมีความสำคัญมากเนื่องจากในอัลกอรีทึมของการควบคุมแบบจูนปรับตัวเองต้องอาศัยพารามิเตอร์ของระบบในการควบคุม ในกรณีที่แทนรูปแบบของการบอกเอกลักษณ์ไม่เหมาะสมก็จะเป็นส่วนหนึ่งที่ทำให้การควบคุมแบบจูนปรับตัวเองล้มเหลว นอกจากปัญหาของการกำหนดรูปแบบซึ่งเป็นปัญหาเกี่ยวกับเสถียรภาพแล้ว ปัญหาที่สำคัญอีกประการหนึ่งได้แก่ปัญหาของสมรรถนะของการควบคุม ปัญหาที่เกิดขึ้นเสมอคือ ปัญหาออฟเซ็ต (Offset) ซึ่งสาเหตุของการเกิดขึ้นกับลักษณะของการควบคุมและสิ่งรบกวนที่เกิดขึ้นในระบบ ตัวอย่างเช่น สิ่งรบกวนที่มีค่าเฉลี่ยไม่เป็น 0 ก็เป็นสาเหตุสำคัญที่ทำให้เกิดออฟเซ็ต

ชนิดของการประมาณมีทั้งแบบไบอัส (Bias) และ ไม่ไบอัส (Unbias) ซึ่งใช้กับข้อมูลที่มีค่าเฉลี่ยไม่เป็น 0 และเป็น 0 ตามลำดับ สำหรับกรณีไม่ไบอัสเนื่องจากสิ่งรบกวนมีค่าเฉลี่ยเป็น 0 เราจะได้ว่า $C(z^{-1})$ ในสมการ (2.2.2) มีค่าเป็น 1 ดังนั้นเราจึงไม่จำเป็นต้องทำการประมาณค่าพารามิเตอร์ของสิ่งรบกวน

วิธีการประมาณที่นิยมใช้มีอยู่ 3 วิธี คือ

1. RLS (Recursive least square) ซึ่งเป็นการประมาณแบบไม่ไบอัส
2. RML (Recurcive maximum likelihood) ซึ่งเป็นการประมาณแบบไบอัส

3. ELS (Extended least square) ซึ่งเป็นการประมาณแบบไม่ไบอัสที่ดัดแปลงจาก RLS

เนื่องจากในปริญญานิพนธ์นี้เราเลือกใช้ตัวควบคุมแบบจูนปรับตัวเองที่ใช้การกำหนดโพล ซึ่งเป็นวิธีที่สามารถกำหนดข้อจำกัด (Constraint) ให้กับผลตอบของระบบได้ และมีความมั่นคงสูง ดังนั้นการเปลี่ยนแปลงพารามิเตอร์ของสิ่งรบกวนจึงมีผลต่อระบบน้อย เราจึงสามารถใช้วิธีการประมาณแบบ RLS ได้

2.3 การประมาณแบบ RLS

พิจารณาระบบจากสมการ 2.2.2 เมื่อประมาณสัญญาณรบกวนให้เป็นแบบไม่มีความสัมพันธ์กับระบบ (uncorrelated noise) $C(z^{-1}) = 1$ และไม่มีเวลาหน่วง $d = 0$ จะได้

$$A(z^{-1})y(k) = B(z^{-1})u(k) + e(k)$$

เมื่อ $z^{-1}\ y(k) = y(k-1)$

$$A(z^{-1}) = 1 + a_1 z^{-1} + \ldots + a_n z^{-n}$$

$$B(z^{-1}) = b_1 z^{-1} + \ldots + b_n z^{-n}$$

จะได้ว่า

$$y(k) = -a_1 y(k-1) - a_2 y(k-2) \ldots - a_n y(k-n) + b_1 u(k-1) + b_2 u(k-2) \ldots + b_n u(k-n) + e(k)$$

นั่นคือ

$$y(k) = O^T X + e$$

$$X(k) = [-y(k-1) \ldots -y(k-n)\ u(k-1) \ldots u(k-n)]^T$$

$$O^T = [\, a_1 \ \ldots \ a_n \ \ b_1 \ \ldots \ b_n \,]$$

เราต้องการประมาณค่าเวคเตอร์ O ซึ่งมีขนาดเท่ากับ m ดังนั้นเราจะต่องมีจำนวนเซ็ตของการวัด r (number of sets of measurement) หรือจำนวนคู่ของข้อมูล u,y ก่อนที่จะทำการประมาณค่าอย่างน้อย m เซ็ตหรือคู่ $r >= m$ ดังนั้นการประมาณค่าจะให้ผลหลังจากการสุ่มข้อมูลไปแล้วอย่างน้อย m+1 ครั้ง โดยที่ $\hat{O}$ จะต้องทำให้ดัชนีราคา (cost index) J(r) น้อยที่สุด

$$J(r) = \sum_{k=1} q(k)[y(k) - \hat{O}^T(r)X(k)]^2$$

นั่นคือ $\hat{O}(r)$ ต้องสอดคล้องกับ : $\dfrac{\partial J(r)}{\partial \hat{O}(r)} = 0$

เมื่อ $\hat{O}$ เป็นเวคเตอร์ที่ได้จากการประมาณค่าเวคเตอร์ O จะได้ว่า

$$\left[\sum_{k=1}^{r} q(k)\, X(k)\, X^{T}(k)\right]\hat{O}(r) = \sum_{k=1}^{r} q(k)y(k)X(k) \quad ...(2.3.1)$$

กำหนด

$$P^{-1}(r) = \sum_{k=1}^{r} q(k)[X(k)X^{T}(k)] \quad ...(2.3.2)$$

$P^{-1}(r)$ จะสามารถหาอินเวอร์สได้ต่อเมื่อ $r >= m$

ดังนั้นสมการ 2.3.1 สามารถจัดรูปใหม่ได้เป็น

$$P^{-1}(r)\hat{O}(r) = q(k)y(k)X(k)$$

หรือ $$\hat{O}(r) = P(r)\sum_{k=1}^{r} q(k)y(k)X(k) \quad ... \quad (2.3.3)$$

จากสมการ 2.3.3 จะจัดใหม่ได้เป็น

$$P^{-1}(r)\hat{O}(r) = \sum_{k=1}^{r-1} q(k)y(k)X(k) + q(r)y(r)X(r) \quad ...(2.3.4)$$

จากสมการ 2.3.1 จะได้

$$\sum_{k=1}^{r-1} q(k)y(k)X(k) = \left[\sum_{k=1}^{r-1} q(k)X(k)X^{T}(k)\right]\hat{O}(r-1) \quad ...(2.3.5)$$

แทนค่าสมการ 2.3.5 ลงใน 2.3.4 จะได้

$$P^{-1}(r)\hat{O}(r) = \left[\sum_{k=1}^{r-1} q(k)X(k)X^{T}(k)\right]\hat{O}(r-1) + q(r)y(r)X(r) \quad ...(2.3.6)$$

ทำการบวกและลบด้านขวามือของสมการ 2.3.6 ด้วย $q(r)X(r)X^{T}O(r-1)$ จะได้

$$P^{-1}(r)\hat{O}(r) = \left[\sum_{k=1}^{r-1} q(k)X(k)X^{T}(k)\right]\hat{O}(r-1) + q(r)X(r)[y(r) - X^{T}\hat{O}(r-1)] + q(r)X(r)X^{T}(r)\hat{O}(r-1)$$

$$= \left[\sum_{k=1}^{r} q(k)X(k)X^{T}(k)\right]\hat{O}(r-1) + q(r)X(r)[y(r) - X^{T}(r)\hat{O}(r-1)] \quad ...(2.3.7)$$

จากคำจำกัดความของ $P^{-1}(r)$ ในสมการ 2.3.2 สมการ 2.3.7 จะเปลี่ยนรูปเป็น

$$P^{-1}(r)\hat{O}(r) = P^{-1}\hat{O}(r-1) + q(r)X(r)[y(r) - X^{T}(r)\hat{O}(r-1)] \quad ...(2.3.8)$$

จะได้ว่า

$$\hat{O}(r) = \hat{O}(r-1) + P(r)q(r)X(r)[y(r) - X^{T}\hat{O}(r-1)] \quad ...(2.3.9)$$

ดังนั้น $\hat{O}(r)$ จะหาได้จากค่าสที่ประมาณก้อนหน้านี้ $\hat{O}(r-1)$ และจากค่าของข้อมูลที่อ่านเข้ามา $y(r), X(r)$ และค่าความสำคัญ $q(r)$

จากสมการ 2.3.2 จะได้ว่า

$$P^{-1}(r) = \sum_{k=1}^{r-1} q(k)[X(x)X^{T}(k)] + q(r)X(r)X^{T}(r)$$

$$= P^{-1}(r-1) + q(r)X(r)X^{T}(r) \quad ...(2.3.10)$$

จากสมการ 2.3.10 เราจะต้องหาอินเวอร์สของเมตริกซ์ $P(r)$ และต้องรู้เมตริกซ์เริ่มต้น (initial matrix) P_o แต่อย่างไรก็ตามเนื่องจากการหา

อินเวอร์สเมตริกซ์จะเสียเวลาในการคำนวณมากเราจึงหลีกเลี่ยงโดย

กำหนด $H(r) = \sqrt{q(r)}\ X(r)$ ...(2.3.11a)

ดังนั้น $H(r)H^T(r) = q(r)X(r)X^T(r)$ ...(2.3.11b)

จากนั้นคูณสมการ 2.3.10 ตลอดด้วย $P(r)$ เข้าทางซ้าย จะได้

$$I = P(r)P^{-1}(r-1) + P(r)H(r)H^T(r) \quad ...(2.3.12)$$

คูณสมการ 2.3.12 ด้วย $P(r-1)$ ทางขวาจะได้

$$P(r-1) = P(r) + P(r)H(r)H^T(r)P(r-1)$$

แต่เนื่องจากค่าพารามิเตอร์ที่เราจะทำการประมาณมีการเปลี่ยนแปลงตามเวลา ค่าผิดพลาดที่เกิดขึ้นจากการประมาณครั้งก่อน ๆ จะมีความสำคัญน้อยกว่าค่าผิดพลาดที่เกิดขึ้นในการประมาณครั้งนี้ ซึ่งจะมีผลต่อการประมาณครั้งต่อไป ดังนั้นการที่เราให้ความสำคัญของข้อมูลทั้งเก่าและใหม่เท่ากันจึงไม่เป็นการถูกต้อง เราจึงต้องมีการปรับปรุงค่าโดยกำหนดค่าแฟคเตอร์การลืม (forgetting factor) , $0 <= \lambda <= 1$ เพื่อลดค่าความสำคัญของข้อมูลเดิมลงไปเรื่อย ๆ ทุกครั้งที่มีการประมาณ ดังนั้นค่าดัชนีราคาใหม่จะเป็นดังสมการ

$$J(r) = \sum_{k=1}^{r} \lambda^{r-k} [y(k) - \hat{O}^T(r)X(k)]^2$$

และจะได้ว่า

$$P(r-1) = P(r) + P(r)H(r)H^T(r)P(r-1) \quad ...(2.3.13)$$

คูณทางขวามือด้วย $H(r)$ จะได้

$$P(r-1)H(r) = P(r)H(r)[\lambda + H^T(r)P(r-1)H(r)] \quad ...(2.3.14)$$

คูณสมการ 2.3.14 ด้วย $[\{\lambda + H^T(r)P(r-1)H(r)\}^{-1} H^T(r)P(r-1)]$ เข้าทางขวา

$$P(r-1)H(r)\{\lambda + H^T(r)P(r-1)H(r)\}^{-1} H^T(r)P(r-1)$$
$$= P(r)H(r)H^T(r)P(r-1) \quad ...(2.3.15)$$

แทนค่า $P(r)H(r)H^T(r)P(r-1)$ จากสมการ 2.3.13 จะได้

$$P(r-1)H(r)\{\lambda + H^T(r)P(r-1)H(r)\}^{-1} H(r)P(r-1)$$
$$= P(r-1) - \lambda P(r) \quad ...(2.3.16)$$

จัดรูปใหม่จะได้

$$P(r) = \frac{P(r-1)[1 - H(r)\{\lambda + H^T P(r-1)H(r)\}^{-1} H^T(r)P(r-1)]}{\lambda} \quad ...(2.3.17)$$

จะเห็นได้ว่า เนื่องจาก $\{\lambda + H^T(r)P(r-1)H(r)\}$ เป็นค่าสเกลาร์จึงไม่ต้องหาอินเวอร์สเมตริกซ์เลยทำให้การประมาณค่าเป็นไปอย่างรวดเร็ว

สำหรับค่าประมาณเริ่มต้นของ P สามารถเลือกได้ตามใจชอบ แต่อย่างไรก็ตามการเลือกค่าเริ่มต้นที่เหมาะสมจะทำให้การคำนวณลู่เข้ารวดเร็วขึ้น

2.4 ตัวควบคุมแบบจูนปรับตัวเองชนิดกำหนดโพล

จากสมการ 2.2.2 เมื่อเราประมาณตัดค่า (neglect) สัญญาณรบกวนและเวลาหน่วงออกไป สมการจะลดรูปเหลือ

$$A(z^{-1})y(k) = B(z^{-1})u(k) \qquad ...(2.4.1)$$

เมื่อ u เป็นสัญญาณควบคุม และ y เป็นเอาต์พุตที่วัดได้

ให้ทรานสเฟอร์ฟังก์ชั่นจากสัญญาณคำสั่ง (Command signal) u_c ไปยังสัญญาณเอาต์พุตที่มีรูปแบบดังต้องการเป็น

$$H_m = B_m / A_m \qquad ...(2.4.2)$$

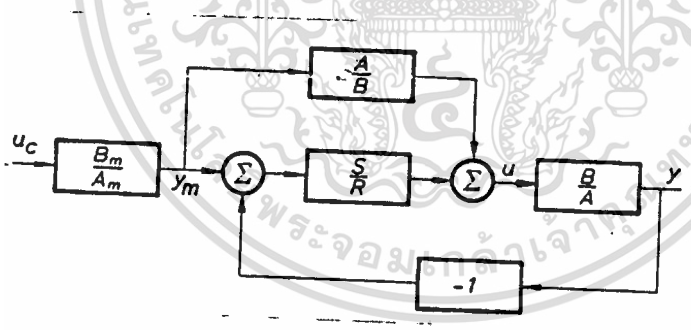
สำหรับสิ่งรบกวนที่เราประมาณตัดทิ้งไปในตอนแรกนั้นเราจะแทนทางอ้อมในรูปออบเซิฟเวอร์โพลีโนเมียล (Observer polynomial) A_o ซึ่งเป็นข้อจำกัด (Constraint) ของ รูปแบบ H_m ที่ต้องการ

กฎการควบคุม u จะอยู่ในรูป

$$Ru = Tu_c - Sy \qquad ...(2.4.3)$$

หรือ $u = (T/R)u_c - (S/R)y$

ดังรูป 2.4.1



รูป 2.4.1 บล็อกไดอะแกรมของกฎการควบคุม u

จากสมการ 2.4.1 และ 2.4.2 จะได้ความสัมพันธ์ระหว่างอินพุตและเอาต์พุตของระบบดังนี้

$$(AR + BS)y = BTu_c \qquad ...(2.4.4)$$

จะได้ว่าความสัมพันธ์นี้จะเท่ากับสมการ 2.4.2

$$BT / (AR + BS) = B_m / a_m \qquad ...(2.4.5)$$

เราต้องหาโพลีโนเมียล R,S,T ที่สอดคล้องกับสมการ 2.4.5 ก่อนแล้วนำกลับไปแทนหา u จาก สมการ 2.4.3

โดยที่ deg R >= deg T ...(2.4.6)

deg R >= deg S ...(2.4.7)

สัญญาณควบคุม u จึงจะถูกต้องสมเหตุผล (causal)

การกำจัดโพลและซีโร

จากสมการ 2.4.5 โพลของระบบลูปปิดจะเป็นคำตอบของสมการคุณสมบัติ (Characteristic equation)

$$AR + BS = 0$$

ซีโรของระบบลูปปิดเป็นซีโรของโพลีโนเมียล B และ T ซึ่งโดยปกติแล้วอันดับ(order) ของระบบจะสูงกว่าอันดับของรูปแบบที่ต้องการ

เมื่อพิจารณาซีโรของระบบลูปเปิด ซึ่งก็คือซีโรของโพลีโนเมียล B ถ้าแฟคเตอร์ของ B ไม่เป็นแฟคเตอร์ของ B_m แล้วก็จะเป็นแฟคเตอร์ของ AR + BS ดังนั้นเราจึงต้องกำจัดทิ้ง (cancel) โดยอาศัยโพลของระบบลูปปิด เนื่องจากระบบลูปปิดจะต้องมีเสถียรภาพและมีเพียงซีโรที่เสถียรเท่านั้นที่เราจะสามารถกำจัดออกไปได้ ดังนั้นเราจะแบ่งส่วน B ออกได้เป็น

$$B = B^+ B^- \quad ...(2.4.8)$$

เมื่อ B^- ประกอบด้วยซีโรที่อยู่ภายนอกยูนิตดิสค์ (unit disc) และ B^+ ประกอบด้วยซีโรภายในยูนิตดิสค์และสัมประสิทธิ์ของกำลังสูงสุดใน B^+ เป็น 1 ซึ่งเราจะเรียกโพลีโนเมียล B^+ นี้ว่าเป็นโมนิค (monic)

เนื่องจาก B^- ไม่สามารถเป็นแฟคเตอร์ของ AR + BS ได้ดังนั้นเราสามารถแบ่ง B_m ออกได้เป็น

$$B_m = B^- B_{m_1} \quad ...(2.4.9)$$

ซึ่งแสดงว่าซีโรที่ไม่เสถียรนั้นเราไม่สามารถจะเปลี่ยนแปลงหรือกำจัดได้จะถูกรวมอยู่ใน B_m เสมอ และเนื่องจาก B^+ เป็นแฟคเตอร์ของ AR + BS ดังนั้นจึงเป็นแฟคเตอร์ของ R ด้วยเช่นกัน จะได้ว่า

$$R = B^+ R_1 \quad ...(2.4.10)$$

ดังนั้นสมการ 2.4.5 จะเขียนได้ใหม่เป็น

$$T / (AR_1 + B^- S) = B_{m_1} / A_m \quad ...(2.4.11)$$

เนื่องจากเราได้ประมาณตัดทิ้งสัญญาณรบกวนไปในตอนแรกดังนั้นเราจะนำกลับมาพิจารณาใหม่โดยแทนในรูป A_o ซึ่งเป็นข้อจำกัดของรูปแบบ H_m ที่ต้องการนั่นก็คือ A_o เป็นแฟคเตอร์ของ AR + BS ดังนั้นเราก็จะได้เงื่อนไข

$$AR_1 + B^- S = A_o A_m \quad ...(2.4.12)$$

และ $$T = B_{m_1} A_o \quad ...(2.4.13)$$

ดังนั้นสมการคุณสมบัติของระบบปิดจะเป็น

$$AR + BS = B^+ A_o A_m \quad ...(2.4.14)$$

การที่จะหาสัญญาณควบคุมในสมการ 2.4.3 เราจำเป็นต้องรู้ค่าโพลีโนเมียล R,S,T ซึ่ง T หาได้จากสมการ 2.4.13 และ R_1 , S สอดคล้องกับ

สมการ 2.4.12 แต่เนื่องจากโพลีโนเมียล R_1 ,S ที่สอดคล้องกับสมการดังกล่าวมีมากมาย ดังนั้นเราจึงต้องหาว่าในบรรดาโพลีโนเมียลเหล่านั้นโพลีโนเมียลใดที่จะทำให้กฎการควบคุมที่ได้จากสมการ 2.4.3 ถูกต้องสมเหตุผลตามข้อจำกัดของมัน (causal)

ทฤษฎี

ในการออกแบบการควบคุมแบบกำหนดโพลจะให้กฎการควบคุมที่ถูกต้องสมเหตุผล

ถ้า $\deg A_m - \deg B_m >= \deg A - \deg B$ ...(2.4.15)

และ $\deg A_o >= 2\deg A - \deg A_m - \deg B^+ - 1$ ...(2.4.16)

ซึ่งจากอสมการ 2.4.15 แสดงว่า ค่าเวลาหน่วงของรูปแบบที่กำหนดจะต้องมีค่ามากกว่าหรือเท่ากับค่าเวลาหน่วงของระบบ

เนื่องจากระบบมักจะถูกรบกวนจากสิ่งรบกวนที่มีความถี่ต่ำ (low-frequency disturbances) และค่าความผิดพลาดของรูปแบบที่กำหนดที่ความถี่ต่ำ (low-frequency modelling errors) ดังนั้นถ้าเราให้อัตราขยายป้อนกลับมีค่าสูงที่ความถี่ต่ำ ๆ (high feedback gain at low frequency) ก็จะลดทอนการรบกวนดังกล่าวได้โดย

กำหนดให้ $R_1 = (z-1)^1 R_2$ ...(2.4.17)

ดังนั้นสมการ 2.4.12 จะกลายเป็น

$$A(z-1)^1 R_2 + B^- S = A_o A_m \quad ...(2.4.18)$$

ซึ่งจะให้คำตอบที่สมเหตุผลเมื่อสอดคล้องกับอสมการ 2.4.15 และ

$\deg A_o >= 2\deg A - \deg A_m + \deg B^+ + 1 - 1$ ...(2.4.19)

จากที่ได้กล่าวมาทั้งหมดเราจะสามารถสรุปอัลกอริทึมของการควบคุมแบบจูนปรับตัวเองที่ใช้การกำหนดโพลได้ดังนี้

อัลกอริทึมของการควบคุมแบบกำหนดโพล

ขั้นที่ 1 ประมาณค่าสัมประสิทธิ์ของโพลีโนเมียล A และ B ในสมการ 2.4.1 โดยวิธี RLS

ขั้นที่ 2 แทนที่ A และ B โดยค่าประมาณที่ได้จากขั้นที่ 1 และแก้สมการ 2.4.18 เมื่อเทียบกับ R_2 และ S และเลือกคำตอบที่สอดคล้องกับ

$\deg S < 1 + \deg A$

และ $\deg R_2 = \deg A_o + \deg A_m - \deg A - 1$

และคำนวณหา R , T และ R_1 จากสมการ 2.4.10 , 2.4.13 และ 2.4.17 ตามลำดับ

ขั้นที่ 3 คำนวณหาสัญญาณควบคุม u จากสมการ 2.4.3
ทำซ้ำขั้นที่ 1,2,3 ทุก ๆ ช่วงการสุ่ม

ซึ่งในทางปฏิบัติแล้วการกำหนดโพลีโนเมียลของโพลของลูปปิดที่ต้องการ (desired closed loop poles polynomial) $A_m(z^{-1})$ จะเลือกโพลที่สำคัญ (dominant poles) เพียงหนึ่งคู่เท่านั้น ส่วนที่เหลือให้ใกล้ ๆ กับจุดกำเนิด (origin) เพื่อให้สะดวกต่อการกำหนดผลตอบสนอง ปกติจะเลือกค่าอัตราส่วนการแกว่ง (damping ratio) และความถี่ของผลตอบแทนการกำหนดโพลโดยตรง ดังสมการ

$$A_m(z^{-1}) = 1 - 2e^{-\zeta w h}\cos wh\sqrt{1-\zeta^2}\, z^{-1} + e^{-2\zeta w h} z^{-2}$$

โดยที่ ζ - อัตราส่วนการแกว่ง
w - ความถี่ที่ต้องการกำหนด
h - คาบของการสุ่ม

บทที่ 3
การออกแบบ

แบ่งได้เป็น 2 ส่วนคือ

-ออกแบบการคำนวณของตัวควบคุม

-ออกแบบวงจรที่ใช้ A/D ; D/A และ วงจรเลียนแบบ

การออกแบบการคำนวณของตัวควบคุม

-เมื่อทำการแปลงแซดของระบบอันดับที่ 2 จะได้ดังนี้

$$H = B(z)/A(z) =$$

$$Z\left[\frac{k1(s+d)}{(s^2+2\zeta\omega s+\omega^2)}\right] = \frac{k(z-b)}{(z^2+a1z+a2)}$$

-ตัวควบคุมเป็นแบบที่ไม่มีการตัดกันของโพลและซีโร่ และเพิ่มเกนที่ความถี่ต่ำโดยเพิ่มโพลที่มีค่าเท่ากับ 1 ให้โอเพ่นลูปทรานเฟอร์ฟังก์ชั่น

-ต้องการระบบปิดที่มีทรานเฟอร์ฟังก์ชั่นเป็น

$$Hm(z) = \frac{Bm(z)}{Am(z)} = \frac{(1+p1+p2)(z-b)}{(1-b)(z^2+p1z+p2)}$$

จากการพิจารณาดีกรีของโพลิโนเมียลต่างๆ (A_o ,R,S,T) ตามทฤษฎีจึงสมมติว่าตัวควบคุมมีค่าพารามิเตอร์ดังนี้

$$A_o(z) = z^1$$

$$R(z) = (z-1)(z+r1)$$

$$S(z) = s0z^2+s1z+s2$$

$$T(z) = t0z^2 = \acute{Bm}\, z^2$$

แทนโพลิโนเมียลที่สร้างขึ้นลงในสมการ

$$\frac{T(z)}{A(z)R(z)+B(z)S(z)} = \frac{A_o(z)\acute{Bm}(z)}{A_o(z)Am(z)}$$

โดย $\acute{B}(z) = K(z-b)$

$$\acute{Bm}(z) = Bm/\acute{B} = \frac{(1+p1+p2)}{K(1-b)}$$

เมื่อแก้สมการจะได้

$$r1 = \frac{(b^4 + p1b^3 + p2b^2)}{(b^2 + a1b + a2)(b-1)} \qquad -b$$

$$s2 = (-r1a2)/(Kb)$$

$$s1 = (s2K - r1a1 + (r1 - 1)a2)/(Kb)$$

$$s0 = (p1 - a1 - (r1 - 1))/K$$

$$t0 = (1 + p1 + p2)/(k(1 - b))$$

นำไปเขียนเป็นสมการหา u

$$u(n) = t0sp - s0y(n0) - s1y(n-1) - s2y(n-2) - (r1-1)u(n-1) + r1u(n-2)$$

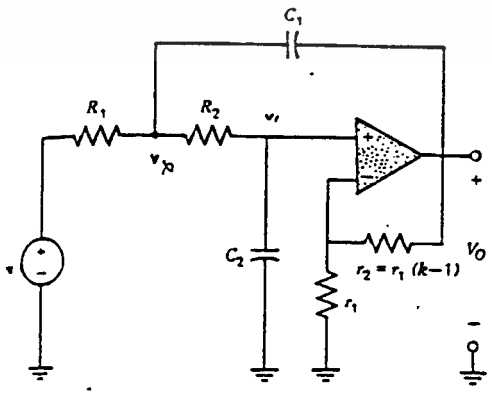
<u>การออกแบบวงจรเลียนแบบระบบ</u>

-ใช้วงจรดังรูปที่ 3.1

-มีทรานส์เฟอร์ฟังชั่นเป็น

$$H(s) = \frac{k/(R1R2C1C2)}{(s^2 + (1/R1C1 + 1/R2C1 + (1-k)/R2C2))s + 1/R1R2C1C2)}$$

$: k = 1 + r2/r1$



รูปที่ 3.1 แสดงวงจรเลียนแบบระบบ

บทที่ 4
ผลการทดลอง

การทดลองนี้ทำขึ้นเพื่อทดสอบการควบคุมระบบโดยใช้วิธี โพล์เพลสเมนต์ โดยทำการประมาณสมการของระบบด้วยวิธี ลีสแสควร์ โดยทดลองกับระบบที่เลียนแบบขึ้นด้วยคอมพิวเตอร์และระบบที่เลียนแบบด้วยวงจรอีเลคทรอนิคส์

ควบคุมระบบที่ถูกจำลองบนคอมพิวเตอร์

-ทำการควบคุมระบบที่มีทรานส์เฟอร์ฟังชั่นเป็น

$$\frac{K(\omega_n^2)}{S^2 + 2\zeta\omega_n S + \omega_n^2} \Rightarrow \frac{K1(Z-b)}{Z^2 + a1Z + a2}$$

-ช่วงเวลาการสุ่มมีค่า 0.33 วินาที

-ค่าระบบบิดที่กำหนดให้กับตัวควบคุมคือ

$$Hm(z) = \frac{(1+p1+p2)(z-b)}{(1-b)(z^2 + p1z + p2)}$$

-ค่า p1 และ p2 กำหนดโดยการเลือก และ ของระบบบิดที่เราต้องการ

-ทำการสุ่มจำนวน 600 ครั้ง n=0 จนถึง n=599

ตอนที่ 1

-ระบบมีค่าต่างๆ ดังนี้ ζ = 0.4

ω_n = 0.31 เรเดียน/วินาที

K = 2

-การประมาณสมการของระบบใช้ค่าฟอเกตติ้งแฟคเตอร์ = 0.97

-ต้องการควบคุมให้ระบบบิดมีค่า ζ =0.707

ω_n =0.25 เรเดียน/วินาที

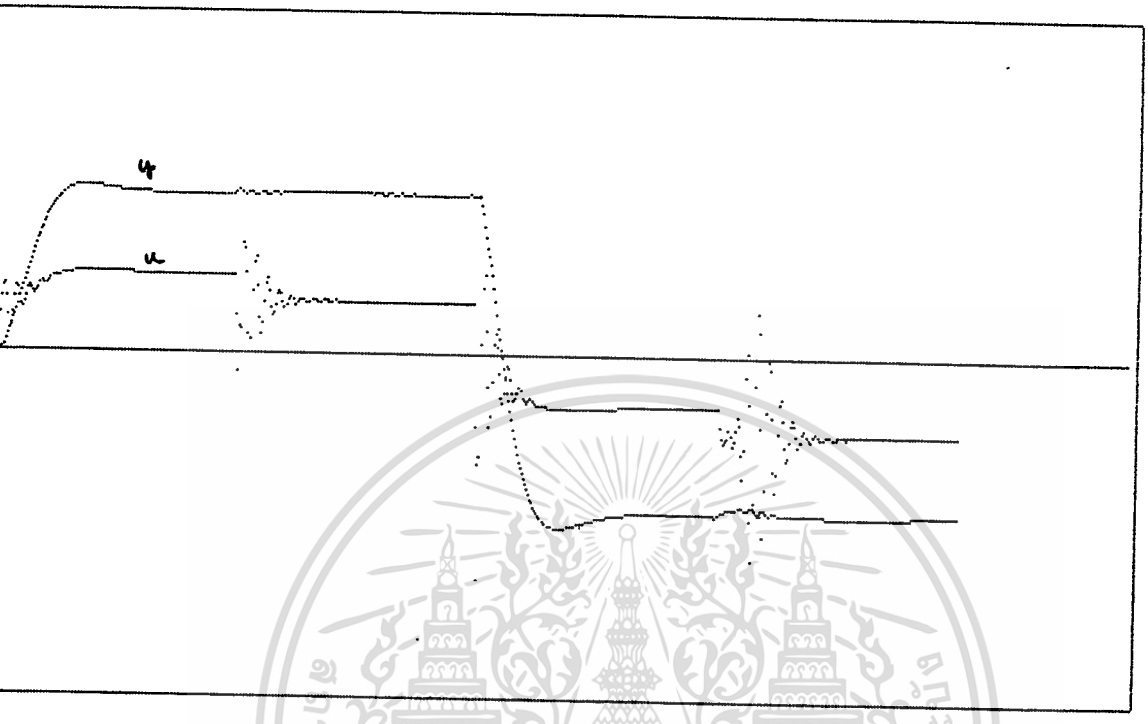
-ทำการป้อนค่า เซตพอนย์ เป็นสเตป 1 หน่วยที่ n= 0

-ที่ n= 150 ค่า K ของระบบเปลี่ยนจาก K=2 เป็น K=3

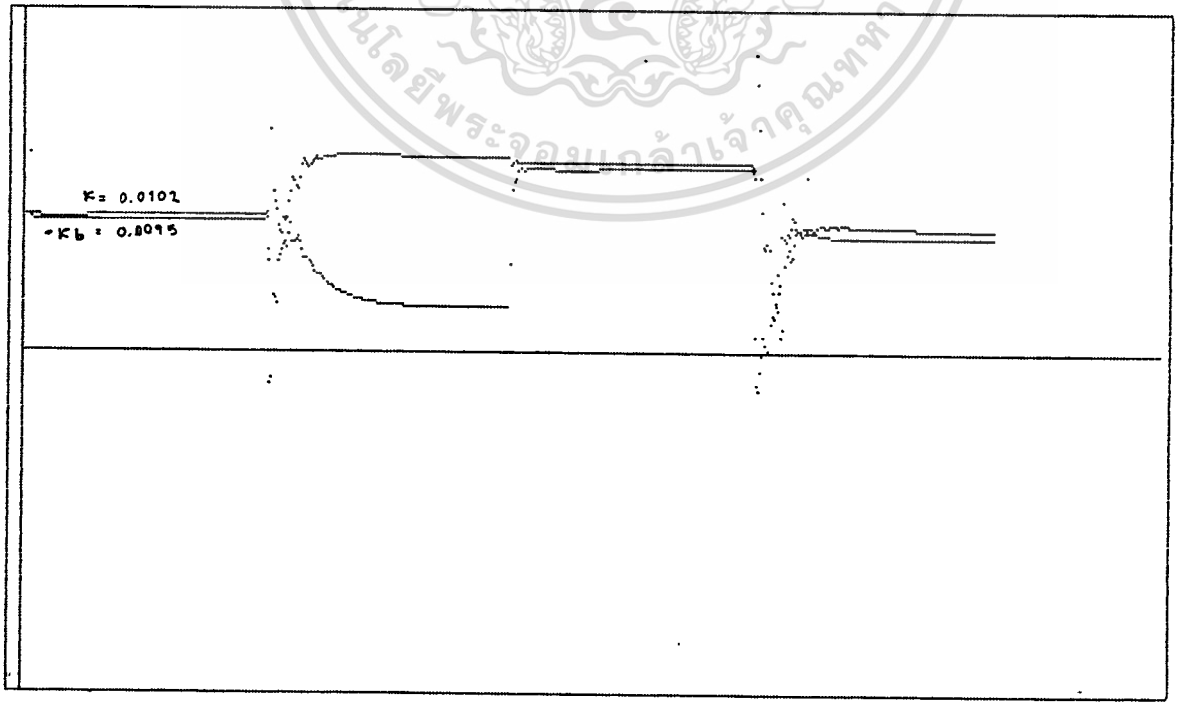
-ที่ n= 300 ค่าเซตพอยต์ เปลี่ยนจาก SP=1 เป็น SP=-1

-ที่ n= 450 ค่า K ของระบบเปลี่ยนจาก K=3 เป็น K=2

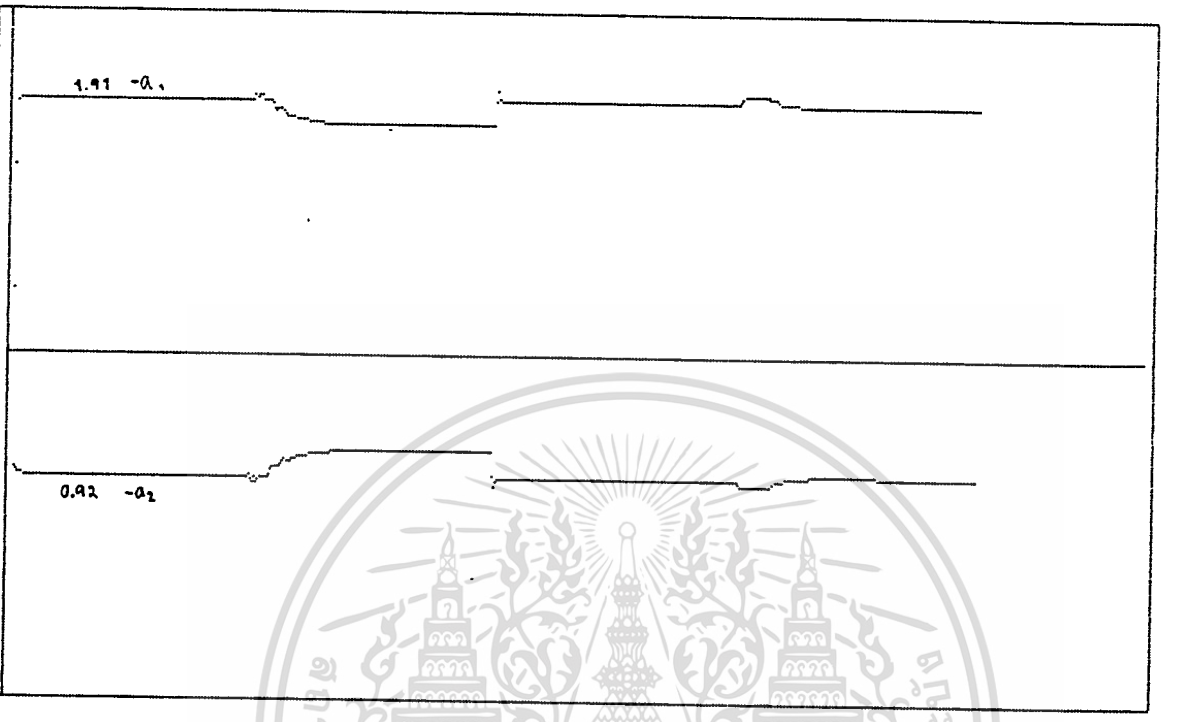
-ได้ผลตอบสนองและค่าพารามิเตอร์ดังรูป 4.1, 4.2, และ 4.3



รูปที่ 4.1 แสดงการตอบสนองของระบบในตอนที่ 1



รูปที่ 4.2 แสดงการประมาณค่าพารามิเตอร์ของระบบในตอนที่ 1



รูปที่ 4.3 แสดงการประมาณค่าพารามิเตอร์ของระบบในตอนที่ 1

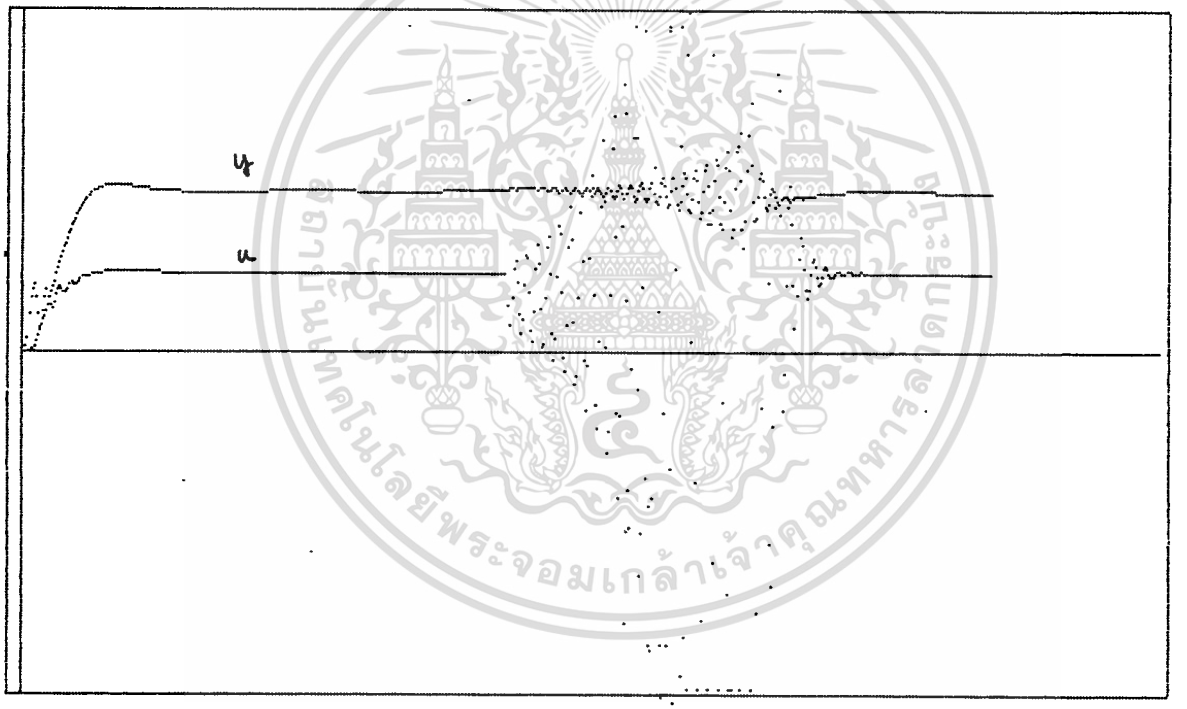
ตอนที่ 2 ทดสอบการควบคุมระบบที่พารามิเตอร์เปลี่ยนแปลงโดยไม่ใช้ตัวควบคุมปรับตัวเอง

-ใช้ระบบควบคุมของตอนที่ 1 แต่ใช้แบบปรับตัวเองจนถึง n=100 จึงหยุดปรับตัวเอง

-ที่ n=300 เปลี่ยนเกน K จาก 2 เป็น 3

-ที่ n=450 เปลี่ยนเกน K จาก 3 เป็น 2

-ผลตอบสนองแสดงได้ดังรูป 4.4 เห็นได้ว่าระบบไม่เสถียร

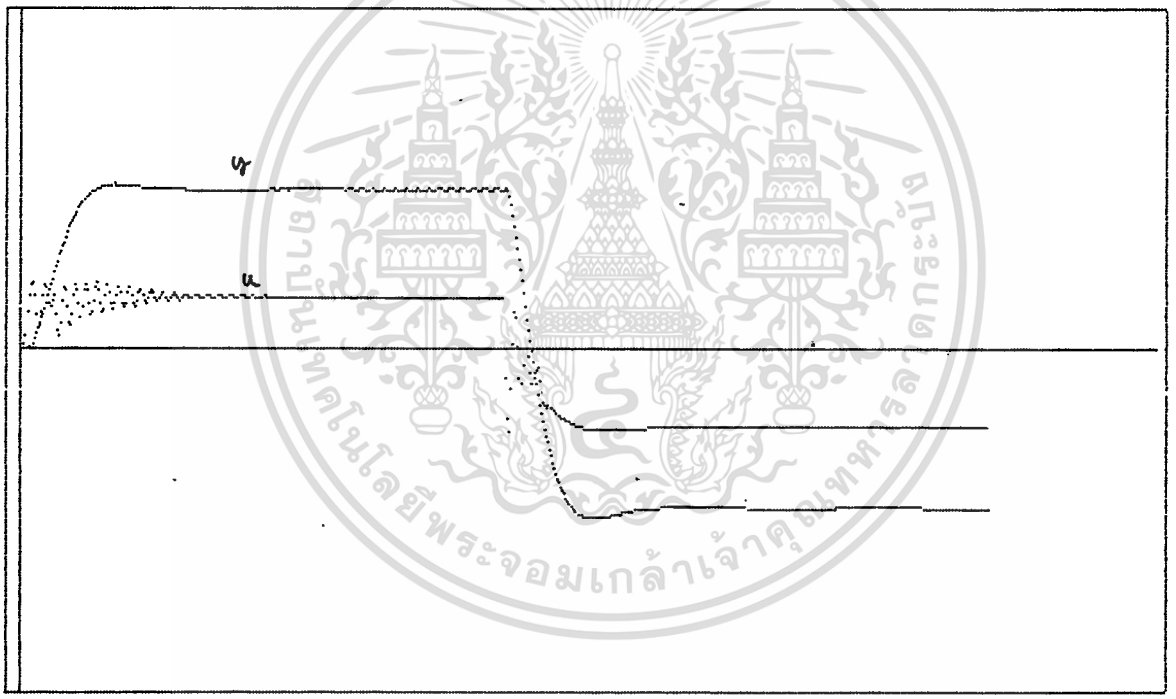


รูปที่ 4.4 แสดงการตอบสนองของระบบในตอนที่ 2

ตอนที่ 3 ทดสอบผลของการเปลี่ยนแปลงค่าเกน (K) ของระบบขณะในช่วง ทรานส์เซียน

-ใช้ระบบควบคุมเดียวกับตอนที่ 1 แต่ให้จุดที่เปลี่ยนเกนของระบบเป็นที่ตำแหน่ง n=50 และn=350 แทนตำแหน่งที่ n=150และn=450

-ผลการตอบสนองดังรูปที่ 4.5 แสดงให้เห็นว่ามีผลต่อการแกว่งของค่า u เล็กน้อยในช่วงทรานส์เซียน

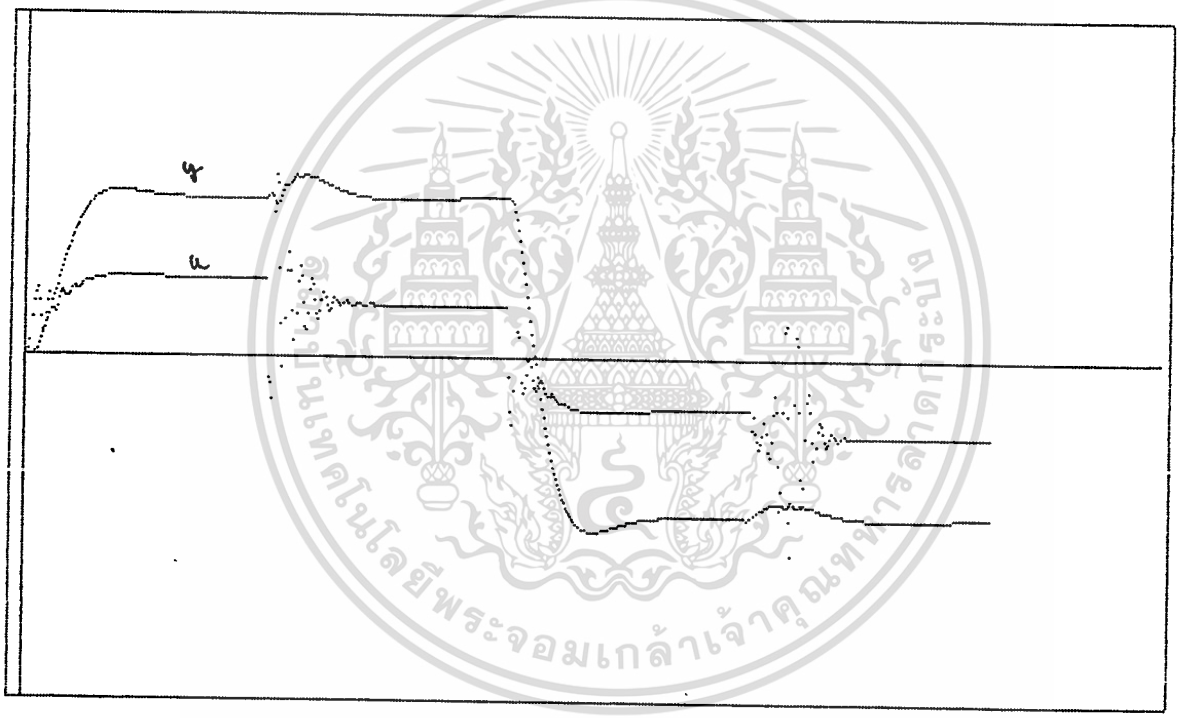


รูปที่ 4.5 แสดงการตอบสนองของระบบในตอนที่ 3

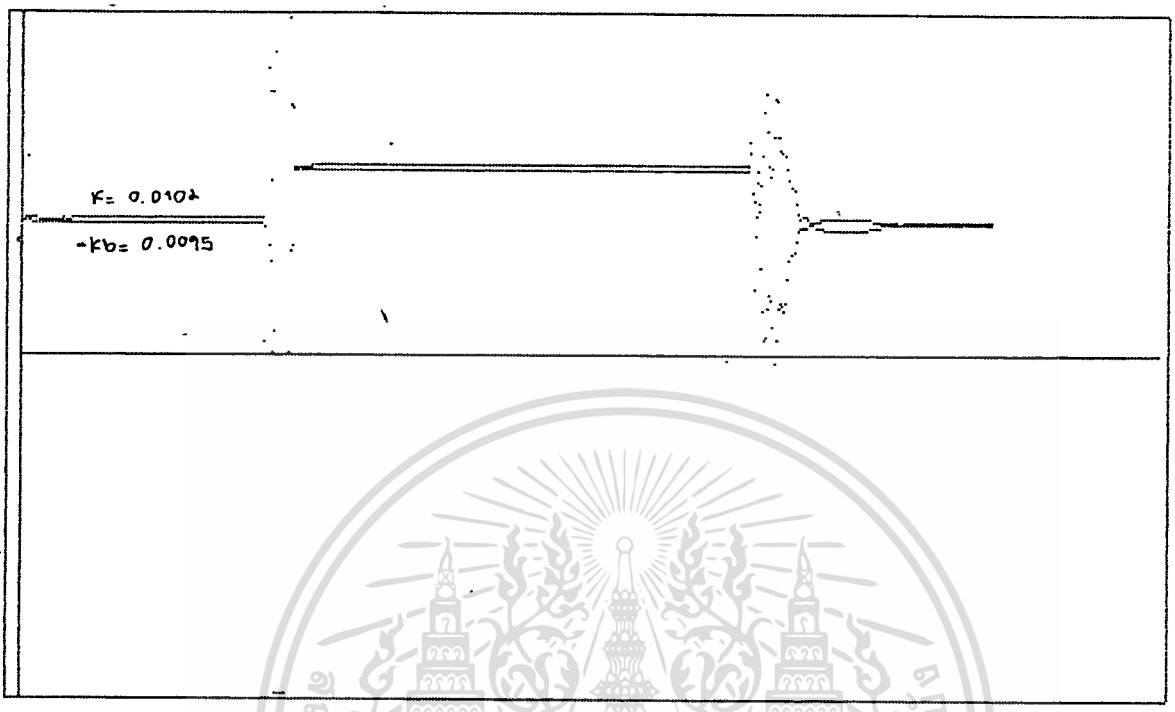
ตอนที่ 4 ทดสอบผลของค่าฟอร์เกตติ้งแฟคเตอร์

-ทำการทดสอบเช่นเดียวกับตอนที่ 1 แต่ในส่วนของการประมาณระบบใช้ค่าฟอร์เกตติ้งแฟคเตอร์เป็น 0.93 แทน 0.97

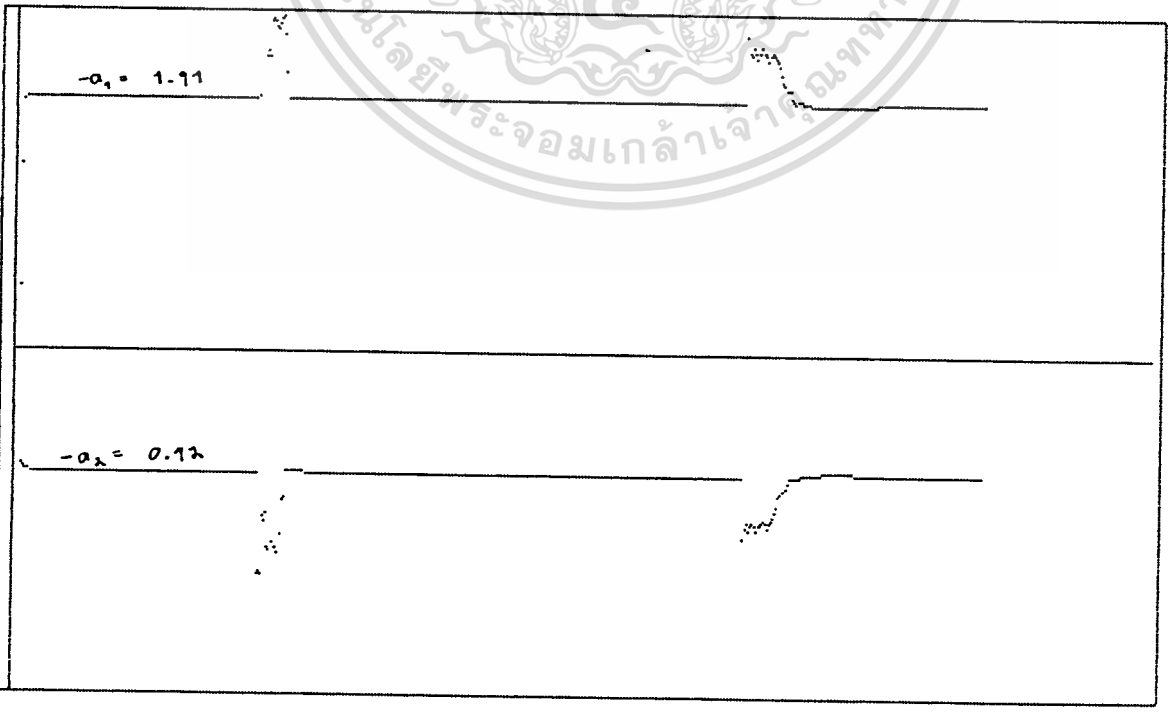
-ผลดังรูปที่ 4.6-4.8 จะเห็นว่าค่าที่ได้แกว่งกว่าในตอนที่ 1 จะประมาณได้เร็วกว่า



รูปที่ 4.6 แสดงการตอบสนองของระบบในตอนที่ 4



รูปที่ 4.7 แสดงการประมาณค่าพารามิเตอร์ของระบบในตอนที่ 4



รูปที่ 4.8 แสดงการประมาณค่าพารามิเตอร์ของระบบในตอนที่ 4

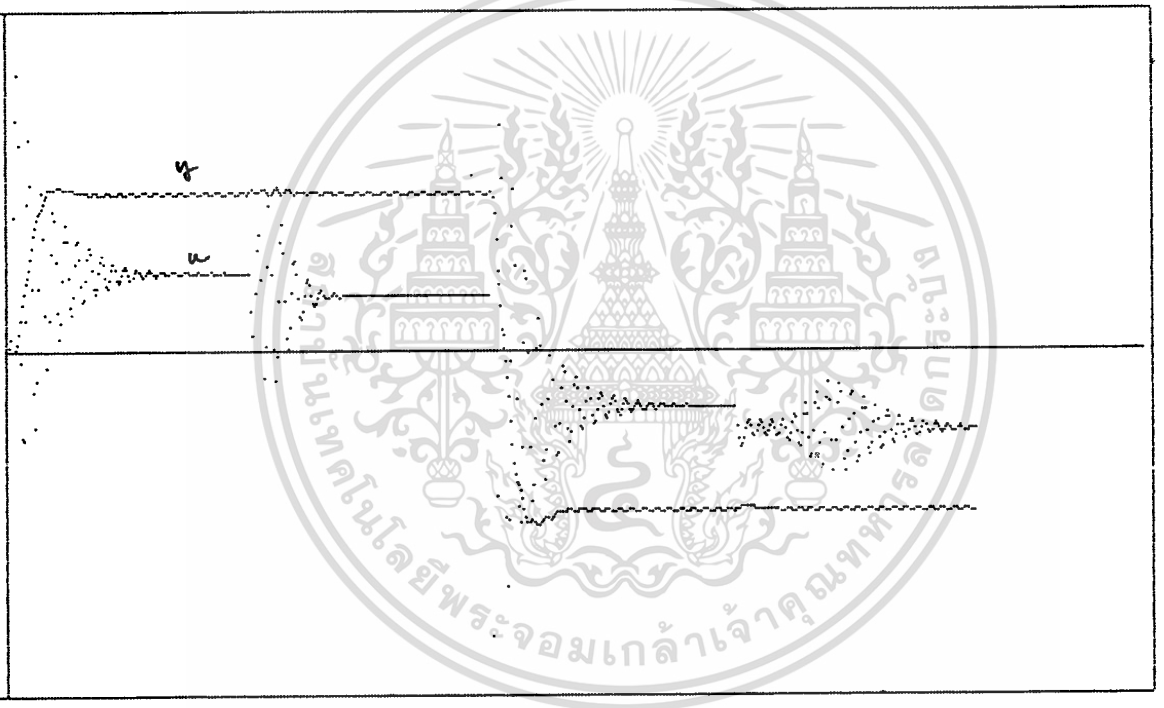
ตอนที่ 5 ทดสอบผลของการกำหนดค่าต่างๆของระบบปิดที่ต้องการ

-ขั้นตอนการทำเช่นตอนที่ 1 แต่เปลี่ยนค่าระบบปิดที่ต้องการเป็น

$\zeta_n = 0.707$

$\omega_n = 0.3$

-จากผลดังรูปที่ 4.9 แสดงให้เห็นว่าเมื่อเลื่อนค่า เข้าไปใกล้ ของระบบจะทำให้ แกว่งมากขึ้น



รูปที่ 4.9 แสดงการตอบสนองของระบบในตอนที่ 5

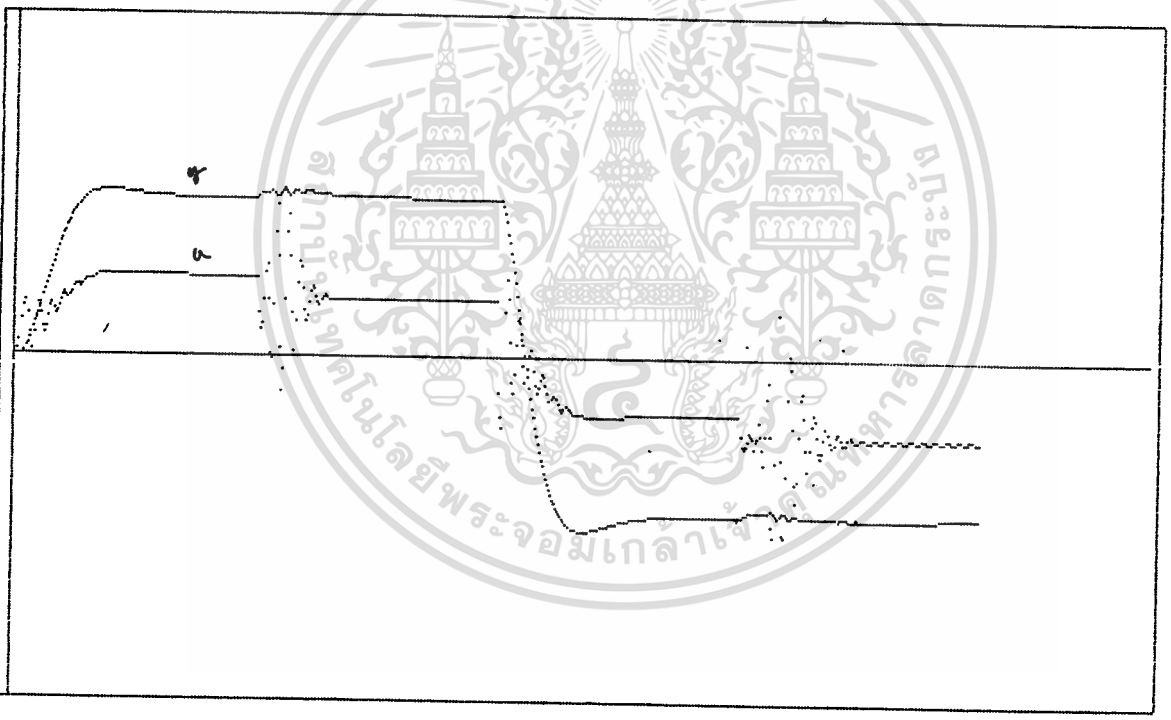
ตอนที่ 6 ทดสอบผลของค่าแดมป์ของระบบที่จะควบคุม

-ทำการทดลองในตอนที่ 1 อีก 2 ครั้ง

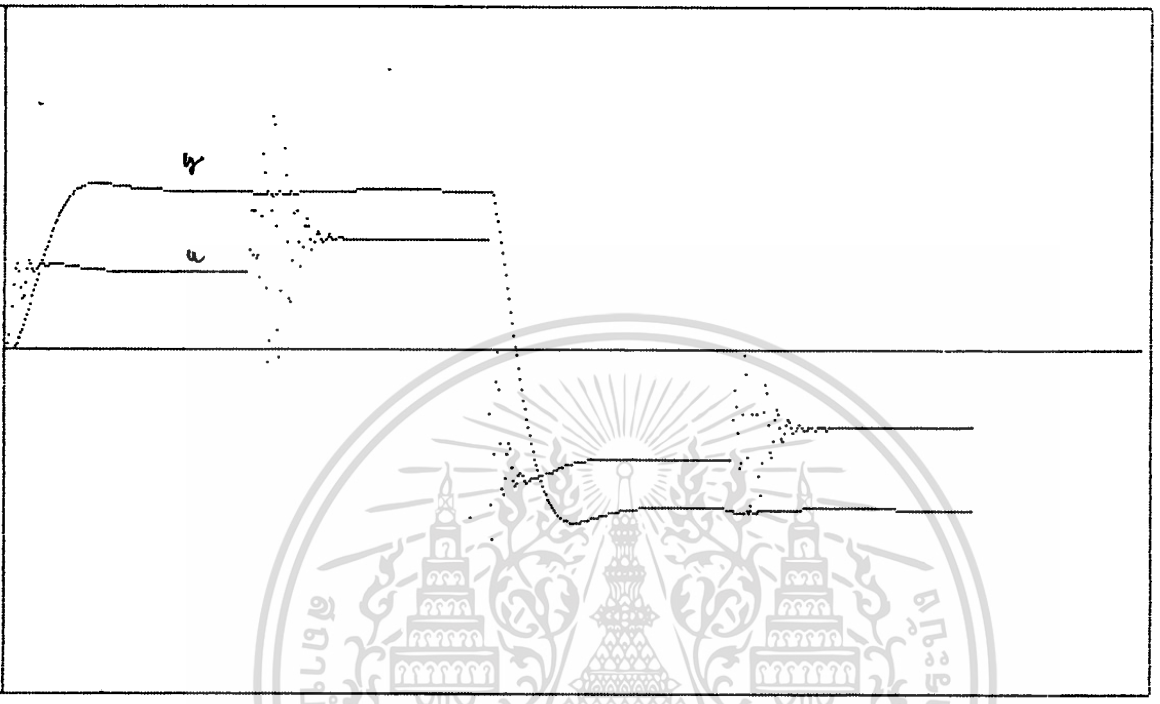
ครั้ง 1 เปลี่ยนค่าแดมป์ ของระบบเป็น 0.1

ครั้ง 2 เปลี่ยนค่าแดมป์ ของระบบเป็น 0.9

-ผลของครั้งที่ 1 และ 2 แสดงได้ดังรูปที่ 4.10 และ 4.11 แสดงให้เห็นถึงผลตอบสนองที่คล้ายกัน



รูปที่ 4.10 แสดงการตอบสนองของระบบในตอนที่ 6



รูปที่ 4.11 แสดงการตอบสนองของระบบในตอนที่ 6

การควบคุมระบบที่เลียนแบบโดยใช้วงจรอีเลคทรอนิคส์

ตอนที่ 7

-นำวงจรในบทที่แล้วมาสร้างระบบที่มีค่าดังนี้

$K = 1.7$

$\omega_n = 0.31$ เรเดียน/วินาที

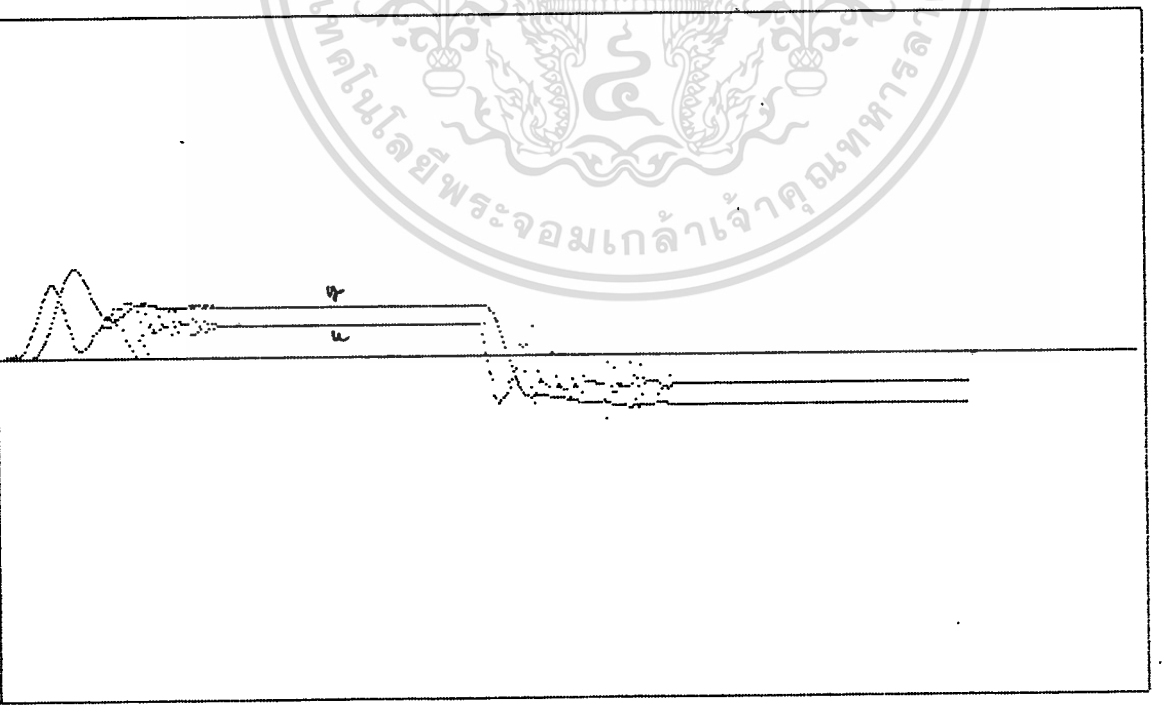
$\zeta = 1.3$

-ระบบควบคุมนี้มีช่วงเวลาการสุ่มเป็น 0.17 วินาที

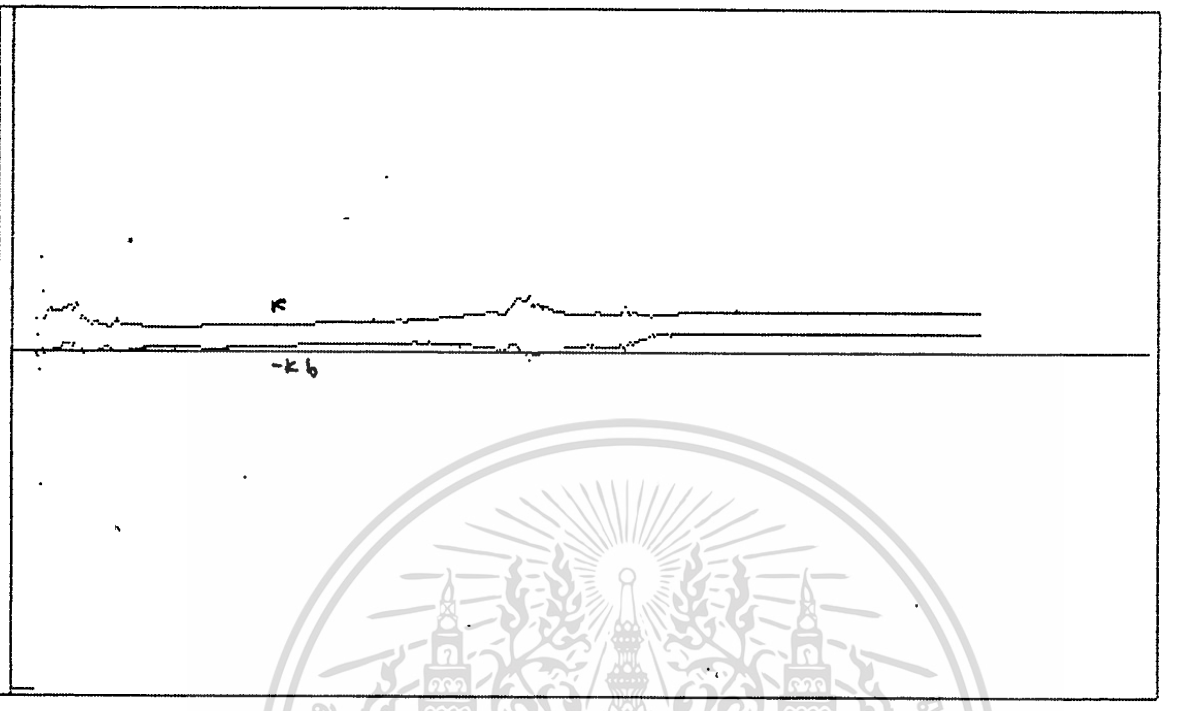
-ควบคุมให้ระบบปิดที่ได้ที่ p1=-1.832;p2=0.839

-ใช้การประมาณค่าระบบโดยมีค่าฟอร์เกตติ้งแฟคเตอร์ = 0.99

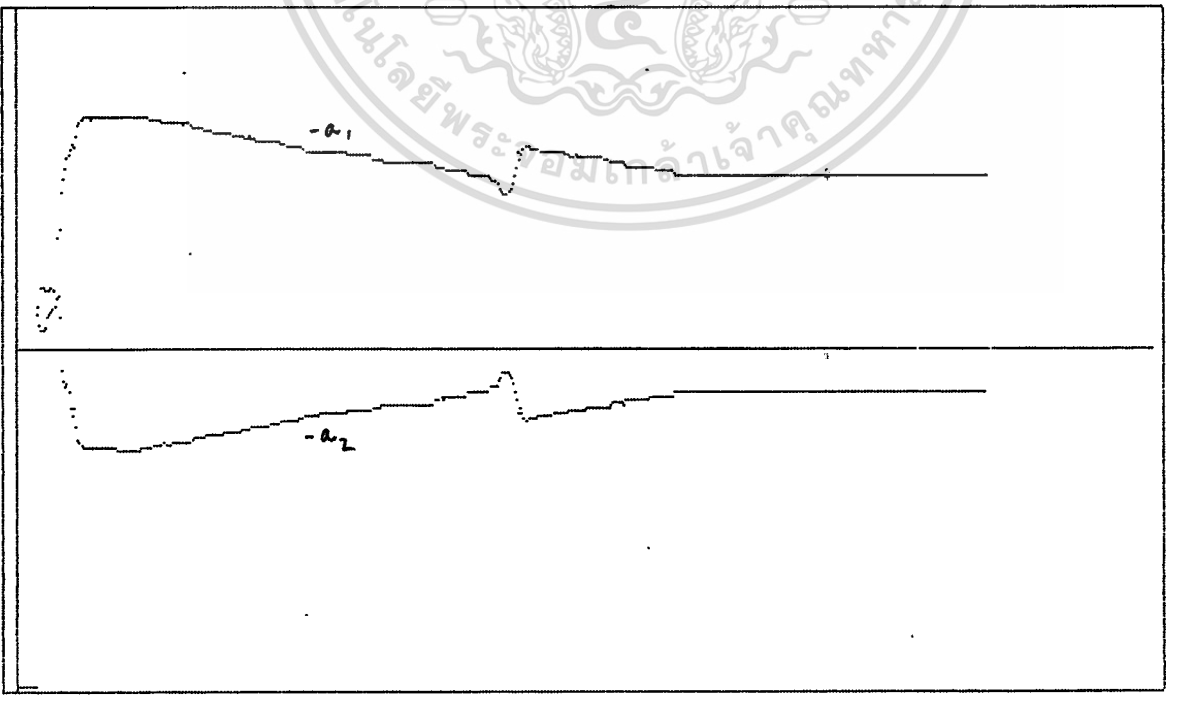
-ผลการทดลองดังรูป 4.12-4.14 จะเห็นว่าค่าพารามิเตอร์ที่ประมาณได้มีค่าเลื่อนไม่คงที่ เป็นผลเนื่องมาจากสัญญาณรบกวนซึ่งเกิดจากวงจร A/D และ D/A ซึ่งถ้าเปลี่ยนเป็นขนาดข้อมูล 12 บิทจะทำให้สัญญาณรบกวนนี้ลดลง



รูปที่ 4.12 แสดงการตอบสนองของระบบในตอนที่ 7



รูปที่ 4.13 แสดงการประมาณค่าพารามิเตอร์ของระบบในตอนที่ 7



รูปที่ 4.14 แสดงการประมาณค่าพารามิเตอร์ของระบบในตอนที่ 7

บทที่ 5
บทสรุปและวิจารณ์

บทนี้เป็นบทสุดท้ายของปริญญานิพนธ์ ซึ่งเป็นการสรุปขั้นตอนการทำปริญญานิพนธ์ที่ได้ทำมา โดยจะกล่าวถึงเหตุผลที่ใช้ แนวการแก้ไขและแนวการปรับปรุง เพื่อพัฒนาตัวควบคุมให้มีประสิทธิภาพในการควบคุมเพิ่มขึ้น

สรุปการทำปริญญานิพนธ์

สำหรับการวิจัยนี้มีจุดประสงค์ใหญ่คือ การพัฒนาการควบคุมโดยใช้ไมโครคอมพิว-เตอร์ทำหน้าที่เป็นตัวควบคุมแบบจูนปรับตัวเอง เพื่อควบคุมกระบวนการที่ไม่ทราบค่าพารา-มิเตอร์ที่แน่นอนให้มีสมรรถนะดีขึ้น รวมทั้งมีความแม่นยำและมีความสะดวกรวดเร็วใน การควบคุม การใช้ตัวควบคุมแบบจูนปรับตัวเองนี้มีข้อได้เปรียบคือ สามารถที่จะให้ผลตอบของกระบวนการที่มีค่าเท่ากับค่าที่ตั้งไว้ไม่ว่ากระบวนการที่ควบคุมอยู่จะมีการเปลี่ยนแปลงพารามิเตอร์หรือมีสิ่งรบกวนต่าง ๆ เกิดขึ้นก็ตาม แต่ทั้งนี้และทั้งนั้นการควบคุมแบบจูนปรับตัวเองนี้จะต้องคำนึงถึงเสถียรภาพ สมรรถนะและการลู่เข้าของระบบซึ่งขึ้นอยู่กับรูปแบบของกระบวนการและอัลกอริทึมที่ใช้

แต่การควบคุมแบบจูนปรับตัวเองมีลักษณะการควบคุมหลายชนิด การนำมาใช้จึงต้องพิจารณาว่าการควบคุมชนิดใดบ้างสามารถนำมาใช้ได้โดยใช้ไมโครคอมพิวเตอร์เป็นตัวควบคุม เนื่องจากการควบคุมแบบจูนปรับตัวเองเป็นการควบคุมแบบเวลาจริง (Real time) ทำให้สมรรถนะของการควบคุมขึ้นอยู่กับขีดความสามารถของไมโครคอมพิวเตอร์และเวลาที่ใช้ในอัลกอริทึมของการควบคุมแบบจูนปรับตัวเอง และเราก็ได้เลือกลักษณะการควบคุมแบบกำหนดโพล ซึ่งสามารถกำหนดผลตอบให้มีลักษณะตามที่กำหนดได้โดยที่มีความมั่นคงเหนือกว่าการควบคุมลักษณะอื่น ๆ และสามารถใช้ได้กับระบบชนิดต่าง ๆ ได้หลากหลายที่สุด เนื่องจากคัญญาณรบกวน (Noise) ที่เกิดขึ้นในกระบวลการมีผลต่อการควบคุมน้อย ทำให้การประมาณค่าของพารามิเตอร์ สามารถใช้ในการประมาณแบบ RLS ได้

จากผลการทดลองจะเห็นว่าการควบคุมชนิดการกำหนดโพลซึ่งสามารถกำหนดลักษณะของผลตอบของกระบวนการได้นั้น ทำให้ผลตอบของกระบวนการและตัวควบคุมค่อนข้างเรียบไม่ไวต่อการเปลี่ยนแปลงของค่าผิดพลาดและสามารถเข้าสู่สภาวะคงตัวได้ตามต้องการ

แนวการปรับปรุงของตัวควบคุม

ตัวควบคุมที่ทำขึ้นจัดว่าเป็นควบคุมที่อยู่ในขั้นทดลองใช้ ปัจจัยที่ใช้ในการควบคุมยังมีประสิทธิภาพไม่มากนัก เช่นโปรแกรมอัลกอริทึมที่ใช้ในการควบคุม อุปกรณ์อินเตอร์เฟส (Interface) ต่าง ๆ โครงสร้างของอัลกอริทึมและการแทนรูปแบบของกระบวนการ เป็นต้น ซึ่งแต่ละส่วนยังมีราบละเอียดปลีกย่อยอยู่ อาจจะต้องปรับปรุงเพื่อความเหมาะสมในการใช้งานมากขึ้น แต่การแก้ในส่วนที่สำคัญ ๆ ได้แก้ไขเรียบร้อยแล้ว สำหรับการพัฒนาและปรับปรุงเพื่อให้ตัวควบคุมมีประสิทธิภาพและสมบุรณ์แบบมากขึ้นจะต้องคำนึงถึง

1. โปรแกรมและอัลกอริทึม เทคนิคการเขียนโปรแกรมมีความสำคัญในการควบคุมเนื่องจากการควบคุมกระบวนการในระบบเชิงเลขนี้อาศัยการคำนวนโดยใช้ไมโครคอมพิวเตอร์ดังนั้นค่าของตัวแปรและพารามิเตอร์ในโปรแกรมจะต้องพิจารณาว่าไม่มีผลกระทบต่อตัวแปรหรือพารามิเตอร์ตัวอื่น ส่นการพัฒนาอัลกอริทึมควรจะให้มีประสิทธิภาพและความมั่นคงสูงมากขึ้น นอกจากนี้ควรพัฒนาเรื่องความเร็วของอัลกอริทึมในการควบคุมและขีดความสามารถ เช่น การเปลี่ยนอัลกอริทึมของการประมาณแบบเชิงเส้นมาใช้การประมาณแบบไม่เชิงเส้น ซึ่งจะทำให้ตัวควบคุมสามารถใช้ได้กับกระบวนการที่ไม่เชิงเส้นได้มีประสิทธิภาพมากขึ้น เป็นต้น

2. โครงสร้างของการแทนรูปแบบของกระบวนการ สำหรับในการทดลองเป็นการควบคุมแบบหนึ่งอินพุทหนึ่งเอาต์พุท อัลกอริทึมที่แทนกระบวนการการจะอยู่ในรูปของโพลิโนเมียล ซึ่งการดัดแปลงเพื่อใช้ในระบบควบคุมกระบวนการที่มีขนาดใหญ่จะทำได้ยากต้องใช้เวลาในการคำนวนมาก ดังนั้นจึงควรเลือกการแทนรูปแบบที่สามารถดัดแปลงได้ง่ายกว่านี้ ซึ่งแนวโน้มของการพัฒนาการควบคุมแบบจูนปรับตัวเองนิยมมาใช้ในการควบคุมชนิดหลายอินพุทหลายเอาต์พุท โดยใช้สมการพลวัตมากขึ้น ซึ่งในอนาคตจะสามารถนำมาประยุกต์ใช้ในงานควบคุมด้วยไมโครคอมพิวเตอร์ได้อย่างแน่นอน

3. อุปกรณ์ที่ใช้ในตัวควบคุม ซึ่งได้แก่ ไมโครโปรเซสเซอร์ วงจร A/D และ D/A ควรเลือกใช้ชนิดที่มีความเร็วสูงและมีความละเอียด (จำนวนบิท) เพิ่มขึ้น จะทำให้การควบคุมกระบวนการมีสมรรถนะมากขึ้น และสามารถคลอบคลุมปัญหาในการควบคุมให้มากขึ้น

ตัวควบคุมในอุตสาหกรรม

การพัฒนาตัวควบคุมในอุตสาหกรรมถึงแม้ในปัจจุบันตัวควบคุมจะเปลี่ยน

จากการควบคุมแบบอนาล็อก (Analog) มาเป็นการควบคุมแบบดิจิตอล (Digital) โดยใช้ไมโครโปรเซสเซอร์เป็นตัวควบคุมแล้วก็ตาม แต่ลักษณะการควบคุมยังใช้การควบคุมแบบ PID เสียเป็นส่วนใหญ่ โดยทั่วไปตัวควบคุมในอุตสาหกรรมจะต้องมีความเชื่อถือ (Reliability) และความสามารถ พิเศษต่าง ๆ เช่น มีฟังก์ชันพิเศษที่ใช้ในระบบควบคุม สามารถทำหน้าที่แบบแยกตัว (Stand alone) หรือติดต่อกับผู้ควบคุม (Operator และ Supervisory) โดยผ่านบัส (Bus) ควบคุมได้ มีระบบความปลอดภัย (Security) ต่าง ๆ เป็นต้น การควบคุมจึงสามารถทำเป็นแบบตามลำดับชั้น (Hierachical control system) ได้ สำหรับตัวควบคุมในปัจจุบันที่น่าสนใจได้แก่ SLPC ตัวควบคุมแบบโปรแกรมได้ (Programmable indicating controller) ซึ่งจัดว่าเป็นตัวควบคุมที่มีประสิทธิภาพ สามารถโปรแกรมใช้งานได้ตามความต้องการของผู้ใช้มีฟังก์ชันการทำงานหลายอย่างได้แก่ การควบคุมแบบ PID , แบบแคสเคด (Cascade control),การควบคุมแบบเลือกชนิดการควบคุมโดยอัติโนมัติ (Autoselector control) และการชดเชยเดดไทม์ (Dead time compensation) เป็นต้น แต่การพัฒนาส่วนใหญ่จะเป็นการพัฒนาทางด้านประสิทธิภาพของการควบคุมให้มีขนาดใหญ่ขึ้นเพื่อใช้ในงานอุตสาหกรรม ซึ่งอย่างไรก็ตาม การพัฒนาตัวควบคุมแบบปรับตัวเอง ในอุตสาหกรรมนั้นก็มีมานานพอสมควร ตัวควบคุมที่น่าสนใจได้แก่ระบบผู้เชี่ยวชาญ (Expert system) เป็นตัวควบคุมแบบปรับตัวเองชนิดหนึ่งที่ในอุตสาหกรรม มีลักษณะการควบคุมแตกต่างไปจากการควบคุมแบบจูนปรับตัวเองเพราะ ระบบผู้เชี่ยวชาญ เป็นโปรแกรมที่ใช้แก้ปัญหาในการควบคุม สามารถควบคุมกระบวนการแบบปรับตัวเองได้โดยการประยุกต์ปัญญาประดิษฐ์ (Artifical intelligent) เข้ากับเทคนิคของการจดจำรูปแบบ (Pattern recognition) ประกอบกับภายในโปรแกรมได้บรรจุความรู้ (Knowledge) ประสพการณ์ (Experience) และความชำนาญของวิศวกรผู้ควบคุม ทำให้ ระบบผู้เชี่ยวชาญ สามารถตัดสินการควบคุมและปรับค่าพารามิเตอร์ของตัวควบคุมได้ด้วยเหตุผลที่ใกล้เคียงกับคน โดยไม่ต้องอาศัยแบบจำลองทางคณิตศาสตร์ของกระบวนการเข้ามาเกี่ยวข้องซึ่งเป็นข้อได้เปรียบเมื่อเทียบกับการควบคุมแบบปรับตัวเองทั่ว ๆ ไป นอกจากนี้ ระบบผู้เชี่ยวชาญ ถูกออกแบบมาใช้ควบคุมกระบวนการแบบ เวลาจริง ทำให้การควบคุมมีประสิทธิภาพและถูกออกแบบให้ฉลาด (Intelligent) ทำให้โปรแกรมสามารถตัดสินใจเองได้ว่าการควบคุมควรจะเป็นอย่างไร ระบบผู้เชี่ยวชาญได้พัฒนาไปจนมี ฟังก์ชัน พิเศษได้แก่ ตัวกำหนดอัตราขยาย (Gain scheduler) เพื่อใช้กับกระบวนการที่ไม่เชิงเส้นจะเห็นว่า ระบบผู้เชี่ยวชาญ

ออกแบบขึ้นด้วยเทคโนโลยีขั้นสูงและมีโครงสร้างที่ซับซ้อนตัวอย่างเช่น ใช้ไมโครโปรเซสเซอร์ที่เป็นแบบมัลติทาสค์ (Multi-tasking) และใช้โปรแกรมในการควบคุมหลายขั้นตอน เป็นต้น จากปัจจัยทั้งสองนี้เป็นความยากที่จะพัฒนาตัวควบคุมในขณะนี้ให้ไปได้ถึง การพัฒนาตัวควบคุมแบบปรับตัวเองที่เป็นไปได้ในขณะนี้อาจจะพัฒนาจากตัวควบคุมแบบจูนปรับตัวเองที่ได้พัฒนานี้ เนื่องจากเป็นการลงทุนที่ไม่สูงนัก ประกอบกับโครงสร้างที่ใช้สามารถดัดแปลงให้เหมาะสมกับการควบคุมได้ง่าย ทั้งสามารถควบคุมกระบวนการได้เป็นอย่างดี ในการพัฒนาเพื่อให้สามารถใช้ในอุตศาหกรรมควรให้มีฟังก์ชั่นการทำงานเหมือนกับตัวควบคุมที่ใช้ในอุตสาหกรรม เพื่อที่สามารถจะนำไปใช้ร่วมกับระบบควบคุมในอุตสาหกรรมได้อย่างมีประสิทธิภาพ ตัวอย่างเช่น สามารถติดต่อกับผู้ควบคุมได้เป็นต้น ซึ่งสิ่งเหล่านี้จะทำให้ตัวควบคุมที่จะพัฒนาเป็นตัวควบคุมแบบปรับตัวเองน่าสนใจมากในอุตสาหกรรม

ภาคผนวก

โปรแกรมลิสต์

```
type stp.pas

Program Self_Tuning_Controller;


type array1=array[1..10,1..10] of real;
Var  sign,i,j,err,nv,kco: integer;
     s   : char;
     mss:string[20];
     ech,sp,det,gain,m : real;
     y,u,ds,di,e,coef,H:array[1..10] of real;
     P0:array1;
     first:boolean;
     k,b,a0,a1,r1,s0,s1,s2,t0,p1,p2:real;
type Image=Record
             Point:Array[0..3,0..8191]of byte;
           end;
label eee;
Procedure HgcMode(Mode:Byte);
  Begin
    Inline
    ($EB/$1A/$00/          {          call  L0007              }
     $20/$61/$50/$52/$0F/  {          db    20H,61H,50H,52H,0FH }
     $19/$06/$19/$19/$02/  {          db    19H,06H,19H,19H,02H }
     $0D/$0B/$0C/$02/$35/  {          db    0DH,0BH,0CH,02H,35H }
     $2D/$2E/$07/$5B/$02/  {          db    2DH,2EH,07H,5BH,02H }
     $58/$58/$02/$03/$00/  {          db    58H,58H,02H,03H,00H }
     $00/                  {          db    00H                 }
     $58/                  { L0007:   pop   ax                  }
     $8B/$76/$04/          {          mov   si,[bp+4]           }
     $0B/$F6/              {          or    si,si               }
     $74/$03/              {          jz    L0008               }
     $BE/$0D/$00/          {          mov   si,000DH            }
     $03/$F0/              { L0008:   add   si,ax               }
     $BA/$BF/$03/          {          mov   dx,03BFH            }
     $B0/$01/              {          mov   al,01               }
     $EE/                  {          out   dx,al               }
     $2E/$8A/$04/          {          mov   al,cs:[si]          }
     $B2/$B8/              {          mov   dl,0B8H             }
     $EE/                  {          out   dx,al               }
     $33/$C9/              {          xor   cx,cx               }
     $B2/$B4/              {          mov   dl,0B4H             }
     $8A/$C1/              { L0009:   mov   al,cl               }
     $EE/                  {          out   dx,al               }
     $46/                  {          inc   si                  }
     $2E/$8A/$04/          {          mov   al,cs:[si]          }
     $42/                  {          inc   dx                  }
     $EE/                  {          out   dx,al               }
     $4A/                  {          dec   dx                  }
     $41/                  {          inc   cx                  }
     $80/$F9/$0C/          {          cmp   cl,0CH              }
     $75/$F0/              {          jnz   L0009               }
     $83/$EE/$0C/          {          sub   si,0CH              }
     $2E/$8A/$04/          {          mov   al,cs:[si]          }
     $0C/$08/              {          or    al,08               }
     $B2/$B8/              {          mov   dl,0B8H             }
     $EE)                  {          out   dx,al               }
  End;
{ ------------------------------------------------------------ }
```

```
Procedure Clear_Image;
  Begin
    Inline
    ($B9/$00/$80/          {       mov   cx,8000H          }
     $B8/$00/$B0/          {       mov   ax,0B000H         }
     $8E/$C0/              {       mov   es,ax             }
     $33/$C0/              {       xor   ax,ax             }
     $33/$FF/              {       xor   di,di             }
     $FC/                  {       cld                     }
     $F3/$AB)              {       rep   stosw             }
  End;

{ ------------------------------------------------------------------------ }

Function BaseAddr(X,Y:Integer):Integer;
  Begin
    Inline
    ($8B/$56/$04/          {       mov   dx,[bp+4]         }
     $B8/$03/$00/          {       mov   ax,3              }
     $23/$C2/              {       and   ax,dx             }
     $B1/$0D/              {       mov   cl,13             }
     $D3/$E0/              {       shl   ax,cl             }
     $50/                  {       push  ax                }
     $D1/$EA/              {       shr   dx,1              }
     $D1/$EA/              {       shr   dx,1              }
     $8B/$C2/              {       mov   ax,dx             }
     $B1/$5A/              {       mov   cl,90             }
     $F6/$E1/              {       mul   cl                }
     $5A/                  {       pop   dx                }
     $03/$C2/              {       add   ax,dx             }
     $8B/$56/$06/          {       mov   dx,[bp+6]         }
     $B1/$03/              {       mov   cl,3              }
     $D3/$EA/              {       shr   dx,cl             }
     $03/$C2/              {       add   ax,dx             }
     $89/$46/$08)          {       mov   [bp+8],ax         }
  End;

{ ------------------------------------------------------------------------ }

Procedure DrawPoint(X,Y:Integer;Color:Byte);
  Begin
    Inline
    ($50/                  {       push  ax                }
     $8B/$46/$08/          {       mov   ax,[bp+8]         }
     $50/                  {       push  ax                }
     $8B/$46/$06/          {       mov   ax,[bp+6]         }
     $50/                  {       push  ax                }
     $BB/BaseAddr/         {       mov   bx,BaseAddr       }
     $FF/$D3/              {       call  bx                }
     $83/$EC/$06/          {       sub   sp,6              }
     $59/                  {       pop   ax                }
     $5A/                  {       pop   dx                }
     $B8/$07/$00/          {       mov   ax,7              }
     $23/$C2/              {       and   ax,dx             }
     $8A/$C8/              {       mov   cl,al             }
     $B0/$80/              {       mov   al,80H            }
     $D2/$E8/              {       shr   al,cl             }
     $5E/                  {       pop   si                }
     $BB/$00/$B0/          {       mov   bx,0B000H         }
```

```
    $8E/$C3/            {           mov   es,bx             }
    $26/$8A/$24/        {           mov   ah,es:[si]        }
    $8A/$4E/$04/        {           mov   cl,[bp+4]         }
    $08/$C9/            {           or    cl,cl             }
    $75/$07/            {           jnz   L0010             }
    $F6/$D0/            {           not   al                }
    $22/$C4/            {           and   al,ah             }
    $EB/$03/$90/        {           jmp   L0011             }
    $0A/$C4/            { L0010:    or    al,ah             }
    $26/$88/$04)        { L0011:    mov   es:[si],al        }
  End;

{ ------------------------------------------------------------------------ }


Procedure Line_AX(X1,X2,Y:Integer;Step:Integer);
  Var I : Integer;
  Begin
    I:=X1;
    Repeat
      DrawPoint(I,Y,255);
      I:=I+Step
    Until I>=X2
  End;

{ ------------------------------------------------------------------------ }

Procedure Line_AY(Y1,Y2,X:Integer;Step:Integer);
  Var I : Integer;
  Begin
    I:=Y1;
    Repeat
      DrawPoint(X,I,255);
      I:=I+Step
    Until I>=Y2
  End;

{ ------------------------------------------------------------------------ }

Procedure DrawSquare(X1,Y1,X2,Y2:Integer);
  Begin
    Line_AX(X1,X2,Y1,1);
    Line_AX(X1,X2,Y2,1);
    Line_AY(Y1,Y2,X1,1);
    Line_AY(Y1,Y2,X2,1)
  End;

{ ------------------------------------------------------------------------ }


Procedure DrawLine(X1,Y1,X2,Y2:Integer);
  Var DeltaX,DeltaY,XStep,YStep,Direction : Integer;
  Begin
    DeltaX:=Abs(X2-X1);
    DeltaY:=Abs(Y2-Y1);
    IF X1>X2 Then XStep:=-1 Else XStep:=1;
    IF Y1>Y2 Then YStep:=-1 Else YStep:=1;
    IF DeltaX=0 Then Direction:=-1
    Else IF DeltaY=0 Then Direction:=1
         Else Direction:=0;
```

```
    While (Not((X1=X2) AND (Y1=Y2)))
    Do Begin
         DrawPoint(X1,Y1,255);
         IF Direction<0
           Then Begin
                  Y1:=Y1+YStep;
                  Direction:=Direction+DeltaX
                End
           Else Begin
                  X1:=X1+XStep;
                  Direction:=Direction-DeltaY
                End
      End
  End;

{ ------------------------------------------------------------------------ }



Var
     n:byte;
     ku,ku:array[0..705] of integer;
     a:array[0..705] of byte;

Procedure System_IdenII;
var i,j,k:integer;
    temp1,temp2:real;
    tv:array[1..10] of real;
begin
  if first=TRUE then
     begin
       first:=FALSE;
       for i:=1 to nv do
          begin
            for j:=1 to nv do
              P0[i,j]:=0;
            P0[i,i]:=1e5;
          end;
     end;
  H[1]:=u[9];
  H[2]:=u[8];
  H[3]:=y[9];
  H[4]:=y[8];
  temp1:=0.99;
  for i:=1 to nv do
    for j:=1 to nv do
       temp1:=temp1+H[j]*P0[j,i]*H[i];

  for i:=1 to nv do
    begin
      tv[i]:=0;
      for j:=1 to nv do
         tv[i]:=tv[i]+H[j]*P0[j,i];
    end;

  for i:=1 to nv do
     for j:=1 to nv do
       P0[i,j]:=(P0[i,j]-tv[i]*tv[j]/temp1)/0.99;
```

```
  temp2:=0;
  for i:=1 to nv do temp2:=temp2+H[i]*coef[i];
  temp2:=y[10]-temp2;ech:=temp2;

  for i:=1 to nv do
    for j:=1 to nv do
      coef[i]:=coef[i]+P0[i,j]*H[j]*temp2;

end; { System_IdenII }


Procedure Controller1;
begin

  u[6]:=u[7];
  u[7]:=u[8];
  u[8]:=u[9];
  u[9]:=u[10];
  u[10]:=(sp-y[10])*1-0.1*u[9];
  port[571]:=trunc(u[10]*0.6)+128;
end;

Procedure Controller2;
begin
  if (coef[1]<>0) and (i>100) then
    begin
      b:=-coef[2]/coef[1];
      r1:=b*(b*b+p1*b+p2)/(b*b-coef[3]*b-coef[4])-b;
      s0:=(p1+coef[3]-r1)/coef[1];
      s1:=(-coef[4]*r1)/coef[1]/b;
      t0:=(1+p1+p2)/coef[1]/(1-b);
    end;
  u[8]:=u[9];
  u[9]:=u[10];
  u[10]:=t0*sp-s0*y[10]-s1*y[9]-r1*u[9];
end;

Procedure Controller3;
begin
  if (coef[1]<>0) and (i>10)  then
    begin
      k:=coef[1];
      b:=-coef[2]/coef[1];
      a0:=-coef[3];a1:=-coef[4];

      r1:=b*b*(b*b+p1*b+p2)/(b*b+b*a0+a1)/(b-1) - b;
      s2:=-a1*r1/k/b;
      s1:=(s2*k-r1*a0+a1*(r1-1))/k/b;
      s0:=(p1-a0-(r1-1))/k;
      t0:=(1+p1+p2)/(k*(1-b));

    end else delay(1);

  u[8]:=u[9];
  u[9]:=u[10];
  u[10]:=t0*sp-s0*y[10]-s1*y[9]-s2*y[8]-(r1-1)*u[9]+r1*u[8];
end;

Procedure Process1;
begin
```

```
  Port[575]:=0;
  y[6]:=y[7];
  y[7]:=y[8];
  y[8]:=y[9];
  y[9]:=y[10];
  y[10]:= (port[575]-128)/0.4;
end;

Procedure Process2;
begin
  y[6]:=y[7];
  y[7]:=y[8];
  y[8]:=y[9];
  y[9]:=y[10];
{  y[10]:= 0.00991*u[9]+0.00938*u[8]+1.8394*y[9]-0.8490*y[8];}
  y[10]:= gain*(0.0102*u[9]+0.0099*u[8])+1.9114*y[9]-0.9214*y[8];
end;

Begin
  { initialize }
  port[571]:=128;readln;
  first:=TRUE;nv:=4;sign:=-1;
  sp:=-50;j:=0;gain:=1;m:=1.4;
  for i:=1 to 10 do
     begin
       u[i]:=0;
       y[i]:=0;
       e[i]:=0;
       coef[i]:=0;
     end;
  p1:=-1.832;p2:=0.839;
  t0:=0.01;r1:=0;s0:=0.01;s1:=0;s2:=0;
  HgcMode(1);
  Clear_Image;
  DrawSquare(0,0,719,257);
  Line_Ax(9,714,128,1);
  Line_Ay(1,257,9,1);
  for i:=1 to 705 do km[i]:=100;

  repeat
  for i:=0 to 599 do
    begin
{      if i mod 200 =0 then begin sp:=-sp;end;}
      if i mod 300 =0 then begin sp:=-sp;end;
{      if i= 300  then begin gain:=gain*m;end;}
{      if i = 150  then begin gain:=gain*m;end;
      if i = 450  then begin gain:=gain/m;end;
 }
      controller3;
      if abs(u[10])>70000. then goto eee;

      if u[10]>320 then u[10]:=320;
      if u[10]<-320 then u[10]:=-320;
      kco:=128+trunc(u[10]*0.4);
      port[571]:=kco;
      process1;
      system_idenii;
      if coef[1]>128/500 then kco:=255;
      if coef[1]<-128/500 then kco:=0;
      if abs(coef[1])<128/500 then kco:=128-trunc(coef[1]*500);
```

```
      DrawPoint(i+10,kco,1);
      if coef[1]>128/500 then kco:=255;
      if coef[1]<-128/500 then kco:=0;
      if abs(coef[1])<128/500 then kco:=128-trunc(coef[2]*500);
      DrawPoint(i+10,kco,1);
      if coef[1]>2.56 then kco:=255;
      if coef[1]<-2.56 then kco:=0;
      if abs(coef[1])<2.56 then kco:=128-trunc(coef[3]*50);
      DrawPoint(i+10,kco,1);
      if coef[1]>2.56 then kco:=255;
      if coef[1]<-2.56 then kco:=0;
      if abs(coef[1])<2.56 then kco:=128-trunc(coef[4]*50);
      DrawPoint(i+10,kco,1);

      if u[10]>320 then ku[i]:=255;
      if u[10]<-320 then ku[i]:=0;
      if abs(u[10])<320 then ku[i]:=128-trunc(u[10]*0.4);
      DrawPoint(i+10,ku[i],1);
      if y[10]>320 then km[i]:=255;
      if y[10]<-320 then km[i]:=0;
      if abs(y[10])<320 then km[i]:=128-trunc(y[10]*0.4);
      DrawPoint(i+10,km[i],1);

{      delay(150);}
{     writeln(coef[1]:10,':',coef[2]:10,':',coef[3]:10,':',coef[4]:10,':err',ech:8);
     writeln(' ',t0:10,':',r1:10,':',s0:10,':',s1:10,':',s2:10);
     writeln(i,':',sp:10,' ',y[10]:10,' ',u[10]:10);
}    end;
  until KeyPressed;
eee:
  HgcMode(0);
  ClrScr;

End.

A>
```

กิตติกรรมประกาศ

ผู้เขียนขอขอบพระคุณ อาจารย์วันชัย ริ้วรุจา อาจารย์ที่ปรึกษาปริญญานิพนธ์ ที่ได้ช่วยเหลือให้คำแนะนำ ข้อคิดเห็นต่าง ๆ ตลอดจนกำลังใจในการทำงานอย่างมาก นอกจากนี้ขอขอบคุณ นายธนากรณ์ มาไพศาลสิน ที่ได้ช่วยเหลือในการทำต้นฉบับ และนายชูชัย น้อยกร ที่ได้ช่วยเหลือในด้านโปรแกรมกราฟฟิก

หากปริญญานิพนธ์นี้พอจะมีประโยชน์อยู่บ้าง ขอมอบความดีนี้ให้ผู้มีพระคุณทั้งหลาย

หนังสืออ้างอิง

1. P.E.Wellstead , P.Prager, and P.Zanker, "Pole assignment self-tuning regulator", PROC.IEE , Vol 126 ,No 8,1979, pp 781-787

2. R.Hasting-James and M.W.Sage, "Recursive generalised-least-squares procedure of online identification of process parameters", PROC.IEE,Vol 116, No 12,1969, pp 2057-2062

3. T.C.Hsia, "On Least Squares Algorithms for System Parameter Identification",IEEE.Trans.on Automatic Control, Vol 2,1976, pp 104-107

4. D.Graupe,"Identification of Systems",Robert E. Krieger Publishing,276 p,1976

5. K.J.Astrom and B.Wittenmark,"Computer controlled system theory and design",Prentice-Hall, pp 93-398, 1984

6. G.Stepanopoulos, "Chemical process control", Prentice-Hall, pp 45-279, 1984

7. F.G.Shinskey, "Process Control System" ,McGraw-Hill, pp 156-164, 1979

8. V.W.Everleigh, "Adaptive Control and Optimization Techniques" ,McGraw-Hill ,434 p, 1967

9. C.H.Houpis and G.B.Lamont ,"Digital Control Systems" ,McGraw-Hill ,667p ,1985

10. Yokogawa Hokushin Electric ,"YEW series 80 Electronic control system",Japan, 1985

11. บุญเสริม แจ้งอรุณ, "การจำลองตัวควบคุมอุณหภูมิแบบจูนปรับตัวเองโดยใช้ไมโครคอมพิวเตอร์" ,การประชุมทางวิชาการวิศวกรรมไฟฟ้าครั้งที่ 10 เล่ม 2, หน้า 109-119 , 2530