การเรียนรู้ของโครงข่ายประสาทเทียม PID สำหรับระบบควบคุมที่ไม่เป็นเชิงเส้นแบบปรับตัวได้

LEARNING ALGORITHM OF NEURAL NETWORK PID FOR ADAPTIVE NONLINEAR CONTROL SYSTEM

อัดนา เซนโต๊ะ

ADNA SENTO

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2561

KMITL-2018-EN-D-018-170

การเรียนรู้ของโครงข่ายประสาทเทียม PID สำหรับระบบควบคุมที่ไม่เป็นเชิงเส้นแบบปรับตัวได้

LEARNING ALGORITHM OF NEURAL NETWORK PID FOR ADAPTIVE NONLINEAR CONTROL SYSTEM

อัดนา เซนโต๊ะ

ADNA SENTO

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรดุษฎีบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2561

KMITL-2018-EN-D-018-170

LEARNING ALGORITHM OF NEURAL NETWORK PID FOR ADAPTIVE NONLINEAR CONTROL SYSTEM

ADNA SENTO

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

DOCTOR OF ENGINEERING IN ELECTRICAL ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2018

KMITL-2018-EN-D-018-170

หัวข้อวิทยานิพนธ์	การเรียนรู้ของโครงข่ายประสาทเทียม PID สำหรับระบบควบคุมที่ไม่เป็นเชิงเส้นแบบปรับตัวได้
ชื่อนักศึกษา	นายอัดนา เซนโต๊ะ
รหัสประจำตัว	57601018
ปริญญา	วิศวกรรมศาสตรดุษฎีบัณฑิต
สาขาวิชา	วิศวกรรมไฟฟ้า
พ.ศ.	2561
อาจารย์ที่ปรึกษาวิทยานิพนธ์	ผศ.ดร. ยุทธนา คิดใจเดียว

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้นำเสนอตัวต้นแบบใหม่ของตัวควบคุม Neural Network PID-Like (NNPID) ซึ่งเป็นตัวควบคุมที่สร้างมาจากโครงข่ายประสาทเทียมโดยเลียนแบบโครงสร้างของตัวควบคุม Proportional-Integral-Derivative (หรือเรียกว่า PID) เนื่องด้วยตัวควบคุมที่นำเสนอนี้มีโครงสร้างเป็นโครงข่ายประสาทเทียมจึงทำให้ประสิทธิภาพของระบบควบคุมขึ้นอยู่กับการปรับค่าน้ำหนัก เรียกว่าอัลกอริทึมการเรียนรู้ (Learning Algorithm) อันส่งผลทำให้มีนักวิจัยมากมายได้พัฒนาตัวควบคุมชนิดนี้ด้วยการคิดค้นอัลกอริทึมการเรียนรู้ซึ่งทำให้การปรับน้ำหนักของโครงข่ายเป็นไปอย่างมีประสิทธิภาพ แต่อย่างไรก็ตามตัวควบคุมนี้ยังมีข้อจำกัดบางประการ เช่นการตั้งค่าน้ำหนักเริ่มต้น การปรับตัวต่อปัจจัยแวดล้อมกระทันหัน ความผิดพลาดของระบบ หรือแม้กระทั่งประสิทธิภาพเองก็ยังต้องได้รับการพัฒนาต่อไป เป็นต้น ดังนั้นวิทยานิพนธ์ฉบับนี้ไม่ได้เพียงแค่นำเสนอตัวต้นแบบของตัวควบคุมเท่านั้น แต่ยังนำเสนออัลกอริทึมการเรียนรู้ใหม่เพื่อปรับปรุงระบบควบคุมให้ดียิ่งขึ้นไปอีก จากผลการดำเนินงานของงานวิจัยในครั้งนี้ ผู้วิจัยได้สร้างตัวต้นแบบใหม่ของตัวควบคุมแบบโครงข่ายประเทียมที่เลียนแบบโครงสร้างของ PID ทั้งหมด 4 แบบดังนี้ แบบที่หนึ่งตัวควบคุม ENNPID ซึ่งเป็นตัวควบคุมที่ใช้อัลกอริทึมการเรียนรู้แบบส่วนขยายของตัวกรองคาลมาน แบบที่สองตัวควบคุมที่ใช้อัลกอริทึมการเรียนรู้แบบไฮบริดของตัวกรองคาลมาน เรียกว่าตัวควบคุม CKF-NNPID แบบที่สามตัวควบคุมที่ใช้อัลกอริทึมการเรียนรู้เบย์แบบประยุกต์ เรียกตัวควบคุม ABF-NNPID และแบบสุดท้ายตัวควบคุมที่ใช้อัลกอริทึมการเรียนรู้แบบ Actor-Critic เรียกว่าตัวควบคุม NNPID-AC เพื่อประเมินประสิทธิภาพของตัวควบคุมที่นำเสนอ เราได้สร้างแบบจำลองของระบบควบคุมแขนกลไฟฟ้าแบบปิดพร้อมด้วยการติดที่ตัวควบคุมไว้ตามข้อต่อต่าง ๆ ของแขนกลพร้อมกันนี้ได้ทดสอบด้วยการเพิ่มโหลด และปัจจัยแวดล้อมที่รบกวนเพื่อพิสูจน์ความทนทานต่อข้อผิดพลาด ซึ่งจากผลการจำลองระบบควบคุมแขนกลไฟฟ้าภายใต้การควบคุมของตัวควบคุมที่นำเสนอพบว่าสามารถปรับค่าน้ำหนักที่เหมาะสมส่งผลทำให้มีตอบสนองที่ดีที่สุดเมื่อเทียบกับตัวควบคุมก่อนหน้าในสถานะการณ์ทั้งมีโหลด ไม่มีโหลด หรือปัจจัยแวดล้อมที่รบกวนอื่น ๆ

Thesis	Learning Algorithm of Neural Network PID for Adaptive Nonlinear Control System
Student	Mr. Adna Sento
Student ID.	57601018
Degree	Doctor of Engineering
Program	Electrical Engineering
Year	2018
Thesis Advisor	Asst.Prof. Dr. Yuttana Kitjaidure

ABSTRACT

This thesis presents a new model of the Neural Network PID-Like controller (NNPID), which is formed by the neural network based on the classical PID controller. Since the proposed NNPID controller is designed by the neural network, the control system performance is depended on the learning algorithm. Consequently, several researchers have been intended to develop the learning algorithm to provide the new weight update algorithm. However, disadvantages of their proposed controller are still unsatisfied such as initial weight, environment factor adaptation, system malfunction adaptation, and the controller performances. Therefore, in this proposed NNPID controller, the new model of the learning algorithms has been provided to acquire the best performance, including an extend Kalman filter algorithm (called the ENNPID controller), a Hybrid of cubature Kalman filter algorithm (called the CKF-NNPID controller), an applied Bayesian filter algorithm (called the ABF-NNPID controller), and hybrid of an Actor-Critic reinforcement algorithm and square-root cubature Kalman filter algorithm (called the NNPID-AC controller). To evaluate the performance of the proposed controllers, the robot arm MATLAB simulations have been implemented and also have been provided the closed-loop control system with the noise environments factors such as load and noise disturbances to prove the robustness and fault tolerance. From the results, the robot arm control system simulation under the control of the proposed controllers can potentially track the error and give the best responses compared with the other previous controller either with or without the load and the noise disturbance.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยความกรุณาจากอาจารย์ที่ปรึกษา ผศ.ดร. ยุทธนา คิดใจเดียว ที่ให้ความช่วยเหลือ ให้คำชี้แนะช่วยแก้ปัญหาตลอดจนให้ความรู้และประการณ์ที่ดีแก่ข้าพเจ้า

ขอขอบพระคุณ รศ.ดร. สุรพันธ์ ยิ้มมั่น รศ.ดร. สุรพันธุ์ เอื้อไพบูลย์ ผศ.ดร. กิตติพล ชิตสกุล ดร. สุรเดช ตรีไตรลักษณะ คณะกรรมการของการสอบวิทยานิพนธ์ที่ได้กรุณาให้คำแนะนำตลอดจนข้อชี้แนะ จนในที่สุดทำให้วิทยานิพนธ์ฉบับนี้สำเร็จลงได้

ขอขอบคุณ สถาบันเทคโนโลยี่ไทย-ญี่ปุ่น (Thai-Nichi Institute of Technology) ที่ให้การสนับสนุนทุนการเรียนปริญญาเอกในครั้งนี้ และขอขอบคุณน้อง ๆ ในห้องปฏิบัติการทุกคน

สำหรับคุณงามความดีอันใดที่เกิดจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบให้กับบิดามารดา ซึ่งเป็นที่รักและเคารพยิ่ง ตลอดจนครูอาจารย์ที่เคารพทุกท่านที่ได้ประสิทธิ์ประสาทวิชาความรู้และถ่านทอดประสบการณ์ที่ดีให้แก่ข้าพเจ้า

อัดนา เซนโต๊ะ

สารบัญ

สารบัญตาราง

สารบัญรูป

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันงานด้านหุ่นยนต์ แขนกล และงานอื่น ๆ ที่เกี่ยวข้องกับใช้งานระบบควบคุมมีเพิ่มมากขึ้นอย่างเห็นได้ชัด อีกทั้งยังมีความต้องการระบบที่ชาญฉลาด สามารถปรับตัวเองได้เมื่อสิ่งแวดล้อมเปลี่ยนแปลงและมีประสิทธิภาพเพิ่มมากขึ้นเช่นเดียวกัน ส่งผลให้เกิดการพัฒนาตัวควบคุมที่สามารถรองรับในเป้าหมายดังกล่าวมากขึ้น โดยทั่วไปประสิทธิภาพของระบบควบคุมในเรื่องความถูกต้องความแม่นยำ และการตอบสนองของระบบควบคุมแบบพลวัตขึ้นอยู่กับปัจจัยต่าง ๆ ได้แก่ ค่าคงที่เริ่มต้น (Gain) ความไม่แน่นอนของตัวแปรของแบบจำลอง สิ่งรบกวน การทำงานของระบบควบคุมแบบทันเวลา (Real-time) การใส่โหลด และความไม่เป็นเชิงเส้น เป็นต้น ซึ่งในที่นี่ตัวควบคุม (Controller) มีบทบาทสำคัญอย่างยิ่งต่อประสิทธิภาพของระบบควบคุม โดยในอดีตที่ผ่านมาตัวควบคุมชนิด Proportional-Integral-Derivative (หรือเรียกว่าตัวควบคุม PID) มีโครงสร้างของตัวควบคุมที่ง่าย ไม่ซับซ้อน และสามารถใช้กับระบบควบคุมได้อย่างมีประสิทธิภาพ จึงทำให้ตัวควบคุม PID มีการใช้กันอย่างแพร่หลาย แต่อย่างไรก็ตามเมื่อประสิทธิภาพของตัวควบคุม PID ขึ้นอยู่กับค่าพารามิเตอร์ (Gain) ทำให้ไม่สามารถรับมือกับระบบควบคุมที่มีปัจจัยแวดล้อมดังที่ได้กล่าวไว้ในข้างต้นได้หมด อีกทั้งปัจจัยแวดล้อมเฉพาะหน้าบางอย่างก็ยากที่จะรับมือได้ หรือบางทีอาจจะต้องปรับค่าพารามิเตอร์ (Gain) ของตัวควบคุมใหม่จากการเปลี่ยนแปลงดังกล่าว ตัวอย่างเช่น การเกิดข้อผิดพลาดของระบบควบคุม การเปลี่ยนแปลงของสถานที่ การเปลี่ยนแปลงของสิ่งแวดล้อมแบบฉับพลัน เป็นต้น ดังนั้นการศึกษาเพื่อการออกแบบโครงสร้างและอัลกอริทึมสำหรับตัวควบคุมจึงมีความสำคัญอย่างมากเพื่อให้ได้ประสิทธิภาพที่ดีที่สุด และสามารถรองรับต่อปัจจัยแวดล้อมที่เปลี่ยนแปลงดังที่กล่าวมาข้างต้น

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

วิทยานิพนธ์ฉบับนี้มุ่งเน้นการศึกษาเรื่องตัวควบคุม (Controller) ที่มีความสามารถเรียนรู้ปัจจัยแวดล้อมที่เปลี่ยนแปลงทั้งภายนอก และภายใน โดยตัวควบคุมนี้สร้างขึ้นจากโครงข่ายประสาทเทียม (Neural Network) ที่มีโครงสร้างตามรูปแบบของตัวควบคุม PID เรียกว่า “ตัวควบคุม (NNPID)” ทำให้สามารถกำหนดค่าของอินพุตควบคุม (Control Input) สำหรับป้อนให้กับระบบด้วยการใช้

อัลกอริทึมของการปรับค่าน้ำหนัก เรียกว่า “อัลกอริทึมการเรียนรู้ (Learning Algorithm)” เนื่องจากตัวควบคุม NNPID นี้มีความสามารถในการปรับค่าพารามิเตอร์ (Gain) ตามค่าน้ำหนักของโครงข่ายประสาทเทียม และเป็นที่ทราบกันดีว่าการปรับค่าน้ำหนักของโครงข่ายประสาทเทียมขึ้นอยู่กับเทคนิคการเรียนรู้ ด้วยเหตุนี้ผู้วิจัยเล็งเห็นว่าการพัฒนาตัวควบคุมให้มีประสิทธิภาพ และสามารถเรียนรู้ระบบด้วยตัวเองนั้นต้องอาศัยปัจจัยที่ประกอบด้วยโครงสร้างของตัวควบคุมที่เหมาะสม และความสามารถต่อการเรียนรู้ของเทคนิคการปรับค่าพารามิเตอร์ให้เหมาะสม ดังนั้นในงานวิจัยฉบับนี้จึงมุ่งเน้นเรื่องการออกแบบตัวควบคุมแบบ NNPID ที่มีโครงสร้างแบบโครงข่ายประสาทเทียมและอัลกอริทึมการเรียนรู้ในรู้แบบต่าง ๆ ทั้งหมด 4 รูปแบบได้แก่ ตัวควบคุม ENNPID, ตัวควบคุม Hybrid CKF-NNPID, ตัวควบคุม ABF-NNPID และสุดท้ายตัวควบคุม NNPID-AC อันส่งผลให้ได้ค่าพารามิเตอร์ที่เหมาะสมสำหรับนำไปใช้ในการควบคุมเพื่อให้เกิดประสิทธิภาพสูงสุด

1.3 สมมุติฐานของการศึกษา

เนื่องจากการปรับค่าน้ำหนักของตัวควบคุมแบบโครงข่ายประสาทเทียม (ตัวควบคุม NNPID) มีความสามารถตามการออกแบบของอัลกอริทึมการเรียนรู้ และการปรับค่าน้ำหนักของตัวควบคุมชนิดนี้ก็ยังส่งผลต่อประสิทธิภาพของระบบควบคุม เช่น [1 - 6] แต่อย่างไรก็ตาม ตัวควบคุมชนิดนี้ที่ใช้งานอยู่ในปัจจุบันก็ยังพบข้อเสีย เช่น การตั้งค่าเริ่มต้น ความเร็วในการประมวลผล และประสิทธิภาพของตัวควบคุม ดังนั้นหากใช้การออกแบบโครงสร้างที่ดีและนำอัลกอริทึมการเรียนรู้ที่มีความสามารถเรียนรู้รูปแบบของปัจจัยแวดล้อมที่เกี่ยวข้องดังที่กล่าวไว้ในข้างต้น ก็น่าจะสามารถพัฒนาตัวควบคุมที่มีประสิทธิภาพ และมีเสถียรภาพเหมาะสมในการใช้งานด้านระบบควบคุมได้

1.4 ทฤษฎีหรือแนวความคิดที่ใช้ในการวิจัย

พารามิเตอร์ของระบบ สิ่งรบกวนจากภายนอก ความผิดพลาดที่อาจจะเกิดขึ้นภายในระบบ ความไม่เป็นเชิงเส้น และการใส่โหลด ล้วนแล้วแต่เป็นปัจจัยหลักที่ส่งผลต่อประสิทธิภาพของระบบควบคุม โดยอดีตที่ผ่านมา ตัวควบคุม PID สามารถรับมือกับปัจจัยเหล่านี้ และมีโครงสร้างของตัวควบคุมที่ง่าย ไม่ซับซ้อน และสามารถใช้กับระบบควบคุมได้ดี จึงทำให้ตัวควบคุม PID ถูกใช้กันอย่างแพร่หลายเพื่อการปรับปรุงระบบควบคุมให้มีประสิทธิภาพ แต่อย่างไรก็ตาม ข้อเสียของตัวควบคุม PID คือจะต้องหาค่าคงที่ที่เหมาะสมเพื่อให้ระบบควบคุมมีประสิทธิภาพ จึงเกิดงานวิจัยมากมายในเรื่องของการนำอัลกอริทึมอื่น ๆ เข้ามาช่วยทำให้ตัวควบคุม PID มีความสามารถเพิ่มขึ้น ตัวอย่างเช่น 1) เทคนิคของ Ziegler-Nichols [7] เป็นเทคนิคที่ต้องใช้การทดสอบจริงเพื่อนำค่าคงที่ของตัวควบคุม PID ออกมา [8] 2) เทคนิคการใช้ผลตอบสนองความถี่ [9] 3) เทคนิคของการเพิ่มประสิทธิภาพ

(Optimization) ใช้เพื่อหาค่าพารามิเตอร์ (Gain) ที่เหมาะสม [10] ซึ่งส่วนใหญ่มักจะถูกนำมาใช้ในระบบควบคุมที่มีลักษณะการกำหนดหลายเป้าหมาย [11] 4) เทคนิคการปรับค่าด้วยตัวเอง (Adaptive tuning) [12]

นอกจากงานวิจัยในข้างต้นแล้ว ยังมีการพัฒนาอย่างต่อเนื่องเกี่ยวกับงานวิจัยเรื่องการเพิ่มขีดความสามารถของตัวควบคุมแบบ PID ด้วยวิธีการใหม่ ๆ อีกมากมาย เช่น การนำระเบียบวิธีคำนวณของโครงข่ายประสาทเทียบ (Neural Network) มาใช้เพื่อเลียนแบบโครงสร้างของตัวควบคุมแบบ PID ทำให้ค่า Gain ของตัวควบคุมสามารถปรับได้ตามค่าน้ำหนักของตัวโครงข่ายประสาทเทียม เรียกว่าตัวควบคุมชนิดนี้ว่า “NNPID” ด้วยข้อดีนี้ทำให้ตัวควบคุมชนิดนี้ได้รับการพัฒนาอย่างต่อเนื่อง เช่น [1 - 6]

1.5 ขอบเขตการวิจัย

งานวิจัยนี้เป็นการศึกษาการออกแบบตัวควบคุมด้วยโปรแกรม MATLAB/SUMILINK ซึ่งประกอบด้วย 3 ส่วนได้แก่ 1) ส่วนของตัวควบคุม (สำหรับผลิตตัวป้อน, Control input) 2) ส่วนของอัลกอริทึมการเรียนรู้ (สำหรับวิเคราะห์และกำหนดค่าตัว Control Input ที่เหมาะสม และ 3) ส่วนของแขนกล 4 ข้อต่อ โดยการออกแบบส่วนตัวควบคุมและอัลกอริทึมการเรียนรู้จะใช้ Block Function ของโปรแกรม MATLAB/SUMILINK สำหรับเขียนโค้ดโปรแกรม และการสร้างส่วนของแขนกลคือการนำแบบจำลองแขนกลที่อยู่ที่มีในไลบารี่ของโปรแกรม MATLAB โดยได้ทำการปรับปรุงค่าพารามิเตอร์บางส่วนของแขนกลตามค่าจริงที่ใช้

1.6 ขั้นตอนของการศึกษา

แนวทางของงานวิจัยนี้จะเป็นการคิดค้นและพัฒนาตัวควบคุมเพื่อใช้ในงานระบบควบคุม โดยส่วนแรกจะเป็นการสร้างตัวควบคุม (Controller) จากโครงข่ายประสาทเทียมด้วยการเลียนแบบโครงสร้างของตัวควบคุม PID เรียกตัวควบคุมนี้ว่า “NNPID” ส่วนต่อมาเป็นงานวิจัยก็จะนำเสนอแนวทางการสร้างส่วนของอัลกอริทึมการเรียนรู้ (Learning Algorithm) เพื่อปรับปรุงประสิทธิภาพของตัวควบคุม ตามเงื่อนไขปัจจัยแวดล้อมที่กำหนด และเพื่อพิสูจน์ความมีประสิทธิภาพและความมีเสถียรภาพของตัวควบคุมที่งานวิจัยได้นำเสนอ จึงได้สร้างแบบจำลองของระบบควบคุมแขนกลเพื่อการทดสอบตัวควบคุมขึ้นมา ซึ่งเป็นแขนกลที่ประกอบด้วย 4 ข้อต่อที่ควบคุมด้วยระบบมอเตอร์ไฟฟ้า โดยได้นำตัวควบคุมมาติดตั้งที่ตัวมอเตอร์สำหรับควบคุมองศาของการหมุนเพื่อให้เป็นไปตามเป้าหมายที่กำหนดและรับมือกับปัจจัยที่มีผลกระทบต่อระบบควบคุม โดยในวิทยานิพนธ์ฉบับนี้ได้พัฒนาตัวควบคุมที่มีความสามารถรองรับปัจจัยข้างต้นทั้งหมดดังนี้ 1) ตัวควบคุมแบบ NNPID ที่ใช้

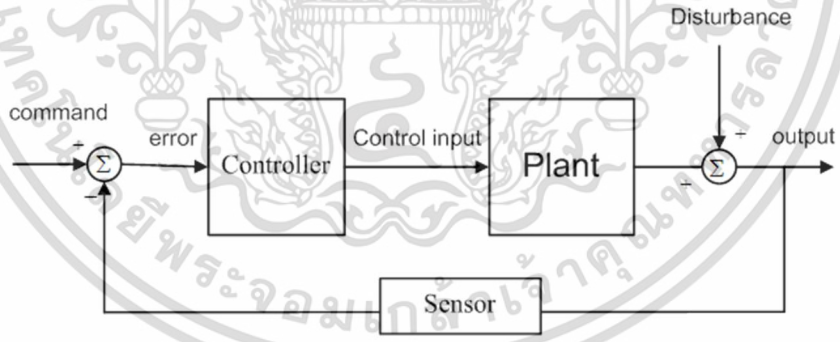
อัลกอริทึมการเรียนรู้แบบส่วนขยายของตัวกรองคาลมานเรียกว่า Neural Network PID-Like using an Extended Kalman filter learning algorithm (ตัวควบคุม ENNPID) [44] 2) ตัวควบคุมแบบ NNPID ที่ใช้อัลกอริทึมการเรียนรู้แบบไฮบริดของตัวกรองคาลมานเรียกว่า Neural Network PID-Like using a Hybrid Cubature Kalman filter learning algorithm (ตัวควบคุม CKF-NNPID) [45] 3) ตัวควบคุมแบบ NNPID ที่ใช้อัลกอริทึมการเรียนรู้ด้วยตัวกรองเบย์แบบประยุกต์ เรียกตัวควบคุมที่ใช้การเรียนรู้นี้ว่า Neural Network PID-Like using applied Bayesian filter algorithm (ตัวควบคุม ABF-NNPID) และ 4) ตัวควบคุม NNPID ที่ใช้อัลกอริทึมการเรียนรู้ปฏิบัติ-วิจารณ์แบบเสริมกำลัง เรียกตัวควบคุมที่ใช้การเรียนรู้นี้ว่า Neural Network PID-Like using Actor-critic Reinforcement Algorithm (ตัวควบคุม NNPID-AC) [46] พร้อมกันนี้ได้นำตัวควบคุม PID แบบมาตรฐานทั่วไป และตัวควบคุมของงานวิจัย [2] เพื่อเปรียบเทียบประสิทธิภาพอีกด้วย

บทที่ 2
งานวิจัยที่เกี่ยวข้อง

โดยทั่วไป การออกแบบตัวควบคุม (Controller) เพื่อให้ระบบควบคุมมีผลตอบสนองที่ดีต้องคำนึงถึงปัจจัยแวดล้อมต่าง ๆ ได้แก่ พารามิเตอร์ของระบบ สิ่งรบกวน ความผิดพลาดที่เกิดขึ้นภายในระบบ ความไม่เป็นเชิงเส้น การเปลี่ยนแปลงกระทันหัน และการใส่โหลด เป็นต้น ดังนั้นงานวิจัยนี้จึงมุ่งเน้นในเรื่องการออกแบบตัวควบคุมด้วยการนำอัลกอริทึมการเรียนรู้ที่สามารถปรับตัวและรับมือกับปัจจัยแวดล้อมต่าง ๆ ได้ โดยในบทนี้จะกล่าวถึงเนื้อหาในส่วนของทฤษฎีของระบบควบคุมและแนวคิดของตัวควบคุม ต่อมาจะเป็นการกล่าวถึงการพัฒนาของตัวควบคุมในอดีต และสุดท้ายจะเป็นการกล่าวถึงส่วนของทฤษฎีและแนวคิดต่าง ๆ ของอัลกอริทึมการเรียนรู้ที่เกี่ยวข้อง

2.1 ทฤษฎีของระบบควบคุม และตัวควบคุม

โดยทั่วไป ระบบควบคุมแบบปิด (Closed-Loop Control System) ประกอบด้วย ตัวระบบ (Plant) ตัวควบคุม (Controller) ซึ่งทำหน้าที่ผลิตตัวป้อน (Control input) สัญญาณป้อนกลับ (Feedback Signal) และสิ่งรบกวนต่าง ๆ (Disturbances) ดังรูปที่ 2.1



รูปที่ 2.1 ระบบควบคุมแบบปิด (Closed-Loop Control System)

โดยในอดีตที่ผ่านมา มีงานวิจัยมากมายที่นำเสนอเกี่ยวกับการปรับปรุงระบบควบคุมให้มีประสิทธิภาพด้วยตัวควบคุม และหนึ่งในตัวควบคุมที่ใช้กันอย่างแพร่หลายในอดีตจนถึงปัจจุบันคือ ตัวควบคุม Proportional-Integral-Derivative (PID) ซึ่งจะเน้นระบบควบคุมที่มีตัวแปรอินพุตและเอาต์พุตตัวเดียว (Single- Input Single-Output System, **SISO**) สามารถเขียนสมการให้อยู่ในรูปไม่ต่อเนื่อง (Discrete) เพื่อให้เหมาะสมต่อการใช้งานดังนี้

$$u(t_k) = K_P e(t_k) + K_I \sum_{i=1}^{k} e(t_i)\Delta t + K_D \frac{e(t_k) - e(t_{k-1})}{\Delta t} \quad (2.1)$$

เมื่อกำหนดให้ $u(t_k)$ และ $e(t_k)$ คือตัวอินพุตป้อนและค่าผิดพลาดที่เกิดขึ้นในระบบ ณ รอบการคำนวณที่ t_k ส่วน Δt คือความห่างของการสุ่มค่าตัวอย่าง

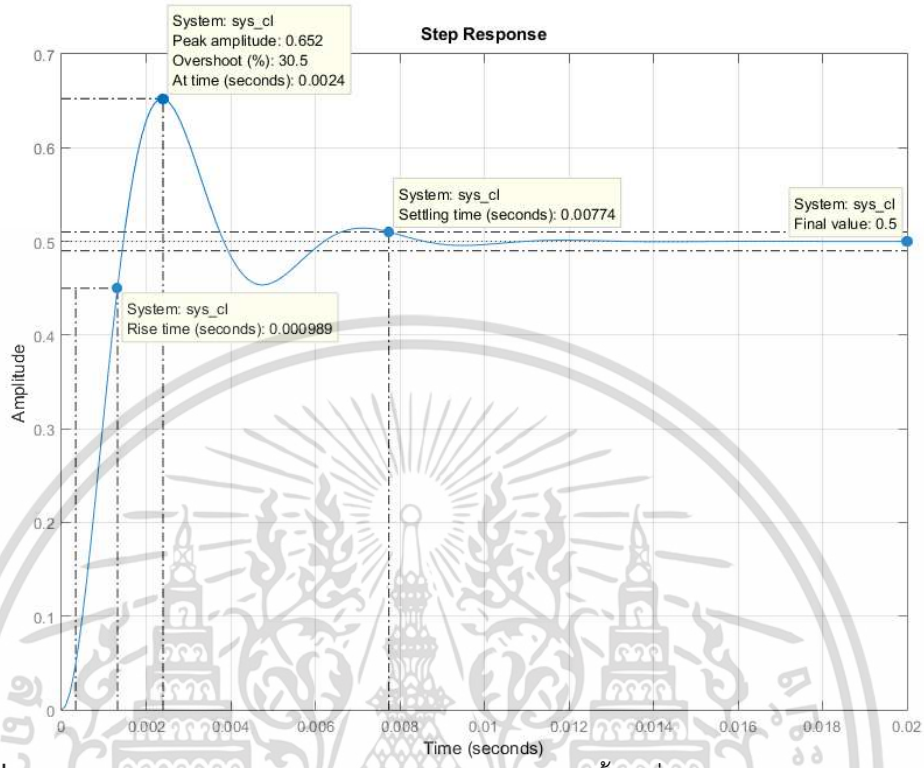
เนื่องจากตัวควบคุม PID นี้มีโครงสร้างของตัวควบคุมที่ง่าย ไม่ซับซ้อน และสามารถใช้กับระบบควบคุมที่อยู่ภายใต้สภาพแวดต่าง ๆ ได้ดี จากรูปที่ 2.1 จะเห็นว่าตัวควบคุมทำหน้าที่ผลิตตัวอินพุตป้อน (Control Input) เพื่อให้ระบบทำงานได้มีประสิทธิภาพจะต้องอาศัยการออกแบบตัวควบคุมซึ่งการออกแบบสามารถใช้หลักการทางมิติเวลา (Time Domain) หรือมิติความถี่ (Frequency Domain) ก็ได้ ซึ่งในกรณีของการออกแบบระบบตามหลักการเชิงความถี่จะมีหัวใจสำคัญของการออกแบบตัวควบคุม PID คือการพิจารณาผลตอบสนองเชิงความถี่ของระบบ (Frequency Response) และเทคนิคเส้นทางเดินราก (Root-Locus) ดังนั้นสามารถเขียนสมการ (2.1) ได้ดังนี้

$$u(s) = e(s)\left(K_P + {K_I}/{s} + K_D s \right) \quad (2.2)$$

เมื่อกำหนดให้ $u(s)$ และ $e(s)$ คือตัวอินพุตป้อน และค่าผิดพลาดที่เกิดขึ้นในระบบในมิติความถี่ (Frequency Domain) ตามลำดับ ส่วน K_P, K_I และ K_D คือค่าคงที่ของตัวควบคุม PID ของส่วนตัวคงที่คูณ ตัวคงที่ปริพันธ์ และตัวคงที่อนุพันธ์ตามลำดับ และ s คือค่าเชิงความถี่ลาปลาซ ซึ่งการออกแบบตัวควบคุม PID ด้วยแนวคิดเชิงความถี่นี้จะต้องสร้างแบบจำลองทางคณิตศาสตร์ของตัวระบบ (Plant) ให้อยู่ในมิติความถี่เสียก่อนซึ่งต้องอาศัยแนวคิดของการแปลงลาปลาซ (Laplace Transform) เพื่อให้สามารถวิเคราะห์ผลตอบสนองของระบบเชิงความถี่ได้ เราเรียกฟังก์ชันของการแปลงลาปลาซนี้ว่า “ฟังก์ชันถ่ายโอน (Transfer Function)” เมื่อคำนวณหาฟังก์ชันถ่ายโอนแล้วก็สามารถนำไปใช้ในการคำนวณหาค่าคงที่ทั้งสาม (K_P, K_I, และ K_D) ของตัวควบคุม PID ด้วยทฤษฎีทางระบบควบคุมเพื่อให้ได้ค่าที่เหมาะสมได้

โดยทั่วไป การพิจารณเรื่องของประสิทธิภาพของระบบควบคุมมักจะใช้การทดสอบการป้อนฟังก์ชันอินพุตแบบขั้นหนึ่งหน่วย (Unit Step Function) แล้วจากนั้นมาศึกษาผลลัพธ์ที่ได้ เรียกว่า “ผลตอบสนองของระบบแบบขั้นหนึ่งหน่วย (Step Response)” และใช้พารามิเตอร์อันได้แก่ ช่วงเวลาขึ้น (Rise Time, T_r) เวลาที่มีค่าแอมพลิจูดสูงสุด (Peak Time, T_P) เวลาการเข้าสู่สถานะเสถียร (Settling Time, T_S) สัดส่วนของค่าสูงสุดที่เกินไปจากค่าคงตัว (Percent Overshoot, %OS) และค่าผิดพลาด ณ สถานะเสถียร (Steady-state error, e_{ss}) ซึ่งแสดงตัวอย่างได้ดังรูปที่ 2.2 ตัวแปรเหล่านี้เป็นตัวชี้วัดความมีประสิทธิภาพของระบบควบคุม และเพื่อให้ได้ระบบควบคุมที่มีประสิทธิภาพจำเป็นต้องปรับเปลี่ยนค่าคงที่ PID ได้แก่ ตัวคงที่คูณ ตัวคงที่ปริพันธ์ และตัวคงที่อนุพันธ์ให้เหมาะสม

ดังที่กล่าวไว้ในข้างต้น ซึ่งการเปลี่ยนค่าคงที่ทั้งสามนั้นจะส่งผลต่อพารามิเตอร์ของผลตอบสนองดังที่กล่าวมาดังรายละเอียดในตารางที่ 2.1



รูปที่ 2.2 ตัวอย่างของผลตอบสนองของระบบแบบขั้นหนึ่งหน่วย (Step Response)

ตารางที่ 2.1 เปรียบเทียบการเพิ่มค่าคงที่ K_P, K_I และ K_D ส่งผลกระทบต่อผลตอบสนอง

ค่าคงที่	พารามิเตอร์				
	T_r	T_P	T_S	%OS	e_{SS}
K_P	ลด	เพิ่ม	เปลี่ยนแปลงเล็กน้อย	ลด	ลด
K_I	ลด	เพิ่ม	เพิ่ม	ลดลงอย่างมีนัยสำคัญ	ลด
K_D	ลดลงเล็กน้อย	ลดลงเล็กน้อย	ลดลงเล็กน้อย	ไม่มีผล	ดีขึ้นถ้า K_D มีค่าน้อย

2.2 ทฤษฎีและแนวคิดต่าง ๆ ของตัวควบคุม และอัลกอริทึมการเรียนรู้ที่เกี่ยวข้อง

หลักการของตัวควบคุม PID คือการปรับค่าคงที่ (Gain) ให้เหมาะสมเพื่อให้ได้มาซึ่งตัวอินพุตป้อน (Control Input) สำหรับการควบคุมระบบให้มีประสิทธิภาพ และด้วยสาเหตุที่เป็นค่าคงที่นี่เอง

ทำให้ตัวควบคุมชนิดนี้มีข้อเสียอยู่หลายประการคือ ประการแรกไม่เหมาะกับระบบควบคุมที่อยู่ภายใต้ปัจจัยแวดล้อมที่ผิดปกติ ยิ่งหากมีสิ่งรบกวน หรือการเปลี่ยนแปลงแบบกระทันหันตัวควบคุม PID นี้แบกรับภาระนี้ได้ไม่ดีมากนัก ประการที่สองคือความไม่เป็นเชิงเส้นของระบบควบคุม ซึ่งในระบบควบคุมทั่วไปมักจะพบโมเดลของระบบที่เป็นไม่เชิงเส้น ทำให้ตัวควบคุม PID ไม่สามารถตอบโจทย์ในงานระบบควบคุมได้ ประการสุดท้ายคือ การหาค่าคงที่ (Gain) ที่เหมาะสมสำหรับสร้างตัวป้อน (Control Input) ให้กับระบบควบคุมเพื่อให้ผลตอบสนองเป็นไปตามเป้าหมายที่กำหนดไว้ ถึงกระนั้นในงานอุตสาหกรรมส่วนใหญ่ก็ยังพบการใช้ตัวควบคุมชนิดนี้อย่างมากมาย เช่น ระบบมอเตอร์ไฟฟ้า ระบบควบคุมแขนกล ระบบควบคุมเครื่องจักร เป็นต้น ด้วยเหตุนี้ทำให้เกิดงานวิจัยมากมายเพื่อปรับปรุงตัวควบคุม PID ให้มีความสามารถเพิ่มขึ้นเพื่อปรับปรุงระบบควบคุมดังกล่าวให้มีประสิทธิภาพมากขึ้นด้วย จากการค้นคว้างานวิจัยในอดีตที่ผ่านมา พบว่าพัฒนาการของตัวควบคุม PID สามารถแบ่งออกเป็น 2 แนวคิดหลัก ๆ คือ 1) การหาค่าคงที่ (Gain) ของตัวควบคุม PID ด้วยการใช้อัลกอริทึมอื่น ๆ เช่น กระบวนการคิดแบบอภิสำนึก (Heuristic algorithm) แนวคิดแบบเชิงความถี่ (Frequency Response) หรือจะเป็นกระบวนการเพิ่มประสิทธิภาพ (Optimization) เป็นต้น และ 2) การสร้างตัวควบคุมที่เลียนแบบ PID ด้วยโครงข่ายประสาทเทียม และทำการปรับปรุงอัลกอริทึมการเรียนรู้ (Learning Algorithm) เพื่อให้ได้ประสิทธิภาพสูงสุด ซึ่งในแต่ละหัวข้อที่กล่าวมาข้างต้น มีรายเอียดดังนี้

2.2.1 การหาค่าคงที่ (Gain) ของตัวควบคุม PID ด้วยการใช้อัลกอริทึม

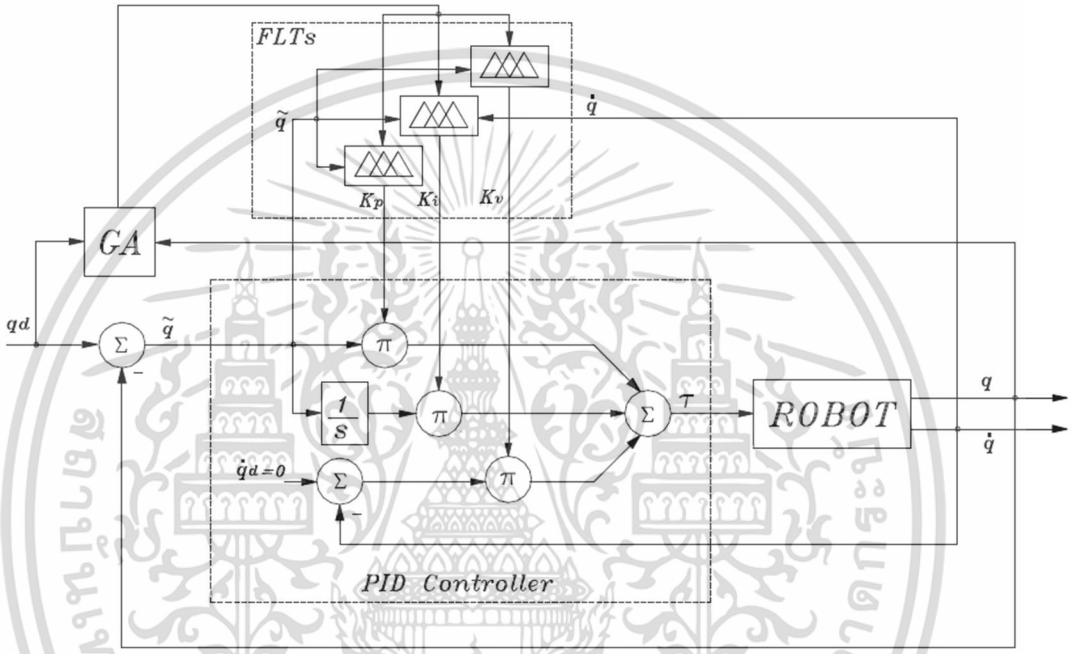
ในส่วนแรกจะกล่าวถึงการใช้อัลกอริทึมมาช่วยหาค่าคงที่ของตัวควบคุม PID ตัวอย่างเช่น เทคนิคของ Ziegler-Nichols [7] เป็นเทคนิคมีมาตั้งแต่ปีค.ศ. 1940 ซึ่งต้องใช้การทดสอบจริงเพื่อนำค่าคงที่ของตัวควบคุม PID ออกมา ซึ่งใช้การทดสอบจริง โดยการหาค่าคงที่ (Gain) จะเริ่มด้วยการกำหนดให้ค่า K_I, และ K_D เป็นศูนย์ จากให้ทดสอบด้วยการเพิ่มค่า K_P จนกว่าระบบควบคุมเกิดภาวะที่เรียกว่า “Undamped” หรือเรียกว่า “Oscillate” และเรียกค่า K_P ณ ตำแหน่งนั้นว่า K_u ให้วัดค่าคาบ (T_u) จากนั้นก็คำนวณค่าที่ K_P, K_I และ K_D จากตารางที่ 2.2 ด้านล่างนี้

ตารางที่ 2.2 การคำนวณ ค่าคงที่ K_P, K_I และ K_D ของ Ziegler–Nichols [7]

ค่าคงที่	ชนิดของตัวควบคุม			
	P	PI	PD	PID
K_P	$0.50K_u$	$0.45K_u$	-	$0.60K_u$
K_I	-	$0.54K_u/T_u$	-	$1.2K_u/T_u$
K_D	-	-	-	$3K_uT_u/40$

โดยในปัจจุบันก็ยังพบงานวิจัยเพื่อปรับปรุงวิธีการของ Ziegler-Nichols ให้มีความสามารถเพิ่มมากขึ้น [9]

อีกหนึ่งอัลกอริทึมที่ใช้หาค่าคงที่ของตัวควบคุม PID คือเทคนิคการใช้อัลกอริทึมเรียนรู้พฤติกรรมของระบบเพื่อการประเมินหรือพยากรณ์ค่า K_P, K_I และ K_D ของตัวควบคุม PID โดยที่อัลกอริทึมจะเรียนรู้ระบบเพื่อสร้างค่า K_P, K_I และ K_D ที่เหมาะสมให้กับตัวควบคุม ตัวอย่างเช่น งานวิจัยที่ใช้แนวคิดตรรกศาสตร์คลุมเครือหรือฟัซซี่ลอจิก (fuzzy logic) มาคำนวณค่า Gain ของตัวควบคุม PID [13] ดังรูปที่ 2.3

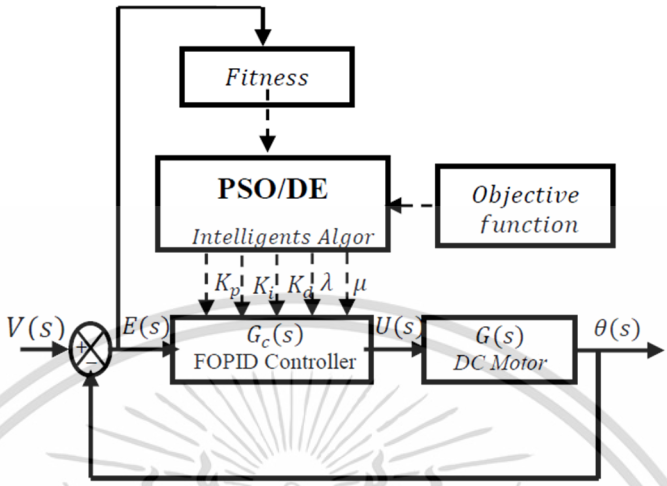


รูปที่ 2.3 การใช้ตรรกศาสตร์คลุมเครือหรือฟัซซี่ลอจิก (fuzzy logic) คำนวณค่า Gain ของตัวควบคุม PID เรียกว่า “การปรับค่าคงที่ PID แบบอัตโนมัติด้วยฟัซซี่”

ที่มา: [13] J. L. Meza, R. Soto and J. Arriaga, "An Optimal Fuzzy Self-Tuning PID Controller for Robot Manipulators via Genetic Algorithm," 2009 Eighth Mexican International Conference on Artificial Intelligence, Guanajuato, 2009, pp. 21-26.

เนื่องจากการตัวควบคุมนี้ได้รับความสนใจเป็นอย่างมาก ทำให้นักวิจัยกลุ่มดังกล่าวพัฒนาต่อยอดจากเดิมเพื่อปรับปรุงประสิทธิภาพของตัวควบคุม อีกทั้งยังมีการพัฒนาอัลกอริทึมสำหรับการหาค่า Gain ของตัวควบคุม PID [14] ดังรูปที่ 2.4 ซึ่งในงานวิจัยดังกล่าวได้ใช้อัลกอริทึมที่เรียกว่า Differential Evolution (DE) and Particle Swarm Optimization (PSO) เพื่อค่าคงที่ (Gain) สำหรับตัวควบคุม PID เพื่อควบคุมความเร็วของมอเตอร์ไฟฟ้า นอกจากนี้ยังพบงานวิจัยที่เกี่ยวข้องกับการพัฒนาการใช้อัลกอริทึมเพื่อการหาค่าคงที่ (Gain) สำหรับตัวควบคุม PID ให้ทำงานอย่างมีประสิทธิภาพ เช่น ตัวควบคุม PID ที่ใช้อัลกอริทึม fractional-order reference model

approximation [15] หรือตัวควบคุมที่ใช้อัลกอริทึมฟัซซี่ [16 - 18] หรือตัวควบคุมที่ใช้อัลกอริทึมการหาค่าคงที่แบบ Lyapunov [19] หรือตัวควบคุมที่ใช้อัลกอริทึมการค่าคงที่แบบ Differential evolution and PSO [20] เป็นต้น



รูปที่ 2.4 การใช้ตรรกศาสตร์คลุมเครือหรือฟัซซี่ลอจิก (fuzzy logic) คำนวณค่า Gain ของตัวควบคุม PID เรียกว่า “การปรับค่าคงที่ PID แบบอัตโนมัติด้วยฟัซซี่” (ต่อ)

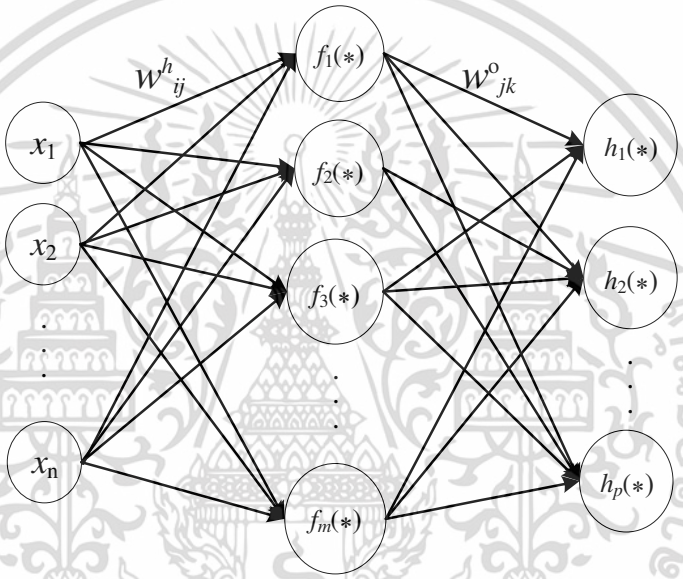
ที่มา: [14] J. L. Meza, V. Santibanez, R. Soto and M. A. Llama, "Fuzzy Self-Tuning PID Semiglobal Regulator for Robot Manipulators," in IEEE Transactions on Industrial Electronics, vol. 59, no. 6, pp. 2709-2717, June 2012.

2.2.2 การพัฒนาตัวควบคุม PID ด้วยโครงข่ายประสาทเทียม

เนื่องด้วยปัจจุบันกระบวนการคำนวณด้วยวิธีโครงข่ายประสาทเทียมกลายเป็นที่ยอมรับ และใช้งานอย่างแพร่หลายอย่างกว้างขวางในงานด้านการแก้ไขปัญหาทางวิศวกรรมมากมาย ตัวอย่างเช่น งานด้านระบบ งานด้านหุ่นยนต์ งานด้านการเงิน เป็นต้น ดังนั้นนอกจากแนวคิดของงานวิจัยเกี่ยวกับการปรับปรุงและการพัฒนาอัลกอริทึมสำหรับหาค่าคงที่ (Gian) ของตัวควบคุมม PID ที่เหมาะสมดังที่กล่าวไว้ในข้างต้นแล้ว ก็ยังพบงานวิจัยที่มุ่งเน้นในเรื่องการเพิ่มขีดความสามารถของตัวควบคุมแบบ PID ด้วยการนำระเบียบวิธีคำนวณแบบโครงข่ายประสาทเทียบ (Neural Network) มาใช้เพื่อการเลียนแบบโครงสร้างของตัวควบคุมแบบ PID ทำให้ค่า Gain ของตัวควบคุมสามารถปรับได้ตามค่าหนักของตัวโครงข่ายประสาทเทียม เรียกว่าตัวควบคุมชนิดนี้ว่า “NNPID” หรือในงานวิจัยบางชิ้นก็ใช้คำว่าตัวควบคุม “PID-Like” ซึ่งในงานวิทยานิพนธ์ฉบับนี้ใช้คำแทนตัวควบคุมที่ใช้โครงข่ายประสาทเทียมเลียนแบบ PID ว่า “ตัวควบคุม NNPID” เพื่อความสะดวกต่อการใช้งาน และจากค้นคว้าศึกษาเรื่องตัวควบคุม NNPID นี้ควรแบ่งการออกแบบตัวควบคุมออกเป็น 2 ส่วนคือ ส่วนการศึกษาของโครงสร้างของตัวควบคุม NNPID เพื่อเลียนแบบการทำงานของตัวควบคุม PID และส่วนการศึกษาเกี่ยวกับอัลกอริทึมการเรียนรู้สำหรับปรับค่าน้ำหนักของโครงข่ายของตัวควบคุม NNPID โดยรายละเอียดของแต่ละหัวข้อมีดังต่อไปนี้

2.2.2.1 โครงสร้างของตัวควบคุม NNPID

ก่อนที่จะกล่าวถึงการพัฒนาของตัวควบคุม NNPID วิทยานิพนธ์ฉบับนี้ขอกล่าวถึงแนวคิดทั่วไปของโครงข่ายประสาทเทียม (Neural Network) เพื่อความเข้าใจเป็นเบื้องต้น โดยแนวคิดของตัวควบคุมแบบโครงข่ายประสาทเทียมนำมาจากแนวคิดของสมองสิ่งมีชีวิตที่มีหน่วยประมวลผลเป็นโนด เรียกว่า “เรียกว่าเซลล์ประสาทเทียม (Neuron)” ซึ่งมีน้ำหนัก (Weight) เชื่อมต่อกันระหว่างโนดและสามารถปรับค่าด้วยวิธีการเรียนรู้แบบวนซ้ำ (Iterative learning algorithm) ส่วนโครงสร้างของโครงข่ายประสาทเทียมขอยกตัวอย่างโครงข่ายประสาทเทียมที่มีโครงสร้างอย่างง่ายที่ประกอบด้วย 3 ส่วนที่เรียกว่า ชั้นอินพุต ชั้นซ่อน และชั้นเอาต์พุต ซึ่งในแต่ละชั้นจะประกอบด้วยโนด และน้ำหนักที่เชื่อมต่อกันดังรูปที่ 2.5



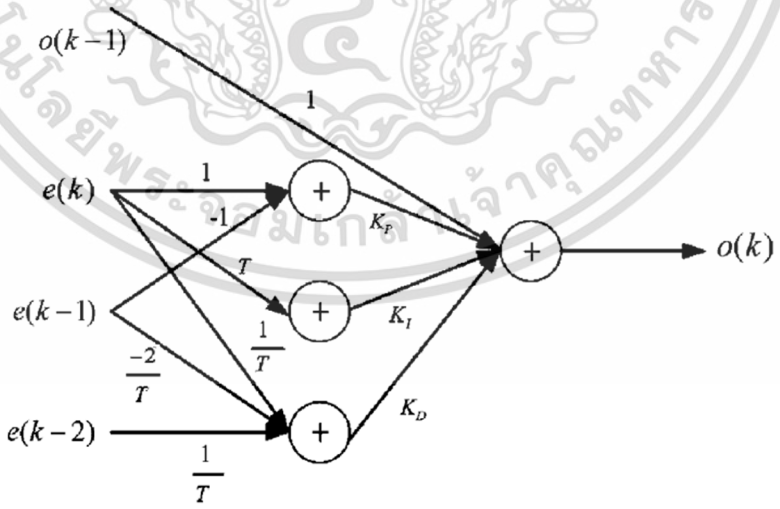
รูปที่ 2.5 โครงสร้างทั่วไปของโครงข่ายประสาทเทียม (Neural Network)

โครงข่ายประสาทเทียมในรูปข้างต้นเรียกว่า “Multi-layer Neural Network” ซึ่งโดยทั่วไปแล้วกระบวนการคำนวณด้วยวิธีโครงข่ายประสาทเทียมจะใช้วิธีการที่เรียกว่า การคำนวณไปข้างหน้า (Feedforward Calculation) เพื่อให้ได้มาซึ่งเอาต์พุตที่ต้องการ และส่วนของการเรียนรู้แบบย้อนกลับ (Backpropagation Learning algorithm) สำหรับคำนวณหาค่าน้ำหนักที่เหมาะสมเพื่อลดค่าผิดพลาดของระบบให้เป็นศูนย์ โดยลำดับแรกจะขอกล่าวถึงส่วนของการคำนวนไปข้างหน้าทำหน้าที่สร้างเอาต์พุตจากอินพุตที่คูณด้วยน้ำหนักที่เชื่อมของแต่ละโนดของแต่ละชั้นเพื่อให้ได้ค่าเอาต์พุตตามเป้าหมายที่กำหนด ซึ่งมีสมการการคำนวณดังนี้

$$u_p = h_p\left(\sum_{m=1}^{m} f\left(\sum_{i=1}^{n} x_i w_{ij}^h + b_j^h\right) w_{jk}^o + b_k^o\right) \tag{2.3}$$

เมื่อกำหนดให้ x, u คืออินพุต และเอาต์พุตของเซลล์ประสาท ตามลำดับ ส่วน w^h คือน้ำหนักที่เชื่อมต่อกันระหว่างชั้นอินพุตกับชั้นซ่อน และ w^o คือน้ำหนักที่เชื่อมต่อกันระหว่างชั้นซ่อนกับขั้นเอาต์พุต และมีค่าไอบัส b^h คือน้ำหนักไบอัสของชั้นซ่อน และ b^o คือน้ำหนักไบอัสของชั้นเอาต์พุต สำหรับแต่ละเซลล์ประสาทของโครงข่ายประสาทเทียมจะมีฟังก์ชันกระตุ้น (Activation Function) แทน $f(*)$ ด้วยฟังก์ชันกระตุ้นสำหรับการแปลงในชั้นซ่อน และ $h(*)$ ด้วยฟังก์ชันกระตุ้นสำหรับการแปลงในชั้นเอาต์พุต ส่วนตัวอย่างของฟังก์ชันกระตุ้นสำหรับการนำมาใช้ได้แก่ ฟังก์ชันเชิงเส้น (*linear* Activation Function) ฟังก์ชันซิกมอยด์ (*sigmoid* Activation Function) ฟังก์ชันไฮเพอร์โบลิกแทนเจนต์ (*tanh* Function) เป็นต้น ซึ่งในแต่ละชนิดของฟังก์ชันกระตุ้นจะให้ผลลัพธ์ที่แตกต่างกันไป ทั้งนี้ขึ้นอยู่กับการประยุกต์ใช้งาน

สำหรับกรณีส่วนของการเรียนรู้แพร่ย้อนกลับ (Backpropagation Learning Algorithm) จะทำหน้าที่เกี่ยวกับการปรับปรุงค่าน้ำหนักของโครงข่ายด้วยวิธีการเรียนรู้แบบวนซ้ำ (Iterative learning algorithm) เพื่อให้ได้เอาต์พุตตามเป้าหมายที่กำหนดไว้ ซึ่งในปัจจุบันมีอัลกอริทึมการเรียนรู้ (Learning Algorithm) มากมาย เช่น อัลกอริทึมการเรียนรู้เดลต้า (Delta Rule Algorithm) อัลกอริทึมการเรียนรู้ตัวกรองเบย์ (Bayesian Filter Learning Algorithm) อัลกอริทึมการเรียนรู้ตัวกรองคาลมาน (Kalman Filter Learning Algorithm) เป็นต้น ทั้งนี้การเลือกอัลกอริทึมสำหรับการเรียนรู้ของโครงข่ายประสาทเทียมนั้นก็ขึ้นอยู่กับงานที่จะนำไปใช้ ทำให้โครงข่ายประสาทเทียมกลายเป็นหนึ่งในศาสตร์ความรู้ในด้านการเรียนรู้ของเครื่องจักร (Machine Learning) ซึ่งมีความสำคัญต่องานหลายด้าน เช่น งานด้านคัดกรอง (Classification) การเรียนรู้ระบบ (System Identification) การพยากรณ์ (Prediction) เป็นต้น



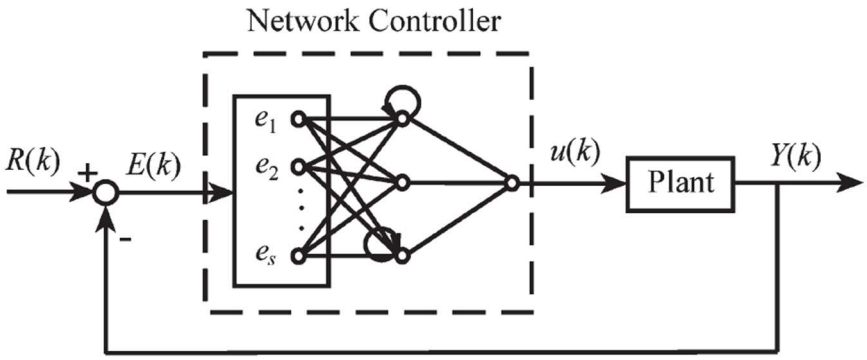
รูปที่ 2.6 โครงสร้างตัวควบคุม NNPID ที่นำเสนอโดย [1]

ที่มา: Ren T.J., Chen T.C., and Chen C.J. “Motion control for a two-wheeled vehicle using a self-tuning PID controller” Control Engineering Practice, vol. 16, no. 3, 2008

ด้วยเหตุนี้ทำให้ตัวควบคุม NNPID ได้รับการพัฒนาอย่างต่อเนื่องด้วยอัลกอริทึมของการเรียนรู้ที่หลากหลายตามการประยุกต์ใช้งาน ซึ่งจากการค้นคว้างานวิจัยพบว่าการพัฒนาตัวควบคุม NNPID เริ่มตั้งแต่ต้นคริสต์ศตวรรษที่ 21 โดยมีงานนักวิจัยมากมายที่ต้องการพัฒนาตัวควบคุม NNPID ซึ่งในช่วงแรกของงานวิจัยก็มุ่งเน้นเกี่ยวกับการสร้างตัวควบคุมโครงข่ายประสาทเทียมด้วยการเลียนแบบตัวควบคุม PID ที่เหมือนกันทุกประการ [1] ดังรูปที่ 2.6 ซึ่งจากภาพดังกล่าวแสดงให้เห็นถึงตัวอย่างของตัวควบคุม NNPID ที่สร้างขึ้นโดยกำหนดให้ชั้นซ่อนของโครงข่ายประสาทเทียมตามสมการของตัวควบคุม PID และกำหนดให้เอาต์พุตของแต่ละโนดของชั้นซ่อนทั้ง 3 โนดแทนด้วยค่า K_P, K_I และ K_D ซึ่งสามารถปรับเปลี่ยนได้ตามอัลกอริทึมของการเรียนรู้ของตัวโครงข่ายประสาทเทียม โดยงานวิจัยนี้ได้พลิกโฉมตัวควบคุม PID ที่ได้ออกแบบสำหรับระบบควบคุมที่มีตัวแปรเดียวด้วยการเพิ่มความสามารถรองรับอินพุตได้หลายอินพุต กล่าวคือสามารถรองรับตัวแปรของระบบควบคุมได้มากกว่า 2 ตัวแปรนั้นเอง แต่อย่างไรก็ตามยังประสบปัญหาเกี่ยวกับการหาค่าน้ำหนักโครงข่ายเริ่มต้น (Initial weight) ของตัวควบคุมที่เหมาะสม

อีกหนึ่งงานวิจัยตัวอย่าง [2] ที่ใช้ตัวควบคุม NNPID ซึ่งได้ออกแบบตัวควบคุมที่ใช้โครงข่ายประสาทเทียมโดยเลียนแบบตัวควบคุม PID เพื่อควบคุมระบบเพนดูลัมผกผัน (Inverted Pendulum) ไดอะแกรมของระบบควบคุมของการทดลองดังกล่าวดังรูปที่ 2.7 ซึ่งงานวิจัยดังกล่าวได้ให้ความสนใจในเรื่องของตัวควบคุมสำหรับระบบควบคุมแบบหลายแปร จากผลการทดลองของงานวิจัยดังกล่าวได้พิสูจน์แล้วว่าตัวควบคุม NNPID สามารถใช้แทนตัวควบคุม PID ได้ เนื่องด้วยโครงสร้างของตัวควบคุม NNPID ที่มีเซลล์ประสาทในชั้นอินพุตหลายเซลล์ ทำให้สามารถรองรับอินพุตได้มากกกว่าตัวแปรเดียว ซึ่งโนดของอินพุตแทนด้วยตัวแปร $\boldsymbol{E}(k)$ ซึ่งประกอบด้วย e_1, e_2, ..., e_s (กำหนดให้ s คือจำนวนอินพุต) ส่วนน้ำหนักของโครงข่ายมีอยู่ 2 ส่วนคือ และส่วนที่เชื่อมต่อระหว่างชั้นซ่อน (Hidden Layer) กับเอาต์พุต (Output Layer) คือส่วนนี้น้ำหนักของโครงข่ายกำหนดให้คงที่ และส่วนที่เชื่อมต่อระหว่างอินพุต (Input Layer) กับชั้นซ่อน (Hidden Layer) ซึ่งน้ำหนักส่วนนี้เป็นส่วนที่สามารถปรับค่าน้ำหนักเพื่อให้ได้ค่าเอาต์พุต (แทนด้วย $\boldsymbol{Y}(k)$) ตามเอาต์พุตเป้าหมายที่กำหนด (แทนด้วย $\boldsymbol{R}(k)$) ตามข้อกำหนดของตัวควบคุม PID

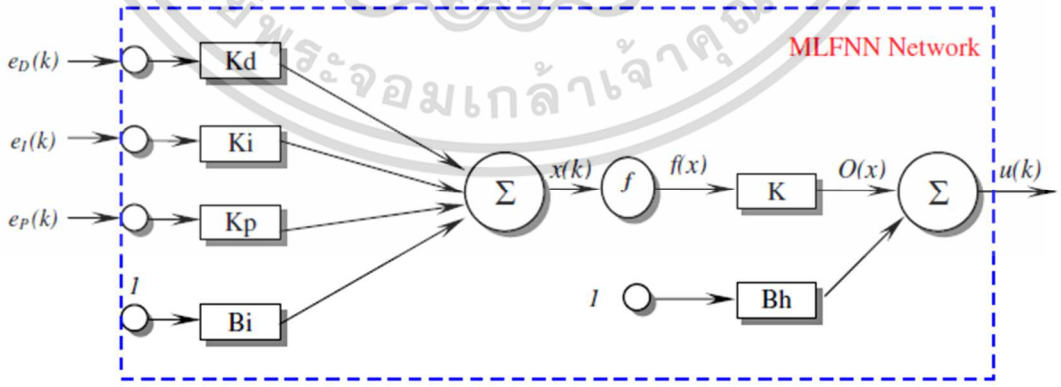
สำหรับอัลกอริทึมของการเรียนรู้เพื่อการปรับค่าน้ำหนักของตัวควบคุม NNPID นี้ได้ใช้อัลกอริทึมการเรียนรู้ของ Gradient Descent ที่เรียกว่า “Resilient Back-propagation Learning Algorithm” จากผลลัพธ์ออกมาเมื่อเทียบกับตัวควบคุม PID แล้วให้ผลเป็นที่น่าพอใจเป็นอย่างมาก แต่อย่างไรก็ตาม ตัวควบคุม NNPID ที่นำเสนอนี้ประสบปัญหาด้านการตั้งค่าเริ่มต้นให้กับน้ำหนักโครงข่ายประสาทเทียมเช่นกัน กล่าวคืออัลกอริทึมนี้จะทำงานได้ดีเมื่อกำหนดค่าเริ่มต้นที่เหมาะสมเท่านั้น ซึ่งจากงานวิจัยดังกล่าวได้ใช้ค่าน้ำหนักเริ่มต้นด้วยวิธีการเดียวกันกับการปรับค่า Gain ของตัวควบคุม PID ที่เขาให้ทำการศึกษาไว้ก่อนหน้านี้



รูปที่ 2.7 โครงสร้างตัวควบคุม NNPID ที่นำเสนอโดย [2]

ที่มา: S. Cong and Y. Liang, "PID-Like Neural Network Nonlinear Adaptive Control for Uncertain Multivariable Motion Control Systems," in IEEE Transactions on Industrial Electronics, vol. 56, no. 10, pp. 3872-3879, Oct. 2009.

อีกหนึ่งในงานวิจัยที่ได้รับความสนใจในเรื่องของการปรับปรุงของตัวควบคุม NNPID ที่ได้นำเสนอโดย H. Huy Anh ในปี 2010 [3] เพื่อนำไปใช้ควบคุมระบบมอเตอร์ไฟฟ้าซึ่งมีไดอะแกรมของตัวควบคุมดังรูปที่ 2.8 โดยมีโครงสร้างของตัวควบคุม NNPID ที่ประกอบด้วย 3 ชั้น คือชั้นอินพุต ชั้นซ่อน และชั้นเอาต์พุต โดยมีชั้นอินพุตรับค่าผิดพลาดจากระบบที่แยกเป็นค่าผิดพลาดของส่วนตัวคูณ (Proportional Error, e_P) ค่าผิดพลาดของตัวปริพันธ์ (Integral Error, e_I) และค่าผิดพลาดของตัวอนุพันธ์ (Derivation Error, e_D) ส่วนชั้นซ่อนประกอบด้วยหนึ่งเซลล์ประสาทที่มีสมการตามข้อกำหนดของตัวควบคุม PID แต่ด้วยการใส่ฟังก์ชัน tanh ของชั้นซ่อนทำให้ต้องเพิ่มการเชื่อมต่อน้ำหนักระหว่างชั้นซ่อนและชั้นเอาต์พุตเพื่อให้ค่าที่เป็นไปได้ในทางปฏิบัติ และสามารถนำไปควบคุมระบบมอเตอร์ไฟฟ้าได้



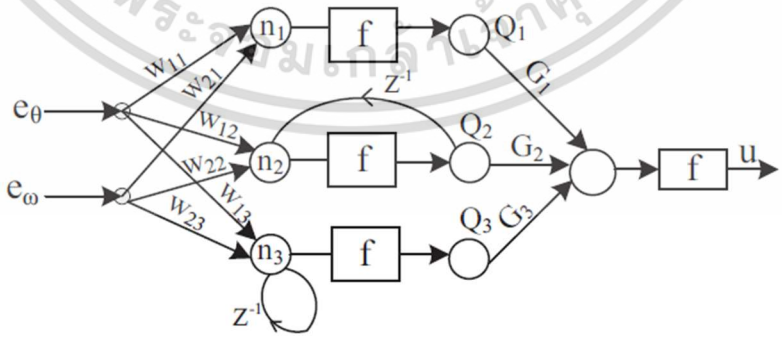
รูปที่ 2.8 โครงสร้างตัวควบคุม NNPID ที่นำเสนอโดย [3]

ที่มา: H. Huy Anh, “Online tuning gain scheduling MIMO neural PID control of the 2-axes pneumatic artificial muscle (PAM) robot arm” Ex-pert Systems with Applications, vol. 37, no. 9, 2010. pp.6547-6560.

ในส่วนของกฎการปรับค่าน้ำหนักของโครงข่ายของตัวควบคุม NNPID นี้ได้ใช้กฎการเรียนรู้ของ Gradient Descent ซึ่งมีการทดสอบหาค่าเริ่มต้นของน้ำหนักก่อนการใช้งานจริง และเช่นเดียวกันกับงานวิจัยที่กล่าวถึงก่อนหน้านี้ ตัวควบคุม NNPID ประสบปัญหาด้านการตั้งค่าเริ่มต้นให้กับน้ำหนักโครงข่ายประสาทเทียมเช่นกัน

ในปี 2014 ก็ได้มีงานวิจัย [5] ที่นำเสนอโดย V. Kumar, P. Gaur และ A. Mittal เกี่ยวกับตัวควบคุม NNPID ที่สามารถรองรับงานระบบควบคุมที่มีตั้งแต่ 2 ตัวแปรขึ้นไปเช่นกัน โครงสร้างของตัวควบคุมนี้มีลักษณะดังรูปที่ 2.9 ตัวควบคุมนี้ได้ยกระดับความสามารถของตัวควบคุมไปอีกขั้นหนึ่ง กล่าวคือตัวควบคุมที่นำเสนอนี้สามารถตอบโจทย์ทั้งในเรื่องของการตั้งค่าเริ่มต้นของน้ำหนักโครงข่ายของตัวควบคุม NNPID ให้สามารถสร้างค่าเริ่มต้นจากตัวพยากรณ์ และการใช้ฟังก์ชันกระตุ้นในชั้นซ่อนทำให้เพิ่มช่วงของการปรับค่าน้ำหนักโครงข่ายเพิ่มขึ้น อีกทั้งยังสามารถปรับค่าได้ทั้งชั้นซ่อนและชั้นเอาต์พุต ส่งผลทำให้โอกาสที่จะปรับหาค่าน้ำหนักเพื่อให้ค่าที่เหมาะสมที่สุดมีมากขึ้นเพราะมีตัวแปรสำหรับการปรับค่ามากขึ้น สำหรับอัลกอริทึมการเรียนรู้เพื่อปรับค่าน้ำหนักของตัวควบคุม NNPID เพื่อให้ค่าเอาต์พุตเป็นไปตามเป้าหมายที่กำหนด ในงานวิจัยนี้ได้ใช้อัลกอริทึมกำลังสองน้อยสุด (Least Square Learning Algorithm)

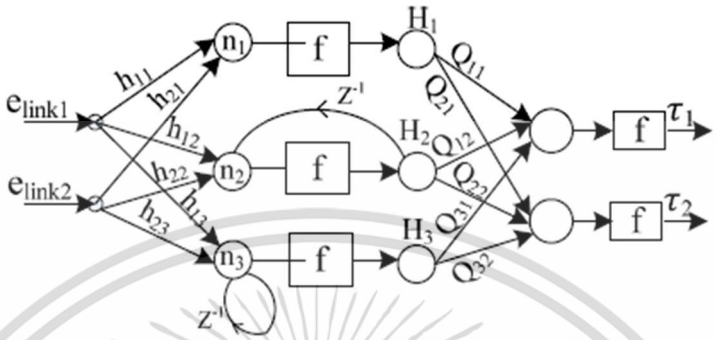
สำหรับการตั้งค่าเริ่มต้นของน้ำหนักโครงข่ายในตัวควบคุม NNPID เขาได้สร้างตัวพยากรณ์ด้วยโครงข่ายประสาทเทียมอีกชุด และเก็บตัวอย่างเพื่อสร้างแบบจำลองขึ้นมาสำหรับพยากรณ์ ท้ายสุดก็นำค่าเริ่มต้นของน้ำหนักไปใช้ในตัวควบคุม NNPID ถึงแม้ว่าตัวควบคุม NNPID ไม่ต้องใช้การทดสอบเพื่อหาค่าเริ่มต้นแล้ว แต่อย่างไรก็ตาม ตัวควบคุม NNPID จะต้องมีการคำนวณค่าน้ำหนักเริ่มต้นทุกครั้งก่อนการใช้งาน ส่งผลทำให้ตัวควบคุมก็ยังประสบปัญหาด้านการตั้งค่าเริ่มต้นให้กับน้ำหนักโครงข่ายประสาทเทียมเช่นกัน พร้อมกันนี้ตัวควบคุมชนิดยังไม่สามารถรองรับการเปลี่ยนแปลงของสภาพแวดล้อมกะทันหัน หรือมีปัจจัยแวดล้อมที่เกินขีดจำกัดของการชุดพยากรณ์



รูปที่ 2.9 โครงสร้างตัวควบคุม NNPID ที่นำเสนอโดย [5]

ที่มา: H. Huy Anh, “Online tuning gain scheduling MIMO neural PID control of the 2-axes pneumatic artificial muscle (PAM) robot arm” Ex-pert Systems with Applications, vol. 37, no. 9, 2010. pp.6547-6560.

เนื่องจากตัวควบคุมชนิดนี้ได้รับความสนใจจากนักวิจัย ทำให้งานวิจัยของกลุ่มนักวิจัยดังกล่าวได้พัฒนาตัวควบคุมให้มีความสามารถเพิ่มมากขึ้นในปี 2016 [21] ซึ่งโครงสร้างของตัวควบคุม NNPID ชนิดนี้มีลักษณะดังรูปที่ 2.10



รูปที่ 2.10 โครงสร้างตัวควบคุม NNPID ที่นำเสนอโดย [21]

ที่มา: Richa S., Vikas K., Prerna G., Mittal A.P., "An adaptive PID like controller using mix locally recurrent neural network for robotic manipulator with variable payload" ***ISA Transactions***, vol. 62, no., 2016. pp. 258-267.

นอกจากนี้ยังพบงานวิจัยเกี่ยวกับการพัฒนาตัวควบคุม NNPID ไปใช้งานระบบควบคุมซึ่งอาจจะแตกต่างกันในเรื่องของอัลกอริทึมการเรียนรู้เพื่อปรับค่าน้ำหนัก แต่ประสบปัญหาคล้ายๆ กันคือเรื่องการตั้งค่าเริ่มต้นให้กับน้ำหนักโครงข่ายประสาทเทียม และยังต้องปรับปรุงประสิทธิภาพของระบบควบคุมต่อปัจจัยแวดล้อมดังที่กล่าวมาข้างต้น [6, 22 - 25]

2.2.2.2 อัลกอริทึมการเรียนรู้สำหรับปรับค่าน้ำหนักของตัวควบคุม NNPID

โดยทั่วไป ประสิทธิภาพของโครงข่ายประสาทเทียมนอกจากจะขึ้นอยู่กับโครงสร้างของโครงข่ายดังที่กล่าวไว้แล้วในหัวข้อที่ผ่านมาแล้ว ยังมีอีกส่วนหนึ่งที่สำคัญต่อการพัฒนาประสิทธิภาพของตัวควบคุม NNPID นี้คือ กฎการปรับค่าน้ำหนักหรือเป็นที่รู้จักกันในชื่อว่า "อัลกอริทึมการเรียนรู้ (Learning Algorithm)" หรือบางทีก็เรียกว่า "อัลกอริทึมการฝึกสอน (Training Algorithm)" ซึ่งในวิทยานิพนธ์ฉบับนี้ขอใช้คำว่า "อัลกอริทึมการเรียนรู้ (Learning Algorithm)" และด้วยเหตุนี้การพัฒนาตัวควบคุมให้มีประสิทธิภาพ และสามารถเรียนรู้ระบบด้วยตัวเองนั้นต้องอาศัยการพัฒนาอัลกอริทึมการเรียนรู้นั้นเอง ดังนั้นอีกหนึ่งหัวข้อที่งานวิจัยนี้มุ่งเน้นคือการพัฒนาอัลกอริทึมการเรียนรู้ให้กับตัวควบคุม NNPID ซึ่งในอดีตที่ผ่านมาการพัฒนาเกี่ยวกับอัลกอริทึมการเรียนรู้ของโครงข่ายประสาทเทียมเพื่อให้ได้ค่าน้ำหนักโครงข่ายที่เหมาะสมสำหรับนำไปใช้ในระบบควบคุมต่าง ๆ เพื่อให้ได้ประสิทธิภาพสูงที่สุดและดีที่สุด อีกทั้งยังสามารถรองรับปัจจัยแวดล้อมที่เปลี่ยนแปลงไปนั้นมีมากมาย ตัวอย่างเช่น อัลกอริทึมการเรียนรู้ของ Perceptron อัลกอริทึมการเรียนรู้ของ Hebb

อัลกอริทึมการเรียนรู้เดลต้า (Delta Rule) หรือเรียกว่าอัลกอริทึมการเรียนรู้แบบกำลังสองน้อยสุด (Least Mean Square Algorithm) อัลกอริทึมการเรียนรู้เชิงแข่งขัน (Competitive Learning Algorithm) อัลกอริทึมการเรียนรู้แพร่ย้อนกลับ (Backpropagation Learning Algorithm) เป็นต้น

นอกจากอัลกอริทึมที่ได้กล่าวไว้ในข้างต้นแล้ว ในปัจจุบันอัลกอริทึมการเรียนรู้ของโครงข่ายประสาทเทียมได้ถูกพัฒนาอย่างมากมายเพื่อการประยุกต์ใช้งานด้านต่าง ๆ เช่น ส่วนปรับปรุงของอัลกอริทึมการเรียนรู้แพร่ย้อนกลับ (Modified Backpropagation Learning Algorithm) อัลกอริทึมการเรียนรู้แบบตัวกรองคาลมานรวมถึงส่วนขยายของตัวกรองคาลมาน (Kalman Filter Learning Algorithm และ Extend Kalman Filter, ***EKF***) อัลกอริทึมการเรียนรู้คาลมานด้วยกฎ Cubature (Cubature Kalman Filter, ***CKF*** และส่วนขยาย (Square-Root Cubature Kalman Filter, ***SCKF***) อัลกอริทึมการเรียนรู้ด้วยตัวกรองเบย์ (Bayesian Filter Learning Algorithm) และอัลกอริทึมการเรียนรู้ปฏิบัติ-วิจารณ์แบบเสริมกำลัง (Actor-critic Reinforcement learning algorithm) เป็นต้น ซึ่งในที่นี่ขอกล่าวถึงเฉพาะส่วนอัลกอริทึมเฉพาะส่วนที่เกี่ยวข้องดังนี้

อัลกอริทึมของการเรียนรู้แบบตัวกรองคาลมาน (Kalman Filter Algorithm) ซึ่งเป็นอัลกอริทึมที่เหมาะสำหรับงานในด้านงานระบบพลวัต (Dynamic System) เช่น ระบบควบคุม ระบบนำทาง ระบบขับเคลื่อนอากาศยาน เป็นต้น และเนื่องจากอัลกอริทึมตัวกรองคาลมานใช้กับระบบควบคุมที่เป็นพลวัต ซึ่งในงานวิจัยนี้จะขอกล่าวถึงระบบสมการของแบบจำลองปริภูมิสถานะ (State Space Model) โดยมีรายละเอียดดังนี้

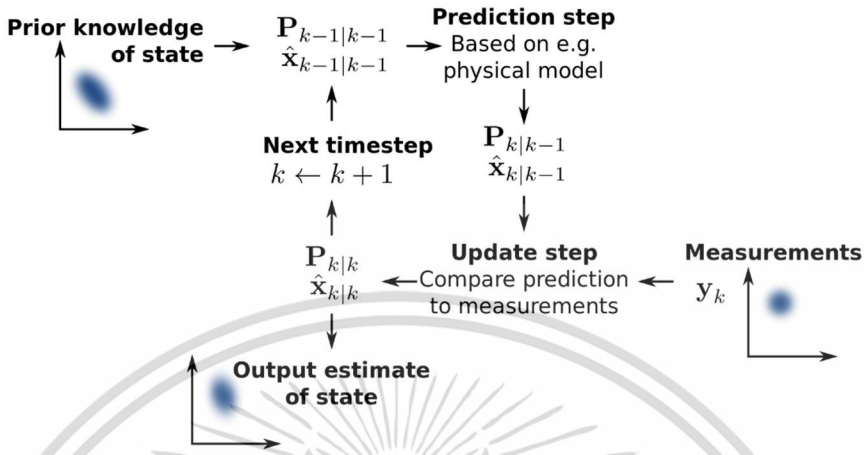
$$x_k = f\left(x_{k-1}\right) + \omega_{k-1} \tag{2.4}$$

$$y_k = h\left(x_k\right) + v_k \tag{2.5}$$

โดยกำหนดให้ x_k และ y_k คือสถานะระบบและเอาต์พุตของระบบ ตามลำดับ ส่วน ω และ v คือค่าสัญญาณรบกวนที่เกิดจากระบบและเกิดจากการวัดค่าเอาต์พุตตามลำดับ สุดท้ายฟังก์ชัน $f(*)$ และ $h(*)$ คือแบบจำลองทางคณิตศาสตร์ของระบบ (System Plant) และเอาต์พุตตามลำดับ

แนวคิดของการเรียนรู้ตัวกรองคาลมานได้รับการออกแบบเพื่อให้ระบบสามารถพยากรณ์ค่าจากสัญญาณป้อนกลับล่วงหน้าอันส่งผลทำให้ทันเวลาต่อการควบคุม โดยจุดประสงค์ของอัลกอริทึมนี้คือการพยากรณ์สถานะ (State, $\boldsymbol{x}$) สำหรับการคำนวณค่าอินพุตควบคุม (Control Input) เพื่อให้ได้เอาต์พุตที่ตรงตามเป้าหมายที่กำหนด (Desired Output, y_d) โดยการปรับค่าตัวแปรต่าง ๆ ด้วยอัลกอริทึมการเรียนรู้แบบวนซ้ำ (Iterative Learning Algorithm) อันประกอบด้วย 2 ขั้นตอนดังนี้ 1) ขั้นตอนการพยากรณ์ (Prediction) เพื่อพยากรณ์สถานะของระบบ (Transition State, $\boldsymbol{x}_{k|k-1}$) และค่าความแปรปรวนร่วมเกี่ยว (Covariance, $\boldsymbol{P}_{k|k-1}$) และ 2) ขั้นตอนการปรับเพิ่มความถูกต้อง

(Correction) ด้วยการนำเอาต์พุตของระบบที่พยากรณ์มาเปรียบเทียบกับค่าเอาต์พุตเป้าหมายที่กำหนดเพื่อคำนวณหาค่าคงที่ของคาลมาล (Kalman gain, $\boldsymbol{K}$)



รูปที่ 2.11 อัลกอริทึมของตัวกรองคาลมาน (Kalman Learning Algorithm)

ที่มา: https://en.wikipedia.org/wiki/PID_controller, Oct. 2018.

จากรูปที่ 2.11 แสดงถึงการคำนวณวนซ้ำของอัลกอริทึมตัวกรองคาลมาน ซึ่งในช่วงแรกของการพัฒนาอัลกอริทึมชนิดนี้เป็นการออกแบบสำหรับนำไปใช้ในระบบควบคุมแบบพลวัตเชิงเส้นเน้นไปทางด้านงานง่าย ๆ ไม่ยุ่งยากมากนัก (ซึ่งในที่นี่วิธีคำนวณของอัลกอริทึมมคาลมานกับระบบที่เป็นเชิงเส้นจะไม่ขอกล่าวถึง) แต่อย่างไรก็ตาม หลังจากที่ได้นำเสนออัลกอริทึมตัวกรองคาลมานไม่นาน กลุ่มนักวิจัยดังกล่าว ก็ได้พัฒนาตัวอัลกอริทึมตัวกรองคาลมานที่สามารถนำมาประยุกต์ใช้งานกับระบบควบคุมพลวัตที่ไม่เป็นเชิงได้ โดยใช้ชื่อว่าส่วนขยายอัลกอริทึมตัวกรองคาลมาน (Extended Kalman Filter, EKF) ซึ่งอาศัยการปรับค่าแบบวนซ้ำ (Iterative Learning Algorithm) และมีรายละเอียดดังนี้

- ขั้นตอนการพยากรณ์ (Prediction)

$$\mathbf{x}_{k|k-1} = f\left(\mathbf{x}_{k-1}\right) \tag{2.6}$$

$$\mathbf{P}_{k|k-1} = \mathbf{J}\left(\mathbf{x}_{k-1}\right)\mathbf{P}_{k-1}\mathbf{J}\left(\mathbf{x}_{k-1}\right) + \mathbf{Q}_{k-1} \tag{2.7}$$

- ขั้นตอนการพิสูจน์ความถูกต้อง (Correction)

$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + \mathbf{K}_k\left(y_d - y_{k|k-1}\right) \tag{2.8}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{J}^T\left(\mathbf{x}_{k|k-1}\right)\left[\mathbf{J}\left(\mathbf{x}_{k|k-1}\right)\mathbf{P}_{k|k-1}\mathbf{J}^T\left(\mathbf{x}_{k|k-1}\right) + \mathbf{R}_k\right]^{-1} \tag{2.9}$$

$$\mathbf{P}_{k|k} = \left(\mathbf{I} - \mathbf{K}_k\mathbf{J}\left(\mathbf{x}_{k|k-1}\right)\right)\mathbf{P}_{k|k-1} \tag{2.10}$$

โดยกำหนดให้ $\mathbf{J}$ และ $\mathbf{I}$ คือเมตริกของ Jacobian ของฟังก์ชัน $f(*)$ และเมตริกเอกลักษณ์ตามลำดับ ส่วนค่า $\mathbf{Q}$ และ $\mathbf{R}$ คือ เมตริกของความแปรปรวนร่วมเกี่ยวของสัญญาณรบกวนที่ได้จากระบบ และเมตริกของความแปรปรวนร่วมเกี่ยวของสัญญาณรบกวนที่ได้จากการวัดค่าเอาต์พุตตามลำดับ
เนื่องจากเมตริก $\mathbf{J}$ เป็นเมตริก Jacobian ซึ่งได้มาจากการใช้กระบวนการแปลงฟังก์ชันที่ไม่เป็นเชิงเส้นให้เป็นเชิงเส้น จึงทำให้ประสิทธิภาพการใช้งานของตัวกรองคาลมานได้ผลได้ไม่ค่อยจะดีมากมัก ส่งผลทำให้มีงานวิจัยเพื่อพัฒนาอัลกอริทึมตัวกรองคาลมานอย่างมากมาย เช่น Iterated extended Kalman filter [26], Unscented Kalman filter [27], Robust extended Kalman filter [28] เป็นต้น และหนึ่งในบรรดาการพัฒนาของอัลกอริทึมตัวกรองคาลมานที่น่าสนใจและประสบความสำเร็จคืออัลกอริทึมตัวกรองคาลมานที่ใช้กฎของ Spherical-radial cubature ซึ่งมีชื่อว่าอัลกอริทึม Cubature Kalman filter (CKF) [29] และส่วนขยายของอัลกอริทึมนี้ Square-root cubature Kalman filter (SCKF) [30]

อัลกอริทึม Cubature Kalman filter ดังกล่าวมีกระบวนการคำนวณที่คล้ายกับอัลกอริทึมส่วนขยายของตัวกรองคาลมาน (Extended Kalman filter) เพียงแต่ดัดแปลงวิธีการหาค่าพยากรณ์ของสถานะระบบ (State) และค่าแปรปรวนร่วมเกี่ยวให้มีประสิทธิภาพเพิ่มมากขึ้น เนื่องจากอัลกริทึมดังกล่าวไม่ใช้กระบวนการของการแปลงระบบสมการให้เป็นเชิงเส้น แต่เริ่มต้นด้วยการสร้างตัวกรองเป็นไม่เป็นเชิงเส้น ทำให้ได้อัลกอริทึมนี้สามารถใช้งานกับระบบที่มีไม่เป็นเชิงเส้นและมีสิ่งรบกวนแบบเกาส์เซียนได้ดี สำหรับขั้นตอนการคำนวณของอัลกอริทึม Cubature Kalman filter (CKF) ดังนี้
1) ขั้นตอนการพยากรณ์ (Prediction)

$$\mathbf{x}_{k|k-1} = \frac{1}{2n}\sum_{i=1}^{2n} f\left(\boldsymbol{\chi}_{i,k|k-1}\right) \tag{2.11}$$

$$\mathbf{P}_{k|k-1} = \frac{1}{2n}\sum_{i=1}^{2n} f\left(\boldsymbol{\chi}_{i,k|k-1}\right) f\left(\boldsymbol{\chi}_{i,k|k-1}\right)^T - \mathbf{x}_{k|k-1}\mathbf{x}_{k|k-1}^T, + \mathbf{Q}_{k-1} \tag{2.12}$$

โดยกำหนดให้ค่าพยากรณ์ของจุด Cubature ($\boldsymbol{\chi}_{k|k-1}$) ซึ่งมีค่าเป็น

$$\boldsymbol{\chi}_{i,k|k-1} = \mathbf{S}_{k-1}\varsigma_i + \mathbf{x}_{k-1}, \mathbf{S}_{k-1} = \sqrt{\mathbf{P}_{k-1}}\,, \tag{2.13}$$

และค่า ζ มีค่าเท่ากับ

$$\varsigma_i = \begin{cases} \sqrt{n}\mathbf{I}_i & i = 1,2,...,n \\ -\sqrt{n}\mathbf{I}_{i-n} & i = n+1, n+2,...,2n \end{cases} \tag{2.14}$$

โดยที่ n คือจำนวนตัวแปรของสถานะระบบ และ $\mathbf{I}$ คือคอลัมลำดับที่ i

2) ขั้นตอนการพิสูจน์ความถูกต้อง (Correction)

$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + \mathbf{K}_k\left(y_d - y_{k|k-1}\right) \tag{2.15}$$

$$\mathbf{K}_k = \frac{1}{2n}\sum_{i=1}^{2n}\mathbf{X}_{i,k|k-1}h\left(\mathbf{X}_{i,k|k-1}\right)^T - \mathbf{x}_{k|k-1}\mathbf{y}_{k|k-1}^T\left[\frac{1}{2n}\sum_{i=1}^{2n}h\left(\mathbf{X}_{i,k|k-1}\right)h\left(\mathbf{X}_{i,k|k-1}\right)^T - \mathbf{y}_{k|k-1}\mathbf{y}_{k|k-1}^T + \mathbf{R}_k\right]^{-1} \tag{2.16}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\left(\frac{1}{2n}\sum_{i=1}^{2n}h\left(\mathbf{X}_{i,k|k-1}\right)h\left(\mathbf{X}_{i,k|k-1}\right)^T - \mathbf{y}_{k|k-1}\mathbf{y}_{k|k-1}^T + \mathbf{R}_k\right)\mathbf{K}_k^T \tag{2.17}$$

โดยกำหนดให้ค่าของจุด Cubature ($\boldsymbol{X}_{i,k|k-1}$) และค่าของพยากรณ์เอาต์พุตมีค่าเป็น

$$\mathbf{X}_{i,k|k-1} = \mathbf{S}_{k|k-1}\varsigma_i + \mathbf{x}_{k|k-1}, \mathbf{S}_{k|k-1} = \sqrt{\mathbf{P}_{k|k-1}} \tag{2.18}$$

$$\mathbf{y}_{k|k-1} = \frac{1}{2n}\sum_{i=1}^{2n}h\left(\mathbf{X}_{i,k|k-1}\right) \tag{2.19}$$

และเพื่อหลีกเลี่ยงการเกิดเมตริกที่ไม่สามารถหาอินเวอร์สได้ (หรือที่เรียกว่า Singularity matrix) จึงได้ปรับปรุงการหาค่าตัวแปรต่าง ๆ ในขั้นตอนของการพยากรณ์ (Prediction) และขั้นตอนการพิสูจน์ความถูกต้อง (Correction) ใหม่ ซึ่งเรียกวิธีการดังกล่าวว่า Square-root cubature Kalman filter (SCKF) ซึ่งมีรายละเอียดดังนี้

1) ขั้นตอนการพยากรณ์ (Prediction)

ขั้นตอนนี้ส่วนที่มีความแตกต่างจากวิธีการของ Cubature Kalman filter คือการหาค่าพยากรณ์ของรากของค่าความแปรปรวนร่วมเกี่ยว ($\boldsymbol{S}_{k|k-1}$) ซึ่งมีค่าเท่ากับ

$$\mathbf{S}_{k|k-1} = QR\left(\frac{1}{\sqrt{2n}}\left[f\left(\boldsymbol{\chi}_{1,k|k-1}\right) - \mathbf{x}_{k|k-1} \quad f\left(\boldsymbol{\chi}_{2,k|k-1}\right) - \mathbf{x}_{k|k-1} \quad \cdots \quad f\left(\boldsymbol{\chi}_{2n,k|k-1}\right) - \mathbf{x}_{k|k-1}\right]\right) \tag{2.20}$$

กำหนดให้ $\boldsymbol{QR}(*)$ คือฟังก์ชัน QR decomposition ซึ่งเมตริก $\boldsymbol{S}_{k|k-1}$ เป็นค่าเมตริกสามเหลี่ยมล่าง (Lower Triangular Matrix) ของฟังก์ชัน QR decomposition และในส่วนของค่าพยากรณ์ของจุด Cubature ค่า ($\boldsymbol{\chi}_{k|k-1}$) และค่าพยากรณ์ของสถานะ ($\boldsymbol{x}_{k|k-1}$) ยังคงใช้การคำนวณตามขั้นตอนของอัลกอริทึม Cubature Kalman filter (CKF) ในข้างต้น

2) ขั้นตอนการพิสูจน์ความถูกต้อง (Correction)

$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + \mathbf{K}_k\left(\mathbf{y}_d - \mathbf{y}_{k|k-1}\right) \tag{2.21}$$

$$\mathbf{K}_k = \frac{\left(\mathbf{X}_{k|k-1} \left(\mathbf{Y}_{k|k-1} \right)^T \Big/ \mathbf{S}_{yy,k|k-1}^T \right)}{\mathbf{S}_{yy,k|k-1}} \tag{2.22}$$

$$\mathbf{S}_{k|k} = QR\left(\frac{1}{\sqrt{2n}} \left[f\left(\mathbf{X}_{1,k|k-1}\right) - \mathbf{x}_{k|k} \quad f\left(\mathbf{X}_{2,k|k-1}\right) - \mathbf{x}_{k|k-1} \quad \cdots \quad f\left(\mathbf{X}_{2n,k|k-1}\right) - \mathbf{x}_{k|k-1} \right] \right) \tag{2.23}$$

โดยกำหนดให้ค่ารากของเมตริกความแปรปรวนร่วมเกี่ยวของเอาต์พุต ($\boldsymbol{S}_{zz,k|k-1}$) มีค่าเป็น

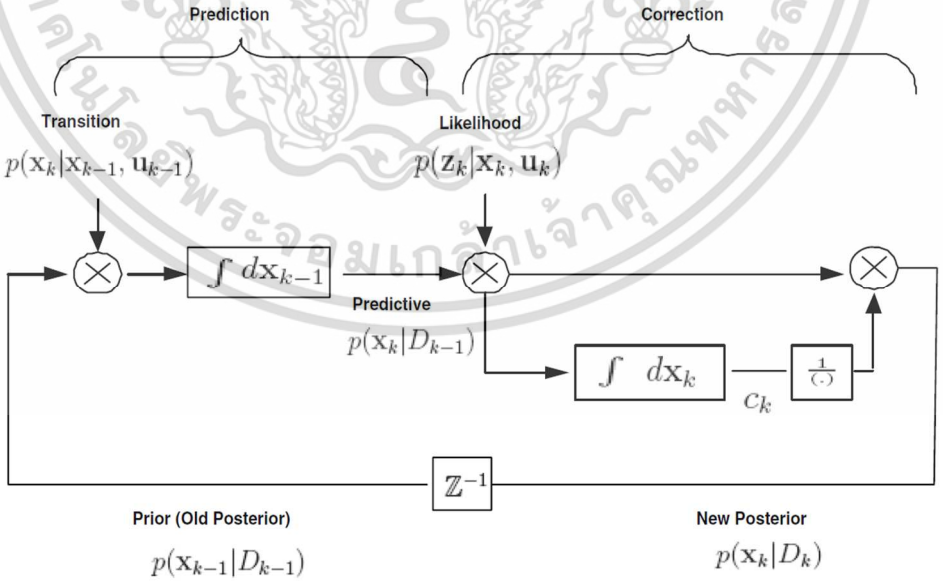
$$\mathbf{S}_{yy,k|k-1} = QR\left(\frac{1}{\sqrt{2n}} \left[h\left(\mathbf{X}_{1,k|k-1}\right) - \mathbf{y}_{k|k-1} \quad \cdots \quad \cdots \quad h\left(\mathbf{X}_{2n,k|k-1}\right) - \mathbf{y}_{k|k-1} \quad \sqrt{\mathbf{R}_k} \right] \right) \tag{2.24}$$

และ $\boldsymbol{y}_{k|k-1}$ และ $\boldsymbol{X}_{k|k-1}$ จะต้องคำนวณด้วยค่าพยากรณ์ของรากของค่าความแปรปรวนร่วมเกี่ยว ($\boldsymbol{S}_{k|k-1}$) ใหม่จากการใช้อัลกอริทึม

$$\mathbf{Y}_{k|k-1} = \left[h\left(\mathbf{X}_{1,k|k-1}\right) - \mathbf{y}_{k|k-1} \quad \cdots \quad \cdots \quad h\left(\mathbf{X}_{2n,k|k-1}\right) - \mathbf{y}_{k|k-1} \right] \tag{2.25}$$

$$\mathbf{X}_{k|k-1} = \left[\mathbf{X}_{1,k|k-1} - \mathbf{x}_{k|k-1} \quad \mathbf{X}_{2,k|k-1} - \mathbf{x}_{k|k-1} \quad \cdots \quad \mathbf{X}_{2n,k|k-1} - \mathbf{x}_{k|k-1} \right] \tag{2.26}$$

และในปัจจุบันก็พบงานวิจัยที่ได้นำอัลกอริทึมไปใช้ในงานด้านต่าง ๆ มากเช่น งานด้านการพยากรณ์การเงิน [31] งานด้านระบบ [32] งานด้านการประมวลผลสัญญาณ [33] เป็นต้น



รูปที่ 2.12 ไดอะแกรมของการเรียนรู้ตัวกรองแบบเบย์ (Bayesian filter algorithm) [34]

ที่มา: Ienkaran A. and Simon H. “Nonlinear Bayesian Filters for Training Recurrent Neural Networks” Springer-Verlag Berlin Heidelberg, vol. ,no, 2008. pp.12-33.

นอกเหนืออัลกอริทึมตัวกรองคาลมานแล้ว อีกหนึ่งอัลกอริทึมที่สำคัญคืออัลกอริทึมตัวกรองการเรียนรู้เบย์ (Bayesian filter algorithm) หากกำหนดให้ระบบพลวัตมีสมการปริภูมิสถานะตามสมการ (2.4) และ (2.5) ดังนั้นการคำนวณหาค่าสถานะใหม่ของระบบด้วยการประมาณค่าจากกระจายตัวของฟังก์ชันความน่าจะเป็นของสถานะปัจจุบัน (New Posterior, $x_{k|k}$) ด้วยกฎของเบย์สามารถหาได้ดังนี้

$$x_{k|k} \approx p\left(x_k \mid \mathrm{D}_k\right) \tag{2.27}$$

เมื่อกำหนดให้ x_k และ D_k คือสถานะของระบบ และเป้าหมายของระบบตามลำดับ โดยสามารถหาค่าสถานะใหม่ของระบบได้ตามไดอะแกรมดังรูปที่ 2.12 ซึ่งอัลกอริทึมมี 2 ขั้นตอนดังนี้1) ขั้นตอนการพยากรณ์การกระจายตัวของสถานะเมื่อกำหนดเอาต์พุตในอดีตล่าสุด ($p(x_k|D_{k-1})$)

$$p(x_k \mid D_{k-1}) = \int_{\Re} p(x_k \mid x_{k-1})\, p(x_{k-1} \mid D_{k-1}) dx_{k-1} \tag{2.28}$$

เมื่อกำหนดให้ค่าการกระจายตัวสถานะในอดีตล่าสุดที่ขึ้นกับเอาต์พุตในอดีตล่าสุด ($p(x_{k-1}|D_{k-1})$) และค่าการกระจายตัวของการเปลี่ยนสถานะปัจจุบันที่ขึ้นกับสถานะในอดีตล่าสุด ($p(x_k|x_{k-1})$) หาได้จากสมการของระบบ

2) ขั้นตอนการหาค่าจากกระจายตัวของฟังก์ชันความน่าจะเป็นของสถานะปัจจุบัน (New Posterior)

$$p(x_k \mid D_k) = \frac{p(y_k \mid x_k)\, p(x_k \mid D_{k-1})}{p(y_k \mid y_{k-1})} \tag{2.29}$$

เมื่อกำหนดให้ค่าการกระจายตัวเอาต์พุตที่ขึ้นกับสถานะล่าสุด ($p(y_k|x_k)$) ซึ่งหาได้จากเอาต์พุตของหาระบบเมื่อใช้การพยากรณ์ และค่าการกระจายตัวเอาต์พุตที่ขึ้นกับเอาต์พุตในอดีตล่าสุด ($p(y_k|D_{k-1})$) ซึ่งหาได้จากสมการ

$$p(y_k \mid D_{k-1}) = \int_{\Re} p(y_k \mid x_k)\, p(x_k \mid D_{k-1}) dx_k \tag{2.30}$$

โดยทั่วไป ค่าสถานะใหม่ของระบบที่หาจากการประมาณค่าจากกระจายตัวของฟังก์ชันความน่าจะเป็นของสถานะปัจจุบัน (New Posterior, $p\left(x_k|D_{k-1}\right)$) ใช้การคำนวณเป็นวนซ้ำตามรอบการทำงานของระบบ ซึ่งในปัจจุบันได้มีงานวิจัยเพื่อพัฒนาในงานด้านต่าง ๆ มากมาย [34-37]

สุดท้ายนี้จะกล่าวถึงอัลกอริทึมที่มีชื่อว่า อัลกอริทึมการเรียนรู้ปฏิบัติ-วิจารณ์แบบเสริมกำลัง (Actor-critic Reinforcement learning algorithm) เป็นอัลกอริทึมที่มีประสิทธิภาพมากอันหนึ่ง

และเป็นหนึ่งในอัลกอริทึมที่ใช้งานด้านระบบที่เป็นพลวัตแบบไม่เป็นเชิงเส้น (Nonlinear Dynamic System) หรือระบบที่เป็นพลวัตแบบสุ่ม (Stochastic Dynamic System) อัลกอริทึมการเรียนรู้ปฏิบัติ-วิจารณ์แบบเสริมกำลังมีหลักการสำคัญอยู่ 2 ประการคือ 1) การพยากรณ์การกระทำ (Actor, u) ที่เหมาะสมจากค่าสถานะ (State, x) ทั้งหมดที่เป็นไปได้ด้วยฟังก์ชัน Policy ($u = \pi(x)$ หรือเรียกว่า "Control Law") และ 2) การหาค่าวิจารณ์ (Critic Value) สำหรับการกระทำที่พยากรณ์ไว้ในข้างต้น (Predicted actor, $\hat{u}$) เพื่อให้ได้ค่าพยากรณ์ที่เหมาะสมที่สุด จึงต้องกำหนดค่าของฟังก์ชันต้นทุน (Cost function) ซึ่งมีค่าเท่ากับ

$$J\left(x_k, u_k\right) = \sum_{k=0}^{n-1} cost\left(x_k, u_k\right) \tag{2.31}$$

กำหนดให้ค่า $cost$ คือค่าของต้นทุน ณ รอบการคำนวณที่ k ใด ๆ ของระบบ ซึ่งสมการในการหาค่าที่เหมาะสมเพื่อให้ระบบอยู่ในสถานะเสถียรประกอบด้วยฟังก์ชัน Value iteration และสมการของฟังก์ชัน Policy ซึ่งมีรายละเอียดดังนี้

$$\pi_k(\mathrm{x}) = \arg\min\left\{J\left(x_k, u_k\right) + v_k\left(next\left(x_k, u_k\right)\right)\right\} \tag{2.32}$$

กำหนดให้ฟังก์ชัน $\arg.\min\{*\}$ คือการหาค่าต่ำสุดของสมการต้นทุน (Cost function, $J(*)$) ส่วนค่า $next(x,u)$ คือค่าพยากรณ์ที่เหมาะสมที่สุดของสถานะต่อไปของระบบ และฟังก์ชัน $v(*)$ คือสมการ Value iteration ซึ่งมีค่าเท่ากับ

$$v_{k+1}(x) = \min\left\{\mathrm{cost}\left(x_k, u_k\right) + v_k\left(next\left(x_{k-1}, u_{k-1}\right)\right)\right\} \tag{2.33}$$

ทั้งสองสมการข้างต้นเรียกว่า "Bellman equation" [38] ซึ่งการหาค่าที่เหมาะสมด้วยวิธีการนี้จะใช้ค่าผิดพลาดที่เกิดจากการเปรียบเทียบระหว่างค่าการประเมินจากค่าการกระทำในปัจจุบันกับค่าในอดีตล่าสุดซึ่งมีค่าเท่ากับ

$$e\left(v_k(x)\right) = \min\left|v_k(x) - v_{k-1}(x)\right| \tag{2.34}$$

สำหรับการใช้งานอัลกอริทึม Actor-critic reinforcement ในปัจจุบันมีการใช้อยู่ 2 แบบด้วยกันคือวิธี Value iteration และวิธี Policy iteration โดยที่วิธีแรกคือการใช้ทั้ง Value iteration จะใช้สมการของ Value iteration โดยที่สมการของค่าการประเมินค่าการกระทำมีค่าเท่ากับ

$$v_k^{\pi^{(i)}}(x) = \left\{ \cos t\left(x_k, u_k^{\pi^{(i)}}\right) + v_k^{\pi^{(i)}}\left(next\left(x_k, u_k^{\pi^{(i)}}\right)\right)\right\} \quad (2.35)$$

สำหรับวิธีที่สองคือการใช้ทั้ง Value iteration และ Policy iteration ซึ่งใช้ทั้งสมการของ Value iteration และสมการของค่าการประเมินค่าการกระทำ (Action-Value Function, $v(x)$) โดยที่สมการของค่าการประเมินค่าการกระทำมีค่าเท่ากับ

$$v_k^{\pi^{(i)}}(x) = \left\{ \cos t\left(x_k, u_k^{\pi^{(i)}}\right) + v_k^{\pi^{(i)}}\left(next\left(x_k, u_k^{\pi^{(i)}}\right)\right)\right\} \quad (2.36)$$

$$\pi_k^{(i+1)}(x) = \arg\min\left\{ \text{cost}\left(x_k, u_k\right) + v_k^{\pi(\text{i})}\left(next\left(x_k, u_k\right)\right)\right\} \quad (2.37)$$

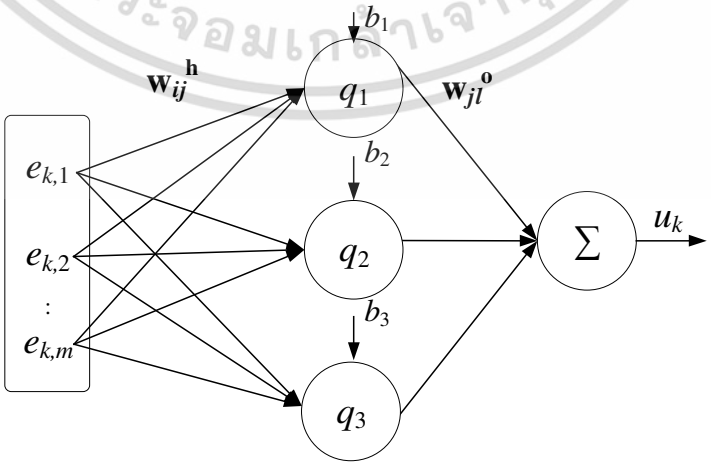
ซึ่งในปัจจุบันได้มีงานวิจัยเพื่อพัฒนาในงานด้านต่าง ๆ มากมาย [39-41]

บทที่ 3
วิธีดำเนินการวิจัย

ในบทนี้จะกล่าวถึงการนำทฤษฎีที่เกี่ยวข้องกับงานวิจัยมาใช้เพื่อดำเนินงานวิจัย ซึ่งในบทนี้จะแบ่งเนื้อหาออกเป็น 3 ส่วนได้แก่ ส่วนแรกคือการออกแบบโครงสร้างของตัวควบคุมที่เลียนแบบ Proportional-Integral-Derivative (PID) ด้วยโครงข่ายประสาทเทียม (NNPID) ส่วนที่สองคือ การใช้อัลกอริทึมการเรียนรู้มาปรับค่าน้ำหนักของตัวควบคุม NNPID เพื่อปรับปรุงประสิทธิภาพ ซึ่งในวิทยานิพนธ์ฉบับนี้จะนำเสนออัลกอริทึมการเรียนรู้ทั้งหมด 4 อัลกอริทึมได้แก่ อัลกอริทึมการเรียนรู้แบบส่วนขยายของตัวกรองคาลมาน (mEKF algorithm) อัลกอริทึมเรียนรู้แบบผสมตัวกรองคาล (Hybrid Cubature Kalman Filter พร้อมด้วย Hybrid Square-root Cubature Kalman Filter) อัลกอริทึมเรียนรู้แบบตัวกรองเบย์ประยุกต์ (Applied Bayesian Filter) และอัลกอริทึมเรียนรู้แบบผสมระหว่างอัลกอริทึมการเรียนรู้ปฏิบัติ-วิจารณ์แบบเสริมกำลังกับอัลกอริทึมตัวกรองคาลมานดีกรีสูง (Hybrid of the Actor-critic Reinforcement with High Order Square-root Cubature Kalman Filter) และส่วนสุดท้ายจะเป็นการกล่าวถึงการสร้างแบบจำลองของระบบควบคุมแบบปิดเพื่อเปรียบเทียบประสิทธิภาพการใช้งานตัวควบคุมแบบต่าง ๆ

3.1 การออกแบบโครงสร้างของตัวควบคุมโครงข่ายประสาทเทียม (NNPID)

การออกแบบโครงสร้างของโครงข่ายประสาทเทียมเพื่อเลียนแบบตัวควบคุม PID หรือที่เรียกว่า ตัวควบคุม NNPID จะใช้โนด (Neuron) ในชั้นซ่อน (Hidden Layer) ของโครงข่ายเป็นตัวเลียนแบบข้อกำหนดของตัวควบคุม PID ดังสมการที่ (2.1) ซึ่งมีโครงสร้างดังรูปที่ 3.1



รูปที่ 3.1 ตัวควบคุมโครงข่ายประสาทเทียมที่เลียนแบบโครงสร้างของตัวควบคุม PID

จากโครงสร้างของตัวควบคุม NNPID ในข้างต้นจะเห็นว่าตัวควบคุมสามารถรองรับอินพุตของระบบได้หลายตัวแปร โดยมีเอาต์พุตเพียงแค่ตัวเดียว เรียกว่า “ระบบแบบอินพุตหลายตัวแปรและเอาต์พุตตัวแปรเดียว (Multi-Input Single-Output, **MISO**)” ทำให้ตัวควบคุมชนิดนี้มีความสามารถที่เหนือกว่าตัวควบคุม PID ทั่วไปที่สามารถรองรับอินพุตได้เพียงตัวแปรเดียวเท่านั้น ซึ่งเรียกว่าระบบควบคุมแบบตัวแปรเดียว (Single-Input Single-Output, **SISO**)

จากรูปที่ 3.1 แสดงถึงโครงข่ายประสาทเทียม 3 ชั้นประกอบด้วยชั้นอินพุต m เซลล์ประสาท ชั้นซ่อน 3 เซลล์ประสาท (จำนวนเซลล์ประสาทของชั้นซ่อนได้กำหนดตามสมการตัวควบคุม PID) และสุดท้ายชั้นเอาต์พุต 1 เซลล์ประสาท ซึ่งสมการสำหรับคำนวณค่าต่าง ๆ ของแต่ละเซลล์ประสาทในชั้นซ่อนสามารถกำหนดได้ดังนี้

$$a_{1,k} = g\left(\sum_{i=1}^{m} w_{1i,k}^{h} e_{i,k}\right) \tag{3.1}$$

$$a_{2,k} = g\left(\sum_{i=1}^{m} w_{2i,k}^{h} e_{i,k} + a_{2,k-1}\Delta t\right) \tag{3.2}$$

$$a_{3,k} = g\left(\left(\sum_{i=1}^{m} w_{3i,k}^{h} e_{i,k} - \sum_{i=1}^{m} w_{3i,k-1}^{h} e_{i,k-1}\right)\frac{1}{\Delta t}\right) \tag{3.3}$$

ส่วนชั้นเอาต์พุตสามารถคำนวณได้ดังนี้

$$u_k = \sum_{l=1}^{3} w_{l,k}^{y} a_{l,k} + b^{y} \tag{3.4}$$

โดยกำหนดให้ $e_{i,k}$ คืออินพุตของตัวควบคุมแทนด้วยค่าผิดพลาดที่เกิดขึ้นจากระบบ (ตัวแปร i บ่งชี้ถึงตัวควบคุมชนิดนี้สามารถใช้กับระบบควบคุมที่มีได้หลายอินพุต) ซึ่งก็คือค่าความแตกต่างระบบค่าเป้าหมายและค่าเอาต์พุตที่วัดได้จริง ณ รอบของการคำนวณลำดับที่ k โดยระหว่างเซลล์ประสาทของชั้นอินพุตของตัวควบคุมจะถูกเชื่อมต่อเข้ากับเซลล์ประสาทในชั้นซ่อนด้วยน้ำหนักที่ซึ่งแทนด้วย w^h สำหรับเซลล์ประสาทในชั้นซ่อนจะมีการแปลงค่าด้วยฟังก์ชันกระตุ้น (Activation function) ที่กำหนดด้วย $g(*)$ และตัวอย่างของฟังก์ชันกระตุ้นได้แก่ Logistic function, Tanh function, Sigmoid function เป็นต้น โดยในที่นี่แต่ละเซลล์ประสาทของชั้นซ่อนนั้นจะกำหนดรูปแบบของการคำนวณที่มีความแตกต่างตามโครงสร้างของตัวควบคุม PID กล่าวคือเซลล์ประสาทในชั้นซ่อนตัวที่ 1 (เซลล์ประสาท $a_{1,k}$) ตัวแทนด้วยตัวคูณ K_P ของตัวควบคุม PID ซึ่งจะมีค่าไบอัสเป็นศูนย์ ต่อมาเซลล์ประสาทในชั้นซ่อนตัวที่ 2 (เซลล์ประสาท $a_{2,k}$) แทนด้วยตัวคูณ K_I ของตัวควบคุม PID ซึ่งเซลล์ประสาทตัวนี้จะมีตัวไบอัสตามค่าของพจน์ปริพันธ์ซึ่งเป็นค่าของเซลล์ประสาทในอดีตล่าสุดคูณอัตรา

เวลาในการสุ่ม (Δt) และสุดท้ายเซลล์ประสาทในชั้นซ่อนตัวที่ 3 (เซลล์ประสาท $a_{3,k}$) จะเป็นตัวแทนของตัวคูณ K_D ของตัวควบคุม PID ซึ่งเซลล์ประสาทในชั้นนี้เป็นการเปรียบเทียบค่าปัจจุบันกับค่าตัวไบอัสตามค่าของพจน์ของอนุพันธ์ซึ่งเป็นค่าของเซลล์ประสาทในอดีตอันล่าสุดแล้วคูณอัตราเวลาในการสุ่ม (Δt)

สำหรับชั้นเอาต์พุตเป็นชั้นของการคำนวณค่าอินพุตควบคุม (Control Input) ให้กับระบบควบคุมที่มีการกำหนดค่าเป็นผลรวมจากเซลล์ประสาททั้ง 3 ในชั้นนี้เองจะมีน้ำหนักซึ่งแทนด้วย w^y เชื่อมเซลล์ประสาทในซ่อนกับเซลล์ประสาทในชั้นเอาต์พุต และในชั้นเอาต์พุตนี้จะไม่มีการใช้ฟังก์ชันกระตุ้นเนื่องจากเป็นชั้นที่จะต้องกำหนดค่าให้เป็นไปตามค่าสัญญาณของระบบควบคุม และจากสมการของตัวควบคุม PID แบบทั่วไปทำให้ค่าไบอัสในชั้นเอาต์พุตกำหนดให้เป็นศูนย์

เมื่อพิจารณาค่าจากการใช้ฟังก์ชันกระตุ้น (Activation function) พบว่าค่าที่ได้หลังจากการแปลงอยู่ในช่วงที่กำหนดตามฟังก์ชันนั้น ๆ และเมื่อต้องการหาจุดต่ำสุดของค่าผิดพลาดทำได้ง่ายขึ้นนั้นเอง กล่าวคือเมื่อค่าของเอาต์พุตแต่ละเซลล์ประสาทที่ผ่านฟังก์ชันกระตุ้นมีค่าต่ำสุด (Stationary) อัลกอริทึมก็จะสามารถค้นหาค่าน้ำหนักโครงข่ายประสาทเทียมที่เหมาะสมสำหรับการการสร้างตัวอินพุตควบคุม (Control Input) ที่เหมาะสมเพื่อชดเชยระบบควบคุมให้ค่าผิดพลาดเป็นศูนย์ได้ง่ายมากขึ้น ในงานวิจัยนี้ได้เลือกใช้ฟังก์ชันที่ชื่อว่า tanh function ซึ่งมีความเหมาะสมสำหรับแปลงค่าที่สามารถเป็นได้ทั้งค่าบวก และค่าลบ แต่อย่างไรก็ตาม ฟังก์ชันของเซลล์ประสาทในชั้นเอาต์พุตกำหนดให้ใช้ฟังก์ชันที่เป็นเชิงเส้น (Linear function) เนื่องจากต้องการสร้างอินพุตควบคุมที่มีค่าที่เป็นตามค่าจริงของระบบ กล่าวคือค่าในชั้นเอาต์พุตเป็นการรวมค่าที่ได้จากเซลล์ประสาทของชั้นซ่อนที่คูณด้วยน้ำหนักของที่เชื่อมระหว่างกันซึ่งเป็นค่าที่ได้จากการแปลงที่อยู่ในช่วงค่าน้อย ๆ ทำให้ไม่สามารถนำไปใช้ควบคุม จึงต้องขยายสัญญาณเหล่านั้นให้เหมาะสมก่อนการนำมาใช้งานจริง

3.2 การออกแบบอัลกอริทึมการเรียนรู้ที่นำเสนอเพื่อปรับปรุงประสิทธิภาพของตัวควบคุม NNPID

โดยทั่วไป งานควบคุมมักจะพิจารณาถึงพารามิเตอร์หรือแบบจำลองของระบบ ตัวอย่างเช่น เมื่อต้องการสร้างระบบควบคุมแขนกลมอเตอร์ไฟฟ้า ก็จะต้องทราบพารามิเตอร์ต่าง ๆ ของระบบ ได้แก่ ค่าพารามิเตอร์ทางด้านไฟฟ้า พารามิเตอร์ทางด้านฟิสิกส์ เป็นต้น ซึ่งค่าพารามิเตอร์เหล่านั้นทำให้ผู้ออกแบบสามารถหาแบบจำลองของระบบสำหรับออกแบบตัวควบคุมที่เหมาะสม แต่ในวิทยานิพนธ์ฉบับนี้นำเสนอแนวคิดใหม่ที่ไม่ต้องคำถึงตัวแบบจำลองของระบบ หรือพารามิเตอร์ใด ๆ ของระบบ หรือแม้กระทั้งปัจจัยแวดล้อมที่ส่งผลต่อระบบ ซึ่งในหัวข้อนี้จะกล่าวถึงการสร้างอัลกอริทึมการเรียนรู้สำหรับใช้งานกับตัวควบคุม ซึ่งประกอบด้วย 4 อัลกอริทึมได้แก่ 1) อัลกอริทึมการเรียนรู้แบบส่วนขยายของตัวกรองคาลมานเรียกว่า modified Extended Kalman filter learning algorithm (mEKF algorithm) 2) อัลกอริทึมการเรียนรู้แบบไฮบริดของตัวกรองคาลมานด้วยกฎ

Cubature เรียกว่า a Hybrid Cubature Kalman filter learning algorithm (Hybrid CKF algorithm) 3) อัลกอริทึมการเรียนรู้ด้วยกฎเบย์ประยุกต์เรียกการเรียนรู้นี้ว่า an Applied Bayesian filter learning algorithm (ABF algorithm) และ 4) อัลกอริทึมการเรียนรู้แบบผสมระหว่างอัลกอริทึมปฏิบัติ-วิจารณ์แบบเสริมกำลังและอัลกอริทึมตัวกรองคาลมานดีกรีสูง (Hybrid of the Actor-critic Reinforcement and High Order Square-root Cubature Kalman Filter, เรียกสั้นๆ ว่า AC reinforcement algorithm)

อัลกอริทึมการเรียนรู้ทั้ง 4 เป็นอัลกอริทึมสำหรับการปรับค่าน้ำหนัก (สถานะ) ของตัวควบคุม NNPID เพื่อรองรับการเปลี่ยนแปลงของระบบควบคุมพลวัตที่ไม่เป็นเชิงเส้น (Nonlinear Dynamic Control System) และเนื่องจากการออกแบบตัวควบคุมที่นำเสนอนี้ไม่ใช้พารามิเตอร์ของระบบ เพราะฉะนั้นในวิทยานิพนธ์เล่มจึงมุ่นเน้นการออกแบบอัลกอริทึมการเรียนรู้ (Learning Algorithm) ของระบบด้วยการพิจารณาการเปลี่ยนแปลงของตัวควบคุมเป็นสำคัญ กล่าวคือในขั้นแรกของการออกแบบระบบควบคุมจะต้องกำหนดแบบจำลองปริภูมิสถานะ (State Space Model) เพื่อของตัวควบคุมเสียก่อน ซึ่งกำหนดให้น้ำหนักของตัวควบคุม NNPID เป็นสถานะของแบบจำลองปริภูมิสถานะ (แทนด้วยเวกเตอร์ $\mathbf{w}$) ประกอบด้วยน้ำหนักของโครงข่ายประสาทเทียมที่เชื่อมต่อระหว่างชั้นอินพุตกับชั้นซ่อน แทนด้วยเมตริกดังนี้

$$\mathbf{w}^h = \begin{bmatrix} w_{11}^h & \cdots & w_{1j}^h & b_1^h \\ w_{21}^h & \cdots & w_{2j}^h & b_2^h \\ w_{31}^h & \cdots & w_{3j}^h & b_3^h \end{bmatrix} \tag{3.5}$$

และน้ำหนักโครงข่ายที่เชื่อมต่อระหว่างชั้นซ่อนกับชั้นเอาต์พุตคือ

$$\mathbf{w}^y = \begin{bmatrix} w_1^y \\ w_2^y \\ w_3^y \end{bmatrix} \tag{3.6}$$

สำหรับเอาต์พุตของตัวควบคุม NNPID กำหนดให้เป็นตัวอินพุตควบคุม (Control input, u) ซึ่งในแต่ละอัลกอริทึมการเรียนรู้มีรายละเอียดดังต่อไปนี้

3.2.1 อัลกอริทึมการเรียนรู้ mEKF algorithm

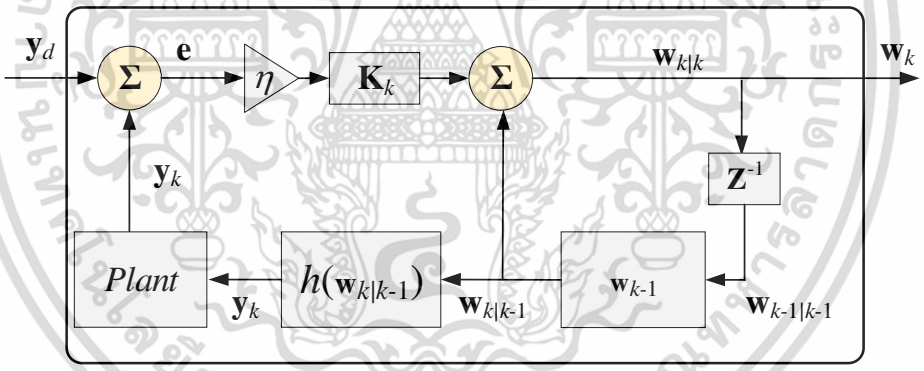
แนวคิดของอัลกอริทึมการเรียนรู้ของ mEKF algorithm คือการใช้อัลกอริทึมคาลมานมาใช้เพื่อเรียนรู้การเปลี่ยนแปลงพฤติกรรมของการเปลี่ยนแปลงของน้ำหนักของตัวควบคุม NNPID และเนื่องจากการปรับค่าน้ำหนักของตัวควบคุม NNPID นั้นขึ้นอยู่กับระบบ (Plant) ซึ่งส่วนใหญ่ระบบ

ควบคุมมักจะเป็นระบบที่เป็นพลวัตแบบไม่เป็นเชิงเส้น ดังนั้นในขั้นตอนแรกจึงกำหนดให้การเปลี่ยนแปลงของน้ำหนักของตัวควบคุม NNPID ด้วยสมการแบบจำลองปริภูมิสถานะ (State Space Model) ดังนี้

$$\mathbf{w}_k = \mathbf{w}_{k-1} + \boldsymbol{\omega}_{k-1} \tag{3.7}$$

$$y_k = h(\mathbf{w}_k) + \mathbf{v}_k \tag{3.8}$$

โดยกำหนดให้ $\mathbf{w}_k$ คือเมตริกของน้ำหนักของตัวควบคุมแบบโครงข่ายประสาทเทียมที่เลียนแบบตัวควบคุม PID (ตัวควบคุม NNPID) ณ รอบการคำนวณที่ k ใด ๆ ซึ่งกำหนดให้เป็นสถานะของระบบ และ $\mathbf{y}_k$ คือเอาต์พุตของระบบ ส่วน $\boldsymbol{\omega}_k$ และ $\mathbf{v}_k$ คือปัจจัยแวดล้อมที่รบกวนจากแบบจำลองของตัวควบคุมและปัจจัยแวดล้อมที่รบกวนจากแบบจำลองของระบบ ณ รอบการคำนวณที่ k ใด ๆ ตามลำดับ สุดท้ายฟังก์ชัน $h(*)$ คือฟังก์ชันของแบบจำลองของระบบที่ต้องการควบคุม (Plant) เราเรียกตัวควบคุม NNPID ที่ใช้อัลกอริทึมนี้ว่า “ตัวควบคุม ENNPID” ซึ่งมีไดอะแกรมของแนวคิดอัลกอริทึมนี้ดังรูปที่ 3.2



รูปที่ 3.2 อัลกอริทึมการเรียนรู้แบบส่วนขยายของตัวกรองคาลมาน (mEKF algorithm)

จากรูปที่ 3.2 เป็นแนวคิดของอัลกอริทึมส่วนขยายของตัวกรองคาลมานที่นำเสนอ (mEKF Algorithm) ในการคำนวณเพื่อปรับค่าน้ำหนักของตัวควบคุม ENNPID ของระบบพลวัตแบบไม่เป็นเชิงเส้นใด ๆ เข้าสู่สถาวะเสถียรซึ่งมีขั้นตอนดังนี้

1) ขั้นตอนการพยากรณ์ค่าแปรปรวนร่วมเกี่ยว ($\mathbf{P}_{k|k-1}$)

$$\mathbf{P}_{k|k-1} = \mathbf{H}(\mathbf{w}_{k-1})\mathbf{P}_{k-1}\mathbf{H}(\mathbf{w}_{k-1}) \tag{3.9}$$

2) ขั้นตอนการพิสูจน์ความถูกต้องของการพยากรณ์ด้วยค่าคงที่คาลมาน ($\mathbf{K}_k$)

$$\mathbf{w}_{k|k} = \mathbf{w}_{k-1} + \eta \mathbf{K}_k \left(y_d - y_k \right) \tag{3.10}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}^T\left(\mathbf{w}_{k|k-1}\right)\left[\mathbf{H}\left(\mathbf{w}_{k|k-1}\right)\mathbf{P}_{k|k-1}\mathbf{H}^T\left(\mathbf{w}_{k|k-1}\right) + \mathbf{R}_k\right]^{-1} \tag{3.11}$$

$$\mathbf{P}_{k|k} = \left(\mathbf{I} - \mathbf{K}_k\mathbf{J}\left(\mathbf{w}_{k|k-1}\right)\right)\mathbf{P}_{k|k-1} \tag{3.12}$$

โดยกำหนดให้ y_d คือค่าเป้าหมายที่ต้องการ สำหรับ $\mathbf{P}$ และ $\mathbf{R}$ คือค่าแปรปรวนร่วมเกี่ยวของระบบและเมตริกของค่าผิดพลาดที่เกิดจากการวัดค่าเอาต์พุต ตามลำดับ โดยที่ $\mathbf{R}$ มีค่าเป็น

$$\mathbf{R}_k = \mathbf{R}_{k-1} + \frac{1}{k}\left(\left(y_d - y_k\right)\left(y_d - y_k\right)^T - \mathbf{R}_{k-1}\right) \tag{3.13}$$

ส่วนเมตริก $\mathbf{H}(\mathbf{w}_k)$ คือเมตริก Jacobian ของฟังก์ชันค่าความผิดพลาดที่เกิดขึ้นของระบบซึ่งมีค่าเป็น

$$\mathbf{H}_k = \frac{\partial y_k}{\partial \mathbf{w}_k} = \begin{bmatrix} \frac{\partial y_k}{\partial \mathbf{w}_k^h} & \frac{\partial y_k}{\partial \mathbf{w}_k^y} \end{bmatrix} \tag{3.14}$$

สำหรับการคำนวณเพื่อหาค่าเมตริกของ Jacobian สามารถแยกการพิจารณาได้ดังนี้

$$\mathbf{H}_k^h = \frac{\partial y_k}{\partial \mathbf{w}_k^h} = \left[\frac{\partial y_k}{\partial a_k}\frac{\partial a_k}{\partial n_k^h}\frac{\partial n_k^h}{\partial \mathbf{w}_k^h}\right]$$

$$\mathbf{H}_k^h = \begin{bmatrix} \frac{\partial y_k}{\partial a_{1,k}} & \frac{\partial y_k}{\partial a_{2,k}} & \frac{\partial y_k}{\partial a_{3,k}} \end{bmatrix} \times \begin{bmatrix} \frac{\partial a_{1,k}}{\partial n_{1,k}^h} & \frac{\partial a_{1,k}}{\partial n_{2,k}^h} & \frac{\partial a_{1,k}}{\partial n_{3,k}^h} \\ \frac{\partial a_{2,k}}{\partial n_{1,k}^h} & \frac{\partial a_{2,k}}{\partial n_{2,k}^h} & \frac{\partial a_{2,k}}{\partial n_{3,k}^h} \\ \frac{\partial a_{3,k}}{\partial n_{1,k}^h} & \frac{\partial a_{3,k}}{\partial n_{2,k}^h} & \frac{\partial a_{3,k}}{\partial n_{3,k}^h} \end{bmatrix} \begin{bmatrix} \frac{\partial n_{1,k}^h}{\partial w_{11,k}^h} & \cdots & \frac{\partial n_{1,k}^h}{\partial b_{3,k}^h} \\ \frac{\partial n_{2,k}^h}{\partial w_{11,k}^h} & \ddots & \vdots \\ \frac{\partial n_{3,k}^h}{\partial w_{11,k}^h} & \cdots & \frac{\partial n_{3,k}^h}{\partial b_{3,k}^h} \end{bmatrix} \tag{3.15}$$

กำหนดให้ $n_{1,k}^h$, $n_{2,k}^h$, และ $n_{3,k}^h$ คือค่าของโนดในชั้นซ่อนตัวควบคุม ENNPID ซึ่งมีค่าเท่ากับ

$$n_{1,k}^h = \left(e_1 w_{11}^h + \cdots + e_m w_{1m}^h\right) + b_1^h \tag{3.16}$$

$$n_{2,k}^h = \left(e_1 w_{21}^h + \cdots + e_m w_{2m}^h\right) + b_2^h \tag{3.17}$$

$$n_{3,k}^h = \left(e_1 w_{31}^h + \cdots + e_m w_{3m}^h\right) + b_3^h \tag{3.18}$$

และค่าของเมตริก $\mathbf{H}^{y}_{k}$ มีค่าเท่ากับ

$$\begin{aligned}\mathbf{H}^{y}_{k} &= \frac{\partial y_k}{\partial \mathbf{w}^{y}_{k}} = \left[\frac{\partial y_k}{\partial a_k}\frac{\partial a_k}{\partial n^{y}_{k}}\frac{\partial n^{y}_{k}}{\partial \mathbf{w}^{y}_{k}}\right] \\ &= \left[g'\left(n^{y}_{k}\right)\right]\left[\frac{\partial n^{y}_{k}}{\partial w^{y}_{1,k}} \quad \frac{\partial n^{y}_{k}}{\partial w^{y}_{2,k}} \quad \frac{\partial n^{y}_{k}}{\partial w^{y}_{3,k}} \quad \frac{\partial n^{y}_{k}}{\partial b^{y}_{k}}\right]\end{aligned} \tag{3.19}$$

กำหนดให้ n^{y}_{k} คือของโนดในชั้นเอาต์พุตของตัวควบคุม ENNPID ซึ่งมีค่าเท่ากับ

$$n^{y}_{k} = \left(a_1 w_1^{y} + a_2 w_2^{y} + a_3 w_3^{y}\right) \tag{3.20}$$

อัลกอริทึมเพื่อการปรับค่าน้ำหนักของตัวควบคุม ENNPID ดังอัลกอริทึมที่ 1

<u>อัลกอริทึม 1</u> อัลกอริทึมการเรียนรู้ด้วยตัวกรองคาลมาน (mEKF Algorithm)

Initialization: $\mathbf{w}_0$, $\mathbf{P}_0$, $\mathbf{K}_0$, $\mathbf{J}_0$, $\eta = 0.01$, $\mathbf{R}_0$
for: each step **do** until $\boldsymbol{J}(\mathbf{w}_k)$ meet condition
  predict the covariance: $\mathbf{P}_{k|k-1}$
  calculate the Jacobian matrix $\mathbf{H}_k$:
  calculate the error: $\mathbf{R}_k$
  evaluate the state-value by Kalman gain: $\mathbf{K}_k$
  evaluate the covariance: $\mathbf{P}_{k|k}$
  calculate the cost function:

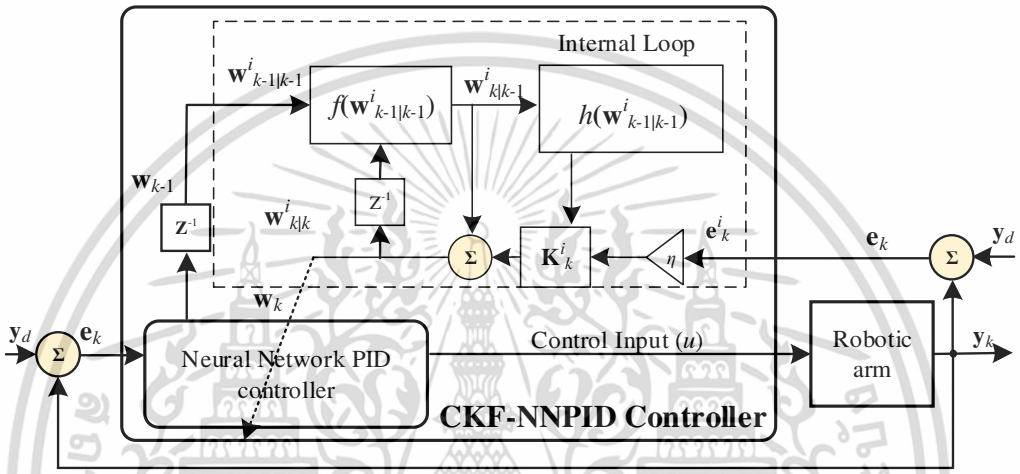
$$J\left(\mathbf{w}\right) = \frac{1}{n}\sum_{i=1}^{n}\left(e_k^{i}\right)^2$$

  update $\mathbf{w}_{k|k}$, $\mathbf{P}_{k|k}$
end for

3.2.2 อัลกอริทึมการเรียนรู้แบบไฮบริด CKF algorithm

เพื่อปรับปรุงประสิทธิภาพของระบบควบคุมให้ดียิ่งขึ้น วิทยานิพนธ์ฉบับนี้จึงมุ่งเน้นการหาอัลกอริทึมสำหรับการปรับค่าน้ำหนักที่เหมาะสมให้กับตัวควบคุม NNPID ซึ่งจากหัวข้อที่แล้วที่ได้กล่าวถึงตัวควบคุม ENNPID [44] ที่ได้เสนออัลกอริทึมการเรียนรู้แบบส่วนขยายของตัวกรองคาลมาน (mEKF algorithm) และถึงแม้ว่าตัวควบคุมดังกล่าวมีประสิทธิภาพที่เหนือกว่าอัลกอริทึมที่ผ่าน แต่ก็ยังพบข้อบกพร่องอีกมากมายเช่น การตั้งค่าเริ่มต้นของน้ำหนัก ($\mathbf{w}_0$) การออกแบบของอัลกอริทึมที่ต้องใช้สมการอนุพันธ์ทำให้เกิดการสูญเสียข้อมูลไปบางส่วน และส่วนค่าภาวะชั่วครู่ (Transient) ที่

ยังคงต้องปรับปรุงต่อไป เป็นต้น และเพื่อพัฒนาตัวควบคุม NNPID ให้มีความสามารถขึ้นไปอีกขั้นหนึ่ง ดังนั้นงานวิจัยนี้จึงนำเสนอตัวควบคุมใหม่ที่มีชื่อว่า ตัวควบคุม Hybrid CKF-NNPID ซึ่งมีไดอะแกรมของอัลกอริทึมนี้ดังรูปที่ 3.3 ตัวควบคุม Hybrid CKF-NNPID นี้ใช้อัลกอริทึมการเรียนรู้แบบไฮบริดของตัวกรองคาลมานด้วยกฎ Cubature Kalman Filter (CKF) ซึ่งเป็นตัวกรองคาลมานที่ใช้งานอย่างแพร่หลายและมีประสิทธิภาพ อัลกอริทึมชนิดนี้ได้ใช้การออกแบบที่อยู่บนพื้นฐานของตัวกรองแบบไม่เป็นเชิงเส้น และด้วยเหตุนี้ทำให้ตัวควบคุมที่นำเสนอนี้สามารถแก้ไขปัญหาในเรื่องข้อบกพร่องดังกล่าวที่ข้างต้นได้



รูปที่ 3.3 ระบบควบคุมที่ใช้ตัวควบคุม Hybrid CKF-NNPID ซึ่งใช้อัลกอริทึมการเรียนรู้แบบไฮบริดของตัวกรองคาลมาน

สำหรับขั้นตอนการคำนวณของอัลกอริทึมนี้เริ่มต้นด้วยการกำหนดพฤติกรรมการเปลี่ยนแปลงของน้ำหนักของตัวควบคุม NNPID ต่อระบบ (Plant) ด้วยสมการปริภูมิสถานะซึ่งพัฒนามาจากจากสมการที่ (3.7) และ (3.8) ดังนี้

$$\mathbf{w}_k = f\left(\mathbf{w}_{k-1}\right) + \boldsymbol{\omega}_{k-1} \tag{3.21}$$

$$y_k = h\left(\mathbf{w}_k\right) + \mathbf{v}_k \tag{3.22}$$

โดยกำหนดให้ $f(*)$ และ $h(*)$ คือฟังก์ชันของแบบจำลองของการพยากรณ์น้ำหนักโครงข่ายและฟังก์ชันของแบบจำลองของระบบ (Plant) ตามลำดับ อัลกอริทึมการเรียนรู้ของตัวควบคุม Hybrid CKF-NNPID ประกอบด้วยแนวคิดที่สำคัญได้แก่ ประการที่หนึ่งคือเพิ่มส่วนของการคำนวณภายในลูป (Inner loop) เพื่อขจัดปัญหาเรื่องของการกำหนดค่าน้ำหนักเริ่มต้นของตัวควบคุม ประการที่สองคือการใช้หลักการคำนวณค่าพยากรณ์น้ำหนักและค่าความแปรปรวนร่วมเกี่ยวของกฎ Cubature ทำให้ค่าน้ำหนักที่ถูกปรับเป็นไปได้อย่างมีประสิทธิภาพ และมีผลตอบสนองที่เหนือกว่าตัวควบคุม ENNPID

ในส่วนของขั้นตอนการคำนวณของอัลกอริทึมการเรียนรู้แบบไฮบริดของตัวกรองคาลมานของตัวควบคุม Hybrid CKF-NNPID มีดังนี้

1) ขั้นตอนการพยากรณ์น้ำหนักของตัวควบคุม Hybrid CKF-NNPID ($\mathbf{w}_{k|k\text{-}1}$) และ ค่าแปรปรวนร่วมเกี่ยว ($\mathbf{P}_{k|k\text{-}1}$)

$$\mathbf{w}_{k|k-1} = \frac{1}{2n}\sum_{l=1}^{2n}\mathbf{W}^{*}_{l,k|k-1} \tag{3.23}$$

$$\mathbf{P}_{k|k-1} = \frac{1}{2n}\sum_{l=1}^{2n}\left(\mathbf{W}^{*}_{l,k|k-1}\mathbf{W}^{*T}_{l,k|k-1}\right) - \mathbf{w}_{k|k-1}\mathbf{w}^{T}_{k|k-1} \tag{3.24}$$

ซึ่งกำหนดให้ $\mathbf{W}^{*}_{l,k\text{-}1|k\text{-}1}$ คือค่าจุดของ cubature ของแบบจำลองของตัวควบคุม Hybrid CKF-NNPID ซึ่งสามารถหาได้จาก

$$\mathbf{W}^{*}_{l,k|k-1} = f\left(\mathbf{S}_{k-1|k-1}\xi_l + \mathbf{w}_{k-1}\right) \tag{3.25}$$

และค่า ξ มีค่าเท่ากับ

$$\xi_l = \begin{cases} \sqrt{n}\mathbf{I}_l & l = 1,2,...,n \\ -\sqrt{n}\mathbf{I}_{l-n} & l = n+1, n+2,...,2n \end{cases}$$

โดยที่ n คือจำนวนตัวแปรของสถานะระบบ และ I คือคอลัมน์ลำดับที่ l ของเมตริกเอกลักษณ์ ส่วนค่าของ $\mathbf{S}_{l,k\text{-}1|k\text{-}1}$ คือค่ารากที่สองของความแปรปรวนร่วมเกี่ยวซึ่งมีค่าเท่ากับ

$$\mathbf{P}_{k-1|k-1} = \mathbf{S}_{k-1|k-1}\mathbf{S}^{T}_{k-1|k-1} \tag{3.26}$$

2) ขั้นตอนการพิสูจน์ความถูกต้องของค่าพยากรณ์ทั้งสองด้วยค่าคาลมาน ($\mathbf{K}_k$) ซึ่งมีค่าเท่ากับ

$$\mathbf{K}_k = \mathbf{P}_{wy,k|k-1}\left[\mathbf{P}_{yy,k|k-1}\right]^{-1} \tag{3.27}$$

$$\mathbf{P}_{yy,k|k-1} = \frac{1}{2n}\sum_{l=1}^{2n}\left[h(\mathbf{W}_{l,k|k-1})h(\mathbf{W}_{l,k|k-1})^{T}\right] - \mathbf{y}_{k|k-1}\mathbf{y}^{T}_{k|k-1} + \mathbf{R}_k \tag{3.28}$$

$$\mathbf{P}_{xy,k|k-1} = \frac{1}{2n}\sum_{l=1}^{2n}\left[\mathbf{W}_{l,k|k-1}h(\mathbf{W}_{l,k|k-1})^{T}\right] - \mathbf{w}_{k|k-1}\mathbf{y}^{T}_{k|k-1} \tag{3.29}$$

$$\mathbf{R}_k = \mathbf{R}_{k-1} + \frac{1}{k}\left(\left(y_d - y_k\right)\left(y_d - y_k\right)^{T} - \mathbf{R}_{k-1}\right) \tag{3.30}$$

โดยกำหนดให้ y_d คือค่าเป้าหมายที่ต้องการ สำหรับ $\mathbf{P}$ และ $\mathbf{R}$ คือค่าแปรปรวนร่วมเกี่ยวของระบบและค่าผิดพลาดตามลำดับ ในส่วนของค่าพยากรณ์ของน้ำหนัก ณ จุด Cubature และเอาต์พุตมีค่าเท่ากับ

$$\mathbf{W}_{l,k|k-1} = \mathbf{S}_{k|k-1}\xi_l + \mathbf{w}_{k|k-1} \tag{3.31}$$

$$\mathbf{y}_{k|k-1} = \frac{1}{2n}\sum_{l=1}^{2n} h\left(\mathbf{W}_{l,k|k-1}\right) \tag{3.32}$$

เพราะฉะนั้นค่าน้ำหนักของการปรับในแต่ละรอบของตัวควบคุม Hybrid CKF-NNPID มีค่าเท่ากับ

$$\mathbf{w}_{k|k} = \mathbf{w}_{k|k-1} + \eta\mathbf{K}_k\left(y_d - y_k\right) \tag{3.33}$$

กำหนดให้ η คือค่าอัตราการเรียนรู้ของคาลมานซึ่งกำหนดให้มีค่าเท่ากับ 0.01 ตลอดการคำนวณ ในที่นี่กำหนดให้ค่าผิดพลาดอันเนื่องมาจากแบบจำลองของตัวควบคุม Hybrid CKF-NNPID เป็นศูนย์ สำหรับขั้นตอนของอัลกอริทึม Hybrid CKF-NNPID เพื่อการปรับค่าน้ำหนักของตัวควบคุมดังอัลกอริทึมที่ 2

<u>อัลกอริทึม 2</u> อัลกอริทึมการเรียนรู้แบบไฮบริดของตัวกรองคาลมานด้วยกฎ Cubature (Hybrid CKF-NNPID Algorithm)

Initialization: $\mathbf{w}_0$, $\mathbf{P}_0$, $\mathbf{K}_0$, $\mathbf{J}_0$, η = 0.01, $\mathbf{R}_0$
for: each step **do** until $\mathbf{J}(\mathbf{w}_k)$ meet condition
  predict the state: $\mathbf{w}_{k|k-1}$
  predict the covariance: $\mathbf{P}_{k|k-1}$
  while (inner <= 20)

$$\hat{u}_k = \mathbf{w}^3_{k|k-1}\left(\sin\left(\mathbf{w}^2_{k|k-1}\left(\left(\mathbf{w}_{k|k-1} + \Delta\mathbf{w}_{k|k-1}\right)\mathbf{x} + \mathbf{b}^1_{k|k-1}\right) + \mathbf{b}^2_{k|k-1}\right)\right)$$

$$\hat{y}_k = NARX\left(\hat{u}_k\right)$$

  calculate the error: $\mathbf{R}_k$
  evaluate the state-value by Kalman gain using Cubature Kalman Filter (Hybrid CKF): $\mathbf{K}_k$
  evaluate the covariance: $\mathbf{P}_{k|k}$
  calculate the cost function:

$$J(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\left(e_k^i\right)^2$$

  update $\mathbf{w}_{k|k}$, $\mathbf{P}_{k|k}$
end for

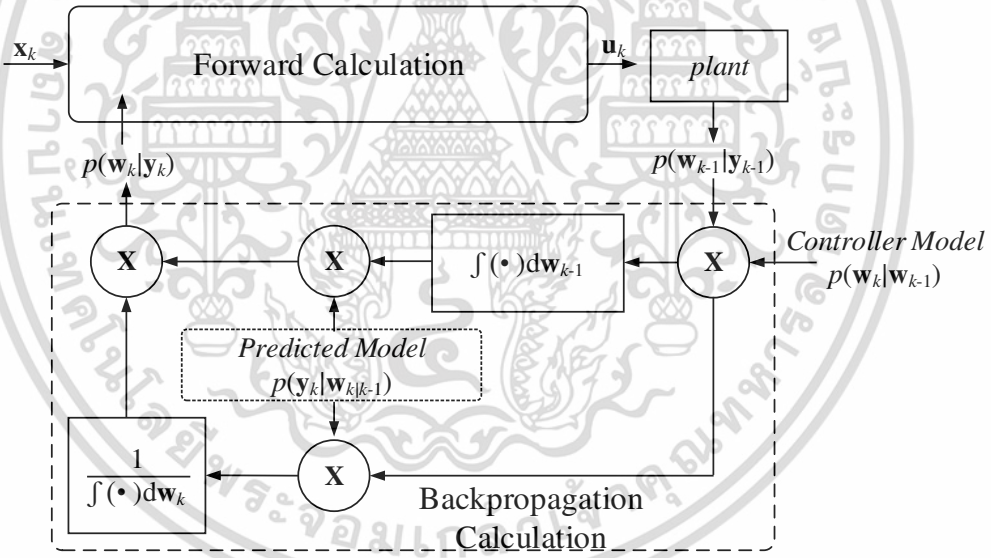
3.2.3 อัลกอริทึมการเรียนรู้แบบตัวกรองเบย์แบบประยุกต์

ในหัวข้อนี้จะกล่าวถึงแนวคิดของอัลกอริทึมการเรียนรู้ด้วยตัวกรองเบย์แบบประยุกต์เพื่อใช้ในตัวควบคุม NNPID หรือเรียกตัวควบคุมชนิดนี้ว่า ABF-NNPID อัลกอริทึมที่นำเสนอนี้จะต้องตั้งสมมุติฐานของพฤติกรรมของการเปลี่ยนแปลงของน้ำหนักของตัวควบคุม NNPID และระบบ (Plant) ที่มีปริภูมิสถานะเชิงสุ่มดังนี้

$$\mathbf{w}_k \approx p(\mathbf{w}_k | \mathbf{w}_{k-1}) \tag{3.34}$$

$$\mathbf{y}_k \approx p(\mathbf{y}_k | \mathbf{w}_k) \tag{3.35}$$

เมื่อกำหนดให้ $p(*)$ คือฟังก์ชันของความน่าจะเป็น และเมื่อใช้กฎของเบย์เพื่อคำนวณค่าสถานะของระบบพลวัตแบบสุ่มข้างต้นซึ่งแสดงถึงการเปลี่ยนแปลงของน้ำหนักของตัวควบคุม ABF-NNPID ดังสมการข้างต้นแล้วสามารถเขียนไดอะแกรมของอัลกอริทึมนี้ดังรูปที่ 3.4



รูปที่ 3.4 อัลกอริทึมการเรียนรู้ของด้วยกฎของเบย์แบบประยุกต์

จากรูปที่ 3.4 เป็นแนวคิดของอัลกอริทึมการเรียนรู้ด้วยตัวกรองเบย์แบบประยุกต์เป็นอัลกอริทึมที่ใช้การพยากรณ์สถานะของระบบด้วยกฎของเบย์ซึ่งมีค่าเท่ากับ

$$p(\mathbf{w}_k | \mathbf{y}_k) = \frac{p(\mathbf{y}_k | \mathbf{w}_k) p(\mathbf{w}_k | \mathbf{y}_{k-1})}{p(\mathbf{y}_k | \mathbf{y}_{k-1})} \tag{3.36}$$

กำหนดให้ $p(\mathbf{w}_k|\mathbf{y}_k)$ คือค่าความน่าจะเป็นของค่าน้ำหนักโครงข่ายของตัวควบคุม ABF-NNPID เมื่อ ซึ่งแต่ละฟังก์ชันของเบย์จะต้องอาศัยการคำนวณแบบวนซ้ำ k รอบเพื่อหาค่าน้ำหนักที่ทำให้ระบบเข้าสู่สภาวะเสถียร ซึ่งจากสมการข้างต้นประกอบด้วย 3 พจน์ พจน์แรกคือ ค่าความน่าจะเป็นของน้ำหนักปัจจุบันที่ขึ้นกับเอาต์พุตที่ผ่านมาล่าสุด จากแนวคิดของงาน [43] สามารถกำหนดให้พจน์ $p(\mathbf{w}_k|\mathbf{y}_{k-1})$ ของสมการนี้มีค่าเท่ากับ

$$p(\mathbf{w}_k|\mathbf{y}_{k-1}) = \int_{\Re} p(\mathbf{w}_k|\mathbf{w}_{k-1})\, p(\mathbf{w}_{k-1}|\mathbf{y}_{k-1}) d\mathbf{w}_{k-1} \tag{3.37}$$

โดยกำหนดให้ $p(\mathbf{w}_1|\mathbf{w}_{k-1})$ คือค่าความน่าจะเป็นของสถานะปัจจุบันที่ขึ้นกับค่าในอดีตล่าสุดเป็น

$$p(\mathbf{w}_k|\mathbf{w}_{k-1}) = \mathbf{w}_{k-1} + \left(-\eta \frac{\partial \mathbf{J}_{k-1}}{\partial \mathbf{w}_{k-1}} \right) \tag{3.38}$$

และกำหนดให้ $p(\mathbf{w}_{k-1}|\mathbf{y}_{k-1})$ คือค่าความน่าจะเป็นของค่าน้ำหนักของตัวควบคุม ABF-NNPID ที่ผ่านมาล่าสุดที่ขึ้นกับเอาต์พุตที่ผ่านมาล่าสุด

พจน์ต่อมาคือค่าความน่าจะเป็นของค่าเอาต์พุตที่ขึ้นกับค่าสถานะปัจจุบัน (ค่าน้ำหนักของตัวควบคุม ABF-NNPID ปัจจุบัน) โดยกำหนดให้เป็นแบบจำลองของตัวระบบ (Plant) ซึ่งมีค่าดังนี้

$$p(y_k|\mathbf{w}_{k|k-1}) = h\left(\mathbf{w}_{k|k-1}, u_k\right) \tag{3.39}$$

และพจน์สุดท้ายคือค่าความน่าจะเป็นของค่าเอาต์พุตที่ขึ้นกับค่าเอาต์พุตที่ผ่านมาล่าสุด มีค่าเป็น

$$p(\mathbf{y}_k|\mathbf{y}_{k-1}) = \int_{\Re} p(\mathbf{y}_k|\mathbf{w}_k)\, p(\mathbf{w}_k|\mathbf{y}_{k-1}) d\mathbf{w}_k \tag{3.40}$$

เพราะฉะนั้นเมื่อแทนค่าลงไปในสมการของการหาค่าน้ำหนักข้างต้นจะได้

$$p(\mathbf{w}_k|\mathbf{y}_k) = \frac{h\left(\mathbf{w}_{k|k-1}\right) \int_{\Re} f\left(\mathbf{w}_k\right) p(\mathbf{w}_{k-1}|\mathbf{y}_{k-1}) d\mathbf{w}_{k-1}}{\int_{\Re} h\left(\mathbf{w}_{k|k-1}\right) p(\mathbf{w}_k|\mathbf{y}_{k-1}) d\mathbf{w}_k} \tag{3.41}$$

เนื่องจากสมการข้างต้นเป็นสมการที่อยู่ในรูปของค่าปริพันธ์ของฟังก์ชันความน่าจะเป็นซึ่งยากต่อการหาคำตอบเป็นอย่างมาก ดังนั้นจึงต้องอาศัยแนวคิดของการแจกแจงแบบเกาส์เซียน (Gaussian

Distribution) มาใช้เพื่อวิเคราะห์ฟังก์ชันที่อยู่ในรูปปริพันธ์ให้สามารถหาคำตอบได้ โดยในขั้นแรกจะกำหนดให้ความสัมพันธ์ของสถานะ (น้ำหนักของตัวควบคุม ABF-NNPID, $\mathbf{w}$) และเอาต์พุตของระบบควบคุม ($\mathbf{y}_k$) เป็นความสัมพันธ์แบบเกาส์เซียนซึ่งมีค่าดังนี้

$$p(\mathbf{w}_k, \mathbf{y}_k) = N\left(\begin{pmatrix}\mathbf{w}_k\\ \mathbf{y}_k\end{pmatrix}; \begin{pmatrix}\mathbf{w}_{k|k}\\ \mathbf{y}_{k|k}\end{pmatrix}, \begin{pmatrix}\mathbf{P}_{ww,k|k} & \mathbf{P}_{wy,k|k}\\ \mathbf{P}_{yw,k|k} & \mathbf{P}_{yy,k|k}\end{pmatrix}\right) \tag{3.42}$$

โดยกำหนดให้ $N(*)$ คือการแจกแจงแบบเกาส์เซียนระหว่างสถานะและเอาต์พุตของระบบที่มีค่ากลางเป็น $(\mathbf{w}_{k|k})$ และค่าความแปรปรวนร่วมเกี่ยวเป็น $(\mathbf{P}_{**k|k})$ เพราะฉะนั้นจากงานวิจัย [42] ทำให้สามารถกำหนดให้ค่าฟังก์ชันของ $p(\mathbf{w}_k|\mathbf{y}_k)$ หรือค่าความน่าจะเป็นของค่าน้ำหนักปัจจุบันที่ขึ้นกับเอาต์พุตปัจจุบันมีค่าเป็น

$$p(\mathbf{w}_k \,|\, \mathbf{y}_k) = N(\mathbf{w}_k; \mathbf{w}_{k|k}, \mathbf{P}_{ww,k|k}) \tag{3.43}$$

และกำหนดให้ $\mathbf{P}_{ww,k|k}$ คือค่าแปรปรวนร่วมเกี่ยวระหว่างค่าสถานะด้วยกันซึ่งมีค่าเท่ากับ

$$\mathbf{P}_{ww,k|k} = \mathbf{P}^{-1}_{ww,k|k-1} - \mathbf{P}_{wy,k|k-1}\mathbf{P}^{-1}_{yy,k|k-1}\mathbf{P}_{yw,k|k-1} \tag{3.44}$$

จากสมการข้างต้นค่าแปรปรวนร่วมเกี่ยวต่าง ๆ สามารถแสดงค่าได้ดังนี้

$$\begin{aligned}\mathbf{P}_{ww,k|k-1} = &\int_{\Re} f(\mathbf{w}_{k-1}) f(\mathbf{w}_{k-1})^T N(\mathbf{w}_{k-1}; \mathbf{w}_{k-1|k-1}, \mathbf{P}_{k-1|k-1}) d\mathbf{w}_{k-1}\\ &-\mathbf{w}_{k|k-1}\mathbf{w}^T_{k|k-1} + \mathbf{Q}_{k-1}\end{aligned} \tag{3.45}$$

$$\mathbf{P}_{wy,k|k-1} = \int_{\Re} \mathbf{w}_k h(\mathbf{w}_k)^T N(\mathbf{w}_k; \mathbf{w}_{k|k-1}, \mathbf{P}_{k|k-1}) d\mathbf{w}_k - \mathbf{w}_{k|k-1}\mathbf{y}^T_{k|k-1} \tag{3.46}$$

$$\begin{aligned}\mathbf{P}_{\mathrm{yy},k|k-1} = &\int_{\Re} h(\mathbf{w}_k) h(\mathbf{w}_k)^T N(\mathbf{w}_k; \mathbf{w}_{k|k-1}, \mathbf{P}_{k|k-1}) d\mathbf{w}_k\\ &-\mathbf{y}_{k|k-1}\mathbf{y}^T_{k|k-1} + \mathbf{R}_k\end{aligned} \tag{3.47}$$

โดยที่ค่าของ $\mathbf{y}_{k|k-1}$ และ $\mathbf{w}_{k|k-1}$ มีค่าดังนี้

$$\mathbf{w}_{k|k-1} = \int_{\Re} f(\mathbf{w}_{k-1}) N(\mathbf{w}_{k-1}; \mathbf{w}_{k-1|k-1}, \mathbf{P}_{ww,k-1|k-1}) d\mathbf{w}_{k-1} \tag{3.48}$$

$$\mathbf{y}_{k|k-1} = \int_{\Re} h(\mathbf{w}_k) N(\mathbf{w}_k; \mathbf{w}_{k|k-1}, \mathbf{P}_{ww,k|k-1}) d\mathbf{w}_k \tag{3.49}$$

จากสมการข้างต้นอยู่ในรูปของค่าปริพันธ์ของฟังก์ชันการแจกแจงแบบเกาส์เซียนซึ่งยากต่อการหาคำตอบเช่นกัน ดังนั้นต้องแปลงให้อยู่ในรูปของปริพันธ์หลายมิติเชิงน้ำหนักแบบทั่วไป (general weighting functions) เสียก่อน ซึ่งสามารถแสดงได้ดังสมการดังนี้

$$\int_{\Re} f(\mathbf{w}_k) N(\mathbf{w}_k; \mathbf{w}_{k|k-1}, \mathbf{P}_{k|k-1}) d\mathbf{w}_k = \frac{1}{\sqrt{\pi^n}} \int_{\Re} f\left(\sqrt{2\mathbf{P}_{k|k-1}}\mathbf{w}_k + \mathbf{w}_{k|k-1}\right) \exp(-\mathbf{w}_k \mathbf{w}_k^T) d\mathbf{w}_k \quad (3.50)$$

ถึงแม้ว่าเมื่อนำสมการที่ในรูปของฟังก์ชันปริพันธ์ของการแจกแจงแบบเกาส์เซียนแปลงไปเป็นสมการที่อยู่ในรูปของปริพันธ์หลายมิติเชิงน้ำหนักแบบทั่วไปแล้วก็ตามก็ยังไม่สามารถหาคำตอบได้ ทำให้ต้องอาศัยแนวคิดของการแปลงค่าให้อยู่ในรูปแบบของ Spherical Radial Cubature Rule [29] มาใช้เพื่อแปลงสมการดังกล่าวให้เป็นฟังก์ชันที่อยู่ในรูปแบบปริพันธ์ที่สามารถหาคำตอบได้ และเพื่อให้ง่ายต่อการคำนวณจึงกำหนดให้ฟังก์ชันของปริพันธ์หลายมิติแบบทั่วไปแทนด้วย $I_N(f)$ มีค่าเป็น

$$I_{\mathrm{N}}(f) = \int_{\Re} f(\mathbf{w}_k) \exp(-\mathbf{w}_k \mathbf{w}_k^T) d\mathbf{w} \quad (3.51)$$

สำหรับขั้นตอนของการแปลงฟังก์ชันปริพันธ์หลายมิติแบบทั่วไปด้วย Spherical Radial Cubature Rule มีหลักการสำคัญคือการเปลี่ยนระบบฟังก์ชันจากรูปแบบของระบบคาทีเซียนไปเป็นระบบรัศมี-แบบทรงกระบอกซึ่งมีขั้นตอนดังนี้

1) กำหนดให้ r แล z คือรัศมีและทิศทางของรัศมีของ r และกำหนดให้ $\mathbf{w} = rz$ และ $\mathbf{w}^T\mathbf{w} = r^2$ ดังนั้นเมื่อแทนค่าไปยังสมการ (3.51) ข้างต้นจะได้

$$I_{\mathrm{N}}(f(\mathbf{w})) = \int_{\Re} \int_{U_n} f(\mathbf{rz}) r^{n-1} \exp(-\mathbf{r}^2) d\sigma(\mathbf{z}) d\mathbf{r} \quad (3.52)$$

โดยที่

$$I_{\mathrm{N}}(f(\mathbf{w})) = \int_{\Re} S(r) r^{n-1} \exp(-r^2) dr \quad (3.53)$$

$$C(r) = \int_{U_n} f(r\mathbf{z}) d\sigma(\mathbf{z}) \quad (3.54)$$

ดังนั้นสมการของปริพันธ์ของฟังก์ชันดังกล่าวมีค่าเท่ากับ

$$C(r) = \int_{U_n} f(r\mathbf{z})\, d\sigma(\mathbf{z}) \approx \omega \sum_{i=1}^{2n} \mathrm{f}\left[u\right]_i \tag{3.55}$$

การแปลงให้อยู่ในรูปของฟังก์ชันปริพันธ์อีกรูปแบบ

$$\int_{\Re} C(r)\, r^{n-1} \exp\left(-r^2\right) dr \approx \frac{1}{2} \int_{\Re} r^{\frac{n}{2}-1} \exp\left(-r^2\right) dr \times C\left(\sqrt{\frac{n}{2}}\right) \tag{3.56}$$

เพราะฉะนั้น สมการของปริพันธ์หลายมิติแบบทั่วไปเชิงน้ำหนักมีค่าเป็น

$$I_{\mathrm{N}}\left(f(\mathbf{w})\right) = \sum_{i=1}^{2n} \omega_i \mathrm{f}\left[\xi_i\right] \tag{3.57}$$

เมื่อกำหนดให้

$$\omega_i = \frac{1}{2n},$$
$$\xi_i = \sqrt{n}\left[\mathbf{I}\right]_i,\ i = 1, 2, 3, \ldots, 2n,$$

สุดท้ายการแปลงให้อยู่ในรูปของปริพันธ์หลายมิติแบบทั่วไปเชิงน้านหักจะอยู่ในรูปของ

$$I_{\mathrm{N}}\left(f(\mathbf{w})\right) = \frac{1}{2n} \sum_{i=1}^{2n} f\left(\sqrt{\mathbf{P}_{k|k-1}}\sqrt{n}\left[\mathbf{I}\right]_i + \mathbf{w}_{k-1|k-1}\right) \tag{3.58}$$

ดังนั้นเมื่อแปลงสมการทั้งหมดตั้งแต่สมการที่ (3.45) ถึง สมการ (3.49) จากอัลกอริทึมของแนวคิดของเบย์แล้วแทนค่าในสมการ (3.41) จะได้เป็น

$$\mathbf{w}_{k|k} = \frac{h\left(\mathbf{w}_{k|k-1}\right)\frac{1}{2n}\sum_{i=1}^{2n} f\left(\sqrt{\mathbf{P}_{ww,k-1|k-1}}\sqrt{n}\left[\mathbf{I}\right]_i + \mathbf{w}_{k-1|k-1}\right)}{\frac{1}{2n}\sum_{i=1}^{2n} h\left(\sqrt{\mathbf{P}_{ww,k|k-1}}\sqrt{n}\left[\mathbf{I}\right]_i + \mathbf{w}_{k|k-1}\right)} \tag{3.59}$$

เมื่อกำหนดให้ค่าตัวแปร ($\mathbf{w}_{k|k-1}$) และ ($\mathbf{P}_{ww,k|k-1}$) มีค่าตามที่ระบุไว้ในส่วนของอัลกอริทึมตามกฎ Spherical Radial Cubature Rule ที่ได้กล่าวไว้ในหัวข้อที่ผ่านมา

ทั้งนี้ค่าผิดพลาดอันเนื่องมาจากแบบจำลอง ($\mathbf{Q}$) ของตัวควบคุม ABF-NNPID มีค่าเป็นศูนย์ และเพื่อความสะดวกในการใช้งานของอัลกอริทึมเพื่อการปรับค่าน้ำหนักของตัวควบคุม ABF-NNPID จึงมีขั้นตอนดังต่อไปนี้

<u>อัลกอริทึม 3</u> ลกอริทึมการเรียนรู้ด้วยตัวกรองเบย์แบบประยุกต์ (Applied Bayesian Filter Algorithm)

Initialization: $\mathbf{w}_0$, $\mathbf{P}_0$, $\mathbf{K}_0$, $\mathbf{J}_0$, $\eta = 0.01$, $\mathbf{R}_0$

for: each step **do** until $\boldsymbol{J}(\mathbf{w}_k)$ meet condition

predict the state: $\mathbf{w}_{k|k-1}$

predict the covariance: $\mathbf{P}_{k|k-1}$

Evaluate the output of the plant: $h(\mathbf{w}_{k|k-1})$

calculate the state: $\mathbf{w}_{k|k}$

$$\mathbf{w}_{k|k} = \frac{h\left(\mathbf{w}_{k|k-1}\right)\frac{1}{2n}\sum_{i=1}^{2n} f\left(\sqrt{\mathbf{P}_{ww,k-1|k-1}}\sqrt{n}\left[\mathbf{I}\right]_i + \mathbf{w}_{k-1|k-1}\right)}{\frac{1}{2n}\sum_{i=1}^{2n} h\left(\sqrt{\mathbf{P}_{ww.k|k-1}}\sqrt{n}\left[\mathbf{I}\right]_i + \mathbf{w}_{k|k-1}\right)}$$

calculate the error: $\mathbf{R}_k$

calculate the covariance: $\mathbf{P}_{k|k}$

calculate the cost function:

$$J(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\left(e_k^i\right)^2$$

update $\mathbf{w}_{k|k}$, $\mathbf{P}_{k|k}$

end for

3.2.4 อัลกอริทึมการเรียนรู้ปฏิบัติ-วิจารณ์แบบเสริมกำลัง (Actor-critic Reinforcement Algorithm)

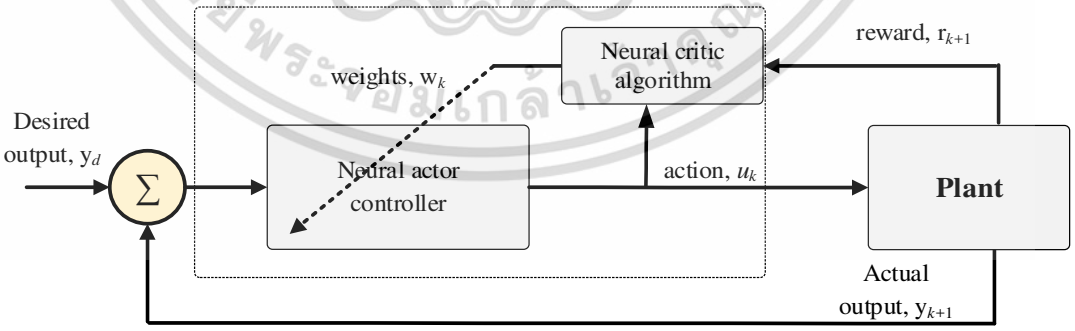
ถึงแม้ว่าตัวควบคุม NNPID ที่ใช้อัลกอริทึมที่นำเสนอในหัวข้อที่ผ่านทั้งหมดได้แก่ อัลกอริทึมการเรียนรู้แบบไฮบริดตัวกรองคาลมาน หรือจะเป็นอัลกอริทึมการเรียนรู้แบบตัวกรองผสมของเบย์แบบประยุกต์จะสามารถเพิ่มประสิทธิภาพและขจัดปัญหาดังที่กล่าวมาข้างต้น เช่น เรื่องการตั้งค่าเริ่มต้นของน้ำหนัก ($\mathbf{w}_0$) เรื่องหลักการออกแบบของอัลกอริทึมที่ต้องใช้สมการอนุพันธ์ทำให้เกิดการสูญเสีย

ข้อมูลไปบางส่วน และค่าภาวะชั่วครู่ (Transient) ของผลตอบสนองของระบบ เป็นต้น แต่อย่างไรก็ตาม การพัฒนาอัลกอริทึมการเรียนรู้ก็ยังต้องดำเนินต่อไปเพื่อขจัดข้อบกพร่องต่าง ๆ หรือแม้กระทั่งการพัฒนาอัลกอริทึมเพื่อเพิ่มประสิทธิภาพของงานวิจัยให้ดียิ่งขึ้นไป วิทยานิพนธ์ฉบับนี้จึงนำเสนอตัวควบคุมที่ใช้อัลกอริทึมโดยใช้หลักการของการทดสอบค่าการกระทำ (Control action, u) เพื่อหาสถานะของระบบ (State, $\mathbf{w}$) ที่ดีที่สุดในบรรดาการกระทำที่เป็นไปได้ทั้งหมดโดยใช้ค่าของ Action-Value Approximation เพื่อวิจารณ์ค่าของการกระทำจากสถานะที่เกิดขึ้นของระบบ เราเรียกตัวควบคุม NNPID ที่ใช้อัลกอริทึมการเรียนรู้แบบผสมระหว่างอัลกอริทึมปฏิบัติ-วิจารณ์แบบเสริมกำลังและอัลกอริทึมตัวกรองคาลมานดีกรีสูง (Hybrid of the Actor-critic Reinforcement and High Order Square-root Cubature Kalman Filter) ว่า “ตัวควบคุม NNPID-AC” โดยในขั้นแรกจะต้องตั้งสมมุติฐานของพฤติกรรมของการเปลี่ยนค่าสถานะ (State, $\mathbf{w}$) ของตัวควบคุมและการกระทำ (Control action, u) เป็นการเปลี่ยนแปลงของพฤติกรรมแบบสุ่ม (Stochastic behavior) เพราะฉะนั้นสมการของการเปลี่ยนแปลงสถานะของตัวควบคุมดังกล่าวสามารถกำหนดเป็นดังนี้

$$\mathbf{w}_k \approx p(\mathbf{w}_k \mid \mathbf{w}_{k-1}) \tag{3.60}$$

$$u_k = f(\mathbf{w}_k) \tag{3.61}$$

กำหนดให้ $p(\mathbf{w}_k|\mathbf{w}_{k\text{-}1})$ เป็นฟังก์ชันของความน่าจะเป็นของค่าการปรับสถานะ (State, $\mathbf{w}$) ในปัจจุบันของตัวควบคุม NNPID เมื่อกำหนดค่าสถานะในอดีตล่าสุด ส่วนตัวแปร u คือการกระทำ (Control action, u) และสุดท้าย $f(*)$ คือฟังก์ชันของการพิจารณาสถานะเพื่อกำหนดกระทำ หรือเรียกว่าฟังก์ชัน Policy สามารถอธิบายตัวควบคุม NNPID-AC ด้วยไดอะแกรมได้ดังรูปที่ 3.5



รูปที่ 3.5 ระบบควบคุมที่อยู่ภายใต้อัลกอริทึมการเรียนรู้ปฏิบัติ-วิจารณ์แบบเสริมกำลัง (Actor-critic Reinforcement Algorithm)

อัลกอริทึมการเรียนรู้ที่นำเสนอนี้ (อัลกอริทึมของตัวควบคุม NNPID-AC) ประกอบด้วย 2 ส่วน คือส่วนของการกระทำ (Actor) และส่วนของการวิจารณ์ (Critic) ในที่นี่จะขอกล่าวถึงการกระทำ

(Actor) โดยในขั้นแรกจะต้องกำหนดให้ตัวควบคุม NNPID ที่มีหน้าผลิตตัวอินพุตควบคุม (Control Input) ตัวกระทำ (Actor, u) ซึ่งมีตัวแปรที่สำคัญในการสร้างตัวกระทำคือ สถานะของตัวควบคุมที่แทนด้วยน้ำหนักของโครงข่ายประสาทเทียม (State, $\mathbf{w}$) และเพื่อให้ได้สถานะของระบบที่เหมาะสมจะต้องทดสอบการกระทำทั้งหมดที่เป็นไปได้ด้วยค่า Action-Value Function ซึ่งจะกล่าวต่อไปในส่วนของขั้นตอนการวิจารณ์ (Critic) ต่อการกระทำดังกล่าวต่อไป

ต่อมาส่วนของการประเมินค่าจากฟังก์ชันคุณค่าของการกระทำ (Action-Value Function, $V(\mathbf{w},u)$) ซึ่งมีค่าเป็น

$$V_{k+1}(\mathbf{w},u)=\frac{1}{2}\left(y_d-y_k\right)^2+E\left[V_k\left(\hat{\mathbf{w}}\right)\right] \tag{3.62}$$

โดยกำหนดให้ $\mathrm{E}[V(\hat{\mathbf{w}},u)]$ เป็นฟังก์ชันค่าต่ำสุดของคุณค่าของการกระทำ (Action-Value Function อันเนื่องมาจากค่าน้ำหนักพยากรณ์ ($\hat{\mathbf{w}}$) ด้วยข้อเสียของการใช้อัลกอริทึมนี้ที่จะต้องประเมินค่สถานะจากการกระทำทั้งหมดที่เป็นไปได้ซึ่งจำเป็นต้องใช้เวลามาก โดยการคำนวณแบบวนซ้ำหรือคำนวณแบบลูป จึงทำให้ต้องหาวิธีการที่ช่วยทำนายเพื่อให้ได้ค่าที่เหมาะสมแทน ซึ่งในงานวิทยานิพนธ์นี้ได้นำเสนออัลกอริทึมตัวกรองคาลมานแบบ Square Root Cubature Kalman (SCKF) ซึ่งมีข้อดีมากมายเช่น เพื่อหลีกเลี่ยงเมตริกที่หาค่าอินเวอร์สไม่ได้ (Matrix Singularity) เพื่อให้ได้ค่าพยากรณ์ที่ใกล้เคียงกับค่าจริงมากที่สุด เป็นต้น และงานวิจัยนี้ก็ได้นำการพยากรณ์ที่มีความถูกต้องเพิ่มมากขึ้นยิ่งขึ้นไปอีกด้วยการใช้อัลกอริทึมของ Square Root Cubature Kalman (SCKF) ที่มีอันดับ 5 เพื่อให้ได้ค่าพยากรณ์ที่ดีที่สุด โดยรายละเอียดของการใช้งานการพยากรณ์ Actor มีขั้นตอนการคำนวณดังนี้

1) ขั้นตอนการพยากรณ์น้ำหนัก ($\mathbf{w}_{k|k\text{-}1}$) และ ค่าแปรปรวนร่วมเกี่ยว ($\mathbf{P}_{k|k\text{-}1}$)

$$\mathbf{w}_{k|k-1}=\frac{1}{2n}\sum_{l=1}^{2n}\mathbf{W}^{*}_{l,k|k-1} \tag{3.63}$$

ซึ่งกำหนดให้ $\mathbf{W}^{*}_{l,k\text{-}1|k\text{-}1}$ คือค่าจุดของ cubature จากแบบจำลองของตัวควบคุม NNPID-AC ซึ่งสามารถหาได้จาก

$$\mathbf{W}^{*}_{l,k|k-1}=f\left(\mathbf{S}_{k-1|k-1}\xi_l+\mathbf{w}_{k-1}\right) \tag{3.64}$$

ซึ่งค่า ξ มีค่าเท่ากับ

$$\xi_l=\begin{cases}\sqrt{n}\mathbf{I}_l & l=1,2,...,n\\ -\sqrt{n}\mathbf{I}_{l-n} & l=n+1,n+2,...,2n\end{cases} \tag{3.65}$$

โดยที่ n คือจำนวนตัวแปรของสถานะระบบ และ $\mathbf{I}$ คือคอลัมลำดับที่ l ของเมตริกเอกลักษณ์ และค่าความแปรปรวนร่วมเกี่ยว ($\mathbf{S}_{k|k-1}$) มีค่าเท่ากับ

$$\mathbf{S}_{k|k-1} = QR\left(\frac{1}{\sqrt{2n}}\left[f\left(Y_{1,k|k-1}\right)-\mathbf{w}_{k|k-1} \quad f\left(\mathbf{\Psi}_{2,k|k-1}\right)-\mathbf{w}_{k|k-1} \quad \cdots \quad f\left(\mathbf{\Psi}_{2n,k|k-1}\right)-\mathbf{w}_{k|k-1}\right]\right) \tag{3.66}$$

กำหนดให้ $QR(*)$ คือฟังก์ชัน QR decomposition โดยที่เมตริก $\mathbf{S}_{k|k-1}$ เป็นค่าเมตริกสามเหลี่ยมล่าง (Lower Triangular Matrix) ของฟังก์ชัน QR decomposition และในส่วนของค่าพยากรณ์ของจุด Cubature ค่า ($\mathbf{\Psi}_{k|k-1}$) และค่าพยากรณ์ของสถานะ ($\mathbf{w}_{k|k-1}$) ยังคงใช้การคำนวณตามขั้นตอนของอัลกอริทึม Cubature Kalman filter (CKF) ในข้างต้น เมื่อกำหนดให้ค่าพยากรณ์ของจุด Cubature ($\mathbf{\Psi}_{k|k-1}$) มีค่าเป็น

$$\mathbf{\Psi}_{i,k|k-1} = \mathbf{S}_{k|k-1}\varsigma_i + \mathbf{w}_{k|k-1} \tag{3.67}$$

2) ขั้นตอนการพิสูจน์ความถูกต้องของ*ค่าพยากรณ์*ด้วยค่าคงที่คาลมาน ($\mathbf{K}_k$) ซึ่งมีค่าเท่ากับ

$$\mathbf{K}_k = \frac{\left(W_{k|k-1}\left(Y_{k|k-1}\right)^T \Big/ S_{\mathrm{yy},k|k-1}\right)}{S_{yy,k|k-1}} \tag{3.68}$$

$$\mathbf{S}_{k|k} = QR\left(\frac{1}{\sqrt{2n}}\left[f\left(W_{1,k|k-1}\right)-\mathbf{w}_{k|k} \quad f\left(W_{2,k|k-1}\right)-\mathbf{w}_{k|k-1} \quad \cdots \quad f\left(W_{2n,k|k-1}\right)-\mathbf{w}_{k|k-1}\right]\right) \tag{3.69}$$

โดยกำหนดให้ค่ารากของเมตริกความแปรปรวนร่วมเกี่ยวของเอาต์พุต ($\mathbf{S}_{yy,k|k-1}$) มีค่าเป็น

$$\mathbf{S}_{yy,k|k-1} = QR\left(\frac{1}{\sqrt{2n}}\left[h\left(W_{1,k|k-1}\right)-y_{k|k-1} \quad \cdots \quad \cdots \quad h\left(W_{2n,k|k-1}\right)-y_{k|k-1} \quad \sqrt{R_k}\right]\right) \tag{3.70}$$

ฟังก์ชัน $\mathbf{h}(*)$ คือฟังก์ชันของแบบจำลองของระบบ (Plant) ในส่วนของ $y_{k|k-1}$ และ $W_{k|k-1}$ จะต้องคำนวณด้วยค่าพยากรณ์ของรากของค่าความแปรปรวนร่วมเกี่ยว ($\mathbf{S}_{k|k-1}$) ใหม่จากการใช้อัลกอริทึม

$$W_{l,k|k-1} = S_{k|k-1}\varsigma_l + \mathbf{w}_{k|k-1} \tag{3.71}$$

$$y_{k|k-1} = \frac{1}{2n}\sum_{l=1}^{2n} h\left(W_{l,k|k-1}\right) \tag{3.72}$$

$$\mathbf{R}_k = \mathbf{R}_{k-1} + \frac{1}{k}\left(\left(\mathbf{y}_d - y_k\right)\left(\mathbf{y}_d - y_k\right)^T - \mathbf{R}_{k-1}\right) \tag{3.73}$$

โดยกำหนดให้ y_d คือค่าเป้าหมายที่ต้องการ $y_{k|k-1}$ และ $W_{k|k-1}$ จะต้องคำนวณด้วยค่าพยากรณ์ของราก ของค่าความแปรปรวนร่วมเกี่ยว ($\mathbf{S}_{k|k-1}$) ในส่วนของค่าพยากรณ์ของน้ำหนักและเอาต์พุตมีค่าเท่ากับ

$$W_{l,k|k-1} = S_{k|k-1}\varsigma_l + \mathbf{w}_{k|k-1} \tag{3.74}$$

$$\mathbf{y}_{k|k-1} = \frac{1}{2n}\sum_{l=1}^{2n} h\left(\mathbf{w}_{l,k|k-1}\right) \tag{3.75}$$

ในส่วนของ $\mathbf{R}$ คือค่าแปรปรวนร่วมเกี่ยวของระบบ

$$\mathbf{R}_k = \mathbf{R}_{k-1} + \frac{1}{k}\left(\left(\mathbf{y}_d - y_k\right)\left(\mathbf{y}_d - y_k\right)^T - \mathbf{R}_{k-1}\right) \tag{3.76}$$

เพราะฉะนั้นน้ำค่าหนักของหนักของการปรับในแต่ละรอบของระบบควบคุมมีค่าเท่ากับ

$$\mathbf{w}_{k|k} = \mathbf{w}_{k|k-1} + \eta\mathbf{K}_k\left(y_d - y_{k|k-1}\right) \tag{3.77}$$

กำหนดให้ η คือค่าอัตราการเรียนรู้ของคาลมานซึ่งกำหนดให้มีค่าเท่ากับ 0.01 ตลอดการคำนวณ ในที่นี่กำหนดให้ค่าผิดพลาดอันเนื่องมาจากแบบจำลองของตัวควบคุม NNPID-AC เป็นศูนย์

เพื่อเพิ่มความถูกต้องให้กับอัลกอริทึม งานวิจัยนี้จึงได้ใช้อัลกอริทึมของ Square Root Cubature Kalman (SCKF) ที่มีอันดับ 5 เพื่อเปรียบเทียบประสิทธภาพซึ่งมีขั้นตอนที่แตกต่างจากวิธีการในขั้นต้นคือการคำนวณส่วนของจุด Cubature ให้ใช้สมการจากเดิมที่ใช้สมการการแปลงให้อยู่ในรูปของปริพันธ์หลายมิติแบบทั่วไปมีค่าเป็น

$$I_{\mathrm{N}}\left(f\left(\mathbf{w}\right)\right) = \frac{1}{2n}\sum_{i=1}^{2n} f\left(\sqrt{\mathbf{\Sigma}_{k|k-1}}\sqrt{n}\left[\mathbf{I}\right]_i + \mathbf{w}_{k-1|k-1}\right) \tag{3.78}$$

ให้เปลี่ยนเป็นสมการดังนี้

$$\begin{aligned} I_{N,5}(h) &\approx \sum_{i=1}^{n_r}\sum_{j=1}^{n_s}\omega_{r,i}\omega_{s,j}h\left[r_i\mathbf{s}_j\right] \\ &= \frac{2}{n+2}h\left(\mathbf{w}_{k|k-1}\right) + \frac{1}{\left(n+2\right)^2}\sum_{j=1}^{n(n-1)/2} h\left(\sqrt{\mathbf{\Sigma}_{k|k-1}}\sqrt{n+2}\left[\mathbf{I}_{N5}^{+}\right]_j + \hat{\mathbf{w}}_{k|k-1}\right) + \frac{1}{\left(n+2\right)^2}\sum_{j=1}^{n(n-1)/2} h\left(-\sqrt{\mathbf{\Sigma}_{k|k-1}}\sqrt{n+2}\left[\mathbf{I}_{N5}^{+}\right]_j + \hat{\mathbf{w}}_{k|k-1}\right) \\ &+ \frac{4-n}{2\left(n+2\right)^2}\sum_{j=1}^{n(n-1)/2} h\left(\sqrt{\mathbf{\Sigma}_{k|k-1}}\sqrt{n+2}\left[\mathbf{I}_{N5}^{-}\right]_j + \hat{\mathbf{w}}_{k|k-1}\right) + \frac{4-n}{2\left(n+2\right)^2}\sum_{j=1}^{n(n-1)/2} h\left(-\sqrt{\mathbf{\Sigma}_{k|k-1}}\sqrt{n+2}\left[\mathbf{I}_{N5}^{-}\right]_j + \hat{\mathbf{w}}_{k|k-1}\right) \end{aligned} \tag{3.79}$$

เมื่อกำหนดให้

$$\left[\mathbf{I}^{+}_{N5}\right]_j = \sqrt{0.5}\left(\mathbf{I}_{ni} + \mathbf{I}_{nj}\right); ni < nj\,,$$
$$\left[\mathbf{I}^{-}_{N5}\right]_j = \sqrt{0.5}\left(\mathbf{I}_{ni} - \mathbf{I}_{nj}\right); ni < nj\,,\ ni,\ nj = 1,\ 2,\ 3,\ \ldots$$

สำหรับขั้นตอนของอัลกอริทึมที่ใช้ในตัวควบคุม NNPID-AC เพื่อการปรับค่าน้ำหนักดังอัลกอริทึมที่ 4

<u>อัลกอริทึม 4</u> อัลกอริทึมการเรียนรู้แบบผสมระหว่างอัลกอริทึมปฏิบัติ-วิจารณ์แบบเสริมกำลังและอัลกอริทึมตัวกรองคาลมานดีกรีสูง

Initialization: $\mathbf{w}_0$, $\mathbf{P}_0$, $\mathbf{K}_0$, $\mathbf{J}_0$, η = 0.01, ε = 0.001, $\mathbf{R}_0$
for: each step **do** until $\mathbf{J}(\mathbf{w}_k)$ meet condition
  predict state: $\mathbf{w}_{k|k-1}$
  predict root of the covariance: $\mathbf{S}_{k|k-1}$
  while (abs($\hat{u}_k$ − u_k) <= ε)
  calculate the output:

$$u_k = \mathbf{w}^3_{k|k-1}\left(\sin\left(\mathbf{w}^2_{k|k-1}\left(\left(\mathbf{w}_{k|k-1} + \Delta\mathbf{w}_{k|k-1}\right)\mathbf{x} + \mathbf{b}^1_{k|k-1}\right) + \mathbf{b}^2_{k|k-1}\right)\right)$$
$$\hat{u}_{\boldsymbol{k}} = NARX\left(y\right)$$

  calculate error: $\mathbf{R}_k$
  evaluate state-value by Kalman gain using Square Root Cubature Kalman Filter (Hybrid SCKF): $\mathbf{K}_k$
  evaluate root of the covariance: $\mathbf{S}_{k|k}$
  calculate cost function:

$$V_{k+1}(\mathbf{w},u) = \frac{1}{2}\left(y_d - y_k\right)^2 + E\left[V_k\left(\hat{\mathbf{w}}\right)\right]$$

  update $\mathbf{w}_{k|k}$, $\mathbf{P}_{k|k}$
end for

3.3 การพิสูจน์ความมีเสถียรภาพของระบบควบคุม

เพื่อพิสูจน์ความมีเสถียรภาพของระบบควบคุม ในงานวิทยานิพนธ์ฉบับนี้จึงได้นำทฤษฎี Lyapunov มาใช้เพื่อวิเคราะห์เพื่อหาคุณสมบัติของระบบที่แสดงถึงความสมดุล ซึ่งในที่นี่ตัวแปรสำคัญที่ส่งผลให้ตัวควบคุมสามารถขับเคลื่อนระบบไปสู่จุดเสถียรคือค่าน้ำหนักของตัวควบคุม NNPID โดยค่าน้ำหนัก ($\mathbf{w}_{k|k}$) ที่ปรับนั้นจะต้องสอดคล้องกับค่าผิดพลาด ($\mathbf{e}_k$) ของระบบ จากทฤษฎีของ

Lyapunov สำหรับระบบควบคุมที่ไม่เป็นเชิงเส้นระบบจะเข้าสู่ภาวะสมดุลได้จะต้องทำให้ค่าฟังก์ชันของสมการ Lyapunov (แทนด้วยตัวแปร $\mathbf{V}_k$) เป็นไปตามสมการข้างล่างนี้

$$\Delta \mathbf{V}_k \leq 0 \tag{3.80}$$

กำหนดให้ค่า $\Delta \mathbf{V}_k$ เป็นค่าเปลี่ยนแปลงของสมการ Lyapunov ซึ่งถ้ากำหนดให้ค่าฟังก์ชันของ Lyapunov $\mathbf{V}_k$ คือผิดพลาด ($\mathbf{e}_k$) ของระบบดังนั้นการเปลี่ยนแปลงสามารถหาได้ดังนี้

$$\begin{aligned}
\Delta \mathbf{V}_k &= \mathbf{V}_{k+1} - \mathbf{V}_k \\
&= \frac{1}{2}\sum_j^m e_{j,k+1}^2 - \frac{1}{2}\sum_j^m e_{j,k}^2 \\
&= \frac{1}{2}\sum_j^m \left[\left(e_{j,k+1} + e_{j,k} \right) \left(e_{j,k+1} - e_{j,k} \right) \right] \\
&= \frac{1}{2}\sum_j^m \left[\left(e_{j,k+1} - e_{j,k} + 2e_{j,k} \right) \left(\Delta e_{j,k} \right) \right] \\
&= \frac{1}{2}\sum_j^m \left[\left(\Delta e_{j,k} + 2e_{j,k} \right) \left(\Delta e_{j,k} \right) \right]
\end{aligned} \tag{3.81}$$

โดยที่ค่าของตัวแปรผิดพลาด ($\mathbf{e}_k$) ของระบบหาได้จาก

$$\begin{aligned}
\Delta e_k &= \frac{\partial E_k}{\partial k} = \frac{\partial E_k}{\partial \mathbf{w}_k} \frac{\partial \mathbf{w}_k}{\partial k} \\
&= \frac{\partial \left(y_{d,k} - y_k \right)}{\partial \mathbf{w}_k} \Delta \mathbf{w}_k \\
&= -\frac{\partial \left(y_k \right)}{\partial \mathbf{w}_k} \Delta \mathbf{w}_k
\end{aligned} \tag{3.82}$$

เมื่อนำสมการข้างต้นแทนค่าในสมการของฟังก์ชัน Lyapunov และจะต้องทำให้ค่าการเปลี่ยนแปลงของสมการ Lyapunov ที่ได้จะต้องน้อยกว่าหรือเท่ากับศูนย์จึงจะทำให้ระบบเข้าสู่ภาวะเสถียรได้

$$\Delta \mathbf{V}_k = \sum_{j=1}^m \left[\left(\frac{\partial e_{j,k}}{\partial \mathbf{w}_k} \Delta \mathbf{w}_k e_{j,k} \right) + \frac{1}{2} \left(\frac{\partial e_{j,k}}{\partial \mathbf{w}_k} \Delta \mathbf{w}_k \right)^2 \right] \leq 0 \tag{3.83}$$

เมื่อกำหนดให้ค่า Δw คือค่าการปรับค่าน้ำหนักของตัวควบคุม NNPID โดยที่ในแต่ละอัลกอริทึมการเรียนรู้จะกำหนดสมการที่แตกต่างกันไปตัว ซึ่งโดยทั่วไปแล้วจะกำหนดให้เป็น

$$\Delta \mathbf{w}_k = -\eta \frac{\partial \mathbf{E}_k}{\partial \mathbf{w}_k} \tag{3.84}$$

โดยกำหนดให้ คืออัตราการเรียนรู้ของโครงข่ายประสาทเทียม และเมื่อนำค่าการปรับค่าน้ำหนักแทนค่าลงในสมการ Lyapunov จะได้

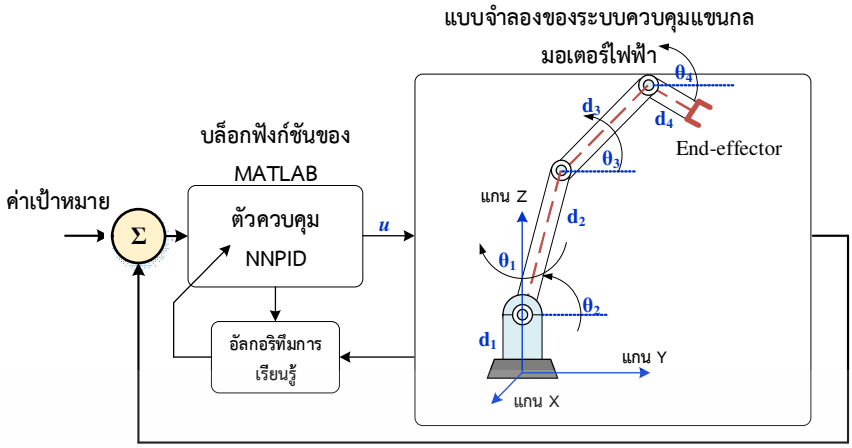
$$\sum_{j=1}^{m}\left[\left(\frac{\partial e_{j,k}}{\partial \mathbf{w}_k}\left(-\eta \frac{\partial \mathbf{E}_k}{\partial \mathbf{w}_k}\right)e_{j,k}\right)+\frac{1}{2}\left(\frac{\partial e_{j,k}}{\partial \mathbf{w}_k}\left(-\eta \frac{\partial \mathbf{E}_k}{\partial \mathbf{w}_k}\right)\right)^2\right] \le 0 \tag{3.85}$$

ดังนั้นในแต่ละรอบของการปรับค่าน้ำหนักของตัวควบคุม NNPID จะต้องกำหนดให้ค่าอัตราการเรียนรู้ของโครงข่ายเป็นไปตามเงื่อนไขดังนี้

$$\eta \le \sum_{j=1}^{m}\left[\frac{2e_{j,k}}{\left(\frac{\partial e_{j,k}}{\partial \mathbf{w}_k}\left(\frac{\partial \mathbf{E}_k}{\partial \mathbf{w}_k}\right)\right)}\right] \tag{3.86}$$

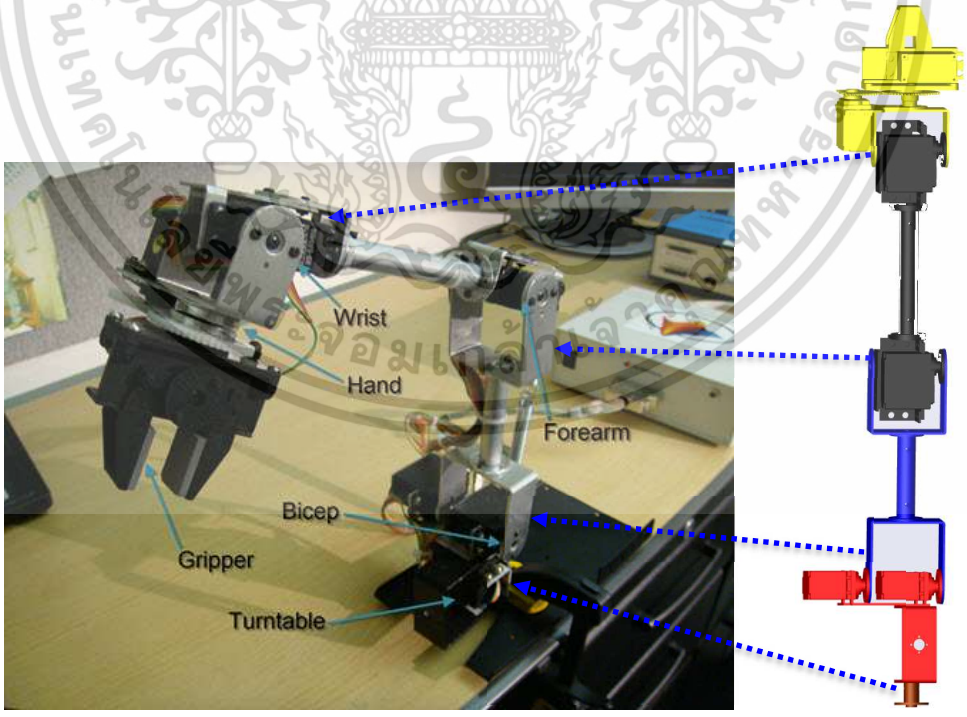
3.4 การสร้างแบบจำลองของระบบควบคุมเพื่อทดสอบประสิทธิภาพของตัวควบคุม NNPID ที่นำเสนอ

เพื่อการทดสอบประสิทธิภาพของตัวควบคุม NNPID จึงได้นำตัวควบคุมไปใช้งานในงานควบคุมระบบแขนกลมอเตอร์ไฟฟ้า โดยในงานวิจัยนี้ได้สร้างระบบควบคุมแบบปิดที่ประกอบด้วยส่วนแบบจำลองของแขนกลมอเตอร์ไฟฟ้าตัว และแบบจำลองตัวควบคุมดังรูปที่ 3.6



รูปที่ 3.6 ระบบควบคุมแขนกลมอเตอร์ไฟฟ้าแบบปิด

ในหัวข้อนี้จะกล่าวถึงวิธีการกำหนดค่าพารามิเตอร์ของแบบจำลองแขนกลมอเตอร์ไฟฟ้าด้วยโปรแกรม MATLAB/SIMULINK ที่ประกอบด้วย 2 ส่วนคือ ส่วนของแบบจำลองแขนกลไฟฟ้า และส่วนของตัวควบคุม NNPID ซึ่งในส่วนของแขนกลไฟฟ้าที่ได้นำมาใช้ในงานวิจัยนี้ได้พัฒนาจากไลบารี่ของโปรแกรม MATLAB ที่มีชื่อว่า Robust Control ซึ่งเป็นหนึ่งใน Toolbox ของ SIMULINK ซึ่งแขนแกนมอเตอร์ไฟฟ้านี้ได้ถูกออกแบบตามแขนกลไฟฟ้าดังรูปที่ 3.7 โดยแขนกลไฟฟ้าประกอบ 4 ข้อต่อ ได้แก่ ข้อที่ 1 ส่วนของฐาน (Turntable) ข้อที่ 2 ส่วนของข้อต่อหัวไหล่ (Bicep) ข้อที่ 3 ส่วนข้อต่อแขน (Forearm) และข้อที่ 4 ส่วนข้อของข้อมือ (Wrist)



(a) (b)

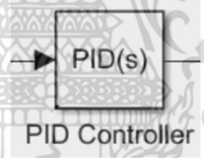
รูปที่ 3.7 (a) แขนกลมอเตอร์ไฟฟ้า (b) แบบจำลองแขนกลมอเตอร์ไฟฟ้าของโปแกรม MATLAB

เมื่อกำหนดให้ค่ามุมของข้อต่อต่าง ๆ ของแขนกลมอเตอร์ไฟฟ้าดังนี้

- θ_1 คือมุมของฐาน (Turntable) และสามารถหมุนตามแนวนอนของตัวฐานหมุนได้ 180 องศา ในแนวแกน x และแกน y
- θ_2 คือมุมของข้อต่อระหว่างข้อหัวไหล่ (Bicep) และส่วนของฐาน (Turntable) โดยสามารถหมุนได้ 180 องศาในแนวแกน z และแกน y/แกน x
- θ_3 คือมุมของข้อต่อระหว่างข้อแขน (Forearm) และข้อหัวไหล่ (Bicep) โดยสามารถหมุนได้ 180 องศาในแนวแกน z และแกน y/แกน x
- θ_4 คือมุมของข้อต่อระหว่างส่วนของข้อมือ (Wrist) และข้อแขน (Forearm) โดยสามารถหมุนได้ 180 องศาในแนวแกน z และแกน y/แกน x

ถึงแม้ว่าแขนกลมอเตอร์ไฟฟ้าที่ใช้ในงานวิจัยมาจากไลบารี่ของ MATLAB แต่เพื่อให้สอดคล้องกับการประยุกต์ใช้งานจริงจึงมีการตั้งค่าพารามิเตอร์ของมอเตอร์ไฟฟ้าตามค่าจริงซึ่งสามารถเข้าไปตั้งค่าในโปรแกรม SIMULINK ดังนี้

- ตั้งค่าส่วนพารามิเตอร์ทางไฟฟ้าประกอบด้วยแรงดันแหล่งจ่ายไฟฟ้า ความเร็วรอบมอเตอร์สูงสุด แรงบิด กระแสขดลวดอาเมเจอร์ และกระแสไฟฟ้าเมื่อไม่มีโหลด
- ตั้งค่าส่วนพารามิเตอร์ทางกลประกอบด้วยความยาวของข้อต่อ น้ำหนักข้อต่อ



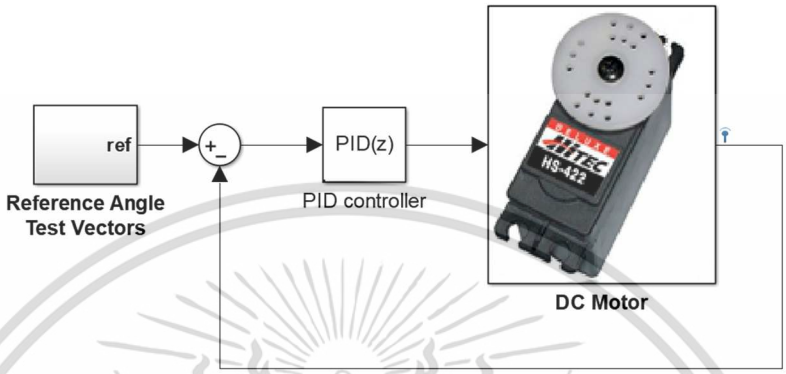
รูปที่ 3.8 แสดงภาพของ PID Simulink Block Function

หลังจากที่ได้กล่าวถึงส่วนประกอบแรกเป็นที่เรียบร้อยแล้ว ในส่วนนี้จะกล่าวถึงการสร้างแบบจำลองของตัวควบคุม ได้แก่ ตัวควบคุม PID และตัวควบคุม NNPID โดยที่ตัวควบคุม PID จะถูกสร้างด้วยฟังก์ชันของ MATLAB/SIMULINK ตามไลบารี่ของโปรแกรม และในส่วนของการกำหนดค่า K_D K_I และ K_D สามารถกำหนดโดยการใช้เครื่องมือ (Tool box) ของ MATLAB/SIMULINK ในโปแกรม โดยจะอธิบายการสร้างไว้เป็น 2 ส่วนซึ่งมีรายละเอียดดังนี้

3.4.1 การสร้างแบบจำลองของตัวควบคุม PID

การสร้างตัวควบคุม PID นี้เป็นการนำมาใช้งานเพื่อเปรียบเทียบประสิทธิภาพกับตัวควบคุมที่ได้นำเสนอได้แก่ ตัวควบคุม ENNPID ตัวควบคุม Hybrid CKF-NNPID ตัวควบคุม ABF-NNPID และตัวควบคุม NNPID-AC ซึ่งในการสร้างตัวควบคุม PID จะใช้เครื่องมือ (Tool box) ของโปรแกรม MATLAB/SIMULINK ดังรูปที่ 3.8 และใช้การปรับค่าคงที่ (Gain) ของ K_D K_I และ K_D ตามรูปแบบ

มาตราฐานของโปรแกรม MATLAB ด้วยทฤษฎีของเส้นทางเดินราก (Root Locus) หรือ ทฤษฎีของผลตอบสนองเชิงความถี่ (Frequency Response) ซึ่งในโปรแกรม MATLAB จะมีเครื่องมือสำหรับปรับค่าที่ใช้เรียกว่า SISO tool กล่าวคือเมื่อต้องการคำนวณค่าคงที่ (Gain) ของตัวควบคุม PID จะต้องสร้างระบบควบคุมขึ้นในโปแกรม MATLAB/SIMULINK ดังรูปที่ 3.9



รูปที่ 3.9 บล็อกไดอะแกรมของระบบควบคุมที่ใช้ตัวควบคุม PID

จากรูปที่ 3.9 จะประกอบด้วยตัวควบคุม PID ซึ่งสามารถปรับค่าคงที่ (Gain) ของ K_D K_I และ K_D ด้วยการเลือกชนิดของตัวควบคุมเพื่อนำมาใช้งานดังตารางที่ 3.1 และเมื่อต้องการใช้งานการปรับค่าคงที่แบบอัตโนมัติให้สามารถทำได้ตามภาคผนวก ก.

ตารางที่ 3.1 แสดงค่าพารามิเตอร์ของตัวควบคุม PID

ชนิดของตัวควบคุม	คำอธิบาย
P	ตัวควบคุมที่ใช้ตัวคูณอย่างเดียว
I	ตัวควบคุมที่ใช้ตัวปริพันธ์อย่างเดียว
PI	ตัวควบคุมที่ใช้ตัวคูณและปริพันธ์
PD	ตัวควบคุมที่ใช้ตัวคูณและอนุพันธ์
PID	ตัวควบคุมที่ใช้ตัวคูณ อนุพันธ์และปริพันธ์

3.4.2 การสร้างแบบจำลองของตัวควบคุม NNPID

รูปที่ 3.10 แสดงภาพของ Simulink Block Function

ในงานวิจัยนี้ได้ใช้บล็อกฟังก์ชันของ SIMULINK ที่เรียกว่า “Simulink Block Function” ดังรูปที่ 3.10 โดยแต่ละตัวควบคุมประกอบด้วยบล็อกฟังก์ชัน 2 บล็อกเพื่อสร้างแบบจำลองของตัวควบคุม NNPID และเขียนโค้ดโปรแกรมของอัลกอริทึมในแบบต่าง ๆ ซึ่งมีรายละเอียดดังนี้

3.3.2.1 ตัวควบคุม NNPID ตามงานวิจัย [2] ใช้โครงสร้างของตัวควบคุมดังรูปที่ 3.1 โดยกำหนดให้น้ำหนักโครงข่ายในชั้นเอาต์พุตมีค่าเป็น 1 และใช้น้ำหนักในชั้นซ่อนในการปรับค่า Gain ของตัวควบคุมส่วนฟังก์ชันกระตุ้น (Activation function) กำหนดให้เป็นฟังก์ชันเชิงเส้นทั้งหมด ซึ่งสามารถเขียนโค้ดโปรแกรมสำหรับโครงสร้างของตัวควบคุมดังรูปที่ ข.1 ในส่วนของอัลกอริทึมการเรียนรู้ให้เขียนตามโค้ดโปรแกรมดังรูปที่ ข.2 (อ้างอิงตามภาคผนวก ข)

3.3.2.2 ตัวควบคุม ENNPID ใช้โครงสร้างของตัวควบคุมชนิดนี้ดังรูปที่ 3.1 โดยกำหนดให้น้ำหนักโครงข่ายในชั้นเอาต์พุตมีค่าเป็น 1 และใช้น้ำหนักในชั้นซ่อนในการปรับค่า Gain ของตัวควบคุมส่วนฟังก์ชันกระตุ้น (Activation function) กำหนดให้เป็นฟังก์ชันเชิงเส้นทั้งหมด ซึ่งสามารถเขียนโค้ดโปรแกรมสำหรับโครงสร้างของตัวควบคุมดังรูปที่ ข.1 ในส่วนของอัลกอริทึมการเรียนรู้ตามโค้ดโปรแกรมดังรูปที่ ข.3 (อ้างอิงตามภาคผนวก ข)

3.3.2.3 ตัวควบคุม Hybrid CKF-NNPID ใช้โครงสร้างของตัวควบคุมดังรูปที่ 3.1 โดยกำหนดให้สามารถปรับค่าน้ำหนักโครงข่ายในชั้นเอาต์พุตและชั้นซ่อนเพื่อหาค่า Gain ของตัวควบคุมส่วนฟังก์ชันกระตุ้น (Activation function) กำหนดให้ชั้นซ่อนเป็นฟังก์ชัน tanh และในชั้นเอาต์พุตเป็นฟังก์ชันเชิงเส้น ซึ่งสามารถเขียนโค้ดโปรแกรมสำหรับโครงสร้างของตัวควบคุมดังรูปที่ ข.1 และในส่วนของอัลกอริทึมการเรียนรู้สามาถเขียนโค้ดโปรแกรมดังรูปที่ ข.4 (อ้างอิงตามภาคผนวก ข)

3.3.2.4 ตัวควบคุม ABF-NNPID ตัวควบคุมชนิดนี้ใช้โครงสร้างดังรูปที่ 3.1 โดยกำหนดให้สามารถปรับค่าน้ำหนักโครงข่ายในชั้นเอาต์พุตและชั้นซ่อนเพื่อหาค่า Gain ของตัวควบคุมส่วนฟังก์ชันกระตุ้น (Activation function) กำหนดให้ชั้นซ่อนเป็นฟังก์ชัน tanh และในชั้นเอาต์พุตเป็นฟังก์ชันเชิงเส้น ซึ่งสามารถเขียนโค้ดโปรแกรมสำหรับโครงสร้างของตัวควบคุมดังรูปที่ ข.1 และในส่วนของอัลกอริทึมการเรียนรู้สามาถเขียนโค้ดโปรแกรมดังรูปที่ ข.5 (อ้างอิงตามภาคผนวก ข)

3.3.2.5 ตัวควบคุม NNPID-AC ใช้โครงสร้างของตัวควบคุมดังรูปที่ 3.1 โดยกำหนดให้สามารถปรับค่าน้ำหนักโครงข่ายในชั้นเอาต์พุตและชั้นซ่อนเพื่อหาค่า Gain ของตัวควบคุมส่วนฟังก์ชันกระตุ้น (Activation function) กำหนดให้ชั้นซ่อนเป็นฟังก์ชัน tanh และในชั้นเอาต์พุตเป็นฟังก์ชันเชิงเส้น ซึ่งสามารถเขียนโค้ดโปรแกรมสำหรับโครงสร้างของตัวควบคุมดังรูปที่ ข.1 และในส่วนของอัลกอริทึมการเรียนรู้สามาถเขียนโค้ดโปรแกรมดังรูปที่ ข.6 (อ้างอิงตามภาคผนวก ข)

บทที่ 4
ผลการทดลอง

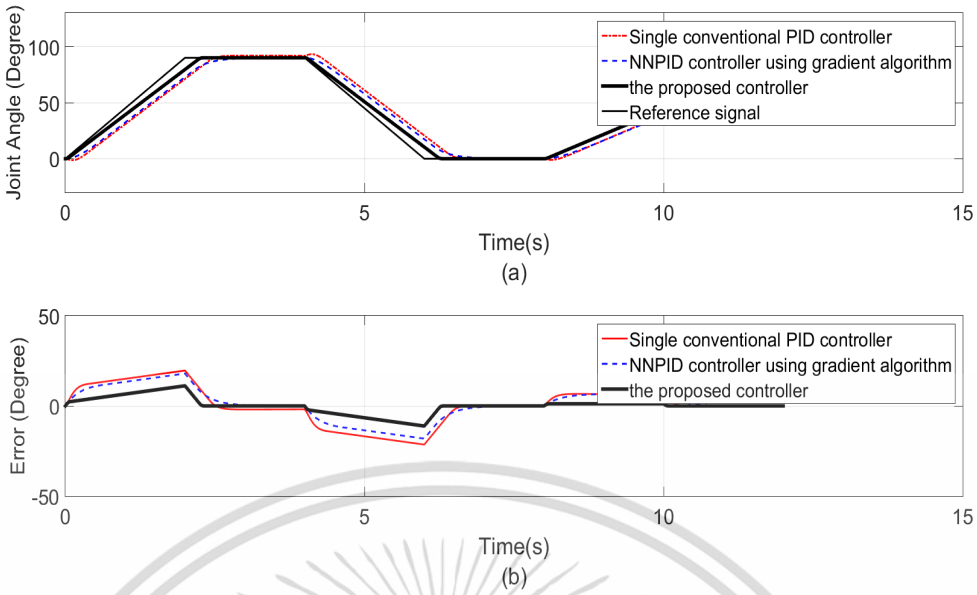
หลังจากที่ได้กล่าวถึงวิธีการดำเนินงานวิจัยเรื่องการออกแบบตัวควบคุมพร้อมด้วยวิธีการสร้างแบบจำลองด้วยโปรแกรม MATLAB ในหัวข้อผ่านมา ในบทนี้จะกล่าวถึงการนำแบบจำลองของระบบควบคุมที่มีตัวควบคุมแบบต่าง ๆ ที่นำเสนอในวิทยานิพนธ์ฉบับนี้มาทดลองควบคุมแขนกลมอเตอร์ไฟฟ้าด้วยแบบจำลองของระบบควบคุมแบบปิด โดยแบ่งการทดลองออกเป็น 4 การทดลอง ได้แก่ อันดับแรกคือการทดลองของระบบควบคุมที่ใช้ตัวควบคุม ENNPID (ตัวควบคุมที่ใช้อัลกอริทึมการเรียนรู้แบบส่วนขยายของตัวกรองคาลมาน, mEKF algorithm) โดยมีการทดสอบ 2 ส่วนคือการทดสอบด้วยแบบจำลองของระบบควบคุมเพนดูลัมผกผัน และการทดสอบด้วยแบบจำลองของระบบควบคุมมอเตอร์ไฟฟ้า ส่วนที่สองคือการทดลองเกี่ยวกับระบบควบคุมที่ใช้ตัวควบคุม Hybrid CKF-NNPID (ตัวควบคุมที่ใช้อัลกอริทึมการเรียนรู้แบบผสมตัวกรองคาล) ควบคุมแขนกลมอเตอร์ไฟฟ้า ส่วนที่สามจะเป็นการกล่าวถึงส่วนของการทดลองของระบบควบคุมที่ควบคุมแขนกลมอเตอร์ไฟฟ้าด้วยตัวควบคุม ABF-NNPID (ตัวควบคุมที่ใช้อัลกอริทึมการเรียนรู้แบบตัวกรองเบย์แบบประยุกต์) และสุดท้ายเป็นส่วนของการทดลองของระบบควบคุมแขนกลมอเตอร์ไฟฟ้าที่ใช้ตัวควบคุม NNPID-AC (ตัวควบคุมที่ใช้อัลกอริทึมเรียนรู้แบบผสมระหว่างอัลกอริทึมปฏิบัติ-วิจารณ์แบบเสริมกำลัง และอัลกอริทึมตัวกรองคาลมานดีกรีสูง) โดยการทดลองทั้งหมดที่ได้กล่าวมาข้างต้นนั้นได้ถูกนำมาเปรียบเทียบในเรื่องประสิทธิภาพการใช้งานกับตัวควบคุม PID ที่นำมาจากไลบารี่ของโปรแกรม MATLAB

4.1 ผลการทดลองของระบบควบคุมที่อยู่ภายใต้ตัวควบคุม ENNPID

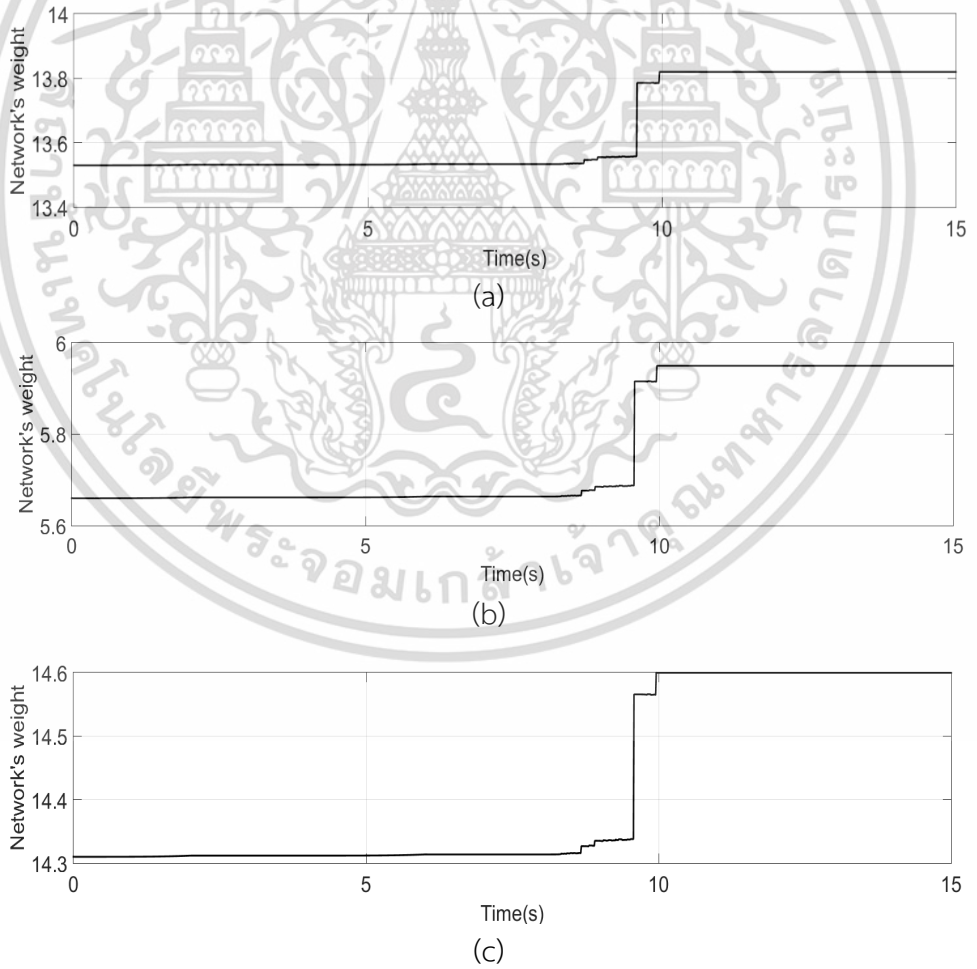
การทดสอบประสิทธิภาพของตัวควบคุม ENNPID ที่นำเสนอนี้ได้ทดสอบกับระบบควบคุมมอเตอร์ไฟฟ้าและระบบควบคุมเพนดูลัมผกผัน (Inverted Pendulum) ซึ่งมีรายละเอียดดังนี้

4.1.1 ระบบควบคุมมอเตอร์ไฟฟ้า

ส่วนนี้จะเป็นการทดลองของระบบควบคุมมอเตอร์ไฟฟ้าแบบปิด โดยมีการป้อนสัญญาณอินพุตให้กับมอเตอร์ไฟฟ้าเคลื่อนตามเป้าหมายที่กำหนด และเซนเซอร์จะเป็นตัววัดการเปลี่ยนแปลงมุมของมอเตอร์เพื่อห่าค่าความแตกต่างระหว่างค่าเป้าหมายและค่าเอาต์พุตจริง (ค่าผิดพลาดที่เกิดจากมุมของมอเตอร์ไฟฟ้าที่ตั้งไว้) ซึ่งผลการทดลองของตัวควบคุม ENNPID ที่ได้นั้นได้ทำการเปรียบเทียบกับตัวควบคุมมาตรฐาน PID เพื่อพิจารณาถึงประสิทธิภาพของตัวควบคุมที่นำเสนอดังรูปที่ 4.1



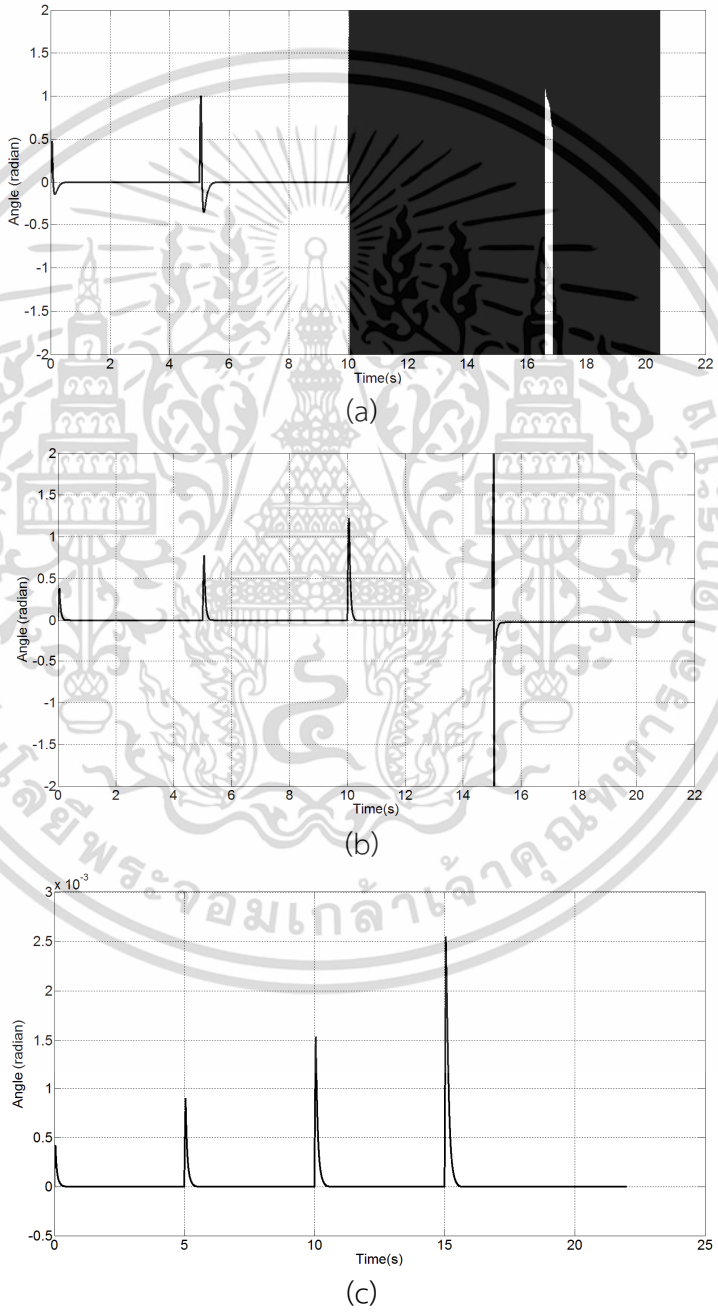
รูปที่ 4.1 (a) ผลตอบสนองขององศาการหมุนของมอเตอร์ไฟฟ้า (b) ผลตอบสนองของค่าผิดพลาดขององศาการหมุนเมื่อเทียบค่าสัญญาณป้อน



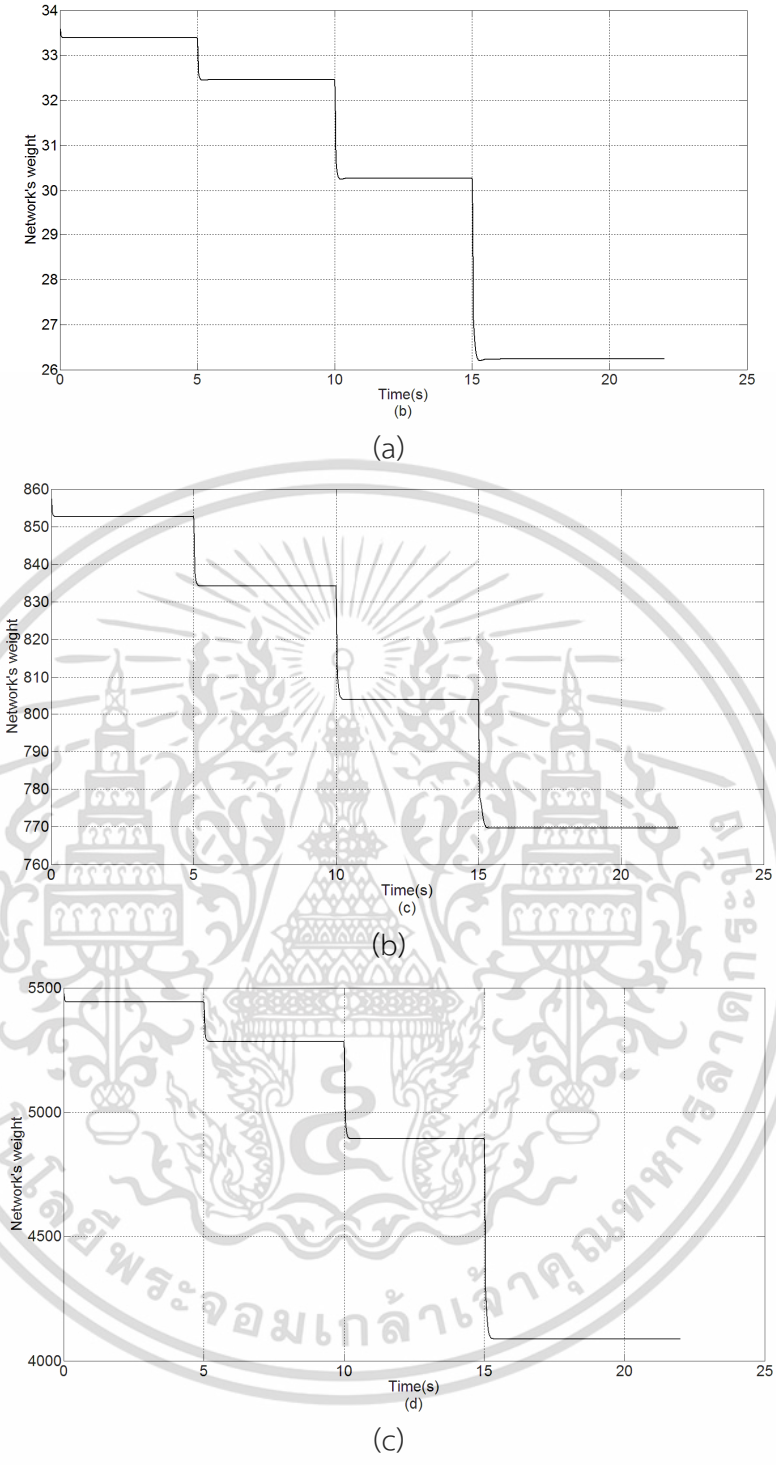
รูปที่ 4.2 การปรับค่าน้ำหนักโครงข่ายของตัวควบคุม NNPID โนดในชั้นซ่อน โดยที่ (a) ค่าน้ำหนักโนดตัวปริพันธ์ (K_I), (b) ค่าน้ำหนักโนดตัวคูณ (K_P) และ (c) ค่าน้ำหนักโนดตัวอนุพันธ์ (K_D)

4.1.2 ระบบควบคุมเพนดูลัมผกผัน

เพื่อพิสูจน์ถึงประสิทธิภาพของตัวควบคุม ENNPID ให้เป็นที่ประจักษ์มากขึ้น จึงนำตัวควบคุม ENNPID มาใช้ในระบบควบคุมของเพนดูลัมผกผันแบบปิด ซึ่งระบบจะต้องหาค่าอินพุตควบคุมที่เหมาะสมเพื่อให้เพนดูลัมอยู่ในสถานะเสถียรตลอดเวลา ซึ่งในที่นี่กำหนดให้มุมของเพนดูลัมเท่ากับศูนย์โดยกำหนดให้อยู่ในแนวแกนตั้งฉากกับพื้นโลก และกำหนดให้เป็นเป้าหมายของระบบ กล่าวคือสถานะของตัวเพนดูลัมจะต้องตั้งฉากกับแกนพื้นโลกเสมอเพื่อรักษาสถานะเสถียรของระบบ โดยมีผลการทดลองการใช้งานตัวควบคุมชนิดนี้ดังนี้



รูปที่ 4.3 ผลตอบสนองของมุมเพนดูลัมที่มีการใส่แรงผลักในวินาทีต่าง ๆ เมื่อใช้ตัวควบคุมชนิดต่าง ๆ โดยที่ (a) ใช้ตัวควบคุม ENNPID, (b) ใช้ตัวควบคุม NNPID [2] และ (c) ใช้ตัวควบคุม PID



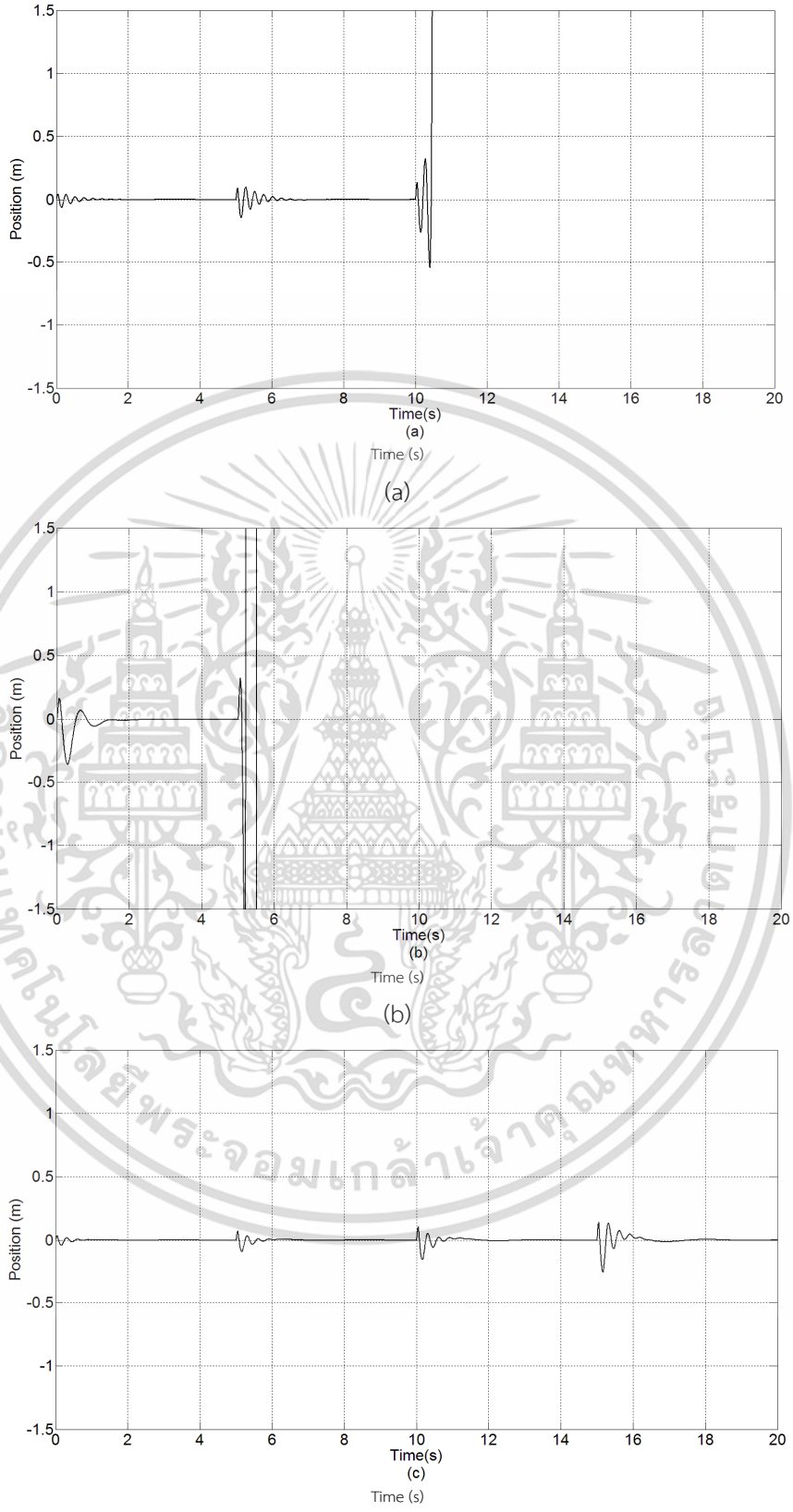
รูปที่ 4.4 แสดงการปรับค่าน้ำหนักโครงข่ายประสาทเทียมของตัวควบคุม NNPID โดยที่ (a) ค่า (K_I), (b) ค่า (K_P) และ (c) ค่า (K_D)

จากรูปที่ 4.1(a) แสดงถึงผลการทดลองของระบบควบคุมมอเตอร์ไฟฟ้ากระแสตรงที่มีการป้อนสัญญาณอินพุตให้เคลื่อนที่ตามที่เป้าหมายที่ตั้งไว้ และยังแสดงผลลัพธ์เป็นมุมของมอเตอร์ไฟฟ้าจริงที่ได้ด้วยการเปรียบเทียบค่าผลตอบสนองของมุมมอเตอร์ไฟฟ้าที่เกิดขึ้นของตัวควบคุมที่นำเสนอ (ตัว

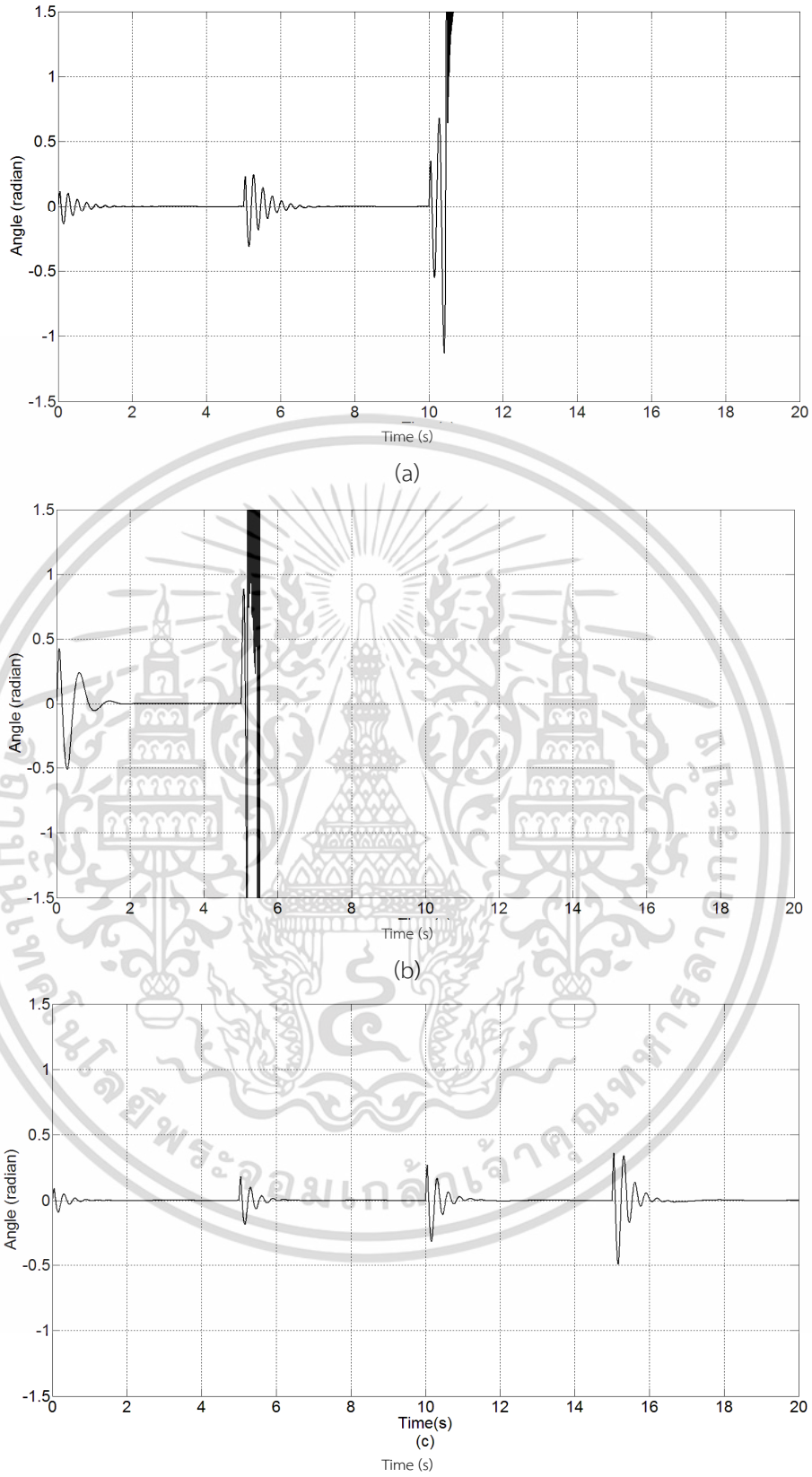
ควบคุม ENNPID) กับค่าผลตอบสนองของตัวควบคุมตัวควบคุม PID และค่าผลตอบสนองของตัวควบคุม NNPID จากงานวิจัย [2] ซึ่งผลการทดลองชี้ให้เห็นว่าประสิทธิภาพของตัวควยคุมที่นำเสนอมีความสามารถติดตามค่าผิดพลาดที่เกิดขึ้นจากระบบได้ดีกว่าตัวควบคุมที่นำมาเปรียบเทียบทั้งสองดังแสดงไว้ในรูปดังรูปที่ 4.1(b) ซึ่งเป็นค่าพลาดที่ขึ้นมาจากทดสอบของระบบมอเตอร์ไฟฟ้าที่มีเป้าหมายเป็นศูนย์ นอกเหนือจากนี้ในวิทยานิพนธ์ฉบับนี้ก็ยังมีผลการทดลองที่แสดงให้เห็นการปรับน้ำหนักของตัวควบคุม ENNPID ที่นำเสนอสามารถปรับค่าใหม่ได้เมื่อระบบมีการเปลี่ยนแปลงดังรูปที่ 4.2

เพื่อให้ผลการทดลองเป็นที่ประจักษ์มากขึ้นจึงได้นำตัวควบคุม ENNPID ที่นำเสนอนี้ไปทดสอบกับระบบควบคุมเพนดูลัมผกผันเพิ่มเติม โดยระบบการทดลองนี้คำนวณค่าอินพุตควบคุม (Control input, u) อย่างเหมาะสมเพื่อให้ระบบเพนดูลัมอยู่ในสถานะเสถียร (ตั้งฉากกับพื้นโลกหรือมีค่ามุมของเพนดูลัมที่วัดได้เป็นศูนย์) ทั้งนี้ในการทดลองยังมีการใส่แรงผลัก 3 ขนาดในวินาทีที่ 5 วินาทีที่ 10 และวินาทีที่ 15 โดยเรียงแรงผลักที่มากขึ้นตามลำดับ เพื่อให้ระบบเกิดการเสียสมดุลทั้งนี้เพื่อพิจารณาผลตอบสนองของของตัวควบคุมต่อการสิ่งรบกวน ซึ่งผลปรากฏว่าตัวควบคุม PID ไม่สามารถรองรับสิ่งรบกวนเมื่อใส่แรงผลักในวินาทีที่ 10 ดังรูปที่ 4.3(a) กล่าวคือระบบของเพนดูลัมไม่สามารถต้านการผลักของแรงจึงทำให้แบบจำลองของเพนดูลัมเกิดการแกว่งไปมา ส่วนในกรณีของระบบควบคุมที่อยู่ภายใต้ตัวควบคุม NNPID [2] นั้นสามารถรับมือการแรงผลักในวินาทีที่ 10 ได้แต่เมื่อใส่แรงผลักในวินาทีที่ 15 ที่มีขนาดมากขึ้นไปอีกทำให้แบบจำลองของเพนดูลัมเกิดการแกว่งไปมาเช่นเดียวกันดังรูปที่ 4.3(b) (กราฟผลการทดลองนี้จะตัดกราฟแสดงผลถึงแค่วินาทีที่ 15 เพื่อความสวยงาม) ทั้งนี้การทดลองนี้แสดงให้เห็นว่าตัวควบคุม ENNPID มีประสิทธิภาพที่เหนือกว่าตัวควบคุมชนิดอื่น ๆ ที่นำมาเปรียบเทียบเนื่องจากตัวควบคุม ENNPID นี้สามารถปรับค่า Gain ของตัวควบคุมได้อย่างอิสระตามการเปลี่ยนแปลงของค่าผิดพลาดของระบบดังแสดงไว้ในรูปที่ 4.4

นอกเหนือการทดลองในระบบควบคุมตัวแปรเดี่ยว (ระบบควบคุมแบบ SISO) แล้ว สำหรับในการทดลองส่วนนี้เป็นการทดสอบตัวควบคุม ENNPID กับระบบควบคุมที่มีหลายตัวแปร (MIMO) กล่าวคือการทดลองนี้จะเป็นการควบคุมระบบเพนดูลัมโดยเป็นการควบคุมทั้งการทรงตัว และตำแหน่งการเคลื่อน และเช่นเดียวกับการทดสอบที่ผ่านคือใส่แรงผลักเพื่อทดสอบ ณ เวลาต่าง ๆ ซึ่งผลปรากฏว่าตัวควบคุมที่นำเสนอนี้ก็ยังสามารถรองรับปัจจัยแวดล้อมเปลี่ยนแปลงที่เกิดขึ้นในดีกว่าตัวควบคุมอื่น ๆ ดังรูปที่ 4.5 และรูปที่ 4.6 ดังนั้นจึงเป็นข้อบ่งชี้ให้เห็นว่าตัวควบคุมที่นำเสนอนี้สามารถควบคุมมุมของมอเตอร์ไฟฟ้ากระแสตรงและเพนดูลัมผกผันได้อย่างมีประสิทธิภาพกว่าตัวควบคุม PID และตัวควบคุม NNPID [2] อีกทั้งยังแสดงให้เห็นว่าการใช้อัลกอริทึมการเรียนรู้ mEKF algorithm ซึ่งเป็นอัลกอริทึมสำหรับการพยากรณ์สถานะของระบบที่เป็นพลวัตแบบไม่เป็นเชิงเส้นได้อย่างมีประสิทธิภาพ และมีศักยภาพมากกว่าอัลกอริทึมการเรียนรู้แบบ Gradient Descent ทั่วไปทั้งในเรื่องประสิทธิภาพและการรองรับระบบควบคุมที่มีหลายตัวแปรของงานระบบควบคุมที่มีค่าเหนือกว่าตัวควบคุม NNPID ตามงานวิจัย [2]



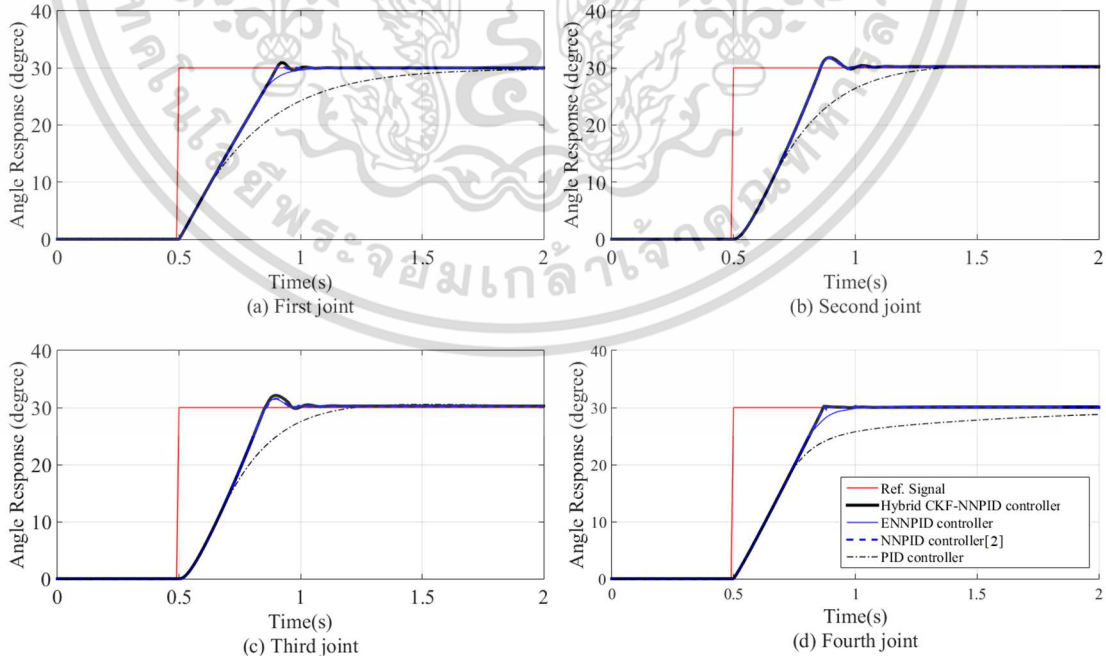
รูปที่ 4.5 ผลตอบสนองของตำแหน่งการเคลื่อนที่ของระบบควบคุมเพนดูลัมผกผันภายใต้ตัวควบคุมต่าง ๆ โดยที่ (a) ใช้ตัวควบคุม PID, (b) ใช้ตัวควบคุม NNPID [2] และ (c) ใช้ตัวควบคุม ENNPID



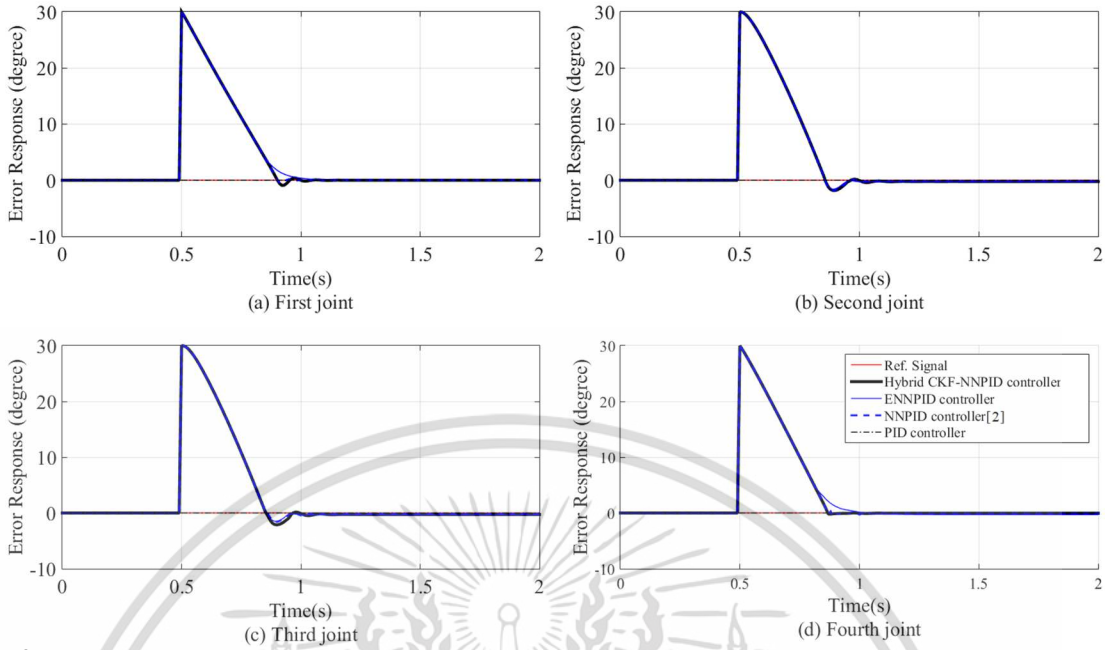
รูปที่ 4.6 ผลตอบสนองของมุมที่เกิดขึ้นของระบบควบคุมเพนดูลัมผกผันด้วยตัวควบคุมแบบต่าง ๆ โดยที่ (a)ใช้ตัวควบคุม PID, (b) ใช้ตัวควบคุม NNPID [2] และ (c) ใช้ตัวควบคุม ENNPID

4.2 ผลการทดลองของระบบควบคุมที่อยู่ภายใต้ตัวควบคุม Hybrid CKF-NNPID

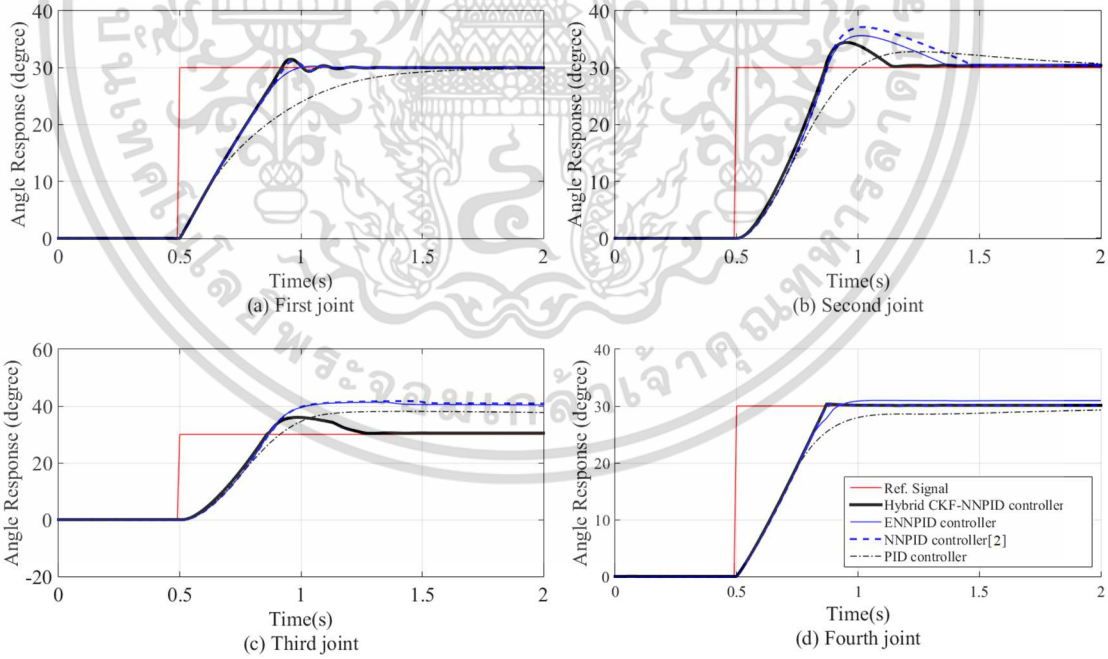
ในหัวข้อนี้จะแสดงผลการทดลองของระบบควบคุมที่ออกแบบโครงสร้างและวิธีการเขียนโค้ดโปรแกรมของตัวควบคุมดังรายละเอียดในหัวข้อที่ 3.1 และหัวข้อที่ 3.2.2 ตามลำดับ ซึ่งเป็นผลการทดลองของระบบควบคุมแขนกลไฟฟ้าภายใต้ตัวควบคุมแบบต่าง ๆ ด้วยการป้อนอินพุตเป้าหมายสัญญาณแบบขั้น (Unit Step) ที่มีหน่วยเป็นองศาของการหมุนเพื่อพิจารณาถึงประสิทธิภาพหรือที่เรียกว่า “Step Response” โดยการทดลองในหัวข้อนี้จะมุ่นเน้นเรื่องการแสดงถึงประสิทธิภาพของตัวควบคุมในสถานการณ์ต่าง ๆ ได้แก่ การทำงานในสถาการณ์ปกติ (ไม่มีภาระโหลด และไม่มีสิ่งรบกวน) การทำงานในสถาการณ์ที่ใส่ภาระโหลดสูงสุด และสุดท้ายการทำงานของระบบในสถานการณ์ที่มีสิ่งรบกวน และพร้อมกันนี้ได้สร้างระบบควบคุมที่อยู่ภายใต้ตัวควบคุม PID ตัวควบคุม NNPID [2] และตัวควบคุม ENNPID มาเปรียบเทียบประสิทธิภาพกับตัวควบคุมที่นำเสนอ (Hybrid CKF-NNPID) ซึ่งจากผลการทดลองจะเห็นได้ว่าตัวควบคุม Hybrid CKF-NNPID สามารถให้ผลตอบสนองของระบบได้ดีกว่าตัวควบคุมอื่น ๆ ดังรูปที่ 4.7 และแสดงกราฟในรูปแบบของค่าผิดพลาดของระบบเพื่อให้ง่ายต่อการพิจารณาการลู่เข้าสู่จุดสมดุลดังรูปที่ 4.8 ในส่วนของการทำงานภายใต้การใส่โหลดที่รับได้สูงสุด ตัวควบคุมที่นำเสนอนี้ก็ยังสามารถปรับค่าน้ำหนักที่เหมาะสมให้แก่ระบบได้ดีกว่าตัวควบคุมอื่น ๆ โดยแสดงเป็นรูปแบบของกราฟผลตอบสนองแบบขั้น และกราฟในรูปแบบของค่าผิดพลาดของระบบดังแสดงในรูปที่ 4.9 และรูปที่ 4.10 ตามลำดับ และสุดท้ายเป็นผลการทดลองที่ชี้ประสิทธิภาพของการทำงานของตัวควบคุมที่นำเสนอก็ยังสามารถทำงานได้ดีกว่าตัวควบคุมอื่น ๆ ภายใต้สภาวะถูกสิ่งรบกวนดังแสดงผลการทดลองในรูปที่ 4.11 และรูปที่ 4.12



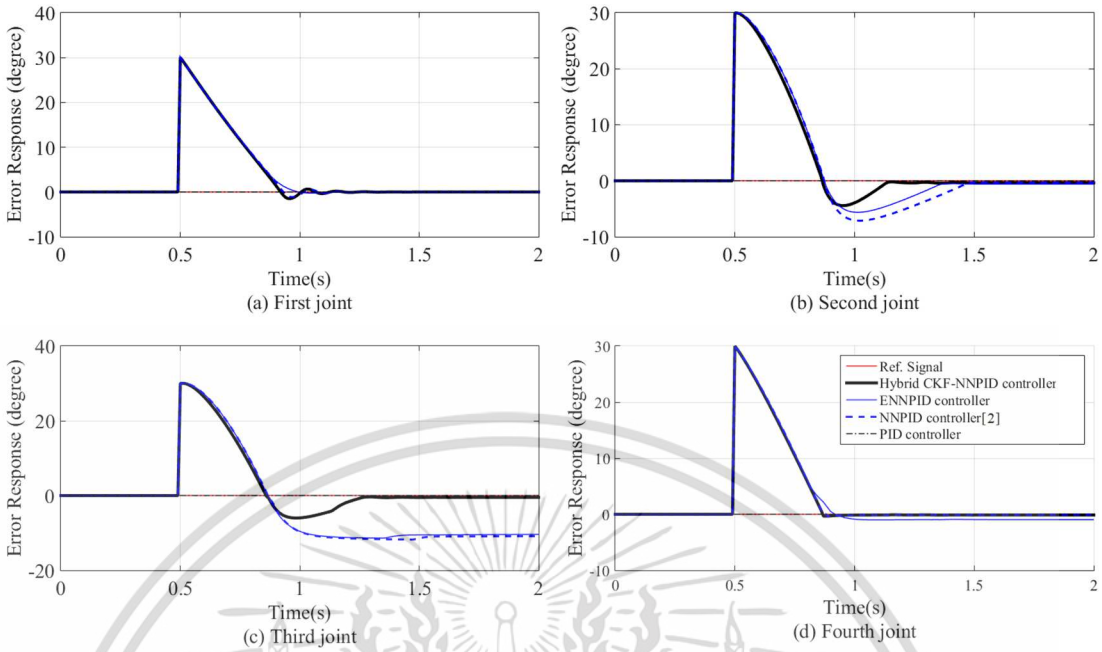
รูปที่ 4.7 ผลตอบสนองของมุมแขนกลไฟฟ้าภายใต้ตัวควบคุมแบบต่าง ๆ แบบไม่มีภาระโหลด (load) โดยที่ (a) ข้อที่ 1 (b) ข้อที่ 2 (c) ข้อที่ 3 (d) ข้อที่ 4



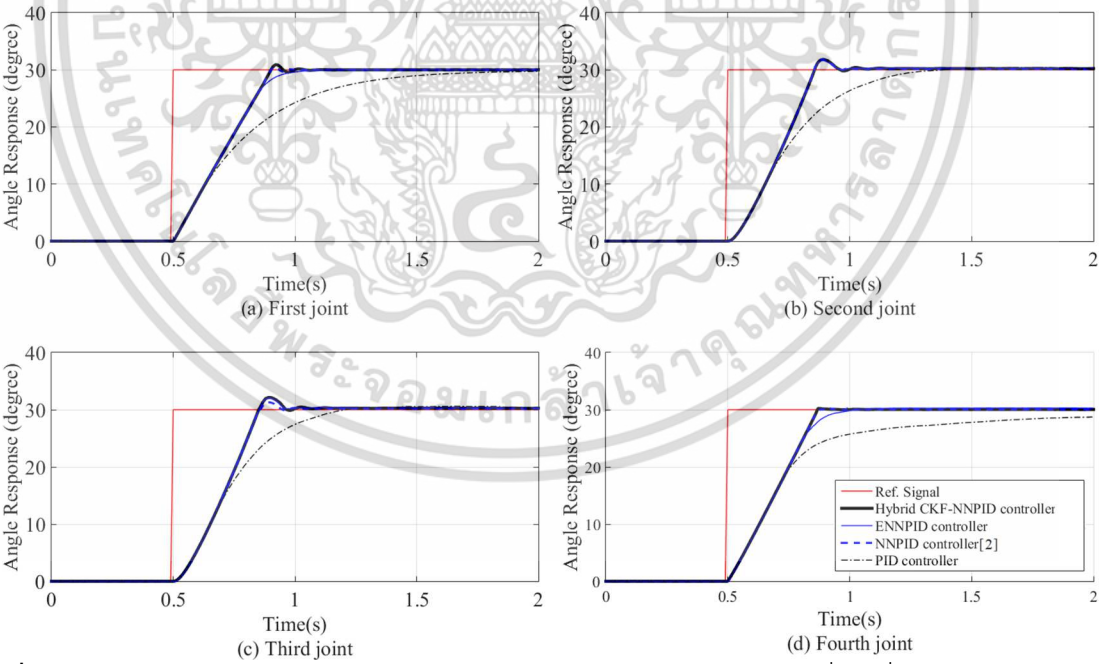
รูปที่ 4.8 ผลตอบสนองของค่าผิดพลาดของมุมแขนกลไฟฟ้าภายใต้ตัวควบคุมแบบต่าง ๆ แบบไม่มีภาระโหลด (load) โดยที่ (a) ข้อที่ 1 (b) ข้อที่ 2 (c) ข้อที่ 3 (d) ข้อที่ 4



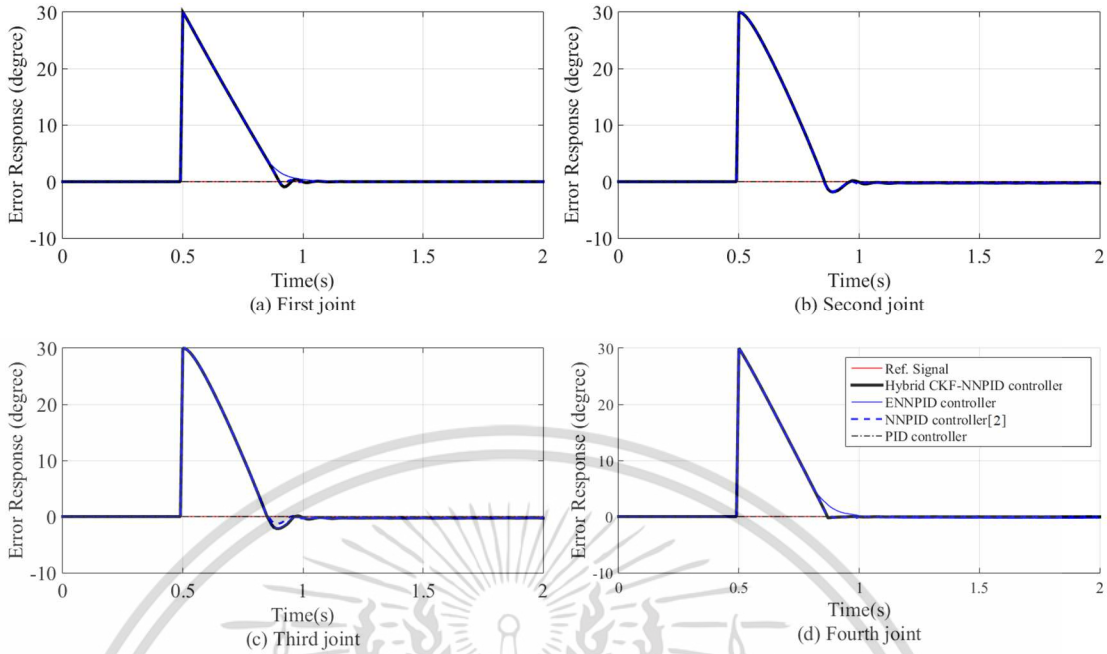
รูปที่ 4.9 ผลตอบสนองของมุมแขนกลไฟฟ้าภายใต้ตัวควบคุมแบบต่าง ๆ เมื่อใส่ภาระโหลดสูงสุด (Maximum load) โดยที่ (a) ข้อที่ 1 (b) ข้อที่ 2 (c) ข้อที่ 3 (d) ข้อที่ 4



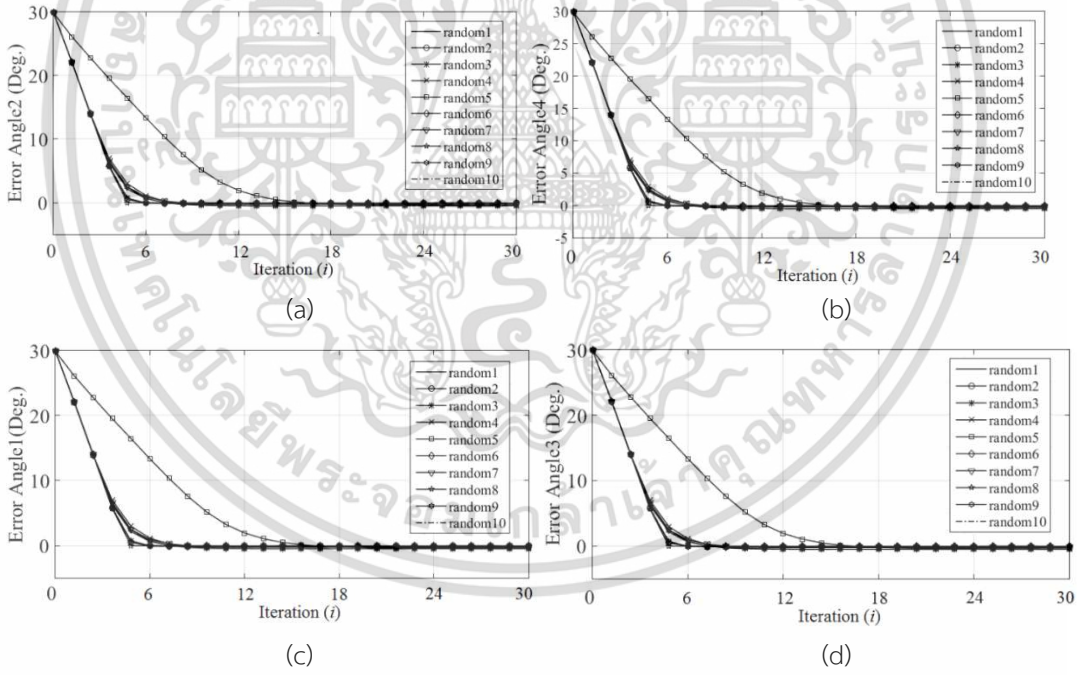
รูปที่ 4.10 ผลตอบสนองค่าผิดพลาดของมุมแขนกลไฟฟ้าภายใต้ตัวควบคุมแบบต่าง ๆ เมื่อใส่ภาระโหลดสูงสุด (Maximum load) โดยที่ (a) ข้อที่ 1 (b) ข้อที่ 2 (c) ข้อที่ 3 (d) ข้อที่ 4



รูปที่ 4.11 ผลตอบสนองของมุมแขนกลไฟฟ้าภายใต้ตัวควบคุมแบบต่าง ๆ เมื่อใส่สิ่งรบกวน (Noise) โดยที่ (a) ข้อที่ 1 (b) ข้อที่ 2 (c) ข้อที่ 3 (d) ข้อที่ 4



รูปที่ 4.12 ผลตอบสนองค่าผิดพลาดของมุมแขนกลไฟฟ้าภายใต้ตัวควบคุมแบบต่าง ๆ เมื่อใส่สิ่งรบกวน (Noise) โดยที่ (a) ข้อที่ 1 (b) ข้อที่ 2 (c) ข้อที่ 3 (d) ข้อที่ 4

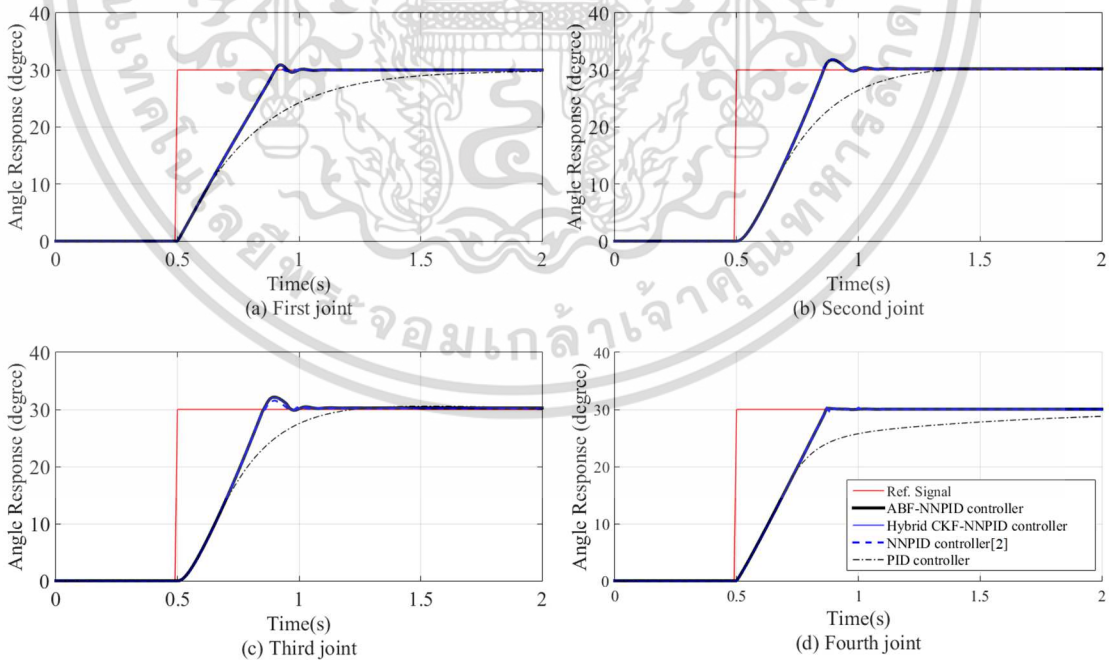


รูปที่ 4.13 การลู่เข้าสู่จุดสมดุลของการสุ่มการค่าเริ่มต้นของตัวควบคุม CKF-NNPID ของข้อแขนกลไฟฟ้าต่าง ๆ โดยที่ (a) ข้อที่ 1 (b) ข้อที่ 2 (c) ข้อที่ 3 (d) ข้อที่ 4

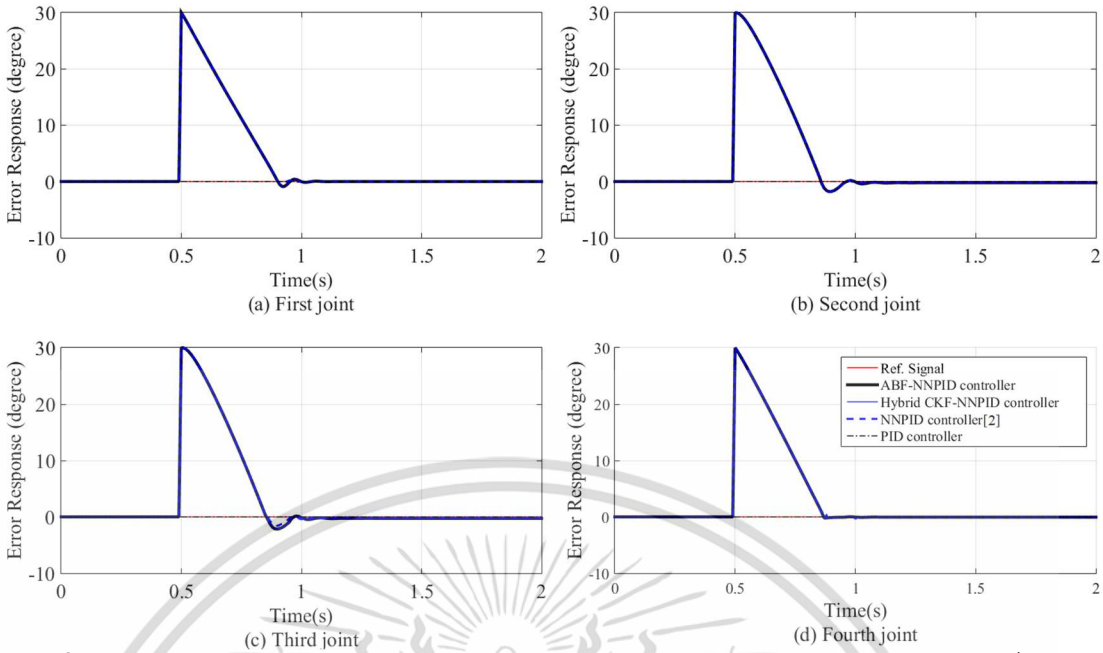
นอกเหนือจากนี้งานวิจัยนี้ยังได้ชี้ให้ถึงความสามารถในการแก้ปัญหาในเรื่องของการหาค่าน้ำหนักเริ่มต้น (Initial weight) ของตัวควบคุมที่นำเสนอนี้อีกด้วย จากรูปที่ 4.13 แสดงถึงการลู่เข้าสู่จุดสมดุลของระบบควบคุมที่สามารถนำค่าคงที่ (Gain) เริ่มต้นด้วยเลขสุ่มที่เป็นช่วง

4.3 ผลการทดลองของระบบควบคุมที่อยู่ภายใต้ตัวควบคุม ABF-NNPID

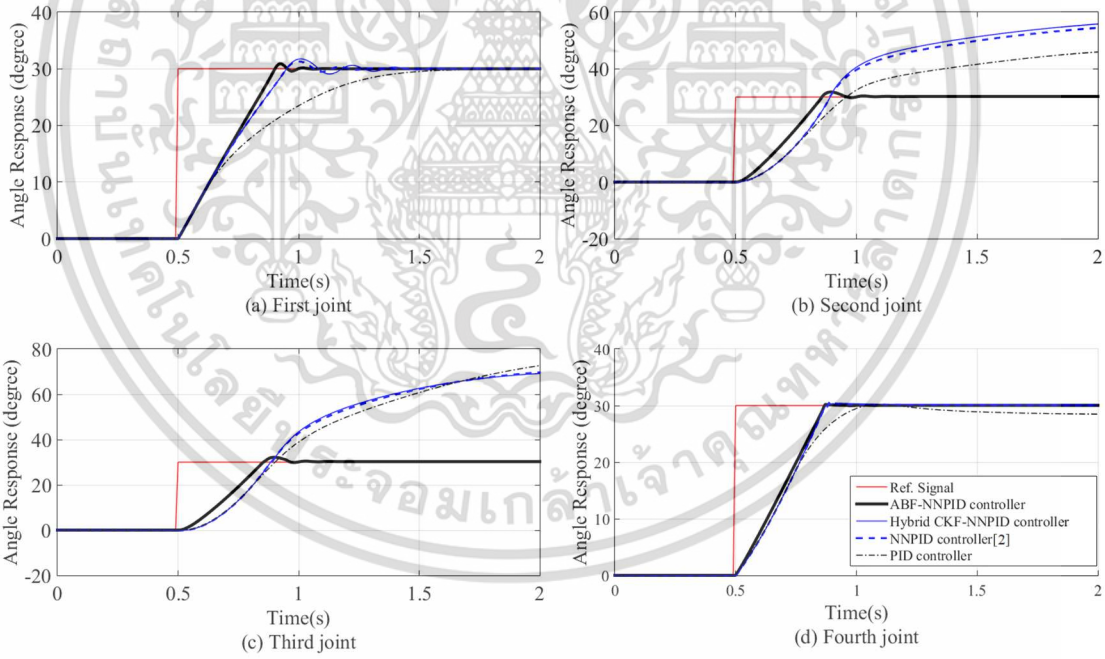
หลังจากที่ได้สร้างแบบจำลองของระบบควบคุมดังที่ได้อธิบายไว้ในหัวข้อที่ผ่านมา พร้อมด้วยวิธีการออกแบบโครงสร้างและวิธีการเขียนโค้ดโปรแกรมของตัวควบคุมตามอัลกอริทึมการเรียนรู้ของเบย์แบบประยุกต์ดังในหัวข้อที่ 3.1 และหัวข้อที่ 3.2.3 ตามลำดับแล้ว ผลปรากฏว่าระบบควบคุมแขนกลไฟฟ้าที่ควบคุมด้วยตัวควบคุมที่นำเสนอนี้มีความเร็วในการลู่เข้าสู่จุดเสถียรได้ดีกว่าตัวควบคุมชนิดอื่น ๆ ทั้งสถานการณ์ปกติ สถานการณ์การใส่โหลดสูงสุด และการถูกสิ่งรบกวนดังรูปที่ 4.14 รูปที่ 4.16 และรูปที่ 4.18 ตามลำดับ โดยเฉพาะอย่างยิ่งผลการทดลองของระบบควบคุมแขนกลไฟฟ้าภายใต้สถานการณ์ที่มีโหลดสูงสุดในข้อต่อที่ 3 และที่ 4 จะให้ผลการทดลองที่ชัดเจนมากเกี่ยวกับความแตกต่างระหว่างระบบควบคุมที่ใช้ตัวควบคุมที่นำเนอและตัวควบคุมอื่น ๆ เนื่องจากต้องรับภาระของน้ำหนักในการขับเคลื่อนหรือต้องใช้แรงบิดมากกว่าข้อต่ออื่น ๆ และเช่นเดียวกันกับการทดลองหัวข้อที่แล้ว ได้มีผลการทดลองที่แสดงกราฟในรูปแบบของค่าผิดพลาดของระบบทำให้ตัวควบคุมนี้สามารถแสดงศักยภาพของตัวควบคุมได้ชัดเจนดังรูปที่ 4.15 รูปที่ 4.17 และรูปที่ 4.19 สำหรับระบบควบคุมที่อยู่ภายใต้สถานการณ์ปกติ สถานการณ์ใส่โหลดสูงสุด และสถานการณ์ที่มีสิ่งรบกวนตามลำดับ นอกเหนือจากนี้ผลการทดลองยังได้แสดงเป็นที่ชัดเจนว่าตัวควบคุม ABF-NNPID มีประสิทธิภาพกว่าตัวควบคุมอื่น ๆ ที่ผ่านมาด้วยการนำภาระโหลดที่ระบบรับได้สูงสุด ซึ่งมีค่าเป็น 2 เท่าของโหลดในการทดลองในหัวข้อที่ 4.2



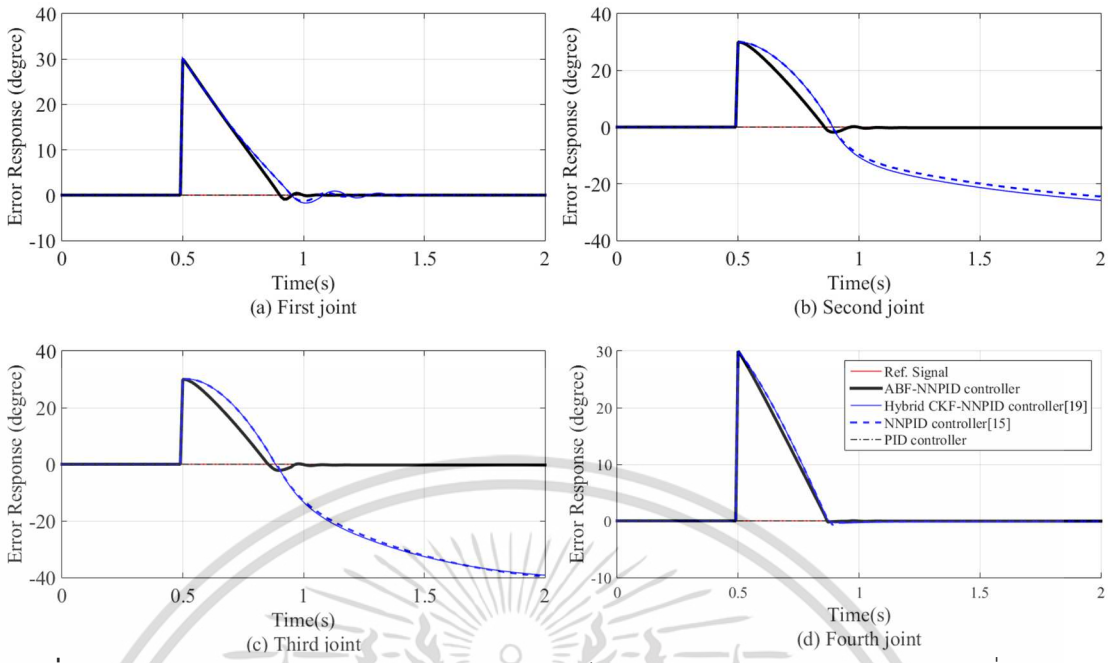
รูปที่ 4.14 ผลตอบสนองของมุมแขนกลไฟฟ้าภายใต้ตัวควบคุมแบบต่าง ๆ เมื่อไม่มีภาระโหลด (Load) โดยที่ (a) ข้อที่ 1 (b) ข้อที่ 2 (c) ข้อที่ 3 (d) ข้อที่ 4



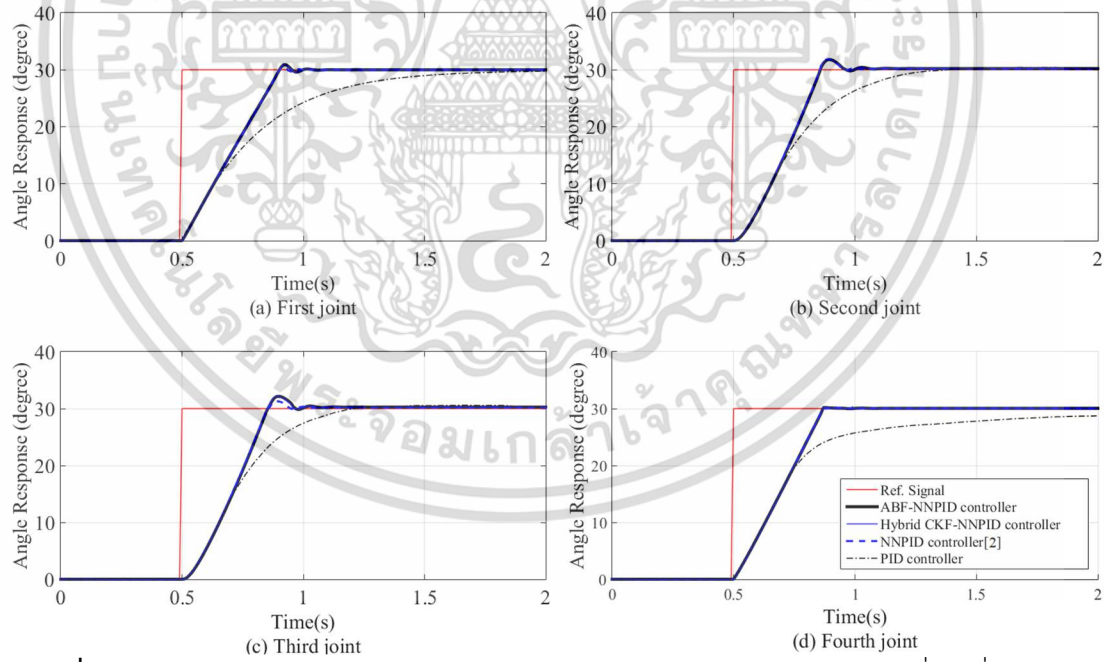
รูปที่ 4.15 ผลตอบสนองค่าผิดพลาดของมุมแขนกลไฟฟ้าภายใต้ตัวควบคุมแบบต่าง ๆ เมื่อไม่มีภาระโหลด (load) โดยที่ (a) ข้อที่ 1 (b) ข้อที่ 2 (c) ข้อที่ 3 (d) ข้อที่ 4



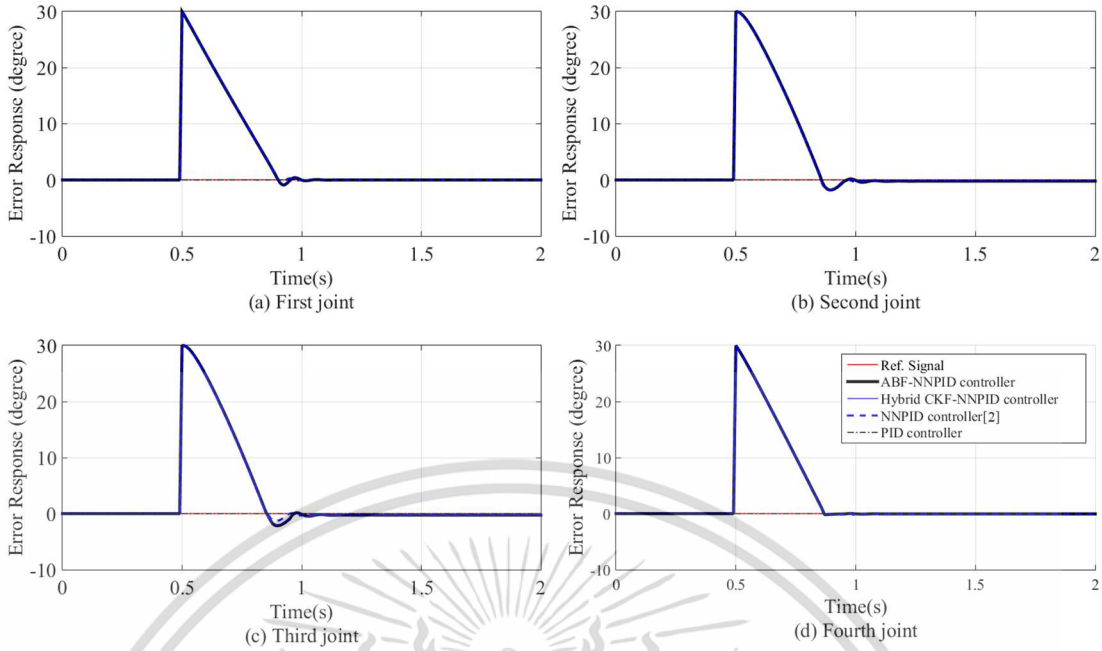
รูปที่ 4.16 ผลตอบสนองของมุมแขนกลไฟฟ้าภายใต้ตัวควบคุมแบบต่าง ๆ เมื่อใส่ภาระโหลดสูงสุด (Maximum load) โดยที่ (a) ข้อที่ 1 (b) ข้อที่ 2 (c) ข้อที่ 3 (d) ข้อที่ 4



รูปที่ 4.17 ผลตอบสนองค่าผิดพลาดของมุมแขนกลไฟฟ้าภายใต้ตัวควบคุมแบบต่าง ๆ เมื่อใส่ภาระโหลดสูงสุด (Maximum load) โดยที่ (a) ข้อที่ 1 (b) ข้อที่ 2 (c) ข้อที่ 3 (d) ข้อที่ 4



รูปที่ 4.18 ผลตอบสนองของมุมแขนกลไฟฟ้าภายใต้ตัวควบคุมแบบต่าง ๆ เมื่อใส่สิ่งรบกวน (Noise) โดยที่ (a) ข้อที่ 1 (b) ข้อที่ 2 (c) ข้อที่ 3 (d) ข้อที่ 4



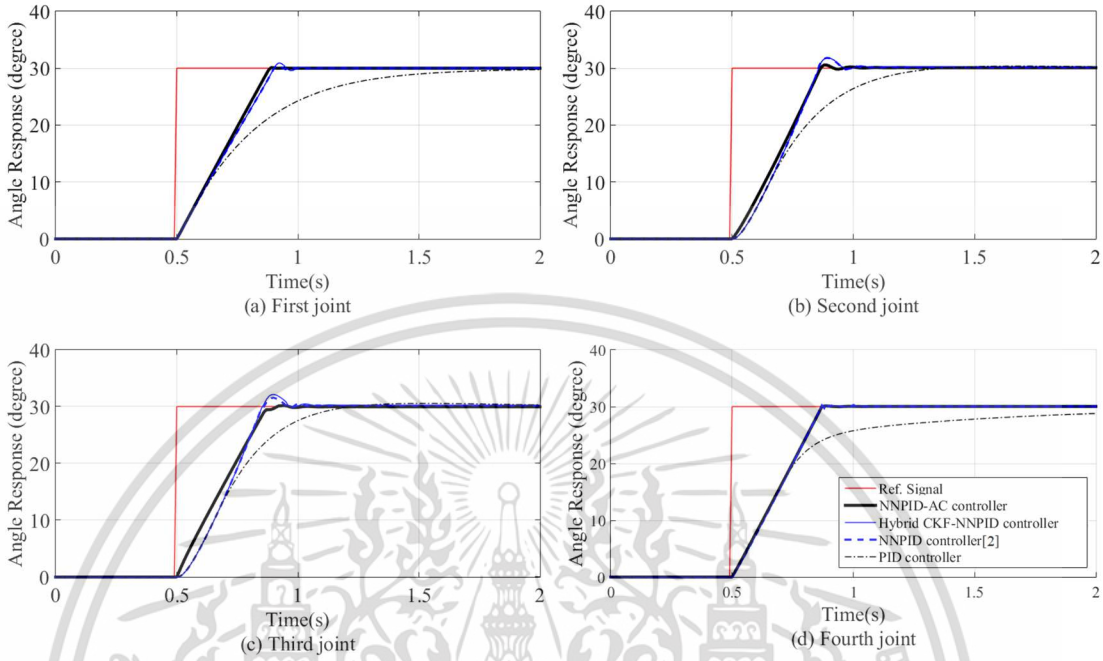
รูปที่ 4.19 ผลตอบสนองค่าผิดพลาดของมุมแขนกลไฟฟ้าภายใต้ตัวควบคุมแบบต่าง ๆ เมื่อใส่สิ่งรบกวน (Noise) โดยที่ (a) ข้อที่ 1 (b) ข้อที่ 2 (c) ข้อที่ 3 (d) ข้อที่ 4

4.4 ผลการทดลองของระบบควบคุมที่อยู่ภายใต้ตัวควบคุม NNPID-AC

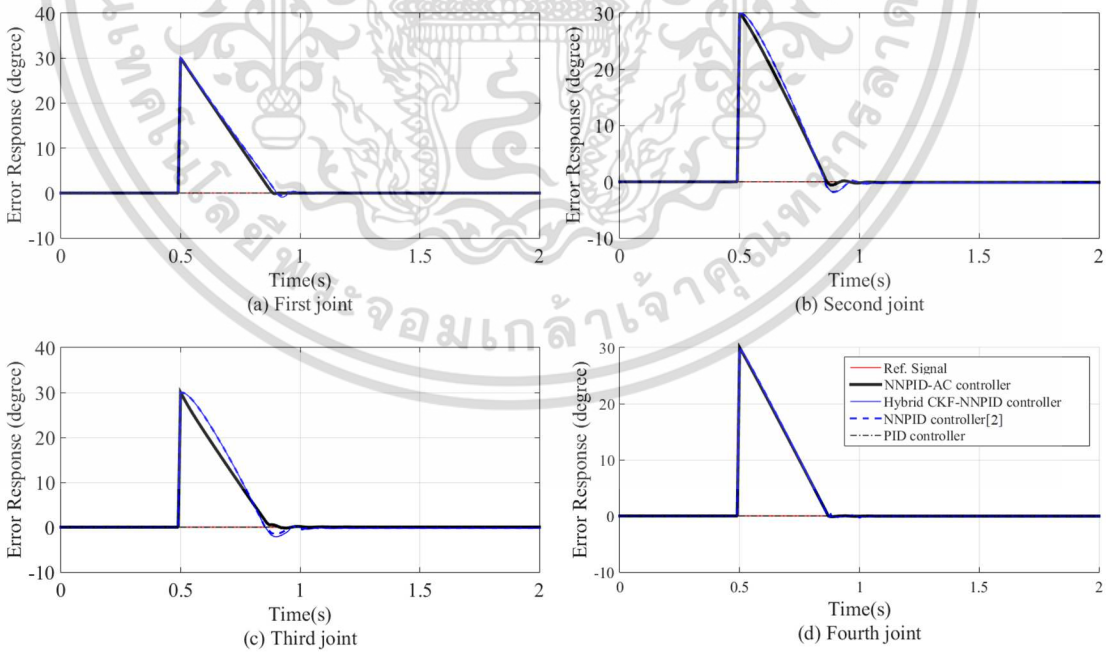
การทดลองของตัวควบคุม NNPID-AC นี้ถือเป็นตัวควบคุมที่พัฒนาถึงขั้นดีที่สุดของวิทยานิพนธ์ฉบับนี้ โดยมีวิธีการออกแบบโครงสร้างและวิธีการเขียนโค้ดโปรแกรมของตัวควบคุมตามอัลกอริทึมการเรียนรู้ปฏิบัติ-วิจารณ์แบบเสริมกำลัง (Actor-critic Reinforcement Algorithm) ดังแสดงรายละเอียดในหัวข้อที่ 3.1 และหัวข้อที่ 3.2.4 ตามลำดับ โดยจะเห็นได้ว่าระบบควบคุมแขนกลไฟฟ้าที่ควบคุมด้วยตัวควบคุม NNPID-AC นี้สามารถควบคุมให้แขนกลเคลื่อนที่ได้อย่างมีประสิทธิภาพทั้งในเรื่องความเร็วในการลู่เข้าสู่สถานะเสถียรกว่าตัวควบคุมชนิดอื่น ๆ ดังรูปที่ 4.20 อีกทั้งยังสามารถรองรับปัจจัยแวดล้อมที่เกิดขึ้นระหว่างการควบคุมไม่ว่าจะเป็นสิ่งรบกวนจากภายนอกดังรูปที่ 4.22 หรือสิ่งรบกวนดังรูปที่ 4.24 ด้วยเหตุนี้จึงสามารถสรุปงานวิจัยครั้งนี้ได้ว่าตัวควบคุม NNPID-AC มีประสิทธิภาพในการใช้งานที่เหนือกว่าตัวควบคุม PID แบบดั่งเดิมหรือตัวควบคุม NNPID ที่ใช้อัลกอริทึมในงานวิจัยก่อนหน้านี้ และตัวควบคุมแบบโครงข่ายประสาทเทียมตัวอื่น ๆ (ได้แก่ ENNPID, Hybrid CKF-NNPID และ ABF-NNPID) พร้อมกันนี้การทดลองนี้ได้แสดงผลในรูปของค่าผิดพลาดของระบบเพื่อให้สามารถอธิบายในส่วนของความสามารถในการลู่เข้าสู่สถานะเสถียรได้ง่ายดังรูปที่ 4.21 รูปที่ 4.23 และรูปที่ 4.25 ซึ่งเป็นผลการทดลองของระบบควบคุมที่อยู่ภายใต้สถานการณ์ปกติ ผลการทดลองที่อยู่ภายใต้การใส่โหลดสูงสุด และผลการทดลองที่อยู่ภายใต้สิ่งรบกวนตามลำดับ

จากรูปของการทดลองที่อยู่ภายใต้สถานการณ์การใส่โหลดสูงสุดของระบบแขนกลไฟฟ้าในส่วนของข้อต่อที่ 3 และที่ 4 ชี้ให้ถึงประสิทธิภาพของการปรับตัวของตัวควบคุมที่นำเสนอได้อย่างชัดเจน

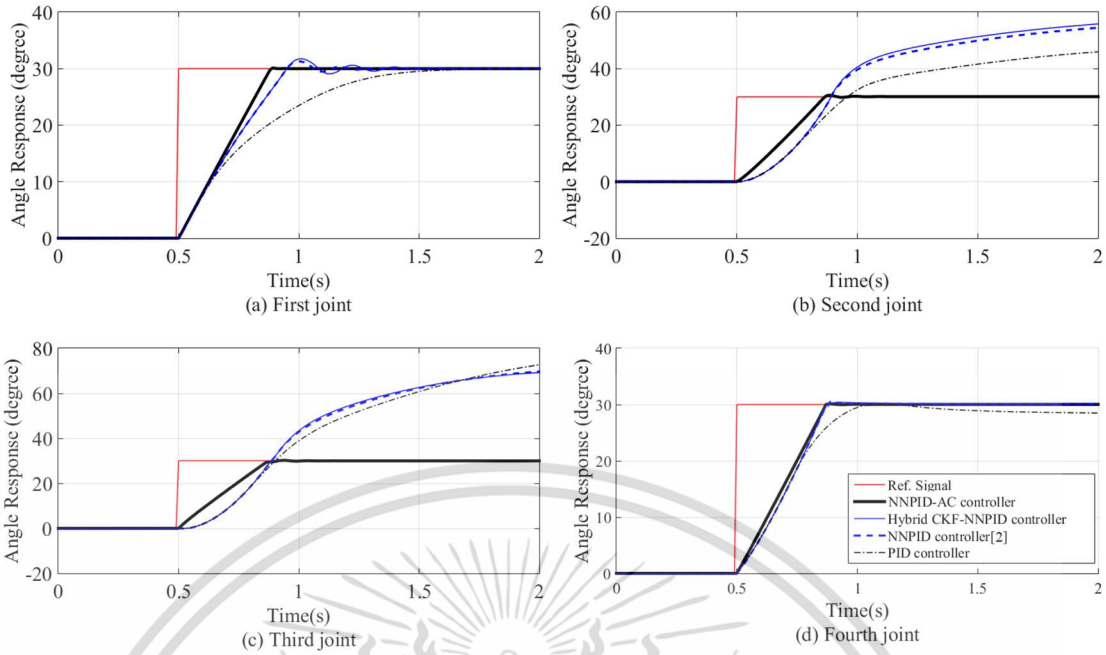
เนื่องจากต้องรับภาระของน้ำหนักในการขับเคลื่อนหรือต้องใช้แรงบิดมากกว่าข้อต่ออื่น ๆ ทำให้ผลการทดลองที่ออกมาความแตกต่างอย่างชัดเจน



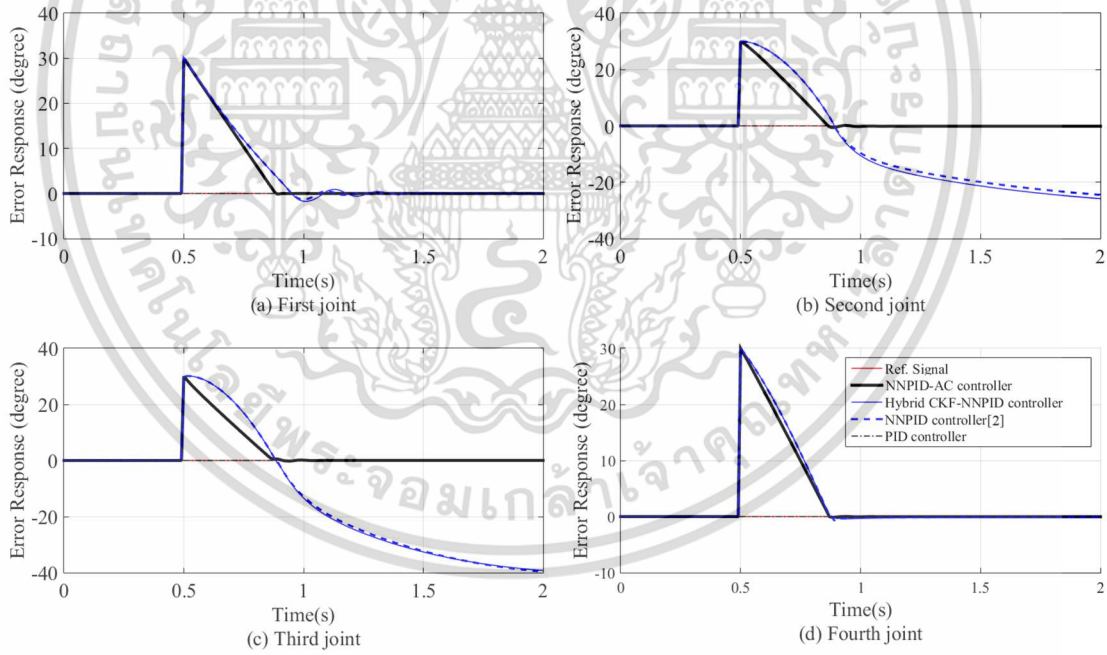
รูปที่ 4.20 ผลตอบสนองของมุมแขนกลไฟฟ้าภายใต้ตัวควบคุมแบบต่าง ๆ เมื่อไม่มีภาระโหลด (Load) โดยที่ (a) ข้อที่ 1 (b) ข้อที่ 2 (c) ข้อที่ 3 (d) ข้อที่ 4



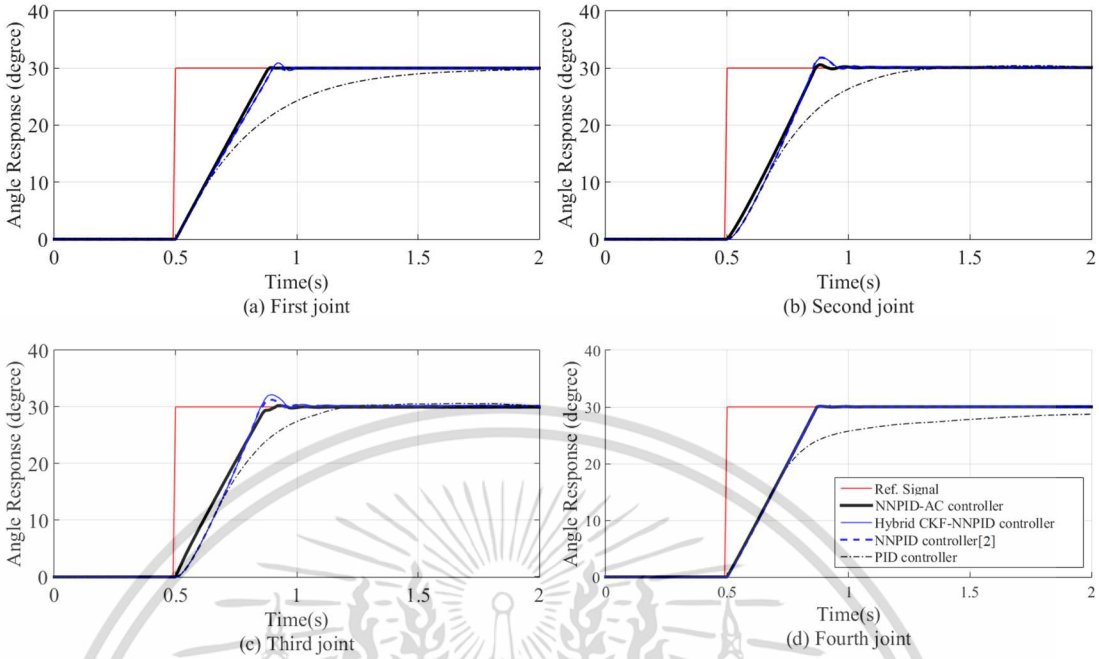
รูปที่ 4.21 ผลตอบสนองค่าผิดพลาดของมุมแขนกลไฟฟ้าภายใต้ตัวควบคุมแบบต่าง ๆ เมื่อไม่มีภาระโหลด (load) โดยที่ (a) ข้อที่ 1 (b) ข้อที่ 2 (c) ข้อที่ 3 (d) ข้อที่ 4



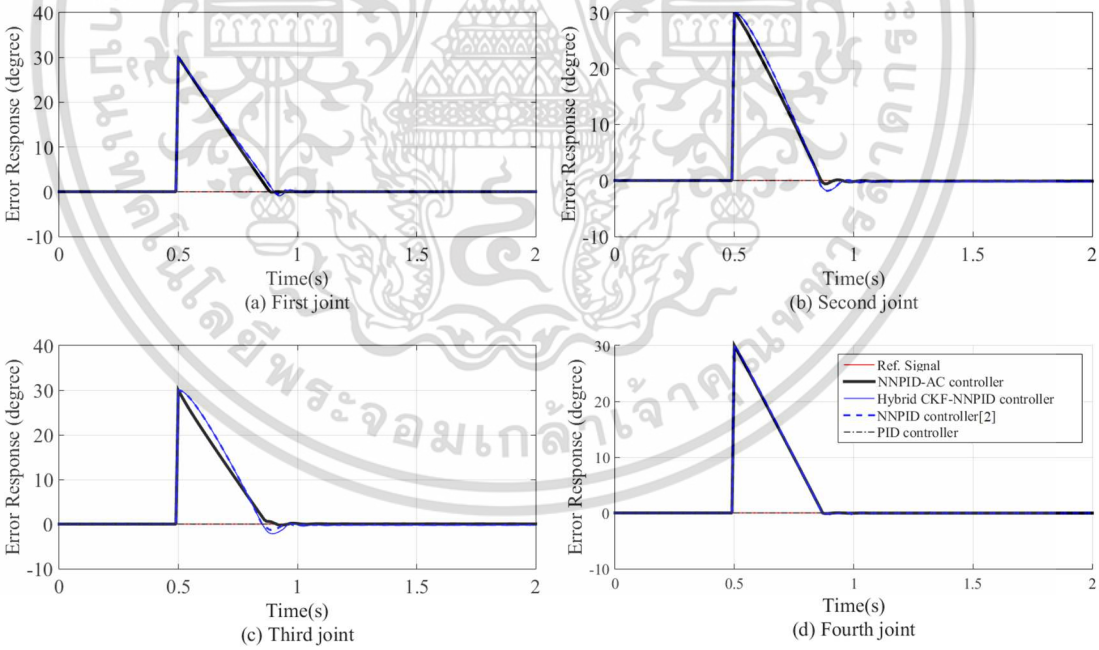
รูปที่ 4.22 ผลตอบสนองของมุมแขนกลไฟฟ้าภายใต้ตัวควบคุมแบบต่าง ๆ ภาระโหลดสูงสุด (Maximum load) โดยที่ (a) ข้อที่ 1 (b) ข้อที่ 2 (c) ข้อที่ 3 (d) ข้อที่ 4



รูปที่ 4.23 ผลตอบสนองค่าผิดพลาดของมุมแขนกลไฟฟ้าภายใต้ตัวควบคุมแบบต่าง ๆ เมื่อใส่ภาระโหลดสูงสุด (Maximum load) โดยที่ (a) ข้อที่ 1 (b) ข้อที่ 2 (c) ข้อที่ 3 (d) ข้อที่ 4



รูปที่ 4.24 ผลตอบสนองของมุมแขนกลไฟฟ้าภายใต้ตัวควบคุมแบบต่าง ๆ เมื่อใส่สิ่งรบกวน (Noise) โดยที่ (a) ข้อที่ 1 (b) ข้อที่ 2 (c) ข้อที่ 3 (d) ข้อที่ 4



รูปที่ 4.25 ผลตอบสนองค่าผิดพลาดของมุมแขนกลไฟฟ้าภายใต้ตัวควบคุมแบบต่าง ๆ เมื่อใส่สิ่งรบกวน (Noise) โดยที่ (a) ข้อที่ 1 (b) ข้อที่ 2 (c) ข้อที่ 3 (d) ข้อที่ 4

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

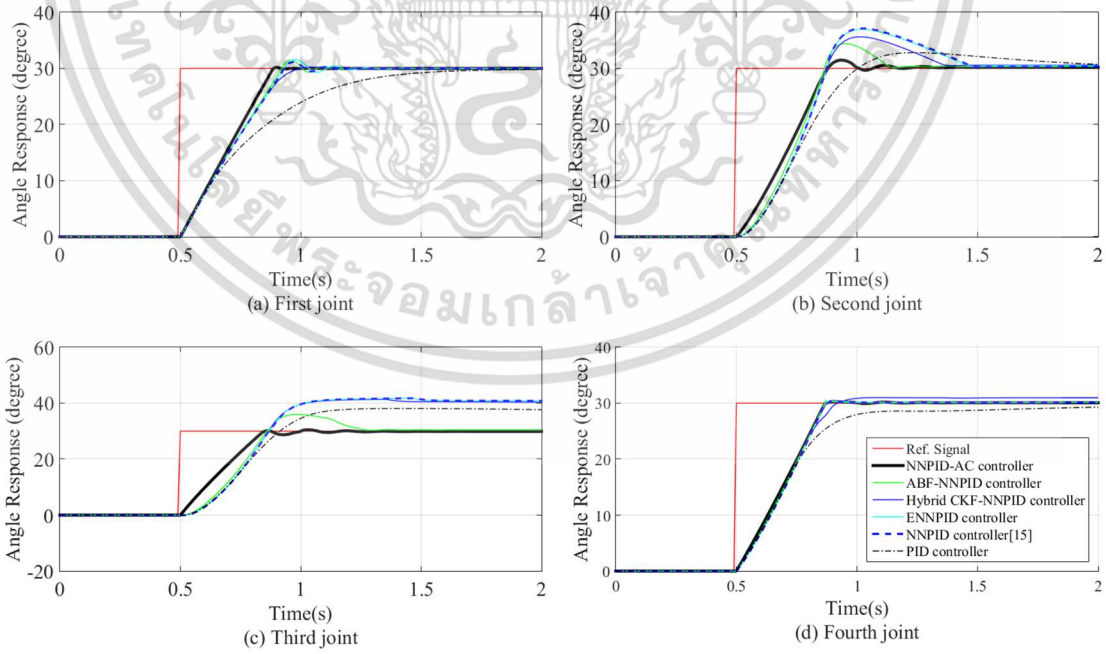
5.1 สรุปผลการทดลอง

วิทยานิพนธ์ฉบับนี้มุ่งเน้นการศึกษาเรื่องตัวควบคุม (Controller) ที่มีความสามารถเรียนรู้ปัจจัยแวดล้อมที่เปลี่ยนแปลงได้ด้วยตัวมันเอง โดยตัวควบคุมนี้สร้างขึ้นจากโครงข่ายประสาทเทียม (Neural Network) ที่มีโครงสร้างตามรูปแบบของตัวควบคุม PID เรียกว่า “ตัวควบคุม NNPID” ทำให้สามารถกำหนดค่าของอินพุตควบคุม (Control Input) สำหรับป้อนให้กับระบบด้วยการใช้อัลกอริทึมของการปรับค่าน้ำหนักเรียกว่า “อัลกอริทึมการเรียนรู้ (Learning Algorithm)” เพราะฉะนั้นการสร้างอัลกอริทึมการเรียนรู้ของตัวควบคุมจึงเป็นส่วนสำคัญต่อประสิทธิภาพของระบบควบคุม ดังนั้นโครงสร้างของตัวควบคุมที่เหมาะสมและความสามารถของอัลกอริทึมการเรียนรู้ของตัวควบคุมในเรื่องการปรับค่า Gain ให้เหมาะสมต่อสภาพแวดล้อมทำให้ได้ค่าน้ำหนักที่เหมาะสมสำหรับนำไปใช้ในการควบคุมเพื่อให้ได้ประสิทธิภาพสูงที่สุดและดีที่สุด

โดยในงานวิจัยนี้ได้พัฒนาตัวควบคุมเพื่อรองรับปัจจัยข้างต้นทั้งหมดดังนี้ 1) ตัวควบคุม ENNPID ที่ใช้อัลกอริทึมการเรียนรู้แบบส่วนขยายของตัวกรองคาลมาน (mEKF learning algorithm) 2) ตัวควบคุม Hybrid CKF-NNPID ที่ใช้อัลกอริทึมการเรียนรู้แบบไฮบริดของตัวกรองคาลมาน (Hybrid Cubature Kalman filter learning algorithm) 3) ตัวควบคุม ABF-NNPID ที่ใช้อัลกอริทึมการเรียนรู้เบย์ประยุกต์ (applied Bayesian filter learning algorithm) และ 4) ตัวควบคุม NNPID-AC ที่ใช้อัลกอริทึมการเรียนรู้ปฏิบัติ-วิจารณ์แบบเสริมกำลัง (Actor-critic reinforcement learning algorithm พร้อมกันนี้ได้นำตัวควบคุม PID แบบมาตรฐานทั่วไป และตัวควบคุมของงานวิจัยอื่น ๆ เพื่อเปรียบเทียบประสิทธิภาพอีกด้วย และเพื่อประเมินความมีประสิทธิภาพของตัวควบคุมของงานวิจัยที่ได้นำเสนอนี้จึงสร้างแบบจำลองของระบบควบคุมแขนกลไฟฟ้าขึ้นมา ซึ่งเป็นแขนกลที่ประกอบด้วย 4 ข้อต่อที่ควบคุมด้วยระบบมอเตอร์ไฟฟ้า และนำตัวควบคุมมาติดตั้งที่ตัวมอเตอร์สำหรับควบคุมองศาของการหมุนเพื่อให้เป็นไปตามเป้าหมายที่กำหนด และเพื่อให้ได้ตัวควบคุมที่สามารถรับมือกับปัจจัยที่มีผลกระทบต่อระบบควบคุม โดยแบบจำลองนี้ได้สร้างจากโปรแกรม MATLAB ซึ่งประกอบด้วย 3 ส่วนได้แก่ 1) ส่วนของตัวควบคุม (สำหรับผลิตตัวป้อน, Control input) 2) ส่วนของอัลกอริทึมการเรียนรู้ (สำหรับวิเคราะห์และกำหนดค่าตัว Control Input ที่เหมาะสม และ 3) ส่วนของแขนกล 4 ข้อต่อ โดยการสร้างส่วนตัวควบคุมและอัลกอริทึมการเรียนรู้จะใช้ Block Function สำหรับเขียนโค้ดโปรแกรม และการสร้างส่วนของแขนกลคือการนำแบบจำลองแขนกลที่อยู่

ที่มีในโปรแกรมไลบารี่ของ MATLAB โดยได้ทำการปรับปรุงค่าพารามิเตอร์บางส่วนของแขนกลตามค่าจริงที่ใช้

จากผลการทดลองที่แสดงในหัวข้อดังที่ได้กล่าวไว้ในบทที่ 4 ชี้ให้เห็นว่าตัวควบคุม ENNPID มีความสามารถที่เหนือกว่าตัวควบคุม NNPID ทั่วไปและตัวควบคุม PID โดยเป็นตัวควบคุมที่ได้นำเสนอในปี 2016 [44] ซึ่งได้กล่าวไว้ในหัวที่ 4.1 ของบทที่ 4 จากนั้นในปีปลาย 2016 ก็พยายามพัฒนาตัวควบคุมเพื่อขจัดปัญหาของตัวควบคุม ENNPID ด้วยการนำเสนอตัวควบคุมที่มีชื่อว่า Hybrid CKF-NNPID [45] ซึ่งมีประสิทธิภาพในการใช้งานที่เหนือกว่าตัวควบคุมที่ผ่านมาดังแสดงผลการทดลองในหัวข้อที่ 4.2 แต่กระทั้งก็ยังมีข้อเสียในเรื่องการเวลาที่ใช้การคำนวณ และพัฒนาประสิทธิภาพให้เพิ่มขึ้น จึงได้มีการนำเสนอตัวควบคุม ABF-NNPID ในปี 2017 ดังแสดงผลการทดลองในหัวข้อที่ 4.3 แต่ก็ยังไม่รับการตีพิมพ์จึงต้องพัฒนาต่อไปเพื่อจำกัดข้อเสียที่มีอยู่ ซึ่งในปี 2018 จึงได้นำเสนอตัวควบคุม NNPID-AC เพื่อลบข้อด้อยทั้งหมดของตัวควบคุมที่ผ่านมาดังแสดงผลการทดลองในหัวข้อที่ 4.4 ซึ่งผลการทดลองชี้เป็นที่ประจักษ์ว่าตัวควบคุม NNPID-AC ที่ได้ออกแบบนั้นมีความสามารถที่เหนือกว่าตัวควบคุมที่ใช้งานในปัจจุบันดังแสดงผลตอบสนองของระบบควบคุมดังรูปที่ 5.1 และแสดงถึงค่าภาวะชั่วครู่ (Transient) ที่ประกอบด้วยค่าผิดพลาด ณ จุดเสถียร เวลาที่แอมพลิจูดสูงสุด เวลาสู่สมดุล และ อัตราส่วนของค่าสูงสุดกับค่าสมดุล ซึ่งเป็นพารามิเตอร์ของผลตอบสนองแบบขั้น (Step Response) สำหรับการพิจารณาเกี่ยวกับประสิทธิภาพของตัวควบคุมดังตารางที่ 5.1 ซึ่งคำนวณมาจากรูปของผลการทดลองของรูปที่ 5.1(b)



รูปที่ 5.1 เปรียบเทียบผลตอบสนองของมุมแขนกลไฟฟ้าด้วยตัวควบคุมแบบต่าง ๆ ทั้งหมดเมื่อใส่ภาระโหลด โดยที่ (a) ข้อที่ 1 (b) ข้อที่ 2 (c) ข้อที่ 3 (d) ข้อที่ 4

ตารางที่ 5.1 เปรียบเทียบค่าผลตอบสนองของตัวควบคุมในแบบต่าง ๆ

ชนิดของตัวควบคุม	พารามิเตอร์ของผลตอบสนอง			
	ค่าผิดพลาดจุดเสถียร (องศา)	เวลาที่แอมพลิจูดสูงสุด (วินาที)	เวลาสู่สมดุล (วินาที)	อัตราส่วนของค่าสูงสุดกับค่าสมดุล (เปอร์เซ็นต์)
NNPID-AC	0.05	0.94	0.98	4.99
ABF-NNPID	0.35	0.95	1.12	13.48
Hybrid CKF-NNPID	0.58	1.02	1.33	16.84
ENNPID	0.47	1.02	1.43	23.27
NNPID	0.31	1.02	1.45	23.52
PID	0.20	1.25	1.96	8.60

จากตารางที่ 5.1 จะเห็นได้ว่าตัวควบคุม NNPID-AC มีค่าพารามิเตอร์ของผลตอบสนองแบบขั้นที่ดีที่สุดไม่ว่าเป็นค่าผิดพลาด ณ สถานะเสถียร ค่าเวลาที่เข้าสู่จุดเสถียร และอัตราส่วนของค่าสูงสุดกับค่าสมดุลที่ต่ำที่สุด โดยถึงแม้ว่าค่าพารามิเตอร์ของผลตอบสนองแบบขั้นของตัวควบคุม PID ในส่วนของค่าผิดพลาดจุดเสถียร และค่าอัตราส่วนของค่าสูงสุดกับค่าสมดุลจะดีกว่าตัวควบคุมแบบโครงข่ายประสาทเทียมแบบอื่น ๆ ยกเว้นตัวควบคุม NNPID-AC ก็ตาม แต่ถ้าหากพิจารณารายละเอียดของค่าในตารางจะพบว่า 1) ผิดพลาดจุดเสถียรของตัวควบคุมต่าง ๆ เป็นที่ยอมรับได้ทั้งหมด 2) หากพิจารณาค่าเวลาเข้าสู่จุดเสถียรจะเห็นได้ชัดว่าตัวควบคุม PID มีค่าที่สูงมากโดยเฉพาะอย่างยิ่งผลตอบสนองของระบบในส่วนข้อต่อที่ 3 พบว่าตัวควบคุมนี้ไม่สามารถปรับให้ผลตอบสนองเข้าสู่จุดเสถียรที่ดีได้ และสุดท้ายหากพิจารณาถึงผลการทดลองให้หัวข้อที่ 4 จะพบว่าตัวควบคุม PID มีประสิทธิภาพที่ด้อยกว่าตัวควบคุมแบบโครงข่ายประสาทเทียมแทบทุกด้าน

5.2 ข้อเสนอแนะและแนวทางการพัฒนา

จากผลการทดลองสามารถพิสูจน์ได้อย่างชัดเจนว่าตัวควบคุมนำเสนอนี้ สามารถปรับปรุงค่าภาวะชั่วครู่ ของระบบควบคุมด้วยการปรับปรุงอัลกอริทึมสำหรับกาเรียนรู้ดังที่ได้กล่าวมาในหัวข้อที่ผ่านมา แต่อย่างไรก็ตาม งานวิทยานิพนธ์นี้จัดทำขึ้นโดยการใช้แบบจำลองของโปรแกรม MATLAB ซึ่งสามารถเขียนโปรแกรมเพื่อสร้างอัลกอริทึมของการเรียนรู้ของตัวควบคุมแบบ NNPID ได้อย่างอิสระโดยไม่มีข้อจำกัดด้านการกำหนดตัวแปรต่าง ๆ ดังนั้นในการนำงานวิจัยนี้ไปพัฒนาต่อในเชิงปฏิบัติควรต้องกำหนดเงื่อนไขของการเขียนโปรแกรมในเรื่องของจุดนิยมของตัวแปร การดำเนินการทางคณิตศาสตร์ การใช้งานฟังก์ชันต่าง เป็นต้น นอกเหนือจากนี้แล้ว เพื่อเป็นการต่อยอดงานวิจัยนี้จึงเห็นว่าน่าจะนำไปใช้กับระบบควบคุมอื่น ๆ เพื่อพิสูจน์ความหลากหลายในการใช้งานต่อไป

เอกสารอ้างอิง

[1] Ren T.J., Chen T.C., and Chen C.J. “Motion control for a two-wheeled vehicle using a self-tuning PID controller” **Control Engineering Practice**, vol. 16, no.3, 2008.

[2] S. Cong and Y. Liang, "PID-Like Neural Network Nonlinear Adaptive Control forUncertain Multivariable Motion Control Systems," in **IEEE Transactions on Industrial Electronics**, vol. 56, no. 10, pp. 3872-3879, Oct. 2009.

[3] H. Huy Anh, "Online tuning gain scheduling MIMO neural PID control of the 2-axes pneumatic artificial muscle (PAM) robot arm", **Expert Systems with Applications**, vol. 37, no. 9, pp. 6547-6560, 2010.

[4] W. Yu and J. Rosen, "Neural PID Control of Robot Manipulators With Application to an Upper Limb Exoskeleton," in **IEEE Transactions on Cybernetics**, vol. 43, no. 2, pp. 673-684, April 2013.

[5] V. Kumar, P. Gaur and A. Mittal, "ANN based self tuned PID like adaptive controller design for high performance PMSM position control", **Expert Systems with Applications**, vol. 41, no. 17, pp. 7995-8002, 2014.

[6] F. Rossomando and C. Soria, "Design and Implementation of Adaptive Neural PID for Non Linear Dynamics in Mobile Robots," in **IEEE Latin America Transactions**, vol. 13, no. 4, pp. 913-918, April 2015.

[7] K.H. Ang, G. Chong, and Y. Li, “PID control system analysis, design, and technology”, **IEEE Transactions on Control Systems Technology**, Vol. 13, No. 4, pp. 559-576, 2005.

[8] D. Rupp, and L. Guzzella, “Adaptive internal model control with application to fueling control”, **Control Engineering Practice**, Vol. 18, No. 8, pp. 873-881, 2010.

[9] C. C. Hang, K. J. Astrom, and W. K. Ho, “Refinements of the Ziegler-Nichols tuning formula”, **IEE Proceedings D - Control Theory and Applications**, Vol. 138, No. 2, pp. 111-118, 1991.

[10] M.N. Anwar, and S. Pan, “A frequency response model matching method for PID controller design for processes with dead-time”, **ISA Transactions**, Vol. 55, pp. 175-187, 2015.

[11] C. Jiang, Y. Ma and C. Wang, "PID controller parameters optimization of hydro-turbine governing systems using deterministic-chaotic-mutation evolutionary programming (DCMEP)", **Energy Conversion and Management**, Vol. 47, No. 9-10, pp. 1222-1230, 2006.

[12] A. B. Sharkawy, “Genetic fuzzy self-tuning PID controllers for antilock braking systems”, **Engineering Applications of Artificial Intelligence**, Vol. 23, No. 7, pp. 1041-1052, 2010.

[13] J. L. Meza, R. Soto and J. Arriaga, "An Optimal Fuzzy Self-Tuning PID Controller for Robot Manipulators via Genetic Algorithm," **2009 Eighth Mexican International Conference on Artificial Intelligence**, Guanajuato, 2009, pp. 21-26.

[14] J J. L. Meza, V. Santibanez, R. Soto and M. A. Llama, "Fuzzy Self-Tuning PID Semiglobal Regulator for Robot Manipulators," in **IEEE Transactions on Industrial Electronics**, vol. 59, no. 6, pp. 2709-2717, June 2012.

[15] B. B. Alagoz, A. Ates, and C. Yeroglu, “Auto-tuning of PID controller according to fractional-order reference model approximation for DC rotor control”, **Mechatronics**, Vol. 23, No. 7, pp. 789-797, 2013.

[16] H. X. Li, L. Zhang, K. Y. Cai, and G. Chen, “An improved robust fuzzy-PID controller with optimal fuzzy reasoning”, **IEEE Transactions on Systems, Man, and Cybernetics***, Part B (Cybernetics)*, Vol. 35, No. 6, pp. 1283-1294, 2005.

[17] A. I. Dounis, P. Kofinas, C. Alafodimos, and D. Tseles, “Adaptive fuzzy gain scheduling PID controller for maximum power point tracking of photovoltaic system”, **Renewable Energy**, Vol. 60, pp. 202-214, 2013.

[18] J. Armendariz, V. Parra-Vega, R. García-Rodríguez and S. Rosales, "Neuro-fuzzy self-tuning of PID control for semiglobal exponential tracking of robot arms", **Applied Soft Computing**, vol. 25, pp. 139-148, 2014

[19] A. Bouguerra, D. Saigaa, K. Kara, and S. Zeghlache, "Fault-Tolerant Lyapunov-Gain-Scheduled PID Control of a Quadrotor UAV", **International Journal of Intelligent Engineering and Systems**, Vol. 8, No. 2, pp. 1-6, 2015.

[20] A. Idir, M. Kidouche, Y. Bensafia, K. Khettab, and S. Tadjer, "Speed Control of DC Motor Using PID and FOPID Controllers Based on Differential Evolution and PSO", **International Journal of Intelligent Engineering and Systems**, Vol. 11, No. 4, pp. 241–249, 2018.

[21] A. Rubaai, and P. Young, "EKF-Based PI-/PD-Like Fuzzy-Neural-Network Controller for Brushless Drives", **IEEE Transactions on Industry Applications**, Vol. 47, No. 6, pp. 2391-2401, 2011.

[22] Jun Kang, Wenjun Meng, Ajith Abraham, Hongbo Liu, An adaptive PID neural network for complex nonlinear system control, ***Neurocomputing***, Vol. 135, 2014, pp. 79-85.

[23] Richa S., Vikas K., Prerna G., Mittal A.P., "An adaptive PID like controller using mix locally recurrent neural network for robotic manipulator with variable payload" **ISA Transactions**, vol. 62, no., 2016. pp. 258-267

[24] P.S. Londhe, S. Mohan, B.M. Patre, L.M. Waghmare, Robust task-space control of an autonomous underwater vehicle-manipulator system by PID-like fuzzy control scheme with disturbance estimator**, Ocean Engineering**, Vol. 139, 2017, pp. 1-13.

[25] Chuanjiang Li, Kok Lay Teo, Bin Li, Guangfu Ma, A constrained optimal PID-like controller design for spacecraft attitude stabilization, **Acta Astronautica**, Vol. 74, 2012, pp. 131-140.

[26] S. A. Merritt, "The iterated extended Kalman phase detector," in **IEEE Transactions on Communications**, vol. 37, no. 5, pp. 522-526, May 1989.

[27] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," **Proc. IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)**, Lake Louise, Alberta, Canada, 2000, pp. 153-158.

[28] J. Zhao, M. Netto and L. Mili, "A Robust Iterated Extended Kalman Filter for Power System Dynamic State Estimation," **in IEEE Transactions on Power Systems**, vol. 32, no. 4, pp. 3205-3216, July 2017.

[29] I. Arasaratnam, and S. Haykin, "Cubature Kalman Filters*",* **IEEE Transactions on Automatic Control**, Vol. 54, No. 6, pp. 1254-1269, 2009.

[30] I. Arasaratnam, S. Haykin and T. R. Hurd, "Cubature Kalman Filtering for Continuous-Discrete Systems: Theory and Simulations," **in IEEE Transactions on Signal Processing**, vol. 58, no. 10, pp. 4977-4993, Oct. 2010.

[31] C. Sheng, J. Zhao, Y. Liu and W. Wang, "Prediction for noisy nonlinear time series by echo state network based on dual estimation", **Neurocomputing**, vol. 82, pp. 186-195, 2012.

[32] Y. Chen, Q. Zhao, Z. An, P. Lv, and L. Zhao, "Distributed Multi-Target Tracking Based on the K-MTSCF Algorithm in Camera Networks", **IEEE Sensors Journal**, Vol. 16, No. 13, pp. 5481-5490, July, 2016.

[33] B. Jia, and M. Xin, "Adaptive cubature Kalman filter with directional uncertainties [Correspondence]", **IEEE Transactions on Aerospace and Electronic Systems**, Vol. 52, No. 3, pp. 1477-1486, 2016.

[34] Ienkaran A. and Simon H. "Nonlinear Bayesian Filters for Training Recurrent Neural Networks" **Springer-Verlag Berlin Heidelberg**, vol. ,no, 2008. pp.12-33.

[35] R. Sameni, M. B. Shamsollahi, C. Jutten and G. D. Clifford, "A Nonlinear Bayesian Filtering Framework for ECG Denoising," **in IEEE Transactions on Biomedical Engineering**, vol. 54, no. 12, pp. 2172-2185, Dec. 2007.

[36] V. Smidl and A. Quinn, "Variational Bayesian Filtering," **in IEEE Transactions on Signal Processing**, vol. 56, no. 10, pp. 5020-5030, Oct. 2008.

[37] A. K. Singh, P. Date and S. Bhaumik, "A Modified Bayesian Filter for Randomly Delayed Measurements," **in IEEE Transactions on Automatic Control**, vol. 62, no. 1, pp. 419-424, Jan. 2017.

[38] R. E. Bellman, **Dynamic Programming**, Princeton University Press, Princeton, NJ. Republished, 2003.

[39] H. Modares, F. L. Lewis, and M. Naghibi-Sistani, "Adaptive Optimal Control of Unknown Constrained-Input Systems Using Policy Iteration and Neural Networks", **IEEE Transactions on Neural Networks and Learning Systems**, Vol. 24, No. 10, pp. 1513-1525, 2013.

[40] B. Kiumarsi, and F. L. Lewis, "Actor–Critic-Based Optimal Tracking for Partially Unknown Nonlinear Discrete-Time Systems", **IEEE Transactions on Neural Networks and Learning Systems,** Vol. 26, No. 1, pp. 140-151, 2015.

[41] J. Škach, B. Kiumarsi, F. L. Lewis, and O. Straka, "Actor-Critic Off-Policy Learning for Optimal Control of Multiple-Model Discrete-Time Systems", **IEEE Transactions on Cybernetics**, Vol. 48, No. 1, pp. 29-40, 2018.

[42] N. Bergman, **Recursive Bayesian Estimation Navigation and Tracking Applications**, Department of Electrical Engineering, Link¨oping University, Link¨oping, Sweden, 1999.

[43] Kolmogorov AN. Sur linterpolation et extrapolation des suite stationaries. **C R Math** 1939; 208: 2043-2045.

[44] A. Sento, and Y. Kitjaidure, "Neural network controller based on PID using an extended Kalman filter algorithm for multi-variable non-linear control system", **2016 Eighth International Conf. on Advanced Computational Intelligence**, pp. 302-309, 2016.

[45] A. Sento, and Y. Kitjaidure, "A hybrid CKF-NNPID controller for MIMO nonlinear control system", **ECTI Transactions on Computer and Information Technology**, Vol. 10, No. 2, pp. 176-184, 2016.

[46] A. Sento, and Y. Kitjaidure, "A Neural Network PID-Like Controller Using a Hybrid of Online Actor-Critic Reinforcement Algorithm with the Square Root Cubature Kalman Filter", **The International Journal of Intelligent Engineering and Systems**, Vol. 11, No. 6, pp. 261-270, 2018..

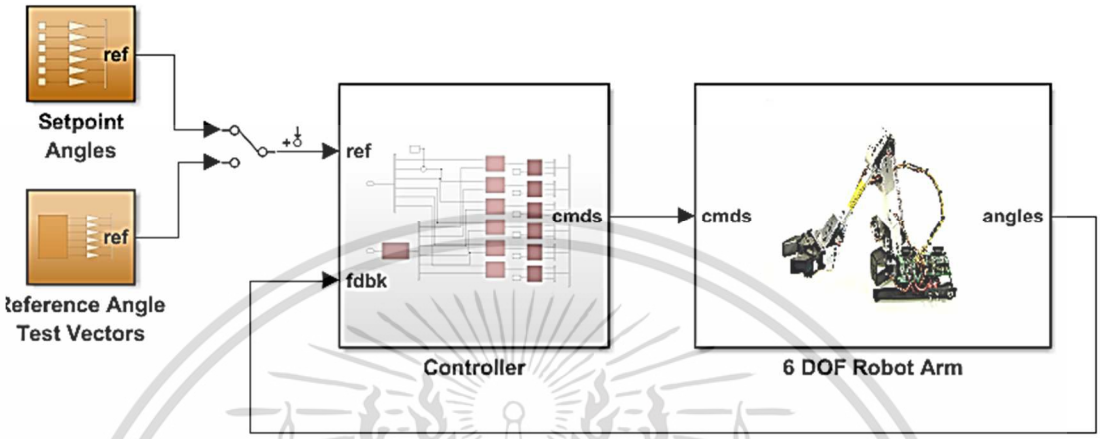
ภาคผนวก

ภาคผนวก ก.

การสร้างแบบจำลองระบบควบคุมด้วยโปรแกรม MATLAB/SIMULINK และการปรับค่าคงที่แบบอัตโนมัติของตัวควบคุม PID ในโปรแกรม MATLAB/SIMULINK

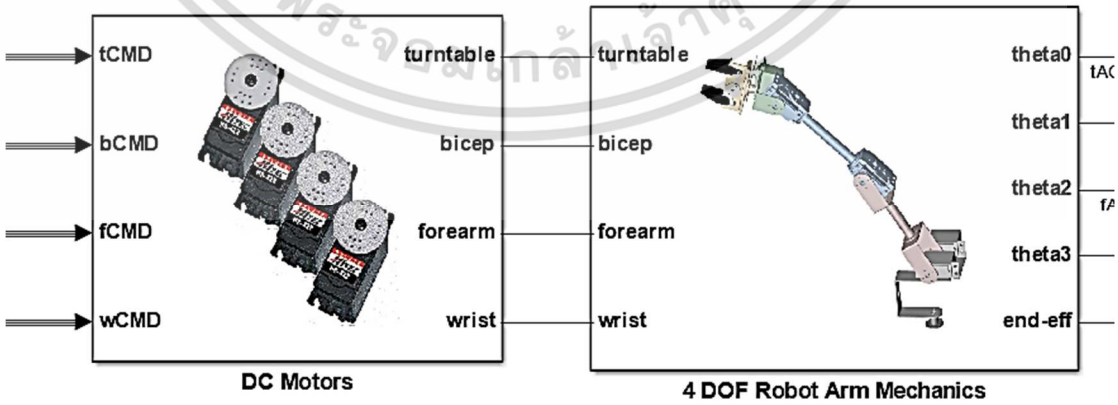
การสร้างแบบจำลองระบบควบคุมด้วยโปรแกรม MATLAB/SIMULINK

ไดอะแกรมของจำลองระบบควบคุมของแขนกลไฟฟ้า และตัวควบคุม NNPID ซึ่งได้พัฒนาจากไลบารี่ของโปรแกรม MATLAB ที่มีชื่อว่า Robust Control ดังรูปที่ ก.1 ซึ่งมีความสามารถสร้างแบบจำลองได้เสมือนมีแขนกลจริงมาใช้งาน



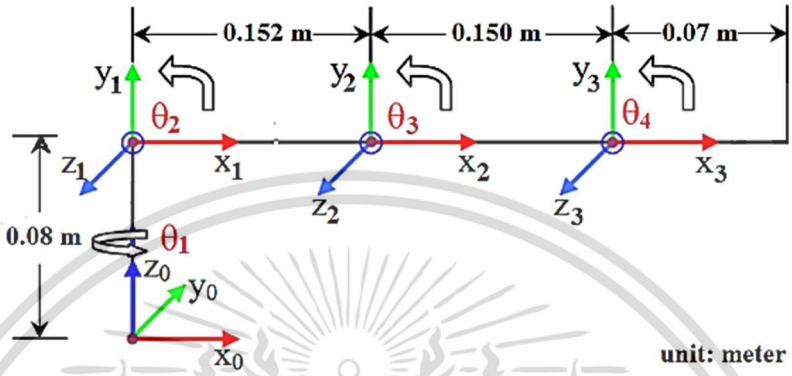
รูปที่ ก.1 แบบจำลองของระบบควบคุมแขนกลไฟฟ้าพัฒนาจากโปรแกรม MATLAB

ส่วนแรกจะขอกล่าวถึงการสร้างแบบจำลองของระบบควบคุมแขนกลไฟฟ้าโดยแขนกลไฟฟ้าประกอบด้วย 4 ข้อได้แก่ ส่วนของฐาน (Turntable) ข้อที่ 2 ส่วนของข้อต่อหัวไหล่ (Bicep) ข้อที่ 3 ส่วนข้อต่อแขน (Forearm) และข้อที่ 4 ส่วนข้อของข้อมือ (Wrist) ในบล็อกของ PID ของโปรแกรม MATLAB/SIMULINK โดยภายในบล็อก MATLAB/SIMULINK ของแขนกลไฟฟ้าได้ออกแบบจากแขนกลมอเตอร์ไฟฟ้า 4 ข้อต่อจริงด้วยการใช้ Toolbox ของโปรแกรม MATLAB/SIMULINK มีลักษณะดังรูปที่ ก.2 ซึ่งมีมอเตอร์ไฟฟ้าเป็นตัวขับเคลื่อนแขนกลโดยในแบบจำลองนี้สามารถตั้งค่าของพารามิเตอร์ด้านไฟฟ้าต่าง ๆ หรือพารามิเตอร์ทางด้านกายภาพให้สอดคล้องกับทางปฏิบัติได้ ดังนั้นหากต้องการนำไปประยุกต์ใช้งานกับแขนกล 4 ข้อต่ออื่นที่นอกเหนือจากที่ทางโปรแกรม MATLAB ได้



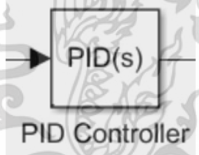
รูปที่ ก.2 แบบจำลองของระบบมอเตอร์ไฟฟ้าเชื่อมต่อกับแขนกลไฟฟ้าของโปรแกรม MATLAB

โดยโครงสร้างของแขนกลไฟฟ้าที่ประกอบด้วย 4 ข้อต่อมีโครงสร้างส่วนของฐาน (Turntable) ยาว 8 เซนติเมตร ส่วนของข้อต่อหัวไหล่ (Bicep) ยาว 15.2 เซนติเมตร ส่วนข้อต่อแขน (Forearm) ยาว 15 เซนติเมตร และส่วนข้อของข้อมือ (Wrist) ยาว 7 เซนติเมตรดังรูปที่ ก.3



รูปที่ ก.3 โครงสร้างแขนกลไฟฟ้า

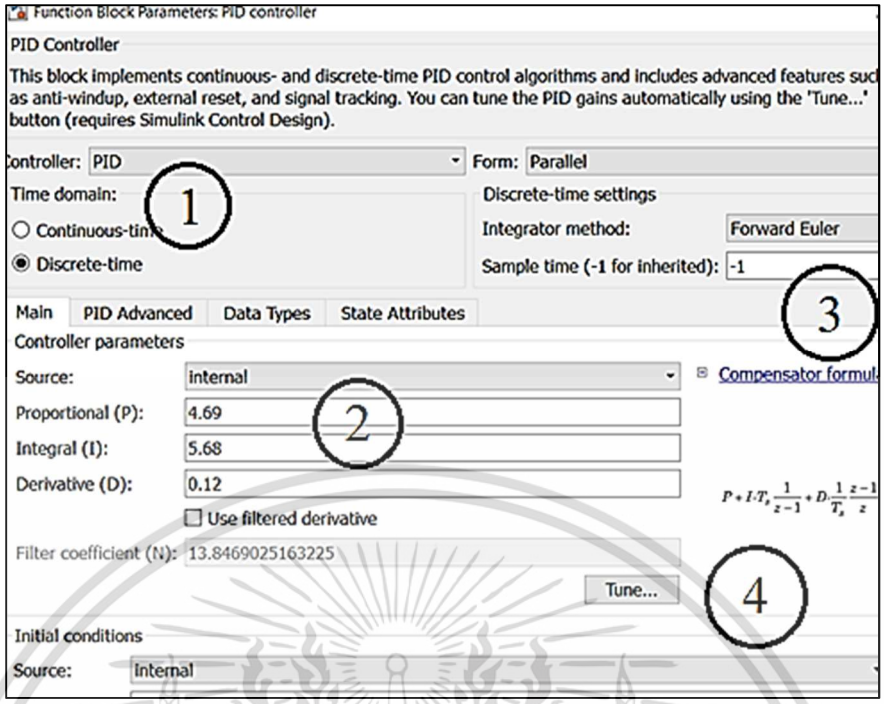
ส่วนต่อมาเป็นเรื่องของการสร้างแบบจำลองของตัวควบคุมซึ่งในงานวิจัยนี้ประกอบด้วยตัวควบคุม 2 แบบได้แก่ ตัวควบคุม PID และตัวควบคุม NNPID ซึ่งในลำดับแรกกล่าวถึงการตั้งค่าคงที่ (Gain) ของ K_D K_I และ K_D ของตัวควบคุม PID โดยวิธีการปรับค่าในโปรแกรม MATLAB โดยตัวควบคุม PID ของโปรแกรม MATAB/SIMULINK มีบล็อกไดอะแกรมดังรูป



รูปที่ ก.4 PID Simulink Block Function

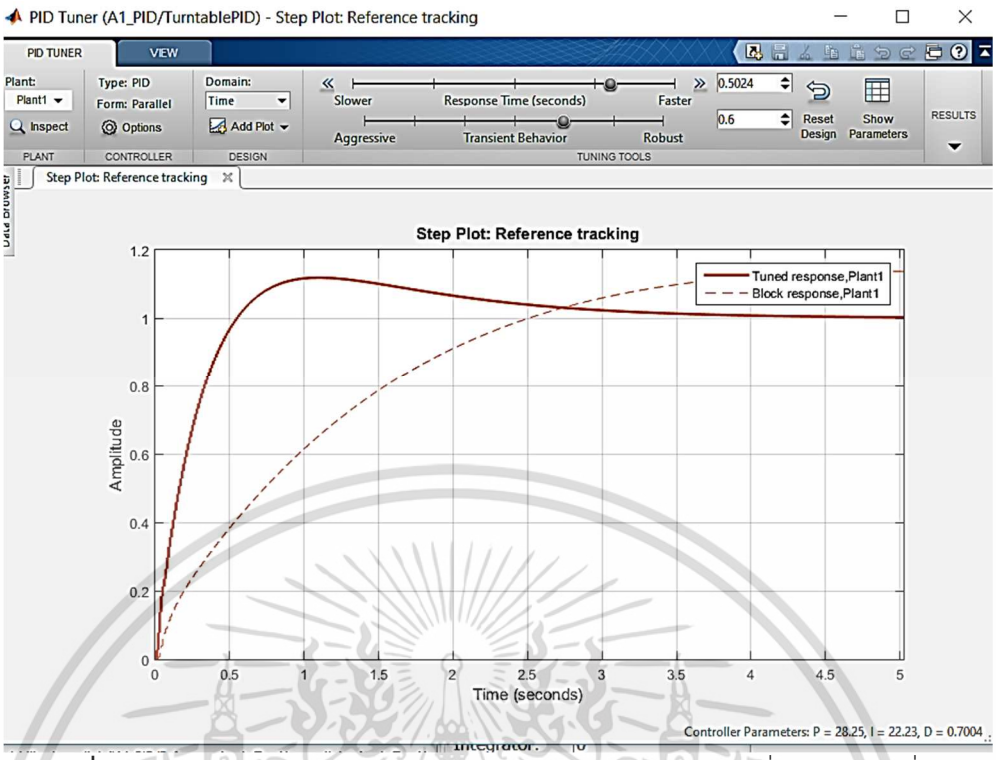
โดยเมื่อต้องการปรับค่าคงที่ (Gain) ของ K_D K_I และ K_D จะต้องคลิก 2 ครั้งที่บล็อกไดอะแกรมตัวควบคุม PIDในข้างต้นจะปรากฏหน้าต่างขึ้นมาดังรูปที่ ก.5 โดยมีส่วนประกอบสำคัญดังนี้

- หมายเลข 1 คือส่วนของการเลือกชนิดของตัวควบคุม PID ซึ่งสามารถเลือกได้ว่าต้องการให้เป็นตัวควบคุมแบบ P, PI, PD และ PID เป็นต้น โดยสามารถตั้งเป็นแบบตัวควบคุมแบบเต็มหน่วย (Discrete) และ แบบต่อเนื่อง (Continuous)
- หมายเลข 2 คือส่วนของค่าคงที่ (Gain) ของ K_D K_I และ K_D ซึ่งเป็นส่วนที่สำคัญสำหรับงานระบบควบคุมโดยสามารถปรับค่าได้ทั้งการป้อนค่าลงในช่องหรือการใช้ตัวช่วยสำหรับการปรับด้วยการใช้ปุ่ม Tune ดังหมายเลข 4 ของดังรูปที่ ก.5
- หมายเลข 3 คือส่วนของการตั้งค่าเวลาของการชักตัวอย่าง
- หมายเลข 4 คือส่วนของตัวช่วยของการปรับค่าคงที่ (Gain) ของ K_D K_I และ K_D



รูปที่ ก.5 แสดงหน้าต่างปรับค่าคงที่ (Gain) ของ K_D K_I และ K_D

จากรูปที่ ก.5 ในการปรับค่าคงที่ (Gain) ของ K_D K_I และ K_D ให้คลิกที่ปุ่ม Tune ดังหมายเลข 4 จากนั้นเมื่อคลิกแล้วจะปรากฏหน้าต่างของโปรแกรมดังรูปที่ ก.6 ซึ่งแสดงเป็นผลตอบสนองแบบขั้นหนึ่งหน่วยของระบบ เมื่อผลตอบสนองที่ไม่เป็นไปตามเป้าหมายสามารถใช้การปรับค่าด้วยการเลื่อนแทบที่อยู่ด้านบนซึ่งประกอบด้วย 2 แทบคือ แทบผลตอบสนองทางเวลา (Response time) และแทบพฤติกรรมของทราสเซียนต์ (Transient Behavior) โดยสามารถเลื่อนจากซ้ายไปขวาเพื่อเพิ่มหรือลดส่วนของผลตอบสนองซึ่งสามารถประเมินจากรูปภาพของผลตอบสนองด้านล่าง และเมื่อได้ค่าผลตอบสนองที่ต้องการแล้วให้กดปุ่ม “Update Block” แล้วจะมีผลทำให้โปรแกรม MATLAB ทำการปรับค่าคงที่ (Gain) ของ K_D K_I และ K_D โดยอัตโนมัติตามรูปของผลตอบของส่วนการปรับ



รูปที่ ก.6 แสดงหน้าต่างของผลตอบสนองของระบบและแทบเพื่อปรับค่าคงที่ (Gain)

ภาคผนวก ข.

การเขียนโค้ดสำหรับตัวควบคุม NNPID

การเขียนโค้ดสำหรับตัวควบคุม NNPID

ในส่วนแรกจะกล่าวถึงส่วนของการเขียนโค้ดสำหรับตัวควบคุม NNPID เพื่อผลิตอินพุตควบคุม (Control Input, u) ตามรายละเอียดดังรูปที่ ข.1 บล็อกไดอะแกรมของตัวควบคุมและบล็อกไดอะแกรมของอัลกอริทึมการเรียนรู้ต่าง ๆ จะทำงานอยู่ภายใต้การคำนวณแบบวนซ้ำตามแนวคิดของระบบควบคุมแบบปิด ดังนั้นโค้ดโปรแกรมสามารถดูได้ตามภาคผนวก ข. ซึ่งเป็นส่วนของโค้ดที่อยู่ภายในลูป (ฟังก์ชัน for-loop หรือ ฟังก์ชัน while-loop)

```
line 1....   delta_t = 0.01;
line 2....   x = [e_k;1];
line 3....   W(2) = h_k1(1,1);
line 4....   W(4) = 0;
line 5....   W(6) = -e_k1*W_k1(5);
line 6....   Wh = [W(1) W(2); W(3) W(4);W(5) W(6)];
line 7....   Wy = [W(7) W(8) W(9)];
line 8....   ph = Wh*x;
line 9....   h = tanh(ph);
line 10...   h(1,1) = tanh(e*W(1)) + W(2)*Ts;
line 11...   h(3,1) = tanh((e*W(5) + W(6))/Ts);
line 12...   u = Wy*[h];
```

รูปที่ ข.1 โค้ดสำหรับตัวควบคุม NNPID

กำหนดให้ W คือเมตริกของน้ำหนักของตัวโครงข่ายประสาทเทียมของตัวควบคุม NNPID ที่ประกอบด้วยสมาชิก 9 ตัว ซึ่งสมาชิก 6 ตัวแรกของเมตริกเป็นน้ำหนักและค่าไบอัสระหว่างชั้นอินพุตและชั้นซ่อน (แทนด้วย Wh) และ 3 ตัวหลังแทนด้วยน้ำหนักระหว่างชั้นซ่อนและชั้นเอาต์พุต (แทนด้วย Wy) ส่วนค่า h และ u คือค่าของเซลล์ประสาทในชั้นซ่อน และอินพุตควบคุม (Control Input, u) ตามลำดับ โดยในแต่ละบรรทัดของโค้ดสามารถอธิบายได้ดังนี้

- *บรรทัดที่ 1 และ 2* เป็นการประกาศตัวแปรของค่า Sampling Time (มีค่าเท่ากับ 0.01 วินาที) และประกาศตัวแปรสำหรับค่าอินพุตของโครงข่ายประสาทเทียมโดยมีสมาชิกในเมตริกเป็นค่าผิดพลาดของระบบ (e_k) และค่าไบอัสของโครงข่ายซึ่งกำหนดให้เป็น 1
- *บรรทัดที่ 3 - 5* เป็นการเก็บค่าตัวค่าน้ำหนักไบอัสของเซลล์ประสาทในชั้นซ่อนตามโครงสร้างของตัวควบคุม PID
- *บรรทัดที่ 6 และ 7* เป็นการกำหนดน้ำหนักและค่าไบอัสระหว่างชั้นอินพุตและชั้นซ่อน (แทนด้วย Wh) และ น้ำหนักของชั้นซ่อนกับชั้นเอาต์พุต (แทนด้วย Wy) ตามลำดับ
- *บรรทัดที่ 8 และ 9* คือการหาค่าผลรวมของผลคูณระหว่างน้ำหนักและอินพุตของเซลล์ประสาทในชั้นซ่อนและใช้ฟังก์ชัน Tanh เพื่อแปลงค่า

- *บรรทัดที่ 10 และ 11* คือคำนวณค่าเซลล์ประสาทของชั้นซ่อนตามข้อกำหนดของโครงสร้าง PID

- *บรรทัดที่ 12* คือคำนวณค่าเอาต์พุต ซึ่งในที่นี่คือการหาค่าอินพุตควบคุม (Control Input, u)

เมื่อสร้างโค้ดของโครงสร้างของตัวควบคุม NNPID เป็นที่เรียบร้อยแล้ว ต่อไปเป็นเนื้อหาในส่วนของการสร้างโค้ดสำหรับอัลกอริทึมการเรียนรู้ที่นำเสนอต่าง ๆ ซึ่งในงานวิทยานิพนธ์ฉบับนี้ได้นำเสนออัลกอริทึม 4 อัลกอริทึมดังที่ได้กล่าวไว้ในหัวข้อที่ 3 ที่ผ่านมากับอัลกอริทึมตามวิจัย [2] เพื่อเปรียบเทียบประสิทธิภาพรวมทั้งหมดเป็น 5 อัลกอริทึม ซึ่งโค้ดโปรแกรมเหล่านี้จะต้องนำมาเขียนใส่ลงใน “Simulink Block Function” อีกบล็อกดังรูปที่ 3.6 โดยมีรายละเอียดดังต่อไปนี้

a) *โค้ดสำหรับแบบจำลองของตัวควบคุม NNPID ด้วยอัลกอริทึมการเรียนรู้ Gradient Descent Learning Algorithm*

จากงานวิจัย [2] สามารถเขียนโค้ดสำหรับอัลกอริทึมการเรียนรู้ได้ดังรูปที่ ข.2

```
line 1....    err = - e;
line 2....    lr = 0.001;
line 3....    if(u-u_k1 == 0)
line 4....     delta_Wh = [0, 0, 0];
line 5....    else
line 6....     delta_Wh(1) = lr * (err*sign(y -
               last_y)/(u-last_u))*e;
line 7....     delta_Wh(2) = lr * (err*sign(y -
               last_y)/(u-last_u))*e;
line 8....     delta_Wh(3)  = lr * (err*sign(y -
               last_y)/(u-last_u))*e;
line 9....    end
line 10....   delta_W = [delta_Wh(1) W(2) delta_Wh(2)
              W(4) delta_Wh(3) W(6) 0 0 0]';
line 11....   New_W = W + delta_W;
```

รูปที่ ข.2 โค้ดสำหรับอัลกอริทึมการเรียนรู้ Gradient Descent Learning Algorithm

<u>คำอธิบายโค้ดโปรแกรม</u>

- *บรรทัดที่ 1 และ 2* เป็นการกำหนดตัวแปร `err` (ค่าผิดพลาดที่เกิดขึ้นของระบบ) และตัวแปรของอัตราการเรียนรู้ (`lr`) ของการปรับค่าน้ำหนักของโครงข่ายประสาทเทียม

- *บรรทัดที่ 3 และ 4* เป็นการเช็คค่าของการเปลี่ยนแปลงค่า Control Input (`u`) กับค่าในอดีตล่าสุด (`u_k1`) โดยมีเงื่อนไขว่าถ้าไม่มีการเปลี่ยนแปลงจะกำหนดให้การปรับค่าน้ำหนักโครงข่าย (`delta_Wh`) ของชั้นอินพุตเป็นศูนย์

- *บรรทัดที่ 5 ถึง 8* กล่าวถึงอีกเงื่อนไขถัดไปคือ แต่ถ้าค่า Control Input มีการเปลี่ยนแปลง จะกำหนดให้น้ำหนักโครงข่าย (`delta_Wh`) ของชั้นอินพุตเป็นไปตามสมการข้างต้น
- *บรรทัดที่ 9* เป็นการปิดฟังก์ชัน `if-else`
- *บรรทัดที่ 10 - 11* เป็นรวมค่าการปรับน้ำหนักโครงข่าย (`delta_Wh`) ของชั้นอินพุตและชั้นเอาต์พุตให้อยู่ในรูปแบบของเมตริก และทำการปรับค่าน้ำหนักโครงข่ายใหม่ (แทนด้วย `New_W`) เพื่อการคำนวณค่าน้ำหนักรอบต่อไป ตามลำดับ

b) โค้ดสำหรับแบบจำลองของตัวควบคุม ENNPID

โค้ดโปรแกรมสำหรับอัลกอริทึมการเรียนรู้ของตัวควบคุม ENNPID ดังรูปที่ ข.3 ซึ่งสร้างตามแนวคิดที่ได้กล่าวไว้ในหัวข้อ 3.2 ที่ผ่านมา ซึ่งค่าเริ่มต้นต่าง ๆ จะถูกกำหนดไว้ในบล็อกของตัว Delay (บล็อกในโปรแกรม SIMULINK) ได้แก่ ค่าแปรปรวนร่วมเกี่ยว (`P`) ค่าคาลมาล (`K`) ค่าฟังก์ชันต้นทุน (`J`) และค่าผิดพลาด (`R`)

```
line 1....   err = - e;
line 2....   lr = 0.001;
line 3....   Pk_k1 = diag(W')*Pk1_k1*diag(W');
line 4....   Hh1 = [W_k1(7) W_k1(8) W_k1(9)];
line 5....   Hh2 = [(1-h_k1(1,1)^2) 0 0;0 (1-
               h_k1(2,1)^2) 0;0 0 (1-
               h_k1(3,1)^2)];
line 6....   Hh3 = [e_k1 1 0 0 0 0;0 0 e_k1 0 0
               0;0 0 0 0 e_k1 1];
line 7....   Hh = (1) *Hh1*Hh2*Hh3;
line 8....   Hy = (1) *[h'];
line 9....   H = [Hh Hy];
line 10...   Rr = Rr_k1+(1/k)*((err_k1,2)^2-Rr_k1);
line 11....  K = Pk1* H'/(Pk1*H' + Rr);
line 12....  P = Pk1 - K*H*Pk1;
line 13....  New_W = W + lr*K*err;
```

รูปที่ ข.3 โค้ดสำหรับอัลกอริทึมการเรียนรู้ mEKF Algorithm

<u>คำอธิบายโค้ดโปรแกรม</u>

- *บรรทัดที่ 1 - 2* คือ กำหนดให้ `err` คือกำหนดค่าผิดพลาดที่เกิดขึ้นของระบบ และกำหนดค่าอัตราการเรียนรู้ของการปรับค่าน้ำหนักของโครงข่ายประสาทเทียม
- *บรรทัดที่ 3* คือการพยากรณ์ค่าความแปรปรวนร่วมเกี่ยว (`Pk_k1`) ด้วยค่าน้ำหนัก (`Wk1_k1`) และความแปรปรวนร่วมเกี่ยว (`Pk1_k1`) ในอดีต

- *บรรทัดที่ 4 - 6* คือ หาค่าเมตริก Jacobian ด้วยการแบ่งเป็น 3 เมตริกเพื่อความง่ายในการคำนวณ ซึ่งแต่ละเมตริกในแต่ละบรรทัดแสดงถึงค่าเมตริก Jacobian ในโครงข่ายที่เป็นชั้นเอาต์พุต (Hh1) ชั้นซ่อน (Hh2) และชั้นอินพุต (Hh3) ตามลำดับ
- *บรรทัดที่ 7 - 9* เป็นการรวมเมตริก Jacobian ในแต่ละชั้นให้เป็นเมตริก H
- *บรรทัดที่ 10 - 13* เป็นการหาความแปรปรวนของความผิดพลาด (แทนด้วย Rr) ค่าคาลมาล (K) และค่าค่าความแปรปรวนร่วมเกี่ยว (P) ตามลำดับ
- *บรรทัดที่ 13* เป็นการปรับค่าน้ำหนักใหม่ (New_W) เพื่อการคำนวณค่าน้ำหนักรอบต่อไป

c) โค้ดสำหรับแบบจำลองของตัวควบคุม Hybrid CKF-NNPID"

โค้ดโปรแกรมสำหรับอัลกอริทึมการเรียนรู้ของตัวควบคุมที่นำเสนอนี้ดังรูปที่ ข.4 ซึ่งสร้างตามแนวคิดที่ได้กล่าวไว้ในหัวข้อที่ผ่านมา ซึ่งค่าเริ่มต้นต่าง ๆ จะถูกกำหนดไว้ในบล็อกของตัว Delay (บล็อกในโปรแกรม SIMULINK) ได้แก่ ค่าแปรปรวนร่วมเกี่ยว (P) ค่าคาลมาล (K) ค่าฟังก์ชันต้นทุน (J) จำนวนลูปภายใน (inner) และ แปรปรวนร่วมเกี่ยวของค่าผิดพลาด (R)

```
line 1....  While (inner <= 20)
line 2....  err = - e,lr = 0.001;
line 3....  n = length(W);
line 4....  Ie = eye(n);
line 5....  [Q,Sk1_k1]=qr(Pk1_k1');
line 6....  for i3=1:2*n
line 7....   if(i3<n+1)
line 8....     damp(:,1,i3) = sqrt(n)*Ie(:,i3);
line 9....   else
line 10....    damp(:,1,i3) = -sqrt(n)*Ie(:,i3-n);
line 11....  end
line 12....    Wk1_k1 = Sk1_k1'*damp(:,1,i3) + W;
line 13....    dif_u = err*(1);
line 14....    dif_Wy = lr*dif_u*h;
line 15....    dif_h = (1-tanh(ph)^2)*Wy*dif_u;
line 16....    dif_Wh = lr*dif_h*e;
line 17....    dif_W = [dif_Wh(1) W(2) dif_Wh(3)
               W(4) dif_Wh(5) W(6) dif_Wy]';
line 18....    Wk_k1_star = Wk1_k1 + dif_W;
line 19....    wk_k1 = wk_k1 + Wk_k1_star;
line 20....    SWk_k1_star = SWk_k1_star + Wk_k1_star
               *Wk_k1_star';
line 21.... end
line 22.... wk_k1_dat = 1/(2*n)*(wk_k1);
line 23.... Pk_k1 = 1/(2*n)*(SWk_k1_star)'-
            (wk_k1_dat)*(wk_k1_dat)';
line 24.... [Q,Sk_k1]=qr(Pk_k1);
line 25.... for i4=1:2*n
```

```
line 26....  if(i4<n+1)
line 27....    damp(:,1,i4) = sqrt(n)*Ie(:,i4);
line 28....  else
line 29....    damp(:,1,i4) = -sqrt(n)*Ie(:,i4-n);
line 30....  end
line 31....  Wk_k1 = Sk_k1'*damp(:,1,i4) +
             wk_k1_dat;
line 32....  Wh = [Wk_k1(1) Wk_k1(2); Wk_k1(3)
             Wk_k1(4);Wk_k1(5) Wk_k1(6)];
line 33....  Wy = [Wk_k1(7) Wk_k1(8) Wk_k1(9)];
line 34....  h = tanh(Wh*x);
line 35....  h(1,1) = tanh(e*Wk_k1(1))+ Wk_k1(2)*Ts;
line 36....  h(3,1) = tanh((e*Wk_k1(5)+
             Wk_k1(6))/Ts
line 37....  u_dat = Wy*[h];
line 38....  narxIn=[u_datk1 u_datk2 y_k1 y_k2 1];
line 39....  narxH = 1./(1+exp(-[narxW]*[narxIn']));
line 40....  narxOut = [narxH',1]*[narxWy];
line 41....  Yk_k1 = Yk_k1 + narxOut;
line 42....  YYk_k1 = YYk_k1 + Yk_k1*Yk_k1';
line 43....  WYk_k1 = WYk_k1 + Wk_k1*Yk_k1';
line 44....  end
line 45....  yk_k1_dat = 1/(2*n)*Yk_k1;
line 46....  Pyy = 1/(2*n)*(YYk_k1) -
              (yk_k1_dat*yk_k1_dat') + Rr;
line 47....  Pxy = 1/(2*n)*(WYk_k1) -
              (wk_k1_dat*yk_k1_dat');
line 48....  K = Pxy/Pyy;
line 49....  P = Pk_k1 - K*Pyy*K';
line 50....  New_W = wk_k1_dat + lr*K*(err);
line 51....  Rr = Rr_k1 + (1/k)*(err^2 - Rr_k1);
line 52....  end
```

รูปที่ ข.4 โค้ดสำหรับอัลกอริทึมการเรียนรู้แบบไฮบริดของตัวกรองคาลมานด้วยกฎ Cubature

<u>คำอธิบายโค้ดโปรแกรม</u>

- *บรรทัดที่ 1* เริ่มต้นการทำงานของ While-loop เพื่อคำนวณค่าการปรับน้ำหนัก
- *บรรทัดที่ 2 - 4* เป็นการกำหนดตัวแปร `err` (ค่าผิดพลาดที่เกิดขึ้นของระบบ) `lr` (ค่าอัตราการเรียนรู้) `n` (ขนาดของเมตริกของตัวควบคุม) และ `Ie` (เมตริกเอกลักษณ์สำหรับคำนวณหาจุด Cubature) ตามลำดับ
- *บรรทัดที่ 5* หาค่ารากที่สองของความแปรปรวนร่วมเกี่ยวในอดีตล่าสุด (`Sk1_k1`)

- *บรรทัดที่ 6 – 21* เป็นการคำนวณหาค่าพยากรณ์ของน้ำหนักของตัวควบคุม Hybrid CKF-NNPID (Wk_k1_dat) และค่าความแปรปรวนร่วมเกี่ยว (Pk_k1) รอบ ๆ จุด Cubature ด้วยฟังก์ชัน For-Loop จากจุด 1 ถึงขนาดของน้ำหนัก (2*n)
- *บรรทัดที่ 7 - 11* เป็นการคำนวณค่าจุดของ Cubature (แทนด้วย damp)
- *บรรทัดที่ 12 - 17* คือคำนวณหาค่าพยากรณ์ของการปรับค่าน้ำหนักด้วยอัลกอริทึม Gradient
- *บรรทัดที่ 18 - 20* เป็นเก็บค่าของการแพร่ของจุด Cubature (Wk_k1_start) ค่าของการพยากรณ์น้ำหนักโครงข่าย (Wk_k1_dat) และค่าความแปรปรวนร่วมเกี่ยว (SWk_k1_star) ตามลำดับ ณ รอบการคำนวณของจุด Cubature
- *บรรทัดที่ 22 และ 23* เป็นคำนวณค่าพยากรณ์ของน้ำหนักโครงข่าย (Wk_k1) และค่าความแปรปรวนร่วมเกี่ยว (Pk_k1) ตามลำดับ
- *บรรทัดที่ 24* หาค่ารากที่สองของความแปรปรวนร่วมเกี่ยว (Pk1_k1) ด้วยฟังก์ชัน qr
- *บรรทัดที่ 25 – 44* เป็นการประเมินค่าความถูกต้องของค่าพยากรณ์ทั้งสอง (น้ำหนักของตัว, Wk_k1_dat และค่าความแปรปรวนร่วมเกี่ยว, Pk_k1) รอบ ๆ จุด Cubature ด้วยค่าคงที่คาลมาน (K) โดยใช้ฟังก์ชัน For-Loop จากจุด 1 ถึงขนาดของน้ำหนัก (2*n)
- *บรรทัดที่ 26 - 30* เป็นการคำนวณค่าจุดของ Cubature (แทนด้วย damp)
- *บรรทัดที่ 31 - 37* ป้อนค่าน้ำหนักที่พยากรณ์ไว้ไปยังตัวควบคุม Hybrid CKF-NNPID เพื่อหาค่าอินพุตควบคุมสำหรับป้อนไปยัง Identification Model ด้วยโครงข่ายประสาทเทียมแบบ NARX รอบ ๆ จุดของ Cubature ด้วยฟังก์ชัน For จากจุด 1 ถึงขนาดของน้ำหนัก (2*n)
- *บรรทัดที่ 37 - 44* เป็นการหาค่าพยากรณ์ของเอาต์พุตของระบบของจุด Cubature (Yk_k1) ค่าเอาต์พุตต่อการแพร่ของจุด Cubature (YYk_k1) และเก็บค่าน้ำหนักต่อการแพร่ของจุด Cubature (WYk_k1) ตามลำดับ
- *บรรทัดที่ 45 - 47* เป็นการเก็บค่าเอาต์พุต (yk_k1_dat) ค่าความแปรปรวนร่วมเกี่ยวของอินพุตต่อเอาต์พุต (Pxy) และค่าความแปรปรวนร่วมเกี่ยวของเอาต์พุตต่อเอาต์พุต (Pyy) ตามลำดับ
- *บรรทัดที่ 48 และ 49* เป็นคำนวณค่าคงที่คาลมาน (K) และค่าความแปรปรวนร่วมเกี่ยว (P) ตามลำดับ
- *บรรทัดที่ 50 และ 51* เป็นคำนวณค่าน้ำหนัก (New_W) และค่าผิดแปรปรวนของความผิดพลาด (Rr) ตามลำดับ

d) โค้ดสำหรับแบบจำลองของตัวควบคุม ABF-NNPID

โค้ดโปรแกรมสำหรับอัลกอริทึมการเรียนรู้ของตัวควบคุมที่นำเสนอนี้ดังรูปที่ ข.5 ซึ่งได้เขียนโค้ดโปรแกรมตามแนวคิดที่ได้กล่าวไว้ในหัวข้อที่ผ่านมา โดยที่ค่าเริ่มต้นต่าง ๆ จะถูกกำหนดไว้ในบล็อกของตัว Delay (บล็อกในโปรแกรม SIMULINK) ได้แก่ ค่าแปรปรวนร่วมเกี่ยว (P) ค่าคาลมาล (K) ค่าฟังก์ชันต้นทุน (J) และ แปรปรวนร่วมเกี่ยวของค่าผิดพลาด (R) แนวคิดของอัลกอริทึมนี้ได้พัฒนามาจากอัลกอริทึมของตัวควบคุม Hybrid CKF-NNPID ด้วยตัวกรองเบย์ (Bayesian Filter) โดยอาศัยกฎ Spherical-Radial Cubature Rule ดังนั้นโค้ดโปรแกรมของอัลกอริทึมจะมีส่วนคล้ายกับอัลกอริทึมของตัวควบคุม Hybrid CKF-NNPID เป็นอย่างมาก เพียงแต่การปรับค่าน้ำหนักจะใช้สามารถของแนวคิดของเบย์

```
line 1....  err = - e;
line 2....  lr = 0.001;
line 3....  n = length(W);
line 4....  Ie = eye(n);
line 5....  [Q,Sk1_k1]=qr(Pk1_k1');
line 6....  for i3=1:2*n
line 7....   if(i3<n+1)
line 8....     damp(:,1,i3) = sqrt(n)*Ie(:,i3);
line 9....   else
line 10....    damp(:,1,i3) = -sqrt(n)*Ie(:,i3-n);
line 11...   end
line 12....    Wk1_k1 = Sk1_k1'*damp(:,1,i3) + W;
line 13....    dif_u = err*(1);
line 14....    dif_Wy = lr*dif_u*h;
line 15....    dif_h = (1-tanh(ph)^2)*Wy*dif_u;
line 16....    dif_Wh = lr*dif_h*e;
line 17....    dif_W = [dif_Wh(1) W(2) dif_Wh(3)
               W(4) dif_Wh(5) W(6) dif_Wy]';
line 18....    Wk_k1_star = Wk1_k1 + dif_W;
line 19....    wk_k1 = wk_k1 + Wk_k1_star;
line 20....    SWk_k1_star = SWk_k1_star + Wk_k1_star
              *Wk_k1_star';
line 21....  end
line 22....  wk_k1_dat = 1/(2*n)*(wk_k1);
line 23....  Pk_k1 = 1/(2*n)*(SWk_k1_star)'-
             (wk_k1_dat)*(wk_k1_dat)';
line 24....  [Q,Sk_k1]=qr(Pk_k1);
line 25....  for i4=1:2*n
line 26....   if(i4<n+1)
line 27....     damp(:,1,i4) = sqrt(n)*Ie(:,i4);
line 28....   else
line 29....     damp(:,1,i4) = -sqrt(n)*Ie(:,i4-n);
line 30....   end
line 31....   Wk_k1 = Sk_k1'*damp(:,1,i4) +
```

```
                wk_k1_dat;
line 32....    Wh = [Wk_k1(1) Wk_k1(2); Wk_k1(3)
                Wk_k1(4);Wk_k1(5) Wk_k1(6)];
line 33....    Wy = [Wk_k1(7) Wk_k1(8) Wk_k1(9)];
line 34....    h = tanh(Wh*x);
line 35....    h(1,1) = tanh(e*Wk_k1(1))+ Wk_k1(2)*Ts;
line 36....    h(3,1) = tanh((e*Wk_k1(5)+
                Wk_k1(6))/Ts
line 37....    u_dat = Wy*[h];
line 38....    narxIn=[u_datk1 u_datk2 y_k1 y_k2 1];
line 39....    narxH = 1./(1+exp(-[narxW]*[narxIn']));
line 40....    narxOut = [narxH',1]*[narxWy];
line 41....    Yk_k1 = Yk_k1 + narxOut;
line 42....    YYk_k1 = YYk_k1 + Yk_k1*Yk_k1';
line 43....    WYk_k1 = WYk_k1 + Wk_k1*Yk_k1';
line 44....  end
line 45....  yk_k1_dat = 1/(2*n)*Yk_k1;
line 46....  Pyy = 1/(2*n)*(YYk_k1) -(yk_k1_dat*yk_k1_dat')
              + Rr;
line 47....  Pxy =1/(2*n)*(WYk_k1)- (wk_k1_dat*yk_k1_dat');
line 48....  P = Pk_k1 - K*Pyy*K';
line 49....  New_W = yk_k1_dat* Pxy/Pyy;
line 50....  Rr = Rr_k1 + (1/k)*(err^2 - Rr_k1);
```

รูปที่ ข.5 โค้ดสำหรับอัลกอริทึมการเรียนรู้ด้วยตัวกรองเบย์แบบประยุกต์ (Applied Bayesian Filter Algorithm)

<u>คำอธิบายโค้ดโปรแกรม</u>

- *บรรทัดที่ 1 - 4* เป็นการกำหนดตัวแปร `err` (ค่าผิดพลาดที่เกิดขึ้นของระบบ) `lr` (ค่าอัตราการเรียนรู้) `n` (ขนาดของเมตริกของตัวควบคุม) และ `Ie` (เมตริกเอกลักษณ์สำหรับคำนวณหาจุด Cubature) ตามลำดับ
- *บรรทัดที่ 5* หาค่ารากที่สองของความแปรปรวนร่วมเกี่ยวในอดีตล่าสุด (`Sk1_k1`)
- *บรรทัดที่ 6 – 21* เป็นการคำนวณหาค่าพยากรณ์ของน้ำหนักของตัวควบคุม Hybrid CKF-NNPID (`Wk_k1_dat`) และค่าความแปรปรวนร่วมเกี่ยว (`Pk_k1`) รอบ ๆ จุด Cubature ด้วยฟังก์ชัน For-Loop จากจุด 1 ถึงขนาดของน้ำหนัก (2*n)
- *บรรทัดที่ 7 - 11* เป็นการคำนวณค่าจุดของ Cubature (แทนด้วย `damp`)
- *บรรทัดที่ 12 - 17* คือคำนวณหาค่าพยากรณ์ของการปรับค่าน้ำหนักด้วยอัลกอริทึม Gradient

- *บรรทัดที่ 18 - 20* เป็นเก็บค่าของการแพร่ของจุด Cubature (Wk_k1_start) ค่าของการพยากรณ์น้ำหนักโครงข่าย (Wk_k1_dat) และค่าความแปรปรวนร่วมเกี่ยว (SWk_k1_star) ตามลำดับ ณ รอบการคำนวณของจุด Cubature
- *บรรทัดที่ 22 และ 23* เป็นคำนวณค่าพยากรณ์ของน้ำหนักโครงข่าย (Wk_k1) และค่าความแปรปรวนร่วมเกี่ยว (Pk_k1) ตามลำดับ
- *บรรทัดที่ 24* หาค่ารากที่สองของความแปรปรวนร่วมเกี่ยว (Pk1_k1) ด้วยฟังก์ชัน qr
- *บรรทัดที่ 25 – 44* เป็นการประเมินค่าความถูกต้องของค่าพยากรณ์ทั้งสอง (น้ำหนักของตัว, Wk_k1_dat และค่าความแปรปรวนร่วมเกี่ยว, Pk_k1) รอบ ๆ จุด Cubature ด้วยค่าคงที่คาลมาน (K) โดยใช้ฟังก์ชัน For-Loop จากจุด 1 ถึงขนาดของน้ำหนัก (2*n)
- *บรรทัดที่ 26 - 30* เป็นการคำนวณค่าจุดของ Cubature (แทนด้วย damp)
- *บรรทัดที่ 31 - 37* ป้อนค่าน้ำหนักที่พยากรณ์ไว้ไปยังตัวควบคุม Hybrid CKF-NNPID เพื่อหาค่าอินพุตควบคุมสำหรับป้อนไปยัง Identification Model ด้วยโครงข่ายประสาทเทียมแบบ NARX รอบ ๆ จุดของ Cubature ด้วยฟังก์ชัน For จากจุด 1 ถึงขนาดของน้ำหนัก (2*n)
- *บรรทัดที่ 37 - 44* เป็นการหาค่าพยากรณ์ของเอาต์พุตของระบบของจุด Cubature (Yk_k1) ค่าเอาต์พุตต่อการแพร่ของจุด Cubature (YYk_k1) และเก็บค่าน้ำหนักต่อการแพร่ของจุด Cubature (WYk_k1) ตามลำดับ
- *บรรทัดที่ 45 - 48* เป็นการเก็บค่าเอาต์พุต (yk_k1_dat) ค่าความแปรปรวนร่วมเกี่ยวของอินพุตต่อเอาต์พุต (Pxy) ค่าความแปรปรวนร่วมเกี่ยวของเอาต์พุตต่อเอาต์พุต (Pyy) และค่าความแปรปรวนร่วมเกี่ยวปัจจุบัน (P) ตามลำดับ
- *บรรทัดที่ 49 และ 50* เป็นคำนวณค่าน้ำหนัก (New_W) และค่าผิดแปรปรวนของความผิดพลาด (Rr) ตามลำดับ

e) โค้ดสำหรับแบบจำลองของตัวควบคุม NNPID-AC

โค้ดโปรแกรมสำหรับอัลกอริทึมการเรียนรู้ของตัวควบคุมที่นำเสนอนี้ดังรูปที่ ข.6 ซึ่งได้เขียนโค้ดโปรแกรมตามแนวคิดที่ได้กล่าวไว้ในหัวข้อ 3.2.4 ที่ผ่านมา โดยที่ค่าเริ่มต้นต่าง ๆ จะถูกกำหนดไว้ในบล็อกของตัว Delay (บล็อกในโปรแกรม SIMULINK) ได้แก่ ค่าแปรปรวนร่วมเกี่ยว (P) ค่าคาลมาล (K) ค่าฟังก์ชันต้นทุน (J) แปรปรวนร่วมเกี่ยวของค่าผิดพลาด (R) ค่าต้นทุน (cost_k1) และค่า ef ซึ่งมีค่าเท่ากับ 0.001 ซึ่งมีรายละเอียดการเขียนโค้ดโปรแกรมใน MATLAB ดังนี้

```
line 1....  While (abs(u_dat-u_predict)<= ef)
line 2....  err = - e, lr = 0.001;
line 3....  n = length(W);
line 4....  Ie = eye(n);
line 5....  [Q,Sk1_k1]=qr(Pk1_k1');
line 6....  for i3=1:2*n
line 7....   if(i3<n+1)
line 8....     damp(:,1,i3) = sqrt(n)*Ie(:,i3);
line 9....   else
line 10....    damp(:,1,i3) = -sqrt(n)*Ie(:,i3-n);
line 11....  end
line 12....    Wk1_k1 = Sk1_k1'*damp(:,1,i3) + W;
line 13....    dif_u = err*(1);
line 14....    dif_Wy = lr*dif_u*h;
line 15....    dif_h = (1-tanh(ph)^2)*Wy*dif_u;
line 16....    dif_Wh = lr*dif_h*e;
line 17....    dif_W = [dif_Wh(1) W(2) dif_Wh(3)
               W(4) dif_Wh(5) W(6) dif_Wy]';
line 18....    Wk_k1_star = Wk1_k1 + dif_W;
line 19....    wk_k1 = wk_k1 + Wk_k1_star;
line 20....    SWk_k1_star = SWk_k1_star +
               Wk_k1_star*Wk_k1_star';
line 21... end
line 22.... wk_k1_dat = 1/(2*n)*(wk_k1);
line 23.... for i3=1:2*n
line 24....  avg_Wk_k1(:,i3) =1/(sqrt(2*n)) *
             (Wk_k1_star(:,i3) - wk_k1_dat);
line 25.... end
line 26.... [discard_val,val] = qr(avg_Wk_k1');
line 27.... Sk_k1 = val(1:n,:)';
line 28.... for i4=1:2*n
line 29....  if(i4<n+1)
line 30....    damp(:,1,i4) = sqrt(n)*Ie(:,i4);
line 31....  else
line 32....    damp(:,1,i4) = -sqrt(n)*Ie(:,i4-n);
line 33....  end
line 34....  Wk_k1 = Sk_k1'*damp(:,1,i4) + wk_k1_dat;
line 35....  Wh = [Wk_k1(1) Wk_k1(2); Wk_k1(3)
             Wk_k1(4);Wk_k1(5) Wk_k1(6)];
line 36....  Wy = [Wk_k1(7) Wk_k1(8) Wk_k1(9)];
line 37....  h = tanh(Wh*x);
line 38....  h(1,1) = tanh(e*Wk_k1(1))+ Wk_k1(2)*Ts;
line 39....  h(3,1) = tanh((e*Wk_k1(5)+ Wk_k1(6))/Ts
line 40....  u_dat = Wy*[h];
line 41....  narxIn=[u_datk1 u_datk2 y_k1 y_k2 1];
line 42....  narxH = 1./(1+exp(-[narxW]*[narxIn']));
line 43....  u_predict = [narxH',1]*[narxWy];
```

```
line 44....   Yk_k1 = Yk_k1 + y_k1;
line 45....   YYk_k1 = YYk_k1 + Yk_k1*Yk_k1';
line 46....   WYk_k1 = WYk_k1 + Wk_k1*Yk_k1';
line 47....  end
line 48....  [discard_val,val] = qr(R');
line 49....  SR_k = val';
line 50....  for i3=1:2*n
line 51....   avg_Syyk_k1(:,i3) = 1/(sqrt(2*n))*
              (Yk_k1(:,i3) - yk_k1_dat);
line 52....  avg_Wyyk_k1(:,i3) = 1/(sqrt(2*n)) *
              (WWk_k1(:,i3) - wk_k1_dat);
line 53....  end
line 54....[discard_val,val] =qr([avg_Syyk_k1,SR_k]');
line 55....  Syyk_k1 = val(1:diR,:)';
line 56....  Pwyk_k1 = avg_Wyyk_k1*avg_Syyk_k1';
line 57....  K = (Pwyk_k1*inv(Syyk_k1'))/Syyk_k1;
line 58....  [discard_val,val] = qr([avg_Wyyk_k1 -
             K*avg_Syyk_k1]');
line 59....  Sk_k = val(1:n,:)';
line 60....  NewW =wk_k1_dat + lr*K*cost;
line 61....  Rr = Rr_k1 + (1/k)*(err^2 - Rr_k1);
line 62....  cost = 0.5*err^2 + cost_k1
line 63....  end %while
```

รูปที่ ข.6 อัลกอริทึมการเรียนรู้ปฏิบัติ-วิจารณ์แบบเสริมกำลัง

คำอธิบายโค้ดโปรแกรม

- *บรรทัดที่ 1 - 4* เป็นตัวแปรที่กำหนดเหมือนกันกับตัวควบคุม NNPID
- *บรรทัดที่ 5* เป็นการคำนวณหาค่ารากที่สองของความแปรปรวนร่วมเกี่ยวในอดีต (`Sk1_k1`)
- *บรรทัดที่ 6 - 21* เริ่มต้นการคำนวณค่าพยากรณ์ของการกระทำ (Actor) ประกอบด้วยค่าน้ำหนักของโครงข่ายของตัวควบคุม NNPID-AC (`Wk_k1_dat`) และค่าความแปรปรวนร่วมเกี่ยว (`Pk_k1`) รอบ ๆ จุด Cubature ด้วยฟังก์ชัน For-Loop จากจุด 1 ถึงขนาดของน้ำหนัก (2*n)
- *บรรทัดที่ 7 - 11* คือคำนวณค่าจุดของ Cubature (แทนด้วย `damp`)
- *บรรทัดที่ 12 - 17* คือคำนวณหาค่า Gradient ของระบบ
- *บรรทัดที่ 18 - 20* เป็นเก็บค่าของการแพร่ของจุด Cubature (`Wk_k1_start`) ค่าของการพยากรณ์น้ำหนักโครงข่าย (`Wk_k1_dat`) และค่าความแปรปรวนร่วมเกี่ยว (`SWk_k1_star`) ตามลำดับ ตามรอบการคำนวณของจุด Cubature

- *บรรทัดที่ 22 - 25* เป็นการคำนวณค่าพยากรณ์ของน้ำหนักโครงข่าย (`Wk_k1`) และค่าความแปรปรวนร่วมเกี่ยว (`Pk_k1`) ตามลำดับ
- *บรรทัดที่ 26* เป็นการคำนวณหาค่ารากที่สองของความแปรปรวนร่วมเกี่ยว (`Pk1_k1`) ด้วยฟังก์ชัน `QR decomposition`
- *บรรทัดที่ 28* เริ่มต้นการตรวจสอบความถูกต้องของค่าพยากรณ์ด้วยค่าคงที่คาลมาน (`K`) รอบ ๆ จุดของ Cubature ด้วยฟังก์ชัน For-Loop จากจุด 1 ถึงขนาดของน้ำหนัก (2*n)
- *บรรทัดที่ 28 - 33* เป็นการคำนวณค่าจุดของ Cubature (แทนด้วย damp)
- *บรรทัดที่ 34 - 47* คือป้อนค่าน้ำหนักที่พยากรณ์ไว้ไปยังตัวควบคุม NNPID เพื่อหาค่าอินพุตควบคุมสำหรับป้อนไปยัง Identification Model ด้วยโครงข่ายประสาทเทียมแบบ NARX ซึ่งเป็นการคำนวณค่าประมาณของตัวอินพุตควบคุม (ซึ่งจะแตกต่างจาก NARX ของตัวควบคุม Hybrid CKF-NNPID) รอบ ๆ จุดของ Cubature ด้วยฟังก์ชัน For จากจุด 1 ถึงขนาดของน้ำหนัก (2*n)
- *บรรทัดที่ 48 - 53* เป็นการเก็บค่าเอาต์พุตต่อการแพร่ของจุด Cubature (`YYk_k1`) และเก็บค่าน้ำหนักต่อการแพร่ของจุด Cubature (`WYk_k1`) ตามลำดับ
- *บรรทัดที่ 55 - 56* เป็นการเก็บค่าของการเอาต์พุต (`yk_k1_dat`) ค่าความแปรปรวนร่วมเกี่ยวของอินพุตต่อเอาต์พุต (`Pxy`) และ ค่าความแปรปรวนร่วมเกี่ยวของเอาต์พุตต่อเอาต์พุต (`Pyy`) ตามลำดับ
- *บรรทัดที่ 57 - 98* เป็นการคำนวณค่าคงที่คาลมาน (`K`) และค่ารากของความแปรปรวนร่วมเกี่ยว (`S`) ตามลำดับ
- *บรรทัดที่ 60* เป็นการคำนวณค่าน้ำหนัก (`New_W`)
- *บรรทัดที่ 61-62* เป็นการคำนวณค่า cost และค่าผิดแปรปรวนของความผิดพลาด (`Rr`) ตามลำดับ.
- *บรรทัดที่ 63* ปิด while-loop

สำหรับโค้ดโปรแกรมข้างต้นใช้สำหรับตัวควบคุม NNPID-AC ด้วยการพยากรณ์ตัว Actor ด้วยการใช้อัลกอริทึม Square-root Cubature Kalman Filter แบบดีกรี 3 ส่วนการใช้การพยากรณ์ตัว Actor ด้วยการใช้ Square-root Cubature Kalman Filter แบบดีกรี 5 จะต้องใช้เปลี่ยนโค้ดโปรแกรมในบรรทัดที่คำนวณค่า damp และค่าตัวพารามิเตอร์ต่าง ๆ ด้วยการใช้สมการ (3.78) เพื่อการแปลงค่าซึ่งโค้ดโปรแกรมนี้ไม่ได้เขียนลงในวิทยานิพนธ์ฉบับนี้

ภาคผนวก ค.

ผลงานวิจัยที่เกี่ยวข้องกับวิทยานิพนธ์ที่ได้รับการตีพิมพ์

ผลงานวิจัยที่เกี่ยวข้องกับวิทยานิพนธ์ที่ได้รับการตีพิมพ์เผยแพร่

บทความวิจัยที่เกี่ยวข้องกับงานวิจัยวิทยานิพนธ์ที่ได้รับการตีพิมพ์เผยแพร่ ในขณะที่ศึกษาอยู่ที่สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทั้งในและนอกประเทศ จำนวน 7 บทความ ประกอบด้วย บทความวิจัยที่ได้ตีพิมพ์ในวารสารวิชาการระดับนานาชาติ จำนวน 1 บทความ ในวารสารวิชาการระดับชาติ จำนวน 1 บทความ และบทความวิจัยที่ได้ตีพิมพ์ในการประชุมวิชาการระดับนานาชาติ จำนวน 5 บทความ มีรายละเอียดดังต่อไปนี้

ผลงานวิจัยที่ได้รับการตีพิมพ์เผยแพร่ในวารสารวิชาการระดับนานาชาติ

1) A Neural Network PID-Like Controller Using a Hybrid of Online Actor-Critic Reinforcement Algorithm with the Square Root Cubature Kalman Filter.

ผลงานวิจัยที่ได้รับการตีพิมพ์เผยแพร่ในวารสารวิชาการระดับชาติ

1) A. Sento, and Y. Kitjaidure, “A hybrid CKF-NNPID controller for MIMO nonlinear control system”, ECTI Transactions on Computer and Information Technology, Vol. 10, No. 2, pp. 176-184, 2016.

ผลงานวิจัยที่ได้รับการตีพิมพ์เผยแพร่ในการประชุมวิชาการระดับนานาชาติ

1) A. Sento, and Y. Kitjaidure, “Neural network controller based on PID using an extended Kalman filter algorithm for multi-variable non-linear control system”, In: Proc. of 2016 Eighth International Conf. on Advanced Computational Intelligence, pp. 302-309, 2016.
2) A. Sento and Y. Kitjaidure, "A hybrid CKF-NNPID controller for MIMO nonlinear control system," 2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Chiang Mai, 2016, pp. 1-6. doi: 10.1109/ECTICon.2016.7561410
3) A. Sento, P. Srisuk and Y. Kitjaidure, "An intelligent system architecture for meal assistant robotic arm," 2017 9th International Conference on Knowledge and Smart Technology (KST), Chonburi, 2017, pp. 166-171. doi: 10.1109/KST.2017.7886080
4) P. Srisuk, A. Sento and Y. Kitjaidure, "Forward kinematic-like neural network for solving the 3D reaching inverse kinematics problems," 2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications

and Information Technology (ECTI-CON), Phuket, 2017, pp. 214-217. doi: 10.1109/ECTICon.2017.8096211.

5) P. Srisuk, A. Sento and Y. Kitjaidure, "Inverse kinematics solution using neural networks from forward kinematics equations," 2017 9th International Conference on Knowledge and Smart Technology (KST), Chonburi, 2017, pp. 61-65. doi: 10.1109/KST.2017.7886084.

ประวัติผู้วิจัย

ชื่อ-นามสกุล นายอัดนา เซนโต๊ะ

ที่อยู่ หมู่บ้านพฤกษาวิลล์ 50/2 ถ. ราษฎร์พัฒนา แขวง ราษฎร์พัฒนา เขตสะพานสูง กรุงเทพฯ 10250

ประวัติการศึกษา

2548 วิศวกรรมศาสตรบัณฑิต สาขาวิชาไฟฟ้า
มหาวิทยาลัยธรรมศาสตร์

2556 วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาอิเล็กทรอนิกส์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

2561 วิศวกรรมศาสตรดุษฎีบัณฑิต สาขาวิชาไฟฟ้า
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ความชำนาญเฉพาะด้าน

1) การออกระบบไฟฟ้า
2) ระบบควบคุมอัจฉริยะ
3) แขนกลไฟฟ้า และหุ่นยนต์อัตโนมัติ
4) การออกแบบระบบงาน IoT
5) ระบบปฏิบัติการลีนุกซ์กับการประยุกต์ใช้งาน

ประสบการณ์การทำงานและผลงานวิจัย

พ.ศ.2548 - 2549 ตำแหน่งวิศวกรรมการผลิต
บริษัท ไทยแอโรว์ จำกัด

พ.ศ.2549 - 2552 ตำแหน่งวิศวกรรมการออกแบบผลิตภัณฑ์
บริษัท จรุงไทยไวร์แอนด์เคเบิ้ล จำกัด (มหาชน)

พ.ศ.2552 - 2553 ตำแหน่งหัวหน้าวิศวกรรมการออกแบบอุปกรณ์ต้นแบบ
บริษัท เอช จี ไทย อีเล็คโทรนิค

พ.ศ.2553 - ปัจจุบัน อาจารย์ประจำสถาบันเทคโนโลยีไทย-ญี่ปุ่น
ประจำห้องวิจัย Computer Engineering Robotics and Technology Research Laboratory (CERT)