

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

การจำลองการเรียนรู้ระบบการทรงตัว

Simulation of learning to control an Inverted Pendulum



เลขหมู่.....
เลขทะเบียน..... 73177
วัน,เดือน,ปี..... 10 11.พ. 2550

b. 11788020
i.

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาตรีวิศวกรรมศาสตรบัณฑิต
สาขาวิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2548

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การจำลองการเรียนรู้ระบบการทรงตัว
Simulation of learning to control an Inverted Pendulum



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร ปริญญาตรี วิศวกรรมศาสตรบัณฑิต
สาขาวิชา อิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2548

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ ปีการศึกษา 2548

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การจำลองการเรียนรู้ระบบการทรงตัว

Simulation of learning to control an Inverted Pendulum

ผู้จัดทำ

1. นายไพฑูรย์ ชัยโชติอนันต์



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แบบจำลองระบบเรียนรู้การทรงตัว

นายไพฑูรย์ ชัยโชติอนันต์ รหัส 45010560
ผศ.ดร.ยุทธนา คัดใจเดียว อาจารย์ที่ปรึกษา
ภาคเรียนที่ 2 ปีการศึกษาที่ 2548

บทคัดย่อ

การเรียนรู้การทรงตัวด้วยโครงข่ายประสาทเทียมชนิด ไม่มีต้นแบบ อาศัยการเรียนรู้ด้วยการให้รางวัลและการยับยั้งการเรียนรู้ การเรียนรู้ดังกล่าวต้องอาศัยการจำลองอินเวอร์ตเพนดูลัม เพื่อให้ค่าผ่านตัวแปร แก่โครงข่ายประสาทเทียม เนื่อง ไขของการเรียนรู้จะขึ้นกับมุมที่สามารถรักษาไว้ให้ก้านนำหนักทรงตัวอยู่ได้ ชนิดของโครงข่ายประสาทเทียมชนิดเซลล์ฟอออกาโนซึ่งพีเจอร์แมพ จะคัดเลือกแรงที่ดีที่สุดกับสถานการณ์นั้น ๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Simulation Model of learning to control an Inverted Pendulum

Mr. Pitoon Chaichodanan ID.45010560

Assist.Prof.Dr.Yuddhana Kidchaidiew Advisor

2005

Abstract

This Thesis concern about learning of an Inverted Pendulum using Artificial Neural Network. An unsupervised Network are learn by rewarding and inhibiting the network. Learning must learn parameter from model of an inverted pendulum that gives the network. The conditions to balance the Pole is related to balance the Pole for a while. Self-Organizing Feature Map is elected as artificial neural network due to its strong firing.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิติกรรมประกาศ

ขอขอบคุณ

มารดา และ พี่ชาย สำหรับ วิธี คำลึงใจความไม่ยอมแพ้
ผศ.ดร. บุศรนา คิดใจเดียว สำหรับทุก ๆ อย่างของ โปรเจกนี้
อาจารย์ เทอดศักดิ์ ลีวหาทอง สำหรับเทคนิคเขียน โปรแกรม
รศ.ดร.กิตติ ไพบุญย์วัฒนกิจ สำหรับคำแนะนำที่ไม่ควรมองข้าม
อ.บุญยชนะ ภูระหงส์ สำหรับแบบฝึกหัดการวิเคราะห์ปัญหาในเชิงสังคมศาสตร์
อ.ธนาบุศุช เชื้อเจริญ มหาลัยนเรศวร สำหรับความห่วงใยการเรียนของข้าพเจ้า
อ.ภูริพงษ์ ทองแข็ง สำหรับพื้นฐานการเขียนภาษาซี
เพื่อน ๆ ห้อง 408 ทุกคน และ เพื่อนพี่น้องภาคอิเล็กทรอนิกส์



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

หน้า

บทคัดย่อ

Abstract

กิตติกรรมประกาศ

สารบัญ

สารบัญรูป

บทที่ 1 บทนำ	1
บทที่ 2 ทฤษฎี ของระบบการทรงตัว และ โครงข่ายประสาทเทียม	
2.1 ทฤษฎี อินเวอร์ตเพนดูลัม (Inverted Pendulum)	3
2.2 โครงข่ายประสาทเทียม (Artificial Neural Network ANN)	7
2.3 แบบจำลองของ โครงข่ายประสาทเทียม Backpropagation	10
2.4 การประมาณค่าฟังก์ชัน	11
2.5 วิธีสุ่มการกระจายค่าปกติ	12
2.6 โครงสร้างแบบไม่มีต้นแบบ (Unsupervised Network)	12
2.7 Self-Organizing Feature Map	13
บทที่ 3 การออกแบบโปรแกรมจำลองการทำงานของประสาทเทียม	
3.1 การออกแบบแบบจำลอง Inverted Pendulum	15
3.2 วิธีการเก็บข้อมูล	19
3.3 การสร้าง Neural Network	21
3.4 เปรียบเทียบการเรียนรู้ของ โครงข่าย	21
3.5 รูปแบบของโครงข่ายประสาทเทียมที่นำไปใช้	22
3.6 การสร้างนำโครงข่ายประสาทเทียมเข้าไปใช้ Simulink	23
3.7 การออกแบบ โครงข่ายประสาทเทียมแบบ Self-Organized Feature Map	25
3.8 องค์ประกอบของ Pendulum ในภาษาซี	26
3.9 การเก็บข้อมูลบนภาษาซี	31
3.10 องค์ประกอบของ โครงข่ายประสาทเทียมแบบไม่มีต้นแบบ	31
3.11 ขั้นตอนการทำงานของ โครงข่ายประสาทเทียม	31

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4	การทดสอบโปรแกรม	
การทดลองที่ 1	กราฟความสัมพันธ์ต่างของตัวแปร	39
การทดลองที่ 2	ความสัมพันธ์ระหว่าง และ	41
การทดลองที่ 3	ทดสอบการแสดงผลการทำงานของระบบ	43
การทดลองที่ 4	แสดงผลการตอบสนองของอินเวอร์ตเพนดูลัมบนภาษาซี	44
การทดลองที่ 5	ผลตอบสนองต่อ Impulse ของระบบ	49
การทดลองที่ 6	การควบคุมอินเวอร์ตเพนดูลัมด้วยโปรแกรมเมทแลป	50
การทดลองที่ 6.1	การสังเกตผลการทดลองการทรงตัวของเพนดูลัมด้วยโครงข่ายประสาทเทียม	50
การทดลองที่ 6.2	ทดสอบการเพิ่มประสิทธิภาพของโครงข่ายประสาทเทียม	51
บทที่ 5	สรุปผลการดำเนินงาน	
สรุปผล		53
ปัญหาการดำเนินงาน		53
บรรณานุกรม		54
ภาคผนวก ก		53
ภาคผนวก ข		72



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูป

	หน้า
รูปที่ 2.1 แสดงแรงต่าง ๆ ที่เกิดขึ้นบน รถ และองค์ประกอบที่จะต้องใช้พิจารณา	4
รูปที่ 2.2 แสดง เซลล์ประสาทหนึ่งเซลล์	7
รูปที่ 2.3 แสดง โครงสร้างของโครงข่ายประสาทเทียมอย่างง่าย ๆ	7
รูปที่ 2.4 แสดง โครงสร้างของฟังก์ชันของประสาทเทียมแบบBackpropagation	8
รูปที่ 2.5 แสดง ฟังก์ชัน โอนถ่าย log sigmoid	10
รูปที่ 2.6 แสดงฟังก์ชัน โอนถ่าย tan sigmoid	10
รูปที่ 2.7 แสดงฟังก์ชัน โอนถ่ายเชิงเส้น	10
รูปที่ 2.8 แสดงตัวอย่างของ Function Approximation Network	11
รูปที่ 2.9 แสดงการกระจายตัวของค่าความน่าจะเป็น	12
รูปที่ 2.10 แสดงโครงข่าย แบบ Competitive Network	13
รูปที่ 3.1 แสดง Block Diagram ของระบบทั้งหมด	16
รูปที่ 3.2 แสดงฟังก์ชันย่อยของฟังก์ชันPID	16
รูปที่ 3.3 แสดงฟังก์ชันย่อยของ ฟังก์ชัน Inverted Pendulum	17
รูปที่ 3.4 แสดงฟังก์ชันรักษามุมคงที่	18
รูปที่ 3.5 แสดงวิธีการส่งค่าให้กับฟังก์ชันสำเร็จรูปของการแสดงผล	18
รูปที่ 3.6 แสดงหน้าต่างของการแสดงผลเสมือนจริง	18
รูปที่ 3.7 แสดงวิธีการเก็บข้อมูลลงในไฟล์และลงบนตัวแปร	19
รูปที่ 3.8 แสดงโครงข่ายประสาทเทียมสำหรับนำไปใช้งาน	22
รูปที่ 3.9แสดงตัวอย่างการเชื่อมต่อของทรานเฟอร์ฟังก์ชันแวกเตอร์ของ หนักและค่าอุปทาน	23
รูปที่ 3.10 แสดงผลที่ได้จากการสร้าง โครงข่ายประสาทเทียม โดยคำสั่ง gensim	23
รูปที่ 3.11 แสดงตัวอย่างนำโครงข่ายประสาทเทียมที่ผ่านการเรียนรู้เรียบร้อยแล้วไปใช้งาน	24
รูปที่ 3.12 แสดง โครงข่ายประสาทเทียมที่นำไปใช้งาน	25
รูปที่ 3.11 แสดงการสืบคลาสของ คลาส CDC	28
รูปที่ 3.12 แสดงวิธีการคิดสมการเพื่อแสดงผลของ ระบบอินเวอร์ตเพนดูลัม	28
รูปที่ 4.1 แสดงความสัมพันธ์ระหว่าง ความเร็วเชิงเส้น มุม และความเร็วเชิงมุมของ แบบจำลอง Inverted Pendulum	39
รูปที่ 4.2 แสดงความสัมพันธ์ของความเร็วเชิงเส้นและมุมของแบบจำลอง	39
รูปที่ 4.3 แสดงความสัมพันธ์ระหว่างความเร็วเชิงเส้นและความเร็วเชิงมุม	40
รูปที่ 4.4 แสดงความสัมพันธ์ของมุมและความเร็วเชิงมุมของแบบจำลอง	40

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 4.5 แสดงความสัมพันธ์ระหว่างแรง ความเร็วเชิงมุม และ ความเร็วเชิงเส้น	41
รูปที่ 4.6 แสดงความสัมพันธ์ระหว่างแรง และ ความเร็วเชิงเส้น	41
รูปที่ 4.7 แสดงความสัมพันธ์ระหว่างแรง และความเร็วเชิงมุม	42
รูปที่ 4.8 แสดง หน้าต่างการทำงานของ โปรแกรม	43
รูปที่ 4.9 แสดงผลตอบสนองตำแหน่งของฐาน	47
รูปที่ 4.10 แสดงผลตอบสนองทางความเร่งของฐาน	47
รูปที่ 4.11 แสดงผลตอบสนองต่อมุมของ เพนคูลัม	48
รูปที่ 4.12 แสดงผลตอบสนองต่อความเร่งเชิงมุมของเพนคูลัม	48
รูปที่ 4.13 ผลตอบสนองต่อ impulse ของระบบใน Time Domain	49
รูปที่ 4.14 แสดงผลการทำงานของ โครงข่ายประสาทเทียมเมื่อใช้ควบคุม การทรงตัวของเพนคูลัม	49
รูปที่ 4.15 วิธีการทดลองการควบคุมอินเวอร์ตเพนคูลัม โดยโครงข่ายประสาทเทียม	50
รูปที่ 4.16 แสดงผลของการทดลองการทรงตัวด้วยโครงข่ายประสาทเทียม	51
รูปที่ 4.17 แสดงวิธีการเพิ่มประสิทธิภาพของการทรงตัวโดยโครงข่ายประสาทเทียม	51
รูปที่ 4.18 แสดงวิธีการเพิ่มประสิทธิภาพการควบคุมอินเวอร์ตเพนคูลัม	52

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1

บทนำ

โครงข่ายประสาทเทียม(Artificial Neural Network) เป็นโครงข่ายเลียนแบบระบบประสาทจริงเพื่อนำไปใช้ในงานต่าง ๆ งานที่อธิบายด้วยระบบสมการได้และอธิบายด้วยระบบสมการไม่ได้ โดยระบบจำเรียนรู้และจำลองความสัมพันธ์ของตัวแปรด้วยวิธีต่าง ๆ

จากประโยชน์ดังกล่าว โครงข่ายประสาทเทียมจึงมักถูกนำไปใช้ในสมัยใหม่ได้เช่น การทำนายหุ้น หรือ การพยากรณ์อากาศ พอที่จะสรุปแยกออกเป็นหัวข้อต่าง ๆ ได้ดังนี้

เทคโนโลยีอวกาศ เช่น ยานยนต์ไร้คนขับประสิทธิภาพสูง, การจำลองเส้นทางการบิน, ระบบควบคุมการบิน, วิเคราะห์ข้อผิดพลาดของเครื่องยนต์

การเงินการธนาคาร เช่น เครื่องอ่านเช็ค, การทำนายค่าเงิน

การทหาร เช่น การปรับปรุงประสิทธิภาพอาวุธ การหาเป้าหมาย, การสอบสวน, การถอดรหัสสัญญาณ

อิเล็กทรอนิกส์ เช่น การเดาอนุกรม, การออกแบบ IC , การควบคุมการผลิต, การวิเคราะห์จุดเสีย, การจำลองแบบไม่เชิงเส้น

บันเทิง เช่น เกมต่าง ๆ , การสร้างภาพเสมือน

การแพทย์ เช่น การวิเคราะห์การกลายพันธุ์ของมะเร็ง, การวิเคราะห์สัญญาณหัวใจหุ่นยนต์ เช่น วิธีการยกของอัตโนมัติ, วิธีการมองเห็น

เสียง เช่น การจดจำ, การลดขนาดของเสียง, การแยกอักขระออกจากเสียง,

การพิมพ์โดยเสียง

โทรคมนาคม เช่น การบีบข้อมูลของสัญญาณ, การหาข้อมูลอัตโนมัติ เป็นต้น

จากการศึกษาและสังเกตวิธีการทรงตัวของสิ่งมีชีวิต สามารถจำลองและอธิบายได้ด้วยสมการต่าง ๆ แต่การที่สิ่งมีชีวิตนั้นจะทรงตัวอยู่ในตำแหน่งที่ไม่เสถียรนั้นทำให้เคลื่อนไหวได้รวดเร็วและเคลื่อนที่ได้ในทิศทางใด ๆ เช่นมนุษย์หรือลิง เป็นต้น จึงไม่สามารถการจำลองและอธิบายในขั้นต่ำ(Low Level Description)ของการทำงานได้ เช่นเราไม่ทราบว่า เราขึ้นได้อย่างไร สมองส่วนควบคุมการทรงตัวทำงานอย่างไร เราจึงรู้เพียงว่าเราต้องขึ้นเท่านั้นเมื่อเปรียบเทียบกับ โครงข่ายประสาทเทียมแล้วพบว่า เป็นโครงสร้างที่ต้องการการเรียนรู้เพื่อสะสมประสบการณ์ในหน่วยความจำ (Weight และ Bias)เช่นเดียวกับมนุษย์ เมื่อยังเด็ก เซลล์สมองยังไม่เจริญและกระตุ้นให้ทำงานมากนักเด็กจึงไม่สามารถเดินได้ทันทีที่ต้องอาศัยเวลาและประสบการณ์(Training) และต้นแบบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การเรียนรู้ของระบบโครงข่ายประสาทเทียมจะมีสองแบบคือระบบที่มีต้นแบบและระบบที่ไม่มีต้นแบบ

การเรียนรู้แบบมีต้นแบบจะเลียนแบบระบบการทรงตัวของทฤษฎีการควบคุมโดยวิธี PID (Proportional Integral Differential) และ LQR (Linear Quadratic Regulation)

ส่วนการเรียนรู้แบบไม่มีต้นแบบจะใช้วิธีการให้รางวัล และการยับยั้งภายในโครงข่ายเพื่อหาคำตอบที่ดีที่สุดใสถานการณ์นั้น

ข้อดีและข้อเสียของโครงข่ายทั้งสองแบบนี้มีความต่างกันเล็กน้อย เช่นความสามารถในการเรียนรู้ ความเร็วในการแก้ปัญหา และคุณสมบัติของการย้ายโครงข่าย (OffLine Learning) เป็นต้น แต่สิ่งที่สามารถเห็นได้ชัดคือการเรียนรู้แบบไม่มีต้นแบบนี้จะมีความซับซ้อนของวิธีการเรียนรู้มากกว่า เพราะภายในตัวโครงข่ายประสาทเทียมต้องตัดสินใจว่าเมื่อใดจะเรียนรู้ (Train Mode) หรือเมื่อใดจะทำงาน (Run Mode) โดยตัดสินใจจากผลการทำงานของอดีตเทียบกับปัจจุบันกำลังสอง จะได้เป็นฟังก์ชันค่าต่ำสุดของเป้าหมายที่ระบบโครงข่ายประสาทเทียมต้องการ

รายงานฉบับนี้แบ่งการศึกษาออกเป็นสองส่วนคือ การศึกษาส่วนการทำงานของโครงข่ายประสาทเทียมแบบมีต้นแบบ (Supervised Neural Network) ภายในปีการศึกษาที่ 1/2548 และการศึกษาการทำงานของโครงข่ายประสาทเทียมแบบไม่มีต้นแบบ (Unsupervised Neural Network) การทำงานของภายในภาคการศึกษาที่หนึ่ง ใช้โมเดลการทำงานของโครงข่ายประสาทเทียมแบบแบ็กพร็อพาคชัน (Backpropagation) ด้วยโครงสร้างแบบต่าง ๆ คือ โครงสร้างเรียงต่อเป็นลำดับ (Feed Forward Network) และ โครงสร้างแบบกึ่งเรียงต่อขนาบ (Casscade Forward Network) ทั้งยังศึกษาเรื่องความสัมพันธ์ของการใช้จำนวนประสาทเทียมจำนวนมากในโครงข่ายเดี่ยวซึ่งมีความผิดพลาดน้อย กับการใช้จำนวนโครงข่ายประสาทเทียมที่เรียนรู้ในเหตุการณ์แตกต่างกันซึ่งมีความผิดพลาดมากกว่า และจำนวนประสาทในโครงข่ายน้อยกว่า แต่ใช้หลาย ๆ โครงข่าย ทำให้ได้ข้อสรุปทางด้านพฤติกรรมการทำงานของโครงข่ายประสาทเทียม จำนวนชนิด และการเรียงต่อของประสาทเทียมแบบต่าง ๆ เพื่อเป็นแนวทางการศึกษา การทำงานของโครงข่ายประสาทเทียม ในภาคการศึกษาที่สอง การศึกษาต่อการทำงานของโครงข่ายการทรงตัวโดยการใช้โครงข่ายประสาทเทียมในภาคการศึกษานี้ จะศึกษาเรื่องการทำงานของโครงข่ายประสาทเทียมที่สามารถเรียนรู้ได้ด้วยตัวเอง โดยไม่ต้องมีต้นแบบ เลือกรูปแบบโครงข่ายลักษณะรวบรวมองค์ความรู้ตัวเอง (Self-Organizing Feature Maps) มาเพื่อเรียนรู้การทรงตัว โดยโครงข่ายดังกล่าว จะทำการปรับเปลี่ยนค่าน้ำหนัก (Weight) และค่าอุปทาน (Bias) ภายในโครงข่ายเมื่อผลของมูมีแนวโน้มดีขึ้น โดยการเพิ่มค่าน้ำหนักหรือลดค่าตัวน้ำหนัก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2

ทฤษฎี ของระบบการทรงตัว และโครงข่ายประสาทเทียม

รายงานฉบับนี้แบ่งภาคทฤษฎีออกเป็นสองช่วงคือ ภาคจำลองการทำงานและแสดงผลของ อินเวอร์ตเพนดูลัม และภาคโครงข่ายประสาทเทียม (Neural Network) ระบบ (Inverted Pendulum)

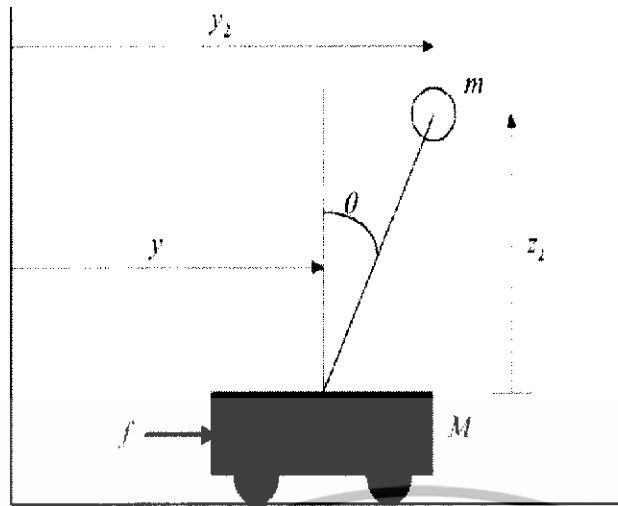
2.1 ทฤษฎี อินเวอร์ตเพนดูลัม (Inverted Pendulum)

ระบบ Inverted pendulum เป็นกรณีศึกษาของวิธีการทรงตัวซึ่งไม่เป็นระบบเชิงเส้น และยังมี ความเสถียรต่อเสถียรภาพของระบบ ในระบบจะประกอบด้วย ก้านถ่วงน้ำหนัก อยู่บน รางซึ่ง สามารถเคลื่อนที่ได้โดยสะดวกบนแกนระนาบ ในบทนี้จะกล่าวถึงสมการการเคลื่อนที่ของระบบ ซึ่งจะสร้างระบบจำลองบนคอมพิวเตอร์ จะใช้ แมทแล็บ 7.0 เวชวลซี พลัส พลัส 6.0 (Visaul C++ 6.0) และ เทอร์โบซีพลัสพลัส 3.0 จำลองระบบทั้งหมดลง บนระบบปฏิบัติการวินโดวส์ เอ็กพี (Windows Xp)

การทำงานบนแมทแล็บจะใช้กับโครงข่ายประสาทเทียมแบบมีต้นแบบเท่านั้น เพราะมี เครื่องมือในการสร้างโครงข่ายประสาทเทียมมากทำให้ลดเวลาในการศึกษาการทดลองมากแต่มีความ ยืดหยุ่นในการทำงานน้อยกว่า ภาษาซีมาค เนื่องจากข้อกำหนดโครงสร้าง และ การกำหนด ภาษาเชิงวัตถุ สามารถทำในภาษาซีได้ง่าย และเป็นโครงสร้างกว่าแมทแล็บมาก

ส่วนการทำงาน และวิธีการ ของโครงข่ายประสาทเทียมแบบไม่มีต้นแบบ นั้น เริ่มจาก การเขียนโปรแกรม ลงบนโปรแกรมวิชวลซี แต่ติดปัญหาเรื่องตำแหน่งหน่วยความจำจึงไม่สามารถ พัฒนาโครงข่ายประสาทเทียมได้ ทำได้เพียงแต่แสดงผลการทำงานของการทำงานของ อินเวอร์ตเพนดูลัมเท่านั้น จึงใช้ เทอโบซีพลัสพลัส 3.0 เพื่อแสดงผล และ จำลองการทำงานของ โครงข่ายประสาทเทียมแทนเนื่องจากการแสดงผลทำได้ง่ายกว่า

สมการการทำงานของคาร์ตของวัตถุบนฐาน โดยการอธิบายทฤษฎีพลังงานรวมแรงที่ เกิดขึ้นบนรถที่ใช้เลี้ยงก้านถ่วงน้ำหนักนั้นเมื่อแตกส่วนประกอบย่อยของแรงทั้งหมดแล้วจะได้ดัง รูปที่ 2.1



รูปที่ 2.1 แสดงแรงต่างๆ ที่เกิดขึ้นบน รถและองค์ประกอบที่จะต้องใช้พิจารณา

- เมื่อ M คือ มวลของรถ
 m คือ มวลก้อนถ่วงน้ำหนัก
 θ คือ มุมอันเนื่องมาจากแรง f
 f คือ แรงที่ทำให้เกิดองค์ประกอบต่าง ประกอบด้วย คือ มุมของก้อนถ่วงน้ำหนัก และ ระยะทาง x ที่รถเคลื่อนที่ไปได้

จากรูปที่ 2.1 เมื่อรวมแรงและคิดพลังงานรวมที่ รถแล้วจะ ได้พลังงานของแต่ละส่วน

$$T1 = \frac{1}{2} M \dot{y}^2 \quad (2.1)$$

และพลังงานรวมที่แกนเพนดูลัมคือ

$$T2 = \frac{1}{2} M \dot{y}_2^2 + \frac{1}{2} M \dot{z}_2^2 \quad (2.2)$$

สำหรับ Free body diagram ของ y_2 และ z_2 มีค่าเป็น

$$y_2 = y + l \sin \theta \quad (2.3)$$

$$z_2 = l \cos \theta \quad (2.4)$$

$$\dot{z}_2 = -l \dot{\theta} \sin \theta \quad (2.5)$$

$$z_3 = l \cos \theta \quad (2.6)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ดังนั้นพลังงานรวมคือ

$$\begin{aligned} T &= T_1 + T_2 \\ &= \frac{1}{2} (M \dot{y}^2 + m(\dot{y}_2^2 + \dot{z}_2^2)) \end{aligned} \quad (2.7)$$

พลังงานศักย์ V ของระบบจะเก็บไว้ที่ ก้านPendulum ดังนั้น

$$v = mgz_2 = mgl \cos \theta \quad (2.8)$$

จาก LaGrange Equation คือ

$$L = T - V \quad (2.9)$$

$$= \frac{1}{2} (M + m) \dot{y}^2 + ml \cos \theta \dot{y} \dot{\theta} + \frac{1}{2} ml^2 \dot{\theta}^2 - mgl \cos \theta \quad (2.10)$$

สำหรับตัวแปร State Space Variable ของระบบมีสองตัว คือ y และ θ ดังนั้นจะได้ สมการของ LaGrange Equation คือ

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{y}} - \frac{\partial L}{\partial y} = 0 \quad (2.11)$$

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = 0 \quad (2.12)$$

แต่

$$\frac{\partial L}{\partial y} = (M + m) \dot{y} + ml \cos \theta \dot{\theta} \quad (2.13)$$

$$\frac{\partial L}{\partial y} = 0 \quad (2.14)$$

$$\frac{\partial L}{\partial \dot{y}} = ml \cos \theta \dot{y} + ml^2 \dot{\theta} \quad (2.15)$$

$$\frac{\partial L}{\partial \dot{y}} = ml \sin \theta \quad (2.16)$$

จะได้สมการของระบบซึ่งเป็น Nonlinear Dynamic Equation คือ

$$(M + m) \ddot{y} + ml \cos \theta \ddot{\theta} - ml \dot{\theta}^2 \sin \theta = f \quad (2.17)$$

$$ml \cos \theta \ddot{y} - ml \sin \theta \dot{y} \dot{\theta} - ml^2 \ddot{\theta} - mgl \sin \theta = 0 \quad (2.18)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สมการที่ 2.17 และ สมการที่ 2.18 สามารถประมาณให้เป็นสมการเชิงเส้นได้ โดยการประมาณให้ มุม θ มีค่าเข้าใกล้ 0 มาก ๆ ดังนั้นจะได้ $\cos\theta = 1$ และ $\sin\theta = 0$ นั่นคือ ประมาณให้ ก้านของ Pendulum มีการเปลี่ยนแปลงมุนน้อยมากจะได้สมการ Linear Pendulum Equation คือ

$$\ddot{y} = \frac{f}{M} - \frac{mg\theta}{M} \quad (2.19)$$

$$\ddot{\theta} = -\frac{f}{Ml} + \left(\frac{M+m}{Ml}\right)g\theta \quad (2.20)$$

เนื่องจากการจำลองการเคลื่อนที่ที่ต้องสังเกตุดลอกการทำงานทุกช่วงของการเคลื่อนที่ดังนั้น ไม่สามารถใช้ในการประมาณเชิงเส้นได้จึงต้องประมาณสมการไม่เชิงเส้นจากสมการที่ 2.17 และ 2.18 ดังนี้

$$\ddot{\theta}_i = g \sin \theta_i + \cos \theta_i \left[\frac{-F - Mpl \dot{\theta}^2 \sin \theta_i}{l} \right] \quad (2.21)$$

$$\ddot{x}_i = \frac{F + Mpl[\dot{\theta}^2 \sin \theta_i - \ddot{\theta} \cos \theta_i]}{Mc + Mp} \quad (2.22)$$

ความเร่งเชิงมุมและความเร่งเชิงเส้นจะถูกนำไปหา ความเร็วเชิงมุม และความเร็วเชิงเส้นแบบ อินทิเกรตจำกัน โดยวิธีการของออยเลอร์จะได้ค่าของความเร็วเชิงเส้น ความเร็วเชิงมุม ตำแหน่งเชิงเส้น และ มุม ด้วยวิธีดังนี้

$$x = x + \left(\frac{T}{\text{STEPS}}\right) \times \dot{x} \quad (2.23)$$

$$\dot{x} = \dot{x} + \left(\frac{T}{\text{STEPS}}\right) \times \ddot{x} \quad (2.24)$$

$$\theta = \theta + \left(\frac{T}{\text{STEPS}}\right) \times \dot{\theta} \quad (2.25)$$

$$\dot{\theta} = \dot{\theta} + \left(\frac{T}{\text{STEPS}}\right) \times \ddot{\theta} \quad (2.26)$$

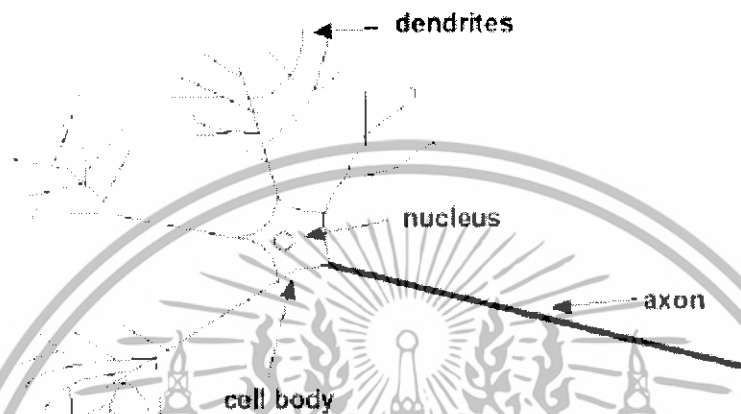
เมื่อ T คือ น้าหนักจำนวนน้อยๆ

STEP คือ ช่วงคาบทั้งหมดของการอินทิเกรต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

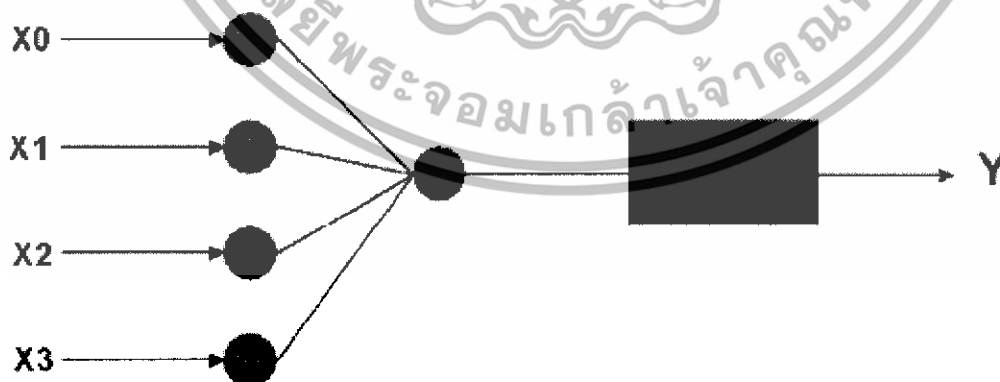
2.2 โครงข่ายประสาทเทียม (Artificial Neural Network ANN)

ระบบโครงข่ายประสาทเทียม เป็นการสร้างสมการเพื่อเรียนรู้ จำลอง และจำแนก ข้อมูลต่าง ๆ ออกจากกัน โดยการประมาณความสัมพันธ์ของตัวแปร และเป้าหมาย โครงข่ายประสาทเทียม เลียนแบบมาจากโครงข่ายประสาทของสิ่งมีชีวิต ดังนั้นจะประกอบด้วย โครงสร้างต่าง ๆ คล้ายกับ เซลล์ประสาท ซึ่งประกอบด้วย ตัวเซลล์ (Cell Body) เดรนที่ไครต์(Dendrite) และ แอกซอน (Axon) ดังรูปที่ 2.2



รูปที่ 2.2 แสดง เซลล์ประสาทหนึ่งเซลล์

เดรนที่ไครต์จะทำหน้าที่รับสัญญาณจาก แอกซอนของเซลล์ประสาทอื่น ๆ ซึ่งสัญญาณจาก เดรนที่ไครต์จะรับสัญญาณจากหลายๆ เซลล์พร้อม ๆ กัน แล้วส่งออกไปเพียงทางเดียวที่ แอกซอน สายโครงข่ายแอกซอน จะเกี่ยวพันกันหลาย ๆ เซลล์ โดยจะเชื่อมต่อกับเซลล์ประสาทอื่น ๆ ทางไซแนปส์ (Synapse) ซึ่งจะเป็นการส่งสัญญาณโดยการกระตุ้นปฏิกิริยาเคมีโดยสัญญาณไฟฟ้าอ่อน ๆ โครงสร้างของโครงข่ายประสาทเทียมง่าย ๆ ดังรูปที่ 2.3



รูปที่ 2.3 แสดง โครงสร้างของโครงข่ายประสาทเทียมอย่างง่าย ๆ

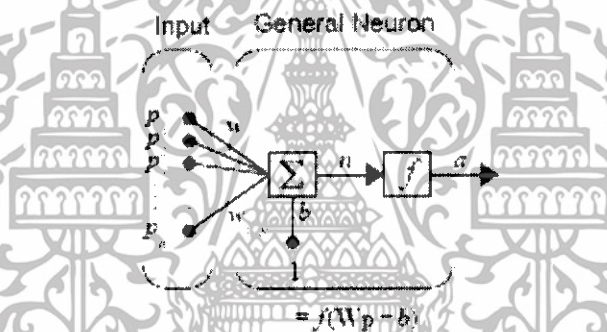
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อได้เปรียบของ โครงข่ายประสาทเทียม

- 1 ข้อดีที่สุดคือ สามารถสอน โครงข่ายประสาทเทียมให้เลียนแบบฟังก์ชันเฉพาะอย่างได้ โดยการปรับเปลี่ยน Weight และ Bias เพื่อใช้กับการประมาณค่าของฟังก์ชันนั้น
- 2 โครงข่ายประสาทเทียมประกอบด้วยกระบวนการประมวลผลแบบขนาน(Parallel Processing) การประมวลผลแบบนี้จะลดเวลาในการคำนวณเมื่อเทียบกับการคำนวณแบบตามขั้นตอน (Sequential Processing)
- 3 โครงข่ายประสาทเทียมจะมีการจำ ซึ่งใช้สำหรับการคำนวณ Weight และ Bias โครงข่ายประสาทเทียมสามารถเรียนรู้โดยมีข้อมูลจำกัดและค่อยๆป้อนเข้าไปให้โครงข่ายประสาทเทียมเรียนรู้ได้ เมื่อการเรียนรู้เสร็จสิ้นแล้วสามารถนำไปใช้กับระบบอื่นได้อีกโดยไม่ต้องเรียนรู้ใหม่(Trained Offline)ข้อดีส่วนนี้จะนำไปใช้ควบคุมการทรงตัว

รูปแบบของการเรียนรู้แบบ แบทหรือพทาถัน (Backpropagation)

โครงสร้างของโครงข่ายประสาทเทียมชนิดนี้มีโครงสร้างของฟังก์ชันดังรูปที่ 2.4



รูปที่ 2.4 แสดงโครงสร้างของฟังก์ชันของประสาทเทียมแบบBackpropagation

โครงสร้างของ Backpropagation สร้างโดยการปรับปรุงโครงสร้าง การเรียนรู้แบบ Widrow-Hoff ให้ใช้สำหรับโครงข่ายหลายชั้น และ สำหรับฟังก์ชันถ่ายโอน ประเภทอนุพันธ์ของสมการ ไม่เชิงเส้น(Nonlinear differentiable transfer function) เวกเตอร์ข้อมูล และการตอบสนองต่อผลของข้อมูลจะถูกใช้สอนจนกระทั่งโครงข่ายประสาทสามารถ ประมาณค่าของฟังก์ชันได้โดยการให้สมาชิกของ อินพุตเทียบกับเอาพุต หรือ จัดหมู่ของอินพุตเวกเตอร์โดยการกำหนดเองก็ได้ จำนวนโครงสร้างโครงข่ายกับไบอัส(Bias) เช่น ชั้นซิกมอยด์และ ลิเนียร์ เอาต์พุต มีผลต่อการประมาณค่าของฟังก์ชันในจำนวนที่จำกัด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การแพร่กลับของสัญญาณเอาต์พุตซึ่งเป็นหัวใจหลักของการทำงานของโครงข่ายชนิดนี้จะแพร่กลับโดยใช้อนุพันธ์ของทรานเฟอร์ฟังก์ชันของเลเยอร์นั้น ๆ เป็นตัวกำหนดเช่นถ้า ใช้เอาต์พุตเป็น logsigmoid มีทรานเฟอร์ฟังก์ชันเป็น

$$f'(1) = \frac{1}{1+e^{-n}} \quad (2.27)$$

เมื่อแพร่สัญญาณกลับจะเป็น

$$\frac{d}{dn} f'(1) = \frac{d}{dn} \left[\frac{1}{1+e^{-n}} \right] \quad (2.28)$$

$$f^{-1}(1) = \frac{e^{-n}}{(1+e^{-n})^2} \quad (2.29)$$

โดยอนุพันธ์ของฟังก์ชันจะเป็นเอาต์พุตของฟังก์ชันถัด ๆ ไป ซึ่งจะซ้ำ ๆ กันไปเรื่อย ๆ จนกว่าจะแพร่กลับหมดครบทุก ๆ ฟังก์ชัน

เมื่อได้ค่าของการแพร่กลับแล้วจะนำไปปรับน้ำหนัก (Weight) ใหม่ด้วยฟังก์ชันเดิมของเลเยอร์นั้น โดยมี ค่าเวกเตอร์การปรับค่าของฟังก์ชัน(s) นั้น ตามลำดับเช่น

$$s^2 = -2F^2(n^2)(t-a) \quad (2.30)$$

$$s^1 = F^1(n^1)(w^2)^T s^2 \quad (2.31)$$

ค่าของ S จะบอกประสิทธิภาพของการปรับน้ำหนัก (weight) ของเลเยอร์นั้น ๆ เมื่อได้ ค่า S ออกมาจะนำไปปรับน้ำหนัก(weight)และค่าอุปทาน(bias)ของประสาทเทียมแต่ละตัวเช่น

$$w^2(1) = w^2(0) - \alpha s^2 (a^1)^T \quad (2.32)$$

$$b^2(1) = b^2(0) - \alpha s^2 \quad (2.33)$$

$$w^1(1) = w^1(0) - \alpha s^1 (a^0)^T \quad (2.34)$$

$$b^1(1) = b^1(0) - \alpha s^1 \quad (2.34)$$

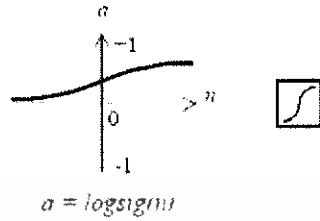
โครงสร้างพื้นฐานของ Backpropagation จะใช้วิธีการเค้นเดสเซนต์ (Gradient Descent Algorithm:gd) แทน Widrow-Hoff ในโครงข่าย weight ถูกคำนวณกลับตลอดในฟังก์ชัน Gradient Descent แล้วจะย้อนกลับด้วยวิธี Gradient Descent อย่างเดิมเพื่อคำนวณโครงข่ายไม่เชิงเส้นหลาย ๆ ชั้น ค่าของตัวแปรบางตัวคำนวณโดยพื้นฐานของปรับปรุงค่าความผิดพลาดปกติ เช่น คอนจูเกชันเค้นเดสเซน และ วิธีของนิวตัน

โดยปกติแล้วโครงข่ายประสาทเทียมสามารถให้ผลลัพธ์ซึ่งไม่เคยใช้สอนมาก่อนได้ อินพุตจะเหนี่ยวนำผลลัพธ์ให้คล้ายกับผลลัพธ์ที่ได้เคยใช้สอนมาก่อน ด้วยคุณสมบัตินี้จึงสามารถใช้ข้อมูลที่มีจำนวนจำกัดในการสอนโครงข่ายประสาทเทียมเพื่อใช้ในหลาย ๆ งานได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3 แบบจำลองของโครงข่ายประสาทเทียม Backpropagation

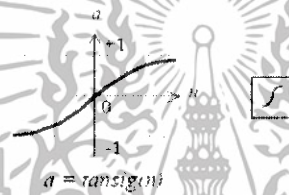
แต่ละอินพุตของโครงข่ายประสาทเทียมจะมีค่าถ่วงน้ำหนัก(Weighted)ด้วยค่า w ผลรวมของ ค่าถ่วงน้ำหนัก และ Bias จากอินพุตผ่านไบแอส b จนถึงฟังก์ชัน โอนถ่าย f โดยฟังก์ชัน โอนถ่ายสามารถเปลี่ยนเป็นอะไรก็ได้ตามความเหมาะสมของผลลัพธ์ (Output) ตามรูปที่ 2.5



Log-Sigmoid Transfer Function

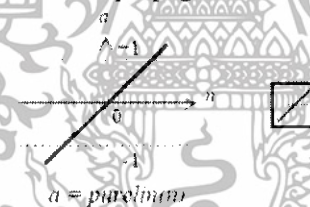
รูปที่ 2.5 แสดง ฟังก์ชัน โอนถ่าย log sigmoid

ฟังก์ชัน logsig ให้เอาพุตมีค่าระหว่าง 0 ถึง 1 เมื่ออินพุตมีค่าตั้งแต่ 0 จนถึงอนันต์ในทางกลับกัน โครงข่ายประสาทหลายๆ ชั้น ใช้ฟังก์ชัน tansig ก็ได้



รูปที่ 2.6 แสดงฟังก์ชัน โอนถ่าย tan sigmoid

ใช้ฟังก์ชันเชิงเส้น purelin กับ โครงข่าย Backpropagation จะทำให้ได้ค่าจริงไม่มีขอบเขต



รูปที่ 2.7 แสดงฟังก์ชัน โอนถ่ายเชิงเส้น

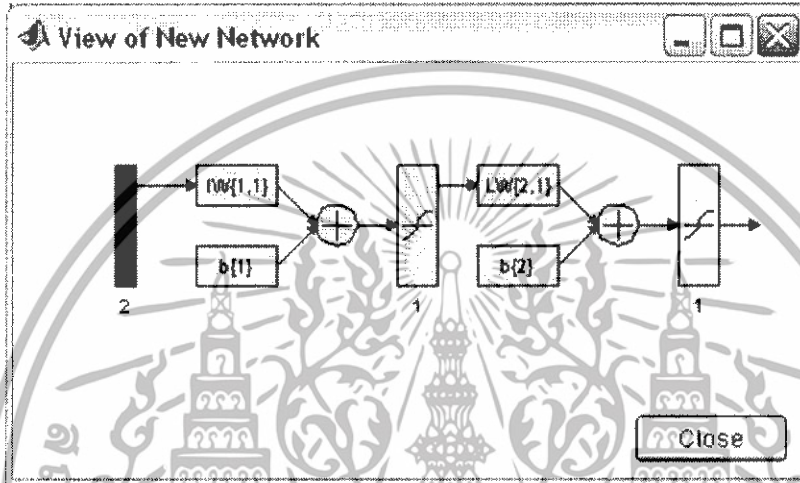
การเลือกใช้ฟังก์ชัน โอนถ่าย tansig และ logsig ให้ค่าผลลัพธ์ไม่เกิน 1 ต่างจาก ฟังก์ชัน purelin ซึ่งให้ค่าผลลัพธ์เป็นค่าใด ๆ

2.4 การประมาณค่าฟังก์ชัน

ในระบบควบคุมจะใช้การประมาณค่าฟังก์ชันเพื่อประมาณค่าของโครงสร้างโครงข่ายประสาทเทียมจนระบบควบคุมย้อนกลับต้องการหาฟังก์ชันเพื่อประมาณค่าโดยการหาความสัมพันธ์ระหว่างค่าของผลลัพธ์และค่าอินพุตในโครงข่ายสองชั้น เช่นในโครงข่ายชั้นแรกใช้ฟังก์ชันโอนถ่ายเป็น Log-Sigmoid และ ฟังก์ชันโอนถ่ายในชั้นที่สองเป็นฟังก์ชันเชิงเส้น(Linear)ดังสมการ

$$f^1(1) = \frac{1}{1+e^{-n}} \quad (2.35) \text{ และ}$$

$$f^2(n) = n \dots \quad (2.36)$$



รูปที่ 2.8 แสดงตัวอย่างของ Function Approximation Network

เมื่อพิจารณาค่าปกติของ Weight และ Bias ของโครงข่ายจะได้

$$n_2^1 = w_{2,1}^1 p + b_2^1 = 0 \Rightarrow p = \frac{b_2^1}{w_{2,1}^1} = \frac{-10}{-10} = 1$$

$$w_{1,1}^1 = 10, w_{2,1}^1 = 10, b_1^1 = -10, b_2^1 = 10$$

$$w_{1,1}^2 = 1, w_{1,2}^2 = 1 \quad (2.37)$$

โครงข่ายดังรูปที่ 2.15 จะตอบสนอง ตัวแปรต่าง ๆ ตามกลุ่มสมการที่ 2.51 เป็นความสัมพันธ์ ระหว่างเอาพุต a^2 ต่ออินพุต p ซึ่งมีค่าอยู่ในช่วง $[-2, 2]$ ผลตอบสนองประกอบด้วยสองขั้นตอนคือ การตอบสนองต่อฟังก์ชัน log-sigmoid ในโครงข่ายประสาทเทียมชั้นแรก การปรับค่าของตัวแปรต่าง ๆ ของโครงข่ายจะเปลี่ยนรูปร่างและตำแหน่ง ของแต่ละขั้นตอนตำแหน่งกลางของขั้นตอนเมื่อให้ค่าของโครงข่ายเป็น 0

$$n_1^1 = w_{1,1}^1 p + b_1^1 = 0 \Rightarrow p = \frac{b_1^1}{w_{1,1}^1} = \frac{-10}{-10} = 1 \quad (2.38)$$

$$n_2^1 = w_{2,1}^1 p + b_2^1 = 0 \Rightarrow p = \frac{b_2^1}{w_{2,1}^1} = \frac{-10}{-10} = -1 \quad (2.39)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.5 วิธีสุ่มการกระจายค่าปกติ

การกระจายค่าปกติของตัวแปร x กับค่าเฉลี่ย μ และ ค่าเบี่ยงเบนมาตรฐาน σ^2 เป็นการกระจายของความเป็นไปได้ในเชิงสถิติ



รูปที่ 2.9 แสดงการกระจายตัวของค่าความน่าจะเป็น

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (2.40)$$

เมื่อค่าของ x ภายในโดเมน $(-\infty, \infty)$

การสุ่มค่าของน้ำหนักและค่าอุปทานใช้สำหรับการสุ่มค่าของแรงให้มีขนาดปกติไม่เกิน 10 นิวตัน

2.6 โครงสร้างแบบไม่มีต้นแบบ (Unsupervised Network)

2.6.1 โครงสร้างรวมแบบง่าย (Simple Associative Network)

การกระตุ้นและการยิงยั้งค่าต่าง ๆ ที่ออกมาจากโครงข่ายประสาท โดยการ ให้แทนค่าลงภายในโครงข่ายเมื่อพิจารณาโครงข่ายดังนี้

ค่าเอาต์พุต a ที่ออกมาจากฟังก์ชัน

$$a = \text{hardlim}(wp + b) \quad (2.41)$$

$$a = \text{hardlim}(wp - 0.5) \quad (2.42)$$

จะมีสองค่า คือ 0 และ 1 แทนระบบ มีการกระตุ้น และ ไม่มีการกระตุ้น

$$a = \begin{cases} 1, \text{stimulus} \\ 0, \text{noresponse} \end{cases} \quad (2.43)$$

2.6.2 กฎของ Kohonen (Kohonen Rule)

โครงสร้างอีกแบบของโครงข่ายประสาทเทียม ซึ่งมีการรวมเอาค่าจากสิ่งแวดล้อมซึ่งคล้ายกับโครงสร้างที่เป็นแบบ Instar ซึ่งมีสมการดังนี้

$$w_i(q) = w_i(q-1) + \alpha(p(q) - w_i(q-1)) \quad , i \in X(q) \quad (2.44)$$

กฎของ Kohonen สามารถปรับน้ำหนัก(weight) ตาม เวกเตอร์อินพุตให้เข้ากันกับการเรียนรู้ของโครงข่ายนั้น ซึ่งต่างจาก กฎ Instar การเรียนรู้จะไม่เกิดจากเอาพุทของโครงข่าย เมื่อมีการนำ Instar เข้าไปใช้กับโครงข่ายนั้น จะให้ค่าออกมาสองค่า คือ 0 หรือ 1 เท่านั้น หลังจากนั้น Kohonen จะทำการเปลี่ยนค่าที่ Instar กำหนด $X(q)$ ไปได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อดีของ Kohonen คือ สามารถเปลี่ยนเงื่อนไขได้ ซึ่งเป็นประโยชน์ในโครงข่ายประสาทเทียมแบบ Self-Organized Feature Map

2.7 Self-Organizing Feature Map

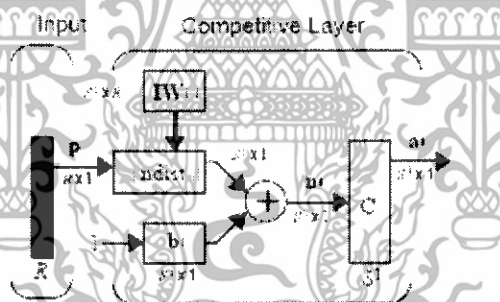
โครงสร้างของการจำลองพฤติกรรมของสิ่งมีชีวิตโดยไม่มีต้นแบบ ทำได้โดยประยุกต์การทำงานอย่างไม่เป็นเชิงเส้นด้วยวิธีการป้อนกลับแบบ on-center / off-surround จากกฎของ Kohonen ได้กล่าวถึงวิธีการเปลี่ยนเงื่อนไขเพื่อการเลือกโครงข่ายที่ดีที่สุด (Winning Neural) เริ่มต้นโดยการตัดสินใจเลือก โครงข่ายที่ดีที่สุด(Winner) และยับยั้งโครงข่ายรอบ ๆ ที่ไม่ถูกเลือกดังสมการ

$$w_i(q) = w_i(q-1) + \alpha(p(q) - w_i(q-1)) \tag{2.45}$$

$$= (1 - \alpha)w_i(q-1) + \alpha p(q) \tag{2.46}$$

2.7.1 Competitive Layer

โครงข่ายชนิดนี้จะทำการกระตุ้นตัวเองและยับยั้งโครงข่ายทุกตัวที่อยู่รอบ ๆ มันเราเขียนเป็นสัญลักษณ์อย่างง่ายคือ



รูปที่ 2.10 แสดงโครงข่าย แบบ Competitive Network

$$a_i = \text{compet}(n_i) \tag{2.47}$$

การทำงานของโครงข่ายชนิดนี้โดยการกำหนดค่าดัชนี(Index)ของโครงข่ายประสาทเทียมตัวที่มี Input ใหญ่ที่สุด และให้นิวรอนตัวอื่น มีค่า Output เป็น 0

$$a_i = \begin{cases} 1, & i = 1^* \\ 0, & i \neq 1^* \end{cases} \text{เมื่อ } n_{1^*} \geq n_i, \forall i, \text{ and } i^* \leq i, \forall n_i = n_{1^*} \tag{2.48}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7.2 Neighbourhood

เป็นประสาทเทียมตัวรอบ ๆ ตัวที่ถูกรับเลือก โดยจะถูกปรับค่าให้ต่ำลงกว่าตัวที่ได้รับเลือกประสาทเทียมตัวที่ไม่ได้รับเลือกที่อยู่ใกล้กับตัวที่ได้รับเลือกจะได้รับผลกระทบของตัวที่ได้รับเลือกมากกว่าตัวที่อยู่ไกลออกไปซึ่งประสาทเทียมแต่ละตัวซึ่งไม่ได้รับเลือกจะสัมพันธ์กับค่า λ ดังนี้

$$\lambda = e^{-\frac{\sqrt{(X_{\text{Firing}} - X_i)^2 + (Y_{\text{Firing}} - Y_i)^2}}{2\sigma}} \quad (2.49)$$

น้ำหนักของชั้น Kohonen จะต้องปรับค่าตามฟังก์ชัน ซึ่งชั้นอัตราการเรียนรู้ α ส่วนหนึ่ง

$$w(q) = \alpha \lambda, w(q-1)(t-a) \quad (2.50)$$



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3 การออกแบบโปรแกรม

บทนี้กล่าวถึงการออกแบบโปรแกรมของสองส่วนคือ การออกแบบการควบคุมอินเวอร์ตเพนดูลัมบนเมทเพล สำหรับการเรียนรู้แบบมีต้นแบบ(Supervised Neural Network) และ การออกแบบโปรแกรมภาษาซี สำหรับการเรียนรู้แบบไม่มีต้นแบบ(Unsupervised Neural Network)

3.1 การออกแบบแบบจำลอง Inverted Pendulum

การสร้างแบบจำลองโดยอ้างอิงจากบทที่ 2 โดยการนำสมการต่างๆ มาเขียนเป็น Block Diagram แล้วจึงทดสอบแบบจำลองเพื่อหาค่าของตัวแปรต่าง

แบบจำลองชุด Inverted Pendulum ประกอบด้วยขั้นตอนดังนี้ คือ

1. เปรียบเทียบสมการของ PID และ สมการของ Inverted Pendulum เพื่อแยกส่วนการทำงานของระบบ โดยจะประกอบด้วยแบบจำลองสามส่วนใหญ่ๆ คือ

1.1 การหาค่าตัวแปรของระบบ

1.2.การแปลงข้อมูลที่ได้จากสมการของ Inverted Pendulum

1.3.การแสดงผล

2. ทดสอบระบบเพื่อเก็บวิธีการเคลื่อนที่ของ Cart และ Pendulum โดยจะแบ่งข้อมูลออกเป็น 5 ชนิดข้อมูลคือ

2.1 ตำแหน่งต่าง ๆ ที่ใช้แรงตามแนวแกน y บังคับให้ Cart ไปตำแหน่งนั้น เก็บในตัวแปร x

2.2 ความเร็วของ Cart ขณะนั้น เก็บเป็นตัวแปร \dot{x}

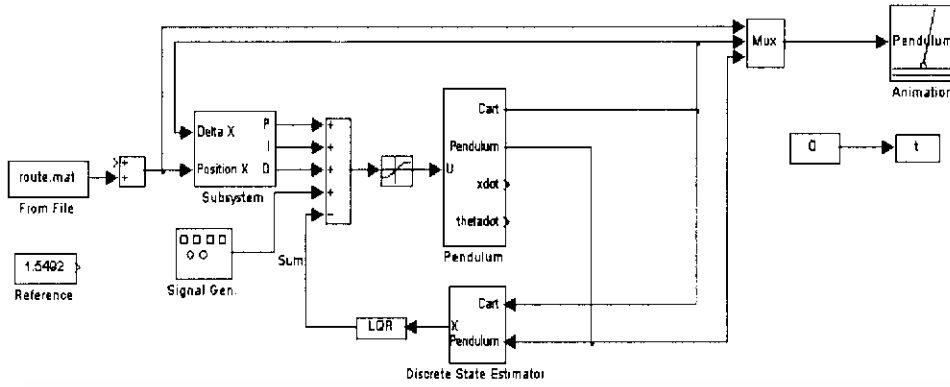
2.3 มุมของก้าน Pendulum เทียบกับแนวแกนตั้งขณะนั้น เก็บเป็นตัวแปร θ

2.4 ความเร็วเชิงมุมของก้าน Pendulum เก็บเป็นตัวแปร $\dot{\theta}$

ตัวแปรทั้งสี่จะมีคู่ของตัวแปรนั้นคือ มีตัวแปรอยู่สองชนิด คือระยะทาง ความเร็ว เป็นคู่ของตัวแปร และ มุมของก้านPendulum ความเร็วเชิงมุมเป็นตัวแปรอีกคู่หนึ่ง ตัวแปรทั้งคู่นี้

ความสัมพันธ์กันในเชิงทางคณิตศาสตร์ซึ่งเป็นอนุพันธ์ของอีกตัวแปรหนึ่งตามสมการที่ 2.17 และ 2.18 ดังรูปที่ 3.1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.1 แสดง Block Diagram ของระบบทั้งหมด

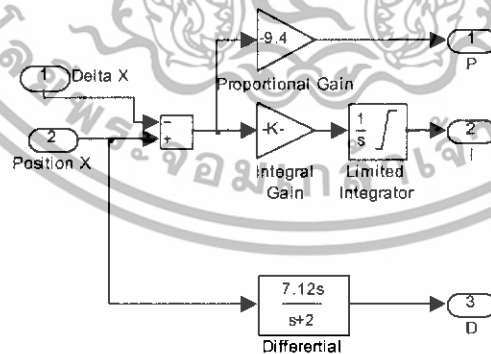
จากรูปเห็นว่าจะประกอบด้วยการทำงานทั้งหมดสามขั้นตอน คือ รับค่าจากเส้นทางการเดินทางของ Cart จาก Route.mat เป็นทางเดินมาตรฐานตลอดการทดลอง หลังจากนั้นจึงส่งค่าเป็นผลต่างระหว่างของตำแหน่งของ Cart ขณะนั้นกับ ตำแหน่งที่ Cart ต้องเคลื่อนที่ไปให้กับ Block ของ PID ซึ่งจะเป็นการคำนวณแรงที่ต้องเคลื่อนที่ Cart ไปยังตำแหน่งนั้น ๆ โดยให้เกิดความผิดพลาดน้อยที่สุดมี โดยการเคลื่อนที่ของแนวแกนราบนั้น จะมีค่าอยู่ในช่วง (-10, 10) โดยเขียนเป็นฟังก์ชันได้

$$K_p = -9.4\Delta X \tag{3.1}$$

$$K_i = \int 0.06\Delta X dt \tag{3.2}$$

$$K_d = \frac{7.12\Delta X}{\Delta X + 2} \tag{3.3}$$

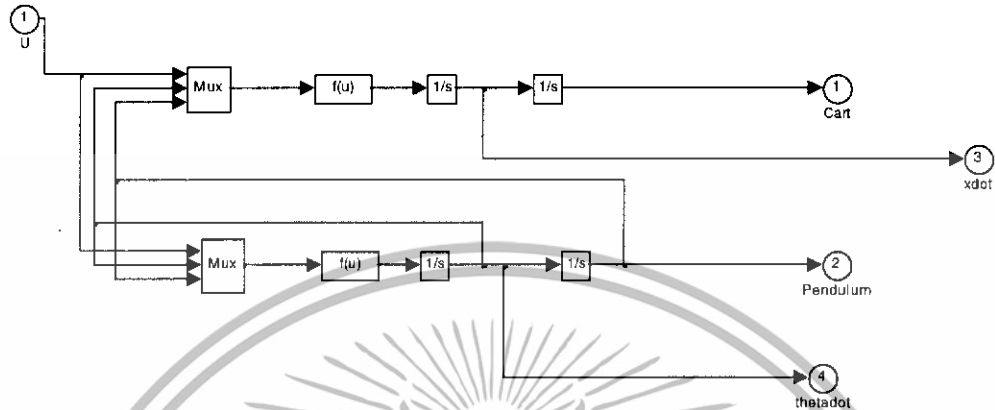
นำมาเขียนเป็น ฟังก์ชันย่อย(Sub System) ได้ดังรูปที่ 3.2



รูปที่ 3.2 แสดงฟังก์ชันย่อยของฟังก์ชันPID

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อได้ค่าจากฟังก์ชัน PID แล้วจะส่งไปให้กับฟังก์ชัน Inverted Pendulum ซึ่งค่าต่าง ๆ จะถูกทำให้เป็นค่าปกติ(Normalized)อยู่ในช่วง (-1, 1) ค่าที่ได้นี้เรียกว่าแรงแทนด้วยตัวแปร f ทั้งนี้แรง f เป็นสาเหตุของการเกิดความไม่เสถียร เมื่อได้ค่าแรง f มาแล้วฟังก์ชัน Inverted Pendulum จะคำนวณค่าของตัวแปรต่าง ๆ ออกมา สี่ค่าดังรูปที่ 3.3



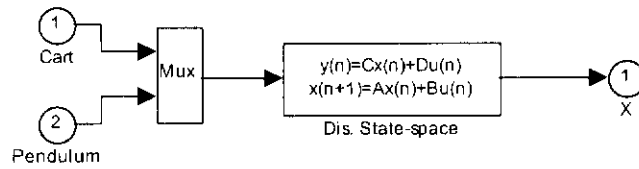
รูปที่ 3.3 แสดงฟังก์ชันย่อยของ ฟังก์ชัน Inverted Pendulum

การทำงานจะเริ่มจากการรับค่าแรงมาแล้วนำมาแทนค่าลงในสมการที่ 2.17 และ 2.18 ซึ่งประมาณค่าใหม่ในรูปของตัวแปรระยะทาง x , \dot{x} , θ และ $\dot{\theta}$ ดังนี้

$$x_{cart}(f, \theta, \dot{\theta}) = \iint \left(\frac{f - g \sin \theta \cos \theta + l \dot{\theta}^2 \sin \theta}{\frac{M}{m} + \theta^2} \right) \quad (3.4)$$

$$\theta_{Pendulum}(f, \theta, \dot{\theta}) = \iint \left(\frac{f \cos \theta + (M + m)g \sin \theta - l \dot{\theta}^2 \sin \theta \cos \theta}{l \left(\frac{M}{m} + \sin^2 \theta \right)} \right) \quad (3.5)$$

นอกจากนี้ยังมีส่วนของครื่องกลับแบบลบตำแหน่งของมุมของ Pendulum อีกคือ ฟังก์ชัน discrete State Estimator และ LQR(Linear Quadratic Regulator) โดย ฟังก์ชัน LQR จะทำให้ค่าของฟังก์ชันมีค่าที่ดีที่สุด



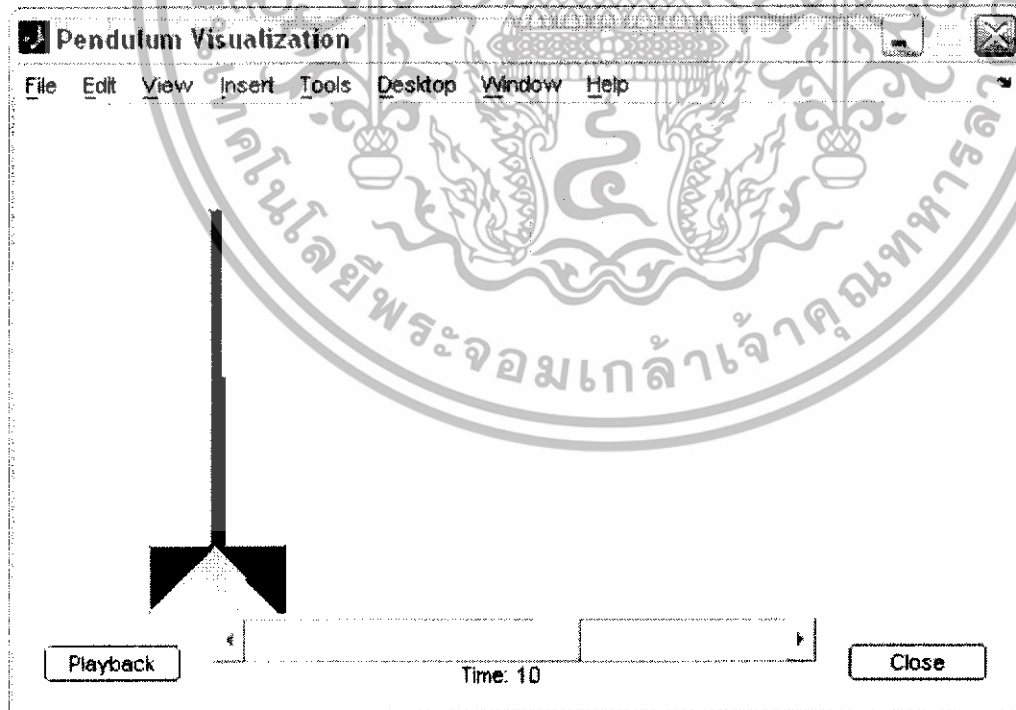
รูปที่ 3.4 แสดงฟังก์ชันรักษามุมคงที่

ส่วนของการแสดงผลใช้การแสดงผลเสมือนจริง(Visual Reality) สำเร็จรูปซึ่งเป็นฟังก์ชันสำเร็จรูปตัวแปรต่าง ๆ จะถูกมัลติเพล็กซ์ ก่อนส่งผ่านให้กับ ฟังก์ชันการแสดงผล ดังรูปที่ 3.5



รูปที่ 3.5 แสดงวิธีการส่งค่าให้กับฟังก์ชันสำเร็จรูปของการแสดงผล

การแสดงผลดังกล่าวสามารถดูผลของการเคลื่อนที่รวมทั้งการเปลี่ยนแปลงค่าต่าง ๆ ได้ทันทีโดยมีหน้าต่างการแสดงผลดังรูปที่ 3.6

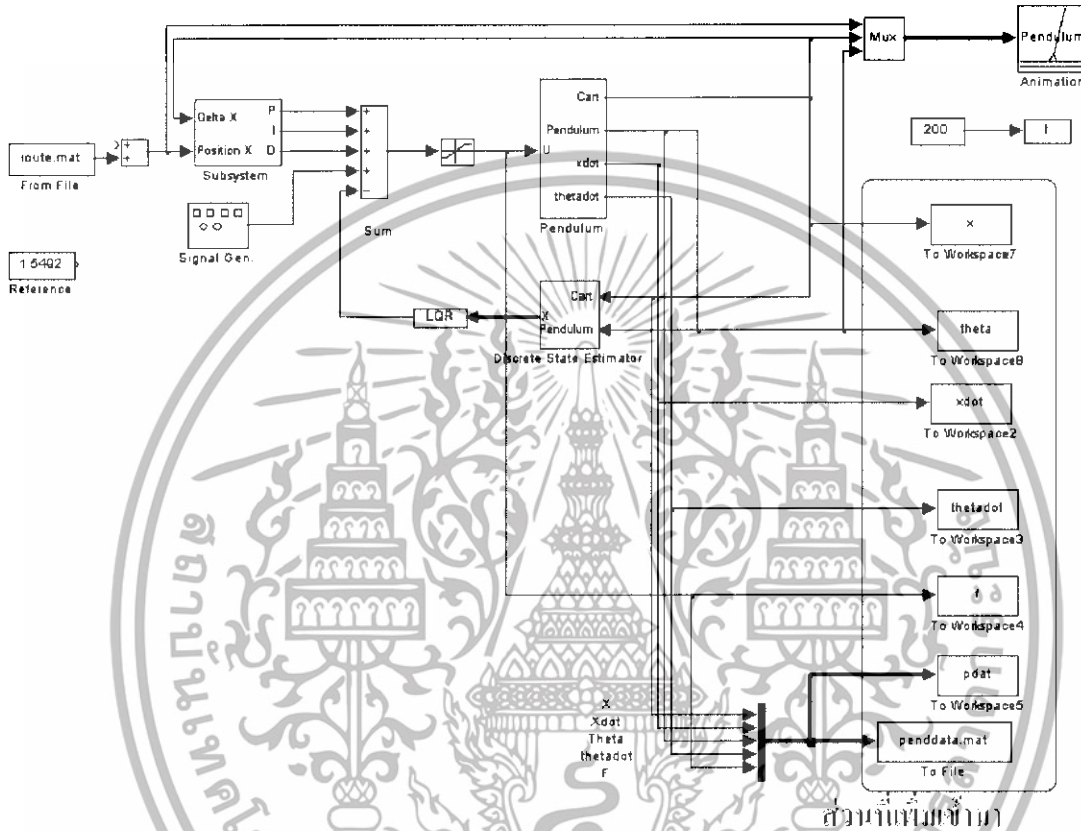


รูปที่ 3.6 แสดงหน้าต่างของการแสดงผลเสมือนจริง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2 วิธีการเก็บข้อมูล

การเก็บข้อมูลมีสองแบบคือ การเก็บลงตัวแปรชั่วคราว และ เก็บลง ไฟล์ การเก็บข้อมูลทั้งสองแบบสามารถเปลี่ยนกลับไปกลับมาได้ ขึ้นอยู่กับความสะดวกเช่นการเคลื่อนที่ไปยังตำแหน่งต่าง ๆ ต้องการค่าที่เหมือนเดิมตลอดการทดลองจึงเก็บลงไฟล์ แต่การคำนวณค่าต่าง ๆ ที่ได้จากแบบจำลองจะเก็บลงบนตัวแปร เช่นการคำนวณค่าของมุมให้อยู่ในค่ามาตรฐาน โดยจะเพิ่มแบบจำลองฟังก์ชันสำเร็จรูปลงในแบบจำลองหลักดังรูปที่ 3.7



รูปที่ 3.7 แสดงวิธีการเก็บข้อมูลลงในไฟล์และลงบนตัวแปร

เมื่อได้ตัวแปรต่าง ๆ แล้วจะต้องทำการแปลงค่าตัวแปรให้อยู่ในช่วง(-1,1) เพื่อเป็นการปรับความสำคัญของข้อมูลต่าง ๆ ให้เท่ากันก่อนป้อนให้กับโครงข่ายประสาทเทียม โดยเขียน โปรแกรมลงบน Matlab ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

clear; //สั่งลบตัวแปรทุกตัวในหน่วยความจำ
load('penddata'); //อ่านตัวแปรที่เก็บไว้ในไฟล์ส่งตัวแปร
raw=ans;
raw =raw'; //แปลงตัวแปรให้อยู่ในมิติที่คำนวณง่าย
x=raw(:,2); //อ่านค่าจากตัวแปรหลักให้อยู่ในตัวแปรย่อย
xdot=raw(:,3); //ซึ่งตัวแปรจะอยู่ใน เมตริก
theta=raw(:,4); //ที่สื่อความหมายยิ่งขึ้น
thetadot=raw(:,5);
f=raw(:,6);
i=0;
max_x=max(x); //หาค่าสูงสุดของตัวแปรต่าง ๆ
max_xdot=max(xdot);
max_theta=max(theta);
max_thetadot=max(thetadot);
max_f=max(f);
for i=1:10053, //ทำ Normalization
    x(i)=x(i)/max_x;
    xdot(i)=xdot(i)/max_xdot;
    theta(i)=theta(i)/max_theta;
    thetadot(i)=thetadot(i)/max_thetadot;
    f(i)=f(i)/max_f;
end //รวมตัวแปรต่างเป็นตัวแปรเดียวกัน
p=cat(2,x,xdot);
p=cat(2,p,theta);
p=cat(2,p,thetadot);
p=p'; //แปลงมิติของข้อมูลให้กลับไปอยู่ในมิติ
f=f';

```

โปรแกรมที่ 1 แปลงข้อมูลต่าง ๆ ให้มีน้ำหนักของข้อมูลเท่า ๆ กัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.3 การสร้าง Neural Network

การสร้างโครงข่ายประสาทเทียมแบบ Backpropagation ต้องทำการสร้าง Feed forward Network ขึ้นมาก่อน หลังจากนั้นฟังก์ชันโครงข่ายประสาทเทียมจะให้ค่ากลับออกมาเป็นตัวแปรตัวหนึ่งพร้อมที่จะนำไปใช้สอนต่อไป Feed forward Network สร้างโดยใช้คำสั่งใน Matlab ดังนี้

$$Net=newff(PR,[s_1,s_2,s_3,\dots,s_n],\{TF_1,TF_2,TF_3,TF_n\},BTF,BLF,PF)$$

เมื่อ PR	คือ	ค่าสูงสุดและค่าต่ำสุดของอินพุตที่
S _i	คือ	ฟังก์ชันโอนถ่ายของแต่ละชั้นของ Neural
TF _i	คือ	ฟังก์ชันการเรียนรู้ของแต่ละชั้นนั้น ๆ
BTF	คือ	ฟังก์ชันการสอนของ Backpropagation
BLF	คือ	ฟังก์ชันการเรียนรู้ค่า Weight และ Bias
PF	คือ	ฟังก์ชันการแก้ไขค่าความผิดพลาดของ โครงสร้าง

3.4 เปรียบเทียบการเรียนรู้ของโครงข่าย

1. Levenberg-Marquardt : `trainlm` เป็นวิธีการเรียนรู้ที่เร็วที่สุดฟังก์ชัน `trainlm` สามารถปรับลดเวลาการเรียนรู้ลงได้แต่ก็เป็นการลดประสิทธิภาพลงเช่นกันการสอน โค้ดวิธี `trainlm` ใช้เวลาในการสอนนานพอสมควรซึ่งสามารถปรับค่าตัวแปรได้โดยคำสั่ง

`"Net.trainParam.mem_reduc n"`

เมื่อ n เป็นจำนวนที่มีความสัมพันธ์กลับกันระหว่างเวลาและหน่วยความจำ

2. Quasi-Newton : `trainbfg` เป็นวิธีการเรียนรู้ช้ากว่าวิธี Levenberg-Marquardt แต่ประสิทธิภาพสูงกว่า

3. Resilient Backpropagation เป็นวิธีการเรียนรู้ช้ากว่าการเรียนรู้แบบ Levenberg-Marquardt ประสิทธิภาพสูงกว่าวิธี Quasi-Newton การปรับตัวแปรต่าง ๆ ของ โครงข่ายประสาทเทียม

การปรับแต่งค่าต่าง ๆ เพื่อความรวดเร็วในการจำลองการทำงานของโครงข่ายหรือ เพื่อความเหมาะสมต่าง ๆ ทั้งนี้การปรับตัวแปร ต้องอาศัยการทดลองเพื่อเปรียบเทียบผลการทดลองซึ่งใช้เวลาในการเก็บผลนานมาก

ตัวแปรต่าง ๆ เช่นการปรับผลการแสดงผล ปรับค่าการเรียนรู้ หรือปรับจำนวนรอบการเรียนรู้เช่น

`net.trainParam.show=100` คือการปรับค่าการแสดงผลการเรียนรู้ครั้งละ 100 รอบ

`net.trainParam.lr =0.05` คือการปรับค่าการเรียนรู้ (Learning Rate) ของแต่ละรอบ

`net.train.epochs =2000` คือการปรับจำนวนรอบทั้งหมดของการเรียน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อสร้างโครงข่ายเรียบร้อยแล้วจะได้ โครงข่ายซึ่งพร้อมจะนำไปสอนต่อไปโดยการใช้ คำสั่ง *train* ดังนี้

$$[\text{net}, \text{tr}, \text{Y}, \text{E}, \text{Pf}, \text{Af}] = \text{train}(\text{NET}, \text{P}, \text{T}, \text{Pi}, \text{Ai}, \text{VV}, \text{TV})$$

เมื่อ $[\text{net}, \text{tr}, \text{Y}, \text{E}, \text{Pf}, \text{Af}]$ คือ ผลลัพธ์ที่ต้องการจากการเรียนรู้ของ Network

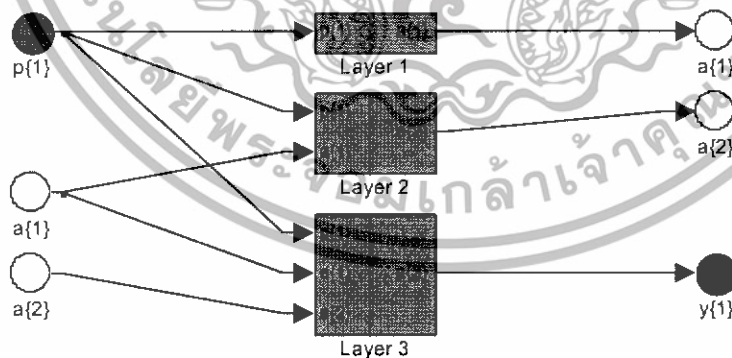
$(\text{NET}, \text{P}, \text{T}, \text{Pi}, \text{Ai}, \text{VV}, \text{TV})$ คือ อินพุตที่ต้องการนำไปสอนประสาทเทียม

การสร้าง Neural Network โดยคำสั่ง *newff* จะประกอบด้วย โครงสร้างเชื่อมกันทุกโหนด ในแต่ละชั้น ทราานเฟอ์ฟังก์ชันจะเป็นอะไรก็ได้ แต่ การใช้ฟังก์ชัน *purelin* จะทำให้ได้ค่าตลอด ช่วงการทำงานต่างจาก *tansig* และ *logsig* ซึ่งมีค่าไม่เกิน 1

```
net = newff(minmax(p), [80, 10, 1], {'tansig', 'logsig', 'purelin'}, 'trainlm');
net.trainParam.show = 100;
net.trainParam.lr = 0.03;
net.trainParam.epochs = 2000;
net.trainParam.goal = 0.1e-9;
[net, f] = train(net, p, f);
```

3.5 รูปแบบของโครงข่ายประสาทเทียมที่นำไปใช้

ภายในโครงสร้างของโครงข่ายประสาทเทียมจะประกอบด้วยเลเยอร์เป็นชั้น ๆ ต่อเรียงกัน อยู่ ซึ่งแต่ละเลเยอร์เชื่อมต่อกันด้วย โครงสร้างแบบเรียงต่อขนาน (Casscase forward Network) ดังรูปที่ 3.8

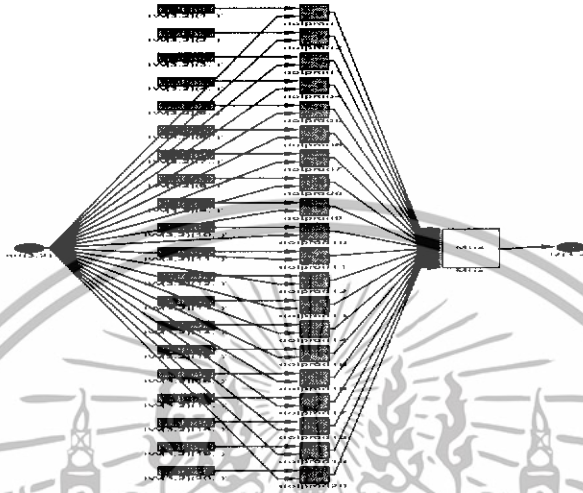


รูปที่ 3.8 แสดงโครงข่ายประสาทเทียมสำหรับนำไปใช้งาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะเห็นว่า โครงสร้างดังกล่าวมีการห้วงเวลาเพื่อเปรียบเทียบกัน โดยการห้วงเวลาคือการนำอดีตมาคำนวณกับค่าปัจจุบันด้วยจึงเป็นข้อได้เปรียบของโครงสร้างชนิดนี้ที่มีการนำค่าของอดีตเป็นต้นแบบของการเรียนปรับค่าน้ำหนักและ ค่าอุปทาน(Weight and Bias)

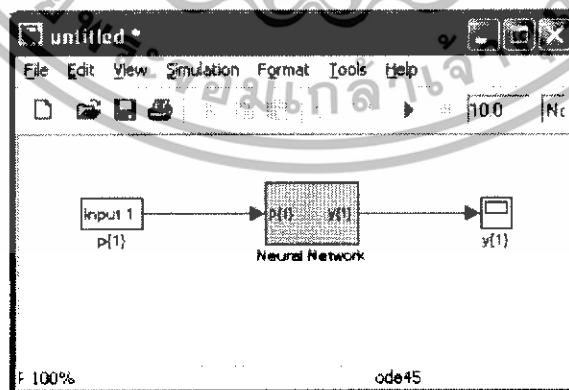
ภายในแต่ละเลเยอร์ประกอบด้วยทราายเฟอร์ฟังก์ชันของแต่ละเลเยอร์นั้น รวมทั้งเวกเตอร์ของ น้ำหนักและ ค่าอุปทาน(Weight and Bias) ดังรูปที่ 3.9



รูปที่ 3.9 แสดงตัวอย่างการเชื่อมต่อของทราายเฟอร์ฟังก์ชัน เวกเตอร์ของ น้ำหนักและ ค่าอุปทาน

3.6 การสร้างนำโครงข่ายประสาทเทียมเข้าไปใช้ Simulink

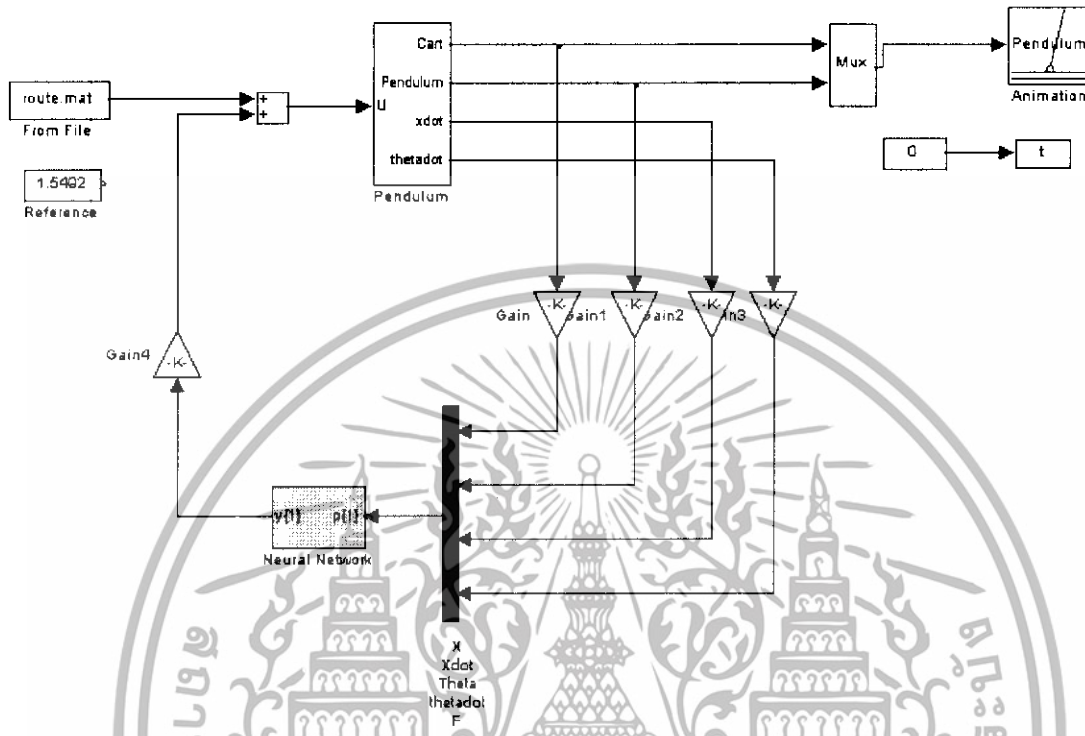
เมื่อสอน โครงข่ายประสาทเทียมแล้วจะ โครงข่ายประสาทเทียมจะปรับค่าของ Weight ให้เข้ามามีความสัมพันธ์กับ เป้าหมายของเรามากที่สุด หลังจากนั้นจะนำ โครงข่ายประสาทเทียมนี้ไปใช้ใน Inverted Pendulum ซึ่งอยู่ใน Simulink ดังนั้นเราต้องทำการสร้าง Block ของ Neural Network ให้เข้าไปอยู่ใน Simulink โดยใช้คำสั่ง `gensim(net, st)`; โปรแกรม Matlab จะทำการสร้าง Block ให้เองโดยอัตโนมัติดังรูปที่ 3.10



รูปที่ 3.10 แสดงผลที่ได้จากการสร้างโครงข่ายประสาทเทียม โดยคำสั่ง `gensim`

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Block ของโครงข่ายประสาทเทียม ดังกล่าวจะประกอบด้วยค่าของ Weight และ Bias ที่เหมาะสมกับระบบของที่เราใช้สอนมัน เมื่อได้ Block โครงข่ายประสาทเทียมแล้วจะนำไปใช้ในแบบจำลองโดยการคัดลอกไปวางบนแบบจำลองของเราได้ทันที ดังรูป



รูปที่ 3.11 แสดงตัวอย่างการนำโครงข่ายประสาทเทียมที่ผ่านการเรียนรู้เรียบร้อยแล้วไปใช้งาน

จากรูปที่ 3.11 ข้อมูลก่อนที่จะป้อนให้กับประสาทเทียมจะต้องผ่านการทำให้เป็นค่าปกติก่อน (Normalization) เพราะข้อมูลที่ใช้สอนประสาทเทียมมีค่าไม่เกินหนึ่ง และทางด้าน Output ก็ต้องแปลงค่าปกติกลับให้เป็นค่าใช้งานด้วย (Denormalized)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.7 การออกแบบ โครงข่ายประสาทเทียมแบบ Self – Organized Feture Map

โดยการใช้โครงข่ายประสาทชนิด Feedforward รวมเข้ากับ Recurrent Network(หรือ Kohone Layer) โดยมี Output Layer เป็น โครงข่ายประสาทชั้นสุดท้าย มีจำนวนประสาทเทียมในแต่ละเลเยอร์ดังนี้ (2,30x30,1) ซึ่งจะมีการคำนวณที่ต้องจองพื้นที่สำหรับ คณิตเป็นจำนวนมาก จึงต้องทำการจองหน่วยความจำไว้เพื่อกันการเขียนทับหน่วยความจำ และ เพื่อความสะดวกของการชี้เพื่ออ่านค่า น้ำหนัก

ส่วนการเรียนของโครงข่ายแบบ Self – Organized Feture Map ใช้การให้คะแนนแทนการปรับค่าน้ำหนัก และ ไม่มีการกำหนดเป้าหมายของการเรียนรู้ เพียงแต่กำหนดว่า มุมที่มีค่าน้อยกว่าคะแนนเฉลี่ยจะถูกปรับน้ำหนัก(Weight)ด้วยเงื่อนไข

```

If(dScore>dScoreMean){
Target[0]=Pole.F;
TrainUnits.(Net,Input,Target);
}
    
```

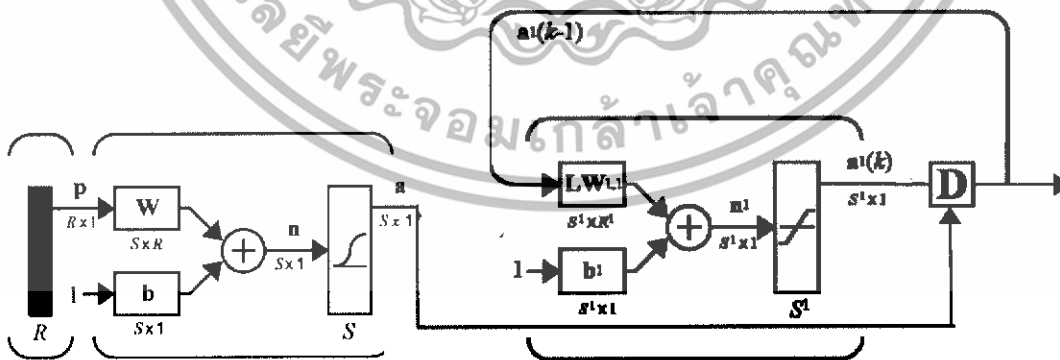
แต่ถ้าไม่เข้าเงื่อนไขจะทำการปรับค่า dScoreMean ของ winner ด้วยฟังก์ชัน
 $Net \rightarrow KohonenLayer \rightarrow dScoreMean[Net \rightarrow Winner] += Net \rightarrow Gamma * (dScore - dScoreMean);$

โครงข่ายแบบที่มีการยังยั้งดังกล่าวจะเรียกว่า Competitive Network ซึ่งจะทำการปรับค่าน้ำหนักด้วยวิธีของ Kohonen

ทุกๆ รอบการทำงาน จะมีการปรับค่าการเรียนรู้ ของทุก ๆ เลเยอร์ และ ปรับค่าซ้ำยังรอบ ๆ winner ด้วยฟังก์ชัน

```

Net->Alpha = 0.5 * pow(0.01, (REAL) n / TRAIN_STEPS);
Net->Alpha_ = 0.5 * pow(0.01, (REAL) n / TRAIN_STEPS);
Net->Alpha__ = 0.005;
Net->Gamma = 0.05;
Net->Sigma = 6.0 * pow(0.2, (REAL) n / TRAIN_STEPS);
    
```



รูปที่ 3.12 แสดง โครงข่ายประสาทเทียมที่นำไปใช้งาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.8 องค์ประกอบของ Pendulum ในภาษาซี

องค์ประกอบของ Pendulum จะถูกเขียนโดยการกำหนดคุณสมบัติต่าง ๆ ด้วยแนวคิดการเขียนโปรแกรมเชิงวัตถุ องค์ประกอบต่าง ๆ จะถูกบรรยายไว้ในตัวแปร POLE ซึ่งประกอบด้วยสมาชิก $w, wDot, x, xDot$ และ F แทนตัวแปร มุม ความเร็วเชิงมุม ระยะทาง ความเร็ว และ แรงที่ให้กับฐาน(Cart) ตามลำดับ ดังนี้

```
typedef struct {          /* STATE VARIABLES OF A POLE:*/
    REAL    x;           /* - position of cart [m]          */
    REAL    xDot;        /* - velocity of cart [m/s]       */
    REAL    w;           /* - angle of pole [°]            */
    REAL    wDot;        /* - angular velocity of pole [°/s] */
    REAL    F;           /* - force applied to cart        */
} POLE;                  /* during current time step [N]   */
```

ส่วนวิธีการต่าง ๆ นั้นจะใช้วิธีการของ Euler เพื่อการหาค่าของตัวแปรจากสมการอนุพันธ์อันดับสอง

```
void SimulatePole(POLE *Pole) {
    INT s;
    REAL x, xDot, xDotDot;
    REAL w, wDot, wDotDot;
    REAL F;
    x    = Pole->x;
    xDot = Pole->xDot;
    w    = (Pole->w / 180) * PI;
    wDot = (Pole->wDot / 180) * PI;
    F    = Pole->F;
    for (s=0; s<STEPS; s++){
        wDotDot = (G*sin(w) + cos(w) * ((-F - Mp*L*(wDot)*(wDot)*sin(w)) / (Mc+Mp))) /
            (L * ((REAL) 4/3 - (Mp*(cos(w))*(cos(w))) / (Mc+Mp)));

        xDotDot = (F + Mp*L * ((wDot)*(wDot)*sin(w) - wDotDot*cos(w))) / (Mc+Mp);

        x      += (T / STEPS) * xDot;
        xDot   += (T / STEPS) * xDotDot;
        w      += (T / STEPS) * wDot;
        wDot   += (T / STEPS) * wDotDot;
    }
    Pole->x      = x;
    Pole->xDot   = xDot;
    Pole->w      = (w / PI) * 180;
    Pole->wDot   = (wDot / PI) * 180;
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.5.1 เงื่อนไขความเสถียรของ ก้าน Pendulum

เงื่อนไขของการทรงตัวได้ด้วยเงื่อนไข Boolean ซึ่งจะส่งค่ากลับออกมาเป็น จริงหรือ เท็จเมื่อมุม อยู่ภายในมุม $\pm 60^\circ$ เมื่อทำกับมุมปกติด้วยฟังก์ชัน

```

BOOL PoleStillBalanced(POLE* Pole)
{
    return (Pole->w >= -60) && (Pole->w <= 60);
}

```

3.5.2 เงื่อนไขเริ่มต้นของระบบ

เมื่อเริ่มต้นระบบจะต้องทำการตั้งค่า ๆ ให้กับระบบ โดยมุมจะอยู่ในตำแหน่งตามแนวความเร่งโลก ความเร็ว ตำแหน่งของฐานอยู่ตรงกลาง เมื่อต้องการเริ่มต้นทำงานตัวแปรทุกตัวยังอยู่ในตำแหน่งเดิมยกเว้นมุม เพราะถ้า มุมเป็น 0° จะทำให้ ระบบอยู่ในความเสถียร จึงต้องกำหนดค่าเริ่มต้นน้อย ๆ แบบสุ่มให้กับมุมด้วย ฟังก์ชัน RandomEqualREAL();

การกำหนดค่าต่าง ๆ เริ่มต้นดังนี้

```

void InitializePole(POLE* Pole)
{

```

```

    do {
        Pole->x      = 0;
        Pole->xDot    = 0;
        Pole->w      = RandomEqualREAL(179,181);
        Pole->wDot    = 0;
        Pole->F      = 0;
    } while (Pole->w == 0);
}

```

เมื่อเริ่มทำงานให้กำหนดมุมใหม่ ด้วยคำสั่ง

```
Pole.w = RandomEqualREAL(-1,1);
```

เมื่อมีแรงมาทำกับฐานจะต้องส่งผ่านตัวแปร Pole.F เช่น มีแรงขนา 100 นิวตัน ทำกับฐานต้องเขียน

```

Pole.F = 100;
SimulatePole(&Pole);

```

ฟังก์ชัน SimulatePole จะให้ค่าตัวแปร มุม ความเร็วเชิงมุม ระยะทาง และความเร็วออกมาผ่าน

ตัวแปร Pole

3.5.3 การแสดงผลบน วิชวลซี พลัส พลัส 6

การแสดงผลการทำงานของโปรแกรมโดยแสดงแบบ อนิเมชัน(Animation) บน Visual ต้องทำการกำหนด RECT Class ก่อนจึงเรียกฟังก์ชันย่อย ๆ ออกมาใช้ได้ คลาสของการแสดงสืบทอดมาจากคลาสหลักคือคลาส CObject



รูปที่ 3.11 แสดงการสืบทอดของ คลาส CDC

```

CDC *cdc = dlg->GetDC();
dlg->BeginPaint(&ps);
    
```

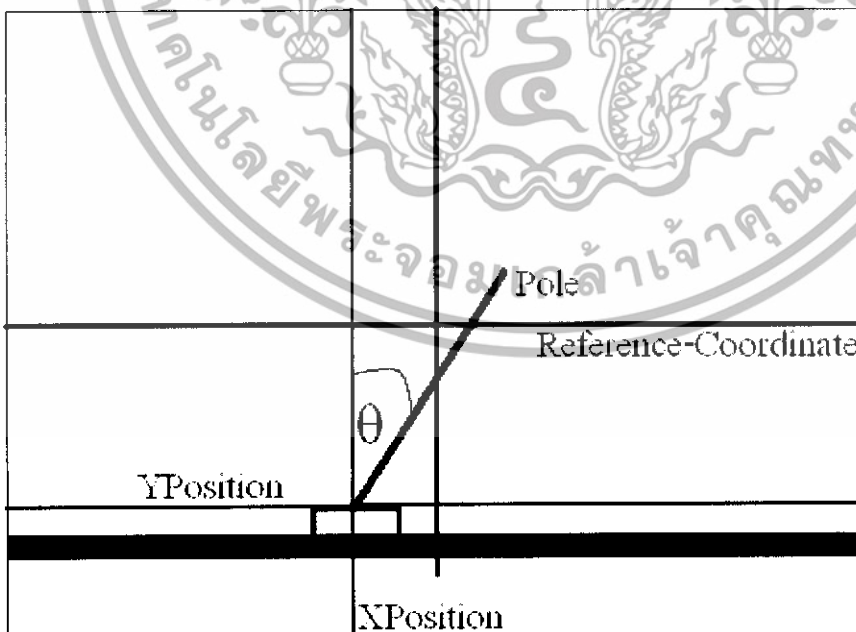
```

RECT rt;
PAINTSTRUCT ps;
CString str;
    
```

เมื่อ rt แทน Struct ของ Rectangle ซึ่งมีองค์ประกอบทั้งสี่ เป็น

- LONG left;
- LONG top;
- LONG right;
- LONG bottom;

โปรแกรมการแสดงผลอนิเมชัน โดยการใช้คำสั่ง LineTo , MoveTo และ Rectangle มีข้อได้เปรียบเสียเปรียบกันไม่มากนัก ส่วนมาก LineTo และ MoveTo มักใช้ในการวาดองค์ประกอบเส้นตรงที่มีจำนวนมากมีรูปร่างไม่แน่นอน ส่วน Rectangle ใช้สำหรับวาดรูปสี่เหลี่ยมสองมิติโดยการกำหนดจุดเพียงสองจุด



รูปที่ 3.12 แสดงวิธีการคิดสมการเพื่อแสดงผลของ ระบบอินเวอร์ตเพนดูลัม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตำแหน่งของฐานบนระนาบแกน Y นั้นมีค่าคงที่ ส่วน แกน X จะเปลี่ยนค่าได้ตั้งเป็นตัวแปร XPosition ความสูงของฐานมีขนาด 20 จุดพิกเซล ความกว้างของฐานมีขนาด 60 จุดพิกเซลสามารถเขียนลงจุดหมุนของแกนจะอยู่ตำแหน่งเดียวกันกับฐาน ส่วนปลายอีกด้านสามารถหมุนได้ตามมุม θ ปลายของสมการจะมีค่าเป็น

$$(\text{PoleLengt} \times \sin(\text{Angle}), \text{PoleLengt} \times \cos(\text{Angle})); \quad (3.6)$$

แต่เมื่อฐานมีการเคลื่อนที่ต้องทำการย้ายแกนให้เป็น

$$((275 + \text{XPosition}) + \text{PoleLengt} \times \sin(\text{Angle}), 340 + 60 - 20 - \text{PoleLengt} \times \cos(\text{Angle})) \quad (3.7)$$

การเคลื่อนที่ของฐานและก้านน้ำหนัก ทำได้โดยการป้อนมุมและตำแหน่งให้กับตัวแปร XPosition และ Angle ตามลำดับดังนี้

```
// Draw A Base
rt.left = 275-30-(-XPosition);
rt.right= 275+30-(-XPosition);
rt.top   = 380;
rt.bottom = 400;
cdc->Rectangle(&rt);
Pole.xDot=0.9*Pole.xDot;
Pole.wDot=0.996*Pole.wDot;
// Draw A Pole
cdc->MoveTo(275+XPosition+i,380);
cdc->LineTo((275+XPosition+i)+PoleLengt*sin(PI*Angle/180.00)
,340+60-20-PoleLengt*cos(PI*Angle/180));
```

เมื่อจบการทำงาน หนึ่งรอบจะต้องวาดหน้าจอใหม่และ คืนค่าให้กับหน่วยความจำการแสดงผลด้วย

คำสั่ง

```
dlg->EndPaint(&ps);
dlg->ReleaseDC(cdc);
```

ถ้าไม่ทำการวาดหน้าจอใหม่และคืนค่าให้กับหน่วยความจำจะทำให้หน่วยความจำไม่พอในการแสดงผล
ครั้งต่อไป

3.5.4 การแสดงผลบนการทำงานของ เทอโบซี

การแสดงผลการทำงานของระบบด้วยเทอโบซีจะแตกต่างออกไปจากการแสดงผลบนวิชวลซีคือการแสดงผลจะอาศัยคำสั่งแตกต่างกันออกไป สามารถแบ่งส่วนของการแสดงผลได้ดังนี้

1. ทำการกำหนดเงื่อนไขของการติดต่อกกราฟฟิกด้วยฟังก์ชัน Initgraph();
2. เมื่อแสดงผลเสร็จแล้วจะต้องทำการยกเลิกการติดต่อกกับกราฟฟิกด้วยคำสั่ง closegraph());

การแสดงผลของเทอโบซีจะใช้วิธีการลบนหน้าจอดีจากวิชวลซีคือ จะทำการเก็บค่าของจุดเฉพาะที่วาดลงไปเท่านั้นแล้วจึงลบเฉพาะจุดที่เขียนลงไป การแสดงผลจึงมีการกระพริบอยู่เสมอดังฟังก์ชัน

```
void Draw(double XPosition,double Angle,char stt(20)){
double OldXPosition,OldAngle;
double OldCartWidth,OldCartHight,OldPoleLenght,OldRailHight;
setcolor(15);
OldXPosition = XPosition;
OldAngle = Angle;
XPosition=getmaxx()/2-CartWidth/2+XPosition;
// Draw a Cart and Pole
rectangle(1,getmaxy()-RailHight,getmaxx()-1,getmaxy()-1); //Draw A Rail
rectangle(XPosition,getmaxy()-RailHight-CartHight,XPosition+CartWidth,getmaxy()-CartHight);
//Draw A Cart
moveto(XPosition+CartWidth/2.0,getmaxy()-RailHight-CartHight); //Draw A Pole
lineto(XPosition+CartWidth/2.0+PoleLenght*sin(PI*Angle/180.0),getmaxy()-RailHight-CartHight-
PoleLenght*cos(PI*Angle/180.0)); //Draw A Pole
outtextxy(10,getmaxy()-10,stt);
outtextxy(1.1,FallTimer);
outtextxy(1.20,Stt1);
outtextxy(1.30,Stt2);
outtextxy(1.40,Stt3);
delay(DelayConstance-60);
//Clear a cart and Pole
setcolor(0);
OldXPosition=getmaxx()/2-CartWidth/2+OldXPosition;
rectangle(1,getmaxy()-RailHight,getmaxx()-1,getmaxy()-1); //Draw A Rail
rectangle(OldXPosition,getmaxy()-RailHight-CartHight,OldXPosition+CartWidth,getmaxy()-CartHight);
//Draw A Cart
moveto(OldXPosition+CartWidth/2.0,getmaxy()-RailHight-CartHight); //Draw A Pole
lineto(OldXPosition+CartWidth/2.0+PoleLenght*sin(PI*OldAngle/180.0),getmaxy()-RailHight-CartHight-
PoleLenght*cos(PI*OldAngle/180.0)); //Draw A Pole
outtextxy(10,getmaxy()-10,stt);
outtextxy(1.1,FallTimer);
outtextxy(1.20,Stt1);
outtextxy(1.30,Stt2);
outtextxy(1.40,Stt3);

// cleardevice();
}
```

การแสดงผลการทำงานทุก ๆ อย่างควรจะเขียนผ่านฟังก์ชันนี้เพราะการลบนหน้าจอเพื่อเขียนใหม่จะทำได้ไม่ซับซ้อนมากนัก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.9 การเก็บข้อมูลบนภาษาซี

การเก็บข้อมูลด้วยการเขียนลงไฟล์แล้วนำไปแสดงผลบนโปรแกรมแมทแลป มีขั้นตอนการทำงานคือ

1. สร้างไฟล์พอยเตอร์ FILE *f;
2. สร้างไฟล์ที่ต้องการเขียนลงไปโดย f = fopen(f, "som.txt", "w");
3. เขียนข้อมูลที่ต้องการลงบนไฟล์ที่พอยเตอร์ชื่ออยู่เช่น ถ้าต้องการเก็บข้อมูลของมุมและความเร่งเชิงมุม
ดังนี้ fprintf(f, " %lf %lf ", Pole.w, Pole.wDot);

เมื่อเสร็จสิ้นการทำงานของโปรแกรมแล้วจะได้ข้อมูลในไฟล์ ชื่อ som.txt แล้วนำไปให้โปรแกรมแมทแลปแสดงผลของกราฟข้อมูลออกมา

3.10 องค์ประกอบของโครงข่ายประสาทเทียมแบบไม่มีต้นแบบ

โครงข่ายประสาทเทียมแบบ Self-Organizing Fature Map ประกอบด้วย โครงข่ายประสาทเทียมสามประเภทคือ Input layer, Kohonen layer และ Output layer การเขียนโครงข่ายประสาทเทียมจะต้องให้จำนวนประสาทเทียมในแต่ละชั้นเป็นจำนวนมากโดยการบรรยายคุณสมบัติของประสาทเทียมไว้ก่อนแล้วจึงสร้างประสาทเทียมแต่ละตัวจากคุณสมบัติที่ได้บรรยายไว้ โดยอธิบายไว้สองแบบคือ Layer และ Net ซึ่งในแต่ละ Net จะมีองค์ประกอบของ Layer ภายใน

3.11 ขั้นตอนการทำงานของโครงข่ายประสาทเทียม

การทำงานของโครงข่ายประสาทเทียมแบบเรียนรู้ด้วยตัวเองนั้นจะมีการแบ่งการทำงานออกเป็นสองส่วนคือ ส่วนของการเรียนรู้และส่วนของการนำไปใช้ ดังนั้นต้องมีการเก็บค่าเอาท์พุทเพื่อจะได้ทราบว่าเมื่อใดต้องเรียนรู้และเมื่อใดจึงนำค่าไปใช้ การเรียนรู้นั้นจะเกิดขึ้นเมื่อเกิดการให้รางวัลขึ้น จะมีเพียงประสาทเทียมตัวเดียวเท่านั้นที่จะให้ค่าเอาคัพุทออกมาได้ เมื่อโครงข่ายประสาทเทียมได้รับตัวแปร Pole จากฟังก์ชัน SimulatePole(&Pole) แล้ว จะให้รางวัลกับค่าแรงต่อเหตุการณ์นั้น ถ้ามุมของ Pole.w มีค่าไม่อยู่ในช่วง $\pm 60^\circ$ (Bool PoleStillBalanced())แล้วจะทำการเรียนรู้รอบใหม่

ค่าในชั้น Kohonen Layer ที่ถูกปรับตามสถานการณ์นั้น จะเป็นไปได้สองทางโดยอ้างอิงจากคะแนนของประสาทเทียมตัวที่ชนะในเลเยอร์ เมื่อเกิดคะแนนที่มากกว่า ค่าเฉลี่ยคะแนนทั้งหมดประสาทเทียมจะทำการเรียนรู้ใหม่ ดังนี้

```
if (dScore > dScoreMean) {
    Target[0] = Pole.F;
    TrainUnits(Net, Input, Target);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ดังนั้นในฟังก์ชันนี้สามารถเขียนได้ดังนี้

```
void TrainNet(NET* Net)
{
    INT n,t;
    POLE Pole;
    REAL wOld, wNew, ScoreOld, ScoreNew, dScore, dScoreMean, StepSize;
    REAL Input[N];
    REAL Output[M];
    REAL Target[M];
    char Stt[100];
    char ch;
    n = 0;
    while (n<TRAIN_STEPS) {
        t = 0;
        InitializePole(&Pole);
        wOld      = Pole.w;
        ScoreOld   = ScoreOfPole(&Pole);
        SimulatePole(&Pole);
        wNew      = Pole.w;
        ScoreNew   = ScoreOfPole(&Pole);
        while (PoleStillBalanced(&Pole) AND (t<BALANCED)) {
            n++;
            t++;
            Net->Alpha  = 0.5 * pow(0.01, (REAL) n / TRAIN_STEPS);
            Net->Alpha_  = 0.5 * pow(0.01, (REAL) n / TRAIN_STEPS);
            Net->Alpha__ = 0.005;
            Net->Gamma   = 0.05;
            Net->Sigma   = 6.0 * pow(0.2, (REAL) n / TRAIN_STEPS);
            Input[0]    = wOld;
            Input[1]    = wNew;
            SetInput(Net, Input);
            PropagateNet(Net);
            GetOutput(Net, Output);
            Pole.F      = Output[0];

            StepSize    = Net->KohonenLayer->StepSize[Net->Winner];
            Pole.F      += StepSize * RandomNormalREAL(0, 10);

            sprintf(Stt,"Neural Network Control An Inverted Pendulum.Iteration=%5.2d (ESC Exit)
",n);
            Draw(Pole.x*XConstance,Pole.w,Stt);
            // ThetaGraph(Pole.xDot,Pole.wDot);
            Pole.xDot   = 0.94*Pole.xDot;
            Pole.w      = 0.96*Pole.w;
            wOld        = Pole.w;
            ScoreOld    = ScoreOfPole(&Pole);

            SimulatePole(&Pole);

            wNew        = Pole.w;
            ScoreNew    = ScoreOfPole(&Pole);
            dScore      = ScoreNew - ScoreOld;
            dScoreMean  = Net->KohonenLayer->dScoreMean[Net->Winner];
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

sprintf(Stt1,"Pole's Angle(Degree) : %5.4f",Pole.w);
sprintf(Stt2,"Force (N) : %5.4f",Pole.F);
while(kbhit()){
ch=getch();
switch(ch){
case 27:
closegraph();
FinalizeApplication(&Net);
exit(0);
break;
}
}

//delay(40);
if (dScore > dScoreMean) {
Target[0] = Pole.F;
TrainUnits(Net, Input, Target);
}
Net->KohonenLayer->dScoreMean[Net->Winner] += Net->Gamma * (dScore -
dScoreMean);
}
if (PoleStillBalanced(&Pole)){
sprintf(FallTimer, "Pole still balanced after %0.1fs", t * T);
SuccessCounter++;
sprintf(Stt3,"Success :%4d",SuccessCounter);
}
else{
sprintf(FallTimer, "Pole fallen after %0.1fs", (t+1) * T);
sprintf(Stt3,"Success :%4d",SuccessCounter);
}
}
}
}

```

เห็นว่าจะมีการแสดงผลควบคู่กับการเรียนรู้และการปรับค่าต่างๆ อยู่พร้อมๆ กัน โดยจะไม่มีการทำงานอย่างใดอย่างหนึ่งให้เสร็จก่อน แล้วจึงค่อยทำงานอีกส่วนถัดไป เรียกการทำงานชนิดนี้ ว่าแอนดิลโฟรเซส (Handle Processes)

เมื่อพิจารณาภาคผนวก ตามสามารถอธิบายขั้นตอนการทำงานของระบบทั้งหมดได้เป็นส่วนๆ ดังนี้

1. ธรรมชาติของการตกของก้านเพนดูลัม และการเคลื่อนที่ของฐาน การกำหนดค่าเฉพาะต่างๆ ของระบบ เช่น น้ำหนักของก้านเพนดูลัม และ ฐาน ความยาวของ ก้านเพนดูลัม ความเร่งของโลก ค่าคงที่การอินตริเกต คุณสมบัติเชิงพลศาสตร์ของระบบ เช่น ความเร่งเชิงมุม ความเร็วเชิงมุม ความเร่งเชิงเส้น ความเร็วเชิงเส้น และ แรง

2. ส่วนของการทำงานของโครงข่ายประสาทเทียม

2.1 คุณสมบัติของเลเยอร์ จะอธิบายไว้ในตัวแปรรวม ประกอบด้วยน้ำหนัก คะแนน จำนวนประสาทในเลเยอร์ จำนวนเอาต์พุตของเลเยอร์และขนาดของการเพิ่มขึ้นของน้ำหนัก

2.2 คุณสมบัติของประสาทเทียมในแต่ละเลเยอร์ คุณสมบัติต่าง ๆ จะถูกอธิบายไว้ในตัวแปรรวมตัวเดียวกัน เพื่อนำมาสร้างเป็นเลเยอร์ต่าง ๆ ดังนั้น แต่ละเลเยอร์จะมีโครงสร้างของประสาทเหมือนกัน เพียงแต่บางคุณสมบัติจะไม่ถูกนำมาใช้ในบางเลเยอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.3 การสุ่มค่าต่างๆ ซึ่งจะมีฟังก์ชันย่อย ๆ ซ้อน ๆ กันไป เช่นการกำหนดช่วงของการสุ่มทั้งหมด การสุ่มในขอบเขตจำกัด และการสุ่มแบบกระจายปกติ ทุก ๆ ฟังก์ชัน จะมีการส่งค่าต่อกันไปเรื่อย ๆ โดยผ่านฟังก์ชันนั้นๆ

2.4 การกำหนดเงื่อนไขเริ่มต้นภายในชั้น Kohonen การกำหนดค่าเริ่มต้นซึ่งเป็นการกำหนดค่าคงที่ลงไปในแลเยอร์ให้มีค่าสำหรับการคำนวณได้ โดยจะต้องกำหนดค่าของชั้นการเพิ่มน้ำหนักเป็น 1 เท่ากันทุก ๆ จำนวนประสาทในแลเยอร์ และค่าคะแนนเฉลี่ยให้เป็น 0 ก่อนที่จะทำการเรียนรู้ใช้ฟังก์ชัน

```
void InitializeApplication(NET* Net)
{
    INT i;
    for (i=0; i<Net->KohonenLayer->Units; i++) {
        Net->KohonenLayer->StepSize[i] = 1;
        Net->KohonenLayer->dScoreMean[i] = 0;
    }
}
```

2.5 การจองพื้นที่ของหน่วยความจำของแต่ละประสาทเทียมของแต่ละแลเยอร์ ซึ่งจะต้องจองทุกๆ ประสาทเทียมทุกตัว เพราะง่ายต่อการนำดัชนีไปใช้เพื่อการปรับค่าน้ำหนัก และสามารถกำหนดหน่วยความจำได้จำกัด โดยการจองหน่วยความจำแบ่งได้ออกเป็นสองกลุ่มคือการจองแบบอาศัยเพียงความกว้างของชนิดข้อมูลเท่านั้นซึ่งเหมาะกับข้อมูลที่มีขนาดแน่นอน และการจองข้อมูลเป็นกลุ่มใหญ่ต่อ ๆ กัน ทั้งหมดนี้อาศัยฟังก์ชัน

```
void GenerateNetwork(NET* Net){
    INT i;
    Net->InputLayer = (LAYER*) malloc(sizeof(LAYER));
    Net->KohonenLayer = (LAYER*) malloc(sizeof(LAYER));
    Net->OutputLayer = (LAYER*) malloc(sizeof(LAYER));
    Net->InputLayer->Units = N;
    Net->InputLayer->Output = (REAL*) calloc(N, sizeof(REAL));
    Net->KohonenLayer->Units = C;
    Net->KohonenLayer->Output = (REAL*) calloc(C, sizeof(REAL));
    Net->KohonenLayer->Weight = (REAL**) calloc(C, sizeof(REAL*));
    Net->KohonenLayer->StepSize = (REAL*) calloc(C, sizeof(REAL));
    Net->KohonenLayer->dScoreMean = (REAL*) calloc(C, sizeof(REAL));
    Net->OutputLayer->Units = M;
    Net->OutputLayer->Output = (REAL*) calloc(M, sizeof(REAL));
    Net->OutputLayer->Weight = (REAL**) calloc(M, sizeof(REAL*));
    for (i=0; i<C; i++) {
        Net->KohonenLayer->Weight[i] = (REAL*) calloc(N, sizeof(REAL));
    }
    for (i=0; i<M; i++) {
        Net->OutputLayer->Weight[i] = (REAL*) calloc(C, sizeof(REAL));
    }
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6 การนำ โครงข่ายเข้ามาเพื่อปรับน้ำหนัก โดยการนำค่าเอาต์พุต โครงข่ายใด ๆ ที่ต้องการนำเข้ามาซึ่งเป็น โครงข่ายของ เอาต์พุตเลเยอร์เข้ามาเป็น อินพุตของ อินพุตเลเยอร์อาศัยฟังก์ชัน

```
void SetInput(NET* Net, REAL* Input)
{
  INT i;
  for (i=0; i<Net->InputLayer->Units; i++) {
    Net->InputLayer->Output[i] = Input[i];
  }
}
```

นอกจากนั้นการรับค่าเอาต์พุตออกมาเก็บเอาไว้เพื่อนำไปให้กับเลเยอร์ถัดไปก็มีความสำคัญไม่แพ้กัน คล้ายกับการนำโครงข่ายเข้าไปปรับค่าอินพุต ซึ่งต้องมีพารามิเตอร์เป็น โครงข่ายและ อินพุต การรับค่าเอาต์พุตก็ต้อง มี โครงข่ายที่ต้องส่งค่าออกและมีตัวแปรที่นำมาปรับค่าเอาต์พุตเช่นกัน อาศัยฟังก์ชัน

```
void GetOutput(NET* Net, REAL* Output)
{
  INT i;
  for (i=0; i<Net->OutputLayer->Units; i++) {
    Output[i] = Net->OutputLayer->Output[i];
  }
}
```

2.7 การหาค่าต่ำสุดของ Kohonen Layer หรือการหาประสาทเทียมตัวที่ชนะ โดยการนำแต่ละ เอาต์พุตของอินพุตเลเยอร์ลบกับน้ำหนักแต่ละตัวของ Kohonen เลเยอร์ ผลลัพธ์กำลังสองจะเป็นผลลัพธ์ของ Kohonen Layer เพื่อให้หาค่าต่ำสุด หรือ ผลต่างกำลังสองของเอาต์พุตของอินพุตเลเยอร์ คูณกับน้ำหนักใน Kohonen layer คือ ค่าน้อยที่สุดที่ต้องการนำไปเปรียบเทียบกับค่าอื่น ๆ ในวิธีเดียวกันด้วยฟังก์ชัน

```
void PropagateToKohonen(NET* Net)
{
  INT i,j;
  REAL Out, Weight, Sum, MinOut;

  for (i=0; i<Net->KohonenLayer->Units; i++) {
    Sum = 0;
    for (j=0; j<Net->InputLayer->Units; j++) {
      Out = Net->InputLayer->Output[j];
      Weight = Net->KohonenLayer->Weight[i][j];
      Sum += sqr(Out - Weight);
    }
    Net->KohonenLayer->Output[i] = sqrt(Sum);
  }
  MinOut = MAX_REAL;
  for (i=0; i<Net->KohonenLayer->Units; i++) {
    if (Net->KohonenLayer->Output[i] < MinOut)
      MinOut = Net->KohonenLayer->Output[Net->Winner = i];
  }
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.8 การกระจายค่าฟังก์ชัน ของประสาทเทียมที่ชนะออกไปยัง Output Layer หลักการเดียวกับข้อข้างบน แต่เป็นการกระจายค่าออกจาก Output Layer เท่านั้นอาศัยฟังก์ชัน

```
void PropagateToOutput(NET* Net)
{
  INT i;

  for (i=0; i<Net->OutputLayer->Units; i++) {
    Net->OutputLayer->Output[i] = Net->OutputLayer->Weight[i][Net->Winner];
  }
}
```

การกระจายค่าทั่วโครงข่ายประสาทเทียม อาศัยการกระจายตามข้อ 2.8 และ 2.9 ตามลำดับ

2.9 สำหรับโครงข่ายประสาทเทียมแบบ Self-Organizing Fature Map แล้วจะมีการให้รางวัล (การคูณด้วยค่ามาก) และการขยับยั้ง (การคูณด้วยค่าน้อย) เวกเตอร์ซึ่งชนะจะได้รับการคูณด้วยค่า λ (double Lamda;) เพื่อปรับค่าอัตราการเรียนรู้ให้มากขึ้นกว่าค่าอัตราการเรียนรู้ของประสาทเทียมตัวอื่น ๆ การเรียนรู้ชนิดนี้เกิดขึ้นในชั้น Kohonen Layer ดังฟังก์ชัน

```
void TrainKohonen(NET* Net, REAL* Input)
{
  INT i,j;
  REAL Out, Weight, Lambda, StepSize;
  for (i=0; i<Net->KohonenLayer->Units; i++) {
    for (j=0; j<Net->InputLayer->Units; j++) {
      Out = Input[j];
      Weight = Net->KohonenLayer->Weight[i][j];
      Lambda = Neighborhood(Net, i);
      Net->KohonenLayer->Weight[i][j] += Net->Alpha * Lambda * (Out - Weight);
    }
    StepSize = Net->KohonenLayer->StepSize[i];
    Net->KohonenLayer->StepSize[i] += Net->Alpha * Lambda * -StepSize;
  }
}
```

2.10 สำหรับ Output Layer แล้วจะลักษณะการปรับ weight คล้ายๆ กันคือจะอาศัยระยะห่างจากเวกเตอร์ชนะเหมือนกัน แต่จะ ไม่มีการนำค่า Search Step มาคิดด้วย ดังฟังก์ชัน

```
void TrainOutput(NET* Net, REAL* Output)
{
  INT i,j;
  REAL Out, Weight, Lambda;
  for (i=0; i<Net->OutputLayer->Units; i++) {
    for (j=0; j<Net->KohonenLayer->Units; j++) {
      Out = Output[i];
      Weight = Net->OutputLayer->Weight[i][j];
      Lambda = Neighborhood(Net, j);
      Net->OutputLayer->Weight[i][j] += Net->Alpha * Lambda * (Out - Weight);
    }
  }
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.11 การเรียนรู้ของโครงข่ายประสาทเทียม เริ่มจากการสุ่มค่าเข้าไปในโครงข่าย เพื่อสุ่มแรง ออกมา ดังนั้น เมื่อเริ่มแรกแทบไม่สามารถตั้งให้ ก้านเพนดูลัมตั้งอยู่ได้ การเรียนรู้ของทั้ง โครงข่ายประสาทเทียมมีดังนี้

```
void TrainNet(NET* Net)
{
    INT n,t;
    POLE Pole;
    REAL wOld, wNew, ScoreOld, ScoreNew, dScore, dScoreMean, StepSize;
    REAL Input[N];
    REAL Output[M];
    REAL Target[M];

    double StartTime,StopTime;

    char Stt[100];
    char ch;
    n = 0;
    while (ch!=27) {
        t = 0;
        InitializePole(&Pole);
        wOld = Pole.w;
        ScoreOld = ScoreOfPole(&Pole);
        SimulatePole(&Pole);
        wNew = Pole.w;
        ScoreNew = ScoreOfPole(&Pole);
        while (PoleStillBalanced(&Pole) AND (t<BALANCED)) {
            n++;
            t++;
            Net->Alpha = 0.5 * pow(0.01, (REAL) n / TRAIN_STEPS);
            Net->Alpha_ = 0.5 * pow(0.01, (REAL) n / TRAIN_STEPS);
            Net->Alpha__ = 0.005;
            Net->Gamma = 0.05;
            Net->Sigma = 6.0 * pow(0.2, (REAL) n / TRAIN_STEPS);
            Input[0] = wOld;
            Input[1] = wNew;
            SetInput(Net, Input);
            PropagateNet(Net);
            GetOutput(Net, Output);
            Pole.F = Output[0];

            StepSize = Net->KohonenLayer->StepSize[Net->Winner];
            Pole.F += StepSize * RandomNormalREAL(0, 10);

            sprintf(Stt,"Neural Network Control An Inverted Pendulum.Iteration=%5.2d (ESC
            Exit) ",n);
            Draw(Pole.x*XConstance,Pole.w,Stt);
            Pole.xDot = 0.94*Pole.xDot;
            Pole.w = 0.96*Pole.w;
            wOld = Pole.w;
            ScoreOld = ScoreOfPole(&Pole);
        }
    }
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

SimulatePole(&Pole);

wNew      = Pole.w;
ScoreNew  = ScoreOfPole(&Pole);
dScore    = ScoreNew - ScoreOld;
dScoreMean = Net->KohonenLayer->dScoreMean[Net->Winner];
sprintf(Stt1,"Pole's Angle(Degree) : %5.4f",Pole.w);

sprintf(Stt2,"Force   (N)   : %5.4f",Pole.F);
while(kbhit()){
    ch=getch();
    switch(ch){
        case 27:
            closegraph();
            FinalizeApplication(&Net);
            exit(0);
//            break;
        }
    }
if (dScore > dScoreMean) {
    Target[0] = Pole.F;
    TrainUnits(Net, Input, Target);
}
Net->KohonenLayer->dScoreMean[Net->Winner] += Net->Gamma * (dScore -
dScoreMean);
}
if (PoleStillBalanced(&Pole)){
    sprintf(FallTimer, "Pole still balanced after %0.1fs", t * T);
    SuccessCounter++;
    sprintf(Stt3,"%d",SuccessCounter);
    sprintf(Stt3,"Success      :%4d",SuccessCounter);
}
else{
    sprintf(FallTimer, "Pole fallen after %0.1fs", (t+1) * T);
    sprintf(Stt3,"Success      :%4d",SuccessCounter);
}
}
}
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4 การทดสอบโปรแกรม

บทนี้จะกล่าวถึงการทดสอบ โปรแกรมทั้งในส่วนของการแสดงผลและ โครงข่ายประสาทเทียมภาคการแสดงผลจะแบ่งการทดสอบโดยการสังเกตลักษณะต่าง ๆ ของเพนดูลัมแล้วเก็บค่าออกมาแสดงผล

การทดลองที่ 1 กราฟความสัมพันธ์ต่างของตัวแปร $x, \theta, \dot{\theta}$

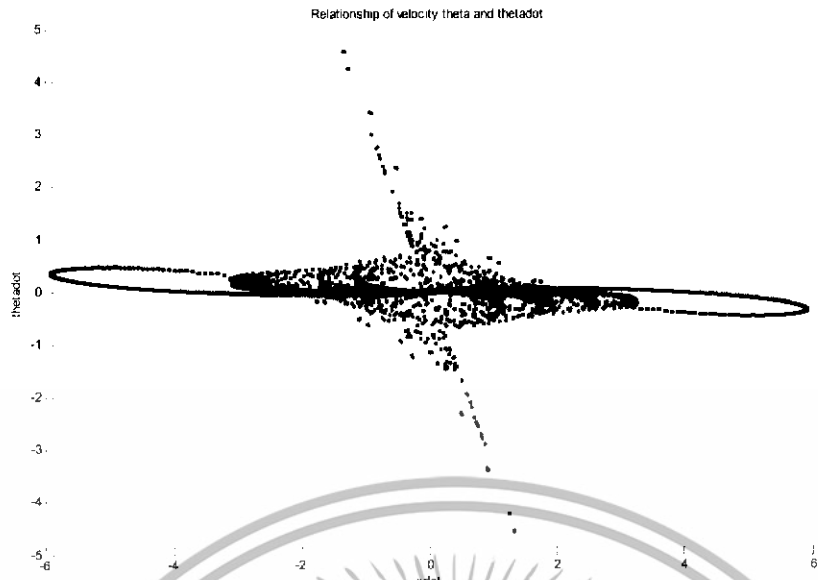
ทำการเก็บข้อมูลต่าง ๆ จากแบบจำลองที่สร้างขึ้นมาในบทที่ 3 ลงในตัวแปร $x, \dot{\theta}, \theta$ จะได้กราฟดังรูป



รูปที่ 4.1 แสดงความสัมพันธ์ระหว่าง ความเร็วเชิงเส้น มุม และความเร็วเชิงมุมของแบบจำลอง Inverted Pendulum

รูปที่ 4.2 แสดงความสัมพันธ์ของความเร็วจึงเส้นและมุมของแบบจำลอง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.3 แสดงความสัมพันธ์ระหว่างความเร็วเชิงเส้นและความเร็วเชิงมุม



รูปที่ 4.4 แสดงความสัมพันธ์ของมุมและความเร็วเชิงมุมของแบบจำลอง

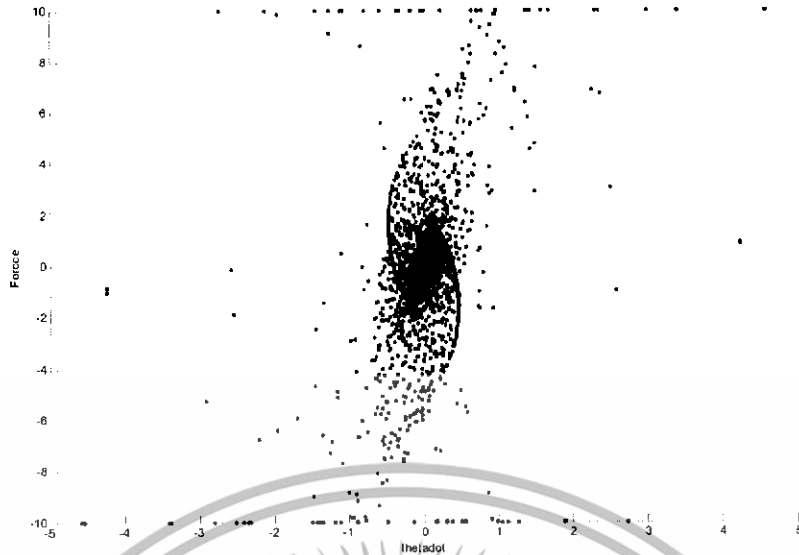
การทดลองที่ 1 ไม่ได้สื่อความหมายใด เพื่อการจัดหมู่ข้อมูลใดที่จะให้ประโยชน์ต่อการสร้างโครงข่ายประสาทเทียมแต่การทดลองนี้แสดงให้เห็นการพยายามรักษาระดับและมุมของการรักษาระดับซึ่งมีขนาดเล็กมาก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทดลองที่ 2 ความสัมพันธ์ระหว่าง f และ $\dot{x}, \dot{\theta}$



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



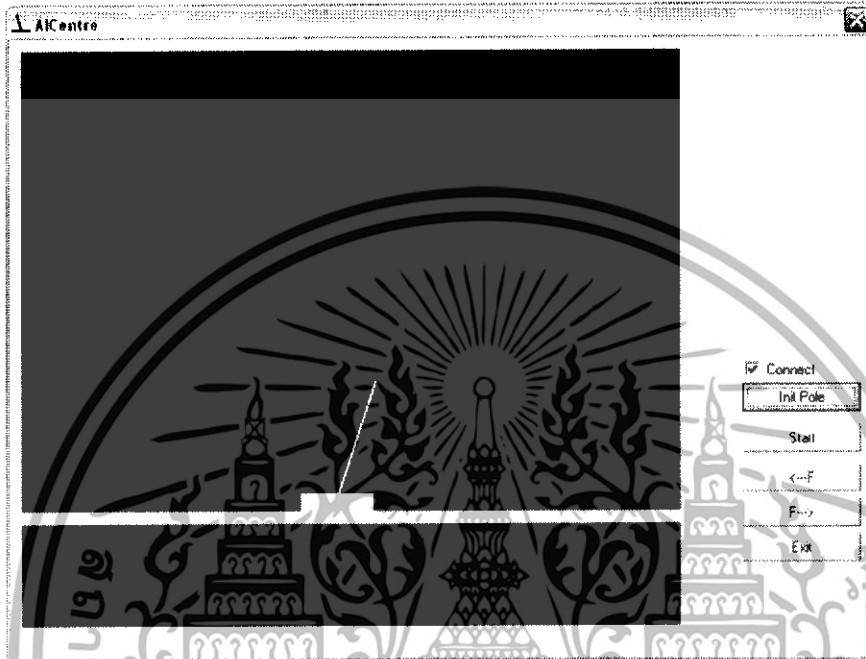
รูปที่ 4.7 แสดงความสัมพันธ์ระหว่างแรง และความเร็วเชิงมุม

ผลการทดลองที่ 2 แสดงความไม่เป็นเชิงเส้นอย่างสูงของระบบอินเวอร์ตเพนดูลัมเพื่อใช้
ทฤษฎีการป้อนกลับ PID และ LQR เป็นส่วนควบคุม จะเห็นว่าการเกาะกลุ่มกันของมุม ความเร็ว
เชิงมุม ความเร่งเชิงเส้น และ ระยะทาง ดังนั้นการทดลองส่วนนี้จะสัมพันธ์กับการแบ่งกลุ่มโดย
โครงข่ายประสาทเทียมแบบ Self-Organizing Fature Map ซึ่งสามารถหาเวกเตอร์ซึ่งใช้ชี้แบ่งแยก
กลุ่มของข้อมูลได้

การทดลองที่ 3 ทดสอบการแสดงผลการทำงานของระบบ

การทดลองที่ 3 แสดงการทำงานโดยรวมของโปรแกรม ซึ่งเขียนบน Visual C++ 6.0 มีคุณสมบัติ ดังนี้

1. สามารถแสดงผลตอบสนองของอินเวอร์ตเพนคูลัมได้
2. มีการเพิ่มแรงรบกวนระบบ
3. สามารถเริ่มต้นระบบในตำแหน่งเริ่มต้นได้



รูปที่ 4.8 แสดง หน้าต่างการทำงานของ โปรแกรม

การทดลองที่สามไม่สามารถเพิ่มโครงข่ายประสาทเทียมลงไปได้เนื่องจากผู้ทดลองไม่มีความชำนาญเกี่ยวกับวิธีการเรียกใช้ฟังก์ชันต่างๆ จึงทำได้เพียงการจำลองการเคลื่อนที่ ของระบบอินเวอร์ตเพนคูลัมซึ่งได้เขียนเงื่อนไขง่ายๆ ดังนี้

```
if(Pole.w>359) { Pole.w=0; Pole.xDot=-xDotOld; }
if(Pole.w<-359) { Pole.w=0; Pole.xDot=xDotOld; }
if(XPosition<-160){ XPosition=160; Pole.x=XPosition/100.0;}
if(XPosition>160) { XPosition=-160; Pole.x=XPosition/100.0;}
```

แต่การทรงตัวของระบบอินเวอร์ตเพนคูลัมทำได้ไม่นานนัก เพราะจะเกิดการออสซิลเลต ด้วยแรงที่มากขึ้นเรื่อยๆ จนกระทั่งไม่เข้าเงื่อนไขใดเลยก้านเพนคูลัมจึงตก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทดลองที่ 4 แสดงผลการตอบสนองของอินเวอร์ตเพนดูลัมบนภาษาซี

1. เขียนโปรแกรมดังนี้

```
#include<stdio.h>
#include<conio.h>

#include<math.h>
#include<stdlib.h>
typedef int    BOOL;
typedef int    INT;
typedef double REAL;

#define FALSE    0
#define TRUE     1
#define NOT      !
#define AND      &&
#define OR       ||

#define MIN_REAL    -HUGE_VAL
#define MAX_REAL    +HUGE_VAL
#define MIN(x,y)    ((x)<(y) ? (x) : (y))
#define MAX(x,y)    ((x)>(y) ? (x) : (y))

#define PI          (2*asin(1))
#define sqr(x)      ((x)*(x))

typedef struct {
    /* A LAYER OF A NET: */
    INT    Units; /* - number of units in this layer */
    REAL*  Output; /* - output of ith unit */
    REAL** Weight; /* - connection weights to ith unit */
    REAL*  StepSize; /* - size of search steps of ith unit */
    REAL*  dScoreMean; /* - mean score delta of ith unit */
} LAYER;

/* SIMULATION PARAMETERS FOR THE POLE: */
#define L    1 /* - length of pole [m] */
#define Mc   1 /* - mass of cart [kg] */
#define Mp   1 /* - mass of pole [kg] */
#define G    9.81 /* - acceleration due to gravity [m/s2] */
#define T    0.01 /* - time step [s] */
#define STEPS 10 /* - simulation steps in one time step */

typedef struct {
    /* STATE VARIABLES OF A POLE: */
    REAL    x; /* - position of cart [m] */
    REAL    xDot; /* - velocity of cart [m/s] */

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    REAL    w;        /* - angle of pole [°]          */
    REAL    wDot;     /* - angular velocity of pole [°/s]      */
    REAL    F;        /* - force applied to cart              */
} POLE;      /* during current time step [N]        */
void InitializeRandoms()
{
    srand(4715);
}
REAL RandomEqualREAL(REAL Low, REAL High)
{
    return ((REAL) rand() / RAND_MAX) * (High-Low) + Low;
}
REAL RandomNormalREAL(REAL Mu, REAL Sigma)
{
    REAL x,fx;

    do {
        x = RandomEqualREAL(Mu-3*Sigma, Mu+3*Sigma);
        fx = (1 / (sqrt(2*PI)*Sigma)) * exp(-sqr(x-Mu) / (2*sqr(Sigma)));
    } while (fx < RandomEqualREAL(0, 1));
    return x;
}
void InitializePole(POLE* Pole)
{
    do {
        Pole->x = 0;
        Pole->xDot = 0;
        Pole->w = RandomEqualREAL(-1, 1);
        Pole->wDot = 0;
        Pole->F = 0;
    } while (Pole->w == 0);
}
void SimulatePole(POLE* Pole)
{
    INT s;
    REAL x, xDot, xDotDot;
    REAL w, wDot, wDotDot;
    REAL F;

    x = Pole->x;
    xDot = Pole->xDot;
    w = (Pole->w / 180) * PI;
    wDot = (Pole->wDot / 180) * PI;
    F = Pole->F;
    for (s=0; s<STEPS; s++) {

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

wDotDot = (G*sin(w) + cos(w) * ((-F - Mp*L*sqr(wDot)*sin(w)) / (Mc+Mp))) /
(L * ((REAL) 4/3 - (Mp*sqr(cos(w))) / (Mc+Mp)));

xDotDot = (F + Mp*L * (sqr(wDot)*sin(w) - wDotDot*cos(w))) / (Mc+Mp);

x += (T / STEPS) * xDot;
xDot += (T / STEPS) * xDotDot;
w += (T / STEPS) * wDot;
wDot += (T / STEPS) * wDotDot;
}
Pole->x = x;
Pole->xDot = xDot;
Pole->w = (w / PI) * 180;
Pole->wDot = (wDot / PI) * 180;
}

BOOL PoleStillBalanced(POLE* Pole)
{
return (Pole->w >= -60) AND (Pole->w <= 60);
}
REAL ScoreOfPole(POLE* Pole)
{
return -sqr(Pole->w);
}

int main(){
POLE Pole;
FILE *f;
int i=0;
char ch;

f = fopen("SOM.txt", "w");

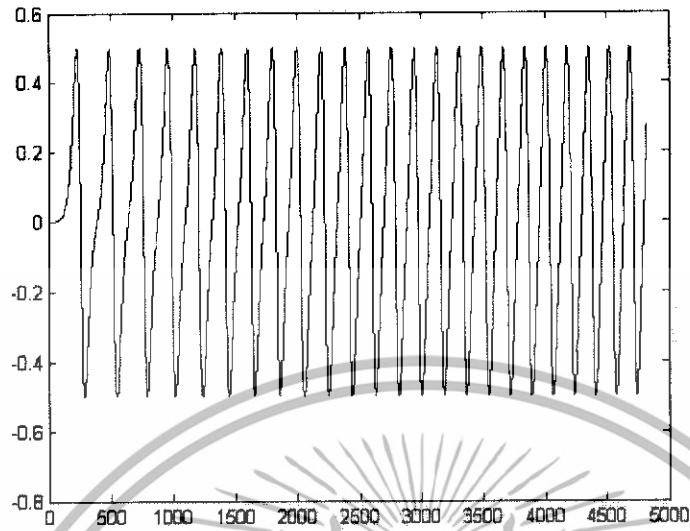
InitializePole(&Pole);
printf("Init Finish\n");

while(!kbhit()){
SimulatePole(&Pole);
fprintf(f,"%5.5f %5.5f %5.5f %5.5f %5.5f\n",Pole.x,Pole.xDot,Pole.w,Pole.wDot,Pole.F);
printf("%5.5f %5.5f %5.5f %5.5f %5.5f\n",Pole.x,Pole.xDot,Pole.w,Pole.wDot,Pole.F);
}
printf("Finish");
getch();
return 0;
}

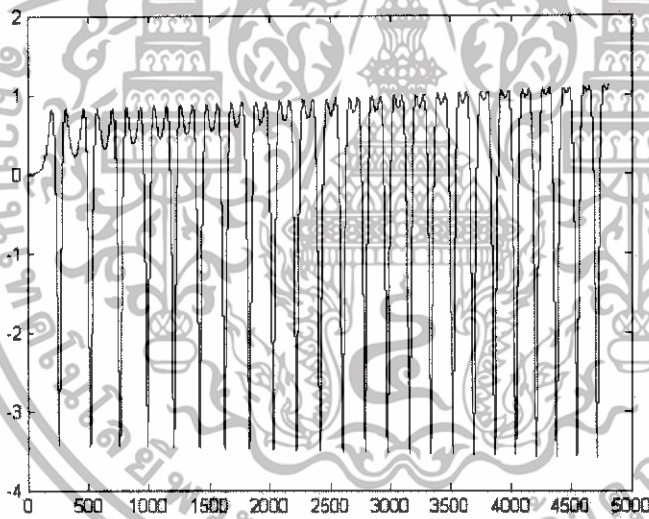
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2. คอมพิวเตอร์โปรแกรม ผลตอบสนองของเพนดูลัมจะออกมาที่ ไฟล์ som.txt
ได้ผลการทดลองดังนี้

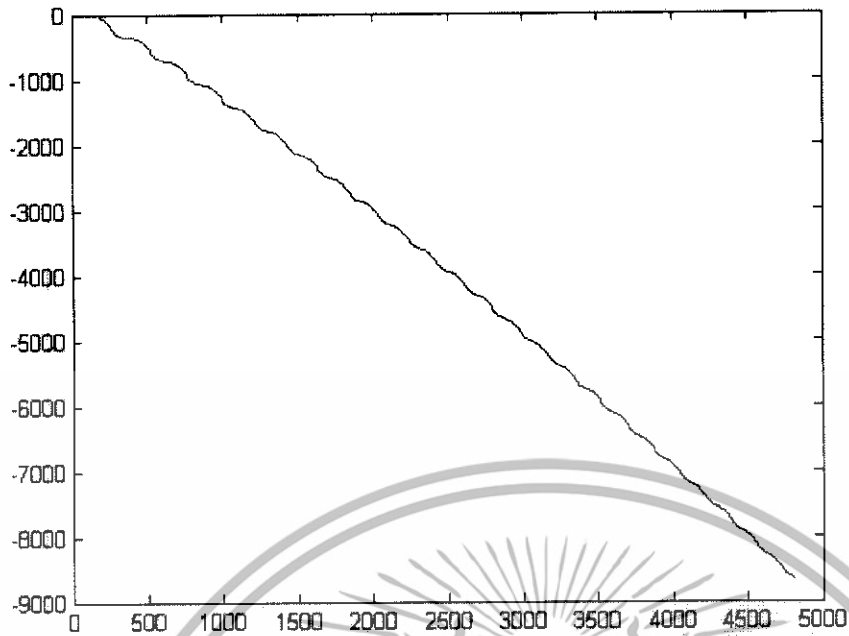


รูปที่ 4.9 แสดงผลตอบสนองตำแหน่งของฐาน

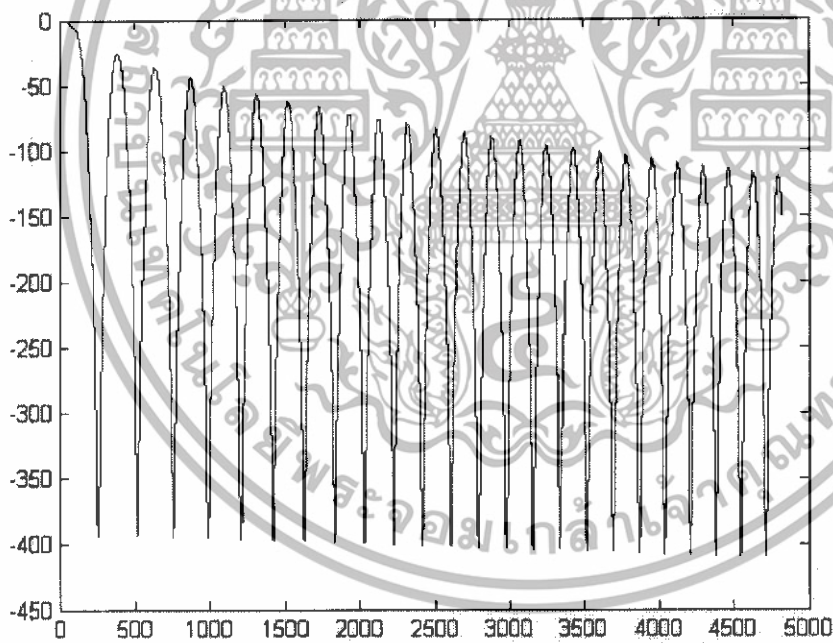


รูปที่ 4.10 แสดงผลตอบสนองทางความเร่งของฐาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



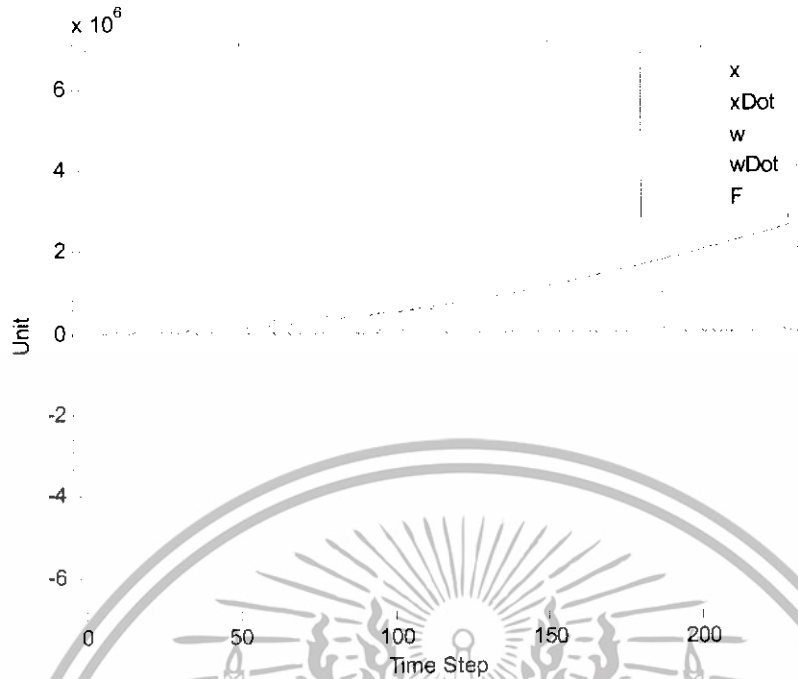
รูปที่ 4.11 แสดงผลตอบสนองต่อมุมของเพนดูลัม



รูปที่ 4.12 แสดงผลตอบสนองต่อความเร่งเชิงมุมของเพนดูลัม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทดลองที่ 5 ผลตอบสนองต่อ Impulse ของระบบ



รูปที่ 4.13 ผลตอบสนองต่อ impulse ของระบบใน Time Domain



รูปที่ 4.14 แสดงผลการทำงานของโครงข่ายประสาทเทียมเมื่อใช้ควบคุมการทรงตัวของเพนดูลัม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทดลองที่ 6 การควบคุมอินเวอร์ตเพนดูลัมด้วยโปรแกรมแมทแลป

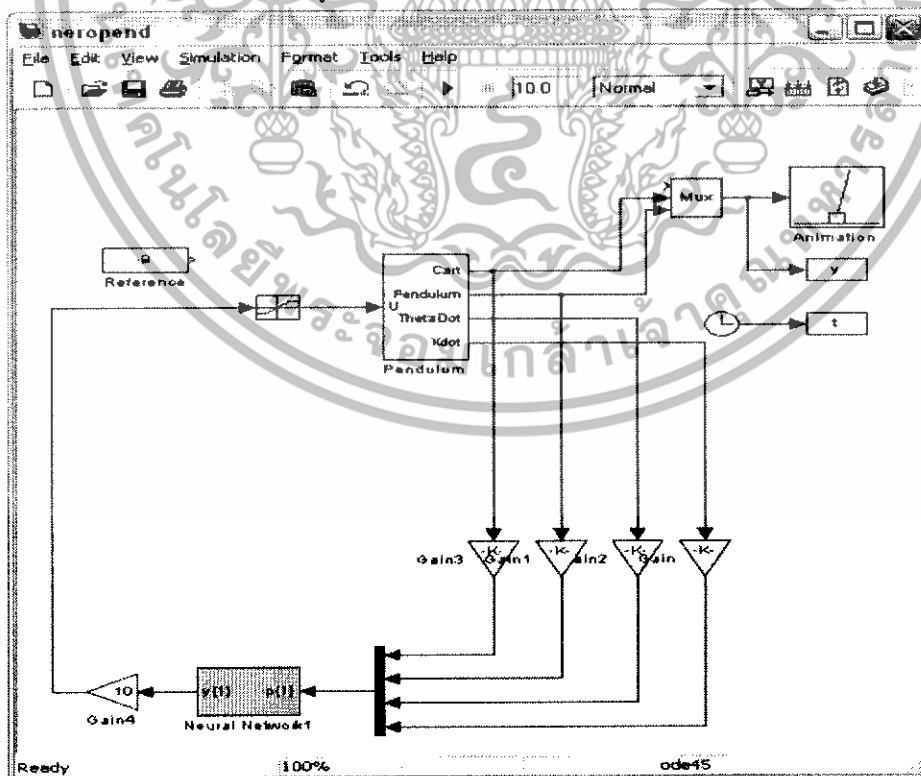
แบ่งการทดลองออกเป็นสองส่วนคือการทดลองการทรงตัวของเพนดูลัมและ การทดลองประสิทธิภาพของการทรงตัวของเพนดูลัมด้วยการใช้โครงข่ายประสาทเทียมหลายๆ แบบขนานกัน

การทดลองที่ 6.1 การสังเกตผลการทดลองการทรงตัวของเพนดูลัมด้วยโครงข่ายประสาทเทียม

การสังเกตว่าโครงข่ายประสาทเทียมสามารถควบคุมเพนดูลัมให้ไม่ตกได้ ใช้ช่วงเวลาหนึ่ง ถือว่าสามารถควบคุมได้ ดังรูปที่ 4.15 แสดงลักษณะของโครงข่ายประสาทเทียมแบบเรียงต่อขนาน (Casscade Forward) ซึ่งมีโครงสร้างดังนี้

```
net=newcfc(minmax(p),[8,40,20,10,1],{'tansig','tansig','tansig','tansig','purelin'},'trainlm');
net.trainParam.show=100;
net.trainParam.lr=0.01;
net.trainParam.mc=0.3;
net.trainParam.epochs=1000;
net.trainParam.goal=0.0001;
[net,f]=train(net,p,f);
a=sim(net,p);
gensim(net,1);
```

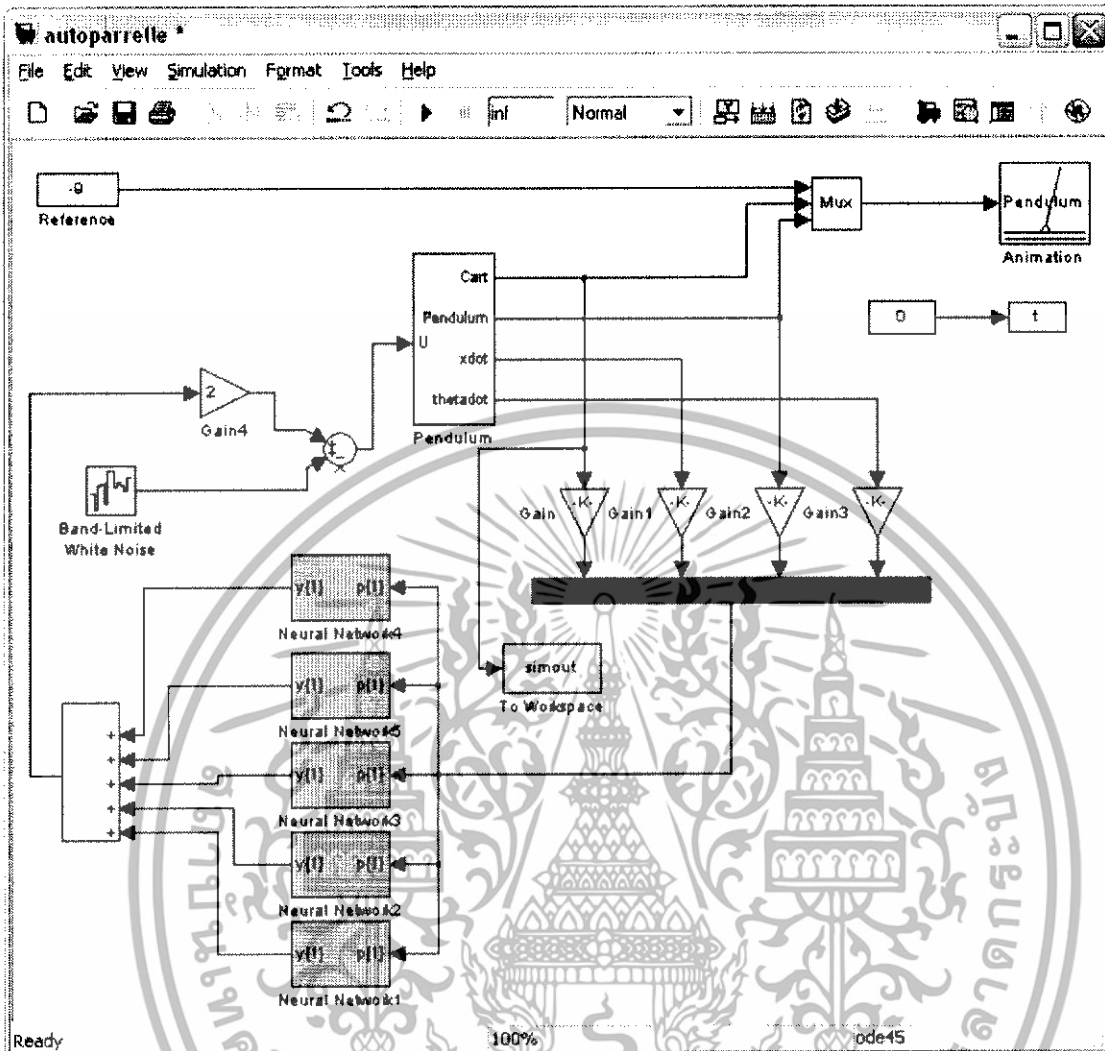
เป็นโครงข่ายประสาท 5 เลเยอร์ มีทรานเฟอร์ฟังก์ชันยกเว้นเอาต์พุตเป็น tansigmoid ส่วนเอาต์พุตเป็น Linear มีจำนวนประสาทเทียมในแต่ละเลเยอร์ดังนี้ 8 , 40 ,20 , 10 ,1 ซึ่งเป็นค่าจากการคาดเดาของผู้ทดลองเอง สามารถสร้างเป็นระบบได้ดังรูปที่ 4.15



รูปที่ 4.15 วิธีการทดลองการควบคุมอินเวอร์ตเพนดูลัม โดยโครงข่ายประสาทเทียม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทดลองที่ 6.2 แสดงให้เห็นคุณสมบัติ สำคัญบางประการของโครงข่ายประสาทเทียมนั่นคือการใช้โครงข่ายประสาทซึ่งเรียนรู้มาต่าง กรณีกัน ใช้ร่วมกัน โดยการเฉลี่ยค่าผลลัพธ์ของแต่ละ โครงข่ายประสาทร่วมกัน



รูปที่ 4.18 แสดงวิธีการเพิ่มประสิทธิภาพควบคุมอินเวอร์ตเพนดูลัม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5 บทสรุป

5.1 สรุปผล

การศึกษาระบบไม่เสถียรไม่เชิงเส้น และการควบคุมระบบดังกล่าวสามารถทำได้หลายวิธี เช่น ใช้ทฤษฎีการควบคุมหรือ ทฤษฎีการคำนวณแบบอ่อน(Soft Computing) เพื่ออำนวยความสะดวกในการทำงาน และลดต้นทุนได้บางส่วน การศึกษาทฤษฎี ควบคุมประยุกต์ เป็นทางเลือกอีกทาง เพื่อตอบปัญหาที่ไม่จำเป็น ต้องมีผู้เชี่ยวชาญเฉพาะทาง ซึ่งสามารถตั้งเพียงเป้าหมายของงานแล้วให้โปรแกรมประยุกต์จำลองและหาค่าตัวเอง

จากการจำลองและ ทดสอบทฤษฎี ควบคุมประยุกต์ ควบคุมระบบเชิงเส้นที่จำลองมาจากลักษณะทางกายภาพ แล้วอนุมานว่าผลลัพธ์ของลักษณะทางกายภาพที่เหมือนกันจะให้ผลลัพธ์เหมือนกันกับระบบประสาท ดังเช่นการใช้ ควบคุมประยุกต์ สำหรับการควบคุมการทรงตัวของอินเวอร์ตเพนดูลัม ควบคุมประยุกต์สามารถนำไปใช้ควบคุมการทรงตัวได้แต่ยังมีประสิทธิภาพต่ำ อาจเนื่องจากจำนวนประสาทเทียมหรือ จำนวนชั้นน้อยเกินไป หรืออาจเป็นเพราะเหตุการณ์ที่นำมาสอนให้กับโครงข่ายนั้นครอบคลุมกรณีต่างๆ ไม่เพียงพอ แต่สมมุติฐานข้อนี้มีน้ำหนักน้อยเนื่องจากขัดกับการใช้โครงข่ายประสาทแบบไม่มีต้นแบบ ซึ่งมีจำนวนประสาทเทียมน้อยกว่าโครงข่ายประสาทเทียมแบบมีต้นแบบมาก แต่สามารถทำงานได้ดีใกล้เคียงกับโครงข่ายประสาทเทียมแบบมีต้นแบบ ดังนั้นเรื่องของจำนวนประสาทในเลเยอร์มีผลไม่มากนัก

ข้อสังเกตเกี่ยวกับรูปร่าง หรือ ลักษณะโครงข่ายของโครงข่ายประสาทเทียมเป็นข้อที่น่าสนใจอีกข้อหนึ่ง ซึ่งทั้งการเรียนรู้แบบมีต้นแบบ และไม่มีต้นแบบใช้ลักษณะโครงข่ายคนละแบบกัน และลักษณะการเรียนรู้ต่างกัน ซึ่งการปรับน้ำหนักของโครงข่ายประสาทเทียมแบบไม่มีต้นแบบจะเป็นอิสระต่อข้อมูล เนื่องจากตัวโครงข่ายประสาทมีการสุ่มค่าเพื่อหาค่าแรงที่เหมาะสมเก็บไว้ได้ ต่างจากโครงข่ายประสาทแบบมีต้นแบบซึ่งมีอินพุตเป็นข้อมูลตายตัว จับคู่กับเอาต์พุตข้อมูลตายตัว ไม่สามารถสร้างเหตุการณ์ได้เอง

การนำโครงข่ายประสาทเทียมที่สร้างขึ้นมาไปควบคุมระบบจริงยังไม่สามารถทำได้เพราะการสุ่มสัญญาณยังมีความถี่ต่ำเกินไป คือ น้อยกว่า 200 *KHz* ผ่านการควบคุมผ่านทางพอร์ชานาน อาจแก้ไขได้โดยการติดต่อผ่านทางพอร์ตอื่น เช่น PCI หรือ USB อาจให้ความเร็วที่มากกว่า

โครงข่ายประสาทเทียมที่สร้างขึ้นสามารถใช้ในทางอิเล็กทรอนิกส์การแพทย์ซึ่งเป็นเป้าหมายสูงสุดคือการนำไปเรียนรู้คลื่นสัญญาณสมอง แล้วเรียนรู้ว่าสัญญาณแบบไหนควบคุมอวัยวะส่วนใด แล้วใช้คุณสมบัติที่ทดสอบมานำไปควบคุมอวัยวะส่วนนั้นโดยสามารถปรับวิธีการเรียนรู้ให้เหมาะสมกับแต่ละบุคคลได้

5.2 ปัญหาการดำเนินงาน

การใช้โครงข่ายประสาทเทียมต้องใช้หน่วยความจำมาก การเขียนทับตำแหน่งหน่วยความจำเป็นปัญหาที่พบบ่อยมาก

บรรณานุกรม

1. Khalil Sultan ,“ Inverted Pendulum Analysis Design and Implementation”IEEE,2003
2. Tim Callinan, “Artificial Neural Network identification and control of the inverted pendulum”, - .2003
3. Hagan Martin T., “Neural Network design”,- PWS ,1996
4. Hard S. Sutton and Andrew G. Barto., “Reinforcement Learning An Introduction”,- The MIT Press Cambridge, Messachusetts, 1998



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก

โปรแกรมจำลองการทำงานอินเวอร์ตเพนดูลัมบน วิชาวลซี พลัสพลัส 6

AICenterDialog.cpp

// AICentreDlg.cpp : implementation file

//

#include "windows.h"

#include "stdafx.h"

#include "AICentre.h"

#include "AICentreDlg.h"

#include "Serial.h"

#include "stdio.h"

#include "math.h"

#include <string.h>

#ifdef _DEBUG

#define new DEBUG_NEW

#undef THIS_FILE

static char THIS_FILE[] = __FILE__ ;

#endif

#define REAL double

#define INT int

#define PoleLenght 100

//#define PI 3.1428571428571428571428571428571

#define SerirL_Port 5

int SerialFlag =1;

int StartFlag =1;

int InitPoleFlag =1;

int FFlag =2;

double Force =0;

double Omega =0;

////////////////////////////////////

// Pendulum Property

////////////////////////////////////

#define L 1//.42 /* - length of pole [m] */

#define Mc 1.5//.5 /* - mass of cart [kg] */

#define Mp 0.6//.2 /* - mass of pole [kg] */

#define G 9.81 /* - acceleration due to gravity [m/s²] */

#define T 0.01 /* - time step [s] */

#define STEPS 1 /* - simulation steps in one time step */

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

typedef struct {
    /* STATE VARIABLES OF A POLE: */
    REAL x; /* - position of cart [m] */
    REAL xDot; /* - velocity of cart [m/s] */
    REAL w; /* - angle of pole [°] */
    REAL wDot; /* - angular velocity of pole [°/s] */
    REAL F; /* - force applied to cart */
} POLE; /* during current time step [N] */
////////////////////////////////////////////////////////////////

#define FALSE 0
#define TRUE 1
#define NOT !
#define AND &&
#define OR ||

#define MIN_REAL -HUGE_VAL
#define MAX_REAL +HUGE_VAL
#define MIN(x,y) ((x)<(y) ? (x) : (y))
#define MAX(x,y) ((x)>(y) ? (x) : (y))

#define PI (2*asin(1))
#define sqr(x) ((x)*(x))

typedef struct {
    /* A LAYER OF A NET: */
    INT Units; /* - number of units in this layer */
    REAL* Output; /* - output of ith unit */
    REAL** Weight; /* - connection weights to ith unit */
    REAL* StepSize; /* - size of search steps of ith unit */
    REAL* dScoreMean; /* - mean score delta of ith unit */
} LAYER;

typedef struct {
    /* A NET: */
    LAYER* InputLayer; /* - input layer */
    LAYER* KohonenLayer; /* - Kohonen layer */
    LAYER* OutputLayer; /* - output layer */
    INT Winner; /* - last winner in Kohonen layer */
    REAL Alpha; /* - learning rate for Kohonen layer */
    REAL Alpha_; /* - learning rate for output layer */
    REAL Alpha__; /* - learning rate for step sizes */
    REAL Gamma; /* - smoothing factor for score deltas */
    REAL Sigma; /* - parameter for width of neighborhood */
} NET;
////////////////////////////////////////////////////////////////
// Function Decaretion
void SimulatePole(POLE *Pole);
void InitializePole(POLE* Pole);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

REAL RandomEqualREAL(REAL Low, REAL High);
double ScoreOfPole(POLE* Pole);
BOOL PoleStillBalanced(POLE* Pole);
REAL RandomNormalREAL(REAL Mu, REAL Sigma);
void InitializeRandoms();
////////////////////////////////////////////////////////////////

#define ROWS      25
#define COLS      25

#define N         2
#define C         (ROWS * COLS)
#define M         1

#define TRAIN_STEPS 10000
#define BALANCED   100

FILE*      f;

void InitializeRandoms()
{
    srand(4715);
}

REAL RandomEqualREAL(REAL Low, REAL High)
{
    return ((double) rand() / RAND_MAX) * (High-Low) + Low;
}

REAL RandomNormalREAL(REAL Mu, REAL Sigma)
{
    REAL x,fx;

    do {
        x = RandomEqualREAL(Mu-3*Sigma, Mu+3*Sigma);
        fx = (1 / (sqrt(2*PI)*Sigma)) * exp(-(x-Mu)*(x-Mu) / (2*(Sigma)*(Sigma)));
    } while (fx < RandomEqualREAL(0, 1));
    return x;
}

void InitializePole(POLE* Pole)
{
    do {
        Pole->x = 0;
        Pole->xDot = 0;
        Pole->w = RandomEqualREAL(179,181);
        Pole->wDot = 0;
        Pole->F = 0;
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

} while (Pole->w == 0);
}

```

```

void SimulatePole(POLE *Pole) {
    INT s;
    REAL x, xDot, xDotDot;
    REAL w, wDot, wDotDot;
    REAL F;

```

```

    x = Pole->x;
    xDot = Pole->xDot;
    w = (Pole->w / 180) * PI;
    wDot = (Pole->wDot / 180) * PI;
    F = Pole->F;
    for (s=0; s<STEPS; s++) {

```

```

        wDotDot = (G*sin(w) + cos(w) * ((-F - Mp*L*(wDot)*(wDot)*sin(w)) /
(Mc+Mp))) /
        (L * ((REAL) 4/3 - (Mp*(cos(w))*cos(w))) / (Mc+Mp));

```

```

        xDotDot = (F + Mp*L * ((wDot)*(wDot)*sin(w) - wDotDot*cos(w))) / (Mc+Mp);

```

```

        x += (T / STEPS) * xDot;
        xDot += (T / STEPS) * xDotDot;
        w += (T / STEPS) * wDot;
        wDot += (T / STEPS) * wDotDot;
    }

```

```

    Pole->x = x;
    Pole->xDot = xDot;
    Pole->w = (w / PI) * 180;
    Pole->wDot = (wDot / PI) * 180;
}

```

```

BOOL PoleStillBalanced(POLE* Pole)
{
    return (Pole->w >= -60) && (Pole->w <= 60);
}

```

```

double ScoreOfPole(POLE* Pole)
{
    return -sqr(Pole->w);
}

```

```

UINT PendProcess(LPVOID pParam) {
    char stt[50];
    int i=0;
    double Angle=0;
    double XPosition=1;
    double wOld=0;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

double xDotOld=0;
// CSerial serial;
CAICentreDlg *dlg;
dlg = (CAICentreDlg *)pParam;
POLE Pole;
RECT rt;
PAINTSTRUCT ps;
CString str;

HPEN penField, penBk;

penField = CreatePen(0,2, RGB(255,255,255));
penBk = CreatePen(0,0, RGB(0,0,0));
InitializePole(&Pole);

NET Net;
//////////
//TrainNet(&Net);
//////////
while (1) {
    CDC *cdc = dlg->GetDC();
    dlg->BeginPaint(&ps);
    {
        while(StartFlag==1){
            // Draw Field
            cdc->FillSolidRect(10, 10, 550, 470, RGB(0,64,0));
            cdc->SelectObject(penField);
            // Draw a Rail
            rt.left = 10 + 30 + 245 - 275;//235
            rt.right = 10 + 30 + 245 + 275; //335
            rt.top = 10 + 30 + 360 - 5; //260
            rt.bottom = 10 + 30 + 360 + 5; //160
            cdc->Rectangle(&rt);
        }
        if(InitPoleFlag==0){
            InitializePole(&Pole);
            InitPoleFlag=0;
            Pole.x = 0;
            Pole.w = RandomEqualREAL(-1,1);
            Pole.xDot = 0;
            Pole.wDot = 0;
            Pole.F = 0;
            InitPoleFlag= 1;
        }
        cdc->FillSolidRect(10, 10, 550, 480, RGB(0,64,0));
        cdc->SelectObject(penField);
    }
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

*****
                                Disturb Condition
*****
/

if(FFlag==1){      Pole.F=400;      FFlag=2;      }
if(FFlag==0){      Pole.F=-400; FFlag=2;      }
/*****
*****
                                Get Pendulum Information
*****
/

SimulatePole(&Pole);
/*****
*****
                                Scale Value for Display
*****
/
XPosition=100*Pole.x;
Angle=Pole.w;

if(Pole.w>0&&Pole.w<=8 ){Pole.F=4;}
else if (Pole.w>12 && Pole.w<18 ){Pole.F=20;}
else if (Pole.w>18 && Pole.w<20){Pole.F=40;}
else if (Pole.w>20 && Pole.w<30){Pole.F=60;}

if(Pole.w<0&&Pole.w>-4){Pole.F=-4;}
else if (Pole.w<-12&&Pole.w>-18){Pole.F=-20;}
else if (Pole.w<-18 && Pole.w>-20){Pole.F=-40;}
else if (Pole.w<-20 && Pole.w>-30){Pole.F=-60;}
/*****
*****
                                Border Condition
*****
/
if(Pole.w>359) {      Pole.w=0;      Pole.xDot=-xDotOld;}
if(Pole.w<-359) {      Pole.w=0;      Pole.xDot=xDotOld;  }
if(XPosition<-160){      XPosition=160;
Pole.x=XPosition/100.0;}
if(XPosition>160) {      XPosition=-160;
Pole.x=XPosition/100.0;}

// Draw a Rail
rt.left = 10 + 30 + 245 - 275; //235
rt.right = 10 + 30 + 245 + 275 ; //335
rt.top = 10 + 30 + 360 - 5; //260

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

rt.bottom = 10 + 30 + 360 + 5;           //160
cdc->Rectangle(&rt);
// End Draw Rail
// Draw A Base
rt.left = 275-30*(-XPosition);
rt.right= 275+30*(-XPosition);
rt.top   = 380;
rt.bottom = 400;
cdc->Rectangle(&rt);
Pole.xDot=0.8*Pole.xDot;
Pole.x=0.9*Pole.x;
Pole.wDot=0.993*Pole.wDot;
SimulatePole(&Pole);
// Draw A Pole
cdc->MoveTo(275+XPosition,380);
cdc->LineTo((275+XPosition)+PoleLenght*sin(PI*Angle/180.00),340+60-
20-PoleLenght*cos(PI*Angle/180));

SimulatePole(&Pole);

dlg->EndPaint(&ps);
dlg->ReleaseDC(cdc);
Pole.F=0;
}
Sleep(15);
}
return 0;
}

class CAboutDlg : public CDialog
{
public:
    CAboutDlg();
// Dialog Data
//{{AFX_DATA(CAboutDlg)
enum { IDD = IDD_ABOUTBOX };
//}}AFX_DATA
// ClassWizard generated virtual function overrides
//{{AFX_VIRTUAL(CAboutDlg)
protected:
    virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV
support
    //}}AFX_VIRTUAL
// Implementation
protected:
    //{{AFX_MSG(CAboutDlg)
    //}}AFX_MSG

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
   //{{AFX_DATA_INIT(CAboutDlg)
   //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(CAboutDlg)
   //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
   //{{AFX_MSG_MAP(CAboutDlg)
    // No message handlers
   //}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CAICentreDlg dialog

CAICentreDlg::CAICentreDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CAICentreDlg::IDD, pParent)
{
   //{{AFX_DATA_INIT(CAICentreDlg)
    // NOTE: the ClassWizard will add member initialization here
   //}}AFX_DATA_INIT
    // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CAICentreDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(CAICentreDlg)
    // NOTE: the ClassWizard will add DDX and DDV calls here
   //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAICentreDlg, CDialog)
   //{{AFX_MSG_MAP(CAICentreDlg)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDOK, OnOk)
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

ON_WM_MOUSEMOVE()
ON_BN_CLICKED(IDC_BUTTON1, OnClickme)
ON_BN_CLICKED(IDC_CHECK1, OnCheck1)
ON_WM_LBUTTONDOWN()
ON_BN_CLICKED(IDC_BUTTON2, OnStart)
ON_BN_CLICKED(IDC_BUTTON3, OnInitPole)
ON_BN_CLICKED(IDC_BUTTON4, OnSerial1)
ON_BN_CLICKED(IDC_BUTTON5, OnCSerial)
//}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////
// CAICentreDlg message handlers

BOOL CAICentreDlg::OnInitDialog()
{
    CDialog::OnInitDialog();

    // Add "About..." menu item to system menu.

    // IDM_ABOUTBOX must be in the system command range.
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL)
    {
        CString strAboutMenu;
        strAboutMenu.LoadString(IDS_ABOUTBOX);
        if (!strAboutMenu.IsEmpty())
        {
            pSysMenu->AppendMenu(MF_SEPARATOR);
            pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX,
strAboutMenu);
        }
    }

    // Set the icon for this dialog. The framework does this automatically
    // when the application's main window is not a dialog
    SetIcon(m_hIcon, TRUE);           // Set big icon
    SetIcon(m_hIcon, FALSE);        // Set small icon

    // TODO: Add extra initialization here
    //NET* Net;
    AfxBeginThread(PendProcess, this);

    return TRUE; // return TRUE unless you set the focus to a control
}

void CAICentreDlg::OnSysCommand(UINT nID, LPARAM lParam)

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

{
    if ((nID & 0xFFF0) == IDM_ABOUTBOX)
    {
        CAboutDlg dlgAbout;
        dlgAbout.DoModal();
    }
    else
    {
        CDialog::OnSysCommand(nID, lParam);
    }
}
// If you add a minimize button to your dialog, you will need the code below
// to draw the icon. For MFC applications using the document/view model,
// this is automatically done for you by the framework.
void CAICentreDlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this); // device context for painting
        SendMessage(WM_ICONERASEBKGND, (WPARAM)
dc.GetSafeHdc(), 0);
        // Center icon in client rectangle
        int cxIcon = GetSystemMetrics(SM_CXICON);
        int cyIcon = GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialog::OnPaint();
    }
}
// The system calls this to obtain the cursor to display while the user drags
// the minimized window.
HCURSOR CAICentreDlg::OnQueryDragIcon()
{
    return (HCURSOR) m_hIcon;
}

void CAICentreDlg::OnClickme()
{
    // TODO: Add your control notification handler code here
    exit(0);
}

void CAICentreDlg::OnOk()

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

{
    // TODO: Add your control notification handler code here
}

void CAICentreDlg::OnMouseMove(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
}

void CAICentreDlg::OnCheck1()
{
    // TODO: Add your control notification handler code here

//    CSerial serial;
//    if (serial.Open(Serir1_Port, 38400))
//        AfxMessageBox("Port opened successfully");
//    else
//        AfxMessageBox("Failed to open port!");
}

void CAICentreDlg::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
    //CDialog::OnLButtonDown(nFlags, point);
}

void CAICentreDlg::OnStart()
{
    // TODO: Add your control notification handler code here
    StartFlag=~StartFlag;
}

void CAICentreDlg::OnInitPole()
{
    // TODO: Add your control notification handler code here
    InitPoleFlag=0;
}

void CAICentreDlg::OnSerial1()
{
    // TODO: Add your control notification handler code here
    FFlag=1;
}

void CAICentreDlg::OnCSerial()
{
    // TODO: Add your control notification handler code here
    FFlag=0;
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

AICenter.cpp

// AICentre.cpp : Defines the class behaviors for the application.

//

```
#include "stdafx.h"
#include "AICentre.h"
#include "AICentreDlg.h"
```

```
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
```

```
////////////////////////////////////
// CAICentreApp
```

```
BEGIN_MESSAGE_MAP(CAICentreApp, CWinApp)
   //{{AFX_MSG_MAP(CAICentreApp)
    // NOTE - the ClassWizard will add and remove mapping macros here.
    // DO NOT EDIT what you see in these blocks of generated code!
    //}}AFX_MSG
    ON_COMMAND(ID_HELP, CWinApp::OnHelp)
END_MESSAGE_MAP()
```

```
////////////////////////////////////
// CAICentreApp construction
```

```
CAICentreApp::CAICentreApp()
{
    // TODO: add construction code here,
    // Place all significant initialization in InitInstance
}
```

```
////////////////////////////////////
// The one and only CAICentreApp object
```

```
CAICentreApp theApp;
```

```
////////////////////////////////////
// CAICentreApp initialization
```

```
BOOL CAICentreApp::InitInstance()
{
    if (!AfxSocketInit())
    {
        AfxMessageBox(IDP_SOCKETS_INIT_FAILED);
        return FALSE;
    }
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    AfxEnableControlContainer();

#ifdef _AFXDLL
    Enable3dControls();           // Call this when using MFC in a shared
DLL
#else
    Enable3dControlsStatic();    // Call this when linking to MFC statically
#endif

    CAICentreDlg dlg;
    m_pMainWnd = &dlg;
    int nResponse = dlg.DoModal();
    if (nResponse == IDOK)
    {
        // TODO: Place code here to handle when the dialog is
        // dismissed with OK
    }
    else if (nResponse == IDCANCEL)
    {
        // TODO: Place code here to handle when the dialog is
        // dismissed with Cancel
    }

    // Since the dialog has been closed, return FALSE so that we exit the
    // application, rather than start the application's message pump.
    return FALSE;
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

Serial.cpp
// Serial.cpp

#include "stdafx.h"
#include "Serial.h"

CSerial::CSerial()
{
    memset( &m_OverlappedRead, 0, sizeof( OVERLAPPED ) );
    memset( &m_OverlappedWrite, 0, sizeof( OVERLAPPED ) );
    m_hIDComDev = NULL;
    m_bOpened = FALSE;
}

CSerial::~CSerial()
{
    Close();
}

BOOL CSerial::Open( int nPort, int nBaud )
{
    if( m_bOpened ) return( TRUE );

    char szPort[15];
    char szComParams[50];
    DCB dcb;

    wsprintf( szPort, "COM%d", nPort );
    m_hIDComDev = CreateFile( szPort, GENERIC_READ |
GENERIC_WRITE, 0, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL |
FILE_FLAG_OVERLAPPED, NULL );
    if( m_hIDComDev == NULL ) return( FALSE );

    memset( &m_OverlappedRead, 0, sizeof( OVERLAPPED ) );
    memset( &m_OverlappedWrite, 0, sizeof( OVERLAPPED ) );

    COMMTIMEOUTS CommTimeOuts;
    CommTimeOuts.ReadIntervalTimeout = 0xFFFFFFFF;
    CommTimeOuts.ReadTotalTimeoutMultiplier = 0;
    CommTimeOuts.ReadTotalTimeoutConstant = 0;
    CommTimeOuts.WriteTotalTimeoutMultiplier = 0;
    CommTimeOuts.WriteTotalTimeoutConstant = 5000;
    SetCommTimeouts( m_hIDComDev, &CommTimeOuts );

    wsprintf( szComParams, "COM%d:%d,n,8,1", nPort, nBaud );

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

m_OverlappedRead.hEvent = CreateEvent( NULL, TRUE, FALSE, NULL );
m_OverlappedWrite.hEvent = CreateEvent( NULL, TRUE, FALSE, NULL );

dcb.DCBlength = sizeof( DCB );
GetCommState( m_hIDComDev, &dcb );
dcb.BaudRate = nBaud;
dcb.ByteSize = 8;
unsigned char ucSet;
ucSet = (unsigned char) ( ( FC_RTSCTS & FC_DTRDSR ) != 0 );
ucSet = (unsigned char) ( ( FC_RTSCTS & FC_RTSCTS ) != 0 );
ucSet = (unsigned char) ( ( FC_RTSCTS & FC_XONXOFF ) != 0 );
if( !SetCommState( m_hIDComDev, &dcb ) ||
    !SetupComm( m_hIDComDev, 10000, 10000 ) ||
    m_OverlappedRead.hEvent == NULL ||
    m_OverlappedWrite.hEvent == NULL ){
    DWORD dwError = GetLastError();
    if( m_OverlappedRead.hEvent != NULL ) CloseHandle(
m_OverlappedRead.hEvent );
    if( m_OverlappedWrite.hEvent != NULL ) CloseHandle(
m_OverlappedWrite.hEvent );
    CloseHandle( m_hIDComDev );
    return( FALSE );
}

m_bOpened = TRUE;
return( m_bOpened );
}

BOOL CSerial::Close( void )
{
    if( !m_bOpened || m_hIDComDev == NULL ) return( TRUE );

    if( m_OverlappedRead.hEvent != NULL ) CloseHandle(
m_OverlappedRead.hEvent );
    if( m_OverlappedWrite.hEvent != NULL ) CloseHandle(
m_OverlappedWrite.hEvent );
    CloseHandle( m_hIDComDev );
    m_bOpened = FALSE;
    m_hIDComDev = NULL;

    return( TRUE );
}

BOOL CSerial::WriteCommByte( unsigned char ucByte )
{

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    BOOL bWriteStat;
    DWORD dwBytesWritten;

    bWriteStat = WriteFile( m_hIDComDev, (LPSTR) &ucByte, 1,
&dwBytesWritten, &m_OverlappedWrite );
    if( !bWriteStat && ( GetLastError() == ERROR_IO_PENDING ) ){
        if( WaitForSingleObject( m_OverlappedWrite.hEvent, 1000 ) )
dwBytesWritten = 0;
        else{
            GetOverlappedResult( m_hIDComDev, &m_OverlappedWrite,
&dwBytesWritten, FALSE );
            m_OverlappedWrite.Offset += dwBytesWritten;
        }
    }

    return( TRUE );
}

int CSerial::SendData( const char *buffer, int size )
{
    if( !m_bOpened || m_hIDComDev == NULL ) return( 0 );

    DWORD dwBytesWritten = 0;
    int i;
    for( i=0; i<size; i++){
        WriteCommByte( buffer[i] );
        dwBytesWritten++;
    }

    return( (int) dwBytesWritten );
}

int CSerial::ReadDataWaiting( void )
{
    if( !m_bOpened || m_hIDComDev == NULL ) return( 0 );

    DWORD dwErrorFlags;
    COMSTAT ComStat;

    ClearCommError( m_hIDComDev, &dwErrorFlags, &ComStat );

    return( (int) ComStat.cbInQue );
}

int CSerial::ReadData( void *buffer, int limit )

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

{
    if( !m_bOpened || m_hIDComDev == NULL ) return( 0 );

    BOOL bReadStatus;
    DWORD dwBytesRead, dwErrorFlags;
    COMSTAT ComStat;

    ClearCommError( m_hIDComDev, &dwErrorFlags, &ComStat );
    if( !ComStat.cbInQue ) return( 0 );

    dwBytesRead = (DWORD) ComStat.cbInQue;
    if( limit < (int) dwBytesRead ) dwBytesRead = (DWORD) limit;

    bReadStatus = ReadFile( m_hIDComDev, buffer, dwBytesRead,
&dwBytesRead, &m_OverlappedRead );
    if( !bReadStatus ){
        if( GetLastError() == ERROR_IO_PENDING ){
            WaitForSingleObject( m_OverlappedRead.hEvent, 2000 );
            return( (int) dwBytesRead );
        }
        return( 0 );
    }
    return( (int) dwBytesRead );
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ข

Som.Cpp

```

#include <time.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <graphics.h>
#include <dos.h>

typedef int     BOOL;
typedef int     INT;
typedef double  REAL;

#define FALSE    0
#define TRUE     1
#define NOT      !
#define AND      &&
#define OR       ||

#define MIN_REAL  -HUGE_VAL
#define MAX_REAL  +HUGE_VAL
#define MIN(x,y)  ((x)<(y) ? (x) : (y))
#define MAX(x,y)  ((x)>(y) ? (x) : (y))

#define PI        (2*asin(1))
#define sqr(x)    ((x)*(x))

#define PoleLenght 100
#define XConstance 10

#define RailHight 20
#define CartHight 20
#define CartWidth 40
#define DelayConstance 80
FILE*      f;
time_t first, second;
typedef struct { /* A LAYER OF A NET: */
    INT     Units; /* - number of units in this layer */
    REAL*   Output; /* - output of ith unit */
    REAL**  Weight; /* - connection weights to ith unit */
    REAL*   StepSize; /* - size of search steps of ith unit */
    REAL*   dScoreMean; /* - mean score delta of ith unit */
} LAYER;

typedef struct { /* A NET: */
    LAYER*   InputLayer; /* - input layer */
    LAYER*   KohonenLayer; /* - Kohonen layer */
    LAYER*   OutputLayer; /* - output layer */
    INT     Winner; /* - last winner in Kohonen layer */
    REAL    Alpha; /* - learning rate for Kohonen layer */

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

REAL    Alpha_;    /* - learning rate for output layer */
REAL    Alpha__;   /* - learning rate for step sizes */
REAL    Gamma;     /* - smoothing factor for score deltas */
REAL    Sigma;     /* - parameter for width of neighborhood */
} NET;

char Stt1[80],Stt2[80],Stt3[40];
char FallTimer[80];
float XPlotDistance;
int SuccessCounter=0;
void InitGraph(){
    int gdriver = DETECT, gmode, errorcode;
//    int left, top, right, bottom;

    initgraph(&gdriver, &gmode, "");

    errorcode = graphresult();
    if (errorcode != grOk) /* an error occurred */
    {
        printf("Graphics error: %s\n", grapherrormsg(errorcode));
        printf("Press any key to halt:");
        getch();
        exit(1);
    }
}
void Draw(double XPosition,double Angle,char stt[20]){

    double OldXPosition,OldAngle;
    double OldCartWidth,OldCartHight,OldPoleLenght,OldRailHight;
    setcolor(15);
    OldXPosition = XPosition;
    OldAngle = Angle;

    XPosition=getmaxx()/2-CartWidth/2+XPosition;
    // if(XPosition>getmaxx()){ XPosition=0;}
    // if(XPosition<0){XPosition=getmaxx();}
    // Draw a Cart and Pole
    rectangle(1,getmaxy()-RailHight,getmaxx()-1,getmaxy()-1);
    //Draw A Rail

    rectangle(XPosition,getmaxy()-RailHight-
    CartHight,XPosition+CartWidth,getmaxy()-CartHight); //Draw
    A Cart
    moveto(XPosition+CartWidth/2.0,getmaxy()-RailHight-CartHight);
    //Draw A Pole
    lineto(XPosition+CartWidth/2.0+PoleLenght*sin(PI*Angle/180.0),getmaxy()-
    RailHight-CartHight-PoleLenght*cos(PI*Angle/180.0)); //Draw A Pole

    outtextxy(10,getmaxy()-10,stt);
    outtextxy(1,1,FallTimer);
    outtextxy(1,20,Stt1);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

outtextxy(1,30,Stt2);
outtextxy(1,40,Stt3);

delay(DelayConstance-60);

//Clear a cart and Pole
setcolor(0);
OldXPosition=getmaxx()/2-CartWidth/2+OldXPosition;
rectangle(1,getmaxy()-RailHight,getmaxx()-1,getmaxy()-1);
//Draw A Rail
rectangle(OldXPosition,getmaxy()-RailHight-
CartHight,OldXPosition+CartWidth,getmaxy()-CartHight);
//Draw A Cart
moveto(OldXPosition+CartWidth/2.0,getmaxy()-RailHight-CartHight);
//Draw A Pole

lineto(OldXPosition+CartWidth/2.0+PoleLenght*sin(PI*OldAngle/180.0),getmaxy()-
RailHight-CartHight-PoleLenght*cos(PI*OldAngle/180.0)); //Draw A Pole
outtextxy(10,getmaxy()-10,Stt);
outtextxy(1,1,FallTimer);
outtextxy(1,20,Stt1);
outtextxy(1,30,Stt2);
outtextxy(1,40,Stt3);
// getch();
// cleardevice();
}

void ThetaGraph(double xDot, double wDot){
int YConst=0;
XPlotDistance=XPlotDistance+0.05;
setviewport(10,10, getmaxx()-10,getmaxy()/5,1);
rectangle(1,1,getmaxx()-20,getmaxy()/5-12);
line(1,YConst+(getmaxy()/5-12)/2.0,getmaxx()-20,(getmaxy()/5-12)/2.0);
// if(wDot>2000){YConst+20;}
putpixel(XPlotDistance,(getmaxy()/5-12)/2.0+wDot/5.0,4);
putpixel(XPlotDistance,(getmaxy()/5-12)/2.0+xDot*5,5);

setviewport(0,0,getmaxx(),getmaxy(),1);
// fprintf(f,"%lf %lf\n",xDot,wDot);
//cleardevice();
// closegraph();

}
/*****
*****
RANDOMS DRAWN FROM DISTRIBUTIONS
*****
*****/

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void InitializeRandoms()
```

```
{
  srand(4715);
}
```

```
REAL RandomEqualREAL(REAL Low, REAL High)
```

```
{
  return ((REAL) rand() / RAND_MAX) * (High-Low) + Low;
}
```

```
REAL RandomNormalREAL(REAL Mu, REAL Sigma)
```

```
{
  REAL x,fx;

  do {
    x = RandomEqualREAL(Mu-3*Sigma, Mu+3*Sigma);
    fx = (1 / (sqrt(2*PI)*Sigma)) * exp(-sqr(x-Mu) / (2*sqr(Sigma)));
  } while (fx < RandomEqualREAL(0, 1));
  return x;
}
```

```

/*****
*****
APPLICATION-SPECIFIC CODE
*****
*****/

```

```
#define ROWS 30
#define COLS 30
```

```
#define N 2
#define C (ROWS * COLS)
#define M 1 //1
```

```
#define TRAIN_STEPS 10000
#define BALANCED 1000
```

```
void InitializeApplication(NET* Net)
```

```
{
  INT i;

  for (i=0; i<Net->KohonenLayer->Units; i++) {
    Net->KohonenLayer->StepSize[i] = 1;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

Net->KohonenLayer->dScoreMean[i] = 0;
}
f = fopen("xwdot.txt", "w");
}

```

```

void WriteNet(NET* Net)

```

```

{
  INT r,c;
  REAL x,y,z;

  fprintf(f, "\n\n");
  for (r=0; r<ROWS; r++) {
    for (c=0; c<COLS; c++) {
      x = Net->KohonenLayer->Weight[r*ROWS+c][0];
      y = Net->KohonenLayer->Weight[r*ROWS+c][1];
      z = Net->OutputLayer->Weight[0][r*ROWS+c];
      fprintf(f, "([%5.1f°, %5.1f°], %5.1fN) ", x, y, z);
    }
    fprintf(f, "\n");
  }
}

```

```

void FinalizeApplication(NET* Net)

```

```

{
  fclose(f);
}

```

```

/*****
*****

```

SIMULATING THE POLE

```

*****
*****/

```

/* SIMULATION PARAMETERS FOR THE

```

POLE: */
#define L      1.2//1      /* - length of pole [m]          */
#define Mc     1.5//1      /* - mass of cart [kg]           */
#define Mp     0.6//1      /* - mass of pole [kg]          */
#define G      9.81        /* - acceleration due to gravity [m/s²] */
#define T      0.1         /* - time step [s]              */
#define STEPS  10         /* - simulation steps in one time step */

```

```

typedef struct {          /* STATE VARIABLES OF A POLE:    */
  REAL    x;             /* - position of cart [m]         */
  REAL    xDot;         /* - velocity of cart [m/s]      */
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    REAL    w;        /* - angle of pole [°]          */
    REAL    wDot;     /* - angular velocity of pole [°/s]    */
    REAL    F;        /* - force applied to cart             */
} POLE;          /* during current time step [N]      */

```

```
void InitializePole(POLE* Pole)
```

```

{
  do {
    Pole->x = 0;
    Pole->xDot = 0;
    Pole->w = RandomEqualREAL(-1,1);
    Pole->wDot = 0;
    Pole->F = 0;
  } while (Pole->w == 0);
}

```

```
void SimulatePole(POLE* Pole)
```

```

{
  INT s;
  REAL x, xDot, xDotDot;
  REAL w, wDot, wDotDot;
  REAL F;

  x = Pole->x;
  xDot = Pole->xDot;
  w = (Pole->w / 180) * PI;
  wDot = (Pole->wDot / 180) * PI;
  F = Pole->F;
  for (s=0; s<STEPS; s++) {

    wDotDot = (G*sin(w) + cos(w) * ((-F - Mp*L*sqr(wDot)*sin(w)) / (Mc+Mp))) /
              (L * ((REAL) 4/3 - (Mp*sqr(cos(w))) / (Mc+Mp)));

    xDotDot = (F + Mp*L * (sqr(wDot)*sin(w) - wDotDot*cos(w))) / (Mc+Mp);

    x += (T / STEPS) * xDot;
    xDot += (T / STEPS) * xDotDot;
    w += (T / STEPS) * wDot;
    wDot += (T / STEPS) * wDotDot;
  }
  Pole->x = x;
  Pole->xDot = xDot;
  Pole->w = (w / PI) * 180;
  Pole->wDot = (wDot / PI) * 180;
}

```

```
BOOL PoleStillBalanced(POLE* Pole)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

{
return (Pole->w >= -60) AND (Pole->w <= 60);
}

```

```

REAL ScoreOfPole(POLE* Pole)

```

```

{
return -sqr(Pole->w);
}

```

```

/*****
*****

```

INITIALIZATION

```

*****
*****/

```

```

void GenerateNetwork(NET* Net)

```

```

{
INT i;

Net->InputLayer = (LAYER*) malloc(sizeof(LAYER));
Net->KohonenLayer = (LAYER*) malloc(sizeof(LAYER));
Net->OutputLayer = (LAYER*) malloc(sizeof(LAYER));

Net->InputLayer->Units = N;
Net->InputLayer->Output = (REAL*) calloc(N, sizeof(REAL));

Net->KohonenLayer->Units = C;
Net->KohonenLayer->Output = (REAL*) calloc(C, sizeof(REAL));
Net->KohonenLayer->Weight = (REAL**) calloc(C, sizeof(REAL*));
Net->KohonenLayer->StepSize = (REAL*) calloc(C, sizeof(REAL));
Net->KohonenLayer->dScoreMean = (REAL*) calloc(C, sizeof(REAL));

Net->OutputLayer->Units = M;
Net->OutputLayer->Output = (REAL*) calloc(M, sizeof(REAL));
Net->OutputLayer->Weight = (REAL**) calloc(M, sizeof(REAL*));

for (i=0; i<C; i++) {
Net->KohonenLayer->Weight[i] = (REAL*) calloc(N, sizeof(REAL));
}
for (i=0; i<M; i++) {
Net->OutputLayer->Weight[i] = (REAL*) calloc(C, sizeof(REAL));
}
}

```

```

void RandomWeights(NET* Net)

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

{
  INT i,j;

  for (i=0; i<Net->KohonenLayer->Units; i++) {
    for (j=0; j<Net->InputLayer->Units; j++) {
      Net->KohonenLayer->Weight[i][j] = RandomEqualREAL(-30, 30);
    }
  }
  for (i=0; i<Net->OutputLayer->Units; i++) {
    for (j=0; j<Net->KohonenLayer->Units; j++) {
      Net->OutputLayer->Weight[i][j] = 0;
    }
  }
}

```

```

void SetInput(NET* Net, REAL* Input)

```

```

{
  INT i;

  for (i=0; i<Net->InputLayer->Units; i++) {
    Net->InputLayer->Output[i] = Input[i];
  }
}

```

```

void GetOutput(NET* Net, REAL* Output)

```

```

{
  INT i;

  for (i=0; i<Net->OutputLayer->Units; i++) {
    Output[i] = Net->OutputLayer->Output[i];
  }
}

```

```

/*****
*****

```

PROPAGATING SIGNALS

```

*****
*****/

```

```

void PropagateToKohonen(NET* Net)

```

```

{
  INT i,j;
  REAL Out, Weight, Sum, MinOut;

  for (i=0; i<Net->KohonenLayer->Units; i++) {

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

Sum = 0;
for (j=0; j<Net->InputLayer->Units; j++) {
    Out = Net->InputLayer->Output[j];
    Weight = Net->KohonenLayer->Weight[i][j];
    Sum += sqr(Out - Weight);
}
Net->KohonenLayer->Output[i] = sqrt(Sum);
}
MinOut = MAX_REAL;
for (i=0; i<Net->KohonenLayer->Units; i++) {
    if (Net->KohonenLayer->Output[i] < MinOut)
        MinOut = Net->KohonenLayer->Output[Net->Winner = i];
}
}

void PropagateToOutput(NET* Net)
{
    INT i;

    for (i=0; i<Net->OutputLayer->Units; i++) {
        Net->OutputLayer->Output[i] = Net->OutputLayer->Weight[i][Net->Winner];
    }
}

void PropagateNet(NET* Net)
{
    PropagateToKohonen(Net);
    PropagateToOutput(Net);
}

/*****
*****
TRAINING THE NET
*****
*****/

REAL Neighborhood(NET* Net, INT i)
{
    INT iRow, iCol, jRow, jCol;
    REAL Distance;

    iRow = i / COLS; jRow = Net->Winner / COLS;
    iCol = i % COLS; jCol = Net->Winner % COLS;

    Distance = sqrt(sqr(iRow-jRow) + sqr(iCol-jCol));
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

return exp(-sqr(Distance) / (2*sqr(Net->Sigma)));
}

void TrainKohonen(NET* Net, REAL* Input)
{
    INT i,j;
    REAL Out, Weight, Lambda, StepSize;

    for (i=0; i<Net->KohonenLayer->Units; i++) {
        for (j=0; j<Net->InputLayer->Units; j++) {
            Out = Input[j];
            Weight = Net->KohonenLayer->Weight[i][j];
            Lambda = Neighborhood(Net, i);
            Net->KohonenLayer->Weight[i][j] += Net->Alpha * Lambda * (Out - Weight);
        }
        StepSize = Net->KohonenLayer->StepSize[i];
        Net->KohonenLayer->StepSize[i] += Net->Alpha * Lambda * -StepSize;
    }
}

void TrainOutput(NET* Net, REAL* Output)
{
    INT i,j;
    REAL Out, Weight, Lambda;

    for (i=0; i<Net->OutputLayer->Units; i++) {
        for (j=0; j<Net->KohonenLayer->Units; j++) {
            Out = Output[i];
            Weight = Net->OutputLayer->Weight[i][j];
            Lambda = Neighborhood(Net, j);
            Net->OutputLayer->Weight[i][j] += Net->Alpha * Lambda * (Out - Weight);
        }
    }
}

void TrainUnits(NET* Net, REAL* Input, REAL* Output)
{
    TrainKohonen(Net, Input);
    TrainOutput(Net, Output);
}

void TrainNet(NET* Net)
{
    INT n,t;
    POLE Pole;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

REAL wOld, wNew, ScoreOld, ScoreNew, dScore, dScoreMean, StepSize;
REAL Input[N];
REAL Output[M];
REAL Target[M];

double StartTime, StopTime;
struct dostime_ t tt;
char Stt[100];
char ch;
n = 0;
// _dos_gettime(&t);

while (n<TRAIN_STEPS) {
// first = time(NULL); /* Gets system
// time */

t = 0;
InitializePole(&Pole);

wOld = Pole.w;
ScoreOld = ScoreOfPole(&Pole);
SimulatePole(&Pole);
wNew = Pole.w;
ScoreNew = ScoreOfPole(&Pole);
while (PoleStillBalanced(&Pole) AND (t<BALANCED)) {

_dos_gettime(&tt);
StartTime=tt.hsecond;

n++;
t++;
Net->Alpha = 0.5 * pow(0.01, (REAL) n / TRAIN_STEPS);
Net->Alpha_ = 0.5 * pow(0.01, (REAL) n / TRAIN_STEPS);
Net->Alpha__ = 0.005;
Net->Gamma = 0.05;
Net->Sigma = 6.0 * pow(0.2, (REAL) n / TRAIN_STEPS);
Input[0] = wOld;
Input[1] = wNew;
SetInput(Net, Input);
PropagateNet(Net);
GetOutput(Net, Output);
Pole.F = Output[0];

StepSize = Net->KohonenLayer->StepSize[Net->Winner];
Pole.F += StepSize * RandomNormalREAL(0, 10);

printf(Stt,"Neural Network Control An Inverted Pendulum.Iteration=%5.2d (ESC
Exit) ",n);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

// fprintf(f,"%d\n",n);
Draw(Pole.x*XConstance,Pole.w,Stt);
// ThetaGraph(Pole.xDot,Pole.wDot);

Pole.xDot = 0.94*Pole.xDot;
Pole.w = 0.96*Pole.w;
wOld = Pole.w;
ScoreOld = ScoreOfPole(&Pole);

SimulatePole(&Pole);

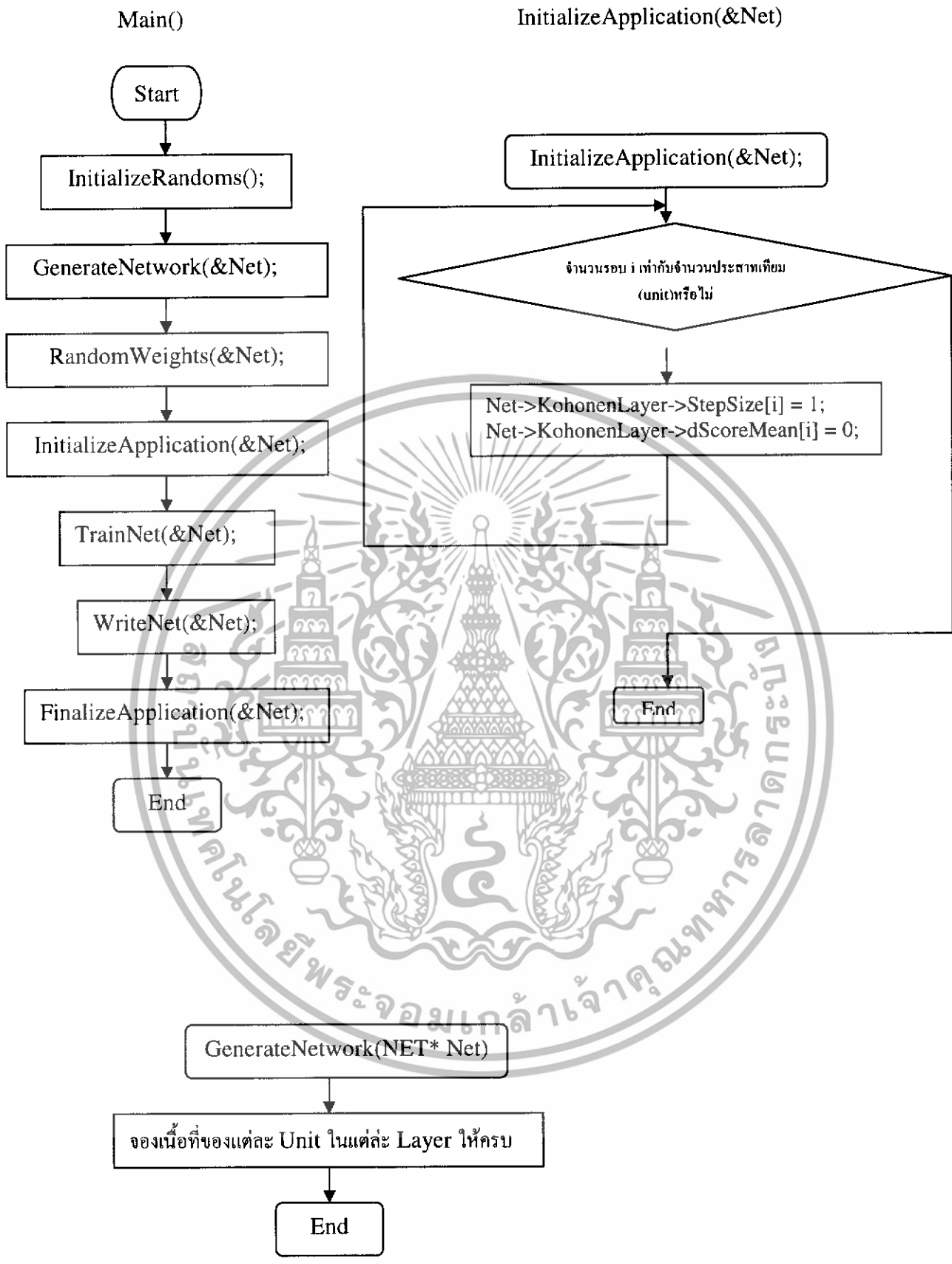
wNew = Pole.w;
ScoreNew = ScoreOfPole(&Pole);
dScore = ScoreNew - ScoreOld;
dScoreMean = Net->KohonenLayer->dScoreMean[Net->Winner];
sprintf(Stt1,"Pole's Angle(Degree) : %5.4f",Pole.w);

sprintf(Stt2,"Force (N) : %5.4f",Pole.F);
while(kbhit()){
    ch=getch();
    switch(ch){
        case 27:
            closegraph();
            FinalizeApplication(&Net);
            exit(0);
            break;
    }
}
if (dScore > dScoreMean) {
    Target[0] = Pole.F;
    TrainUnits(Net, Input, Target);
}
Net->KohonenLayer->dScoreMean[Net->Winner] += Net->Gamma * (dScore -
dScoreMean);
}

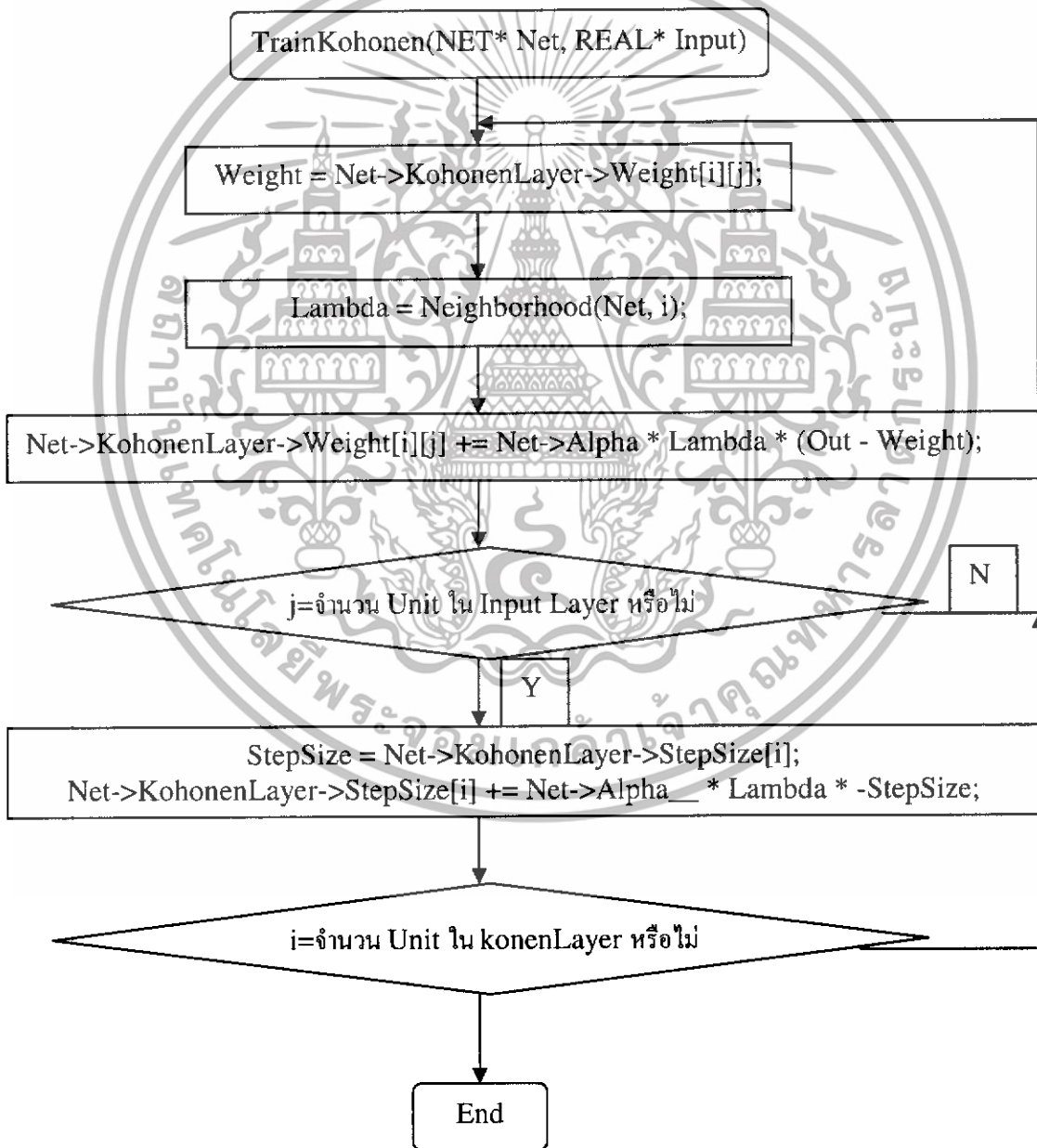
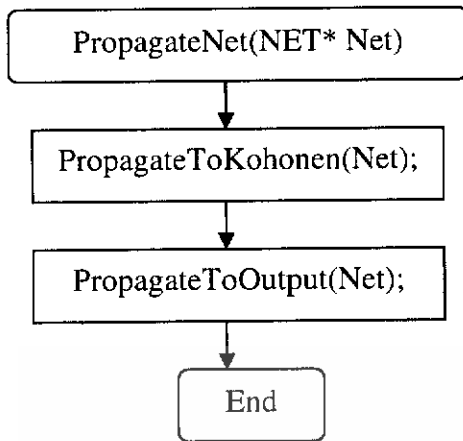
if (PoleStillBalanced(&Pole)){
    sprintf(FallTimer, "Pole still balanced after %0.1fs", t * T);
    SuccessCounter++;
    sprintf(Stt3,"%d",SuccessCounter);
    sprintf(Stt3,"Success :%4d",SuccessCounter);
}
else{
    sprintf(FallTimer, "Pole fallen after %0.1fs", (t+1) * T);
    //SuccessCounter++;
    sprintf(Stt3,"Success :%4d",SuccessCounter);
}
}

```

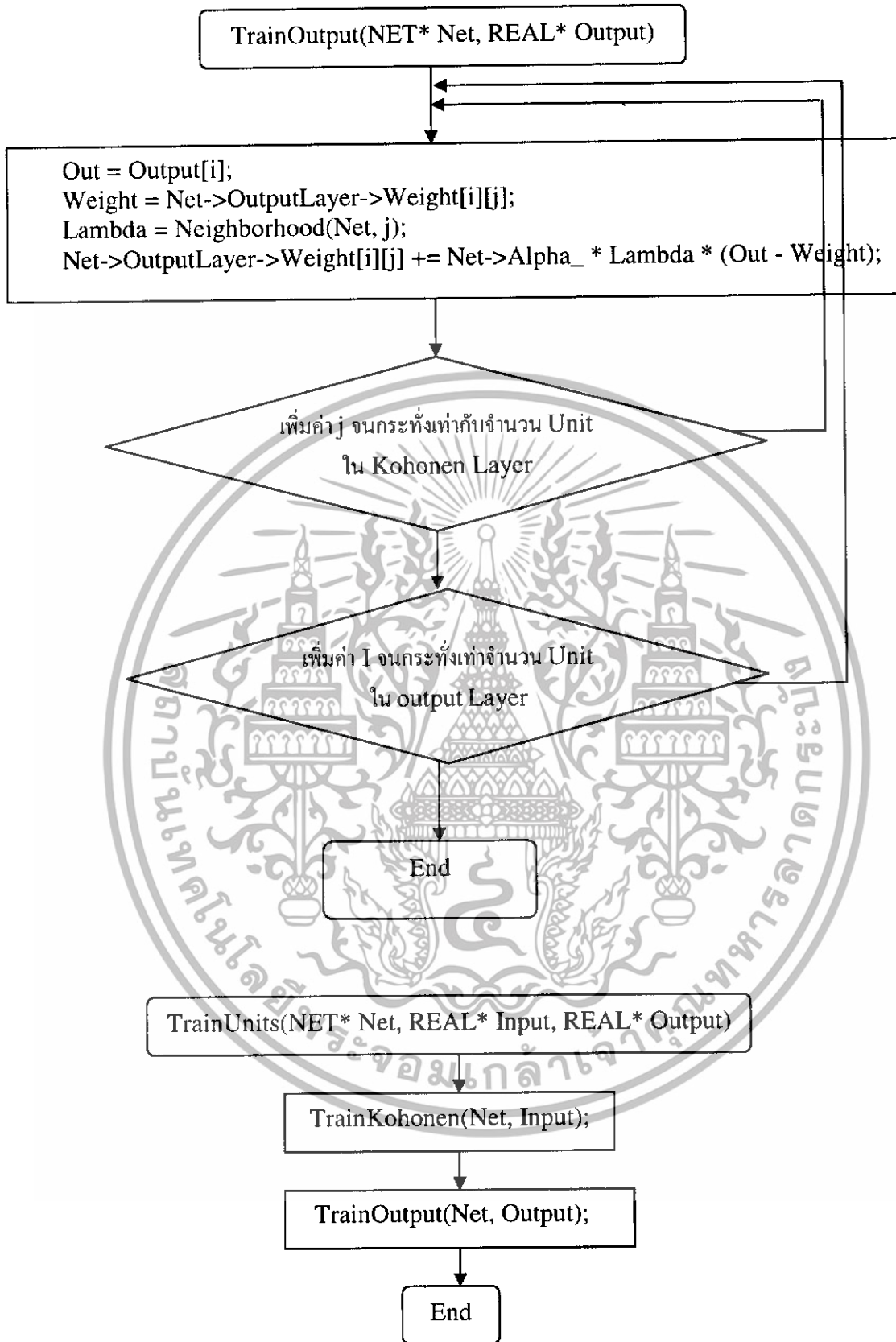
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่าจะกรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้