ตัวควบคุมแบบฟัซซี่สำหรับอินเวอร์เท็ดเพนดูลัม

Inverted Pendulum Fuzzy Controller

โดย

นาย นันทวัช อัชชารกูร

รหัส 42067051

อาจารย์ที่ปรึกษา

ดร. อาริต ธรรมโน

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน

หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ภาคเรียนที่ 2 ปีการศึกษา 2543

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชื่อหัวข้อ	ตัวควบคุมแบบฟัซซี่สำหรับอินเวอร์เท็ดเพนดูลัม
นักศึกษา	นาย นันทวัช อัชชารกูร
อาจารย์ที่ปรึกษา	ดร. อาริต ธรรมโน
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2543

บทคัดย่อ

อินเวอร์เท็ดเพนดูลัม (เพนดูลัมที่ตั้งกลับหัว) เป็นระบบทางกลที่จะไม่มีเสถียรภาพด้วยตัวมันเองอยู่ตลอดเวลา (สามารถอธิบายได้โดยสมการทางอนุพันธ์ลำดับที่สอง) ถ้าปราศจากการควบคุมที่ดี โดยส่วนประกอบทางกลจะประกอบด้วย แท่งเพนดูลัมที่ติดตั้งอยู่บนรถเลื่อน โดยตัวเพนดูลัมและรถเลื่อนสามารถเคลื่อนที่ได้ในแกนเดียว (คือ สามารถเคลื่อนที่ไปและกลับตามลู่เส้นตรงได้เท่านั้น) และที่ปลายของแท่งเพนดูลัมจะมีตุ้มน้ำหนักติดอยู่ โดยระบบอินเวอร์เท็ดเพนดูลัมนั้นจะเป็นระบบกระบวนการแบบที่มีอินพุทเดียวแต่มีหลายเอาท์พุท (โดย อินพุทของระบบนี้คือ แรงจากมอเตอร์ที่ขับสายพานไปผลักตัวรถเลื่อน เอาท์พุทของระบบนี้จะเป็นมุมของเพนดูลัม และตำแหน่งของตัวรถเลื่อน) สัญญาณที่ส่งไปทำการควบคุมเพียงตัวเดียวของระบบนี้จะใช้สั่งการขับเคลื่อนมอเตอร์สายพานเพื่อไปผลักตัวรถเลื่อน

จุดประสงค์ของโครงงานนี้คือต้องการที่จะควบคุมเพนดูลัมให้ตั้งสมดุลอยู่ที่ตำแหน่งตรงกลางของทางวิ่ง นั่นคือต้องการควบคุมให้ได้ทั้งมุมของเพนดูลัมและตำแหน่งของตัวรถเลื่อน ตัวควบคุมที่ใช้สำหรับระบบนี้จะเป็นตัวควบคุมแบบฟัซซี่ ซึ่งจะมีข้อดีกว่าการใช้ตัวควบคุมแบบอื่นตรงที่สามารถมีได้หลายอินพุทและหลายเอาท์พุท และไม่ยุ่งยากที่จะต้องหาสมการแบบจำลองทางคณิตศาสตร์ของระบบออกมาก่อน ตัวควบคุมจะสร้างเป็นซอฟท์แวร์ในลักษณะที่ติดต่อกับผู้ใช้งานแบบกราฟฟิกที่เป็นรูปภาพเคลื่อนไหวจำลองการทำงานจริงของอินเวอร์เท็ดเพนดูลัม และให้ผู้ใช้งานสามารถปรับแต่งค่าอัตราขยายในการควบคุมได้เพื่อเปรียบเทียบข้อดีข้อเสียในการปรับอัตราขยายค่าต่างๆ และทำการปรับค่าอัตราขยายให้อยู่ในค่าที่เหมาะสมกับระบบได้

Title	Inverted Pendulum Fuzzy Controller
Student	Mr. Nuntawat Asharakul
Advisor	Dr. Arit Thammano
Level of Study	Master of Science in Information Technology
Major	Information Science
Academic Year	2000

Abstracts

An Inverted Pendulum is a mechanical system which is an inherently unstable system (described by second-order differential equations) without providing a good control. The mechanical system consists of a pendulum rod mounted on a moving cart, the pendulum and cart have one degree of freedom (i.e. the cart can only move back and forth along the track), and there is a pitch mass at the end of the pendulum rod. The inverted pendulum system is a single-input-multi-output process system (i.e. the single input of this system is the force from pulley motor to the cart, the outputs of this system are the pendulum angle and the cart position.). The single control signal is used to drive the pulley motor to force the cart.

The project objective is normally to balance the pendulum with the cart in the middle of the track. The controller of this system is using Fuzzy Control Algorithm, its advantage prior to the other control algorithm is that it could be multi-input and multi-output system, and not necessary to find out any complex mathematical model equation of the system. The creating controller is a graphic user interface software that includes an animation graphic of a running inverted pendulum system, and user can tune the controller gain in any different value to be most suitable to the system.

กิตติกรรมประกาศ

โครงงานชุดนี้สำเร็จลุล่วงได้ด้วยดี ด้วยคำแนะนำมากมายรวมทั้งความเอาใจใส่ดูแลที่มีค่ายิ่งจาก *ดร.อาริต ธรรมโน* ซึ่งเป็นอาจารย์ที่ปรึกษาของโครงงานนี้ ผู้จัดทำรู้สึกซาบซึ้งในความช่วยเหลือและขอกราบขอบพระคุณท่านอาจารย์ผู้นี้เป็นอย่างสูง

นอกจากนี้ต้องขอขอบคุณเพื่อนในรุ่น IS7 หลายท่านที่ได้ให้แนะนำช่วยเหลือผู้จัดทำในเรื่องแนวทางการเขียนและแก้ไขโปรแกรมให้สามารถทำงานได้ประสบผลสำเร็จ

ขอบคุณบุคลากรในคณะเทคโนโลยีสารสนเทศทุกท่านทั้งที่ SCB(ปัจจุบันย้ายไปอยู่ตึก Shin'3) และที่ลาดกระบัง ที่ให้ความช่วยเหลือในด้านการศึกษาและ ในด้านการบริการต่างๆ อย่างน่าประทับใจ

ขอบคุณพ่อ แม่และพี่สาวของผู้จัดทำ ที่คอยห่วงใยและให้กำลังใจผู้จัดทำตลอดเวลา

ขอบคุณ บริษัท OMRON Electronic (Thailand) ซึ่งได้อนุญาตให้ผู้จัดทำสละเวลาการทำงานส่วนหนึ่งมาทำโครงงานนี้จนประสบความสำเร็จ แม้ว่าจะต้องเสียเวลาการทำงานไปบ้างแต่ทางบริษัทไม่เคยได้ต่อว่าแต่ยังให้การสนับสนุน และยังให้ใช้เครื่องคอมพิวเตอร์และเครื่องพิมพ์ในการพิมพ์เอกสารโครงงานอีกด้วย

สุดท้ายขอกราบขอบพระคุณ คุณพระศรีรัตนตรัย และสิ่งศักดิ์สิทธิ์ทั้งหลาย ที่ช่วยเป็นที่พึ่งทางจิตใจ ดลบันดาลให้ผู้จัดทำสามารถฝ่าฟันอุปสรรคต่างๆ จนประสบความสำเร็จลุล่วงไปได้ด้วยดี

คุณค่าและประโยชน์อันพึงได้รับจากโครงงานนี้ ผู้จัดทำขอยกประโยชน์และความดีทั้งหมดให้แด่ผู้มีอุปการะคุณทุกท่าน

นันทวัช อัชชารกูร

ผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

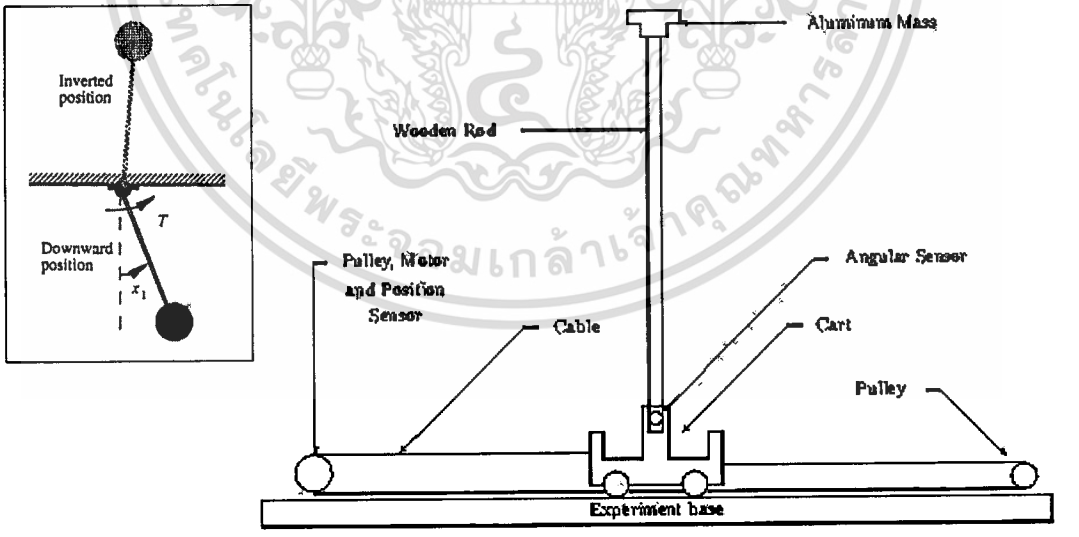
สารบัญภาพ

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

บทที่ 1
บทนำ

การศึกษาการควบคุมอินเวอร์เท็ดเพนดูลัมเป็นตัวอย่างของการศึกษาทฤษฎีระบบควบคุมอัตโนมติเพื่อนำไปใช้ในการควบคุมระบบทั่วไป ซึ่งการศึกษามักจะเริ่มจากการควบคุมระบบทางกลที่มีโครงสร้างไม่ยุ่งยากนัก อินเวอร์เท็ดเพนดูลัมถือเป็นตัวอย่างหนึ่งที่นำมาใช้ในการศึกษาและเป็นต้นแบบในการออกแบบและทดสอบชุดควบคุมที่มีต่อการทำงานของระบบ เนื่องจากอินเวอร์เท็ดเพนดูลัมเป็นระบบที่ไม่เสถียรและจะอยู่ในสภาวะที่สมดุลไม่ได้เลยถ้าปราศจากการควบคุมที่ดี โดยการศึกษาการควบคุมอินเวอร์เท็ดเพนดูลัมนั้นได้ทำการศึกษาและพัฒนารูปแบบและวิธีการควบคุมเรื่อยมาหลายวิธีการ ซึ่งการศึกษาการควบคุมอินเวอร์เท็ดเพนดูลัมสำหรับโครงงานนี้ จะเป็นการประยุกต์ใช้วิธีการควบคุมแบบฟัซซี่ซึ่งถือเป็นวิธีการควบคุมที่มีความยืดหยุ่นสูงเนื่องจากไม่ได้ยึดติดกับพารามิเตอร์ในการคำนวณและไม่ได้เกี่ยวข้องกับทฤษฎีการออกแบบจำลองการควบคุมทางคณิตศาสตร์มากนักซึ่งแบบจำลองทางคณิตศาสตร์นั้นถ้าออกแบบมาไม่ดีหรือไม่ครอบคลุมถึงทั้งระบบอย่างพอเพียงก็อาจทำการควบคุมได้ไม่ดีพอ



รูปที่ 1.1 ไดอะแกรมทางกล (Mechanical Diagram) ของอินเวอร์เท็ดเพนดูลัม โดยรูปมุมบนซ้ายแสดงถึงเพนดูลัมแบบธรรมดา (Simple Pendulum) ซึ่งจะตกลงสู่จุดสมดุลตรงกลางได้อย่างมีเสถียรภาพเมื่อไม่มีแรงรบกวนจากภายนอกมากระทำ แต่ถ้าเป็นอินเวอร์เท็ดเพนดูลัมอยู่ในตำแหน่งตั้งกลับหัวจะล้มลงไม่สามารถรักษาเสถียรภาพได้ที่จุดสมดุลตรงกลางถ้าปราศจากควบคุมโดยต้องทำการส่งแรงจากภายนอกมากระทำตลอดเวลา

1.1 ประวัติความเป็นมาและความสำคัญของปัญหา

การศึกษาการควบอินเวอร์เท็ดเพนดูลัมได้เคยทำการศึกษาและสร้างชุดทดลองกันมาแล้วหลายครั้ง โดยมีการศึกษาพัฒนาขึ้นมาเป็นลำดับที่น่าสนใจดังต่อไปนี้

Schaefer และ Cannon (1966) ได้ออกแบบชุดควบคุมอินเวอร์เท็ดเพนดูลัมโดยใช้ทฤษฎี Bang-bang Control ซึ่งส่งค่าแรงกระทำให้กับระบบได้เพียง 3 ขั้น คือ บวก (+) ลบ (-) และศูนย์ (0)

Rwyson และ Luenberger (1970) สร้างแบบจำลองทางคณิตศาสตร์ของอินเวอร์เท็ดเพนดูลัม โดยใช้วิธี State Space และการประเมินค่าตัวแปรของ Luenberger ซึ่งเป็นทฤษฎีที่ประยุกต์ไปสู่การควบคุมการปล่อยจรวดในเวลาต่อมา

Shozo Mori, Hiroyoshi Nishhara และ Katsuhisa Furuta (1975) ได้ออกแบบชุดควบคุมอินเวอร์เท็ดเพนดูลัมในลักษณะปล่อยให้ห้อยลงไปตามปกติเองก่อน แล้วทำการควบคุมให้ขึ้นมาตั้งกลับหัวโดยประยุกต์รวมเอาวิธี Bang-bang Control และ State Space เข้าไว้ในการควบคุมด้วยกัน โดยใช้ไมโครคอมพิวเตอร์เป็นตัวสร้างสัญญาณ Bang-bang Control ให้ทำเพนดูลัมขึ้นมาตั้งก่อนในช่วงแรกแล้วจึงตัดระบบไปสู่ชุดควบคุมอนาลอกแบบ State Space ควบคุมให้ตั้งตรงอยู่ต่อไป

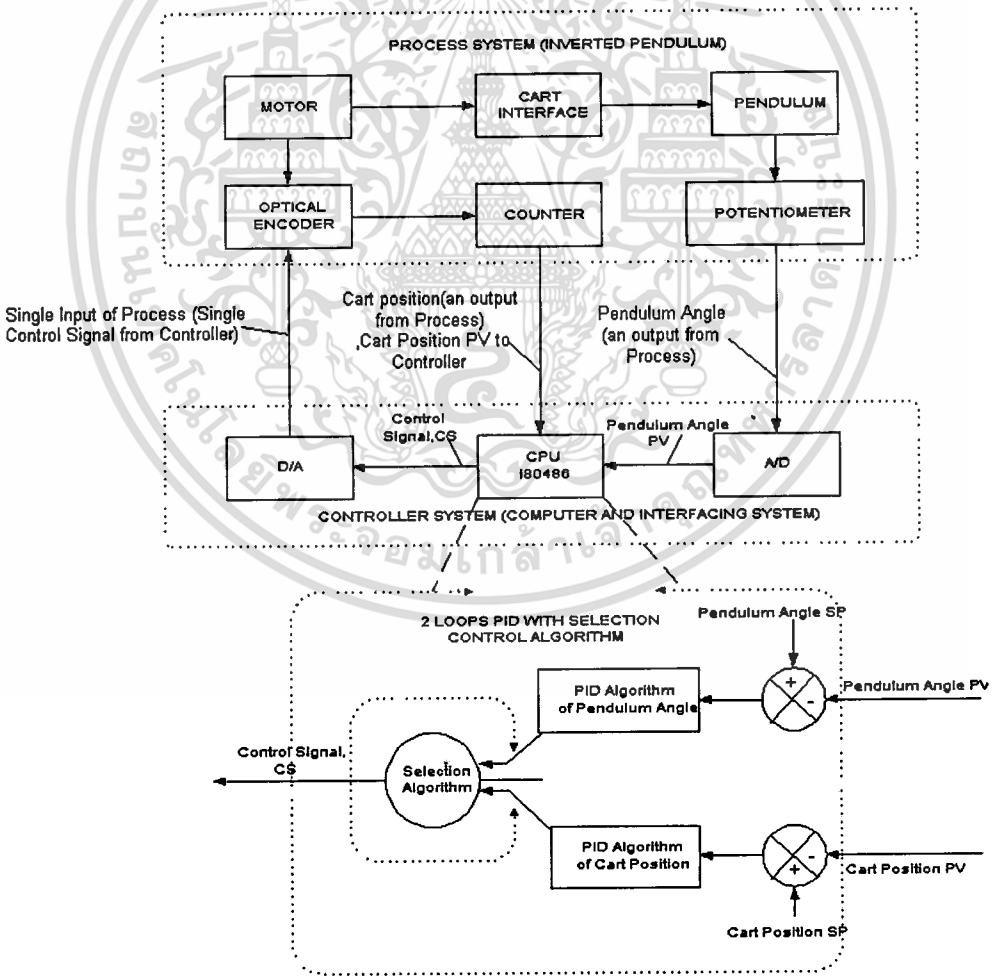
Katsuhisa Furata ได้ทำการพัฒนาชุดควบคุมร่วมกับ Hiroyuki Hajiware และ Kazuhiro Kosuge (1980) ต่อไปโดยเปลี่ยนมาใช้มินิคอมพิวเตอร์ NOVA 1200 และใช้ Sampling Period 4 ms และสร้างชุดควบคุมอินเวอร์เท็ดเพนดูลัมบนรางเอียงขึ้น โดยใช้วิธีการควบคุมให้สมดุลแบบ State Space ที่ยังใช้ชุดควบคุมแบบอนาลอกเช่นเดิม

W. Duchting, W. Schonberger, U. Gotte และ P.Arndt (1983) เปลี่ยนรูปแบบจากระบบอินเวอร์เท็ดเพนดูลัมเป็นระบบลูกเหล็กกลิ้งบนรางโค้งบนตัวรถ และได้ใช้วิธีการสร้าง Root Locus และเปลี่ยนคุณลักษณะไปเมื่อเปลี่ยนค่าอัตราขยายแต่ละตัว แต่ความสัมพันธ์ของสมการการเคลื่อนที่ยังคงมีลักษณะเหมือนเดิม

Feng Zu-ren, Yin Zheng-qi และ Chen Hui-tang(1987) สร้างชุดควบคุมดับเบิลอินเวอร์เท็ดเพนดูลัมโดยใช้ไมโครโพรเซสเซอร์ TP-801 เป็นชุดควบคุมแบบดิจิตอลใช้ Sampling Period ได้สูงถึง 20 ms และใช้วิธีการสร้าง Discrete State Equation ทำการควบคุมโดยเปลี่ยนค่าอัตราขยายของตัวควบคุมไปเรื่อยๆ

พงษ์เทพ วิทิศลัดดา (1993) ได้สร้างชุดควบคุมดิจิตอลอินเวอร์เท็ดเพนดูลัมโดยใช้ไมโครคอมพิวเตอร์ PC AT 80286 ความเร็ว 8 MHz เป็นตัวควบคุม และใช้วิธี State Space จากผลการทดลองเนื่องจากค่าคงที่ทางเวลาของระบบ (system time constant) ที่ได้มีค่าประมาณ 2 วินาทีขึ้นไป จึงจำเป็นต้องใช้ระยะทางในการวิ่งของรถมากกว่า 0.2 เมตรที่ได้สร้างไว้ ทำให้ไม่สามารุควบคุมการทำงานได้

นันทวัช อัชชารกูร, วิชิต แซ่ล้อ และ สรายุธ สินศักดิ์จรุงเดช (1995) ได้สร้างชุดควบคุมดิจิตอลอินเวอร์เท็ดเพนดูลัมขึ้นโดยใช้ไมโครคอมพิวเตอร์ PC ที่ใช้ intel 80486 ความเร็ว 66 MHz เป็นตัวควบคุม และสร้างชุดตั้งเวลาด้วย 8253 ให้สามารถเปลี่ยนค่า Sampling Period ได้ตั้งแต่ 1 μs ขึ้นไป โดยใช้วิธีการควบคุมแบบ PID ซึ่งสามารถโดยปกติแล้วจะควบคุมได้เพียงเพียงแค่ลูปเดียว คือ เลือกจะควบคุมมุมของเพนดูลัมหรือควบคุมตำแหน่งของตัวรถ อย่างใดอย่างหนึ่ง ไม่สามารถควบคุมแบบสองลูปได้พร้อมกัน แต่ได้มีการลองประยุกต์ใช้วิธีการ Selection Control กับ PID Algorithm 2 ลูปแต่มีสัญญาณที่ส่งออกไปสั่งการควบคุมเพียงแค่ตัวเดียว คือ ให้ตัวควบคุมทำการเลือกว่าจะนำสัญญาณไปควบคุมมุมเพนดูลัมให้ได้ก่อนแล้วจึงค่อยๆ ทำการควบคุมตำแหน่งรถเลื่อนต่อไป โดยทำการทดลองปรับพารามิเตอร์ตัวควบคุม PID ของทั้งสองลูปแยกกันและทำการวัดการตอบสนองจากการปรับค่า PID ทั้งสองลูปนั้นให้เหมาะสม ตามบล็อกไดอะแกรมรูปที่ 2 ด้านล่างนี้



***รูปที่ 1.2** บล็อกไดอะแกรมองค์ประกอบระบบควบคุมอินเวอร์เท็ดเพนดูลัม และอัลกอริทึมในการควบคุมที่เป็น 2 loops PID with Selection Control.*

จากการที่ได้ทดลองปรับพารามิเตอร์หลายๆ ค่าของ PID ทั้งสองลูปแต่ก็ยังไม่สามารถทำการควบคุมทั้งมุมของเพนดูลัมและตำแหน่งรถเลื่อนไปพร้อมๆ กันในเวลาเดียวกันได้ เนื่องจากเป็นข้อจำกัดของวิธีการควบคุมแบบ PID ที่เหมาะกับการควบคุมระบบแบบอินพุทและเอาท์พุทเดียว (Single-input-single-output system) หรือถ้าจะใช้วิธีการแบบ Feedforward ร่วมกับ Cascade PID ก็สามารถใช้ควบคุมค่าเอาท์พุทของกระบวนการ (Process Output หรือ Process Variable) ได้เพียงตัวเดียวเท่านั้น

การควบคุมด้วยวิธีนี้จึงไม่เหมาะที่จะนำมาใช้ควบคุมอินเวอร์เท็ดเพนดูลัมที่ต้องการจะควบคุมให้ได้ทั้งมุมเพนดูลัมและตำแหน่งของตัวรถไปพร้อมๆ กัน

วิธีการควบคุมที่สามารถทำการควบคุมระบบกระบวนการที่มีอินพุทเดียวแต่มีหลายเอาท์พุทในลักษณะเช่น ระบบอินเวอร์เท็ดเพนดูลัมนี้ได้อย่างแน่นอนจริง ได้แก่ วิธีการ State-Space ซึ่งเป็นวิธีการที่จะต้องทำการหาแบบจำลองทางคณิตศาสตร์ (Mathematical Model) ของระบบออกมาให้ได้อย่างถูกต้อง แล้วต้องทำการวิเคราะห์ในเรื่องของการตอบสนองของระบบ เช่น และการวิเคราะห์การตอบสนองทางความถี่ (Frequency Response) โดยวิเคราะห์ Bode Plot , หาฟังก์ชันการถ่ายโอน (Transfer Function) ของระบบซึ่งจะหาออกมาเป็นสมการทางอนุพันธ์แล้วจึงแปลงรูปมาเป็นสมการรูปแบบลาปลาซ (Laplace Transform) แล้วจึงนำมาทำเป็นฟังก์ชันการควบคุมแบบป้อนกลับ (Feedback Control Function) ซึ่งจะต้องทำการวิเคราะห์แนวโน้มการควบคุมด้วยวิธีการต่างๆ เช่น การวิเคราะห์แบบ Root Locus แล้วนำมาทำการหาสมการของฟังก์ชันการควบคุมโดยสามารถเขียนได้ในรูปแบบของเมตริกซ์ที่มีหลายมิติซึ่งรูปแบบเมตริกซ์ของ State-Space นั้นสามารถนำเสนอได้ในลักษณะของสมการเชิงเส้นของระบบหลายๆ สมการ

การใช้วิธีการ State-Space ทำออกมาเป็นฟังก์ชันการควบคุมนั้น สามารถทำการควบคุมได้อย่างถูกต้องอย่างแท้จริงก็โดยที่จะต้องหาแบบจำลองทางคณิตศาสตร์ ทำการวิเคราะห์คำนวณ และทำการแปลงสมการออกมาได้อย่างถูกต้องในทุกขั้นตอนจริงๆ ซึ่งจะต้องทำการวิเคราะห์และคำนวณที่ยุ่งยากซับซ้อนและเสียเวลามาก และหากขาดตกบกพร่องไปแม้แต่สิ่งเดียว หรือแม้แต่ขั้นตอนเดียว ก็จะไม่สามารถทำการควบคุมระบบได้อย่างที่ต้องการ

1.2 ลักษณะในภาพรวมของวิธีการควบคุมแบบฟัซซี่

วิธีการควบคุมแบบฟัซซี่ (Fuzzy Control)เป็นวิธีการที่ใช้การวิเคราะห์ทางตรรกตามความรู้สึกนึกคิดของผู้วิเคราะห์และอาศัยแนวความคิดจากความรู้สึกและประสบการณ์ของผู้วิเคราะห์นั้นตั้งเป็นกฎเงื่อนไขพื้นฐาน (Rule-Base) ในการควบคุม โดยเป็นกฎที่นำอินพุทเข้ามากระทำการกันทางลอจิก(ทางตรรก)แบบฟัซซีลอจิก (Fuzzy Logic) บนพื้นฐานของฟัซซีเซ็ท (Fuzzy Set) โดยฟัซ

ซีลอจิกแตกต่างจากลอจิกแบบธรรมดา ตรงที่ค่าความจริง (predicate) ของฟัซซีลอจิกจะไม่ได้มีแค่ค่าที่เป็นจริงหรือเท็จ (true or false) ที่เป็นค่าลอจิก 1 หรือ 0 เท่านั้น แต่สามารถเป็นค่าระดับที่ต่อเนื่องได้ตั้งแต่ค่า 0 ถึง 1 (คือเป็นระดับลอจิกที่มีความกำกวมไม่สามารถแยกออกได้อย่างเด็ดขาดว่าจะเป็นลอจิก 0 หรือ 1) ซึ่งงานบางระบบอาจจะมีความกำกวมหรือลำบากเกินไปที่จะทำเป็นแบบจำลองทางคณิตศาสตร์ออกมาหรือทำออกมาได้ก็ซับซ้อนเกินไปในการที่จะมาทำการวิเคราะห์ต่อ วิธีการ Fuzzy Control จึงเป็นวิธีการที่สะดวกในการนำมาใช้สำหรับระบบงานในลักษณะนี้ เช่น งานระบบควบคุมที่มีความยุ่งยากซับซ้อน (Sophisticated Control system), งานพยากรณ์ทางธุรกิจ (Business forecasting) ฯลฯ

แต่เนื่องจากวิธีการนี้เป็นวิธีการที่จะต้องอาศัยความชำนาญและประสบการณ์ในการถ่ายทอดความรู้สึกนึกคิดในการแบ่งช่วงทางตรรกของผู้วิเคราะห์โดยส่วนใหญ่ ดังนั้นผู้วิเคราะห์จึงจะต้องรู้การตอบสนองในด้านต่างๆ ของระบบกระบวนการนั้นเป็นอย่างดีจึงจะถ่ายทอดความรู้สึกนึกคิดทางตรรกออกมาใช้กับวิธีการนี้ได้อย่างถูกต้อง ซึ่งถ้าผู้วิเคราะห์ไม่รู้จักระบบหรือไม่มีความชำนาญกับระบบนั้นดีพอก็อาจวิเคราะห์ออกมาไม่ถูกต้องและทำการควบคุมไม่ได้อย่างที่ต้องการเช่นกัน

1.2.1 ความเหมาะสมในการประยุกต์ใช้การควบคุมแบบฟัซซี่

การควบคุมแบบฟัซซี่นั้นจะเหมาะสมในการนำมาประยุกต์ใช้เป็นระบบควบคุมสำหรับระบบกระบวนการที่มีลักษณะดังต่อไปนี้:-

- เป็นระบบกระบวนการที่มีความซับซ้อนอย่างมาก ไม่สามารถสร้างแบบจำลองทางคณิตศาสตร์ออกมาในรูปง่ายได้
- เป็นระบบกระบวนการที่มีความไม่เป็นเชิงเส้นอย่างมาก ไม่สามารถแปลงหรือเขียนแทนออกมาในรูปของสมการเชิงเส้นได้
- รู้จักและมีความเชี่ยวชาญในระบบกระบวนการนั้นเป็นอย่างดี สามารถทำการประมวลผลทางแนวความคิดในการควบคุมกระบวนการนั้นออกมาได้อย่างชัดเจนแน่นอน

การควบคุมแบบฟัซซี่นั้นไม่เหมาะสมในการนำมาประยุกต์ใช้เป็นระบบควบคุมสำหรับระบบกระบวนการที่มีลักษณะดังต่อไปนี้:-

- การใช้วิธีการควบคุมแบบดั้งเดิมสามารถให้ผลการควบคุมออกมาได้อย่างดีมากอยู่แล้ว
- สามารถทำการแปลงเป็นรูปแบบสมการทางคณิศาสตร์ได้อย่างง่ายดาย หรือมีแบบจำลองทางคณิตศาสตร์ที่ดีพออยู่แล้ว

- ไม่สามารถถ่ายทอดแนวความคิดความรู้สึกฯในการควบคุมระบบกระบวนการนั้นออกมาได้อย่างชัดเจน

ตัวอย่างการใช้ระบบควบคุมแบบฟัซซี่:-

- การควบคุมประตูเขื่อนอัตโนมัติของโรงไฟฟ้าพลังน้ำ (Automatic control of dam gates for hydroelectric-powerplants):- Tokio Electric Pow.
- การควบคุมอย่างง่ายของหุ่นยนต์ (Simplified control of robots):- Hirota, Fuji Electric, Toshiba, Omron
- การเล็งกล้องสำหรับการถ่ายภาพกีฬาระยะไกล(Camera aiming for the telecast of sporting events):- Omron
- การป้องกันการแกว่งของอุณหภูมิที่ไม่ต้องการในเครื่องปรับอากาศ (Preventing unwanted temperature fluctuation in air-conditioning system):- Mitsubishi, sharp
- ระบบพยากรณ์ในการรับรู้แผ่นดินไหวล่วงหน้า(Prediction system for early recognition of earthquakes):- Inst. of Seismology Bureau of Metrology of Japan
- ระบบรับรู้ลายมือ วัตถุ เสียง(Recognition of handwriting, objects, voice):- CSK, Hitashi,. Hosai University, Ricoh

1.3 ความมุ่งหมายและวัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาลักษณะการทำงานและคุณลักษณะต่างๆ ของระบบควบคุมแบบฟัซซี่
2. สามารถนำระบบควบคุมแบบฟัซซี่ไปประยุกต์ใช้ในงานควบคุมระบบที่มีความซับซ้อนและไม่เป็นเชิงเส้น (Non-linear system) ได้อย่างมีประสิทธิภาพ
3. สามารถปรับแต่ง (Tuning) ระบบควบคุมแบบฟัซซี่ให้สามารถทำการควบคุมระบบกระบวนการของอินเวอร์เท็ดเพนดูลัมให้ตอบสนองต่อการควบคุมได้อย่างมีประสิทธิภาพ

1.4 สมมุติฐานของโครงงาน

อินเวอร์เท็ดเพนดูลัมซึ่งเป็นกระบวนการที่อยู่ภายใต้การควบคุมของโครงงานนี้ ถือเป็นกระบวนการที่จะมีตัวแปรที่อยู่ภายใต้การควบคุม 2 ตัวคือ มุมของแกนเพนดูลัม และระยะทางของตัวรถเลื่อน ซึ่งถือเป็นอินพุท 2 ตัวของตัวควบคุมแบบฟัซซี่ แต่เอาท์พุทที่จะส่งออกไปสั่งควบคุมกระบวนการจะมีเพียงตัวเดียวคือ แรงที่จะไปผลักตัวรถเลื่อน ซึ่งระบบควบคุมแบบฟัซซี่นี้เป็น

ระบบควบคุมที่มีได้หลายอินพุทและหลายเอาท์พุท (Multiple Input – Multiple Output) เพียงกำหนดกฎเงื่อนไข (Rule-base) ฟัซซี่เซ็ทและฟังก์ชันความเป็นสมาชิกที่เหมาะสม และทำการปรับแต่งค่าสเกลตัวขยายทางด้านอินพุทและเอาท์พุทให้ได้ค่าที่เหมาะสมกับกระบวนการนี้ ก็จะสามารถทำการควบคุมกระบวนการนี้ได้ดี

1.5 ขอบเขตของโครงงาน

โครงงานนี้จะเป็นการประยุกต์ใช้การควบคุมแบบฟัซซี่ในการเป็นตัวควบคุมอินเวอร์เท็ดเพนดูลัม โดยจะทำสร้างระบบจำลองการทำงานของระบบกระบวนการอินเวอร์เท็ดเพนดูลัมออกมาให้เหมือนระบบจริง และจะนำระบบควบคุมแบบฟัซซี่เข้ามาทำการควบคุมระบบให้มีเสถียรภาพอย่างที่ต้องการ โดยตัวจำลองการทำงานและระบบควบคุมที่จะนำเสนอจะสร้างขึ้นมาเป็นโปรแกรมที่มีการติดต่อกับผู้ใช้เป็นแบบกราฟฟิก (Graphical user interface: GUI) เพื่อให้ผู้ใช้สามารถทำการทดลองปรับแต่งค่าในการควบคุมให้มีการตอบสนองได้ตามที่ต้องการ และผู้นำไปศึกษาค้นคว้าเพิ่มเติมสามารถทำความเข้าใจและเรียนรู้ได้อย่างสะดวกและรวดเร็ว

1.6 ขั้นตอนการทำงานของโครงงาน

1. ศึกษาทฤษฎีของระบบฟัซซี่ ได้แก่ ฟัซซี่เซ็ท, ฟัซซี่ลอจิก, ฟังก์ชันความเป็นสมาชิกทางฟัซซี่ และอัลกอริทึมในการทำงานของระบบควบคุมแบบฟัซซี่
2. ศึกษาส่วนประกอบต่างๆ และค่าพารามิเตอร์ที่เกี่ยวข้องของระบบควบคุมแบบฟัซซี่ที่จะนำมาใช้ในการปรับแต่งระบบควบคุม เพื่อเลือกรูปแบบและค่าในการปรับแต่งระบบควบคุมที่เหมาะสม
3. สร้างแบบจำลองกระบวนการ (Process Simulation Model), สร้างตัวจำลองการทำงานของกระบวนการอินเวอร์เท็ดเพนดูลัม (Inverted Pendulum Simulator) และตัวควบคุมแบบฟัซซี่สำหรับกระบวนการนี้ โดยโครงงานนี้ จะเขียนใช้ภาษา C++ ในการสร้างและพัฒนาโปรแกรม
4. ทดสอบการทำงานของโปรแกรมและปรับปรุงแก้ไขระบบควบคุม โดยทดลองทำการปรับแต่งค่าในการควบคุมให้มีผลตอบสนองการควบคุมออกมาในรูปแบบที่ดีที่สุด
5. จัดทำเอกสารประกอบโครงงาน

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

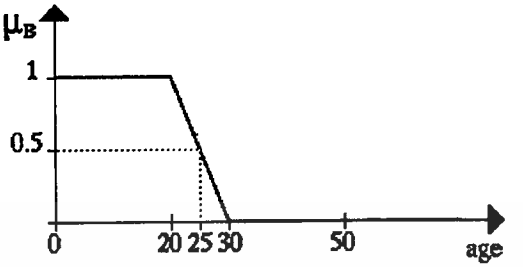
2.1 ระบบฟัซซี่ (Fuzzy System)

2.1.1 ฟัซซี่เซ็ท (Fuzzy Set)

เซ็ทของค่าทางคณิตศาสตร์ โดยทั่วไปแล้วจะเป็นลักษณะอย่างง่ายๆ คือ ค่านั้นอยู่ในเซ็ทนี้หรือค่านั้นไม่อยู่ในเซ็ทนี้ เช่น เซ็ท A เป็นเซ็ทของค่าจำนวนจริงตั้งแต่ 5 ถึง 8 เขียนแทนเป็นช่วงได้เป็น $A=[5,8]$ ดังรูปที่ 2.1

รูปที่ 2.1 เซ็ทแบบธรรมดา (Crisp Set) ของ A=[5,8]

เซ็ทลักษณะตามรูปที่ 2.1 นั้น ถ้าค่าขององค์ประกอบใดๆไม่เป็นสมาชิกของเซ็ทนี้ก็จะถือว่าองค์ประกอบนั้นมีค่าทางตรรกเป็น 0 แต่ถ้าค่าขององค์ประกอบนั้นเป็นสมาชิกของเซ็ทนี้ก็จะถือว่าองค์ประกอบนั้นมีค่าทางตรรกเป็น 1 เซ็ทแบบนี้จะถือเป็นเซ็ทแบบปกติที่เรียกว่า *Crisp Set* ซึ่งค่าความเป็นสมาชิกทางตรรกที่มี 2 ค่า (0 กับ 1) นั้นจะไม่ละเอียดและไม่มีความยืดหยุ่นพอสำหรับการนำไปใช้งานในบางกรณี เช่น ถ้าบอกว่าเซ็ท *B={เซ็ทของผู้เยาว์}=[อายุ0ปี,อายุ20ปี]* นั่นหมายความว่าถ้าอายุ 20 ปี 1 วันก็ไม่ถือว่าเป็นผู้เยาว์แล้ว ซึ่งดูแล้วการตีความจะไม่ดีพอสำหรับกรณีนี้ ควรจะบอกว่าอายุ 0 ถึง 20 ยังมีความเป็นผู้เยาว์อยู่ พออายุมากกว่า 20 ขึ้นไปก็เริ่มมีความเป็นผู้เยาว์น้อยลง และหมดความเป็นผู้เยาว์ตอนอายุ 30 ปี ดังรูปที่ 2.2



รูปที่ 2.2 ถ้าอายุ 0 ถึง 20 ปีจะมีความเป็นผู้เยาว์เป็น 1 ถ้าอายุ 25 ปีก็จะถือว่ามีความเป็นผู้เยาว์อยู่ 0.5

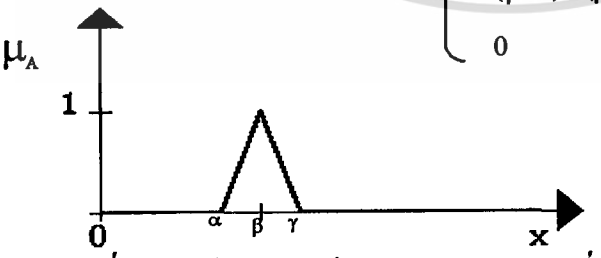
รูปแบบเซ็ทตามรูปที่ 2.2 นั้นสามารถมีค่าทางตรรกของความเป็นสมาชิกได้ตั้งแต่ค่า 0 ถึง 1 เซ็ทแบบนี้จะถือเป็น*ฟัซซี่เซ็ท (Fuzzy Set)*

2.1.2 ฟังก์ชันความเป็นสมาชิก (Membership function)

ฟังก์ชันความเป็นสมาชิกใช้เพื่อแสดงขอบเขตของค่าระดับความเป็นสมาชิกในแต่ละฟัซซี่เซ็ท ฟังก์ชันความเป็นสมาชิกของฟัซซี่เซ็ทมีด้วยกันหลายแบบและที่นิยมใช้กันได้แก่

- ฟังก์ชันความเป็นสมาชิกแบบสามเหลี่ยม (Triangular membership function): ฟัซซี่เซ็ทที่ใช้ฟังก์ชันความเป็นสมาชิกแบบนี้จะมีรูปร่างดังรูปที่ 2.3 ซึ่งค่าระดับความเป็นสมาชิก, μ_A ของฟังก์ชันจะนิยามได้ดังนี้:-

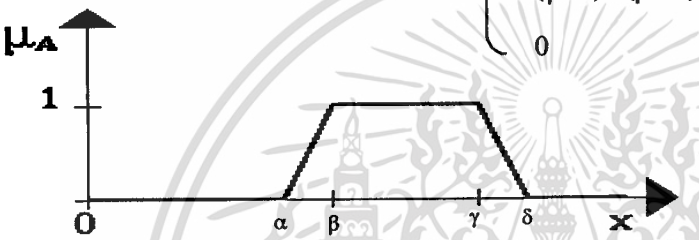
$$\mu_A(x) = \begin{cases} 0 & ; x < \alpha \\ (x-\alpha)/(\beta-\alpha) & ; \alpha \le x \le \beta \\ (\gamma-x)/(\beta-\alpha) & ; \beta \le x \le \gamma \\ 0 & ; x > \gamma \end{cases} \quad \text{.....(2.1)}$$



รูปที่ 2.3 ฟังก์ชันความเป็นสมาชิกแบบสามเหลี่ยม

- ฟังก์ชันความเป็นสมาชิกแบบสี่เหลี่ยมคางหมู (Trapezoidal membership function): ฟัซซี่เซ็ทที่ใช้ฟังก์ชันความเป็นสมาชิกแบบนี้จะมีรูปร่างดังรูปที่ 2.4 ซึ่งค่าระดับความเป็นสมาชิก, μ_A ของฟังก์ชันจะนิยามได้ดังนี้:-

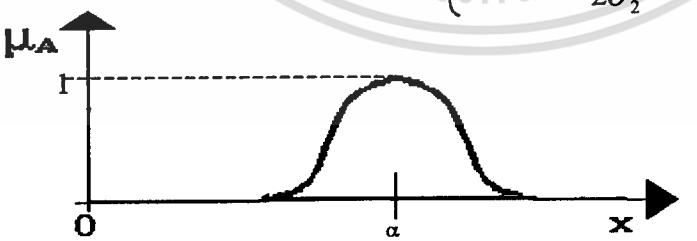
$$\mu_A(x) = \begin{cases} 0 & ; x < \alpha \\ (x-\alpha)/(\beta-\alpha) & ; \alpha \le x \le \beta \\ 1 & ; \beta \le x \le \gamma \\ (\gamma-x)/(\beta-\alpha) & ; \beta \le x \le \delta \\ 0 & ; x > \delta \end{cases} \quad \text{.....(2.2)}$$



รูปที่ 2.4 ฟังก์ชันความเป็นสมาชิกแบบสี่เหลี่ยมคางหมู

- ฟังก์ชันความเป็นสมาชิกแบบรูประฆังคว่ำ (Gaussian membership function): ฟัซซี่เซ็ทที่ใช้ฟังก์ชันความเป็นสมาชิกแบบนี้จะมีรูปร่างดังรูปที่ 2.5 ซึ่งค่าระดับความเป็นสมาชิก, μ_A ของฟังก์ชันจะนิยามได้ดังนี้:-

$$\mu_A(x) = \begin{cases} \exp\{-\dfrac{(x-c_1)^2}{2\sigma_1^2}\} & ; x \le \alpha \\ \exp\{-\dfrac{(x-c_2)^2}{2\sigma_2^2}\} & ; x > \alpha \end{cases} \quad \text{.....(2.3)}$$

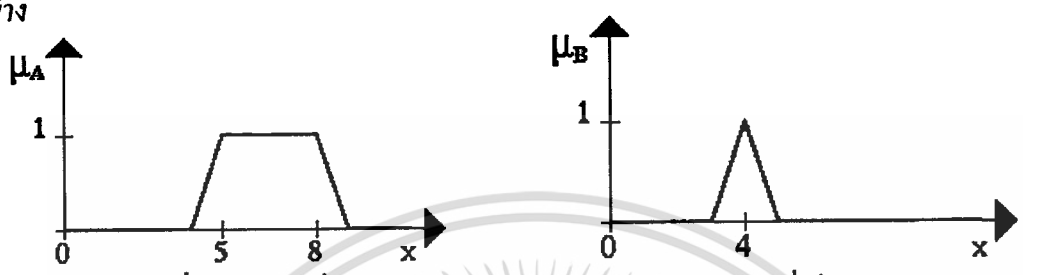


รูปที่ 2.5 ฟังก์ชันความเป็นสมาชิกแบบรูประฆังคว่ำ

2.1.3 การกระทำการกันของฟัซซี่เซ็ท (Operations of fuzzy sets)

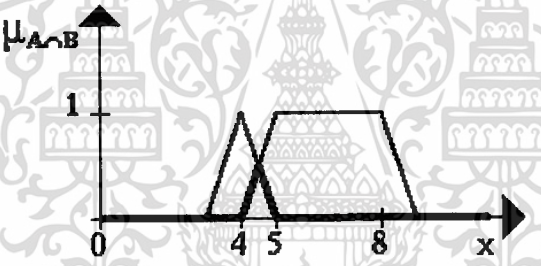
การกระทำการกันของฟัซซี่เซ็ทก็คล้ายกับการกระทำการกันของ Crisp Set ดังตัวอย่างด้านล่าง

ตัวอย่าง



รูปที่ 2.6 a) ฟัซซี่เซ็ท A b) ฟัซซี่เซ็ท B

- การ Intersection หรือ AND Operation จะได้ผลลัพธ์ตามรูปที่ 2.7 ที่เป็นฟัซซี่เซ็ทเส้นหนา

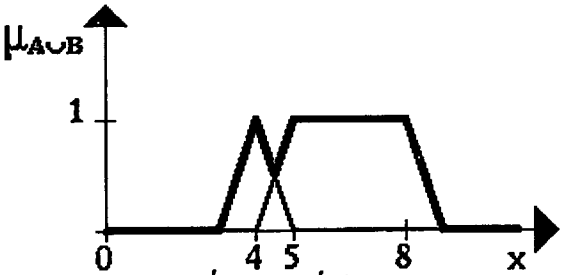


รูปที่ 2.7 ฟัซซี่เซ็ท A AND B

โดยทั่วไป การ Intersection ของฟัซซี่จะใช้ตัวกระทำการในการหาค่าต่ำสุดดังสมการ

$$A \cap B \leftrightarrow \mu_{A \cap B}(x) = \min\{\mu_A(x), \mu_B(x)\} \quad \text{.....(2.4)}$$

- การ Union หรือ OR Operation จะได้ผลลัพธ์ตามรูปที่ 2.8 ที่เป็นฟัซซี่เซ็ทเส้นหนา

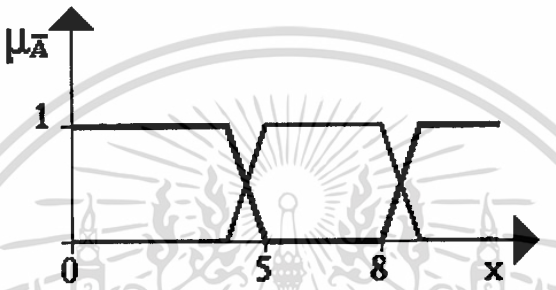


รูปที่ 2.8 ฟัซซี่เซ็ท A OR B

โดยทั่วไป การ Union ของฟัซซี่เซ็ทจะใช้ตัวกระทำการในการหาค่าต่ำสุดดังสมการ

$$A \cup B \leftrightarrow \mu_{A \cup B}(x) = \max\{\mu_A(x), \mu_B(x)\} \quad \text{.....(2.5)}$$

- Complement หรือ Negation จะได้ผลลัพธ์ตามรูปที่ 2.9 ที่เป็นฟัซซี่เซ็ทเส้นหนา



รูปที่ 2.9 NOT ฟัซซี่เซ็ท A (Negation of the fuzzy set A)

โดยทั่วไป การ Union ของฟัซซี่เซ็ทจะใช้ตัวกระทำการในการหาค่าต่ำสุดดังสมการ

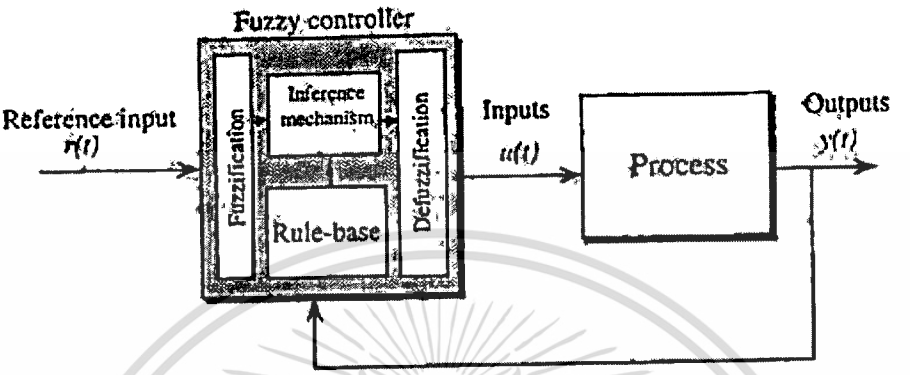
$$A' \leftrightarrow \mu_{A'}(x) = 1 - \mu_A(x) \quad \text{.....(2.6)}$$

2.2 การควบคุมแบบฟัซซี่ (Fuzzy Control)

การควบคุมแบบฟัซซี่นั้นจะมีหลักการต่างจากการควบคุมตามวิธีการแบบดั้งเดิม (Conventional Control) พอสมควร ซึ่งวิธีการควบคุมแบบดั้งเดิมนั้นโดยทั่วไปจะต้องทำการหาแบบจำลองทางคณิตศาสตร์ (Mathemetical Model) ขององค์ประกอบส่วนต่างๆ ของระบบกระบวนการ (Process System) แล้วนำมาทำการสร้างระบบควบคุม (Control System) ออกมาในรูปของสมการทางอนุพันธ์ แต่สำหรับวิธีการควบคุมแบบฟัซซี่นั้นจะอาศัยประสบการณ์ความชำนาญของผู้วิเคราะห์ในการที่จะถ่ายทอดออกมาเป็นค่าตัวแปรทางความคิดความรู้สึก (Linguistic Variable) ซึ่งสามารถอธิบายค่าปริมาณของตัวแปรพวกนี้ได้จากฟัซซี่เซ็ท (Fuzzy Set) โดยจะอธิบายตามตัวอย่างซึ่งเป็นการนำวิธีการควบคุมแบบฟัซซี่ไปใช้สำหรับการควบคุมอินเวอร์เท็ดเพนดูลัมในเนื้อหาบทถัดไป

2.2.1 องค์ประกอบหลักของตัวควบคุมแบบฟัซซี่

ตัวควบคุมแบบฟัซซี่ (Fuzzy Controller) จะมีองค์ประกอบหลักอยู่ 4 อย่างดังรูปที่ 2.10



รูปที่ 2.10 บล็อกไดอะแกรมตัวควบคุมแบบฟัซซี่ โดย Reference Input, r(t) คือ ค่าที่เราต้องการตั้งไว้เพื่อทำการควบคุมกระบวนการให้ได้ค่านี้ (Setpoint)

โดยในคือตัวควบคุมแบบฟัซซี่จะมีส่วนประกอบหลักต่างๆ คือ

1. *กฎเงื่อนไขพื้นฐาน (Rule-Base)*:- เป็นกลุ่มของกฎเงื่อนไขถ้า-แล้ว (if-then rules) ซึ่งเป็นกฎที่ตั้งจากความรู้สึกนึกคิดทางตรรกของผู้ชำนาญระบบนั้นว่าถ้าเป็นเงื่อนไขนั้นควรจะทำอย่างไรเพื่อให้การควบคุมดี โดยกฎเงื่อนไขพื้นฐานนี้มักออกแบบมาให้อยู่ในรูปแบบดังนี้

   กฎข้อที่ i : ถ้า x_1 *เป็* $A_1^{(i)}$ *และ...และ* x_n *เป็* $A_n^{(i)}$ *แล้ว y เป็* $B^{(i)}$ .....(2.7).

2. *ส่วนที่ทำการแปลงให้เป็นค่าทางฟัซซี่ (Fuzzification interface)*:- เป็นส่วนที่แปลงค่าอินพุทของตัวควบคุมหรือตัวแปรกระบวนการ (Controller inputs or Process Variable) ที่เป็นตัวเลขให้มาเป็นข้อมูลค่าทางฟัซซี่ซึ่งให้*กลไกการวิเคราะห์*สามารถใช้*กฎเงื่อนไขพื้นฐาน*ในการทำการวิเคราะห์ค่าพวกนี้ได้ โดยการแปลงให้เป็นค่าทางฟัซซี่นั้นก็จะเป็นการนำค่าอินพุทที่เป็นค่าตัวเลขปกติ, x ไปเป็นอินพุทของฟังก์ชันความเป็นสมาชิกที่ใช้

3. *กลไกการวิเคราะห์ทางฟัซซี่ (Fuzzy inference mechanism)*:- เป็นส่วนที่จะทำการคัดเลือก*กฎเงื่อนไขพื้นฐาน*เพื่อนำอินพุทที่เป็นค่าทางฟัซซี่มาทำการวิเคราะห์ออกมาเป็นเอาท์พุทค่าทางฟัซซี่ โดยกลไกการวิเคราะห์ก็มีใช้กันอยู่หลายวิธี ได้แก่

- การใช้ตัวดำเนินการแบบหาค่าน้อยที่สุด (Min-Operation of fuzzy implication)

$$\mu_{A\rightarrow B}(x, y) = \min\{\mu_A(x), \mu_B(y)\} \quad \text{.....(2.8)}$$

- การใช้ตัวดำเนินการแบบผลคูณ (Product-Operation of fuzzy implication)

$$\mu_{A\rightarrow B}(x, y) = \mu_A(x) \cdot \mu_B(y) \quad \text{.....(2.9)}$$

- การวิเคราะห์โดยใช้กฎเชิงเลขคณิต (Arithmetic Rule of fuzzy implication)

$$\mu_{A\rightarrow B}(x, y) = \min\{1, 1 - \mu_A(x) + \mu_B(y)\} \quad \text{.....(2.10)}$$

- การวิเคราะห์โดยกฎค่าสูงสุดของค่าต่ำสุด (Maxmin Rule of fuzzy implication)

$$\mu_{A\rightarrow B}(x, y) = \max\{\min\{\mu_A(x), \mu_B(y)\}, 1 - \mu_A(x)\} \quad \text{.....(2.11)}$$

- การวิเคราะห์โดยกฎบูลีน (Boolean Rule of fuzzy implication)

$$\mu_{A\rightarrow B}(x, y) = \max\{1 - \mu_A(x), \mu_B(y)\} \quad \text{.....(2.12)}$$

- การวิเคราะห์โดยกฎของ Goguen (Goguen's Rule of fuzzy implication)

$$\mu_{A\rightarrow B}(x, y) = \begin{cases} 1 & ; \mu_A(x) \le \mu_B(y) \\ \dfrac{\mu_B(y)}{\mu_A(x)} & ; \mu_A(x) > \mu_B(y) \end{cases} \quad \text{.....(2.13)}$$

4. *ส่วนที่ทำการแปลงออกมาเป็นค่าเพื่อส่งออกไปสั่งการควบคุม (Defuzzification interface)*:- จะทำการคำนวณแปลงค่าทางฟัซซี่จากการวิเคราะห์ของ*กลไกการวิเคราะห์*ออกมาเป็นค่าผลลัพธ์ที่เป็นตัวเลข และส่งออกไปสั่งการควบคุมอินพุทของกระบวนการ โดยมีอยู่ด้วยกันหลายวิธีได้แก่

- วิธีการหาค่ากลางของพื้นที่ (Center of area defuzzification) หรืออาจเรียกว่าการหาจุดศูนย์ถ่วง (center of gravity, COG) หรือ แบบเซนทรอยด์ (centroid)

$$output = \frac{\sum_i b_i \int \mu_{(i)}}{\sum_i \int \mu_{(i)}} \quad \text{.....(2.14)}$$

โดย b_i คือ ค่าที่จุดศูนย์กลางของฟัซซี่เซ็ทย่อยแต่ละรูป ในฟัซซี่เซ็ทรวม

$\int \mu_{(i)}$ คือ พื้นที่ใต้กราฟของฟังก์ชันความเป็นสมาชิกของฟัซซี่เซ็ทย่อยแต่ละรูป ในฟัซซี่เซ็ทรวม

ซึ่งการคำนวนพื้นที่ใต้กราฟของฟังก์ชันความเป็นสมาชิกของฟัซซี่เซ็ทฟังก์ชันความเป็นสมาชิกรูปกราฟสามเหลี่ยมที่มีความเป็นสมาชิกสูงสุดเป็น 1 ซึ่งจะถูกตัดยอดออกโดยค่าระดับความเป็นสมาชิก จะมีค่าเป็น:-

$$w\left(h - \frac{h^2}{2}\right) \text{.....(2.15)}$$

โดย w คือ ค่าความกว้างของฐานของรูปกราฟสามเหลี่ยมของฟัซซี่เซ็ทฟังก์ชันความเป็นสมาชิก

h คือ ค่าความสูงของรูปกราฟสามเหลี่ยมที่ถูกตัดยอดออก ซึ่งก็คือค่าระดับความเป็นสมาชิกของฟัซซี่เซ็ทย่อยแต่ละรูปนั่นเอง

- วิธีการหาจุดศูนย์กลางเฉลี่ย (Center-Average) ซึ่งสามารถคำนวนได้ง่ายกว่าวิธีการ COG โดยสามารถคำนวณค่าออกมาได้จากสมการ:-

$$output = \frac{\sum_i b_i \mu_{premise\,(i)}}{\sum_i \mu_{premise\,(i)}} \quad \text{.....(2.16)}$$

โดย $\mu_{premise\,(i)}$ คือ ค่าระดับความเป็นสมาชิกของฟัซซี่เซ็ทผลลัพธ์ย่อยแต่ละรูป

วิธีการหาค่าเอาท์พุทผลลัพธ์ จากการหาจุดศูนย์กลางของรูปกราฟของฟัซซี่เซ็ทฟังก์ชันความเป็นสมาชิกผลลัพธ์รวม (Defuzzification) ทั้งสองวิธีการนี้ จะคำนวณค่าออกมาได้แตกต่างกัน ซึ่งการจะ

พิจารณาว่าวิธีการไหนจะดีกว่ากันในการนำไปใช้ควบคุมระบบกระบวนการของเรานั้นจะต้องขึ้นอยู่กับการทดสอบจำลองการทำงาน (simulation) และการนำไปใช้งาน (implemention) ต่อไป

บทที่ 3
หลักการตัวควบคุมแบบฟัซซี่สำหรับอินเวอร์เท็ดเพนดูลัม

สำหรับแนวทางการสร้างตัวควบคุมในเนื้อหาส่วนนี้จะอธิบายเป็นตัวอย่างอย่างง่ายที่ประยุกต์สำหรับการควบคุมเฉพาะมุมของเพนดูลัมซึ่งถือเป็นเอาท์พุทตัวหลักของระบบกระบวนการนี้ โดยถ้าต้องการควบคุมตำแหน่งของตัวฐานรถเลื่อนด้วยก็เพียงทำการเพิ่มฟัซซี่เซ็ทของอินพุทตำแหน่งฐานรถเลื่อนเข้ามาอีกและกำหนดช่วงใช้งานให้เหมาะสม

3.1 รูปแบบแนวคิดในการสร้างตัวควบคุมมุมของอินเวอร์เท็ดเพนดูลัม

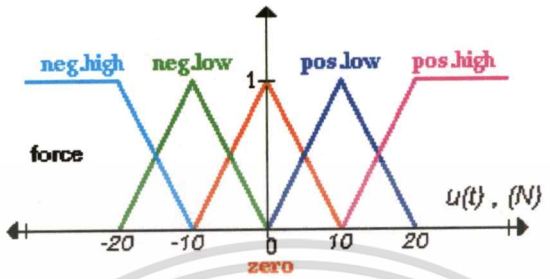
จะต้องหาว่าควรจะให้ตัวรถเลื่อนมีความเร็วมากหรือน้อยอย่างไรเพื่อส่งแรงไปควบคุมการผลักตัวรถเลื่อนให้สามารถควบคุมมุมของเพนดูลัมได้

3.1.1 กำหนดฟัซซี่เซ็ทของอินพุทและเอาท์พุท

ในที่นี้จะใช้ฟังก์ชันความเป็นสมาชิกแบบสามเหลี่ยม โดยแบ่งฟัซซี่เซ็ทออกเป็นช่วงๆ ดังนี้

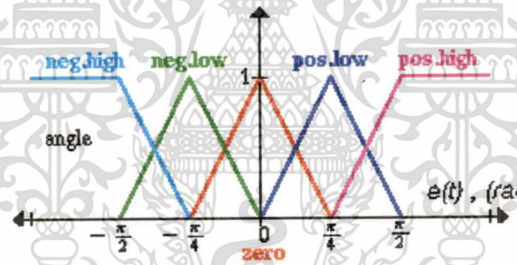
- ไปทางลบมาก:- negative high, NH (สีฟ้าอ่อน)
- ไปทางลบน้อย:- negative low, NL (สีเขียว)
- เป็นศูนย์:- zero, Z (สีแดง)
- ไปทางบวกน้อย:- positive low, PL (สีน้ำเงิน)
- ไปทางบวกมาก:- positive high, PH (สีชมพู)

กำหนดฟัซซี่เซ็ทของเอาท์พุทแรงจากมอเตอร์ที่ไปผลักตัวรถเลื่อน

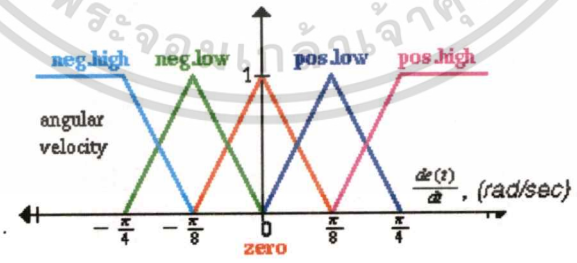


รูปที่ 3.1 ฟัซซี่เซ็ทของแรงจากมอเตอร์ที่ควบคุมการผลักตัวรถเลื่อนของอินเอวร์เท็ดเพนดูลัม

กำหนดฟัซซี่เซ็ทของอินพุทซึ่งเป็นตัวแปรจากกระบวนการในที่นี้คือค่ามุมและความเร็วเชิงมุมของเพนดูลัม



(a)



(b)

รูปที่ 3.2 (a) ฟัซซี่เซ็ทมุมของเพนดูลัม และ (b) ฟัซซี่เซ็ทความเร็วเชิงมุมของเพนดูลัม

3.1.2 สร้างกฎเงื่อนไขพื้นฐาน (Rule-Base)

จะมีเป็นรูปแบบ *ถ้า (if)* เหตุพื้นฐาน (premise)*แล้ว (then)* ผลลัพธ์สืบเนื่อง (Consequence)

จากแนวคิดที่จะใช้ควบคุมเพนดูลัม สามารถเขียนกฎเงื่อนไขได้ตามตัวอย่างซึ่งได้แก่

- *ถ้า (if)* มุมเป็นศูนย์ (Z) *และ (AND)* ความเร็วเชิงมุมเป็นศูนย์ (Z) *แล้ว (then)* แรงจากมอเตอร์ที่ควบคุมการผลักตัวรถเลื่อนจะเป็นศูนย์ (Z)
- *ถ้า (if)* มุมเป็นศูนย์ (Z) *และ (AND)* ความเร็วเชิงมุมเป็นบวกน้อย (PL) *แล้ว (then)* แรงจากมอเตอร์ที่ควบคุมการผลักตัวรถเลื่อนจะเป็นบวกน้อย (PL)

โดยกฎเงื่อนไขทุกข้อสามารถเขียนสรุปเป็นตารางความสัมพันธ์ที่ 1 ซึ่งแสดงให้เห็นถึงอินพุทกระบวนการสองตัวที่เข้ามาว่า *ถ้า* อินพุทแต่ละตัว(ซึ่งในที่นี้คือค่ามุมและความเร็วเชิงมุม)มีค่าเป็นสมาชิกของฟัซซี่เซ็ทใด *แล้ว* จะได้เอาท์พุท(คือแรงจากมอเตอร์ที่ไปผลักตัวรถเลื่อน)ออกมามีค่าเป็นสมาชิกของฟัซซี่เซ็ทใด

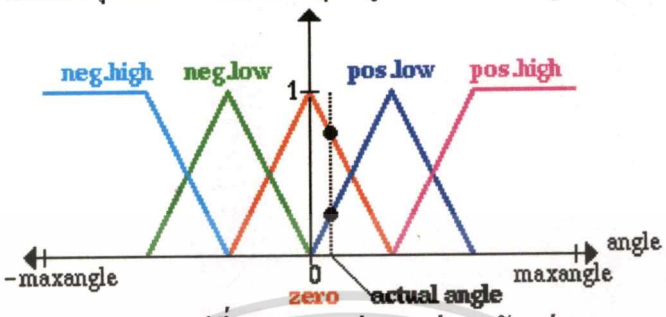
แรงผลักรถเลื่อน		มุม				
		NH	NL	Z	PL	PH
ความเร็วเชิงมุม	NH	NH	NH	NH	NL	Z
	NL	NH	NH	NL	Z	PL
	Z	NH	NL	Z	PL	PH
	PL	NL	Z	PL	PH	PH
	PH	Z	PL	PH	PH	PH

ตารางที่ 3.1 กฎเงื่อนไขพื้นฐาน แสดงเป็นตารางความสัมพันธ์ของอินพุทสองตัว (คือมุมและความเร็วเชิงมุมของเพนดูลัม)กับเอาท์พุท(แรงผลักรถ)

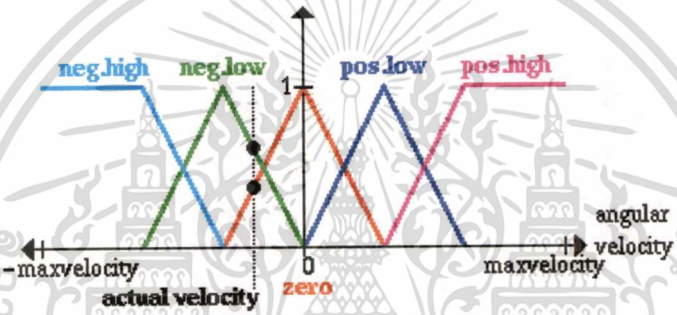
3.1.2.1 พิจารณาเมื่อมีอินพุทเข้ามายังตัวควบคุม

อินพุทซึ่งเป็นค่าตัวเลขนั้นก็จะถูกทำให้เป็นค่าทางฟัซซี่โดยพิจารณาที่ค่าความเป็นสมาชิกในฟัซซี่เซ็ทของอินพุทนั้น **(นำอินพุทที่เป็นค่าตัวเลขมาทำการ Fuzzicification)**

ตัวอย่างเช่น ถ้าค่าของมุมและความเร็วเชิงมุมอยู่ที่ตำแหน่งตามรูปที่ 3.3 และ 3.4 ตามลำดับ



รูปที่ 3.3 ตำแหน่งของค่ามุมตัวอย่าง



รูปที่ 3.4 ตำแหน่งของค่าความเร็วเชิงมุมตัวอย่าง

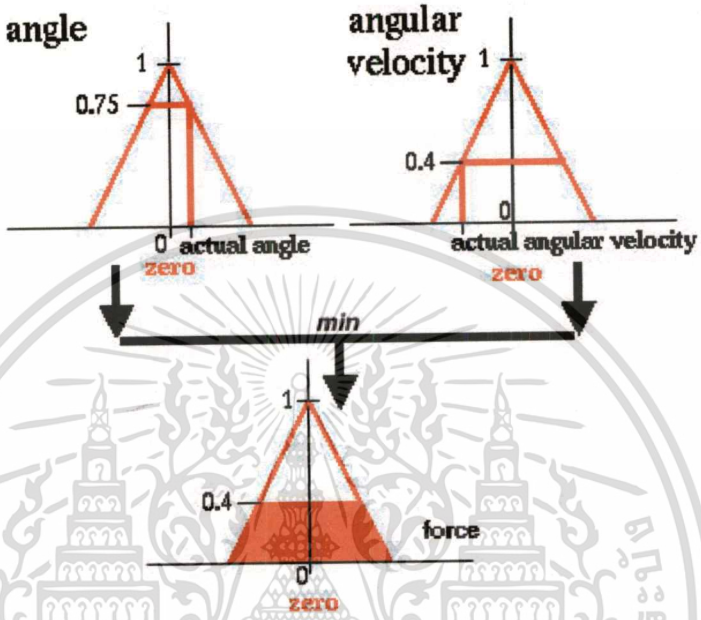
3.1.3 กลไกการวิเคราะห์ (Inference Mechanism) เพื่อหาค่าเอาท์พุทของตัวควบคุม

จากค่าอินพุทตัวอย่างข้างต้น จะเห็นได้ว่าอินพุทแต่ละตัวนั้นจะไปตกอยู่ในช่วงของฟัซซี่เซ็ท 2 ช่วง ดังนั้นอินพุทที่มี 2 ตัวจึงต้องนำกฎ (Rule-base) มาพิจารณาทั้งหมด 4 กรณี ได้แก่

- พิจารณาเลือกใช้กฎ:- *ถ้า*มุมเป็นศูนย์ (Z) *และ*ความเร็วเชิงมุมเป็นศูนย์ (Z) *แล้ว* แรงจากมอเตอร์ที่ควบคุมการผลักตัวรถเลื่อนจะเป็นศูนย์ (Z)

ในตัวอย่างนี้ จะทำการพิจารณาเฉพาะเซ็ทของช่วงที่เป็นศูนย์ก่อน จากรูปที่ 3.3 จะเห็นได้ว่าระดับความเป็นสมาชิกของเซ็ทที่มีค่าเป็นศูนย์ของมุมเพนดูลัมตัวอย่างค่านี้จะอยู่ที่ระดับ 0.75 และ จากรูปที่ 3.4 จะเห็นได้ว่าระดับความเป็นสมาชิกของเซ็ทที่มีค่าเป็นศูนย์ของความเร็วเชิงมุมเพนดูลัมตัวอย่างค่านี้จะเป็น 0.4 ตามกฎที่เลือกมานั้น กลไกการวิเคราะห์ในที่นี้จะทำการนำค่าระดับความเป็นสมาชิกทั้งสองนี้มาทำการ AND กันโดยคิดจากการหาค่าต่ำสุดของ 0.75 กับ 0.4 คือ min(0.75,0.4) = 0.4

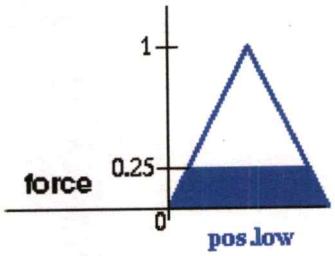
แล้วนำค่าที่ระดับนี้มาเป็นค่าของฟัซซี่เซ็ทแรงจากมอเตอร์ที่ควบคุมการผลักตัวรถเลื่อนในช่วงของเซ็ทค่าที่เป็นศูนย์ตามกฎเงื่อนไขที่เลือกใช้ข้างต้น ดังรูปที่ 3.5 ด้านล่างนี้



รูปที่ 3.5 หาค่าระดับความเป็นสมาชิกของฟัซซี่เซ็ทของความเร็วรถเลื่อนในช่วงค่าที่เป็นศูนย์ (Z)

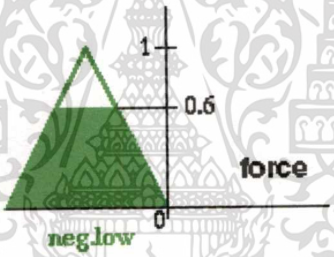
จากรูปที่ 3.3 และ 3.4 จะเห็นได้ว่าค่าของมุมและความเร็วเชิงมุมที่จุดนี้นั้นจะมีระดับความเป็นสมาชิกอยู่ในเซ็ทช่วงอื่นด้วย ซึ่งจะต้องทำการพิจารณาให้ครบทุกกรณีของกฎพื้นฐานที่ค่าอินพุทของกระบวนการสองตัวนี้เข้าเงื่อนไขโดยทำเช่นเดียวกับกรณีแรกดังต่อไปนี้

- พิจารณาเลือกใช้กฎ:- *ถ้า* มุมเป็นบวกน้อย (PL) *และ*ความเร็วเชิงมุมเป็นศูนย์ (Z) *แล้ว* แรงจากมอเตอร์ที่ควบคุมการผลักตัวรถเลื่อนจะเป็นบวกน้อย (PL)



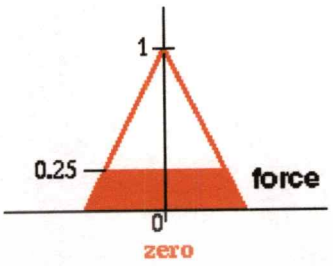
รูปที่ 3.6 ได้ระดับความเป็นสมาชิกของฟัซซี่เซ็ทของแรงจากมอเตอร์ที่ควบคุมการผลักตัวรถเลื่อนในช่วงค่าที่เป็นบวกน้อย (PL)

- พิจารณาเลือกใช้กฎ:- *ถ้า* มุมเป็นศูนย์ (Z) *และ*ความเร็วเชิงมุมเป็นลบน้อย (NL) *แล้ว* แรงจากมอเตอร์ที่ควบคุมการผลักตัวรถเลื่อนจะเป็นลบน้อย (NL)



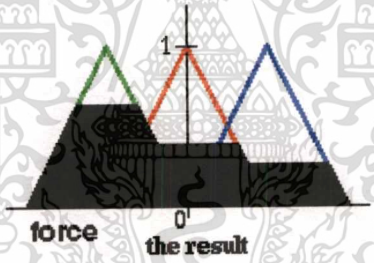
รูปที่ 3.7 ได้ระดับความเป็นสมาชิกของฟัซซี่เซ็ทของแรงจากมอเตอร์ที่ควบคุมการผลักตัวรถเลื่อนในช่วงค่าที่เป็นบวกน้อย (NL)

- พิจารณาเลือกใช้กฎ:- *ถ้า* มุมเป็นบวกน้อย (PL) *และ*ความเร็วเชิงมุมเป็นลบน้อย (NL) *แล้ว* แรงจากมอเตอร์ที่ควบคุมการผลักตัวรถเลื่อนจะเป็นศูนย์ (Z)



รูปที่ 3.8 ได้ระดับความเป็นสมาชิกของฟัซซี่เซ็ทของแรงจากมอเตอร์ที่ควบคุมการผลักตัวรถเลื่อนในช่วงค่าที่เป็นศูนย์ (Z) เป็น 0.25

จากนั้นนำ Fuzzy Set ผลลัพธ์จากทุกกรณีมา OR กันโดยใช้การ Max Operation จะได้ฟัซซี่เซ็ทผลลัพธ์รวมทุกกรณีที่เป็นแรงของรถเลื่อน ตามรูปที่ 3.9

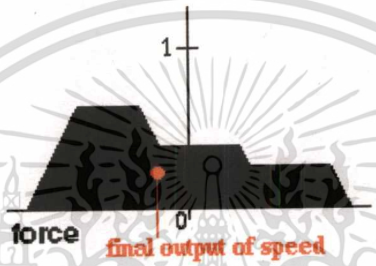


รูปที่ 3.9 ผลลัพธ์ฟัซซี่เซ็ทความเร็วรถเลื่อนจากทุกกรณีรวมกันโดยการ OR

วิธีการตามขั้นตอนที่ได้แสดงมาข้างต้น เป็นขั้นตอนในการ วิเคราะห์หาค่าระดับความเป็นสมาชิกที่เป็นไปได้ทั้งหมดของฟัซซี่เซ็ทของผลลัพธ์รวม (Inference Step) โดยวิธีการนี้มีลักษณะเป็นรูปแบบ Sum of Product ที่ใช้การ Operation หาค่าแบบ Maxmin วิธีการตามที่ได้กล่าวมาข้างต้นเป็นวิธีการหนึ่งที่ใช้กันโดยทั่วไปของขั้นตอน Inference Step ที่เรียกว่า วิธี **"Mamdani"**

3.1.4 หาค่าเอาท์พุท(ค่าแรงจากมอเตอร์ไปผลักรถที่ออกไปจะสั่งควบคุมเพนดูลัม) (นำค่าเอาท์พุทที่เป็นค่าทางฟัซซี่มาทำการ Defuzzification)

ตามรูปที่ 3.10 เป็นการหาค่าความเร็วรถเลื่อนในฟัซซี่เซ็ทผลลัพธ์ของความเร็วรถเลื่อนทุกกรณีรวมกัน โดยหาค่าที่จุดศูนย์กลางของรูป

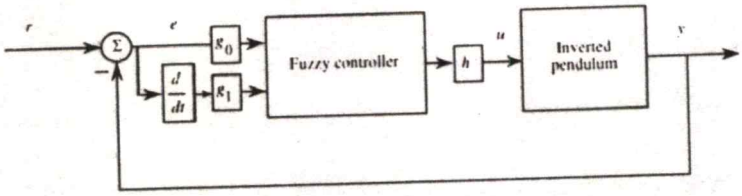


รูปที่ 3.10 ได้ผลลัพธ์ออกมาเป็นค่าของความเร็วรถเลื่อน

สำหรับการหาค่าจุดศูนย์กลางของรูปว่าตัวเลขที่จุดศูนย์กลางจะเป็นค่าเท่าไร (Defuzzication) สามารถทำได้หลายวิธีการ สำหรบวิธีการที่ใช้กันโดยทั่วไปมากที่สุดได้แก่ วิธีหาจุดศูนย์ถ่วงของพื้นที่ Center of Gravity, COG และ วิธีหาจุดศูนย์กลางเฉลี่ย Center Average ซึ่งได้อธิบายรายละเอียดไว้แล้วในเนื้อหาบทที่ 2

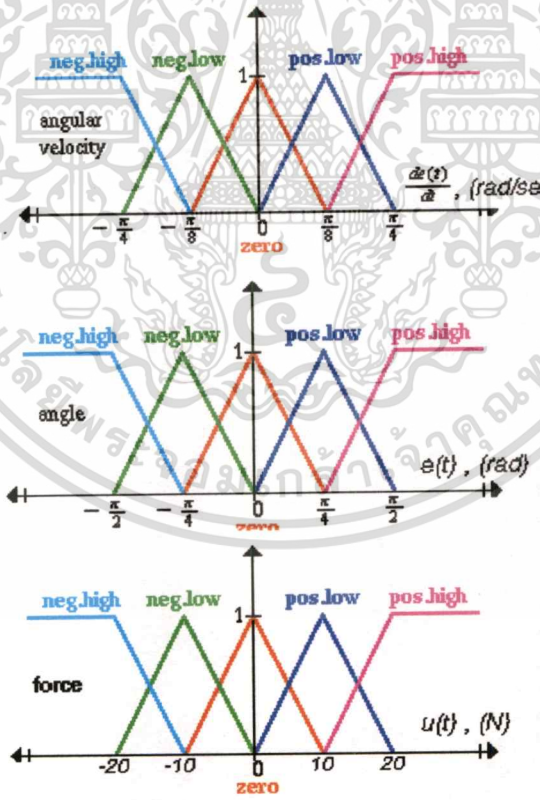
3.2 รูปแบบการปรับแต่ง (Tuning) ให้ได้การควบคุมที่ดี

ในการสร้างตัวควบคุมแบบฟัซซี่นั้นสิ่งสำคัญอันดับแรกคือการหาว่าค่าอินพุทแต่ละตัวที่เข้ามาแต่ละค่านั้นควรจะมีค่าอยู่ในช่วงฟัซซี่เซ็ทใด แล้วค่าเอาท์พุทที่คำนวนออกมาได้ควรจะมีค่าที่จะส่งออกไปสั่งควบคุมเป็นค่าเท่าไร ซึ่งในทางปฏิบัติแล้วเราไม่สามารถระบุออกมาเป็นค่าที่แน่นอนได้ทีเดียวในครั้งแรกว่าค่าแบบใดจะเป็นค่าที่ดีที่สุด ดังนั้นเราจึงต้องให้ระบบควบคุมสามารถทำการปรับแต่งค่าพวกนี้ได้ (Tuning) โดยการปรับแต่งค่าเหล่านี้เราสามารถทำการปรับแต่งได้โดยการปรับเลื่อนค่าไปมา (Offset) หรือ ทำการปรับขนาดช่วงของค่าโดยการเพิ่มพารามิเตอร์ตัวคูณในการปรับอัตราขยายของค่าอินพุท (Input Scaling Gain) และค่าเอาท์พุท (Output Scaling Gain) เข้าไปกับส่วนของตัวควบคุมดังบล็อกไดอะแกรมรูปที่ 3.11



รูปที่ 3.11 บล็อกไดอะแกรมตัวควบคุมอินเวอร์เท็ดเพนดูลัมแบบฟัซซี่ที่มีบล็อกตัวขยายปรับขนาดช่วง (Scaling Gain) คือ g_0, g_1 และ h

การปรับอัตราขยายของค่าอินพุทและเอาท์พุทให้กับตัวควบคุมนั้นเท่ากับเป็นการปรับความไวในการทำงานของตัวควบคุม โดยสามารถดูได้ตามตัวอย่างต่อไปนี้ โดยกำหนดค่าสเกลเริ่มต้นกับฟัซซี่เซ็ทของอินพุททั้งสองตัวและเอาท์พุทของตัวควบคุมไว้ดังรูปที่ 3.12

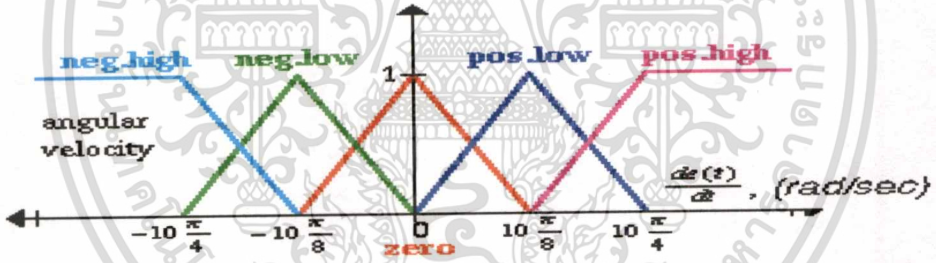


รูปที่ 3.12 ฟัซซี่เซ็ทของอินพุทและเอาท์พุทของตัวควบคุมที่ได้ใส่ค่าเป็นสเกลเริ่มต้นลงไป

3.2.1 การปรับแต่งค่าอัตราขยายทางด้านอินพุท (Input Scaling Gain)

การปรับค่าอัตราขยายทางด้านอินพุท g_I นั้นจะไปมีผลต่อค่าแต่ละช่วงของฟังก์ชันความเป็นสมาชิกของฟัซซี่เซ็ทของอินพุท โดยจะทำให้ขนาดค่าของสเกลทางอินพุทเปลี่ยนไปด้วยค่า $1/g_I$ เท่าจากเดิม ซึ่งสามารถพิจารณาได้ในลักษณะ ดังนี้

ถ้าให้อัตราขยายทางด้านอินพุท $g_I < 1$ ฟังก์ชันความเป็นสมาชิกของแต่ละฟัซซี่เซ็ทของอินพุทนั้นก็จะขยายออกกว้างขึ้นเนื่องจากค่าอินพุทที่มีค่าสูงจะถูกตัวคูณทำให้มีค่าน้อยลงกว่าเดิมก่อนที่จะเข้าไปแปลงเป็นค่าทางฟัซซี่ ซึ่งถ้ามองฟัซซี่เซ็ทของอินพุทเป็นภาพรวมมาจากอินพุทที่เข้ามาจากภายนอกโดยตรงก็จะหมายถึงว่าค่าสเกลแต่ละช่วงของฟัซซี่เซ็ทก็จะเป็นค่าที่มากขึ้น $1/g_I$ เท่า ดังตัวอย่างรูปที่ 3.13 ซึ่งได้จากการปรับอัตราขยายทางด้านอินพุทของความเร็วเชิงมุมให้ $g_I = 0.1$ ซึ่งจะทำให้ค่าสเกลของอินพุทขยายกว้างขึ้นออกมา 10 เท่า

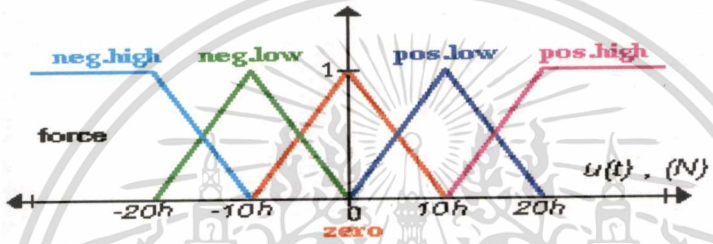


รูปที่ *3.13* ฟังก์ชันความเป็นสมาชิกของฟัซซี่เซ็ทของความเร็วเชิงมุมที่ได้ปรับเปลี่ยนสเกลเมื่อทำการปรับอัตราขยายทางด้านอินพุท เป็น $g_I = 0.1$

3.2.2 การปรับแต่งค่าอัตราขยายทางด้านเอาท์พุท (Output Scaling Gain)

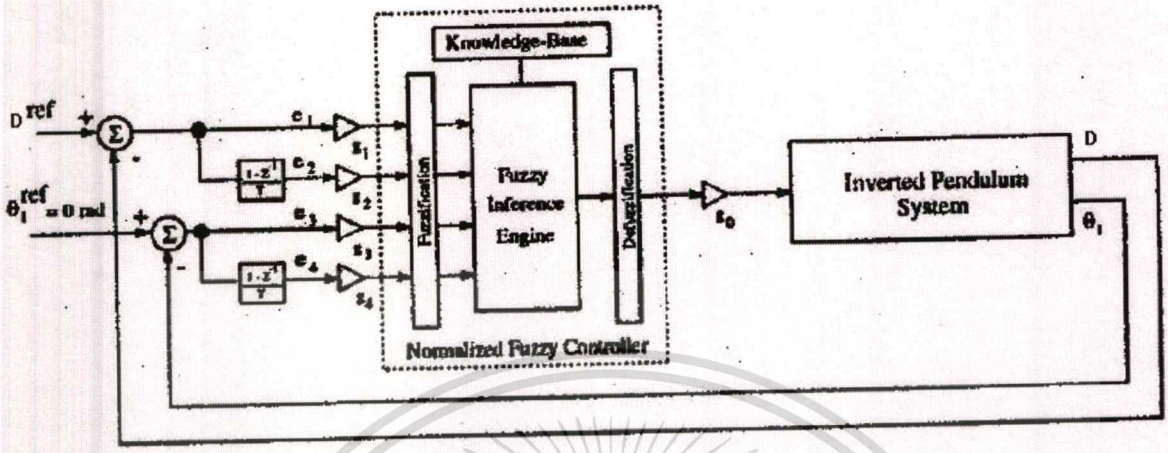
การปรับอัตราขยายทางด้านเอาท์พุทของตัวควบคุม h นั้น ก็สามารถพิจารณาได้คล้ายกันกับการพิจารณาการปรับอัตราขยายทางด้านอินพุท โดยจะไปมีผลต่อค่าแต่ละช่วงของฟังก์ชันความเป็นสมาชิกของฟัซซี่เซ็ทของเอาท์พุทให้ขนาดค่าของสเกลทางเอาท์พุทเปลี่ยนไปด้วยค่า h เท่าจากเดิม ซึ่งจะเป็นในทางกลับกันกับของทางด้านอินพุท ซึ่งจากตัวอย่างข้างต้นสามารถพิจารณาได้ในลักษณะดังนี้

- ถ้าให้อัตราขยายทางด้านเอาท์พุท $h > 1$ ฟังก์ชันความเป็นสมาชิกของแต่ละฟัซซี่เซ็ทของเอาท์พุทนั้นก็จะขยายออกกว้างขึ้น ซึ่งหมายถึงว่าค่าสเกลแต่ละช่วงของฟังก์ชันความเป็นสมาชิกก็จะเป็นค่าตัวเลขที่มากขึ้น h เท่า
- ในทางกลับกันถ้าให้อัตราขยายทางด้านเอาท์พุท $h < 1$ ฟังก์ชันความเป็นสมาชิกของแต่ละฟัซซี่เซ็ทของเอาท์พุทนั้นก็จะบีบแคบลง ซึ่งหมายถึงว่าค่าสเกลแต่ละช่วงของฟังก์ชันความเป็นสมาชิกก็จะเป็นค่าตัวเลขที่น้อยลง h เท่า



รูปที่ 3.14 ฟังก์ชันความเป็นสมาชิกของฟัซซี่เซ็ทของแรงที่ได้ปรับเปลี่ยนสเกลเมื่อทำการปรับอัตราขยายทางด้านเอาท์พุทเป็นค่า h เท่า

ในการปรับแต่งตัวควบคุมโดยการปรับค่าอัตราขยายของตัวควบคุมว่าควรจะตั้งไว้ที่ค่าเท่าใดนั้น ในทางปฏิบัติจะต้องมาทำการทดสอบกับระบบจริงแล้วค่อยๆ ทำการปรับแต่งดูแนวโน้มการตอบสนอง โดยพิจารณาในทุกกรณีให้คลุมช่วงที่จะใช้งานจริง และพิจารณาตามความเหมาะสมของแต่ละกระบวนการที่ทำการควบคุม

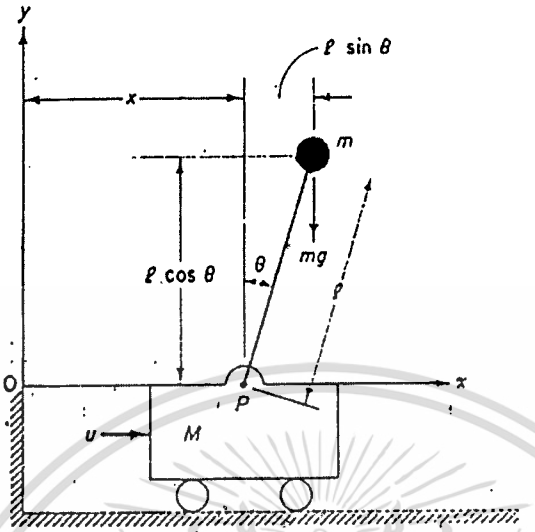


รูปที่ 3.15 บล็อกไดอะแกรมระบบควบคุมแบบฟัซซี่สำหรับอินเวอร์เท็ดเพนดูลัมที่ต้องการควบคุมทั้งมุมเพนดูลัมและระยะทางของรถเลื่อน

โดยระบบควบคุมแบบฟัซซี่สำหรับอินเวอร์เท็ดเพนดูลัมที่ต้องการควบคุมทั้งมุมเพนดูลัมและระยะทางของรถเลื่อนด้วยก็สามารถเขียนบล็อกไดอะแกรมดังรูปที่ 3.15 ซึ่งหลักการสร้างตัวควบคุมในส่วนอินพุทจากกระบวนการที่เป็นระยะทางวิ่งของรถเลื่อนก็สามารถสร้างได้โดยใช้หลักการเช่นเดียวกับที่ได้กล่าวมาแล้วสำหรับอินพุทจากกระบวนการที่เป็นมุมเพนดูลัม

3.3 การจำลองการทำงานของระบบควบคุมแบบฟัซซี่ (Simulation of Fuzzy Control System)

ในการสร้างการจำลองการทำงานของระบบควบคุมนั้นเป็นเรื่องเลี่ยงไม่ได้ที่เราจะต้องหาแบบจำลองทางคณิตศาสตร์ของกระบวนการของเราออกมาก่อนจึงจะสามารถนำมาจำลองการทำงานของกระบวนการได้ว่าอินพุทของกระบวนการเข้าไปเป็นอะไรแล้วจะให้เอาท์พุทตัวแปรกระบวนการออกมาเป็นอะไร สำหรับในกรณีของอินเวอร์เท็ดเพนดูลัมก็เช่นกันเราสามารถหาแบบจำลองทางคณิตศาสตร์มาได้จากสมการทางกลศาสตร์ตามรูปที่ 3.16



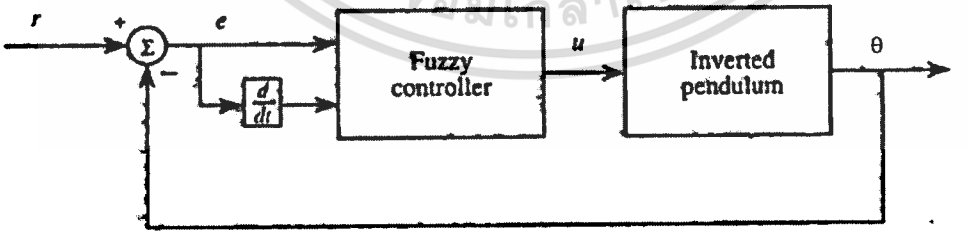
รูปที่ 3.16 Mechanical Free Body Diagram ของอินเวอร์เท็ดเพนดูลัมที่แสดงแรงในแนวแกนที่เกี่ยวข้องทั้งหมด

จากรูปที่ 3.16 เราสามารถหาสมการทางอนุพันธ์ของระบบทางกลระหว่างมุมเพนดูลัมและแรงผลักรถออกมาได้เป็นสมการอนุพันธ์อันดับสองดังสมการด้านล่าง

$$Ml\frac{d^2\theta}{dt^2} - (M+m)g\theta = -u$$

$$\ddot{\theta} = \frac{-u + (M+m)g\theta}{Ml} \qquad \text{.....(3.1)}$$

ซึ่งเมื่อพิจารณาเฉพาะการควบคุมมุมเพนดูลัมโดยวิเคราะห์จากบล็อกไดอะแกรมอย่างง่ายในรูปที่ 3.17



รูปที่ 3.17 บล็อกไดอะแกรมในรูปง่ายโดยพิจารณาเฉพาะตัวแปรกระบวนการที่เป็นมุมของเพนดูลัม

จากรูปที่ 3.17 จะพิจารณาว่าตัวควบคุม (Fuzzy Controller) แทนด้วย $f(e,\dot{e})$ ซึ่งทำให้ได้ค่า $u = f(e,\dot{e})$ โดยค่าอ้างอิงที่เราต้องการ $r = 0$, $e = r - \theta$ ซึ่งสมการทางอนุพันธ์ที่ไม่เป็นเชิงเส้นปกติจะสามารถนำมาเขียนในรูปแบบเวกเตอร์ของ nonlinear function ได้ดังสมการด้านล่าง

$$\dot{x}(t) = F(x(t), r(t)) \tag{3.2}$$

$$\theta = G(x(t), r(t))$$

โดย $x = [x_1, x_2, \ldots, x_n]^T$ เป็น state vector, $F = [F_1, F_2, \ldots, F_n]^T$ เป็นเวกเตอร์ของ nonlinear function, G เป็น nonlinear function ของแต่ละ state และค่าอินพุทอ้างอิง กับเอาท์พุทของกระบวนการ ซึ่งในกรณีนี้ state vector ของตัวแปรระบบทั้งหมดคือ

$$x = [x_1, x_2, x_3]^T = [\theta\ ,\ \dot{\theta}\ ,\ u]^T \tag{3.3}$$

จากสมการ (3.3) เราจะได้

$$\dot{x}_1 = x_2 = F_1(x, r)$$

และจากสมการ (3.1) และ (3.3) เราจะได้

$$\dot{x}_2 = \frac{-x_3 + (M - m)gx_1}{Ml} = F_2(x, r)$$

ซึ่งสมการเป็นรูปแบบสมการ nonlinear อย่างง่ายที่สามารถนำไปจำลองการทำงาน (simulation) ได้ โดยในการจำลองการทำงานนั้น เราสามารถใช้วิธีการ Euler's Method ในการประมาณตัวแปรอนุพันธ์ $\dot{x}$ ในรูปสมการ (2) ได้ดังสมการด้านล่าง

$$\frac{x(kh + h) - x(kh)}{h} = F(x(kh), r(kh), kh) \tag{3.4}$$

$$\theta = G(x(kh), r(kh), kh)$$

โดย h หมายถึงขนาดความห่างแต่ละสเตปในการคำนวน(เป็นการนำมาแทน dt) ซึ่งเปรียบเทียบได้จาก $x(t)$ โดย $t = kh$ (เป็นการประมาณของอนุพันธ์) โดยค่า h น้อยๆ ก็จะสามารถประมาณค่าออกมาได้ถูกต้องขึ้น

จากสมการ (4) สามารถเขียนได้เป็น

$$x(kh+h) = x(kh) + hF(x(kh), r(kh), kh)$$

$$\theta = G(x(kh), r(kh), kh)$$

โดยค่า $k = 0, 1, 2, \ldots$ ในการจำลองการทำงานก็จะทำการคิดวนไปแต่ละสเตปโดยคำนวน $x(h)$, $x(2h)$, $x(3h), \ldots$ ไปเรื่อยๆ โดยค่าเริ่มต้นของเวกเตอร์ $x(0)$ จะต้องทำการกำหนดหรือสมมุติขึ้นมาก่อน เพื่อนำมาหาการตอบสนองของระบบกับอินพุทอ้างอิงที่เป็นเซ็ทพอยท์ $r(kh)$

ตัวอย่างการเขียน Code สำหรับ การจำลองการทำงานของ Inverted Pendulum ที่เป็น Java Applet

```
//Pendulum moving with manual control simulator
//(moving only the pendulum base by moving the mouse left or right.

import java.applet.*;
import java.awt.*;

public class PendelMove extends Applet

   implements Runnable
{

   int xa,xe,ya,ye,xea,yea,xaa,xab,xeb,yeb,yg,breite,hohe,lange,kugel;
   double winkel,warten;
   boolean  richtung,start;
   Thread Pendel;
   Color bg;

    public void init()
    {
   bg=new Color(255,255,150);
   setBackground(bg);

   Dimension d = size();

   breite=d.width;
   hohe=d.height;

   lange=hohe*6/10;

   kugel=(int)lange/10;
   ya=hohe*8/10;
   yg=(kugel/2)+3+ya;

   Pendel=new Thread(this);
   Pendel.start();

    }


    public void start()
    {

   winkel=Math.PI/2;
   xa=breite/2;

   xaa=xa;
   xab=xa;

   xe=(int)(xa+lange*Math.cos(winkel));
   ye=(int)(ya-lange*Math.sin(winkel));

   xeb=xe;
   yeb=ye;
   xea=xe;
   yea=ye;
```

```
start=false;
warten=1;

if(Math.random()<0.5)
   richtung=true;
else
   richtung=false;

Pendel.resume();


 }

 public void stop()
 {
Pendel.suspend();
 }


 public void paint (Graphics g)
 {
update(g);

 }

 public void update (Graphics g)
 {


g.setColor(bg);
g.drawLine(xab,ya,xeb,yeb);
g.fillOval(xeb-(kugel/2),yeb-(kugel/2),kugel,kugel);

g.drawLine(xaa,ya,xea,yea);
g.fillOval(xea-(kugel/2),yea-(kugel/2),kugel,kugel);


g.setColor(Color.black);
g.drawLine(xa,ya,xe,ye);
g.fillOval(xe-(kugel/2),ye-(kugel/2),kugel,kugel);

if(!start)
{
g.setColor(Color.red);
g.fillRect(0,yg,breite-1,hohe-1);
}
 }

 public void run()
 {
    while(true)
    {
if (start)
    {

if(richtung==true)
{
  if(winkel<Math.PI)
    winkel+=0.01;
```

```
}
else
{
  if(winkel>0)
    winkel-=0.01;
}

warten=(Math.PI/2-Math.abs(winkel-Math.PI/2));
warten*=warten*12;
warten+=5;
xeb=xea;
yeb=yea;
xea=xe;
yea=ye;
xab=xaa;
xaa=xa;

xe=(int)(xa+lange*Math.cos(winkel));
ye=(int)(ya-lange*Math.sin(winkel));

if ((xe!=xea)||(ye!=yea))
  repaint();

}

try
{
  Thread.sleep((int)warten);
}
  catch(InterruptedException e)
{
  return;
}
}
 }

 public boolean mouseDown (Event e, int x, int y)
 {
start=!start;

return true;

 }

 public boolean mouseMove (Event e, int x, int y)
 {

if(start&&(y>yg))
{

if(xe<=x)
  richtung=true;
else
  richtung=false;

xeb=xea;
yeb=yea;
xea=xe;
yea=ye;
```

```
  xab=xaa;
  xaa=xa;

  if (Math.abs(xe-x)>=lange)
  {
    if(richtung==true)
      xa=xe+lange;

    else
      xa=xe-lange;
  }
  else
    xa=x;

  winkel=Math.acos((double)(xe-xa)/lange);
  repaint();
  }

  return true;
   }
}
```

บทที่ 4
การสร้างโปรแกรมตัวควบคุมแบบฟัซซี่สำหรับอินเวอร์เท็ดเพนดูลัม

4.1 รูปแบบแนวทางการสร้างตัวควบคุมแบบฟัซซี่ (Fuzzy Controller Implementation Format)

4.1.1 การกำหนดอาร์เรย์และสับรูทีนของตัวควบคุมแบบฟัซซี่ (Fuzzy Controller Arrays and Subroutines)

สำหรับแนวทางในการเขียนโปรแกรมเพื่อสร้างตัวควบควบคุมแบบฟัซซี่นั้น จะเริ่มพิจารณากันจากการสร้างอาร์เรย์และสับรูทีนของฟังก์ชันความเป็นสมาชิกของฟัซซี่เซ็ทของกฎเงื่อนไขพื้นฐานต่างๆ ซึ่งจากตัวอย่างฟังก์ชันความเป็นสมาชิกของฟัซซี่เซ็ทของอินพุทและเอาท์พุทแต่ละตัวจะมีฟัซซี่เซ็ทอยู่ทั้งหมด 5 ช่วง ซึ่งตัวอย่างต่อไปนี้จะแทนค่าลำดับที่ของฟัซซี่เซ็ทแต่ละตัวเรียงจาก NH, NL, Z, PL, PH ด้วย ค่าฟัซซี่เซ็ทที่ 0, 1, 2, 3, 4 ตามลำดับ เพื่อใช้เป็น index ในการอ้างถึงฟัซซี่เซ็ทแต่ละช่วง และให้ตัวแปร x1 แทน $e(t)$ หรือมุมเพนดูลัม และ x2 แทน $\frac{de(t)}{dt}$ หรือความเร็วเชิงมุมของเพนดูลัม ซึ่งเราจะกำหนดอาร์เรย์และฟังก์ชันต่างๆ ดังต่อไปนี้

- ให้ mf1[i](หรือ mf2[j]) แทนค่าของฟังก์ชันความเป็นสมาชิกที่เกี่ยวข้องกับอินพุทตัวที่ 1(หรือ 2) และค่าทางความคิดความรู้สึก (linguistic-numeric value) i(j) (แทน NH, NL, Z, PL, PH ด้วยค่า i(หรือ j) ที่ 0, 1, 2, 3, 4 ตามลำดับ) โดยในโปรแกรมนั้น mf1[i] ก็คือสับรูทีนในการคำนวนค่าความเป็นสมาชิกของฟังก์ชันความเป็นสมาชิกตัวที่ i^{th} ให้กับค่า numeric value ของอินพุทตัวแรก x1 (โดยสับรูทีนก็สามารถใช้สมการเส้นตรงแทนฟังก์ชันความเป็นสมาชิกแบบสามเหลี่ยมได้) แล้วก็ทำเช่นเดียวกันสำหรับ mf2[j]
- ให้ rule[i,j] แทนตัวชี้ผลลัพธ์สืบเนื่องของกฎเงื่อนไขที่มี linguistic-numeric value "i" เป็นพจน์แรกในเหตุพื้นฐาน (premise) แรก และ "j" เป็นพจน์สองในเหตุพื้นฐาน (premise) ที่สอง ดังนั้น rule[i,j] จึงเป็นลักษณะของ matrix ที่เป็นลักษณะตารางของกฎเงื่อนไขพื้นฐาน ซึ่งตามตัวอย่างนี้ก็จะได้ออกมาตามลักษณะดังนี้

$$rule[i,j] = \begin{bmatrix} 4 & 4 & 4 & 3 & 2 \\ 4 & 4 & 3 & 2 & 1 \\ 4 & 3 & 2 & 1 & 0 \\ 3 & 2 & 1 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 \end{bmatrix}$$

- ให้ prem[i,j] แทนเหตุพื้นฐาน (premise) ของกฎเงื่อนไขที่มี linguistic-numeric value "i" เป็นพจน์แรก เหตุพื้นฐาน (premise) แรก และ "j" เป็นพจน์สองในเหตุพื้นฐาน (premise) ที่สอง ของอินพุท x1 และ x2
- ให้ center[k] แทนค่าที่จุดกลางของฟังก์ชันความเป็นสมาชิกของเอาท์พุทตัวที่ k^{th} (ซึ่งในที่นี้เราให้ k = 0, 1, 2, 3, 4) โดยจุดกลางก็คือตำแหน่งจุดยอดของสามเหลี่ยมแต่ละรูป
- ให้ areaimp[k,h] แทนพื้นที่ใต้กราฟสามเหลี่ยมของฟังก์ชันความเป็นสมาชิกของเอาท์พุทตัวที่ k^{th} ซึ่งถูกตัดยอดออกไปเหลือความสูง h ซึ่งก็ areaimp[k,h] คือสับรูทีนที่ใช้ในการคำนวนพื้นที่ใต้กราฟของฟังก์ชันความเป็นสมาชิกที่ได้ผ่านกลไกการคิดมาแล้ว(implied fuzzy set)
- ให้ imps[i,j] แทนพื้นที่ใต้กราฟของฟังก์ชันความเป็นสมาชิกของฟัซซี่เซ็ทที่ผ่านกลไกการคิดมาแล้วของกฎพื้นฐานที่มี linguistic-numeric value "i" เป็นพจน์แรก เหตุพื้นฐาน (premise) แรก และ "j" เป็นพจน์สองในเหตุพื้นฐาน (premise) ที่สอง ของอินพุท x1 และ x2

4.1.2 แนวทางการ coding ในส่วนของตัวควบคุมแบบฟัซซี่ (Fuzzy Controller Pseudocode)

แนวทางนี้เป็นตัวอย่างสำหรับในกรณีที่ใช้ในการคำนวนเอาท์พุทของตัวควบคุมแบบฟัซซี่ โดยอินพุทตัวควบคุมอยู่ 2 ตัว

1. รับค่า x1 และ x2
   (รับค่าอินพุทเข้ามายังตัวควบคุม)
2. คำนวณ mf1[i] และ mf2[j] สำหรับทุก i, j
   (หาค่าของฟังก์ชันความเป็นสมาชิกทั้งหมดที่ให้ค่าต่างๆสำหรับ x1 และ x2)
3. คำนวณ prem[i,j] = min[mf1[i],mf2[j]] ของทุก i,j
   (หาค่าของฟังก์ชันความเป็นสมาชิกของเหตุพื้นฐาน(premise) ของ x1 และ x2 จากการหาค่าต่ำสุด)
4. คำนวณ imps[i,j] = areaimp[rule[i,j], prem[i,j]] ของทุก i,j
   (หาพื้นที่ใต้กราฟของฟังก์ชันความเป็นสมาชิกของฟัซซี่เซ็ทที่ผ่านกลไกการคิดมาแล้วทั้งหมดที่เป็นไปได้)
5. ให้ num = 0, den = 0
   (กำหนดค่าเริ่มต้นให้กับตัวเศษและส่วนของ COG)
6. For i=0 to 4, For j=0 to 4
   (สร้างลูปวนผ่านพื้นที่ทั้งหมดเพื่อหา COG)

num = num+imps[i,j]*center[rule[i,j]]

(คำนวนตัวเศษ (numerator) ของ COG)

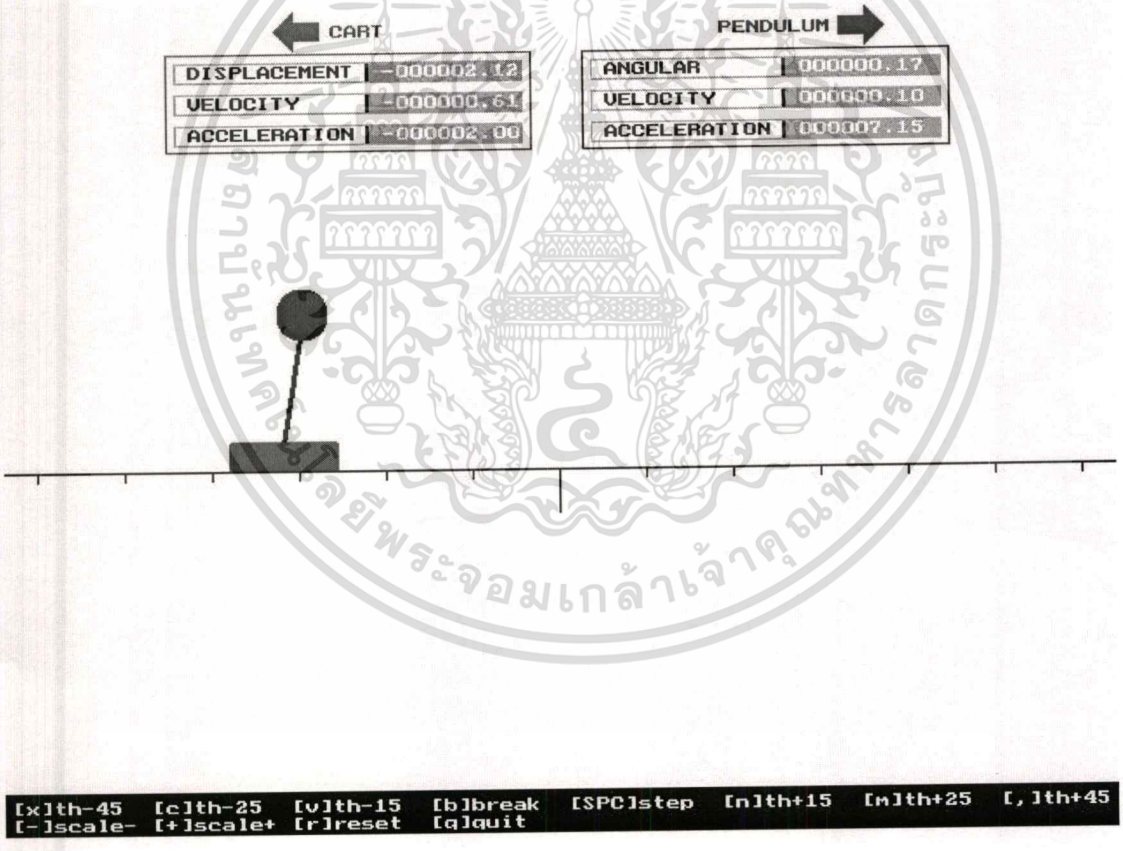
den = den+imps[i,j]

(คำนวนตัวส่วน (denominator) ของ COG)

7. Next i, Next j
8. เอาท์พุท ucrisp = num/den

   (ค่าเอาท์พุทที่ได้จากการคำนวนโดยตัวควบคุม)
9. วนกลับไปขั้นตอนที่ 1 เพื่อรับค่ามารอบใหม่

ซึ่งจะวนลูปเพื่อทำการเอาท์พุทออกไปควบคุมไปเรื่อยๆ จนกว่าจะสั่งหยุดการทำงานของโปรแกรม



รูปที่ 4.1 ภาพแสดงผลการทำงานของโปรแกรมจำลองการควบคุมอินเวอร์เท็ดเพนดูลัมแบบฟัซซี่

สำหรับรายละเอียดการใช้งานโปรแกรมนี้ สามารถดูได้ใน ภาคผนวก ก - วิธีการใช้งานโปรแกรมจำลองการควบคุมแบบฟัซซี่ของอินเวอร์เท็ดเพนดูลัม

4.2 ส่วนประกอบของโปรแกรมจำลองการควบคุมอินเวอร์เท็ดเพนดูลัมแบบฟัซซี่

ตัวโปรแกรมจะแบ่งเป็นส่วนประกอบต่างๆ ดังนี้

1. ตัวจำลองการทำงานของอินเวอร์เท็ดเพนดูลัม

   file name: invpend1.cpp

   ดูรายละเอียด Code ของโปรแกรมได้ในภาคผนวก ข-1

2. ตัวสร้างกฎพื้นฐานในการควบคุม(Rule-base)ของมุมpendulum

   file name: pndrule1.cpp

   ดูรายละเอียด Code ของโปรแกรมได้ในภาคผนวก ข-2

3. ตัวสร้างกฎพื้นฐานในการควบคุม(Rule-base)ของระยะทางตัวฐานรถเลื่อนcart

   file name: pndrule2.cpp

   ดูรายละเอียด Code ของโปรแกรมได้ในภาคผนวก ข-3

4. โปรแกรมหลักของระบบควบคุมแบบฟัซซี่สำหรับอินเวอร์เท็ดเพนดูลัม

   file name: pnd1sys.cpp

   ดูรายละเอียด Code ของโปรแกรมได้ในภาคผนวก ข-4

บทที่ 5
สรุปผลการทำงาน

5.1 สรุปผลการทำงานของโปรแกรม

โครงงานนี้เป็นการนำเสนอการสร้างระบบควบคุมแบบฟัซซี่สำหรับอินเวอร์เท็ดเพนดูลัม (Inverted Pendulum Fuzzy Control System) โดยทำการสร้างตัวจำลองการทำงานของอินเวอร์เท็ดเพนดูลัมและตัวควบคุมแบบฟัซซี่สำหรับอินเวอร์เท็ดเพนดูลัม โดยอินเวอร์เท็ดเพนดูลัมถือเป็นตัวอย่างหนึ่งที่นำมาใช้ในการศึกษาและเป็นต้นแบบในการออกแบบและทดสอบชุดควบคุมที่มีต่อการทำงานของระบบ เนื่องจากอินเวอร์เท็ดเพนดูลัมเป็นระบบที่ไม่เสถียรและจะอยู่ในสภาวะที่สมดุลไม่ได้เลยถ้าปราศจากการควบคุมที่ดี จุดประสงค์ของโครงงานนี้คือต้องการที่จะควบคุมเพนดูลัมให้ตั้งสมดุลอยู่ที่ตำแหน่งตรงกลางของทางวิ่ง นั่นคือต้องการควบคุมให้ได้ทั้งมุมของเพนดูลัมและตำแหน่งของตัวรถเลื่อน ตัวควบคุมที่ใช้สำหรับระบบนี้จะเป็นตัวควบคุมแบบฟัซซี่ ซึ่งจะมีข้อดีกว่าการใช้ตัวควบคุมแบบอื่นตรงที่สามารถมีได้หลายอินพุทและหลายเอาท์พุท และไม่ยุ่งยากที่จะต้องหาสมการแบบจำลองทางคณิตศาสตร์ของระบบออกมาก่อน ตัวควบคุมที่สร้างขึ้นมาเป็นซอฟท์แวร์ในลักษณะที่ติดต่อกับผู้ใช้งานแบบกราฟฟิกที่เป็นรูปภาพเคลื่อนไหวจำลองการทำงานจริงของอินเวอร์เท็ดเพนดูลัม และโดยให้ผู้ใช้ใส่ค่าอินพุทที่จะไปทำการรบกวนระบบและสังเกตผลการควบคุมทีละสเตป

สำหรับโปรแกรมจำลองการควบคุมที่เขียนขึ้นมาของโครงงานนี้ เขียนขึ้นด้วยภาษาซี โดยวิธีการใช้งานโปรแกรมได้อธิบายไว้ในภาคผนวก ก และ Source Code ของตัวโปรแกรมได้แสดงไว้ในภาคผนวก ข

5.2 ข้อเสนอแนะ

โปรแกรมที่ได้สร้างขึ้นมานี้สร้างขึ้นมาให้ใช้ทำการทดลองได้ในระดับหนึ่งซึ่งยังไม่สมบูรณ์นัก ในที่นี้ยังไม่ได้มีส่วนของการทดลองในด้านการปรับค่าอัตราขยายของระบบควบคุม โดยการทำงานของโปรแกรมนี้ยึดตามค่าเบื้องต้นที่ได้กำหนดไว้จากฟังก์ชันความเป็นสมาชิกของฟัซซี่เซ็ทแต่ละช่วงเท่านั้นยังไม่ได้ทำส่วนของตัวคุณค่าอัตราขยาย ควรจะได้มีการสร้างเพิ่มเติมในส่วนของการให้ผู้ใช้งานทดลองปรับค่าอัตราขยายได้ด้วย และตัวโปรแกรมนี้ยังไม่มีส่วนแสดงผลที่เป็นกราฟการตอบสนองของตัวระบบว่าจะใช้เวลาในการตอบสนองการควบคุมต่อค่าอัตราขยาย

ต่างๆ อย่างไร เพื่อนำมาศึกษาผลของการปรับค่าอัตราขยายค่าต่างๆ ของตัวควบคุมที่จะมามีผลต่อการควบคุมตัวระบบนี้

5.3 ประโยชน์ที่ได้รับ

โครงงานนี้ได้นำเสนออัลกอริทึมในการสร้างระบบควบคุมแบบฟัซซี่ (Fuzzy Control Algorithm) ซึ่งเนื้อหาทั้งหมดในโครงงานนี้ได้อธิบายถึงลำดับขั้นตอนและแนวความคิดในการสร้างระบบควบคุมแบบฟัซซี่อย่างง่ายๆ โดยผู้ที่จะประยุกต์ใช้วิธีการควบคุมแบบฟัซซี่ไปทำการสร้างระบบควบคุมสามารถกำหนดช่วงพารามิเตอร์ต่างๆ ไม่ว่าจะเป็นช่วงฟัซซี่เซ็ทแต่ละช่วงและฟังก์ชันความเป็นสมาชิกเองได้ตามความเหมาะสมกับระบบที่ต้องการนำไปควบคุมนั้นๆ หลักการสร้างโปรแกรมและตัวอย่างโปรแกรมที่อยู่ในโครงงานนี้สามารถนำไปใช้เป็นแนวทางอย่างง่ายๆ ในการเขียนโปรแกรมสร้างระบบควบคุมแบบฟัซซี่สำหรับระบบอื่นได้โดยต้องปรับเปลี่ยนพารามิเตอร์ต่างๆ ตามความเหมาะสม

บรรณานุกรม

นันทวัช อัชชารกูร, สรายุธ สินศักดิ์จรุงเดช และ วิชิต แซ่ล้อ. 2538. **ดิจิตอลอินเวอร์เท็ดเพนดูลัม.** ปริญญานิพนธ์. คณะวิศวกรรมศาสตร์. ภาควิชาระบบควบคุม. สถาบันเทคโนโลยีพระจอมเกล้าธนบุรี

Carnegie, Mellon. **Control Tutorials for Matlab.** http://rclsgi.eng.ohio-state.edu/matlab/. The University of Michigan.

Fuzzy Logic Laboratorium Linz-Hagenberg (FILL Homepage). www.fill-uni-linz.ac.at

Mathwork Inc. 1999. User's Guide version 2. **Fuzzy Logic Toolbox for use with MATLAB.**

Lee, C.C. 1990. IEEE Trans. Syst. vol. 20. no.2. Mar./Apr. 1990. Pg.404-435. **Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Part I/II.** Man Cybern.

Passino, Kevin M. and Stephen, Yurkovich. 1998. **Fuzzy Control.** Menlo Park. CA. :Addison-Wesley.

Woodyatt, A. and Middleton, R.H. 1997. **Constraints for single input and two output systems.** Technical Report. Callaghan. NSW 2308. University of Newcastle.

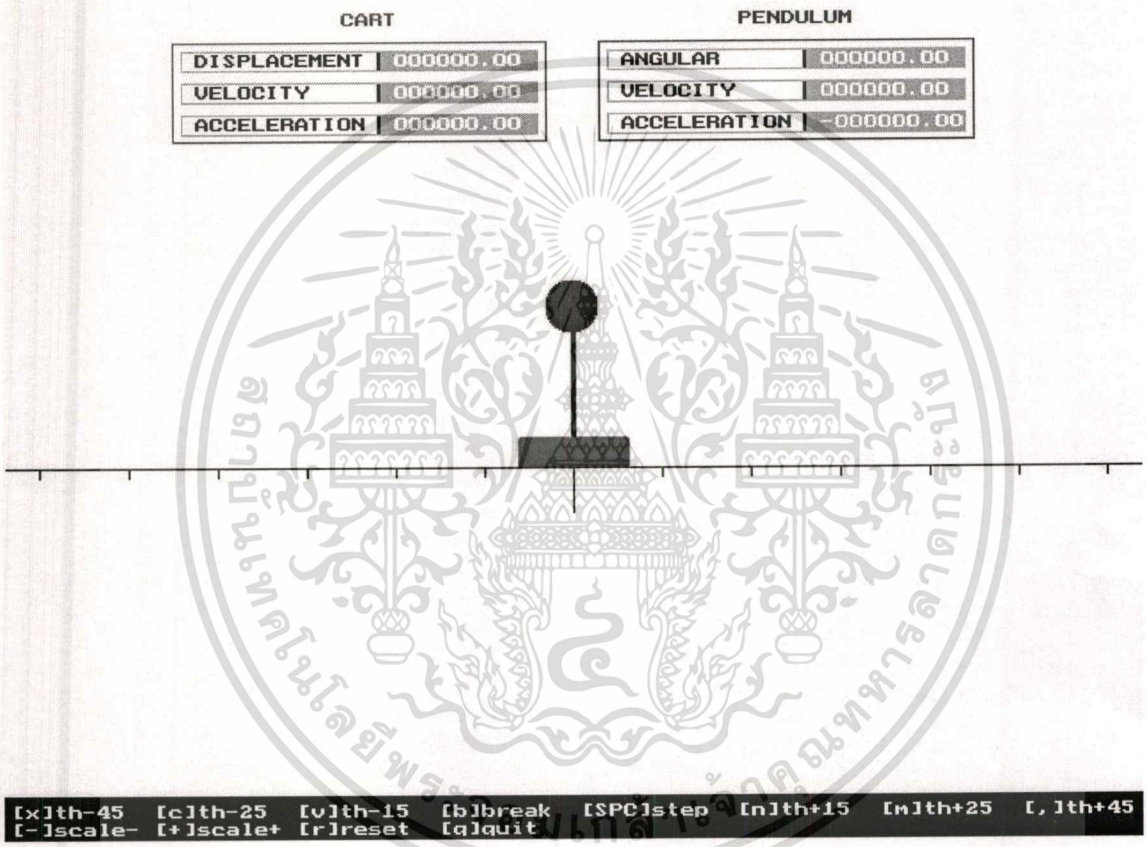
Woodyatt, A. and Freudenberg, J. 1999, **Fundamental Constraints for the Inverted Pendulum Problem.** Callaghan. NSW2308. Technical Report EE9716. University of Newcastle.

ภาคผนวก ก.

วิธีการใช้โปรแกรมตัวควบคุมแบบฟัซซี่สำหรับอินเวอร์เท็ดเพนดูลัม

ก.1 เปิดเรียกใช้โปรแกรม pend1sys.exe

จะขึ้นแสดงหน้าจอดังรูปที่ ก-1



รูปที่ ก-1 ภาพแสดงผลเริ่มต้นเมื่อเรียกใช้โปรแกรม pend1sys.exe

หน้าจอที่แสดงผลนั้นจะแสดงผลตัวเลขไว้ในกรอบสองช่อง

- กรอบช่องแรกคือ ตำแหน่งของตัวรถเลื่อน (CART) ซึ่งแสดงค่าระยะทางการวิ่ง (Displacement), ความเร็วรถเลื่อน (Velocity) และความเร่งรถเลื่อน (Acceleration)
- กรอบช่องที่สองคือ ตำแหน่งของเพนดูลัม (CART) ซึ่งแสดงค่ามุม (Angular), ความเร็วของเพนดูลัม (Velocity) และความเร่งของเพนดูลัม (Acceleration)

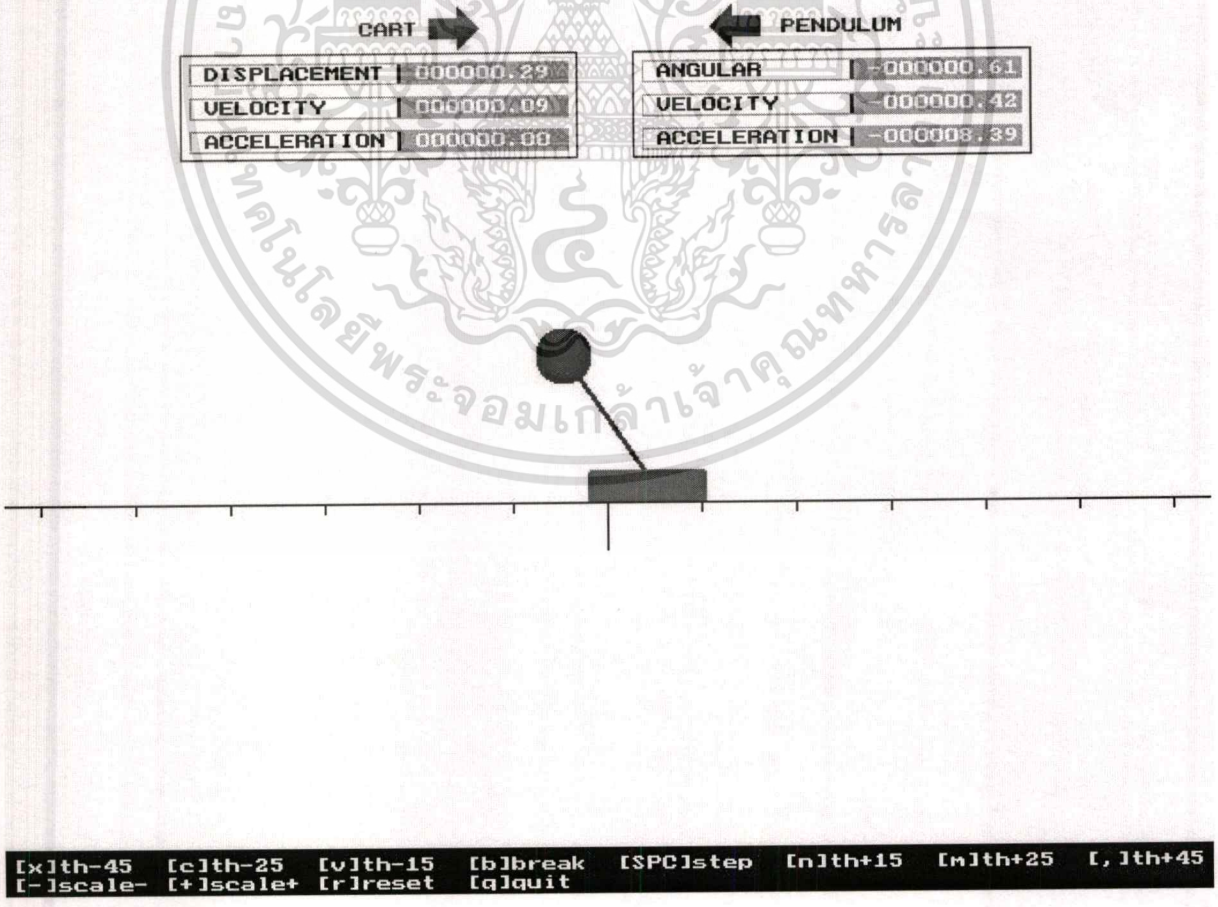
ก.2 การใส่แรงเพื่อสร้างการรบกวนระบบ (Disturbance) ของอินเวอร์เท็ดเพนดูลัม

- การกดปุ่ม [x] จะเป็นการใส่แรงไปกระทำที่ตัวฐานเลื่อนจากทางด้านซ้ายเป็นค่า 45
- การกดปุ่ม [c] จะเป็นการใส่แรงไปกระทำที่ตัวฐานเลื่อนจากทางด้านซ้ายเป็นค่า 25
- การกดปุ่ม [v] จะเป็นการใส่แรงไปกระทำที่ตัวฐานเลื่อนจากทางด้านซ้ายเป็นค่า 15
- การกดปุ่ม [n] จะเป็นการใส่แรงไปกระทำที่ตัวฐานเลื่อนจากทางด้านขวาเป็นค่า 15
- การกดปุ่ม [m] จะเป็นการใส่แรงไปกระทำที่ตัวฐานเลื่อนจากทางด้านขวาเป็นค่า 25
- การกดปุ่ม [,] จะเป็นการใส่แรงไปกระทำที่ตัวฐานเลื่อนจากทางด้านขวาเป็นค่า 45

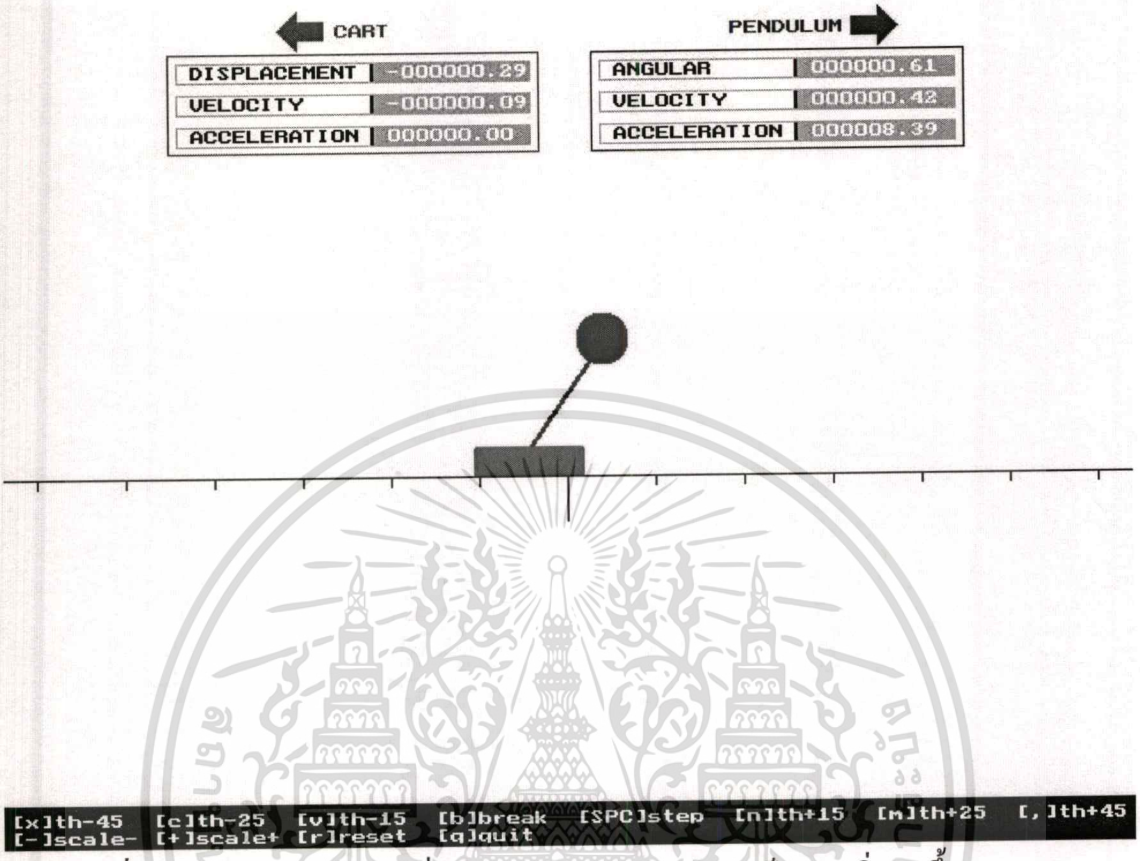
โดยดูตัวอย่างการแสดงผลได้ดังรูปที่ ก-2 และ ก-3

โดยในขณะที่ตัวกระบวนการเพนดูลัมกำลังเคลื่อนที่อยู่จะมีลูกศรแสดงการเคลื่อนที่ของตัวรถและตัวเพนดูลัมแสดงขึ้นมา

สามารถเพิ่มหรือลดค่ามุมของเพนดูลัมได้ตลอดเวลา โดยในขณะที่ระบบกำลังทำการควบคุมอยู่นั้น สามารถกดปุ่มเพื่อเพิ่มหรือลดค่ามุมเพื่อทำการกวนระบบได้



รูปที่ ก-2 ภาพแสดงผลจะภาพเริ่มต้น และมาทำการกดปุ่ม[x] เป็นการลดมุมลง 45 องศา



รูปที่ ก-3 ภาพแสดงผลจะภาพเริ่มต้น และมาทำการกดปุ่ม[,] เป็นการเพิ่มมุมขึ้น 45 องศา

ก.3 การสั่งให้ทำการควบคุม

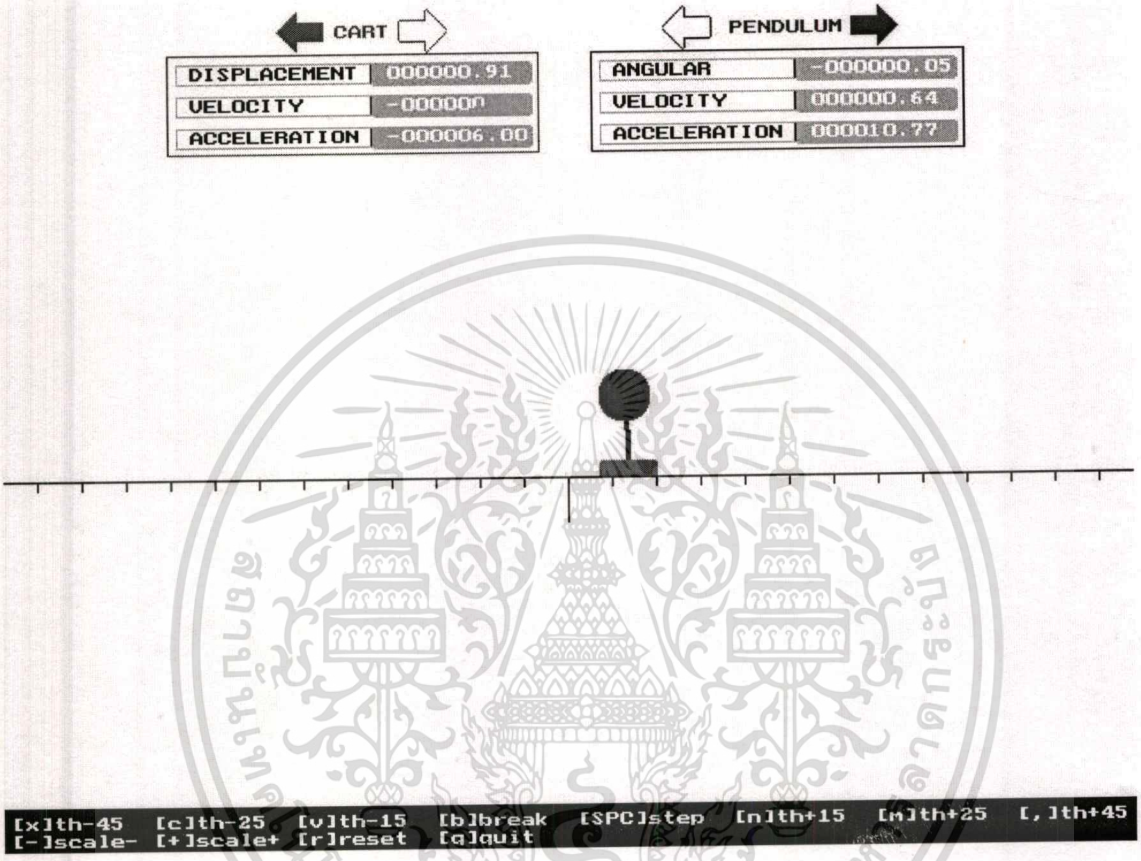
- การสั่งให้ทำการควบคุมแบบต่อเนื่อง โดยการกดปุ่ม [b] และสามารถสั่งให้หยุดทำการควบคุมโดยจะทำการหยุดสถานะของตัวเพนดูลัมไว้ที่จุดนั้น โดยการกดปุ่ม [b] อีกครั้ง
- การสั่งให้ทำการควบคุมเพื่อดูค่าของมุม และตำแหน่งของตัวรถทีละสเตป โดยการกดปุ่ม [spc] (ปุ่มspace bar)

ก.4 การสั่งเพิ่มและลดขนาดภาพในการแสดงผล

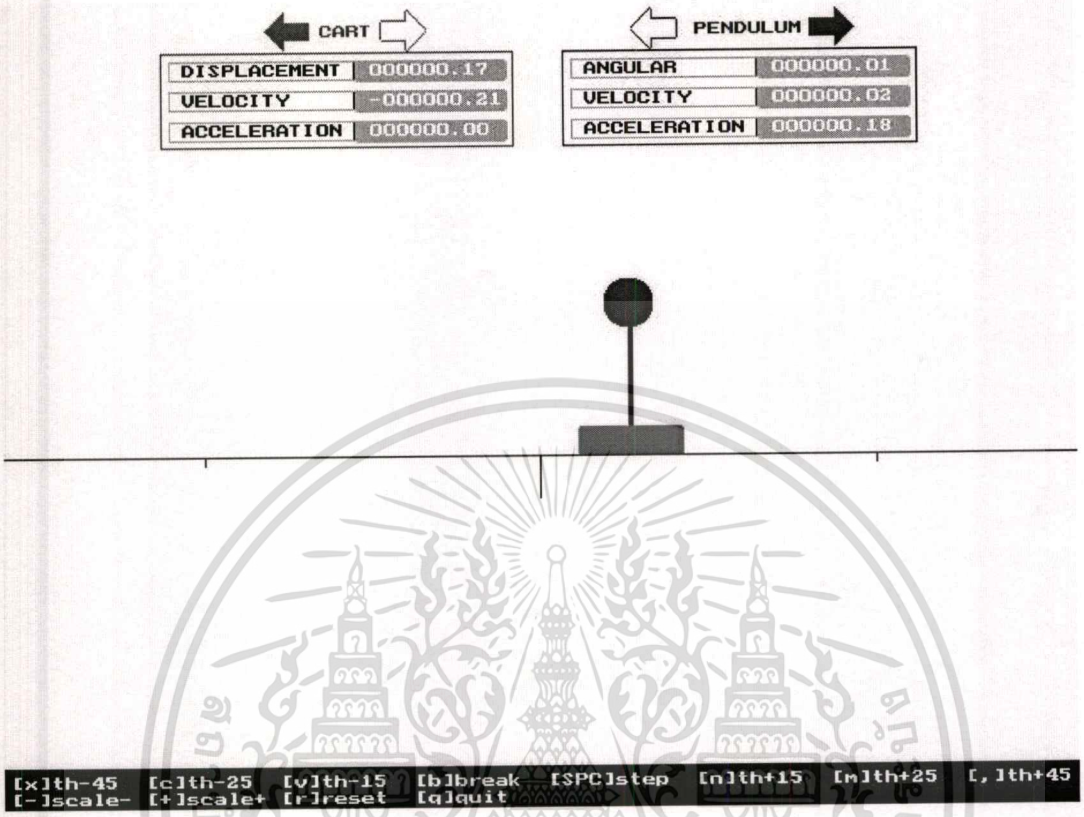
ในขณะที่ตัวเพนดูลัมกำลังเคลื่อนที่ในระหว่างที่ถูกควบคุมนั้น บางครั้งระยะทางการเคลื่อนที่ของตัวลดเลื่อนในขณะที่กำลังแสดงผลนั้นอาจไม่ยาวพอ

- สามารถลดขนาดสเกลต่อช่องของการแสดงผลได้โดยการกดปุ่ม [-] ซึ่งจะสามารถทำให้เห็นระยะทางการวิ่งที่อยู่ในหน้าจอได้ยาวขึ้น

- สามารถขยายขนาดสเกลต่อช่องของการแสดงผลได้โดยการกดปุ่ม [+] ซึ่งจะสามารถทำให้เห็นระยะทางการวิ่งที่อยู่ในหน้าจอในน้อยลงแต่เห็นภาพการวิ่งได้ละเอียดขึ้น



รูปที่ ก-4 ภาพแสดงผลเมื่อทำการลดขนาดสเกลต่อช่องโดยการกดปุ่ม [-]



รูปที่ ก-5 ภาพแสดงผลเมื่อทำการลดขนาดสเกลต่อช่องโดยการกดปุ่ม [+]

ภาคผนวก ข.

Source Code (ภาษา C) ของโปรแกรม

1. ตัวจำลองการทำงานของอินเวอร์เท็ดเพนดูลัม

file name: invpend1.cpp

```
/*********************************************************************
   Simulator of one-dimensional one-stage inverted pendulum
   File name: invpend1.cpp
   Date last modified: 12-2-2001

*********************************************************************/

#ifdef __TURBOC__
#include <graphics.h>
#else
#include <graph.h>
#endif
#include <math.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>

#define  PL    1.0
#define  LSCALE 150

#define  R1     15
#define  R2     10
#define  R3     5
#define  cCt    GREEN
#define  cPd    RED
#define  cBk    WHITE

typedef  struct {int x,y;} G;
typedef  struct {float x,y;} P;

int      npts1=4,npts3=7,npts4=14;
int      h,w,r,old_x1,old_y1,old_x2,old_y2,x_0,y_0,pux1,puy1,pux2,puy2;
int      r_id1,r_id0, manual,fuzzy;
float    ux,old_pu,dpu,uy;
float    px,py,pxi,pth,pu,pl,pl1,pl2,plen,plen1,pdxi,pdth,pcs,psn,g,pq;
float    st,st2,acc0,acc;
char     oStr[120];
G    cart_0[4]={{-60,0},{60,0},{60,30},{-60,30}}, cart[4],bomp[14];
G  arrow1[7]={{0,5},{-15,5},{-15,11},{-26,0},{-15,-11},{-15,-5},{0,-5}};
G  arrow2[7]={{0,5},{15,5},{15,11},{26,0},{15,-11},{15,-5},{0,-5}};
G  bomp0[14]={{-2,5},{0,10},{-12,10},{-15,7},{-30,7},{-33,4},{-36,2},{-36,-
2},{-33,-4},{-30,-7},{-15,-7},{-12,-10},{0,-10},{-2,-5}};
G        temp[20];
int      bomp_x,bomp_y,dist;
char     mem1[1000], mem2[600], mem3[600];
int      dispx,dispy;


/* Function Prototypes, added 12-10-1993 */
void clscrn(void);
void showMark(void);
void showPend(void);
void erasePend(void);
```

```
void scale(float s);
void showPend(void);
void eraseMark(void);
void angle(float a);
void head_line(void);


void Polygon(int npts, G *polyp, int c){
#ifdef __TURBOC__
   setcolor(c);
   setfillstyle(SOLID_FILL,c);
   fillpoly(npts,(int far *)polyp);
#else
   _setcolor(c);
   _polygon(_GFILLINTERIOR,(struct xycoord _far *)polyp,(short)npts);
#endif
}

void Moveto(int x,int y)
{
#ifdef __TURBOC__
   moveto(x,y);
#else
  _moveto(x,y);
#endif
}

void Lineto(int x,int y)
{
#ifdef __TURBOC__
   lineto(x,y);
#else
   _lineto(x,y);
#endif
}

void Setcolor(int c)
{
#ifdef __TURBOC__
   setcolor(c);
#else
   _setcolor(c);
#endif
}

void pr_put(int x, int y, int c)
{
   putpixel(x,y,c);
}

void pr_text(int x, int y, int c, char *s)
{
   setcolor(c);
   moveto(x,y);
   outtext(s);
}


void pr_rop(int x, int y, int w, int h,int c)
{
   G rect[4];
   rect[0].x=x;  rect[0].y=y+h;
   rect[1].x=x+w;  rect[1].y=y+h;
   rect[2].x=x+w;  rect[2].y=y;
   rect[3].x=x;  rect[3].y=y;
   Polygon(4, rect,c);
}

void pr_vector(int x1, int y1, int x2, int y2,int c,int l)
```

```
{
  if(1) setlinestyle(SOLID_LINE,1,THICK_WIDTH);
  else setlinestyle(SOLID_LINE,1,NORM_WIDTH);
  setcolor(c);
  moveto(x1,y1);
  lineto(x2,y2);
}

void pr_polygon_2(int xx,int yy,int npts,G *xy,int fc)
{
   int i;
   for(i=0;i<npts;i++) {
      temp[i].x=xx+xy[i].x;
      temp[i].y=yy-xy[i].y;
   }
   Polygon(npts,temp,fc);
}

void draw_box(int x1,int y1,int x2,int y2,int color)
{
   pr_vector(x1,y1,x1,y2,color,0); pr_vector(x1,y2,x2,y2,color,0);
   pr_vector(x2,y2,x2,y1,color,0); pr_vector(x2,y1,x1,y1,color,0);
}

void drawPara(int x1,int y1,char *str)
{
   int x2,y2;
   x2=x1+200; y2=y1+11;
   draw_box(x1,y1,x2,y2,7);
   pr_rop(x1+110,y1+1,90,9,BLUE);
   pr_text(x1,y1+2,BLUE,str);
}

void clscrn()
{
    pr_rop(0,0,639,479,WHITE);
}

void head_line()
{
    int x1,y1,x2,y2,hi;
    x1=w/2-150; y1=h/2-75; x2=w/2+150; y2=h/2+75;
    pr_rop(x1+5,y1+5,x2-x1,y2-y1,RED);
    pr_rop(x1,y1,x2-x1,y2-y1,BLUE);
    draw_box(x1+10,y1+10,x2-10,y2-10,WHITE);
    pr_text(x1+8,y1+35,YELLOW,"   INVERTED PENDULUM DEMONSTRATION");
  pr_text(x1+8,y1+45,YELLOW,"        FUZZY CONTROLLER");
    pr_text(x1+8,y1+65,YELLOW,"       Created by Nuntawat A");
  pr_text(x1+8,y1+75,YELLOW,"      Advised by Dr.Arit T");
  pr_text(x1+8,y1+85,YELLOW,"    FACULTY OF IT, KMITL, 2001");
    pr_text(x1+8,y1+110,RED,  "       Press any key to start");
    getch();
    clscrn();
    pr_rop(0,h-25,w,25,BLUE);
    pr_text(4,h-20,YELLOW,
    "[x]th-45  [c]th-25  [v]th-15  [b]break  [SPC]step  [n]th+15  [m]th+25
[,]th+45");
    pr_text(4,h-10,YELLOW,
    "[-]scale-  [+]scale+ [r]reset  [q]quit");
    x1=w/2-220; x2=w/2+20;
    hi=18; y1=35;
    drawPara(x1,y1," DISPLACEMENT "); drawPara(x2,y1," ANGULAR "); y1+=hi;
    drawPara(x1,y1," VELOCITY "); drawPara(x2,y1," VELOCITY "); y1+=hi;
    drawPara(x1,y1," ACCELERATION ");  drawPara(x2,y1," ACCELERATION ");
    draw_box(x1-5,30,w/2-15,86,GREEN);
    draw_box(x2-5,30,545,86,RED);
    pr_text(182,14,GREEN," CART");
    pr_text(406,14,RED," PENDULUM");
}
```

```
void ccprint(int x,int y,float v)
{
  pr_rop(x-6*8,y-1,88,10,7);
  sprintf(oStr,"%09.2f",v); oStr[11]=0;
  pr_text(x-5*8,y,WHITE,oStr);
}

void drawarrow(int x1,int y1,int dw,int color1,int color2)
{
    pr_polygon_2(x1-dw,y1,npts3,arrow1,color1);
    pr_polygon_2(x1+dw,y1,npts3,arrow2,color2);
}

void showdata(float acc1,float acc2)
{
   int y,x1,x2,dy;

    x1=w/2-60;    x2=w/2+180;
    y=37; dy=18;
    ccprint(x1,y,pu);     ccprint(x2,y,pth); y+=dy;
    ccprint(x1,y,dpu);    ccprint(x2,y,pdth); y+=dy;
    ccprint(x1,y,acc1);    ccprint(x2,y,acc2);  y+=dy;
    y=17;
    x1=205; x2=442;
    if(dpu>0) drawarrow(x1,y,22,cBk,cCt);
    else if(dpu<0) drawarrow(x1,y,22,cCt,cBk);else
drawarrow(x1,y,22,cBk,cBk);
    if(pdth>0) drawarrow(x2,y,40,cBk,cPd);
    else if(pdth<0) drawarrow(x2,y,40,cPd,cBk);else drawarrow
(x2,y,40,cBk,cBk);
}

void showMark()
{
    int xi,xm; float uux;
    uux=ux;
    if((uux*100.0)<2) uux=0.025;
    pr_vector(1,y_0,w-1,y_0,BLACK,0);
    for(xi=0;;xi+=100){
  xm=uux*(float)xi;
  if(xm>x_0)  break;
  pr_vector(x_0+xm,y_0+1,x_0+xm,y_0+5,BLACK,0);
  pr_vector(x_0-xm,y_0+1,x_0-xm,y_0+5,BLACK,0);
    }
    pr_vector(x_0,y_0+1,x_0,y_0+cart[2].y+10,BLUE,0);
}

void  eraseMark()
{
    pr_rop(0,y_0,w,10,WHITE);     /* clear pixrect    */
}

void showPend()
{
   pux1=(float)x_0+pu*ux*LSCALE;  puy1=y_0-cart[2].y;
   pux2=pux1+(int)px; puy2=puy1-(int)py;
   dispy=(int)((float)puy1-py*LSCALE);
   dispx=(int)((float)x_0+pu*ux*LSCALE+px*LSCALE);
   if((pux1>(w+200))||(pux1<-200)) return;
   putimage(dispx-r,dispy-r,mem1,COPY_PUT); /* pixrect  copy          */
   pr_vector(pux1,puy1,dispx,dispy,RED,1);
   pr_polygon_2(pux1,y_0-1,npts1,cart,GREEN);
   showMark();   old_x1=pux1; old_y1=puy1; old_x2=dispx; old_y2=dispy;
}

void erasePend()
{
   setcolor(WHITE);
```

```
    pr_rop(old_x2-r-1,old_y2-r-1,2*r+1,2*r+1,WHITE);
    pr_vector(old_x1,old_y1,old_x2,old_y2,WHITE,1);
    pr_polygon_2(old_x1,y_0-1,npts1,cart,WHITE);
}

void pstep(float q)
{
  float pddth;
  pq=q;
  pddth=-(pl1*psn*pcs*pdth*pdth+q*pcs-g*psn)/(pl-pl1*pcs*pcs);
  pth+=pdth*st+st2*pddth;
  while(pth>3.1415926) pth-=6.2831852; while(pth<-3.1415926) pth+=6.2831852;
  pdth+=st*pddth;
  pxi+=pdxi*st+st2*q;  pdxi+=st*q;
  pcs=(float)cos(pth); psn=(float)sin(pth);
  pu=pxi-pl1*psn;
  px=plen1*psn; py=plen*pcs;
  dpu=(pu-old_pu)/st;
  showdata(q,pddth);
  old_pu=pu;
}


void initPend(float l,float m,float M)
{
  int i;
  g=9.8;
  pl=l; pl1=m*l/(m+M); st2=st*st/2;
  px=0; pu=0; pth=0; pdth=0; pdxi=0; pxi=0;
  pl2=pl-pl1; pcs=1; psn=0;old_pu=0;
  r=R1;
  x_0=w/2; y_0=h/2+30;ux=0.5; uy=0.5; plen=1*ux; py=plen;plen1=plen;
  for(i=0;i<4;i++)
  {
    cart[i].x=(float)cart_0[i].x*ux;  cart[i].y=(float)cart_0[i].y*uy;
  }
  old_x1=x_0; old_y1=y_0+cart[2].y;
  old_x2=x_0; old_y2=(float)old_y1-(float)PL*ux*LSCALE;
  dist=100;
  showMark();
  showPend();
  pstep(0) ;
}


void scale(float s)
{   int i;
    eraseMark();   ux=ux*s;   uy=uy*s;  showMark();  erasePend();
    if((ux>=0.1)&&(ux<1.0))
    {
        for(i=0;i<4;i++)
        {
            cart[i].x=(float)cart_0[i].x*ux;
            cart[i].y=(float)cart_0[i].y*uy;
        }
   plen=(float)PL*ux; plen1=plen;
    }
    showPend();
}

void angle(float a)
{
    erasePend();
      pth+=a;
    while(pth>3.1415926) pth-=6.2831852; while(pth<-3.1415926)
pth+=6.2831852;

    showPend();
```

```
}

void drawbomp(int color)
{
 pr_polygon_2(bomp_x,bomp_y,npts4,bomp,color);
}


void sendbomp(int n)
{ int i,ssgn,ddist,sr,ppx,ppy;float ppc,pps;
  ssgn=-abs(n)/n;ddist=ssgn*dist;
  ppx=dispx;ppy=dispy;
  ppc=pcs;pps=psn;
  bomp_x=ddist*ppc+ppx; bomp_y=ppy+ddist*pps;
  for(i=0;i<14;i++)
  {
       bomp[i].x=ssgn*bomp0[i].x*ppc-bomp0[i].y*pps;
       bomp[i].y=-ssgn*bomp0[i].x*pps-bomp0[i].y*ppc;
  }
  drawbomp(BLACK);
  for(i=0;i<dist;i+=2)
  {
     drawbomp(cBk);bomp_x=ssgn*((dist-i))*ppc+ppx;
     bomp_y=ppy+ssgn*(dist-i)*pps;
     drawbomp(BLACK);
     sr=dist-i-abs(bomp0[6].x+4);if(sr<=r) break;
  }
  drawbomp(cBk);
}

change (char c)
{
  switch(c)
  {
    case 'x':case 'X':sendbomp(-1);angle(-0.6);return(4);
    case 'c':case 'C':sendbomp(-1);angle(-0.4);return(4);
    case 'v':case 'V':sendbomp(-1);angle(-0.2);return(4);
/*    case  0x20:angle(0);return(1);*/
    case 'n':case 'N':sendbomp(1);angle(0.2);return(4);
    case 'm':case 'M':sendbomp(1);angle(0.4);return(4);
    case ',':case '<':sendbomp(1);angle(0.6);return(4);
    case 'b':case 'B':return(2);
 /*   case 'f':return(3);*/
    case 'r':case 'R':erasePend();eraseMark();ux=0.5;uy=0.5;initPend((float)
PL,1,1);
       return(0);
    case '+':case '=': scale(2.0);return(1);
    case '-':case '_':scale(0.5);return(1);
    case 'q':case 'Q':return(5);
    default :return(0);
  }
}



void drawSph(int x,int y,int rid,int c,char *mem)
{
    int xsq,ysq,rsq;
    rsq=rid*rid;
    for(y=-rid;y<=rid;y++)
    { ysq = y*y;
      for(x=-rid;x<=rid;x++)
      {
        xsq=x*x;
        if((xsq+ysq)<=rsq)
      pr_put(rid+x,rid+y,c);
      }
```

```
    }
   getimage(x-rid,y-rid,rid*2-1,rid*2-1,mem);
}


int init_pdemo()
{
   int graphdriver = DETECT;
   int graphmode;
   int errorcode;

   initgraph(&graphdriver,&graphmode,"EGAVGA");
   errorcode=graphresult();
   if(errorcode!=grOk) printf("Graphics Error:%s\n",grapherrormsg
(errorcode));

   h=480;                          /* get width of pixrect          */
   w=640;                          /* get HEIGTH of pixrect         */
   clscrn();
   drawSph(30,30,R1,RED,mem1);
   drawSph(60,60,R2,RED,mem2);
   drawSph(100,100,R3,RED,mem3);
   clscrn();
   acc0=10; st=0.05;
   manual=1; fuzzy=1;
   head_line();
   initPend((float)PL,0.5,0.5);
   return(0);
}

void close_pdemo()
{
    closegraph();
}

int pdemo(float *f,float *x,float *dx,float *t,float *dt)
{
  int ret=1;
  float q;
  q=*f;
  if(kbhit()||manual)
  {
    switch(getch())
    {
    case 'x':case 'X':sendbomp(-1);angle(-0.6);
       pstep(0);erasePend();showPend();break;
    case 'c':case 'C':sendbomp(-1);angle(-0.4);
       pstep(0);erasePend();showPend();break;
    case 'v':case 'V':sendbomp(-1);angle(-0.2);
       pstep(0);erasePend();showPend();break;
/*    case  0x20:angle(0);break;*/
    case 'n':case 'N':sendbomp(1);angle(0.2);
       pstep(0);erasePend();showPend();break;
    case 'm':case 'M':sendbomp(1);angle(0.4);
       pstep(0);erasePend();showPend();break;
    case ',':case '<':sendbomp(1);angle(0.6);
       pstep(0);erasePend();showPend();break;
    case 'b':case 'B':manual=1-manual;break;
    case 'f':case 'F':fuzzy=1-fuzzy;break;
    case 'r':case 'R':erasePend();eraseMark();ux=0.5;uy=0.5;initPend((float)
PL,1,1);
       break;
    case '+':case '=': scale(2.0); break;
    case '-':case '_':scale(0.5); break;
    case 'q':case 'Q':ret=0;break;
    default :break;
    }
  }
  if(fuzzy) pstep(q);
```

```
  else pstep(0);
  erasePend();
  showPend();
  *x=pu; *dx=dpu; *t=pth; *dt=pdth;
  return(ret);
}
/*
main(){
   float a,b,c,d,e;
   a=0.0;
   init_pdemo();
   while(pdemo(&a,&b,&c,&d,&e)) ;
   close_pdemo();
}
*/

#if 0

//  First initialize the inverted pendulum:

//     init_pdemo();

/*  Then for an input value f (control force), the inverted pendulum will
move
  one step as a function of force : */

//     pdemo(&f, &x, &dx, &theat, &dtheta);

#endif
```

2. ตัวสร้างกฎพื้นฐานในการควบคุม(Rule-base)ของมุมpendulum

file name: pndrule1.cpp

```
/*  C Function performs fuzzy inference.
*   Original(Source) File Name pndrule1.cpp */

#ifndef _FIU_TYPES_IN_
#define _FIU_TYPES_IN_
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
typedef  float    float32;
typedef double    float64;
typedef char    int8;
typedef short int   int16;
typedef long    int32;
typedef unsigned char uint8;
typedef unsigned short int  uint16;
typedef unsigned long uint32;
#endif

/* Data structure for mbfs, edit only if you are sure */
#define  gradetype uint8 /* for truth values */
#define  valuetype uint8 /* for max i/o values */
#define  maxtype   uint8 /* max of gradetype and valuetype */
#define  temptype  float32 /* for temp values */
/* Maximal 32 bit unsigned long integer:*/
#define MAX_UNSIGNED_LONG 4294967295

#ifndef _FIU_MOTYPES_
#define _FIU_MOTYPES_
typedef  struct { void *m; int16 type, num;} MBF_OUT;
#endif
#ifndef _FIU_MLUT_A_
#define _FIU_MLUT_A_
typedef  gradetype  MBF_LUT_A;
#endif
#ifndef _FIU_MPCO_AA_
#define _FIU_MPCO_AA_
typedef struct { valuetype val; gradetype grd; } MBF_POLY_AA;
typedef struct { valuetype val; gradetype grd;
     valuetype cval; gradetype cgrd; } MBF_CUBI_AA;
typedef struct { int8 opt; void *opd;int16  n1, n2;
     maxtype opp; int16 ord; } MBF_OPR_AA;
#endif

#ifndef _FIU_CODEDEF_
#define _FIU_CODEDEF_
typedef struct{uint16 t; uint8 c;} CODE;
#endif
#define MBF_LUT  MBF_LUT_A
#define MBF_POLY MBF_POLY_AA
#define MBF_CUBI MBF_CUBI_AA
#define MBF_OPR  MBF_OPR_AA

/* Function prototypes (Mbf evaluation, defuzzification) */
#ifndef _FIU_P_MEVAL_AAO_
#define _FIU_P_MEVAL_AAO_
gradetype Mbf_Eval_Poly_AAO(MBF_POLY*, int16, valuetype);
gradetype Mbf_Eval_Cubi_AAO(MBF_CUBI*, int16, valuetype);
gradetype Mbf_Eval_Opr_AAO(MBF_OPR*, int16, valuetype);
gradetype Mbf_Eval_Lut_AAO(MBF_LUT*, int16, valuetype);
#endif
```

```
#ifndef _FIU_P_REPORT_ERROR
#define _FIU_P_REPORT_ERROR
void Report_Error(char *str);
#endif
#ifndef _FIU_P_REVAL_AAO_
#define _FIU_P_REVAL_AAO_
Rule_Eval_AAO(CODE *, MBF_POLY **, temptype *, gradetype *, gradetype *,
int16, int16, uint16);
#endif
#ifndef _FIU_P_OPER_AO_
#define _FIU_P_OPER_AO_
temptype Fl_Operation_AO(temptype, temptype, int16);
#endif

/* Constants */
#ifndef _FIU_CONSTS_IN_
#define _FIU_CONSTS_IN_
#define MF_LUT  0
#define MF_PLG  1
#define MF_OPR  2
#define MF_CUB  3
#define MF_SGT  4
#define MIN0
#define MAX1
#define PROD  2
#define PSUM  3
#define BINTER  4
#define BUNION  5
#define SHIFT 6
#define CUT7
#define CONC  8
#define NOT9
#define BCUT    10
#define WEIGHT  11
#define DM_CENTROID 0
#define DM_AMAX 1
#define DM_LMAX 2
#define DM_RMAX 3
#define DM_NODFZ 255
#endif

/* Macros */
#define Max_Grd 255
#define Min1(x,y)  ((x)<(y)?(x):(y))
#define Max1(x,y)  ((x)<(y)?(y):(x))
#define Prd1(x,y)  ((x)*(((temptype) y)/Max_Grd))
#define Sum1(x,y)  ((x)+(y)-(x)*(((temptype) y)/Max_Grd))
#define Bdi1(x,y)  (((x)+(y))>Max_Grd?((x)+(y)-Max_Grd):0)
#define Bdu1(x,y)  (((x)+(y))<Max_Grd?((x)+(y)):Max_Grd)

 /* define MBFs */
MBF_POLY pndrule1_mbf0[4]={{0,255}, {15,255}, {63,0}, {254,0}};
MBF_POLY pndrule1_mbf1[5]={{0,0}, {15,0}, {63,255}, {111,0}, {254,0}};
MBF_POLY pndrule1_mbf2[5]={{0,0}, {63,0}, {111,255}, {143,0}, {254,0}};
MBF_POLY pndrule1_mbf3[5]={{0,0}, {111,0}, {143,255}, {191,0}, {254,0}};
MBF_POLY pndrule1_mbf4[5]={{0,0}, {143,0}, {191,255}, {239,0}, {254,0}};
MBF_POLY pndrule1_mbf5[4]={{0,0}, {191,0}, {239,255}, {254,255}};
MBF_POLY pndrule1_mbf6[1]={{0, 255}};
MBF_POLY pndrule1_mbf7[1]={{62, 255}};
MBF_POLY pndrule1_mbf8[1]={{112, 255}};
MBF_POLY pndrule1_mbf9[1]={{142, 255}};
MBF_POLY pndrule1_mbf10[1]={{192, 255}};
MBF_POLY pndrule1_mbf11[1]={{254, 255}};
MBF_POLY pndrule1_mbf12[2]={{0,0}, {255,255}};
MBF_POLY * pndrule1_mbfs[6]={pndrule1_mbf6, pndrule1_mbf7, pndrule1_mbf8,
pndrule1_mbf9,
   pndrule1_mbf10, pndrule1_mbf11, };
CODE  pndrule1_code[79] ={{2,0},{6,96},
{1,0},{7,112},
```

```
{1,0},{6,112},
{0,0},{8,112},
{0,0},{7,112},
{0,0},{6,120},
{0,0},
{3,0},{6,96},
{2,0},{7,112},
{1,0},{8,112},
{0,0},{9,120},
{1,0},
{5,0},{6,96},
{4,0},{7,112},
{4,0},{6,112},
{3,0},{8,112},
{3,0},{7,112},
{2,0},{8,112},
{1,0},{9,112},
{0,0},{10,120},
{2,0},
{5,0},{7,96},
{4,0},{8,112},
{3,0},{9,112},
{2,0},{10,112},
{2,0},{9,112},
{1,0},{11,112},
{1,0},{10,112},
{0,0},{11,120},
{3,0},
{5,0},{8,96},
{4,0},{9,112},
{3,0},{10,112},
{2,0},{11,120},
{4,0},
{5,0},{11,96},
{5,0},{10,112},
{5,0},{9,112},
{4,0},{11,112},
{4,0},{10,112},
{3,0},{11,120},
{5,0},
{0,192},
};

/* Function body of the FIU module */
void exec_pndrule1(int8 input0, int8 input1,
  int8 *output0 )
 {
   /* Inside this function, the variables
      corresponds to these in the FIL file as follows: */
   /* input0 = pend_angle */
   /* input1 = pend_velocity */
   /* output0 = force */
   #define InVar     2
   #define OutVar    1
   #define InTv      12
   #define OutMaxTv  6
   int16 k;
   temptype dfz_fm, dfz_fz;
   temptype input[InVar], output[OutVar];
   gradetype tv_input[InTv];
   gradetype tv_output[1];

   /* Ranges for the variables */
   temptype input_range[2][3]={{-127.000000, 127.000000, 1.000000e+00},
  {-127.000000, 127.000000, 1.000000e+00}};
   temptype output_range[1][3]={{-127.000000, 127.000000, 1.000000e+00}};

  input[0]=input0;
  input[1]=input1;
```

```
  for (k=0;k<InVar;k++){
     dfz_fz=input[k];
     if(dfz_fz<input_range[k][0]) dfz_fz=input_range[k][0];
     if(dfz_fz>input_range[k][1]) dfz_fz=input_range[k][1];
     dfz_fm = (dfz_fz-input_range[k][0]) / input_range[k][2]+0.5;
     input[k] = dfz_fm;
    }

    tv_input[0] =Mbf_Eval_Poly_AAO(pndrule1_mbf0, 4, (valuetype)input[0]);
    tv_input[1] =Mbf_Eval_Poly_AAO(pndrule1_mbf1, 5, (valuetype)input[0]);
    tv_input[2] =Mbf_Eval_Poly_AAO(pndrule1_mbf2, 5, (valuetype)input[0]);
    tv_input[3] =Mbf_Eval_Poly_AAO(pndrule1_mbf3, 5, (valuetype)input[0]);
    tv_input[4] =Mbf_Eval_Poly_AAO(pndrule1_mbf4, 5, (valuetype)input[0]);
    tv_input[5] =Mbf_Eval_Poly_AAO(pndrule1_mbf5, 4, (valuetype)input[0]);
    tv_input[6] =Mbf_Eval_Poly_AAO(pndrule1_mbf0, 4, (valuetype)input[1]);
    tv_input[7] =Mbf_Eval_Poly_AAO(pndrule1_mbf1, 5, (valuetype)input[1]);
    tv_input[8] =Mbf_Eval_Poly_AAO(pndrule1_mbf2, 5, (valuetype)input[1]);
    tv_input[9] =Mbf_Eval_Poly_AAO(pndrule1_mbf3, 5, (valuetype)input[1]);
    tv_input[10] =Mbf_Eval_Poly_AAO(pndrule1_mbf4, 5, (valuetype)input[1]);
    tv_input[11] =Mbf_Eval_Poly_AAO(pndrule1_mbf5, 4, (valuetype)input[1]);
   Rule_Eval_AAO(pndrule1_code, pndrule1_mbfs, output, tv_output, tv_input,
0, 1, 6);

  /* Denormalize output values  */
      *output0=output[0] *output_range[0][2]+output_range[0][0];
   #undef InVar
   #undef OutVar
   #undef InTv
   #undef OutMaxTv
}

#ifndef _FIU_MEVAL_AAO_
#define _FIU_MEVAL_AAO_
/* Lookup table MBF evaluation subroutine */
gradetype Mbf_Eval_Lut_AAO(MBF_LUT *mbfp, int16 num, valuetype x)
{
   MBF_LUT *p;
   p=mbfp;
   if (x<=p[num-2]) return p[0];
   if (x>=p[num-1]) return p[num-3];
   return p[(int16)(x-p[num-2])];
 }

/* Polygon MBF evaluation subroutine */
gradetype Mbf_Eval_Poly_AAO(MBF_POLY *mbfp, int16 num, valuetype x)
{
   MBF_POLY  *p;
   int16  i;
   temptype  a, b;
   gradetype g;
   p=mbfp;
   if(x<=p->val)  return p->grd;
   if(x>=(p+num-1)->val)  return (p+num-1)->grd;
   for(i=0; i<num-1; i++){
      p=mbfp+i; if(x>p->val&&x<=(p+1)->val) break;
      }
   a = (temptype)((p+1)->grd) - (temptype) (p->grd);
   a = a/((temptype)((p+1)->val)- (temptype) (p->val));
   b = (temptype)(p->grd)-a*(p->val);
   a = a*x+b;
   if (a<0.0) g=0; else g=(gradetype)(a+0.5);
   return (g);
 }

/* Cubic MBF evaluation subroutine */
gradetype Mbf_Eval_Cubi_AAO(MBF_CUBI *mbfp, int16 num, valuetype x)
{
   MBF_CUBI  *p;
   int16  i;
```

```
   temptype x1, x2, x3, y1, y2, y3, k1, k2, k3, xx, yy;
   p=mbfp;
   if(x<=p->val)  return p->grd;
   if(x>=(p+num-1)->val)  return (p+num-1)->grd;
   for(i=0; i<num-1; i++){
      p=mbfp+i;  if(x>p->val&&x<=(p+1)->val) break;
      }
   x1 = (temptype)(p->val);      y1 = (temptype)(p->grd);
   x3 = (temptype)((p+1)->val); y3 = (temptype)((p+1)->grd);
   k1 = (y3-y1)/(x3-x1);
   xx = (temptype)x;
   if((p->cval)==0&&(p->cgrd)==0) return (gradetype) (k1*(xx-x1)+y1);
   x2 = (temptype)(p->cval); y2 = (temptype)(p->cgrd);
   k1 = y1/((x1-x2)*(x1-x3));  k2 = y2/((x2-x1)*(x2-x3));
   k3 = y3/((x3-x1)*(x3-x2));
   yy = k1*(xx-x2)*(xx-x3)+k2*(xx-x1)*(xx-x3)+k3*(xx-x1)*(xx-x2);
   if(yy<0.0) yy=0.0;
   if (yy>(temptype)MAX_UNSIGNED_LONG) yy=(temptype)MAX_UNSIGNED_LONG-0.5;
   return (gradetype) (yy+0.5);
 }


/* Operation defined MBF evaluation subroutine */
gradetype Mbf_Eval_Opr_AAO(MBF_OPR *mbfp, int16 num, valuetype x)
{
   MBF_OPR  *p;
   int16  i;
   temptype *temp, ret;
   if (temp=(temptype *) malloc((num+1)*sizeof(temptype)), temp==NULL){
       Report_Error("Not Enough Memory (Mbf_Eval_Opr_())");
       return(0); }
   for (i=0;i<num;i++){
      p=mbfp+i;
      if (p->opt==-1){
  if (p->n1==MF_LUT)
  temp[p->ord]=(temptype)Mbf_Eval_Lut_AAO((MBF_LUT *)p->opd, p->n2, x-
(valuetype)(p->opp));
  if (p->n1==MF_PLG)
  temp[p->ord]=(temptype)Mbf_Eval_Poly_AAO((MBF_POLY *)p->opd, p->n2, x-
(valuetype)(p->opp));
  if (p->n1==MF_CUB)
  temp[p->ord]=(temptype)Mbf_Eval_Cubi_AAO((MBF_CUBI *)p->opd, p->n2, x-
(valuetype)(p->opp));
  }
      if ((p->opt)<SHIFT && (p->opt)>-1) temp[p->ord]=Fl_Operation_AO(temp
[p->n1], temp[p->n2], p->opt);
      if (p->opt==SHIFT) temp[p->ord]=temp[p->n1];
      if (p->opt==CUT) temp[p->ord]=Min1(temp[p->n1], p->opp);
      if (p->opt==CONC){
   if ((p->opp)>x) temp[p->ord]=temp[p->n1];
   else temp[p->ord]=temp[p->n2];}
      if (p->opt==NOT) temp[p->ord]=Max_Grd-temp[p->n1];
      if (p->opt==BCUT){
   if ((p->opp)>temp[p->n1]) temp[p->ord]=0;
   else temp[p->ord]=temp[p->n1]; }
      if (p->opt==WEIGHT) temp[p->ord]=temp[p->n1]*((p->opp)/(temptype)
Max_Grd);
    }
   if (temp[num-1]<0.0) ret=0; else ret=(gradetype)(temp[num-1]+0.5);
   free(temp); return ret;
 }
#endif
/* Logic operations */
#ifndef _FIU_OPER_AO_
#define _FIU_OPER_AO_
temptype Fl_Operation_AO(temptype n1, temptype n2, int16 opt)
{
   temptype temp;
     if (opt==MIN) temp=Min1(n1, n2);
```

```
      if (opt==MAX) temp=Max1(n1, n2);
      if (opt==PROD) temp=Prd1(n1, n2);
      if (opt==PSUM) temp=Sum1(n1, n2);
      if (opt==BINTER) temp=Bdi1(n1, n2);
      if (opt==BUNION) temp=Bdu1(n1, n2);
   return temp;
}
#endif

/* Evaluate Rules */
#ifndef _FIU_REVAL_AAO_
#define _FIU_REVAL_AAO_
Rule_Eval_AAO(CODE *code, MBF_POLY **s, temptype *output,
  gradetype *tv_output, gradetype *tv_input, int16 ap, int16 op, uint16 m)
{
int16 i,v,l, t1;
uint8 c, c1;
CODE *p;
valuetype val_left, val_right, value;
temptype tv_or, tv_and, tv_max, fz, fm;
   p=(CODE *)code; i=p->t;  c=p->c;
   l=0; v=0; fm=0; fz=0; tv_and=0; tv_or=0;
   val_left=0; val_right=0; tv_max=0;
   while (1){
      if ((c&0x80)==0x80){
    output[v]=0;
    switch(c&0x0F){
    case 0: if (fabs(fm)>0.0001) output[v]=fz/fm; break;
    case 1: output[v]=(val_left+val_right)/2.0; break;
    case 2: output[v]=val_left; break;
    case 3: output[v]=val_right; break;
    case 4: output[v]=tv_or; break;
    default: if (fabs(fm)>0.0001) output[v]=fz/fm; break;
    }
    v++; l=0; fm=0; fz=0; tv_and=0; tv_or=0;
    val_left=0; val_right=0; tv_max=0;
    if ((c&0x40)==0x40) break;
   }
      else {
    if ((c&0x40)==0x40) tv_and=Fl_Operation_AO(tv_input[i],tv_and, ap);
    else tv_and=tv_input[i];
    if ((c&0x20)==0x20){
       if ((c&0x10)==0x10) tv_or=Fl_Operation_AO(tv_and, tv_or, op);
       else tv_or=tv_and;
       }
    if ((c&0x08)==0x08) {
       p++; c1=p->c;  t1=p->t;
       if ((c&0x04)==0x04) {
         if (c1==0){
        if (tv_or>=tv_max){
          value=(s[t1])->val;
          if (value<=val_left){ tv_max=tv_or;val_left=value;}
          if (value>=val_right){ tv_max=tv_or; val_right=value;}
          }
        }
          else {
      tv_output[v*m+l] = (gradetype) tv_or;
      l++;
      }
          }
       else {
         if (c1==0) {
       fz=fz+tv_or*(s[t1])->val;
       fm=fm+tv_or;
       }
          else {
       tv_output[v*m+l] = (gradetype) tv_or;
       l++;
       }
```

```
        }
      }
   }
     p++; c=p->c; i=p->t;                    .
    }
  return 0;
}
#endif

/* Procedure to report errors, user may modify it */
#ifndef _FIU_REPORT_ERROR
#define _FIU_REPORT_ERROR
void Report_Error(char *str)
{
   printf("%s\n",str);
}
#endif
#undef  gradetype
#undef  valuetype
#undef  maxtype
#undef  temptype
#undef  MBF_LUT
#undef  MBF_POLY
#undef  MBF_CUBI
#undef  MBF_OPR
#undef  Max_Grd
#undef  Prd1
#undef  Sum1
#undef  Bdi1
#undef  Bdu1
```

3. ตัวสร้างกฎพื้นฐานในการควบคุม(Rule-base)ของระยะทางตัวฐานรถเลื่อนcart

file name: pndrule2.cpp

```
/*  C Function performs fuzzy inference.
*   Original(Source) File Name pndrule2.cpp */

#ifndef _FIU_TYPES_IN_
#define _FIU_TYPES_IN_
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
typedef  float    float32;
typedef double    float64;
typedef char    int8;
typedef short int   int16;
typedef long    int32;
typedef unsigned char uint8;
typedef unsigned short int  uint16;
typedef unsigned long uint32;
#endif

/* Data structure for mbfs, edit only if you are sure */
#define  gradetype uint8 /* for truth values */
#define  valuetype uint8 /* for max i/o values */
#define  maxtype   uint8 /* max of gradetype and valuetype */
#define  temptype  float32 /* for temp values */
/* Maximal 32 bit unsigned long integer:*/
#define MAX_UNSIGNED_LONG 4294967295

#ifndef _FIU_MOTYPES_
#define _FIU_MOTYPES_
typedef  struct { void *m; int16 type, num;} MBF_OUT;
#endif
#ifndef _FIU_MLUT_A_
#define _FIU_MLUT_A_
typedef  gradetype  MBF_LUT_A;
#endif
#ifndef _FIU_MPCO_AA_
#define _FIU_MPCO_AA_
typedef struct { valuetype val; gradetype grd; } MBF_POLY_AA;
typedef struct { valuetype val; gradetype grd;
     valuetype cval; gradetype cgrd; } MBF_CUBI_AA;
typedef struct { int8 opt; void *opd;int16  n1, n2;
     maxtype opp; int16 ord; } MBF_OPR_AA;
#endif

#ifndef _FIU_CODEDEF_
#define _FIU_CODEDEF_
typedef struct{uint16 t; uint8 c;} CODE;
#endif
#define MBF_LUT  MBF_LUT_A
#define MBF_POLY MBF_POLY_AA
#define MBF_CUBI MBF_CUBI_AA
#define MBF_OPR  MBF_OPR_AA

/* Function prototypes (Mbf evaluation, defuzzification) */
#ifndef _FIU_P_MEVAL_AAO_
#define _FIU_P_MEVAL_AAO_
gradetype Mbf_Eval_Poly_AAO(MBF_POLY*, int16, valuetype);
gradetype Mbf_Eval_Cubi_AAO(MBF_CUBI*, int16, valuetype);
gradetype Mbf_Eval_Opr_AAO(MBF_OPR*, int16, valuetype);
gradetype Mbf_Eval_Lut_AAO(MBF_LUT*, int16, valuetype);
#endif
#ifndef _FIU_P_REPORT_ERROR
```

```
#define _FIU_P_REPORT_ERROR
void Report_Error(char *str);
#endif
#ifndef _FIU_P_REVAL_AAO_
#define _FIU_P_REVAL_AAO_
Rule_Eval_AAO(CODE *, MBF_POLY **, temptype *, gradetype *, gradetype *,
int16, int16, uint16);
#endif
#ifndef _FIU_P_OPER_AO_
#define _FIU_P_OPER_AO_
temptype Fl_Operation_AO(temptype, temptype, int16);
#endif

/* Constants */
#ifndef _FIU_CONSTS_IN_
#define _FIU_CONSTS_IN_
#define MF_LUT  0
#define MF_PLG  1
#define MF_OPR  2
#define MF_CUB  3
#define MF_SGT  4
#define MIN0
#define MAX1
#define PROD  2
#define PSUM  3
#define BINTER  4
#define BUNION  5
#define SHIFT 6
#define CUT7
#define CONC  8
#define NOT9
#define BCUT    10
#define WEIGHT  11
#define DM_CENTROID 0
#define DM_AMAX 1
#define DM_LMAX 2
#define DM_RMAX 3
#define DM_NODFZ 255
#endif

/* Macros */
#define Max_Grd 255
#define Min1(x,y)  ((x)<(y)?(x):(y))
#define Max1(x,y)  ((x)<(y)?(y):(x))
#define Prd1(x,y)  ((x)*(((temptype) y)/Max_Grd))
#define Sum1(x,y)  ((x)+(y)-(x)*(((temptype) y)/Max_Grd))
#define Bdi1(x,y)  (((x)+(y))>Max_Grd?((x)+(y)-Max_Grd):0)
#define Bdu1(x,y)  (((x)+(y))<Max_Grd?((x)+(y)):Max_Grd)

 /* define MBFs */
MBF_POLY pndrule2_mbf0[4]={{0,255}, {16,255}, {64,0}, {255,0}};
MBF_POLY pndrule2_mbf1[5]={{0,0}, {16,0}, {64,255}, {112,0}, {255,0}};
MBF_POLY pndrule2_mbf2[5]={{0,0}, {64,0}, {112,255}, {144,0}, {255,0}};
MBF_POLY pndrule2_mbf3[5]={{0,0}, {112,0}, {144,255}, {192,0}, {255,0}};
MBF_POLY pndrule2_mbf4[5]={{0,0}, {144,0}, {192,255}, {240,0}, {255,0}};
MBF_POLY pndrule2_mbf5[4]={{0,0}, {192,0}, {240,255}, {255,255}};
MBF_POLY pndrule2_mbf6[1]={{0, 255}};
MBF_POLY pndrule2_mbf7[1]={{51, 255}};
MBF_POLY pndrule2_mbf8[1]={{102, 255}};
MBF_POLY pndrule2_mbf9[1]={{153, 255}};
MBF_POLY pndrule2_mbf10[1]={{204, 255}};
MBF_POLY pndrule2_mbf11[1]={{255, 255}};
MBF_POLY pndrule2_mbf12[2]={{0,0}, {255,255}};
MBF_POLY * pndrule2_mbfs[6]={pndrule2_mbf6, pndrule2_mbf7, pndrule2_mbf8,
pndrule2_mbf9,
  pndrule2_mbf10, pndrule2_mbf11, };
CODE  pndrule2_code[79] ={{2,0},{7,96},
{2,0},{6,112},
{1,0},{8,112},
```

```
{1,0},{7,112},
{1,0},{6,112},
{0,0},{9,112},
{0,0},{8,112},
{0,0},{7,112},
{0,0},{6,120},
{0,0},
{3,0},{6,96},
{2,0},{8,112},
{1,0},{9,112},
{0,0},{10,120},
{1,0},
{4,0},{6,96},
{3,0},{7,112},
{2,0},{9,112},
{1,0},{10,112},
{0,0},{11,120},
{2,0},
{5,0},{6,96},
{4,0},{7,112},
{3,0},{8,112},
{2,0},{10,112},
{1,0},{11,120},
{3,0},
{5,0},{7,96},
{4,0},{8,112},
{3,0},{9,112},
{2,0},{11,120},
{4,0},
{5,0},{11,96},
{5,0},{10,112},
{5,0},{9,112},
{5,0},{8,112},
{4,0},{11,112},
{4,0},{10,112},
{4,0},{9,112},
{3,0},{11,112},
{3,0},{10,120},
{5,0},
{0,192},
};

/* Function body of the FIU module */
void exec_pndrule2(uint8 input0, uint8 input1,
  uint8 *output0 )
 {
   /* Inside this function, the variables
      corresponds to these in the FIL file as follows: */
   /* input0 = theta */
   /* input1 = omega */
   /* output0 = voltage */
   #define InVar     2
   #define OutVar    1
   #define InTv      12
   #define OutMaxTv  6
   int16 k;
   temptype dfz_fm, dfz_fz;
   temptype input[InVar], output[OutVar];
   gradetype tv_input[InTv];
   gradetype tv_output[1];

   /* Ranges for the variables */
   temptype input_range[2][3]={{0.000000, 255.000000, 1.000000e+00},
  {0.000000, 255.000000, 1.000000e+00}};
   temptype output_range[1][3]={{0.000000, 255.000000, 1.000000e+00}};

  input[0]=input0;
  input[1]=input1;
  for (k=0;k<InVar;k++){
```

```
     dfz_fz=input[k];
     if(dfz_fz<input_range[k][0]) dfz_fz=input_range[k][0];
     if(dfz_fz>input_range[k][1]) dfz_fz=input_range[k][1];
     dfz_fm = (dfz_fz-input_range[k][0]) / input_range[k][2]+0.5;
     input[k] = dfz_fm;
    }

    tv_input[0] =Mbf_Eval_Poly_AAO(pndrule2_mbf0, 4, (valuetype)input[0]);
    tv_input[1] =Mbf_Eval_Poly_AAO(pndrule2_mbf1, 5, (valuetype)input[0]);
    tv_input[2] =Mbf_Eval_Poly_AAO(pndrule2_mbf2, 5, (valuetype)input[0]);
    tv_input[3] =Mbf_Eval_Poly_AAO(pndrule2_mbf3, 5, (valuetype)input[0]);
    tv_input[4] =Mbf_Eval_Poly_AAO(pndrule2_mbf4, 5, (valuetype)input[0]);
    tv_input[5] =Mbf_Eval_Poly_AAO(pndrule2_mbf5, 4, (valuetype)input[0]);
    tv_input[6] =Mbf_Eval_Poly_AAO(pndrule2_mbf0, 4, (valuetype)input[1]);
    tv_input[7] =Mbf_Eval_Poly_AAO(pndrule2_mbf1, 5, (valuetype)input[1]);
    tv_input[8] =Mbf_Eval_Poly_AAO(pndrule2_mbf2, 5, (valuetype)input[1]);
    tv_input[9] =Mbf_Eval_Poly_AAO(pndrule2_mbf3, 5, (valuetype)input[1]);
    tv_input[10] =Mbf_Eval_Poly_AAO(pndrule2_mbf4, 5, (valuetype)input[1]);
    tv_input[11] =Mbf_Eval_Poly_AAO(pndrule2_mbf5, 4, (valuetype)input[1]);
   Rule_Eval_AAO(pndrule2_code, pndrule2_mbfs, output, tv_output, tv_input,
0, 1, 6);

  /* Denormalize output values  */
      *output0=output[0] *output_range[0][2]+output_range[0][0];
   #undef InVar
   #undef OutVar
   #undef InTv
   #undef OutMaxTv
}

#ifndef _FIU_MEVAL_AAO_
#define _FIU_MEVAL_AAO_
/* Lookup table MBF evaluation subroutine */
gradetype Mbf_Eval_Lut_AAO(MBF_LUT *mbfp, int16 num, valuetype x)
{
   MBF_LUT *p;
   p=mbfp;
   if (x<=p[num-2]) return p[0];
   if (x>=p[num-1]) return p[num-3];
   return p[(int16)(x-p[num-2])];
 }

/* Polygon MBF evaluation subroutine */
gradetype Mbf_Eval_Poly_AAO(MBF_POLY *mbfp, int16 num, valuetype x)
{
   MBF_POLY  *p;
   int16  i;
   temptype  a, b;
   gradetype g;
   p=mbfp;
   if(x<=p->val)  return p->grd;
   if(x>=(p+num-1)->val)  return (p+num-1)->grd;
   for(i=0; i<num-1; i++){
      p=mbfp+i; if(x>p->val&&x<=(p+1)->val) break;
      }
   a = (temptype)((p+1)->grd) - (temptype) (p->grd);
   a = a/((temptype)((p+1)->val)- (temptype) (p->val));
   b = (temptype)(p->grd)-a*(p->val);
   a = a*x+b;
   if (a<0.0) g=0; else g=(gradetype)(a+0.5);
   return (g);
 }

/* Cubic MBF evaluation subroutine */
gradetype Mbf_Eval_Cubi_AAO(MBF_CUBI *mbfp, int16 num, valuetype x)
{
   MBF_CUBI  *p;
   int16  i;
   temptype x1, x2, x3, y1, y2, y3, k1, k2, k3, xx, yy;
```

```
   p=mbfp;
   if(x<=p->val)  return p->grd;
   if(x>=(p+num-1)->val)  return (p+num-1)->grd;
   for(i=0; i<num-1; i++){
      p=mbfp+i;  if(x>p->val&&x<=(p+1)->val) break;
      }
   x1 = (temptype)(p->val);     y1 = (temptype)(p->grd);
   x3 = (temptype)((p+1)->val); y3 = (temptype)((p+1)->grd);
   k1 = (y3-y1)/(x3-x1);
   xx = (temptype)x;
   if((p->cval)==0&&(p->cgrd)==0) return (gradetype) (k1*(xx-x1)+y1);
   x2 = (temptype)(p->cval); y2 = (temptype)(p->cgrd);
   k1 = y1/((x1-x2)*(x1-x3));  k2 = y2/((x2-x1)*(x2-x3));
   k3 = y3/((x3-x1)*(x3-x2));
   yy = k1*(xx-x2)*(xx-x3)+k2*(xx-x1)*(xx-x3)+k3*(xx-x1)*(xx-x2);
   if(yy<0.0) yy=0.0;
   if (yy>(temptype)MAX_UNSIGNED_LONG) yy=(temptype)MAX_UNSIGNED_LONG-0.5;
   return (gradetype) (yy+0.5);
}


/* Operation defined MBF evaluation subroutine */
gradetype Mbf_Eval_Opr_AAO(MBF_OPR *mbfp, int16 num, valuetype x)
{
   MBF_OPR  *p;
   int16  i;
   temptype *temp, ret;
   if (temp=(temptype *) malloc((num+1)*sizeof(temptype)), temp==NULL){
       Report_Error("Not Enough Memory (Mbf_Eval_Opr_())");
       return(0); }
   for (i=0;i<num;i++){
      p=mbfp+i;
      if (p->opt==-1){
  if (p->n1==MF_LUT)
  temp[p->ord]=(temptype)Mbf_Eval_Lut_AAO((MBF_LUT *)p->opd, p->n2, x-
(valuetype)(p->opp));
  if (p->n1==MF_PLG)
  temp[p->ord]=(temptype)Mbf_Eval_Poly_AAO((MBF_POLY *)p->opd, p->n2, x-
(valuetype)(p->opp));
  if (p->n1==MF_CUB)
  temp[p->ord]=(temptype)Mbf_Eval_Cubi_AAO((MBF_CUBI *)p->opd, p->n2, x-
(valuetype)(p->opp));
  }
      if ((p->opt)<SHIFT && (p->opt)>-1) temp[p->ord]=Fl_Operation_AO(temp
[p->n1], temp[p->n2], p->opt);
      if (p->opt==SHIFT) temp[p->ord]=temp[p->n1];
      if (p->opt==CUT) temp[p->ord]=Min1(temp[p->n1], p->opp);
      if (p->opt==CONC){
   if ((p->opp)>x) temp[p->ord]=temp[p->n1];
   else temp[p->ord]=temp[p->n2];}
      if (p->opt==NOT) temp[p->ord]=Max_Grd-temp[p->n1];
      if (p->opt==BCUT){
   if ((p->opp)>temp[p->n1]) temp[p->ord]=0;
   else temp[p->ord]=temp[p->n1]; }
      if (p->opt==WEIGHT) temp[p->ord]=temp[p->n1]*((p->opp)/(temptype)
Max_Grd);
    }
   if (temp[num-1]<0.0) ret=0; else ret=(gradetype)(temp[num-1]+0.5);
   free(temp); return ret;
 }
#endif
/* Logic operations */
#ifndef _FIU_OPER_AO_
#define _FIU_OPER_AO_
temptype Fl_Operation_AO(temptype n1, temptype n2, int16 opt)
{
   temptype temp;
     if (opt==MIN) temp=Min1(n1, n2);
     if (opt==MAX) temp=Max1(n1, n2);
```

```
      if (opt==PROD) temp=Prd1(n1, n2);
      if (opt==PSUM) temp=Sum1(n1, n2);
      if (opt==BINTER) temp=Bdi1(n1, n2);
      if (opt==BUNION) temp=Bdu1(n1, n2);
    return temp;
}
#endif

/* Evaluate Rules */
#ifndef _FIU_REVAL_AAO_
#define _FIU_REVAL_AAO_
Rule_Eval_AAO(CODE *code, MBF_POLY **s, temptype *output,
  gradetype *tv_output, gradetype *tv_input, int16 ap, int16 op, uint16 m)
{
int16 i,v,l, t1;
uint8 c, c1;
CODE *p;
valuetype val_left, val_right, value;
temptype tv_or, tv_and, tv_max, fz, fm;
   p=(CODE *)code; i=p->t;  c=p->c;
   l=0; v=0; fm=0; fz=0; tv_and=0; tv_or=0;
   val_left=0; val_right=0; tv_max=0;
   while (1){
      if ((c&0x80)==0x80){
    output[v]=0;
    switch(c&0x0F){
    case 0: if (fabs(fm)>0.0001) output[v]=fz/fm; break;
    case 1: output[v]=(val_left+val_right)/2.0; break;
    case 2: output[v]=val_left; break;
    case 3: output[v]=val_right; break;
    case 4: output[v]=tv_or; break;
    default: if (fabs(fm)>0.0001) output[v]=fz/fm; break;
    }
    v++; l=0; fm=0; fz=0; tv_and=0; tv_or=0;
    val_left=0; val_right=0; tv_max=0;
    if ((c&0x40)==0x40) break;
   }
      else {
    if ((c&0x40)==0x40) tv_and=Fl_Operation_AO(tv_input[i],tv_and, ap);
    else tv_and=tv_input[i];
    if ((c&0x20)==0x20){
       if ((c&0x10)==0x10) tv_or=Fl_Operation_AO(tv_and, tv_or, op);
       else tv_or=tv_and;
       }
    if ((c&0x08)==0x08) {
       p++; c1=p->c;  t1=p->t;
       if ((c&0x04)==0x04) {
         if (c1==0){
      if (tv_or>=tv_max){
        value=(s[t1])->val;
        if (value<=val_left){ tv_max=tv_or;val_left=value;}
        if (value>=val_right){ tv_max=tv_or; val_right=value;}
        }
      }
         else {
     tv_output[v*m+l] = (gradetype) tv_or;
     l++;
     }
        }
       else {
         if (c1==0) {
      fz=fz+tv_or*(s[t1])->val;
      fm=fm+tv_or;
      }
         else {
      tv_output[v*m+l] = (gradetype) tv_or;
      l++;
      }
        }
```

```
      }
   }
      p++; c=p->c; i=p->t;
     }
  return 0;
}
#endif

/* Procedure to report errors, user may modify it */
#ifndef _FIU_REPORT_ERROR
#define _FIU_REPORT_ERROR
void Report_Error(char *str)
{
   printf("%s\n",str);
}
#endif
#undef  gradetype
#undef  valuetype
#undef  maxtype
#undef  temptype
#undef  MBF_LUT
#undef  MBF_POLY
#undef  MBF_CUBI
#undef  MBF_OPR
#undef  Max_Grd
#undef  Prd1
#undef  Sum1
#undef  Bdi1
#undef  Bdu1
```

4. โปรแกรมหลักของระบบควบคุมแบบฟัซซี่สำหรับอินเวอร์เท็ดเพนดูลัม

file name: pnd1sys.cpp

```
/*****************************************
* C Source Code
* pend1sys.cpp : 10/1/2001
*****************************************/


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <io.h>
#include <mem.h>
#define  __boundToChar(x)     ( (x>127) ?(127) : ((x>-127) ? (x) : (-127)))
#define  __boundToUChar(x)     ( (x>255) ?(255) : ((x>0) ? (x) : (0)))
#define  __boundToInt(x)     ( (x>32768) ?(32768) : ((x>-32768) ? (x) : (-
32768)))
#define  __boundToUInt(x)     ( (x>65535) ?(65535) : ((x>0) ? (x) : (0)))
#include  "c:\tcp\project\attemp\pndrule2.cpp"
#include  "c:\tcp\project\attemp\pndrule1.cpp"
#include "c:\tcp\project\attemp\invpend1.cpp"

/*  System input and output variable definitions */
 float x=0.000000;
  float dx=0.000000;
  float t=0.000000;
  float dt=0.000000;
 FILE    *__debug, *__simul, *__load=NULL;

float  __0;
float  __1;
float  __2;
float  __3;
float  __4;
unsigned char  __5;
float  __6;
float  __7;
float  __8;
char  __9;
float  __10;
float  __11;
float  __12;
float  __13;

float  ___9;

int   __LoadCloseValues(void);  /* Prototype */
int   __LoadCloseValues(){
   float tmp0,tmp1,tmp2,tmp3;
   if(__load==NULL) return(1);
   if(fscanf(__load, "%f", &tmp0)==EOF) return(0);
   x=(float )tmp0;
   if(fscanf(__load, "%f", &tmp1)==EOF) return(0);
   dx=(float )tmp1;
   if(fscanf(__load, "%f", &tmp2)==EOF) return(0);
   t=(float )tmp2;
   if(fscanf(__load, "%f", &tmp3)==EOF) return(0);
   dt=(float )tmp3;
   return(1);
}
```

```
int   __LoadOpenValues(void); /* Prototype */
int   __LoadOpenValues(){
   float tmp0,tmp1,tmp2,tmp3;
   if(__load==NULL) return (0);
   if(fscanf(__load, "%f", &tmp0)==EOF) return(0);
   if(fscanf(__load, "%f", &tmp1)==EOF) return(0);
   if(fscanf(__load, "%f", &tmp2)==EOF) return(0);
   if(fscanf(__load, "%f", &tmp3)==EOF) return(0);
   return(1);
}

#define  _op1(x, y)  y=x*20.000000+127.000000
#define  _op2(x, y)  y=x*50.000000+128.000000
struct _pndrule2{
unsigned char    voltage;
}  _pndrule2_out;

run_pndrule2(unsigned char    theta,
unsigned char    omega,
struct _pndrule2  *_pndrule2_out)
{
   exec_pndrule2(theta, omega, &_pndrule2_out->voltage);
   return 1;
}

#define  _op3(a, b, y)  y=a*(255.000000/3.140000)*5.000000+(b-
127.000000)*0.500000
#define  _op4(x, y)  y=x*(255.000000/3.140000)*0.300000
struct _pndrule1{
char    force;
}  _pndrule1_out;

run_pndrule1(char    pend_angle,
char    pend_velocity,
struct _pndrule1  *_pndrule1_out)
{
   exec_pndrule1(pend_angle, pend_velocity, &_pndrule1_out->force);
   return 1;
}

struct _pdemo{
float    x;
float    dx;
float    t;
float    dt;
}  _pdemo_out;

run_pdemo(float    *f,
struct _pdemo  *_pdemo_out)
{
   return(pdemo(f, &_pdemo_out->x, &_pdemo_out->dx, &_pdemo_out->t,
&_pdemo_out->dt));
}


int  main(int argc, char *argv[]){
init_pdemo();
if(argc>=2)
    __load = fopen(argv[1], "r");
if(argc>=2)
    __LoadCloseValues();
 while(1){
__0 =  x;
__1 =  dx;
 _op1(__0, __2);
__2 =  __2;
 _op2(__1, __3);
__3 =  __3;
__4 =  t;
```

```
__2 = __boundToUChar(__2);
__3 = __boundToUChar(__3);
run_pndrule2(__2, __3, &_pndrule2_out);
__5 =  _pndrule2_out.voltage;
__6 =  dt;
 _op3(__4, __5, __7);
__7 =  __7;
 _op4(__6, __8);
__8 =  __8;
__7 = __boundToChar(__7);
__8 = __boundToChar(__8);
run_pndrule1(__7, __8, &_pndrule1_out);
__9 =  _pndrule1_out.force;
___9 = __9;
if(!run_pdemo(&___9, &_pdemo_out)) break;
__10 =  _pdemo_out.x;
x = __10;
__11 =  _pdemo_out.dx;
dx = __11;
__12 =  _pdemo_out.t;
t = __12;
__13 =  _pdemo_out.dt;
dt = __13;

__LoadOpenValues();
}
if(__load!=NULL) fclose(__load);
close_pdemo();
}
```