การแก้ปัญหาจลนศาสตร์ผกผันแบบเคลื่อนที่เข้าหาเป้าหมายใน 3 มิติ ด้วยโครงข่ายประสาทเทียมจากสมการจลนศาสตร์ผันตรง

FORWARD KINEMATIC-LIKE NEURAL NETWORK FOR SOLVING THE 3D REACHING INVERSE KINEMATICS PROBLEMS

ปัณณวิทย์ ศรีสุข
PANNAWIT SRISUK

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.2560
KMITL-2017-EN-M-040-055

การแก้ปัญหาจลนศาสตร์ผกผันแบบเคลื่อนที่เข้าหาเป้าหมายใน 3 มิติ ด้วยโครงข่ายประสาทเทียมจากสมการจลนศาสตร์ผันตรง

FORWARD KINEMATIC-LIKE NEURAL NETWORK FOR SOLVING THE 3D REACHING INVERSE KINEMATICS PROBLEMS

ปัณณวิทย์ ศรีสุข
PANNAWIT SRISUK

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.2560
KMITL-2017-EN-M-040-055

FORWARD KINEMATIC-LIKE NEURAL NETWORK FOR SOLVING THE 3D REACHING INVERSE KINEMATICS PROBLEMS

PANNAWIT SRISUK

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF
MASTER OF ENGINEERING IN ELECTRONICS ENGINEERING
FACULTY OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2017

KMITL-2017-EN-M-040-055

คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ การแก้ปัญหาจลนศาสตร์ผกผันแบบเคลื่อนที่เข้าหาเป้าหมายใน 3 มิติด้วยโครงข่ายประสาทเทียมจากสมการจลนศาสตร์ผันตรง

Thesis Title Forward Kinematic-like Neural Network for Solving the 3D Reaching Inverse Kinematics Problems

นักศึกษา นายปัณณวิทย์ ศรีสุข

รหัสประจำตัว 59601297

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชา วิศวกรรมอิเล็กทรอนิกส์

อาจารย์ที่ปรึกษาวิทยานิพนธ์ ผศ.ดร.ยุทธนา คิดใจเดียว

หมายเลขวิทยานิพนธ์ KMITL-2017-EN-M-040-055

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
รศ.ดร.สุรพันธ์	ยิ้มมั่น	
รศ.ดร.สุรพันธุ์	เอื้อไพบูลย์	
รศ.ดร.ชูชาติ	ปิณฑวิรุจน์	
ผศ.ดร.ยุทธนา	คิดใจเดียว	

วัน / เดือน/ ปี ที่สอบ วันพฤหัสบดีที่ 6 กรกฎาคม พ.ศ. 2560 เวลา 10.00-12.00 น.

สถานที่สอบ ณ อาคาร A ชั้น 5 ห้องประชุม 3

คณะวิศวกรรมศาสตร์ รับรองแล้ว

(รองศาสตราจารย์ ดร. คมสัน มาลีสี)

คณบดี คณะวิศวกรรมศาสตร์

วันที่ 6 กรกฎาคม พ.ศ. 2560

หัวข้อวิทยานิพนธ์	การแก้ปัญหาจลนศาสตร์ผกผันแบบเคลื่อนที่เข้าหาเป้าหมาย ใน 3 มิติด้วยโครงข่ายประสาทเทียมจากสมการจลนศาสตร์ผันตรง
นักศึกษา	นายปัณณวิทย์ ศรีสุข
รหัสประจำตัว	59601297
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมอิเล็กทรอนิกส์
พ.ศ.	2560
อาจารย์ที่ปรึกษาวิทยานิพนธ์	ผศ.ดร.ยุทธนา คิดใจเดียว

บทคัดย่อ

ปัญหาจลนศาสตร์ผกผัน (Inverse kinematics problem) คือหนึ่งในปัญหาสำคัญของการควบคุมหุ่นยนต์เพราะการแก้ปัญหาจลนศาสตร์ผกผันเป็นการคำนวณหาค่ามุมตามข้อต่อของแขนกลที่เป็นไปได้เมื่อสมมติให้จุดปลายแขนกล (The end-effector) อยู่ที่ตำแหน่งเป้าหมาย (Desired position) หรือเราสามารถควบคุมแขนกลให้ไปหยิบจับวัตถุได้โดยต้องการแค่ตำแหน่งของวัตถุนั้น แต่ขั้นตอนการแก้ปัญหานั้นยุ่งยากซับซ้อนแปรผันตามจำนวนองศาอิสระของแขนกล อีกทั้งวิธีการแก้ปัญหาจลนศาสตร์ผกผันในปัจจุบันมักจะออกแบบการแก้ปัญหาโดยสมมติให้แขนกลอยู่ในระบบ 2 มิติเสมอ ดังนั้นวิทยานิพนธ์ฉบับนี้จึงต้องการนำเสนอวิธีการแก้ปัญหาจลนศาสตร์ผกผันใน 3 มิติด้วยโครงข่ายประสาทเทียมพิเศษที่สร้างจากสมการจลนศาสตร์ผันตรง (Forward kinematics equations) ที่มีอินพุตเป็นมุมข้อต่อ (Angle joints) และเอ้าพุตเป็นระยะขจัดจากจุดปลายแขนกลถึงตำแหน่งเป้าหมาย จากนั้นสอนโครงข่ายด้วยอัลกอรึทึมการเรียนรู้แบบแพร่ย้อนกลับด้วยอัตราการเรียนรู้ไม่คงที่ (Variable learning rate backpropagation algorithm) แล้วทำการพิจารณาค่าน้ำหนักและไบอัสสุดท้ายเพื่อนำมาเป็นค่ามุมคำตอบ โดยวิธีนี้จะมีจุดเด่นที่โครงสร้างของโครงข่ายประสาทเทียมมีขนาดเล็กตามสมการจลนศาสตร์ผันตรง โครงข่ายไม่ต้องถูกสอนด้วยข้อมูลจำนวนมากก่อนและแขนกลเคลื่อนที่เข้าหาตำแหน่งเป้าหมายใน 3 มิติได้เหมือนกับมนุษย์ยื่นมือไปหยิบจับวัตถุ ซึ่งการเคลื่อนที่เข้าสู่เป้าหมายลักษณะนี้อาจทำการประยุกต์เพื่อให้แขนกลเคลื่อนที่หลบหลีกสิ่งกีดขวางได้ต่อไป

Thesis	Forward Kinematic-like Neural Network for Solving the 3D Reaching Inverse Kinematics Problems
Student	Mr.Pannawit Srisuk
Student ID.	59601297
Degree	Master of Engineering
Program	Electronics Engineering
Year	2017
Thesis Advisor	Asst.Prof.Dr.Yuttana Kitjaidure

ABSTRACT

This paper presents the inverse kinematic solutions based on neural networks. General neural network approaches use data of the end-effector positions as an input and angle joints as an output to train the neural network for mapping the input to the output. However, the proposed method creates the custom networks from forward kinematic equations. This special structure makes the network like a position finder with ability to automatically adjust angle joints until the end-effector reaches the desired position by variable learning rate backpropagation algorithm. Then, the solutions of angles can be found from the final weights and bias values. Moreover, the proposed network uses less number of neurons and amount of the solution space is not depend on the training data. Finally, to evaluate the performance algorithm, the MATLAB Program is used to demonstrate a 4-DOF robotic arm movement in 3-dimensional. As a result, the proposed algorithm can help a robotic arm move to the desired position (3D reaching) quickly and correctly.

กิตติกรรมประกาศ

วิทยานิพนธ์เล่มนี้สำเร็จได้ด้วยความกรุณาจากอาจารย์ที่ปรึกษา ผศ.ดร.ยุทธนา คิดใจเดียว ที่คอยให้ความช่วยเหลือและแนะนำแนวทางในการแก้ปัญหาที่เกิดขึ้นในการทำวิจัย อีกทั้งยังช่วยตรวจสอบแก้ไขความถูกต้องของผลงานวิชาการและรูปเล่มวิทยานิพนธ์จนเสร็จสมบรูณ์ นอกจากนั้นยังคอยปลูกฝังให้ฝึกฝนทักษะการเรียนรู้ด้วยตนเองและพร้อมที่จะรับความรู้ใหม่ๆอยู่เสมอ

ขอขอบคุณ นางสาวพิมพ์พร เหมยน้อย หรือพี่พิมพ์ พี่ที่น่ารักของน้องๆ ที่คอยพูดคุยให้กำลังใจและเป็นที่ปรึกษาสำหรับปัญหาชีวิตและปัญหาวิชาการตลอดการเรียนปริญญาโทที่ผ่านมา อีกทั้งยังช่วยตรวจทานแก้ไขรูปเล่มวิทยานิพนธ์ด้วยความละเอียดและตั้งใจจนเสร็จสมบรูณ์

ขอขอบคุณ นายอัดนา เซ็งโต๊ะ หรือพี่อัด ที่ได้นำงานวิจัยต่างๆมาเสนอในช่วงเลือกหัวข้อวิจัย และหนึ่งในนั้นก็เป็นงานวิจัยที่ตัวข้าพเจ้าได้เลือกมาเป็นงานวิจัยต้นแบบในการทำงานวิจัยของข้าพเจ้าเอง อีกทั้งยังคอยชี้แนะแนวทางเพื่อปรับปรุงงานวิจัยให้ดียิ่งขึ้น

สำหรับคุณงามความดีอันใดที่เกิดจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบให้กับบิดามารดา ซึ่งเป็นที่รักและเคารพยิ่ง ตลอดจนครูอาจารย์และผู้มีพระคุณที่เคารพทุกท่าน

ปัณณวิทย์ ศรีสุข

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป(ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

หลักจลนศาสตร์ในหุ่นยนต์เป็นการศึกษาความสัมพันธ์ระหว่างตำแหน่งของแขนกลกับความเร็ว หรือ ความเร่ง ของแขนกลในแต่ละท่อนโดยไม่สนใจแรงที่มากระทำเพื่อนำไปใช้ควบคุมหุ่นยนต์ โดยหลักจลนศาสตร์สามารถแบ่งออกเป็น 2 แบบคือ จลนศาสตร์ผันตรงกับจลนศาสตร์ผกผัน จลนศาสตร์ผันตรงคือการหาตำแหน่งปลายของแขนกลโดยการกำหนดค่ามุมตามข้อต่อต่างๆ ของแขนกล ในทางกลับกันจลนศาสตร์ผกผันคือขั้นตอนการหามุมตามข้อต่อต่างๆจากตำแหน่งเป้าหมายของจุดปลายของแขนกล ซึ่งวิธีนี้สามารถใช้ควบคุมแขนกลไปยังตำแหน่งเป้าหมายได้โดยต้องการเพียงตำแหน่งที่อยู่ของเป้าหมายเท่านั้น ดังนั้นวิธีจลนศาสตร์ผกผันจึงเป็นที่นิยมทั้งในการควบคุมแขนกลอุตสาหกรรมและการสร้างภาพเคลื่อนไหวในงานอนิเมชั่น แต่ขั้นตอนการหาคำตอบของวิธีจลนศาสตร์ผกผันนั้นมีความยุ่งยากซับซ้อน อีกทั้งความซับซ้อนของสมการในการหาคำตอบยังแปรผันตามจำนวนองศาอิสระของแขนกลอีกด้วย จึงเป็นเหตุให้ปัญหาจลนศาสตร์ผกผันถูกวิจัยและพัฒนามาอย่างยาวนาน เพื่อคิดค้นการแก้ปัญหาจลนศาสตร์ผกผันแบบใหม่ที่มีความซับซ้อนน้อยลงแต่ยังคงความแม่นยำในการหาคำตอบไว้

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

วิทยานิพนธ์ฉบับนี้ต้องการนำเสนอวิธีการแก้ปัญหาจลนศาสตร์ผกผันแบบใหม่โดยการสร้างโครงข่ายประสาทเทียมแทนตำแหน่ง x, y, z ของจุดปลายแขนกลจากสมการจลนศาสตร์ผันตรง จากนั้นใช้การเรียนรู้แบบแพร่ย้อนกลับด้วยอัตราการเรียนรู้ไม่คงที่เพื่อลดค่าความผิดพลาดของทั้ง 3 ตำแหน่งโดยมุ่งหวังให้ความซับซ้อนของขั้นตอนการหาคำตอบลดลง ไม่ต้องใช้ข้อมูลจำนวนมากในการสอนโครงข่ายและสามารถเคลื่อนที่เข้าหาตำแหน่งเป้าหมายใน 3 มิติได้

1.3 สมมุติฐานของการศึกษา

คาดว่าจะสามารถสร้างโครงข่ายประสาทเทียมจากสมการจลนศาสตร์ผันตรงเพื่อนำมาแก้ปัญหาจลนศาสตร์ผกผันได้โดยไม่ต้องถูกสอนด้วยข้อมูลจำนวนมากก่อนและสามารถเคลื่อนที่เข้าหาตำแหน่งเป้าหมายใน 3 มิติได้ อีกทั้งโครงสร้างของโครงข่ายประสาทเทียมที่สร้างขึ้นจะมีขนาดเล็กกว่าโครงข่ายประสาทเทียมที่ใช้งานตามปกติ

1.4 ทฤษฎีหรือแนวความคิดที่ใช้ในการวิจัย

การแก้ปัญหาจลนศาสตร์ผกผันสามารถแบ่งออกได้ 3 ประเภทหลัก คือ 1.วิธีทางพีชคณิต 2.วิธีทางเรขาคณิต 3.วิธีการคำนวณแบบวนซ้ำ โดยวิธีที่เป็นที่นิยมมากในปัจจุบันคือ วิธีที่ 3 การคำนวณแบบวนซ้ำ เพราะวิธีนี้ไม่ต้องการสมการที่ซับซ้อนในการคำนวณและจะได้คำตอบเพียงหนึ่งเดียว ซึ่งโครงข่ายประเทียมเป็นหนึ่งในวิธีการคำนวณแบบวนซ้ำที่ถูกนำมาประยุกต์ใช้แก้ปัญหาจลนศาสตร์ผกผันอย่างมากในปัจจุบันเพราะมีการเรียนรู้ที่รวดเร็วและวิธีการเรียนรู้ใหม่ๆก็ถูกคิดค้นขึ้นอย่างต่อเนื่อง แต่งานวิจัยส่วนมากจะใช้ข้อมูลความสัมพันธ์ระหว่างจุดปลายแขนกลกับค่ามุมข้อต่อมาสอนโครงข่ายประสาทเทียมเพื่อให้สร้างความสัมพันธ์ผกผันระหว่างกัน ซึ่งมีข้อเสียคือขอบเขตการหาคำตอบจะถูกจำกัดด้วยข้อมูลที่นำมาสอนและโครงสร้างของโครงข่ายประสาทเทียมมีขนาดใหญ่แปรผันตามขนาดข้อมูล ในขณะเดียวกันมีงานวิจัยบางส่วนที่สร้างโครงข่ายให้มีโครงสร้างเหมือนกับสมการจลนศาสตร์ผันตรงแล้วทำการปรับค่าน้ำหนักจนได้ค่าเอ้าพุตตามที่ต้องการแทนการสร้างความสัมพันธ์ระหว่างข้อมูล ซึ่งมีข้อดีที่ไม่ต้องใช้ข้อมูลจำนวนมากมาสอนและโครงสร้างของโครงข่ายประสาทเทียมมีขนาดเล็กกว่าโครงข่ายประสาทเทียมที่ใช้ในวิธีแรก

1.5 ขอบเขตของการศึกษา

คิดค้นวิธีการแก้ปัญหาจลนศาสตร์ผกผันใน 3 มิติด้วยโครงข่ายประสาทเทียมที่สร้างมาจากสมการจลนศาสตร์ผันตรงของแขนกลและใช้การเรียนรู้แบบแพร่ย้อนกลับด้วยอัตราการเรียนรู้ไม่คงที่ พร้อมทั้งทดสอบประสิทธิภาพของโครงข่ายประสาทเทียมที่สร้างขึ้นด้วยการทดสอบการลู่เข้าคำตอบที่ค่ามุมเริ่มต้นแบบสุ่มและตำแหน่งเป้าหมายแบบสุ่ม สุดท้ายทดลองให้แขนกลเคลื่อนที่ติดตามเส้นกราฟแบบต่างๆ โดยการทดลองทั้งหมดจะถูกจำลองในโปรแกรม MATLAB

1.6 ขั้นตอนของการศึกษา

เริ่มด้วยการศึกษาหาสมการจลนศาสตร์ผันตรงของแขนกลที่เราต้องการใช้ แล้วสร้างโครงข่ายประสาทเทียมแทนตำแหน่ง x,y,z ในสมการจลนศาสตร์ผันตรงจากนั้นใช้อัลกอริทึมการเรียนรู้แบบแพร่ย้อนกลับด้วยอัตราการเรียนรู้ไม่คงที่เพื่อทำการลดค่าความคลาดเคลื่อนของโครงข่ายทั้ง 3 ให้อยู่ในเกณฑ์ที่ยอมรับได้ โดยค่ามุมคำตอบจะหาได้จากค่าน้ำหนักและไบอัสที่ทำการปรับปรุงเสร็จแล้ว

บทที่ 2
งานวิจัยที่เกี่ยวข้อง

ตั้งแต่อดีตถึงปัจจุบันปัญหาจลนศาสตร์ผกผันได้ถูกศึกษาวิจัยมาอย่างยาวนานโดยวิธีที่ถูกคิดค้นขึ้นสามารถแบ่งได้เป็น 3 วิธีหลักๆคือ 1.วิธีทางพีชคณิต (Algebraic Method) 2.วิธีทางเรขาคณิต (Geometry Method) 3.วิธีการคำนวณแบบวนซ้ำ (Iterative Method) แต่ปัจจุบันวิธีการคำนวณที่เป็นที่นิยมคือแบบที่ 3 วิธีการคำนวณแบบวนซ้ำ เนื่องมาจากวิธีที่ 1 และ 2 ล้วนแต่ต้องใช้ความสัมพันธ์แบบผกผันของตำแหน่งจุดปลายแขนกลกับมุมของข้อต่อแขนกล ซึ่งเมื่อแขนกลมีจำนวนข้อต่อมากจะทำให้สมการผกผันมีความซับซ้อนสูงตามไปด้วย แต่วิธีการคำนวณแบบวนซ้ำส่วนใหญ่จะใช้หลักการทางเรขาคณิตมาขยับแขนกลจนกระทั่งเจอคำตอบที่เป็นไปได้ ตัวอย่างการคำนวณของวิธีวนซ้ำมีดังต่อไปนี้

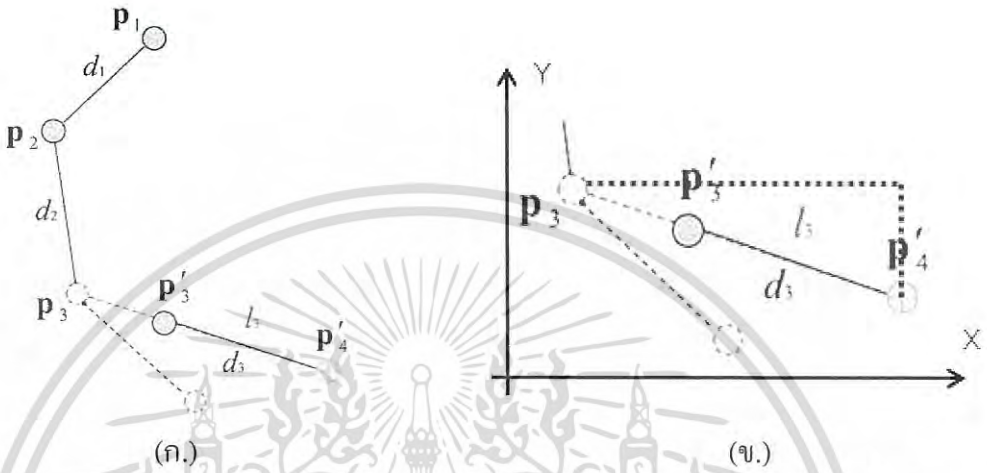
2.1 การแก้ปัญหาจลนศาสตร์ผกผันโดยการคำนวณแบบวนซ้ำด้วยหลักการทางเรขาคณิต

ในปี 2011 งานวิจัยโดย A. Aristidou กับ J. Lasenby [11] ได้คิดค้นวิธีแก้ปัญหาจลนศาสตร์ผกผันโดยใช้หลักการทางเรขาคณิต โดยถูกเรียกย่อๆว่า FABRIK (Forward And Backward Reaching Inverse Kinematics) วิธีนี้มีขั้นตอนการคำนวณไม่ซับซ้อน ซึ่งในงานวิจัยของวิธี FABRIK นี้ได้ทำการทดลองเพื่อเปรียบเทียบความเร็วในการคำนวณของวิธี FABRIK กับวิธีการคำนวณแบบวนซ้ำอื่นๆอย่างเช่น CCD [12], Jacobian Transpose, Jacobian DLS, Jacobian pseudo-inverse DLS (SVD-DLS), FTL and Triangulation ผลคือวิธี FABRIK ใช้เวลาในการคำนวณน้อยที่สุดดังแสดงในรูปที่ 2.1 และขอบเขตของการหาคำตอบก็ไม่ถูกจำกัด ดังนั้นวิธี FABRIK จึงเป็นที่นิยมอย่างมากอีกทั้งในปัจจุบันยังมีงานวิจัยจำนวนมากที่นำวิธี FABRIK นี้ไปปรับปรุงและพัฒนาต่อในเรื่องข้อจำกัดของมุมข้อต่อและลักษณะท่าทางของมุมคำตอบให้ดูเป็นธรรมชาติ รายงานฉบับนี้จึงต้องการอธิบายการทำงานของวิธี FABRIK เพื่อที่จะนำข้อดีข้อเสียมาเปรียบเทียบกับวิธีที่ต้องการนำเสนอ

	Reachable Target				Unreachable Target	
	Number of Iterations	Matlab exe. time (*sec*)	Time per iteration (in *msec*)	Iterations per second	Number of Iterations	Matlab exe. time (*sec*)
FABRIK	15.461	0.01328	0.86	1164	67.564	0.06207
CCD	26.308	0.12356	4.69	213	390.135	3.92869
Jacobian Transpose	1311.190	12.98947	9.90	101	6549.000	33.90473
Jacobian DLS	998.648	10.48051	10.49	95	2881.667	14.87918
Jacobian SVD-DLS	808.797	9.29652	11.50	87	2808.452	15.97591
FTL	21.125	0.02045	0.97	1033	22.325	0.02526
Triangulation	1.000	0.05747	57.47	21	1.000	0.06993

รูปที่ 2.1 ผลลัพธ์การคำนวณของวิธี FABRIK กับวิธีอื่นๆ [11]

วิธี FABRIK จะทำการขยับตำแหน่งข้อต่อเข้าไปยังตำแหน่งเป้าหมายแล้วขยับกลับมายังตำแหน่งฐาน โดยการเลื่อนจุดข้อต่อแต่ละจุดจะทำกการสร้างรูปสามเหลี่ยม ซึ่งกำหนดให้ระยะทางจากจุดตำแหน่งเป้าหมายของข้อต่อกับตำแหน่งเดิมเป็นด้านยาวที่สุดของสามเหลี่ยมและใช้หลักการสามเหลี่ยมคล้ายในการหาตำแหน่งข้อต่อใหม่ที่จะเกิดขึ้นบนเส้นตรงนี้ดังแสดงในรูปที่ 2.2



(ก.) (ข.)

รูปที่ 2.2 การคำนวณหาตำแหน่งใหม่ของข้อต่อด้วยวิธี 3 เหลี่ยมคล้าย (ก.) การขยับจุดข้อต่อแขนกล (ข.) สามเหลี่ยมสมมติเพื่อใช้หาตำแหน่งข้อต่อใหม่ [11]

จะเห็นได้ว่าสามารถคำนวณหาตำแหน่ง P_3' ได้จากสมการสามเหลี่ยมคล้ายโดยกำหนดให้ตัวแปร P_n คือตำแหน่งของจุดข้อต่อที่ n, P_n' คือตำแหน่งใหม่ของจุดข้อต่อ n, d_n คือระยะทางจากจุด P_n ไป P_{n+1} และ ค่า l_n คือระยะทางจากจุด P_n ไป P_{n+1}'

ตำแหน่ง x ใหม่ของข้อต่อที่ n

$$\frac{\ell_n}{P'_{(n+1)x} - P_{nx}} = \frac{\ell_n - d_n}{P'_{nx} - P_{nx}}$$

$$P'_{nx} - P_{nx} = \frac{(\ell_n - d_n)(P'_{(n+1)x} - P_{nx})}{\ell_n}$$

$$P'_{nx} = \frac{(\ell_n - d_n)(P'_{(n+1)x} - P_{nx})}{\ell_n} + P_{nx} \qquad (2.1)$$

ตำแหน่ง y ใหม่ของข้อต่อที่ n

$$\frac{\ell_n}{P_{ny} - P'_{(n+1)y}} = \frac{\ell_n - d_n}{P_{ny} - P'_{ny}}$$

$$P_{ny} - P'_{ny} = \frac{(\ell_n - d_n)(P_{ny} - P'_{(n+1)y})}{\ell_n}$$

$$P'_{ny} = P_{ny} - \frac{(\ell_n - d_n)(P_{ny} - P'_{(n+1)y})}{\ell_n} \qquad (2.2)$$

ทำกระบวนการนี้กับตำแหน่งของแขนกลทั้งหมดโดยทำจากจุดปลายไปหาจุดฐาน (Forward reaching) แล้วขยับจุดฐานกลับไปโดยมีตำแหน่งฐานเดิมเป็นตำแหน่งเป้าหมาย (Backward reaching) โดยขั้นตอนการขยับแขนกลไปหาตำแหน่งเป้าหมายและขยับแขนกลกลับไปตำแหน่งฐานเดิมจะแสดงในรูปที่ 2.3 นอกจากนั้นขั้นตอนการคำนวณยังแสดงไว้ในรูปที่ 2.4

รูปที่ 2.3 การเคลื่อนที่เข้าหาคำตอบของวิธี FABRIK [11]

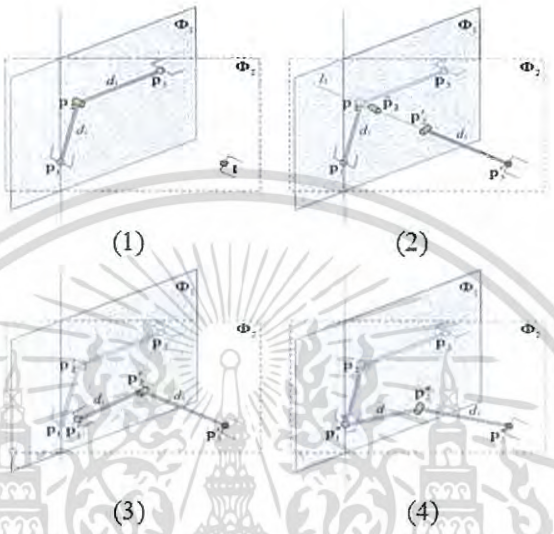
Algorithm 1. A full iteration of the FABRIK algorithm

Input: The joint positions $\mathbf{p}_i$ for $i = 1, \ldots, n$, the target position $\mathbf{t}$ and the distances between each joint $d_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$ for $i = 1, \ldots, n-1$.
Output: The new joint positions $\mathbf{p}_i$ for $i = 1, \ldots, n$.

1.1 *% The distance between root and target*
1.2 $dist = |\mathbf{p}_1 - \mathbf{t}|$
1.3 *% Check whether the target is within reach*
1.4 **if** $dist > d_1 + d_2 + \ldots + d_{n-1}$ **then**
1.5 *% The target is unreachable*
1.6 **for** $i = 1, \ldots, n-1$ **do**
1.7 *% Find the distance r_i between the target $\mathbf{t}$ and the joint position $\mathbf{p}_i$*
1.8 $r_i = |\mathbf{t} - \mathbf{p}_i|$
1.9 $\lambda_i = d_i / r_i$
1.10 *% Find the new joint positions $\mathbf{p}_i$.*
1.11 $\mathbf{p}_{i+1} = (1 - \lambda_i)\,\mathbf{p}_i + \lambda_i \mathbf{t}$
1.12 **end**
1.13 **else**
1.14 *% The target is reachable; thus, set as $\mathbf{b}$ the initial position of the joint $\mathbf{p}_1$*
1.15 $\mathbf{b} = \mathbf{p}_1$
1.16 *% Check whether the distance between the end effector $\mathbf{p}_n$ and the target $\mathbf{t}$ is greater than a tolerance.*
1.17 $dif_A = |\mathbf{p}_n - \mathbf{t}|$
1.18 **while** $dif_A > tol$ **do**
1.19 *% STAGE 1: FORWARD REACHING*
1.20 *% Set the end effector $\mathbf{p}_n$ as target $\mathbf{t}$*
1.21 $\mathbf{p}_n = \mathbf{t}$
1.22 **for** $i = n-1, \ldots, 1$ **do**
1.23 *% Find the distance r_i between the new joint position $\mathbf{p}_{i+1}$ and the joint $\mathbf{p}_i$*
1.24 $r_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$
1.25 $\lambda_i = d_i / r_i$
1.26 *% Find the new joint positions $\mathbf{p}_i$.*
1.27 $\mathbf{p}_i = (1 - \lambda_i)\,\mathbf{p}_{i+1} + \lambda_i \mathbf{p}_i$
1.28 **end**
1.29 *% STAGE 2: BACKWARD REACHING*
1.30 *% Set the root $\mathbf{p}_1$ its initial position.*
1.31 $\mathbf{p}_1 = \mathbf{b}$
1.32 **for** $i = 1, \ldots, n-1$ **do**
1.33 *% Find the distance r_i between the new joint position $\mathbf{p}_i$ and the joint $\mathbf{p}_{i+1}$*
1.34 $r_i = |\mathbf{p}_{i+1} - \mathbf{p}_i|$
1.35 $\lambda_i = d_i / r_i$
1.36 *% Find the new joint positions $\mathbf{p}_i$.*
1.37 $\mathbf{p}_{i+1} = (1 - \lambda_i)\,\mathbf{p}_i + \lambda_i \mathbf{p}_{i+1}$
1.38 **end**
1.39 $dif_A = |\mathbf{p}_n - \mathbf{t}|$
1.40 **end**
1.41 **end**

รูปที่ 2.4 ขั้นตอนการทำงานของ FABRIK [11]

สำหรับตำแหน่งเป้าหมายใน 3 มิติ วิธี FABRIK ต้องประยุกต์วิธีการเล็กน้อยก่อนเริ่มการคำนวณ จะสังเกตเห็นว่าวิธี FABRIK นี้ออกแบบการแก้ปัญหาโดยสมมติให้แขนกลกับตำแหน่งเป้าหมายอยู่ในระนาบเดียวกัน เพราะฉะนั้นในกรณีที่ตำแหน่งเป้าหมายอยู่ใน 3 มิติเราจะต้องทำการหมุนแขนกลให้หันมายังระนาบเดียวกับตำแหน่งเป้าหมายก่อนที่จะทำการคำนวณแบบปกติ ดังแสดงในรูปที่ 2.5



รูปที่ 2.5 วิธี FABRIK ต้องหันแขนกลให้อยู่ในระนาบเดียวกับตำแหน่งเป้าหมายใน 3 มิติก่อนทำการคำนวณแบบปกติ [11]

2.2 การแก้ปัญหาจลนศาสตร์ผกผันด้วยโครงข่ายประสาทเทียม

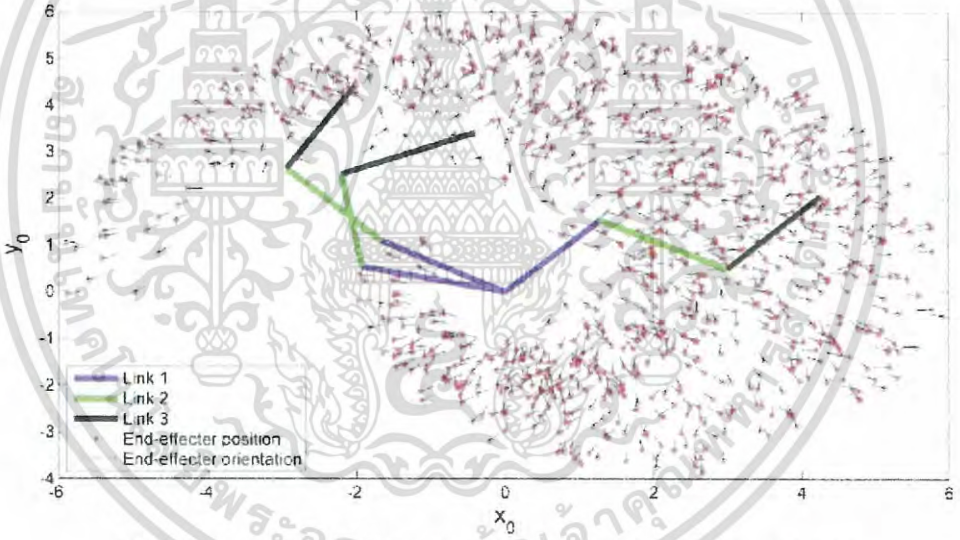
การใช้โครงข่ายประสาทเทียมในการแก้ปัญหาจลนศาสตร์ผกผันถูกศึกษามาอย่างยาวนานแล้ว โดยวิธีการทางโครงข่ายประสาทเทียมจะแบ่งได้เป็น 2 ประเภทหลักๆคือ การใช้โครงข่ายประสาทเทียมสร้างความสัมพันธ์ผกผันระหว่างข้อมูลจุดปลายแขนกลกับมุมข้อต่อ (Mapping) กับการสร้างโครงข่ายประสาทเทียมให้มีโครงสร้างเหมือนกับสมการจลนศาสตร์ผันตรงคล้ายกับวิธี PID-like ในระบบควบคุม

2.2.1 การใช้โครงข่ายประสาทเทียมสร้างความสัมพันธ์ระหว่างข้อมูล

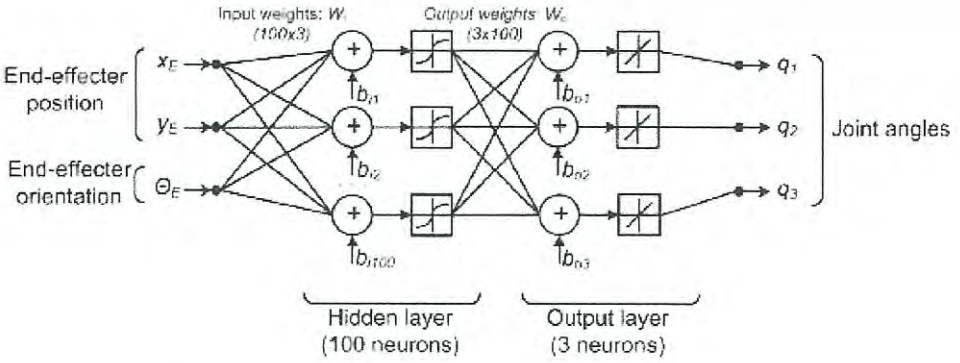
การประยุกต์ใช้โครงข่ายประสาทเทียมกับปัญหาจลนศาสตร์ผกผันถูกคิดค้นขึ้นเมื่อราวๆ ปี 1989 โดย Allon Guez and Ziauddin Ahmad [2] โดยการทำงานจะใช้หลักการเหมือนกับการใช้งานโครงข่ายประสาทเทียมแบบทั่วไปคือ สอนโครงข่ายด้วยคู่ข้อมูลอินพุตและเอ้าพุตจำนวนมากเพื่อให้โครงข่ายสร้างความสัมพันธ์ระหว่างข้อมูล จากนั้นเมื่อมีอินพุตใดๆเข้ามาโครงข่ายก็จะให้ค่าเอ้าพุตที่ใกล้เคียงกับคู่อินพุตที่เคยนำมาสอนโครงข่ายไว้ เพราะฉะนั้นในการแก้ปัญหาจลนศาสตร์ผกผันด้วยโครงข่ายประสาทเทียมแบบปกตินี้ เราจำเป็นต้องสร้างข้อมูลความสัมพันธ์ระหว่างตำแหน่งจุดปลายแขนกล(อินพุต)กับมุมที่ข้อต่อ (เอ้าพุต) เพื่อนำมาสอนโครงข่าย (สร้างข้อมูลได้จากสมการจลนศาสตร์ผันตรง) วิธีนี้มีข้อดีคือเราสามารถสร้างข้อมูลความสัมพันธ์ได้ง่ายจากสมการจลนศาสตร์

ผันตรงและไม่จำเป็นต้องหาค่าผกผันของสมการใดๆ แต่วิธีนี้ก็มีข้อเสียคือขอบเขตในการหาคำตอบจะถูกจำกัดอยู่ในช่วงของข้อมูลที่นำไปสอนและเมื่อจำนวนองศาอิสระของแขนกลมากขึ้นค่ามุมที่เป็นไปได้ ณ ตำแหน่งจุดปลายแขนกลตำแหน่งหนึ่งก็จะมีมากขึ้นตามไปด้วย และเมื่อจำนวนข้อมูลที่จะนำมาสอนมีมากโครงสร้างของโครงข่ายประสาทเทียมก็จะมีขนาดใหญ่แปรผันตามกัน แต่วิธีนี้ก็ถูกศึกษาพัฒนามาอย่างต่อเนื่องจนถึงปัจจุบัน โดยส่วนใหญ่งานวิจัยใหม่ๆจะทำการเลือกชุดคู่ข้อมูลใหม่ๆโดยการเพิ่มค่าตัวแปรอินพุตเพื่อนำมาพิจารณาร่วมกับตำแหน่ง x,y,z ของจุดปลายแขนกล โดยมุ่งหวังให้โครงข่ายสามารถให้ค่ามุมคำตอบที่แม่นยำและเป็นธรรมชาติมากขึ้น [6,7,8,9,10]

ตัวอย่างงานวิจัยในปี 2014 โดย Adrian-Vasile Duka [10] ได้ทำแก้ปัญหาจลนศาสตร์ผกผันสำหรับแขนกลที่มี 3 องศาอิสระใน 2 มิติโดยการสร้างข้อมูลรที่มีอินพุตเป็นตำแหน่งจุดปลายแขนกลและทิศทางของจุดปลายแขนกล ส่วนเอ้าพุตเป็นค่ามุมข้อต่อจำนวน 1,000 ชุดแสดงในรูปที่ 2.6 เพื่อนำไปสอนโครงข่ายประสาทเทียมแบบป้อนไปข้างหน้า (feed-forward neural network) ในรูปที่ 2.7

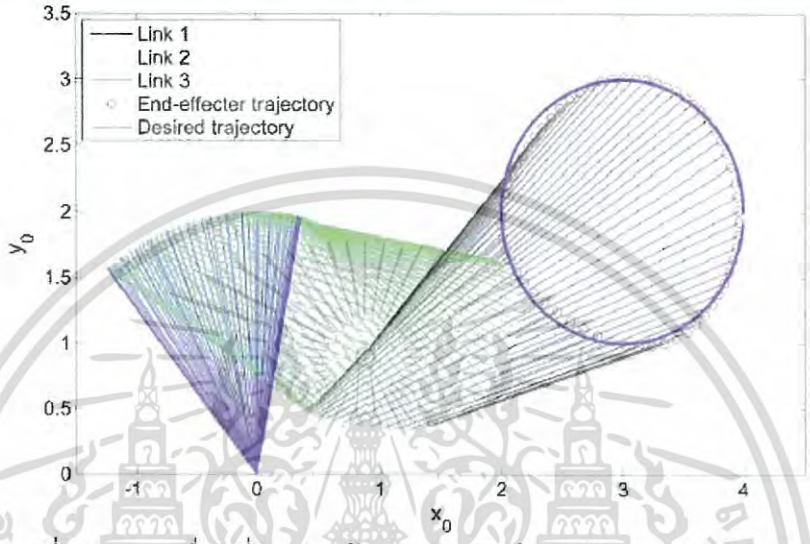


รูปที่ 2.6 ตำแหน่งข้อมูล 1,000 ชุดที่งานวิจัย [10] นำมาสอนโครงข่าย



รูปที่ 2.7 โครงสร้างของโครงข่ายประสาทเทียมในงานวิจัย [10]

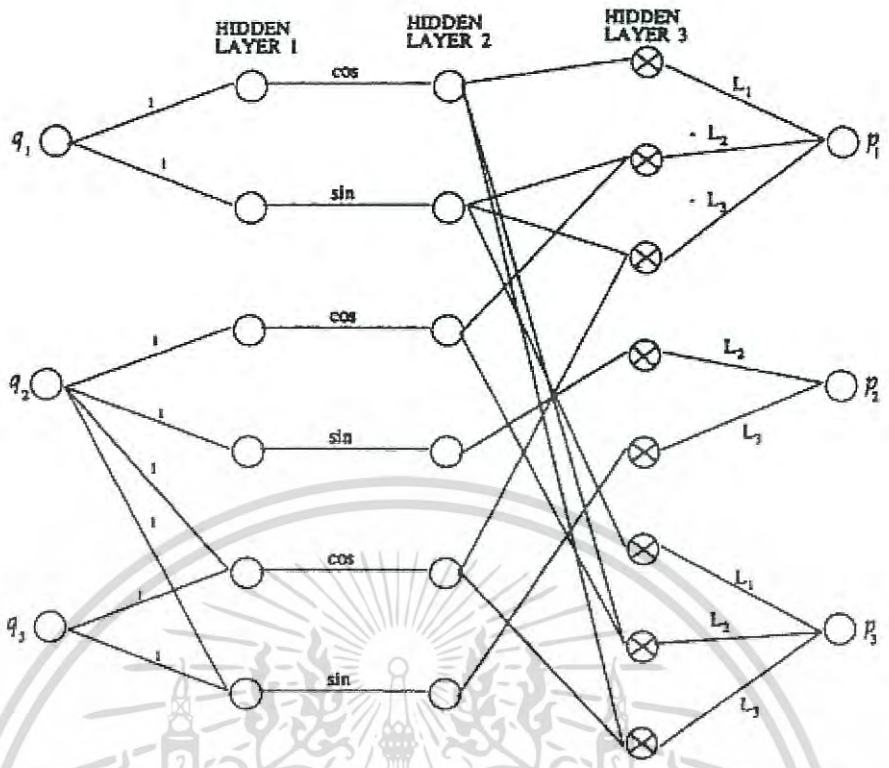
จากรูปที่ 2.7 จะพบว่าโครงข่ายมีอินพุตเป็นตำแหน่งจุดปลายแขนกลใน 2 มิติและเพิ่มค่าทิศทางจุดปลายแขนกลเข้ามาร่วมพิจารณาด้วย ซึ่งงานวิจัยนี้มุ่งหวังในการให้ค่ามุมคำตอบที่มีถ้าท่าทางที่เป็นธรรมชาติมากขึ้นดูได้จากรูปที่ 2.8 ที่งานวิจัยนี้ได้ทำการทดลองให้แขนกลติดตามเส้นกราฟแบบวงกลม จะเห็นได้ว่าลักษณะของแขนกลเคลื่อนที่แบบเป็นระเบียบแบบที่การเคลื่อนที่จริงควรเป็น แต่ความแม่นยำอาจยังไม่มากนัก



รูปที่ 2.8 การเคลื่อนที่ติดตามเส้นกราฟวงกลมใน 2 มิติของงานวิจัย [10]

2.2.2 การใช้โครงข่ายประสาทเทียมเลียนแบบสมการ

ในปี 1999 งานวิจัยโดย Sreenivas Tejomurtula และ Subhash Kak [1] ได้ทำการสร้างโครงข่ายประสาทเทียมจากสมการจลนศาสตร์ผันตรงโดยมีเอ้าพุตเป็นตำแหน่ง x, y, z ของจุดปลายแขนกลและอินพุตเป็นมุมข้อต่อ ดังแสดงในรูปที่ 2.9 ซึ่งสามารถสังเกตได้ว่าโครงสร้างของโครงข่ายประสาทเทียมนี้มีอินพุตและเอ้าพุตตรงกันข้ามกับโครงข่ายประสาทเทียมที่ใช้สร้างความสัมพันธ์ ในงานวิจัยนี้กำหนดให้ฟังก์ชันโคไซน์และไซน์จากสมการจลนศาสตร์ผันตรงเป็นน้ำหนักในชั้นซ่อนที่ 2 และค่าความยาวของแขนกลแต่ละท่อนเป็นค่าน้ำหนักในชั้นซ่อนที่ 4 จากนั้นทำการปรับปรุงค่าน้ำหนักในโครงข่ายด้วยการเรียนรู้แบบแพร่ย้อนกลับ (Backpropagation algorithm) เหมือนกับการปรับมุมแขนกลไปเรื่อยๆจนกระทั่งค่าตำแหน่งทั้ง 3 ใกล้เคียงกับค่าตำแหน่งเป้าหมาย ซึ่งข้อดีของวิธีนี้คือโครงข่ายประสาทเทียมสามารถหาคำตอบได้โดยไม่ต้องถูกสอนก่อนทำให้ขอบเขตของการหาคำตอบไม่ถูกจำกัดอยู่แค่ในข้อมูลที่นำมาสอนและโครงข่ายที่ใช้ก็มีขนาดเล็กกว่าโครงข่ายในแบบแรก แต่เนื่องจากงานวิจัยนี้ใช้ค่าน้ำหนักแบบไม่เป็นเชิงเส้น (ฟังก์ชันโคไซน์และไซน์) จึงอาจจะส่งผลให้การคำนวณเพื่อปรับปรุงค่าน้ำหนักมีความซับซ้อนมากกว่าปกติ ในส่วนของผลการทดลองนั้นงานวิจัยนี้ไม่ได้กล่าวถึงจำนวนรอบและเวลาในการคำนวณ, ขอบเขตของตำแหน่งเป้าหมาย, การรับประกันการลู่เข้า และยังแสดงผลลัพธ์การลู่เข้าคำตอบเพียงแค่ไม่กี่ตำแหน่งเท่านั้นดังแสดงในรูปที่ 2.10



รูปที่ 2.9 โครงข่ายประสาทเทียมจากสมการจลนศาสตร์ผันตรงในงานวิจัย [1]

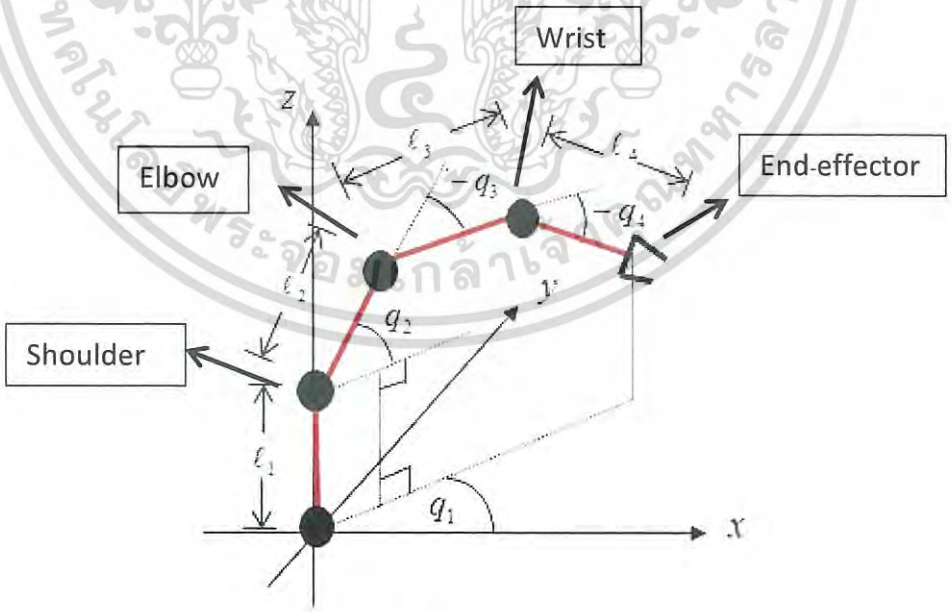
q_1	q_2	q_3	p_1	p_2	p_3
0.785398122	1.309012396	0.104674842	0.062346	0.062346	0.689635
0.785398163	1.805810804	−0.314177173	−0.032776	−0.032776	0.692658
0.785398163	2.356196362	0.523591635	−0.227452	−0.227452	0.515600
1.308984846	1.621695644	−0.418871863	0.010674	0.039834	0.689635
1.308984876	1.309000023	0.418872691	0.010674	0.039834	0.689635
1.570796327	1.544005759	−0.314157775	0.000000	0.056854	0.691276
1.570796327	1.308998440	0.314157757	0.000000	0.056854	0.691276
1.308992696	1.308999632	0.209431202	0.018779	0.070083	0.691276

รูปที่ 2.10 ตัวอย่างค่ามุมคำตอบที่ตำแหน่งจุดปลายแขนกลต่างๆของงานวิจัย [1]

บทที่ 3
หลักการที่เกี่ยวข้อง

3.1 จลนศาสตร์การเคลื่อนไหวของหุ่นยนต์ (Kinematics of robot)

หลักจลนศาสตร์เป็นศาสตร์ที่ศึกษาความสัมพันธ์ระหว่างตำแหน่งข้อต่อกับท่อนแขนกลเพื่อใช้ในการควบคุมแขนกลโดยไม่สนใจแรงที่มากระทำ ซึ่งหลักจลนศาสตร์สามารถแบ่งออกเป็น 2 ส่วน คือ จลนศาสตร์ผันตรง (Forward kinematics) กับ จลนศาสตร์ผกผัน (Inverse kinematics) แต่ก่อนอื่นเราต้องทำการนิยามตัวแปรต่างๆของแขนกลเสียก่อน โดยวิทยานิพนธ์ฉบับนี้จะทำการทดลองกับแขนกลที่มีข้อต่อแบบพับ (Revolute joint) ทั้งหมด รูปที่ 3.1 แสดงถึงชื่อตัวแปรที่ใช้เรียกในการคำนวณของแขนกลที่มี 4 องศาอิสระและมีข้อต่อแบบพับ ซึ่งประกอบไปด้วยข้อต่อหัวไหล่ (Shoulder joint), ข้อต่อข้อศอก (Elbow joint), ข้อต่อข้อมือ (Wrist joint) และจุดปลายแขนกล (End-effector) มีท่อนแขน 4 ท่อนยาว l_1, l_2, l_3, l_4 ตามลำดับและเนื่องจากข้อต่อทั้งหมดเป็นแบบพับ เงาของแขนกลที่ฉายลงบนระนาบ xy จึงเป็นเส้นตรง ส่วนตัวแปรมุม q_1 คือมุมระหว่างเงาของแขนกลบนระนาบ xy กับแกน x, มุม q_2 คือมุมระหว่างท่อนแขน l_2 กับระนาบ xy, มุม q_3 คือมุมระหว่างท่อนแขน l_3 กับแกนของท่อน l_2 และมุม q_4 คือมุมระหว่างท่อนแขน l_4 กับแกนของท่อนแขน l_3



รูปที่ 3.1 แขนกลที่มี 4 องศาอิสระและข้อต่อเป็นแบบพับในระบบพิกัดคาร์ทีเซียน 3 มิติ

3.1.1 จลนศาสตร์ผันตรง (Forward kinematics)

จลนศาสตร์ผันตรงคือการศึกษาเพื่อหาตำแหน่งจุดปลายแขนกลในรูปของมุมตามข้อต่อของแขนกล โดยสมการตำแหน่งของจุดปลายแขนกลที่อยู่ในรูปของตัวแปรมุมข้อต่อจะถูกเรียกว่าสมการจลนศาสตร์ผันตรง (Forward Kinematics Equations) ยกตัวอย่างแขนกลที่มี 4 องศาอิสระในระบบคาร์ทีเซียน 3 มิติในรูปที่ 3.1 สามารถบอกตำแหน่งของจุดปลายแขนกล $(x_{end}, y_{end}, z_{end})$ ได้ด้วยหลักตรีโกณมิติดังแสดงในสมการที่ (3.1), (3.2), (3.3) ตามลำดับ

$$x_{end} = \cos(q_1)[\ell_2 \cos(q_2) + \ell_3 \cos(q_2 + q_3) + \ell_4 \cos(q_2 + q_3 + q_4)] \quad (3.1)$$

$$y_{end} = \sin(q_1)[\ell_2 \cos(q_2) + \ell_3 \cos(q_2 + q_3) + \ell_4 \cos(q_2 + q_3 + q_4)] \quad (3.2)$$

$$z_{end} = \ell_1 + \ell_2 \sin(q_2) + \ell_3 \sin(q_2 + q_3) + \ell_4 \sin(q_2 + q_3 + q_4) \quad (3.3)$$

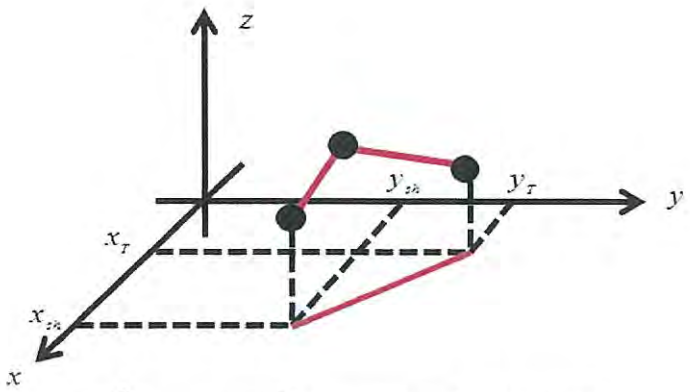
จากสมการพบว่าเราสามารถหาตำแหน่งของจุดปลายแขนกลในรูปของมุมข้อต่อได้โดยไม่ยุ่งยากซับซ้อน และมีรูปแบบที่ตรงไปตรงมาเมื่อจำนวนท่อนของแขนกลเพิ่มขึ้น การนำสมการจลนศาสตร์ผันตรงนี้ไปควบคุมแขนกลนั้นเหมือนกับการปรับค่ามุมตามข้อต่อแขนกลจนกระทั่งจุดปลายแขนกลไปถึงยังตำแหน่งเป้าหมาย (x_d, y_d, z_d)

3.1.2 จลนศาสตร์ผกผัน (Inverse kinematics)

จลนศาสตร์ผกผันจะมีวิธีการที่ตรงกันข้ามกับจลนศาสตร์ผันตรงคือ จลนศาสตร์ผกผันจะตำแหน่งเป้าหมายของจุดปลายแขนกลเป็นอินพุต จากนั้นจะทำการคำนวณหาค่ามุมที่เป็นไปได้เมื่อจุดปลายแขนกลอยู่ ณ ตำแหน่งเป้าหมาย หรือคือเราสามารถควบคุมแขนกลให้ไปยังวัตถุเป้าหมายได้โดยต้องการเพียงแค่ตำแหน่งของวัตถุนั้น วิธีนี้จึงสามารถควบคุมแขนกลได้อย่างสะดวกและรวดเร็ว แต่ขั้นตอนในการคำนวณหาคำตอบก็ยุ่งยากและซับซ้อนตามไปด้วยซึ่งวิธีแก้ปัญหาจลนศาสตร์ผกผันก็มีมากมายหลายทางดังกล่าวไว้ในบทที่ 2 รายงานฉบับนี้ได้ทำการแสดงตัวอย่างการแก้ปัญหาจลนศาสตร์ผกผันใน 3 มิติอย่างง่ายด้วยวิธีทางเรขาคณิตร่วมกับหลักตรีโกณมิติไว้ในหัวข้อ 3.1.2.1

3.1.2.1 ตัวอย่างการคำนวณจลนศาสตร์แบบผกผันใน 3 มิติด้วยวิธีทางเรขาคณิต

เราสมมติให้แขนกลอยู่บนระนาบใดๆ ซึ่งบนระนาบนั้นแขนกลเคลื่อนที่ได้เป็น 2 มิติ แต่ถ้าหากเราทำการหมุนระนาบที่แขนกลอยู่ไปรอบๆ จุดอ้างอิง (ณ ที่นี้คือจุดหัวไหล่) ก็จะส่งผลให้จุดปลายแขนกลเคลื่อนที่ไปยังตำแหน่งต่างๆใน 3 มิติได้ จากนั้นเราจะใช้วิธีการทางเรขาคณิตเพื่อหาค่ามุมฐาน มุมหัวไหล่ และมุมข้อศอกของแขนกล ในการหาค่ามุมหัวไหล่และมุมข้อศอกของแขนกลที่อยู่บนระนาบใดๆนั้นจะมีวิธีการคล้ายกับการคำนวณหาค่ามุมใน 2 มิติที่แขนกลอยู่บนระนาบ xy ส่วนการคำนวณหาค่ามุมฐานของแขนกลนั้นเราจะพิจารณาจากเส้นตรงที่เกิดจากการฉายแขนกลลงบนระนาบ xy



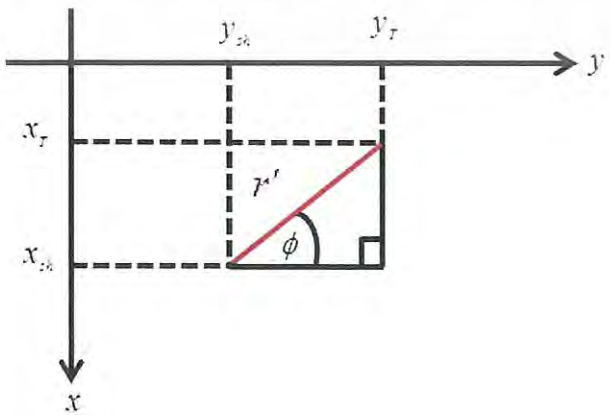
รูปที่ 3.2 แขนกลที่มี 3 องศาอิสระในพิกัด 3 มิติ

เริ่มคำนวณจากมุมฐานของแขนกล การคำนวณหาค่ามุมฐานเราจะต้องทำการกำหนดพื้นที่การทำงานของแขนกลเสียก่อน โดยรายงานฉบับนี้จะให้พื้นที่การทำงานของแขนกลเป็นเศษหนึ่งส่วนสี่ของทรงกลมหรือคือ ช่วงควอดรันต์ $(+x,+y,+z)$ กับ $(-x,+y,+z)$ ในระบบ Cartesian เพราะฉะนั้นมุมฐานของแขนกลจะมีค่าระหว่าง 0 ถึง 180 องศา โดยให้ 0 องศาอยู่ที่แกน $+x$ และมุม 180 องศาอยู่ที่แกน $-x$ จากนั้นเราจะกำหนดให้เส้นตรงที่เกิดจากการฉายแขนกลลงบนระนาบ xy นั้นแทนด้วยตัวแปร r' และทำมุม ϕ กับระนาบแกน y โดยมีจุดหัวไหล่เป็นจุดกำเนิด (x_{sh},y_{sh}) และตำแหน่งเป้าหมายเป็นจุดปลาย (x_T,y_T) ดูรูปที่ 3.3 ประกอบการคำนวณ ทำให้มุมฐานของแขนกลมีค่าเท่ากับ $90+\phi$ ซึ่งเราสามารถคำนวณหาค่า ϕ ได้จากสมการดังต่อไปนี้

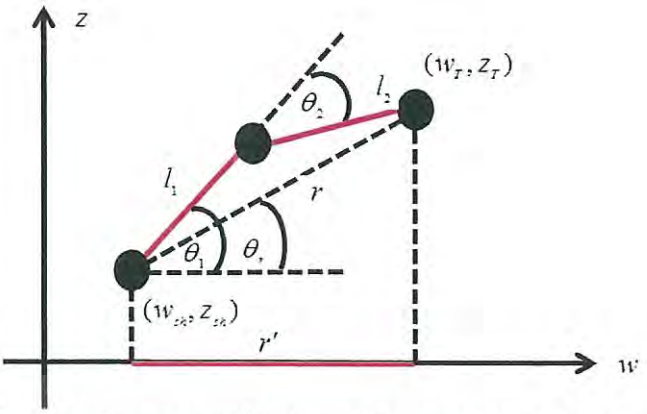
$$\tan(\phi)=\frac{x_{sh}-x_T}{y_T-y_{sh}}$$

$$\phi=\arctan\left(\frac{x_{sh}-x_T}{y_T-y_{sh}}\right) \tag{3.4}$$

ในการคำนวณหาค่ามุมหัวไหล่และมุมข้อศอกใน 2 มิติ เราทำการคำนวณโดยให้แขนกลอยู่ในระนาบ xy แต่การคำนวณใน 3 มิติ เราจะสมมติให้แขนกลนั้นอยู่บนระนาบใดๆ ณ ที่นี้ให้แทนด้วยระนาบ zw ดังแสดงในรูปที่ 3.4



รูปที่ 3.3 การคำนวณหาค่ามุมฐานของแขนกล



รูปที่ 3.4 การคำนวณมุมหัวไหล่และข้อศอกของแขนกลในระนาบ *zw*

จากรูปที่ 3.4 เราแทนเส้นตรงที่เกิดจากการฉายแขนกลลงบนระนาบ xy ด้วยตัวแปร r' โดยมีค่าเท่ากับ

$$r' = w_T - w_{sh} \tag{3.5}$$

เมื่อเราพิจารณารูปที่ 3.4 เราสามารถคำนวณหาค่า r' จาก 3 เหลี่ยมพีทาโกรัสได้เป็น

$$r' = w_T - w_{sh} = \sqrt{(x_{sh} - x_T)^2 + (y_T - y_{sh})^2} \tag{3.6}$$

ในระนาบ zw นี้เราสามารถคำนวณค่า r ที่เป็นระยะขจัดได้จากสมการ

$$r = \sqrt{(w_T - w_{sh})^2 + (z_T - z_{sh})^2} \tag{3.7}$$

แทนสมการ (3.6) ลงในสมการ (3.7) จะได้

$$r = \sqrt{\left(\sqrt{(x_{sh} - x_T)^2 + (y_T - y_{sh})^2}\right)^2 + (z_T - z_{sh})^2}$$

$$r = \sqrt{(x_{sh} - x_T)^2 + (y_T - y_{sh})^2 + (z_T - z_{sh})^2} \tag{3.8}$$

จากรูปที่ 3.4 มุมหัวไหล่มีค่าเท่ากับ θ_1 และมุมข้อศอกมีค่าเท่ากับ 180 - θ_2 เพราะฉะนั้นเราจะทำการหาค่าตัวแปร θ_1 และ θ_2

เราสามารถหาค่า θ_1 จากกฎของ cosine โดยคิดจาก 3 เหลี่ยม rl_1l_2 จะได้สมการ (3.9)

$$l_2^2 = r^2 + l_1^2 - 2r^2 l_1^2 \cos(\theta_1 - \theta_r)$$

$$\cos(\theta_1 - \theta_r) = \frac{l_2^2 - r^2 - l_1^2}{-2r^2 l_1^2}$$

$$\theta_1 = \arccos\left(\frac{l_2^2 - r^2 - l_1^2}{-2r^2 l_1^2}\right) + \theta_r \tag{3.9}$$

โดยที่ $\theta_r = \arctan\left(\frac{z_T - z_{sh}}{w_T - w_{sh}}\right)$ เพราะฉะนั้น

$$\theta_1 = \arccos\left(\frac{l_2^2 - r^2 - l_1^2}{-2r^2 l_1^2}\right) + \arctan\left(\frac{z_T - z_{sh}}{w_T - w_{sh}}\right) \quad (3.10)$$

เราจะหาค่า θ_2 จากกฎของ cosine โดยคิดจากสามเหลี่ยม rl_1l_2 เช่นเดียวกัน ได้สมการ (3.9)

$$r^2 = l_1^2 + l_2^2 - 2l_1^2 l_2^2 \cos(180 - \theta_2)$$

$$\cos(180 - \theta_2) = \left(\frac{r^2 - l_1^2 - l_2^2}{-2l_1^2 l_2^2}\right)$$

$$\theta_2 = 180 - \arccos\left(\frac{r^2 - l_1^2 - l_2^2}{-2l_1^2 l_2^2}\right) \quad (3.11)$$

จากการคำนวณเราจะได้สมการของมุมทั้ง 3 ของข้อต่อแขนกลดังนี้

สมการมุมฐาน

$$q_{Turntable} = 90 + \phi = 90 + \arctan\left(\frac{x_{sh} - x_T}{y_T - y_{sh}}\right) \quad (3.12)$$

สมการมุมหัวไหล่

$$q_{Shoulder} = \theta_1 = \arccos\left(\frac{l_2^2 - r^2 - l_1^2}{-2r^2 l_1^2}\right) + \arctan\left(\frac{z_T - z_{sh}}{w_T - w_{sh}}\right) \quad (3.13)$$

สมการมุมข้อศอก

$$q_{Elbow} = 180 - \theta_2 = 180 - \left[180 - \arccos\left(\frac{r^2 - l_1^2 - l_2^2}{-2l_1^2 l_2^2}\right)\right]$$

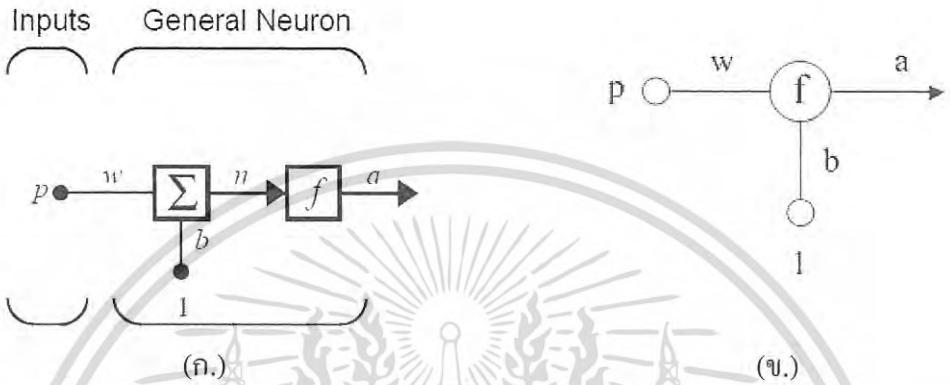
$$q_{Elbow} = \arccos\left(\frac{r^2 - l_1^2 - l_2^2}{-2l_1^2 l_2^2}\right) \quad (3.14)$$

3.2 โครงข่ายประสาทเทียม (Artificial neural network)

โครงข่ายประสาทเทียมเป็นโมเดลทางคณิตศาสตร์ที่มีโครงสร้างและการทำงานเหมือนกับโครงข่ายชีวภาพ (Biological neural networks) ในสมองมนุษย์ โดยโครงข่ายประสาทเทียมถูกสร้างขึ้นมาเพื่อเลียนแบบความสามารถของสมองมนุษย์อย่างเช่นจดจำรูปแบบหรือความสัมพันธ์ (Pattern recognitions) หรือทำการแยกแยะสิ่งของวัตถุต่างๆ (Classify) โดยโครงข่ายประสาทเทียมประกอบด้วยส่วนหลักๆ คือสถาปัตยกรรม (Architecture) และกฎการเรียนรู้ (Learning rule)

3.2.1 สถาปัตยกรรม (Architecture)

เราจะเริ่มต้นจากการอธิบายตัวแปรต่างๆในโครงข่ายประสาทเทียมก่อน โดยพิจารณาจากเซลล์ประสาทเทียมเซลล์เดียว (Single neuron) ในรูปที่ 3.5 โดยที่เซลล์ประสาทเทียมสามารถเขียนได้ทั้งแบบรูปที่ 3.5 (ก.) และ (ข.) โครงสร้างโครงข่ายประสาทเทียมที่มีเซลล์ประสาทเทียมเซลล์เดียวอาจเรียกได้ว่าเป็นโครงข่ายประสาทเทียมแบบเพอร์เซ็ปตรอนชั้นเดียว (Single-layer perceptron)



รูปที่ 3.5 เซลล์ประสาทเทียมเซลล์เดียว [5]

โดย p คือ อินพุต (Input)

w คือ ค่าน้ำหนัก (Weight)

b คือ ค่าไบอัส (bias)

n คือ อินพุตสุทธิ (Net input)

f คือ ทรานสเฟอร์ฟังก์ชัน (Transfer function)

a คือ เอ้าพุต (Output)

เซลล์ประสาทเทียมนี้จะทำการคูณอินพุตกับค่าน้ำหนักแล้วบวกกับค่าไบอัส ซึ่งผลลัพธ์ของการคำนวณนี้จะถูกเรียกว่า อินพุตสุทธิ ดังแสดงในสมการที่ (3.15)

$$n = wp + b \tag{3.15}$$

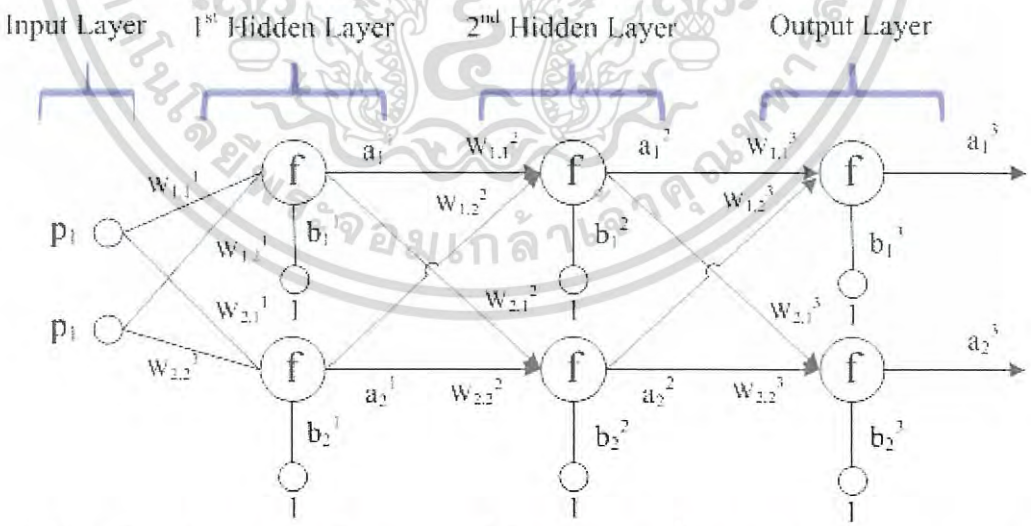
จากนั้นนำอินพุตสุทธินี้เข้าทรานสเฟอร์ฟังก์ชันดังสมการ (3.16) โดยทรานสเฟอร์ฟังก์ชันจะต้องเลือกใช้ให้สัมพันธ์กับชนิดกับปัญหา ซึ่งตัวอย่างของทรานสเฟอร์ฟังก์ชันแสดงไว้ในรูปที่ 3.6

$$a = f(n) \tag{3.16}$$

Name	Input/Output Relation	Icon	MATLAB Function
Hard Limit	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$		hardlim
Symmetrical Hard Limit	$a = -1 \quad n < 0$ $a = +1 \quad n \geq 0$		hardlims
Linear	$a = n$		purelin
Saturating Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n \leq 1$ $a = 1 \quad n > 1$		satlin
Symmetric Saturating Linear	$a = -1 \quad n < -1$ $a = n \quad -1 \leq n \leq 1$ $a = 1 \quad n > 1$		satlins
Log-Sigmoid	$a = \frac{1}{1+e^{-n}}$		logsig

รูปที่ 3.6 ตัวอย่างทรานสเฟอร์ฟังก์ชัน [5]

ขั้นต่อไปเราจะพิจารณาการกำหนดชื่อตัวแปรของโครงสร้างโครงข่ายประสาทเทียมที่มีเซลล์ประสาทเทียมเชื่อมต่อกันอยู่จำนวนมาก ดังรูปที่ 3.7



รูปที่ 3.7 โครงข่ายประสาทเทียมแบบเพอร์เซ็ปตรอนหลายชั้น (Multi-layer perceptron)

จากรูปโครงข่ายนี้จะสามารถแบ่งออกเป็น 3 ชั้นคือ ชั้นที่ 1 หรือชั้นซ่อนที่ 1 (1st hidden layer) ชั้นที่ 2 หรือชั้นซ่อนที่ 2 (2nd hidden layer) และชั้นที่ 3 หรือชั้นเอ้าพุต (Output layer) โดยแต่ละชั้นในรูปจะประกอบไปด้วยเซลล์ประสาทเทียม 2 เซลล์ ซึ่งตัวแปรต่างๆสามารถระบุได้ดังนี้

p_j คือ อินพุตตัวที่ j

$w_{i,j}^{m}$ คือ ค่าน้ำหนักของเซลล์ประสาทเทียมตัวที่ i ในชั้นที่ m สำหรับอินพุตตัวที่ j

b_i^m คือ ค่าไบอัสของเซลล์ประสาทเทียมตัวที่ i ในชั้นที่ m

f_i^m คือ ทรานสเฟอร์ฟังก์ชันของเซลล์ประสาทเทียมตัวที่ i ในชั้นที่ m

a_i^m คือ เอ้าพุตของเซลล์ประสาทเทียมตัวที่ i ในชั้นที่ m

โครงสร้างของโครงข่ายประสาทเทียมที่มีเซลล์ประสาทเทียมเชื่อมต่อกันหลายๆตัวและมีหลายชั้นนี้สามารถเรียกได้ว่าเป็นโครงข่ายประสาทเทียมแบบเพอร์เซ็ปตรอนหลายชั้น (Multi-layer perceptrons) ซึ่งเป็นโครงสร้างโครงข่ายประสาทเทียมแบบพื้นฐานที่นิยมใช้งานกัน

3.2.2 กฎการเรียนรู้ (Learning rule)

กฎการเรียนรู้ในโครงข่ายประสาทเทียม หมายถึงกระบวนการการปรับปรุงค่าน้ำหนัก w และค่าไบอัส b ของโครงข่ายเพื่อให้เอ้าพุตมีค่าเป็นไปตามความต้องการ กระบวนการเรียนรู้ของโครงข่ายประสาทเทียมนั้นมีอยู่มากมาย แต่สามารถแบ่งได้เป็น 3 ประเภทหลักๆคือ 1. การเรียนรู้แบบต้องถูกสอน (Supervised learning) วิธีนี้เป็นวิธีดั้งเดิมของการใช้งานโครงข่ายประสาทเทียมนั่นคือการนำชุดข้อมูลซึ่งประกอบด้วยอินพุตกับค่าเป้าหมายของอินพุตนั้นๆไปสอนโครงข่าย โดยนำอินพุตแต่ละตัวใส่ไปในระบบแล้วให้โครงข่ายปรับค่าน้ำหนักและไบอัสจนสามารถให้ค่าเอ้าพุตตรงกับที่ตั้งไว้ คล้ายกับการประมาณฟังก์ชันความสัมพันธ์ด้วยตัวอย่างชุดข้อมูล 2. การเรียนรู้แบบไม่ต้องถูกสอน (Unsupervised learning) วิธีนี้จะทำการปรับค่าน้ำหนักและไบอัสจากข้อมูลอินพุตโดยไม่ต้องการค่าเป้าหมาย กระบวนการนี้โครงข่ายจะทำการวิเคราะห์และหาความสัมพันธ์ของข้อมูลเองซึ่งมักถูกใช้ในการแบ่งกลุ่มของข้อมูลอินพุต 3. การเรียนรู้แบบเสริมกำลัง (Reinforcement learning) จะคล้ายกับการเรียนรู้แบบต้องถูกสอนแต่แทนที่จะให้ค่าเอ้าพุตที่ตรงกับค่าเป้าหมายของอินพุตนั้น โครงข่ายประเภทนี้จะให้เอ้าพุตเป็นคะแนนความถูกต้องแทน วิธีการนี้มักถูกประยุกต์ใช้ในระบบควบคุม เมื่อเรารู้จักกฎการเรียนรู้ทั้ง 3 ประเภทแล้วต่อไปเราจะแสดงตัวอย่างการปรับปรุงค่าน้ำหนักและไบอัสด้วยการเรียนรู้แบบแพร่ย้อนกลับด้วยอัตราการเรียนรู้คงที่และไม่คงที่

3.2.2.1 การเรียนรู้แบบแพร่ย้อนกลับ (Backpropagation algorithm)

การเรียนรู้แบบแพร่ย้อนกลับใช้หลักการ Steepest descent ในการปรับปรุงค่าน้ำหนักและไบอัส โดยการหาค่าทิศทางความชันของฟังก์ชันค่าคลาดเคลื่อนที่อยู่ในรูปของค่าน้ำหนักแล้วเปลี่ยนแปลงค่าน้ำหนักและไบอัสในทางตรงข้ามกับความชันจนอยู่ในตำแหน่งที่ให้ค่าคลาดเคลื่อนน้อยที่สุด สมการที่ (3.17), (3.18) แสดงสมการปรับปรุงค่าน้ำหนักและไบอัสของวิธีการเรียนรู้แบบ Steepest descent

$$w_{i,j}^{m}(k+1) = w_{i,j}^{m}(k) - \alpha \frac{\partial E}{\partial w_{i,j}^{m}} \tag{3.17}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha \frac{\partial E}{\partial b_i^m} \tag{3.18}$$

โดย α คือค่าอัตราการเรียนรู้

$\frac{\partial E}{\partial w_{i,j}^m}$ คือความชันของฟังก์ชันค่าคลาดเคลื่อนโดยกำหนดให้ค่าน้ำหนักเป็นตัวแปร

$\frac{\partial E}{\partial b_i^m}$ คือความชันของฟังก์ชันค่าคลาดเคลื่อนโดยกำหนดให้ค่าไบอัสเป็นตัวแปร

เทอมอนุพันธ์ของฟังก์ชันค่าคลาดเคลื่อนสามารถจัดให้อยู่ในรูปที่ง่ายขึ้นด้วยกฎลูกโซ่ ตัวอย่างการกระจายเทอมอนุพันธ์ของกฎลูกโซ่เมื่อเราทำการอนุพันธ์เอ้าพุต a ในสมการ (3.16) เทียบกับค่าน้ำหนัก w สามารถกระจายเทอมอนุพันธ์ออกมาได้ดังสมการที่ (3.19)

$$\frac{da}{dw}=\frac{df(n(w))}{dw}=\frac{df(n)}{dn}\times\frac{dn(w)}{dw} \tag{3.19}$$

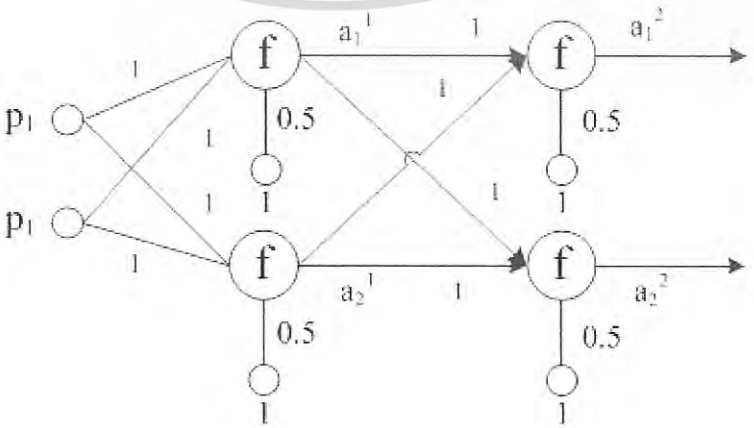
เพราะฉะนั้นหากกำหนดให้ฟังก์ชันค่าคลาดเคลื่อนเป็นฟังก์ชันผลรวมกำลังสอง ค่าคลาดเคลื่อน ณ รอบที่ k ของการคำนวณแสดงได้ในสมการ (3.20) และสามารถกระจายเทอมอนุพันธ์ของค่าคลาดเคลื่อนใน (3.20) เทียบกับตัวแปร w ด้วยกฎลูกโซ่ได้เป็นสมการที่ (3.21) ซึ่งเทอมอนุพันธ์ของค่าคลาดเคลื่อนนี้จะแตกต่างกันในแต่ละชั้นของโครงข่ายดูได้จากตัวอย่างที่ 3.1

$$E=\sum(t(k)-a(k))^2 \tag{3.20}$$

โดย $t(k)$ คือค่าเป้าหมายของเอ้าพุต $a(k)$

$$\frac{\partial E}{\partial w_{i,j}^m}=\frac{\partial E}{\partial a_i^m}\times\frac{\partial a_i^m}{\partial n_i^m}\times\frac{\partial n_i^m}{\partial w_{i,j}^m} \tag{3.21}$$

ตัวอย่างที่ 3.1 การคำนวณเพื่อปรับปรุงค่าน้ำหนักและไบอัสของโครงข่ายประสาทเทียมในรูปที่ 3.8 ด้วยการเรียนรู้แบบแพร่ย้อนกลับด้วยอัตราการเรียนรู้คงที่ โดยกำหนดให้ target $=\begin{bmatrix}10 & 5\end{bmatrix}^T$, input $=\begin{bmatrix}1.5 & 2\end{bmatrix}^T$ ทรานสเฟอร์ฟังก์ชันเป็น purelin ($f(n)=n$) ทั้งหมด อัตราการเรียนรู้ (α) =0.01 และใช้ฟังก์ชันค่าคลาดเคลื่อนแบบผลรวมกำลังสอง



รูปที่ 3.8 โครงข่ายประสาทเทียมสำหรับตัวอย่าง 3.1

ระบบจะทำการปรับปรุงค่าน้ำหนักและไบอัสวนซ้ำไปจนกว่าค่าเอ้าพุตของโครงข่ายจะตรงกับค่าเป้าหมาย ขั้นแรกทำการคำนวณหาค่าคลาดเคลื่อนที่ชั้นเอ้าพุต

$$a^1(1)=\begin{bmatrix} f^1\left(n_1^1\right) \\ f^1\left(n_2^1\right) \end{bmatrix}=\begin{bmatrix} n_1^1 \\ n_2^1 \end{bmatrix}=\begin{bmatrix} w_{1,1}^1 p_1+w_{1,2}^1 p_2+b_1^1 \\ w_{2,1}^1 p_1+w_{2,2}^1 p_2+b_2^1 \end{bmatrix} \tag{3.22}$$

$$a^1(1)=\begin{bmatrix} 1.5+2+0.5 \\ 1.5+2+0.5 \end{bmatrix}=\begin{bmatrix} 4 \\ 4 \end{bmatrix} \tag{3.23}$$

$$a^2(1)=\begin{bmatrix} f^2\left(n_1^2\right) \\ f^2\left(n_2^2\right) \end{bmatrix}=\begin{bmatrix} n_1^2 \\ n_2^2 \end{bmatrix}=\begin{bmatrix} w_{1,1}^2 a_1^1+w_{1,2}^2 a_2^1+b_1^2 \\ w_{2,1}^2 a_1^1+w_{2,2}^2 a_2^1+b_2^2 \end{bmatrix} \tag{3.24}$$

$$a^2(1)=\begin{bmatrix} 4+4+0.5 \\ 4+4+0.5 \end{bmatrix}=\begin{bmatrix} 8.5 \\ 8.5 \end{bmatrix} \tag{3.25}$$

นำค่าเป้าหมายมาลบกับค่าเอ้าพุตได้คลาดเคลื่อนดังสมการที่ (3.26)

$$e(1)=\begin{bmatrix} t_1-a_1^2 \\ t_2-a_2^2 \end{bmatrix}=\begin{bmatrix} 10-8.5 \\ 5-8.5 \end{bmatrix}=\begin{bmatrix} 1.5 \\ -3.5 \end{bmatrix} \tag{3.26}$$

การเรียนรู้แบบแพร่ย้อนกลับจะทำการปรับปรุงค่าน้ำหนักและไบอัสจากชั้นเอ้าพุตไปสู่ชั้นอินพุต เราจึงทำการปรับปรุงค่าน้ำหนักในชั้นที่ 2 ก่อนโดยเริ่มจากค่าน้ำหนัก $w_{1,1}{}^2, w_{1,2}{}^2$ และไบอัส $b_1{}^2$ ในเซลล์ประสาทเทียมตัวที่ 1 การคำนวณจะทำการคิดเทอมอนุพันธ์ทีละเทอม

$$\begin{aligned} \frac{\partial E}{\partial w_{1,1}^2}&=\frac{\partial E}{\partial a_1^2}\times\frac{\partial a_1^2}{\partial n_1^2}\times\frac{\partial n_1^2}{\partial w_{1,1}^2} \\ \frac{\partial E}{\partial w_{1,2}^2}&=\frac{\partial E}{\partial a_1^2}\times\frac{\partial a_1^2}{\partial n_1^2}\times\frac{\partial n_1^2}{\partial w_{1,2}^2} \\ \frac{\partial E}{\partial b_1^2}&=\frac{\partial E}{\partial a_1^2}\times\frac{\partial a_1^2}{\partial n_1^2}\times\frac{\partial n_1^2}{\partial b_1^2} \end{aligned} \tag{3.27}$$

1.พิจารณาเทอม $\frac{\partial E}{\partial a_1^2}$

เนื่องจากค่าน้ำหนัก $w_{1,1}{}^2$ มีผลต่อค่าเอ้าพุต $a_1{}^2$ เท่านั้น จึงคิดแต่ค่าคลาดเคลื่อนของเอ้าพุต $a_1{}^2$

$$\frac{\partial E}{\partial a_1^2}=\frac{\partial}{\partial a_1^2}(t_1-a_1^2)^2=2(t_1-a_1^2)(0-\frac{\partial a_1^2}{\partial a_1^2})=-2(t_1-a_1^2) \tag{3.28}$$

2.พิจารณาเทอม $\frac{\partial a_1^2}{\partial n_1^2}$

เนื่องจากทรานสเฟอร์ฟังก์ชันเป็น purelin ($f(n)=n$) เพราะฉะนั้นอนุพันธ์ของทรานสเฟอร์ฟังก์ชันได้เป็น $\frac{\partial f_{purelin}(n)}{\partial n}=\frac{\partial n}{\partial n}=1$

$$\frac{\partial a_1^2}{\partial n_1^2}=\frac{\partial f_1^2(n_1^2)}{\partial n_1^2}=1 \tag{3.29}$$

3.พิจารณาเทอม $\frac{\partial n_1^2}{\partial w_{1,1}^2}$, $\frac{\partial n_1^2}{\partial w_{1,2}^2}$ และ $\frac{\partial n_1^2}{\partial b_1^2}$

เมื่อแทนค่า $n_1{}^2$ ด้วยสมการที่ (3.24) ได้ค่าอนุพันธ์เป็น

$$\begin{aligned}\frac{\partial n_1^2}{\partial w_{1,1}^2}&=\frac{\partial}{\partial w_{1,1}^2}\left(w_{1,1}^2a_1^1+w_{1,2}^2a_2^1+b_1^2\right)=a_1^1\\ \frac{\partial n_1^2}{\partial w_{1,2}^2}&=\frac{\partial}{\partial w_{1,2}^2}\left(w_{1,1}^2a_1^1+w_{1,2}^2a_2^1+b_1^2\right)=a_2^1\\ \frac{\partial n_1^2}{\partial b_1^2}&=\frac{\partial}{\partial b_1^2}\left(w_{1,1}^2a_1^1+w_{1,2}^2a_2^1+b_1^2\right)=1\end{aligned} \tag{3.30}$$

โดยส่วนใหญ่แล้วมักกำหนดให้เทอมอนุพันธ์ $\frac{\partial E}{\partial a_i^m}\times\frac{\partial a_i^m}{\partial n_i^m}$ เป็นค่าความไว s (Sensitivity) หรือมีความหมายว่าความไวของการเปลี่ยนแปลงค่า E ต่อตัวแปรอินพุตสุทธิตัวที่ i ในชั้นที่ m

$$s^m=\frac{\partial E}{\partial a_i^m}\times\frac{\partial a_i^m}{\partial n_i^m} \tag{3.31}$$

ค่าความไวในชั้นที่ 2 ของโครงข่ายในตัวอย่างเป็น

$$\begin{aligned}s^2&=\begin{bmatrix}\frac{\partial E}{\partial a_1^2}\times\frac{\partial a_1^2}{\partial n_1^2}\\ \frac{\partial E}{\partial a_2^2}\times\frac{\partial a_2^2}{\partial n_2^2}\end{bmatrix}=\begin{bmatrix}-2(t_1-a_1^2)\times\frac{\partial a_1^2}{\partial n_1^2}\\ -2(t_2-a_2^2)\times\frac{\partial a_2^2}{\partial n_2^2}\end{bmatrix}\\ s^2&=\begin{bmatrix}-2(1.5)\times1\\ -2(-3.5)\times1\end{bmatrix}=\begin{bmatrix}-3\\ 7\end{bmatrix}\end{aligned} \tag{3.32}$$

แทนค่าความไวในชั้นที่ 2 และสมการ (3.30) ในสมการที่ (3.27) จะได้เทอมอนุพันธ์ของค่าคลาดเคลื่อนเทียบกับค่าน้ำหนัก $w_{1,1}{}^2$, $w_{1,2}{}^2$ และไบอัส $b_1{}^2$ อยู่ในรูปที่ง่ายขึ้นดังสมการที่ (3.33)

$$\begin{aligned}\frac{\partial E}{\partial w_{1,1}^2}&=s_1^2\frac{\partial n_1^2}{\partial w_{1,1}^2}=s_1^2\times a_1^1\\ \frac{\partial E}{\partial w_{1,2}^2}&=s_1^2\frac{\partial n_1^2}{\partial w_{1,2}^2}=s_1^2\times a_2^1\\ \frac{\partial E}{\partial b_1^2}&=s_1^2\frac{\partial n_1^2}{\partial b_1^2}=s_1^2\times1\end{aligned} \tag{3.33}$$

ทำกระบวนการเดียวกันนี้กับค่าน้ำหนักและไบอัสของเซลล์ประสาทเทียมตัวที่ 2 ในชั้นที่ 2 จะได้เทอมอนุพันธ์ของค่าคลาดเคลื่อนเทียบค่าน้ำหนักและไบอัสเหมือนสมการที่ (3.33) แต่ใช้ค่าความไวคนละค่ากัน

$$\begin{aligned}\frac{\partial E}{\partial w_{2,1}^2} &= s_2^2 \times a_1^1 \\ \frac{\partial E}{\partial w_{2,2}^2} &= s_2^2 \times a_2^1 \\ \frac{\partial E}{\partial b_2^2} &= s_2^2 \times 1\end{aligned} \tag{3.34}$$

แทนค่าเทอมอนุพันธ์สมการ (3.33) และ (3.34) ในสมการที่ (3.17) และ (3.18) จะได้ค่าน้ำหนักและไบอัสในชั้นที่ 2 ใหม่ดังแสดงในสมการที่ (3.35) และ (3.36) ตามลำดับ

ค่าน้ำหนักในชั้นที่ 2 ใหม่

$$w^2(2) = \begin{bmatrix} w_{1,1}^2(1) - \alpha \dfrac{\partial E}{\partial w_{1,1}^2} \\ w_{1,2}^2(1) - \alpha \dfrac{\partial E}{\partial w_{1,2}^2} \\ w_{2,1}^2(1) - \alpha \dfrac{\partial E}{\partial w_{2,1}^2} \\ w_{2,2}^2(1) - \alpha \dfrac{\partial E}{\partial w_{2,2}^2} \end{bmatrix} = \begin{bmatrix} w_{1,1}^2(1) - \alpha \times s_1^2 \times a_1^1 \\ w_{1,2}^2(1) - \alpha \times s_1^2 \times a_2^1 \\ w_{2,1}^2(1) - \alpha \times s_2^2 \times a_1^1 \\ w_{2,2}^2(1) - \alpha \times s_2^2 \times a_2^1 \end{bmatrix} \tag{3.35}$$

$$w^2(2) = \begin{bmatrix} 1 - 0.01 \times -3 \times 4 \\ 1 - 0.01 \times -3 \times 4 \\ 1 - 0.01 \times 7 \times 4 \\ 1 - 0.01 \times 7 \times 4 \end{bmatrix} = \begin{bmatrix} 1.12 \\ 1.12 \\ 0.72 \\ 0.72 \end{bmatrix}$$

ค่าไบอัสในชั้นที่ 2 ใหม่

$$b^2(2) = \begin{bmatrix} b_1^2(1) - \alpha \dfrac{\partial E}{\partial b_1^2} \\ b_2^2(1) - \alpha \dfrac{\partial E}{\partial b_2^2} \end{bmatrix} = \begin{bmatrix} b_1^2(1) - \alpha \times s_1^2 \\ b_2^2(1) - \alpha \times s_2^2 \end{bmatrix} \tag{3.36}$$

$$b^2(2) = \begin{bmatrix} 0.5 - 0.01 \times -3 \\ 0.5 - 0.01 \times 7 \end{bmatrix} = \begin{bmatrix} 0.53 \\ 0.43 \end{bmatrix}$$

ต่อไปทำการปรับปรุงค่าน้ำหนักและไบอัสในชั้นที่ 1 โดยเริ่มจากค่าน้ำหนัก $w_{1,1}{}^1$, $w_{1,2}{}^1$ และไบอัส $b_1{}^1$ ในเซลล์ประสาทเทียมตัวที่ 1 ก่อน

$$\frac{\partial E}{\partial w_{1,1}^1} = \frac{\partial E}{\partial a_1^1} \times \frac{\partial a_1^1}{\partial n_1^2} \times \frac{\partial n_1^1}{\partial w_{1,1}^1}$$
$$\frac{\partial E}{\partial w_{1,2}^1} = \frac{\partial E}{\partial a_1^1} \times \frac{\partial a_1^1}{\partial n_1^1} \times \frac{\partial n_1^1}{\partial w_{1,2}^1} \qquad (3.37)$$
$$\frac{\partial E}{\partial b_1^1} = \frac{\partial E}{\partial a_1^1} \times \frac{\partial a_1^1}{\partial n_1^1} \times \frac{\partial n_1^1}{\partial b_1^1}$$

1.พิจารณาเทอม $\frac{\partial E}{\partial a_1^1}$

เนื่องจากค่าน้ำหนัก $w_{1,1}{}^1$, $w_{1,2}{}^1$ และไบอัส $b_1{}^1$ มีผลต่อค่าเอ้าพุต $a_1{}^1$ ซึ่งเอ้าพุต $a_1{}^1$ ผลต่อค่าเอ้าพุต $a_1{}^2$ และ $a_2{}^2$ เทอมอนุพันธ์ $\partial E / \partial a_1^1$ จึงต้องคิดค่าคลาดเคลื่อนของทั้งสองเอ้าพุต

$$\frac{\partial E}{\partial a_1^1} = \frac{\partial}{\partial a_1^1}\left[(t_1 - a_1^2)^2 + (t_2 - a_2^2)^2\right]$$
$$\frac{\partial E}{\partial a_1^1} = 2(t_1 - a_1^2)(-\frac{\partial a_1^2}{\partial a_1^1}) + 2(t_2 - a_2^2)(-\frac{\partial a_2^2}{\partial a_1^1})$$
$$\frac{\partial E}{\partial a_1^1} = 2(t_1 - a_1^2)\left(-\frac{\partial a_1^2}{\partial n_1^2}\frac{\partial n_1^2}{\partial a_1^1}\right) + 2(t_2 - a_2^2)\left(-\frac{\partial a_2^2}{\partial n_2^2}\frac{\partial n_2^2}{\partial a_1^1}\right)$$

แทนค่า $n_1{}^2$, $n_2{}^2$ ด้วยสมการที่ (3.24) เพื่อหาค่าเทอมอนุพันธ์ $\partial n_1^2 / \partial a_1^1$ และ $\partial n_2^2 / \partial a_1^1$

$$\frac{\partial E}{\partial a_1^1} = -2(t_1 - a_1^2)\frac{\partial a_1^2}{\partial n_1^2} w_{1,1}^2 + -2(t_2 - a_2^2)\frac{\partial a_2^2}{\partial n_2^2} w_{2,1}^2 \qquad (3.38)$$

จากการสังเกตสมการ (3.38) พบว่า 2 เทอมแรกของสมการคือความไวในชั้นที่ 2 จึงจัดรูปใหม่ได้เป็น

$$\frac{\partial E}{\partial a_1^1} = s_1^2 w_{1,1}^2 + s_2^2 w_{2,1}^2 \qquad (3.39)$$

2.พิจารณาเทอม $\frac{\partial a_1^1}{\partial n_1^1}$

เนื่องจากทรานสเฟอร์ฟังก์ชันเป็น purelin ($f(n)=n$) เพราะฉะนั้นอนุพันธ์ของทรานสเฟอร์ฟังก์ชันจะเป็น $\frac{\partial f_{purelin}(n)}{\partial n} = \frac{\partial n}{\partial n} = 1$

$$\frac{\partial a_1^1}{\partial n_1^1} = \frac{\partial f(n_1^1)}{\partial n_1^1} = 1 \qquad (3.40)$$

3.พิจารณาเทอม $\frac{\partial n_1^1}{\partial w_{1,1}^1}$, $\frac{\partial n_1^1}{\partial w_{1,2}^1}$ และ $\frac{\partial n_1^1}{\partial b_1^1}$

เมื่อแทนค่า n_1^1 ด้วยสมการที่ (3.22) จะได้ค่าอนุพันธ์เป็น

$$\frac{\partial n_1^1}{\partial w_{1,1}^1} = \frac{\partial}{\partial w_{1,1}^1}\left(w_{1,1}^1 p_1 + w_{1,2}^1 p_2 + b_1^1\right) = p_1$$
$$\frac{\partial n_1^1}{\partial w_{1,2}^1} = \frac{\partial}{\partial w_{1,2}^1}\left(w_{1,1}^1 p_1 + w_{1,2}^1 p_2 + b_1^1\right) = p_2 \qquad (3.41)$$
$$\frac{\partial n_1^1}{\partial b_1^1} = \frac{\partial}{\partial b_1^1}\left(w_{1,1}^1 p_1 + w_{1,2}^1 p_2 + b_1^1\right) = 1$$

ค่าความไวในชั้นที่ 1

$$s^1 = \begin{bmatrix} \frac{\partial E}{\partial a_1^1} \times \frac{\partial a_1^1}{\partial n_1^1} \\ \frac{\partial E}{\partial a_2^1} \times \frac{\partial a_2^1}{\partial n_2^1} \end{bmatrix} = \begin{bmatrix} \left(s_1^2 w_{1,1}^2 + s_2^2 w_{2,1}^2\right) \times \frac{\partial a_1^1}{\partial n_1^1} \\ \left(s_1^2 w_{1,2}^2 + s_2^2 w_{2,2}^2\right) \times \frac{\partial a_2^1}{\partial n_2^1} \end{bmatrix} \qquad (3.42)$$
$$s^1 = \begin{bmatrix} -3(1) + 7(1) \\ -3(1) + 7(1) + \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

แทนค่าความไวในชั้นที่ 1 และสมการ (3.41) ในสมการที่ (3.37) จะได้เทอมอนุพันธ์ของค่าคลาดเคลื่อนเทียบกับค่าน้ำหนัก $w_{1,1}^1$, $w_{1,2}^1$และไบอัส b_1^1 อยู่ในรูปที่ง่ายขึ้นดังสมการที่ (3.43)

$$\frac{\partial E}{\partial w_{1,1}^1} = s_1^1 \frac{\partial n_1^1}{\partial w_{1,1}^1} = s_1^1 \times p_1$$
$$\frac{\partial E}{\partial w_{1,2}^1} = s_1^1 \frac{\partial n_1^1}{\partial w_{1,2}^1} = s_1^1 \times p_2 \qquad (3.43)$$
$$\frac{\partial E}{\partial b_1^1} = s_1^1 \frac{\partial n_1^1}{\partial b_1^1} = s_1^1 \times 1$$

ทำกระบวนการเดียวกันนี้กับค่าน้ำหนักและไบอัสของเซลล์ประสาเทียมตัวที่ 2 ในชั้นที่ 1 จะได้เทอมอนุพันธ์ของค่าคลาดเคลื่อนเทียบค่าน้ำหนักและไบอัสดังคล้ายสมการ (3.43) แต่ค่าความไวคนละตัวกัน

$$\frac{\partial E}{\partial w_{2,1}^1} = s_2^1 \times p_1$$
$$\frac{\partial E}{\partial w_{2,2}^1} = s_2^1 \times p_2 \qquad (3.44)$$
$$\frac{\partial E}{\partial b_2^1} = s_2^1 \times 1$$

แทนค่าเทอมอนุพันธ์สมการที่ (3.43) และ (3.44) ไปในสมการปรับปรุงค่าน้ำหนักและไบอัส (3.17) และ (3.18) จะได้ค่าน้ำหนักและไบอัสในชั้นที่ 1 ใหม่ดังแสดงในสมการที่ (3.45) และ (3.46) ตามลำดับ

ค่าน้ำหนักในชั้นที่ 1 ใหม่

$$w^1(2)=\begin{bmatrix} w^1_{1,1}(1)-\alpha\dfrac{\partial E}{\partial w^1_{1,1}} \\ w^1_{1,2}(1)-\alpha\dfrac{\partial E}{\partial w^1_{1,2}} \\ w^1_{2,1}(1)-\alpha\dfrac{\partial E}{\partial w^1_{2,1}} \\ w^1_{2,2}(1)-\alpha\dfrac{\partial E}{\partial w^1_{2,2}} \end{bmatrix}=\begin{bmatrix} w^1_{1,1}(1)-\alpha\times s^1_1\times p_1 \\ w^1_{1,2}(1)-\alpha\times s^1_1\times p_2 \\ w^1_{2,1}(1)-\alpha\times s^1_2\times p_1 \\ w^1_{2,2}(1)-\alpha\times s^1_2\times p_2 \end{bmatrix} \tag{3.45}$$

$$w^1(2)=\begin{bmatrix} 1-0.01\times 4\times 1.5 \\ 1-0.01\times 4\times 2 \\ 1-0.01\times 4\times 1.5 \\ 1-0.01\times 4\times 2 \end{bmatrix}=\begin{bmatrix} 0.94 \\ 0.92 \\ 0.94 \\ 0.92 \end{bmatrix}$$

ค่าไบอัสในชั้นที่ 1 ใหม่

$$b^1(2)=\begin{bmatrix} b^1_1(1)-\alpha\dfrac{\partial E}{\partial b^1_1} \\ b^1_2(1)-\alpha\dfrac{\partial E}{\partial b^1_2} \end{bmatrix}=\begin{bmatrix} b^1_1(1)-\alpha\times s^1_1 \\ b^1_2(1)-\alpha\times s^1_2 \end{bmatrix} \tag{3.46}$$

$$b^1(2)=\begin{bmatrix} 0.5-0.01\times 4 \\ 0.5-0.01\times 4 \end{bmatrix}=\begin{bmatrix} 0.46 \\ 0.46 \end{bmatrix}$$

เมื่อทดลองหาเอ้าพุตของโครงข่ายด้วยค่าน้ำหนักและไบอัสใหม่จะพบว่าค่าคลาดเคลื่อนลดลงจากรอบแรก

$$a^1(2)=\begin{bmatrix} (0.94)(1.5)+(0.92)(2)+0.46 \\ (0.94)(1.5)+(0.92)(2)+0.46 \end{bmatrix}=\begin{bmatrix} 3.71 \\ 3.71 \end{bmatrix}$$

$$a^2(2)=\begin{bmatrix} (1.12)(3.71)+(1.12)(3.71)+0.53 \\ (0.72)(3.71)+(0.72)(3.71)+0.43 \end{bmatrix}=\begin{bmatrix} 8.84 \\ 5.77 \end{bmatrix}$$

$$e(2)=\begin{bmatrix} t_1-a^2_1 \\ t_2-a^2_2 \end{bmatrix}=\begin{bmatrix} 10-8.84 \\ 5-5.77 \end{bmatrix}=\begin{bmatrix} 1.16 \\ -0.77 \end{bmatrix} \tag{3.47}$$

จากการคำนวณตัวอย่างเราสามารถสรุปสมการการปรับปรุงค่าน้ำหนักและไบอัสด้วยการเรียนรู้แบบแพร่ย้อนกลับได้เป็นสมการที่ (3.48) และ (3.49)

ค่าน้ำหนักใหม่และค่าไบอัสใหม่

$$w_{i,j}^{m}(k+1) = w_{i,j}^{m}(k) - \alpha \times s_i^m \times a_i^{m-1} \tag{3.48}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha \times s_i^m \tag{3.49}$$

โดยที่ค่าความไว s^m

$$s_i^m = -2 \times f_i'^m(n_i^m) \times (t_i - a_i^m) \qquad \text{สำหรับชั้นเอ้าพุต} \tag{3.50}$$

$$s_i^m = f_i'^m(n_i^m) \times \sum_{k=1}^{R} w_{k,i}^{m+1} s_k^{m+1} \qquad \text{สำหรับชั้นทั่วไป} \tag{3.51}$$

โดยที่ R คือจำนวนเซลล์ประสาทเทียมในชั้น $m+1$ ที่เชื่อมต่อกับเซลล์ประสาทเทียม i ในชั้นที่ m

3.2.2.2 การเรียนรู้แบบแพร่ย้อนกลับด้วยอัตราการเรียนรู้ไม่คงที่ (Variable learning rate backpropagation algorithm)

ในวิธีการเรียนรู้แบบแพร่ย้อนกลับด้วยอัตราการเรียนรู้คงที่นั้นค่าอัตราการเรียนรู้เป็นส่วนสำคัญในการลู่เข้าอย่างมาก เปรียบเสมือนเป็นขนาดในการก้าวเข้าสู่คำตอบ ซึ่งหากกำหนดค่ามากเกินไปก็จะก้าวข้ามคำตอบไปมาไม่สามารถลู่เข้าคำตอบได้ และถ้าหากก้าวเล็กเกินไปก็จะลู่เข้าสู่คำตอบได้ช้า ดังนั้นหากเราทำการเพิ่มหรือลดขนาดอัตราการเรียนรู้ในแต่ละรอบของการคำนวณได้ก็จะสามารถแก้ปัญหาเรื่องการกำหนดอัตราการเรียนรู้ที่เหมาะสมและยังเพิ่มความเร็วในการลู่เข้าคำตอบของการเรียนรู้แบบแพร่ย้อนกลับอีกด้วย แต่ก่อนจะทำการปรับค่าอัตราการเรียนรู้เราต้องรู้จักค่าโมเมนตัมเสียก่อน

ค่าโมเมนตัมถูกใส่เข้าไปในสมการปรับปรุงค่าน้ำหนักและไบอัสเพื่อให้ลดความไวในการเปลี่ยนแปลงค่าทั้งสองโดยใช้หลักการของตัวกรองความถี่ต่ำผ่าน

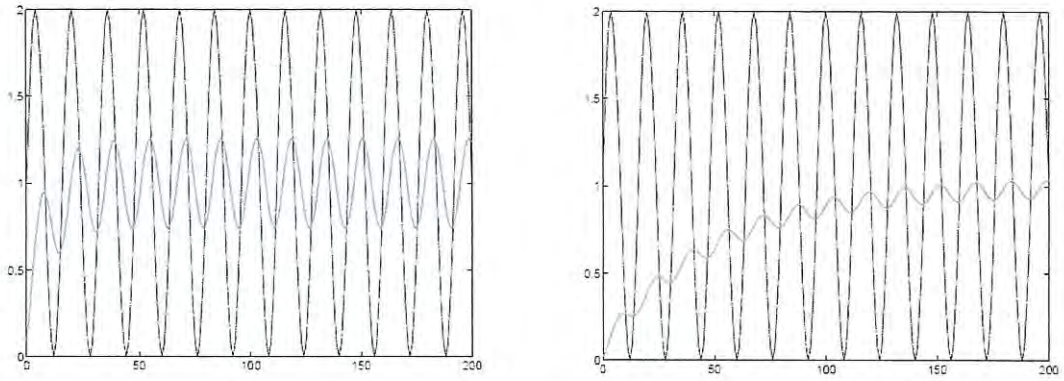
$$y(k) = y(k-1)\gamma + (1-\gamma)x(k) \tag{3.52}$$

โดยที่ $y(k)$ คือเอ้าพุต

γ คือค่าโมเมนตัมโดยที่ $0 \le \gamma < 1$

$x(k)$ คืออินพุต

ตัวอย่างการใช้ค่าโมเมนตัมลดความไวในการเปลี่ยนแปลงของสัญญาณแสดงในรูปที่ 3.9 โดยกำหนดให้อินพุตเป็นสัญญาณไซน์ $1+\sin(2\pi t/16)$ แล้วใส่ค่าโมเมนตัมขนาดต่างๆเพื่อดูการตอบสนองของสัญญาณเอ้าพุต กราฟสีดำแทนด้วยสัญญาณอินพุต กราฟสีน้ำเงินแทนด้วยสัญญาเอ้าพุต



(ก.) (ข.)

รูปที่ 3.9 ผลกระทบของค่าโมเมนตัมที่ทำให้สัญญาณเรียบขึ้น (ก.) γ =0.9 (ข.) γ =0.98 [5]

จากนั้นเราสามารถนำค่าโมเมนตัมมาใช้ โดยมองเทอมการปรับปรุงค่าน้ำหนักและไบอัสในสมการที่ (3.48),(3.49) เป็นอินพุตของของโมเมนตัม

$$\Delta w_{i,j}^{m}(k) = -\alpha \times s_i^m \times a_i^{m-1} \tag{3.53}$$

$$\Delta b_i^m(k) = -\alpha \times s_i^m \tag{3.54}$$

ใส่ค่าโมเมนตัมไปในเทอมการปรับปรุงค่าน้ำหนักและไบอัสได้

$$\Delta w_{i,j}^{m}(k) = \Delta w_{i,j}^{m}(k-1)\gamma - (1-\gamma) \times \alpha \times s_i^m \times a_i^{m-1} \tag{3.55}$$

$$\Delta b_i^m(k) = \Delta b_i^m(k-1)\gamma - (1-\gamma) \times \alpha \times s_i^m \tag{3.56}$$

เมื่อนำสมการที่ (3.55),(3.56) แทนในสมการที่ (3.48),(3.49) จะได้สมการปรับปรุงค่าน้ำหนักและไบอัสด้วยการเรียนรู้แบบแพร่ย้อนกลับที่มีค่าโมเมนตัมร่วมด้วย ซึ่งมีข้อดีตรงที่หากเรากำหนดค่าอัตราการเรียนรู้ใหญ่เกินไปกับการปรับปรุงแบบปกติโครงข่ายจะไม่สามารถลู่เข้าคำตอบได้ แต่ถ้าหากใช้ค่าอัตราการเรียนรู้เดียวกันนี้กับการปรับปรุงค่าน้ำหนักแบบมีเทอมโมเมนตัม ค่าโมเมนตัมจะช่วยลดความไวของการเปลี่ยนแปลงค่าน้ำหนัก โครงข่ายจะกลับมาลู่เข้าคำตอบได้

หลักการปรับอัตราการเรียนรู้มีมากมายหลายวิธีแต่ ณ ที่นี้จะแสดงการปรับอัตราการเรียนรู้อย่างง่ายแบบตรงไปตรงมาที่ใช้ร่วมกับค่าโมเมนตัมวิธีหนึ่ง ซึ่งแบ่งได้เป็น 3 กรณีดังนี้

1.ถ้าค่าปัจจุบันยังอยู่ห่างไกลจากตำแหน่งเป้าหมายมากและการปรับปรุงเคลื่อนที่ไปถูกทาง (ค่าคลาดเคลื่อนลดลง) ให้เพิ่มขนาดอัตราการเรียนรู้ด้วยการคูณค่าคงที่ η (Increment term) โดยที่ $\eta > 1$ และปรับค่าโมเมนตัมเป็นค่าเริ่มต้น

2.เมื่อค่าคลาดเคลื่อนจากเป้าหมายน้อยกว่าที่กำหนด (ปกติคือประมาณ 1-5%) ให้คงที่ค่าอัตราการเรียนรู้นั้นไว้ และปรับค่าโมเมนตัมเป็นค่าเริ่มต้น

3.เมื่อค่าปัจจุบันเคลื่อนที่ไปผิดทิศทาง(ค่าคลาดเคลื่อนเพิ่มขึ้น) ให้ทำการลดขนาดอัตราการเรียนรู้ลงด้วยการคูณค่าคงที่ ρ (Decrement term) โดยที่ $\rho < 1$ และปรับค่าโมเมนตัมเป็นศูนย์

ตัวอย่างขั้นตอนการปรับเปลี่ยนค่าอัตราการเรียนรู้ที่ใช้ในการเรียนแบบแพร่ย้อนกลับด้วยอัตราการเรียนรู้แบบปรับค่าได้แสดงไว้ในรูปที่ 3.10

```
%Find error
1 e(k)=target-out(k)
%Find percent of error
2 ep=(e(k)/target)*100
3 if k>=2
    %Increased learning rate
4    if(abs(e(k))<abs(e(k-1)))&&(abs(ep)>5)
5        alpha=alpha*1.02
6        r=0.2
    %Decreased learning rate
7    else if(abs(e(k))>abs(e(k-1)))&&(abs(ep)>5)
8        alpha=alpha*0.7;
9        r=0;
10   else
    %Percent error is less than 5%, then, no ed
11       r=0.2
12   end
13 end
```

รูปที่ 3.10 ขั้นตอนการปรับเปลี่ยนค่าอัตราการเรียนรู้อย่างง่าย

บทที่ 4
การแก้ปัญหาจลนศาสตร์ผกผันด้วยโครงข่ายประสาทเทียมจากสมการจลนศาสตร์ผันตรง

รายงานฉบับนี้ต้องการแก้ไขปัญหาจลนศาสตร์ผกผันด้วยการสร้างโครงข่ายประสาทเทียมให้มีโครงสร้างเหมือนกับสมการจลนศาสตร์ผันตรง แล้วทำการสอนโครงข่ายจนค่าคลาดเคลื่อนระหว่างจุดปลายแขนกลกับตำแหน่งเป้าหมายน้อยกว่าค่าที่ตั้งไว้ จากนั้นมุมคำตอบสามารถหาได้จากค่าน้ำหนักสุดท้าย ดังนั้นกระบวนการทำงานจึงสามารถแบ่งออกได้เป็น 2 ส่วนหลักคือ การสร้างโครงข่ายและการสอนโครงข่าย รายงานฉบับนี้ได้ทำการทดลองกับแขนกลที่มีข้อต่อแบบพับ (Revolute joint) ในระบบคาร์ทีเซียน 3 มิติ เพราะฉะนั้นเราสามารถนำสมการจลนศาสตร์ผันตรงที่ได้กล่าวไว้ในบทที่ 3 มาใช้ในการคำนวณได้

4.1 โครงข่ายประสาทเทียมจากสมการจลนศาสตร์ผันตรงแบบที่ 1

4.1.1 การสร้างโครงข่ายประสาทเทียมแบบที่ 1

ในช่วงแรกของงานวิจัยเราได้สร้างโครงข่ายประสาทเทียม 3 โครงข่ายจากสมการจลนศาสตร์ผันตรงของแขนกลที่มี 3 องศาอิสระใน (4.1),(4.2),(4.3) โดยที่โครงข่ายมีอินพุตเป็นมุมข้อต่อและเอ้าพุตเป็นตำแหน่ง x,y,z ของจุดปลายแขนกลตามลำดับ [3]

$$x_{end} = \cos(q_1)[\ell_2 \cos(q_2) + \ell_3 \cos(q_2 + q_3)] \tag{4.1}$$

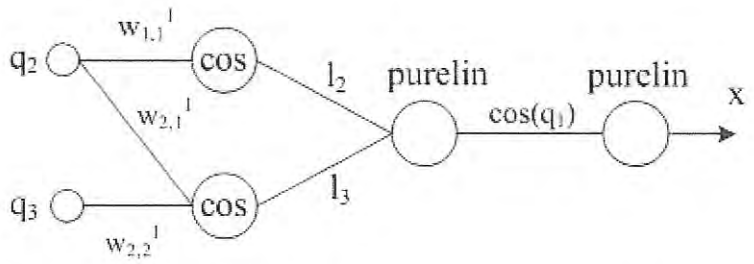
$$y_{end} = \sin(q_1)[\ell_2 \cos(q_2) + \ell_3 \cos(q_2 + q_3)] \tag{4.2}$$

$$z_{end} = \ell_1 + \ell_2 \sin(q_2) + \ell_3 \sin(q_2 + q_3) \tag{4.3}$$

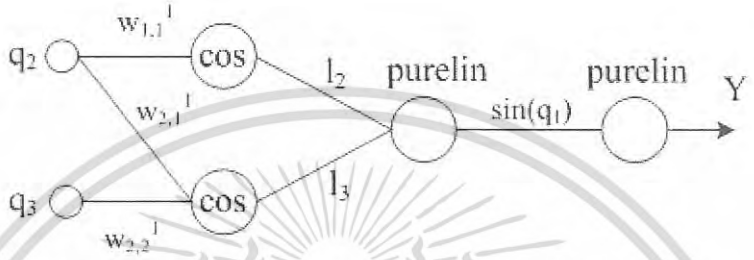
จากสมการเรากำหนดให้ฟังก์ชันโคไซน์และไซน์เป็นทรานสเฟอร์ฟังก์ชันในชั้นซ่อนที่ 1 ของโครงข่าย ยกเว้นเทอม $cos(q_1)$, $sin(q_1)$ ในสมการ (4.1) (4.2) จะถูกกำหนดเป็นค่าน้ำหนักในชั้นเอ้าพุต ซึ่งค่ามุม q_1 หาได้จากสมการที่ (4.4) โดยใช้หลักตรีโกณมิติที่ได้กล่าวไว้ในหัวข้อที่ 3.1.2.1

$$q_1 = \arctan\left(\frac{y_d - y_{ref}}{x_d - x_{ref}}\right) \tag{4.4}$$

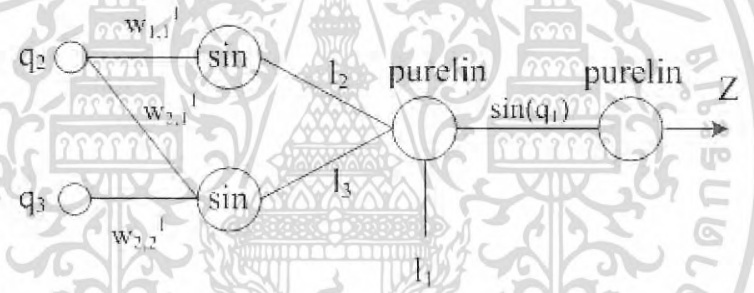
ส่วนค่าความยาวแขนกลถูกกำหนดเป็นค่าน้ำหนักในชั้นซ่อนที่ 1 จึงสามารถสร้างโครงข่ายได้ดังรูปที่ 4.1, 4.2, 4.3



รูปที่ 4.1 โครงข่ายประสาทเทียมตำแหน่ง x จากสมการจลนศาสตร์ผันตรง



รูปที่ 4.2 โครงข่ายประสาทเทียมตำแหน่ง y จากสมการจลนศาสตร์ผันตรง



รูปที่ 4.3 โครงข่ายประสาทเทียมตำแหน่ง z จากสมการจลนศาสตร์ผันตรง

เอ้าพุตของโครงข่ายสามารถเขียนได้ดังนี้

$$x = w_{1,1}^3[w_{1,1}^2 \cos(w_{1,1}^1 q_2) + w_{1,2}^2 \cos(w_{2,1}^1 q_2 + w_{2,2}^1 q_3)] \quad (4.5)$$

$$y = w_{1,1}^3[w_{1,1}^2 \cos(w_{1,1}^1 q_2) + w_{1,2}^2 \cos(w_{2,1}^1 q_2 + w_{2,2}^1 q_3)] \quad (4.6)$$

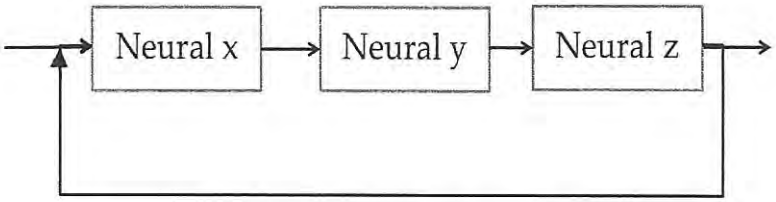
$$z = w_{1,1}^3[w_{1,1}^2 \cos(w_{1,1}^1 q_2) + w_{1,2}^2 \cos(w_{2,1}^1 q_2 + w_{2,2}^1 q_3) + b_1^2] \quad (4.7)$$

จากสมการ (4.5) (4.6) (4.7) พบว่าเอ้าพุตของโครงข่ายมีลักษณะคล้ายกับสมการจลนศาสตร์ผันตรง โดยขั้นต่อไปเราจะทำการสอนโครงข่ายจนให้ค่าตำแหน่งทั้ง 3 ที่ถูกต้อง

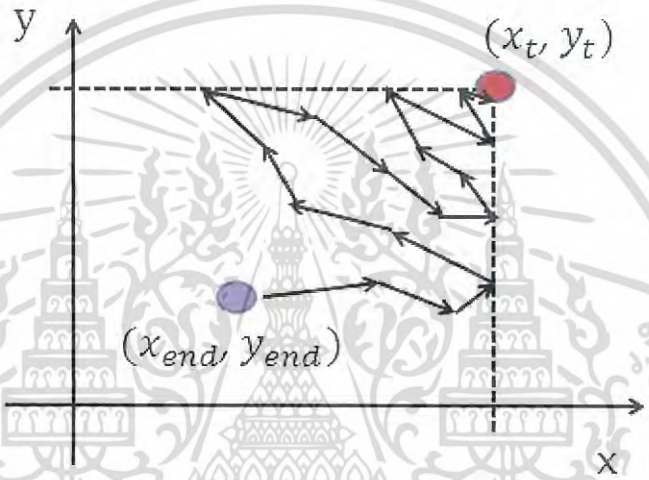
4.1.2 การเรียนรู้และการหาคำตอบของโครงข่ายประสาทเทียมแบบที่ 1

จากโครงข่ายที่ได้เราสามารถปรับค่าตำแหน่ง x, y, z ได้ด้วยการปรับค่าน้ำหนักในโครงข่าย โดยการสอนโครงข่ายทีละโครงข่ายด้วยการเรียนรู้แบบแพร่ย้อนกลับด้วยอัตราเรียนรู้ไม่คงที่ (Variable learning rate backpropagation algorithm) จนกระทั่งค่าคลาดเคลื่อนของตำแหน่ง x, y, z ของแต่ละโครงข่ายน้อยกว่าค่าที่ตั้งไว้ โดยรูปที่ 4.4 แสดง block diagram ของระบบและ

รูปที่ 4.5 แสดงตัวอย่างการเคลื่อนที่สู่ตำแหน่งเป้าหมายของโครงข่ายแบบที่ 1 โดยระบบจะทำการปรับปรุงตำแหน่งของจุดปลายแขนกลทีละมิติ จนกระทั่งเข้าใกล้จุดเป้าหมาย



รูปที่ 4.4 การปรับปรุงโครงข่ายตำแหน่งทีละโครงข่าย



รูปที่ 4.5 ตัวอย่างเส้นทางการเคลื่อนที่สู่ตำแหน่งเป้าหมายเมื่อทำการปรับปรุงโครงข่ายตำแหน่งทีละโครงข่ายใน 2 มิติ

แต่ในวิธีที่ต้องการนำเสนอนี้เป็นการปรับค่าน้ำหนักในชั้นแรกเท่านั้นหรือคือปรับค่าแค่ตัวแปรมุมเพียงอย่างเดียว เพราะการควบคุมแขนกลในความเป็นจริงเป็นการปรับแค่เพียงค่ามุมของแขนกลโดยที่ขนาดแขนกลถูกกำหนดให้คงที่ เพราะฉะนั้นค่าเริ่มต้นของโครงข่ายจึงถูกกำหนดไว้ดังนี้

ชั้นอินพุต : มุม q_1 หาจากสมการ (4.1), มุม q_2, q_3 ค่าเริ่มต้นในช่วง [0, 2π]

ชั้นซ่อนที่ 1 : ค่าน้ำหนัก $w_{1,1}^1, w_{2,1}^1, w_{2,2}^1$ เริ่มต้นเท่ากับ 1

ชั้นซ่อนที่ 2 : ค่าน้ำหนัก $w_{1,1}^2, w_{1,2}^2$ กำหนดเป็นค่าคงที่เท่ากับค่าความยาวแขนกล l_2, l_3

ชั้นเอ้าพุต : ค่าน้ำหนัก $w_{1,1}^3$ คือค่าฟังก์ชันโคไซน์และไซน์ของ q_1

เมื่อแทนค่าเริ่มต้นในสมการเอ้าพุตของโครงข่าย (4.5) (4.6) (4.7) จะได้สมการดังนี้

$$x = \cos(q_1)[\ell_2 \cos(w_{1,1}^1 q_2) + \ell_3 \cos(w_{2,1}^1 q_2 + w_{2,2}^1 q_3)] \quad (4.8)$$

$$y = \sin(q_1)[\ell_2 \cos(w_{1,1}^1 q_2) + \ell_3 \cos(w_{2,1}^1 q_2 + w_{2,2}^1 q_3)] \quad (4.9)$$

$$z = \ell_1 + \ell_2 \sin(w_{1,1}^1 q_2) + \ell_3 \sin(w_{2,1}^1 q_2 + w_{2,2}^1 q_3) \quad (4.10)$$

เมื่อลองเปรียบเทียบสมการเอ้าพุตเหล่านี้กับสมการจลนศาสตร์ผันตรง จะเห็นได้ว่าพจน์ผลคูณของค่าน้ำหนักชั้นแรกกับค่ามุมอินพุต เปรียบได้กับตัวแปรมุมในสมการจลนศาสตร์ผันตรง เพราะฉะนั้นเมื่อโครงข่ายปรับค่าน้ำหนักจนเอ้าพุตตรงกับตำแหน่งเป้าหมายแล้ว ค่ามุมคำตอบจึงสามารถหาได้จากผลคูณของค่าน้ำหนักสุดท้าย (w_{final}) กับค่ามุมอินพุต (q^{in}) ดังแสดงต่อไปนี้

ค่ามุม q_2

$$q_2 = w^1_{1\,final} q^{in}_2 \tag{4.11}$$

ค่ามุม q_3

$$q_2 + q_3 = w^1_{2\,final} q^{in}_2 + w^1_{3\,final} q^{in}_3$$
$$q_3 = (w^1_{2\,final} q^{in}_2 + w^1_{3\,final} q^{in}_3) - q_2 \tag{4.12}$$

วิธีนี้สามารถแก้ปัญหาจลนศาสตร์ผกผันสำหรับแขนกล 3 องศาอิสระใน 3 มิติได้โดยมีข้อดีคือวิธีนี้ทำการหาค่ามุม q_1 ด้วยหลักตรีโกณมิติก่อนนำไปใส่โครงข่าย ทำให้ลดการคำนวณของโครงข่ายลงได้ แต่ก็มีข้อเสียคือในขณะที่ทำการปรับโครงข่ายตำแหน่งใดตำแหน่งหนึ่งนั้น ค่าคลาดเคลื่อนของตำแหน่งอื่นๆอาจจะเพิ่มขึ้นสวนทางกันเพราะวิธีนี้ทำการปรับโครงข่ายทีละตัวโดยไม่ได้นำค่าคลาดเคลื่อนของโครงข่ายตัวอื่นๆมาคิดร่วมด้วย ซึ่งหมายความว่าอาจมีตำแหน่งเป้าหมายบางตำแหน่งที่วิธีนี้จะลู่เข้าช้าหรือไม่สามารถลู่เข้าได้ อีกทั้งวิธีนี้ยังมีปัญหาเรื่องค่ามุมเริ่มต้นที่ไม่สามารถกำหนดให้มุมเริ่มต้น q_{in} เป็นศูนย์ได้ ดูจากสมการที่ (4.11) และ (4.12) เมื่อกำหนดให้ q_{in} = 0 จะทำให้ $q_{2out}=0$, $q_{3out}=-q_{2out}$ ซึ่งทำให้ค่า q_{out} คงที่และไม่ขึ้นกับค่าน้ำหนักแล้ว

4.2 โครงข่ายประสาทเทียมจากสมการจลนศาสตร์ผันตรงแบบที่ 2

ในงานวิจัยชิ้นที่ 2 [4] ทำการสร้างโครงข่ายประสาทเทียมที่มีอินพุตเป็นมุมข้อต่อและเอ้าพุตเป็นระยะทางแบบยุคลิดจากจุดปลายแขนกล $(x_{end}, y_{end}, z_{end})$ ถึงตำแหน่งเป้าหมาย (x_d, y_d, z_d) ซึ่งเมื่อกำหนดให้เอ้าพุตเป็นค่าระยะทางคลาดเคลื่อนแบบยุคลิดนั้นจะทำให้เราต้องรวมเอ้าพุตของโครงข่ายทั้ง 3 จากงานวิจัยชิ้นที่ 1 [3] เข้าด้วยกันเป็นเอ้าพุตเดียวตามสมการระยะทางแบบยุคลิด

$$\text{Euclidean Distance} = \sqrt{(x_t - x_{end})^2 + (y_t - y_{end})^2 + (z_t - z_{end})^2} \tag{4.13}$$

วิธีนี้ทำให้ค่าคลาดเคลื่อนของตำแหน่งจุดปลายแขนทั้ง 3 ถูกนำมาพิจารณาในการปรับปรุงค่าน้ำหนักของโครงข่ายพร้อมๆกัน หรือค่าคลาดเคลื่อนทั้ง 3 จะลดลงไปพร้อมกันในทุกๆรอบของการปรับค่าน้ำหนักนั้นเอง โดยในงานวิจัยที่ 2 นี้ได้ทำการทดลองกับแขนกล 4 องศาอิสระใน 3 มิติ ซึ่งขั้นตอนหลักๆสามารถแบ่งออกเป็น การปรับเปลี่ยนโครงข่ายประสาทเทียมตำแหน่งแล้วรวมเข้าด้วยกัน, การสอนโครงข่ายและการหามุมคำตอบ

4.2.1 การสร้างโครงข่ายประสาทเทียมแบบที่ 2

งานวิจัยที่ 2 มีจุดประสงค์เพื่อสร้างโครงข่ายที่มีมุมข้อต่อเป็นอินพุตทั้งหมด (งานวิจัยที่ 1 ใช้มุม q_1 เป็นค่าน้ำหนัก) ซึ่งเมื่อสามารถกำหนดมุม q_1 เป็นอินพุตได้ ระบบจะสามารถปรับระนาบของแขนกลไปยังระนาบของตำแหน่งเป้าหมายได้เองโดยไม่ต้องให้ผู้ใช้คำนวณด้วยหลักตรีโกณมิติมาป้อนเหมือนวิธี [3,11,12] ดังนั้นจึงจำเป็นต้องปรับรูปแบบสมการจลนศาสตร์ผันตรงเสียก่อน จากที่กล่าวไว้ข้างต้นว่าเราต้องการสร้างโครงข่ายจากสมการจลนศาสตร์ผันตรงของแขนกล 4 องศาอิสระใน 3 มิติ (3.1),(3.2),(3.3) แต่จากการสังเกตพบว่าสมการ (3.1),(3.2) มีเทอมผลคูณของฟังก์ชันไซน์และโคไซน์ (สมการ (3.3) อยู่ในรูปผลรวมแล้ว) ซึ่งโดยปกติแล้วโครงสร้างของโครงข่ายประสาทเทียมมักอยู่ในรูปของผลรวมของผลคูณเอ้าพุตกับค่าน้ำหนัก เราจึงต้องทำการเปลี่ยนรูปสมการ (3.1),(3.2) ให้อยู่ในรูปผลรวมของฟังก์ชันไซน์และโคไซน์ด้วยเอกลักษณ์ผลคูณตรีโกณมิติในสมการ (4.14), (4.15) เสียก่อน

$$\cos(A)\cos(B) = 0.5(\cos(A+B) + \cos(A-B)) \tag{4.14}$$

$$\sin(A)\cos(B) = 0.5(\sin(A+B) + \sin(A-B)) \tag{4.15}$$

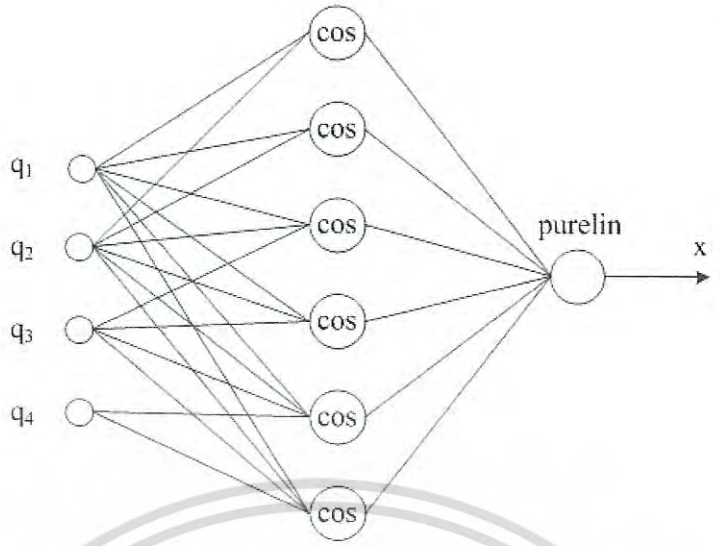
ทำการแทนสมการ (4.14) (4.15) ไปใน (3.1) (3.2) จะได้สมการจลนศาสตร์ผันตรงในรูปของผลรวมของฟังก์ชันไซน์และโคไซน์ดังแสดงในสมการ (4.16) (4.17)

$$x_{end} = \begin{bmatrix} 0.5\ell_2\cos(q_1+q_2) + 0.5\ell_2\cos(q_1-q_2) \\ +0.5\ell_3\cos(q_1+q_2+q_3) + 0.5\ell_3\cos(q_1-q_2-q_3) \\ +0.5\ell_4\cos(q_1+q_2+q_3+q_4) + 0.5\ell_4\cos(q_1-q_2-q_3-q_4) \end{bmatrix} \tag{4.16}$$

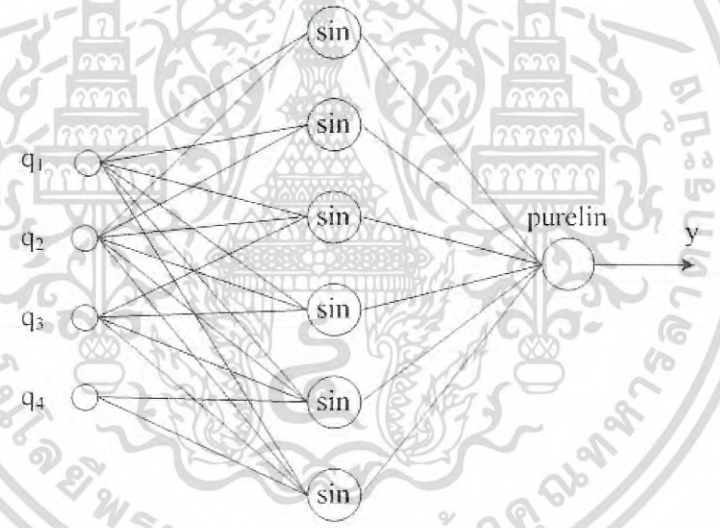
$$y_{end} = \begin{bmatrix} 0.5\ell_2\sin(q_1+q_2) + 0.5\ell_2\sin(q_1-q_2) \\ +0.5\ell_3\sin(q_1+q_2+q_3) + 0.5\ell_3\sin(q_1-q_2-q_3) \\ +0.5\ell_4\sin(q_1+q_2+q_3+q_4) + 0.5\ell_4\sin(q_1-q_2-q_3-q_4) \end{bmatrix} \tag{4.17}$$

จะเห็นได้ว่าตอนนี้สมการ (4.16) และ (4.17) อยู่ในรูปผลรวมไซน์และโคไซน์ซึ่งเหมาะสำหรับการนำไปใช้สร้างโครงข่ายประสาทเทียมโดยให้มุมข้อต่อทั้งหมดเป็นอินพุต

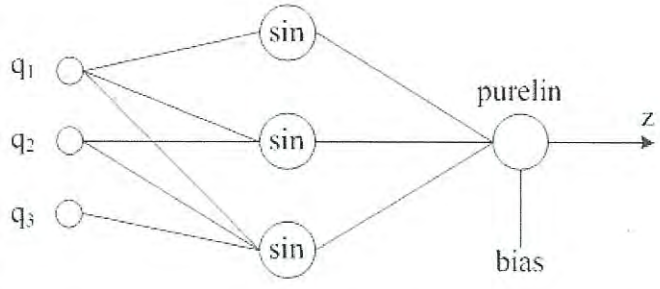
ขั้นต่อไปเราจะทำการสร้างโครงข่ายจากสมการจลนศาสตร์ผันตรงที่ปรับรูปแล้วสมการที่ (4.16), (4.17), (3.3) และกำหนดให้ฟังก์ชันไซน์และโคไซน์ในสมการเป็นทรานสเฟอร์ฟังก์ชันของโครงข่าย, ค่าน้ำหนักในชั้นซ่อนที่ 1 คือตัวแปรสำหรับปรับค่ามุม, ค่าน้ำหนักในชั้นซ่อนที่ 2 คือค่าคงที่ 1 กับ -1 หรือสัมประสิทธิ์ของตัวแปรมุมในสมการ (4.16), (4.17), (3.3) และค่าน้ำหนักในชั้นซ่อนที่ 3 คือสัมประสิทธิ์ของฟังก์ชันไซน์และโคไซน์ในสมการ (4.16), (4.17), (3.3) โครงข่ายประสาทเทียมที่ได้แสดงไว้ในรูปรูปที่ 4.6, 4.7 และ 4.8



รูปที่ 4.6 โครงข่ายประสาทเทียมแทนตำแหน่ง x ของจุดปลายแขนกลแบบที่ 2



รูปที่ 4.7 โครงข่ายประสาทเทียมแทนตำแหน่ง y ของจุดปลายแขนกลแบบที่ 2



รูปที่ 4.8 โครงข่ายประสาทเทียมแทนตำแหน่ง z ของจุดปลายแขนกลแบบที่ 2

เอ้าพุตของทั้ง 3 โครงข่ายสามารถเขียนได้ดังนี้

$$x = w_{1,1}^2 \cos(w_{1,1}^1 q_1 + w_{1,2}^1 q_2) + w_{1,2}^2 \cos(w_{2,1}^1 q_1 + w_{2,2}^1 q_2) + w_{1,3}^2 \cos(w_{3,1}^1 q_1 + w_{3,2}^1 q_2 + w_{3,3}^1 q_3)$$
$$+ w_{1,4}^2 \cos(w_{4,1}^1 q_1 + w_{4,2}^1 q_2 + w_{4,3}^1 q_3) + w_{1,5}^2 \cos(w_{5,1}^1 q_1 + w_{5,2}^1 q_2 + w_{5,3}^1 q_3 + w_{5,4}^1 q_4) \quad (4.18)$$
$$+ w_{1,6}^2 \cos(w_{6,1}^1 q_1 + w_{6,2}^1 q_2 + w_{6,3}^1 q_3 + w_{6,4}^1 q_4)$$

$$y = w_{1,1}^2 \sin(w_{1,1}^1 q_1 + w_{1,2}^1 q_2) + w_{1,2}^2 \sin(w_{2,1}^1 q_1 + w_{2,2}^1 q_2) + w_{1,3}^2 \sin(w_{3,1}^1 q_1 + w_{3,2}^1 q_2 + w_{3,3}^1 q_3)$$
$$+ w_{1,4}^2 \sin(w_{4,1}^1 q_1 + w_{4,2}^1 q_2 + w_{4,3}^1 q_3) + w_{1,5}^2 \sin(w_{5,1}^1 q_1 + w_{5,2}^1 q_2 + w_{5,3}^1 q_3 + w_{5,4}^1 q_4) \quad (4.19)$$
$$+ w_{1,6}^2 \sin(w_{6,1}^1 q_1 + w_{6,2}^1 q_2 + w_{6,3}^1 q_3 + w_{6,4}^1 q_4)$$

$$z = b_1^2 + w_{1,1}^2 \sin(w_{1,1}^1 q_2) + w_{1,2}^2 \sin(w_{2,1}^1 q_2 + w_{2,2}^1 q_3) + w_{1,3}^2 \sin(w_{3,1}^1 q_2 + w_{3,2}^1 q_3 + w_{3,3}^1 q_4) \quad (4.20)$$

เนื่องจากเราต้องการสร้างโครงข่ายให้เอ้าพุตเป็นค่าคลาดเคลื่อนจากตำแหน่งเป้าหมายตามสมการยุคลิด (4.13) เราจึงต้องเปลี่ยนเอ้าพุตของแต่ละโครงข่ายให้เป็นค่าคลาดเคลื่อนจากตำแหน่งเป้าหมายกำลังสอง โดยการเพิ่มค่าไบอัสเป็นตำแหน่งเป้าหมายแบบติดลบ ($-x_d, -y_d, -z_d$) และเปลี่ยนทรานสเฟอร์ฟังก์ชันชั้นเอ้าพุตเป็นฟังก์ชันกำลังสอง จากนั้นรวมเอ้าพุตของโครงข่ายทั้ง 3 เข้าด้วยกัน แล้วนำไปเข้าทรานสเฟอร์ฟังก์ชันรากที่สองก็จะได้โครงข่ายที่มีเอ้าพุตเป็นระยะทางคลาดเคลื่อนจากตำแหน่งเป้าหมายแบบยุคลิดแล้ว โครงข่ายประสาทเทียมที่ได้แสดงไว้ในรูปที่ 4.9

ค่าเริ่มต้นต่างๆของโครงข่ายในชั้นซ่อนที่ 1, 2 และ 3 จะกำหนดคล้ายกับโครงข่ายในงานวิจัยที่ 1 ซึ่งเป็นไปดังนี้

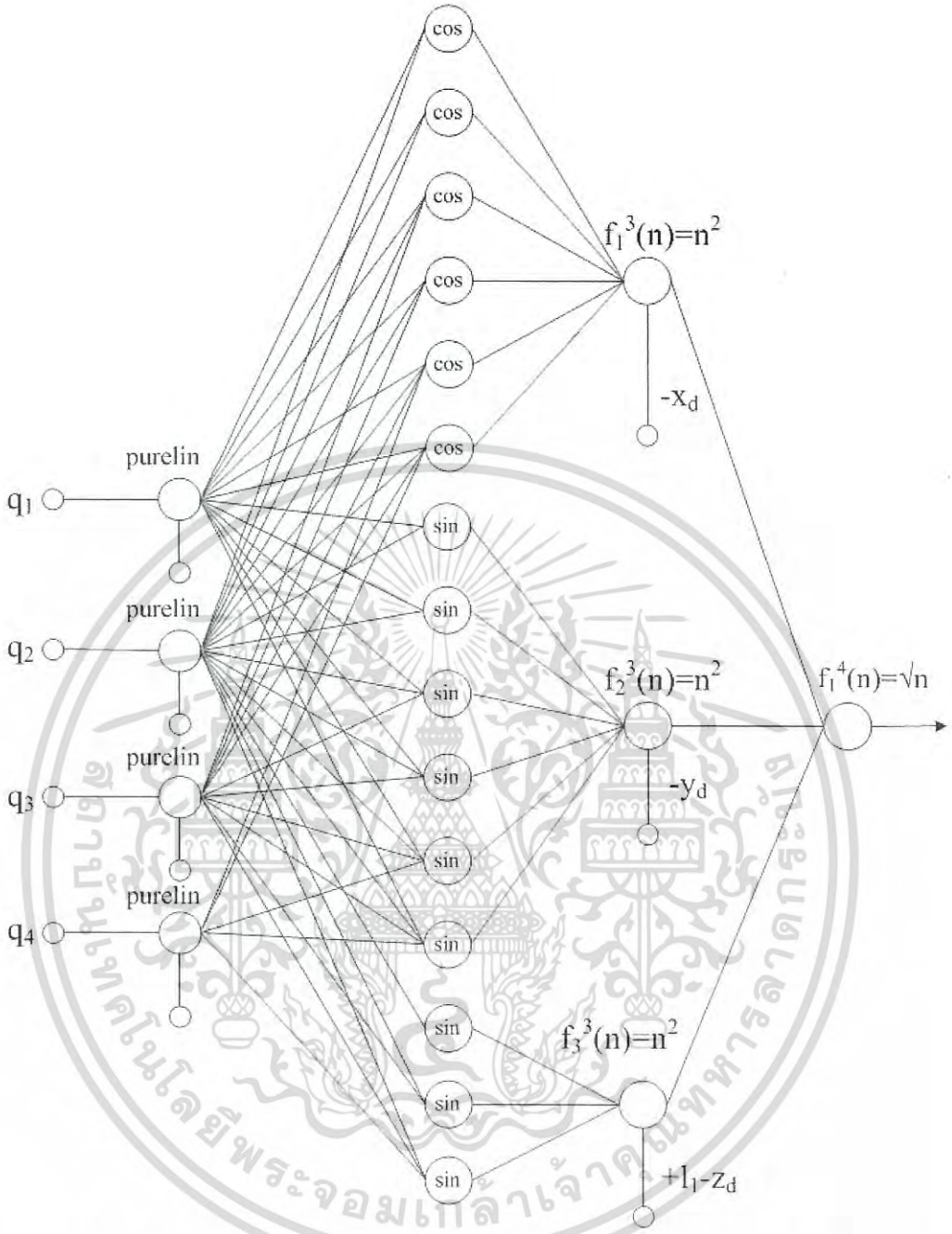
ชั้นอินพุต : มุม $q_1\ q_2 q_3\ q_4$ เริ่มต้นในช่วง $[0,2\pi]$

ชั้นซ่อนที่ 1 : ค่าน้ำหนัก $w_{1,1}^1$, $w_{2,1}^1$, $w_{3,1}^1$, $w_{4,1}^1$ เริ่มต้นเท่ากับ 1 , b_1^1 ถึง b_4^1 เริ่มต้นเท่ากับ 0 และทำการปรับปรุงค่าจนโครงข่ายลู่เข้าคำตอบ

ชั้นซ่อนที่ 2 : ค่าน้ำหนักเท่ากับ 1 และ -1 หรือสัมประสิทธิ์ของตัวแปร q ในสมการ (4.16), (4.17), (3.3) ตามลำดับ

ชั้นซ่อนที่ 3 : ค่าน้ำหนักเท่ากับค่าความยาวแขนกลหารสองหรือคือสัมประสิทธิ์ของฟังก์ชันไซน์และโคไซน์ในสมการ (4.16),(4.17),(3.3) ตามลำดับและไบอัส b_1^3 ถึง b_3^3 เป็นค่าติดลบของตำแหน่งเป้าหมาย $-x_d$, $-y_d$, l_1-z_d ตามลำดับ

ชั้นเอ้าพุต : $w_{1,1}^4$ ถึง $w_{1,3}^4$ เท่ากับ 1 หรือคือสัมประสิทธิ์ของค่าคลาดเคลื่อนแต่ละตำแหน่งในสมการที่ 4.13



รูปที่ 4.9 โครงข่ายประสาทเทียมที่มีอินพุตเป็นมุมข้อต่อและเอ้าพุตเป็นค่าคลาดเคลื่อนระหว่างจุดปลายแขนกลกับตำแหน่งเป้าหมาย

เมื่อแทนค่าเริ่มต้นไปในโครงข่ายแล้วพิจารณาเอ้าพุตของชั้นซ่อนที่ 3 พบว่าคล้ายกับค่าคลาดเคลื่อนกำลังสองแล้วเนื่องจากค่าไบอัสติดลบตำแหน่งเป้าหมายที่เพิ่มเข้ามา ดังแสดงในสมการที่ (4.24),(4.25),(4.26) ส่วนของค่าเริ่มต้นของค่าน้ำหนักและไบอัสในชั้นแรกจะถูกอธิบายเพิ่มเติมในหัวข้อที่ 4.2.2

เอ้าพุตของชั้นซ่อนที่ 3 ของโครงข่ายประสาทเทียมแบบรวมค่าคลาดเคลื่อน

$$a_1^3 = \begin{pmatrix} w_{1,1}^3 \cos(w_{1,1}^2 a_1^1 + w_{1,2}^1 a_2^1) + w_{1,2}^3 \cos(w_{2,1}^2 a_1^1 + w_{2,2}^1 a_2^1) \\ + w_{1,3}^3 \cos(w_{3,1}^2 a_1^1 + w_{3,2}^1 a_2^1 + w_{3,3}^1 a_3^1) + w_{1,4}^3 \cos(w_{4,1}^2 a_1^1 + w_{4,2}^1 a_2^1 + w_{4,3}^1 a_3^1) \\ + w_{1,5}^3 \cos(w_{5,1}^2 a_1^1 + w_{5,2}^2 a_2^1 + w_{5,3}^2 a_3^1 + w_{5,4}^2 a_4^1) \\ + w_{1,6}^3 \cos(w_{6,1}^2 a_1^1 + w_{6,2}^2 a_2^1 + w_{6,3}^2 a_3^1 + w_{6,4}^2 a_4^1) + b_1^3 \end{pmatrix}^2 \quad (4.21)$$

$$a_2^3 = \begin{pmatrix} w_{2,1}^3 \sin(w_{7,1}^2 a_1^1 + w_{7,2}^1 a_2^1) + w_{2,2}^3 \sin(w_{8,1}^2 a_1^1 + w_{8,2}^1 a_2^1) \\ + w_{2,3}^3 \sin(w_{9,1}^2 a_1^1 + w_{9,2}^1 a_2^1 + w_{9,3}^1 a_3^1) + w_{2,4}^3 \sin(w_{10,1}^2 a_1^1 + w_{10,2}^1 a_2^1 + w_{10,3}^1 a_3^1) \\ + w_{2,5}^3 \sin(w_{11,1}^2 a_1^1 + w_{11,2}^2 a_2^1 + w_{11,3}^2 a_3^1 + w_{11,4}^2 a_4^1) \\ + w_{2,6}^3 \sin(w_{12,1}^2 a_1^1 + w_{12,2}^2 a_2^1 + w_{12,3}^2 a_3^1 + w_{12,4}^2 a_4^1) + b_2^3 \end{pmatrix}^2 \quad (4.22)$$

$$a_3^3 = \begin{pmatrix} w_{3,1}^3 \sin(w_{13,1}^2 a_2^1) + w_{3,2}^3 \sin(w_{14,1}^2 a_2^1 + w_{14,2}^2 a_3^1) \\ + w_{3,3}^3 \sin(w_{15,1}^1 a_2^1 + w_{15,2}^1 a_3^1 + w_{15,3}^1 a_4^1) + b_3^3 \end{pmatrix}^2 \quad (4.23)$$

แทนค่าน้ำหนักและไบอัสเริ่มต้นในสมการ (4.21), (4.22), (4.23) จะได้

$$a_1^3 = \begin{pmatrix} 0.5\ell_2 \cos(a_1^1 + a_2^1) + 0.5\ell_2 \cos(a_1^1 - a_2^1) \\ + 0.5\ell_3 \cos(a_1^1 + a_2^1 + a_3^1) + 0.5\ell_3 \cos(a_1^1 - a_2^1 - a_3^1) \\ + 0.5\ell_4 \cos(a_1^1 + a_2^1 + a_3^1 + a_4^1) + 0.5\ell_4 \cos(a_1^1 - a_2^1 - a_3^1 - a_4^1) + (-x_d) \end{pmatrix}^2 \quad (4.24)$$

$$a_1^3 = \left(x_{end} + (-x_d)\right)^2$$

$$a_2^3 = \begin{pmatrix} 0.5\ell_2 \sin(a_1^1 + a_2^1) + 0.5\ell_2 \sin(a_1^1 - a_2^1) \\ + 0.5\ell_3 \sin(a_1^1 + a_2^1 + a_3^1) + 0.5\ell_3 \sin(a_1^1 - a_2^1 - a_3^1) \\ + 0.5\ell_4 \sin(a_1^1 + a_2^1 + a_3^1 + a_4^1) + 0.5\ell_4 \sin(a_1^1 - a_2^1 - a_3^1 - a_4^1) + (-y_d) \end{pmatrix}^2 \quad (4.25)$$

$$a_2^3 = \left(y_{end} + (-y_d)\right)^2$$

$$a_2^3 = \left(\ell_2 \sin(a_1^1 + a_2^1) + \ell_3 \sin(a_1^1 + a_2^1 + a_3^1) + \ell_4 \sin(a_1^1 + a_2^1 + a_3^1 + a_4^1) + (\ell_1 - z_d)\right)^2 \quad (4.26)$$

$$a_2^3 = \left(z_{end} + (-z_d)\right)^2$$

สมการเหล่านี้เป็นค่าคลาดเคลื่อนกำลังสองของแต่ละตำแหน่งแล้ว ซึ่งทำให้เอ้าพุตของโครงข่ายประสาทเทียมแบบที่ 2 เป็นค่าคลาดเคลื่อนระหว่างจุดปลายแขนกลกับตำแหน่งเป้าหมายแบบยุคลิดแล้วดังสมการที่ (4.27)

$$a_1^4 = \sqrt{w_{1,1}^4 a_1^3 + w_{1,2}^4 a_2^3 + w_{1,3}^4 a_3^3}$$

$$a_1^4 = \sqrt{\left(x_{end} + (-x_d)\right)^2 + \left(y_{end} + (-y_d)\right)^2 + \left(z_{end} + (-z_d)\right)^2} \quad (4.27)$$

4.2.2 การเรียนรู้และการหาคำตอบของโครงข่ายประสาทเทียมแบบที่ 2

เราได้ทำการปรับค่าน้ำหนักแบบเดียวกับงานวิจัยที่ 1 คือสอนโครงข่ายด้วยการเรียนรู้แบบแพร่ย้อนกลับด้วยอัตราการเรียนรู้ไม่คงที่ (Variable learning rate backpropagation algorithm) แต่หากลองพิจารณาสมการเอ้าพุตของโครงข่าย (4.24), (4.25), (4.26) เทียบกับสมการจลนศาสตร์ผันตรงในรูปผลบวก (4.16), (4.17), (3.3) พบว่าเราสามารถกำหนดให้เอ้าพุตของชั้นซ่อนที่ 1 (a_1^1, a_2^1, a_3^1, a_4^1) เป็นตัวแปรมุม q_1, q_2, q_3, q_4 ได้ตามลำดับ

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} a_1^1 \\ a_2^1 \\ a_3^1 \\ a_4^1 \end{bmatrix} = \begin{bmatrix} w_{1,1\ final}^1 q_1^{in} + b_{1\ final}^1 \\ w_{1,2\ final}^1 q_2^{in} + b_{2\ final}^1 \\ w_{1,3\ final}^1 q_3^{in} + b_{3\ final}^1 \\ w_{1,4\ final}^1 q_4^{in} + b_{4\ final}^1 \end{bmatrix} \tag{4.28}$$

จากสมการพบว่าการหาค่ามุมคำตอบจากโครงข่ายทำได้ง่ายและตรงไปตรงมามากขึ้น และเนื่องจากงานวิจัยนี้ได้ใส่ค่าไบอัสในชั้นซ่อนที่ 1 ด้วย ทำให้เราสามารถปรับเปลี่ยนค่ามุมได้ด้วยตัวแปรน้ำหนักและตัวแปรไบอัส ซึ่งสามารถแก้ไขปัญหาค่ามุมเริ่มต้นเป็นศูนย์ในโครงข่ายแบบที่ 1 ได้ เพราะเมื่อค่ามุมเริ่มต้นเป็นศูนย์โครงข่ายแบบที่ 2 สามารถปรับค่ามุมได้ด้วยพจน์ไบอัส

การกำหนดค่าเริ่มต้นของค่าน้ำหนักและไบอัสนั้นมีผลกับท่าทางเริ่มต้นของแขนกล จากสมการ (4.28) พบว่าเมื่อกำหนดค่าน้ำหนักและไบอัสเริ่มต้นเป็น 1 และ 0 ตามลำดับ ท่าทางเริ่มต้นของแขนกลจะเป็นไปตามสมการ $q^{initial}=q^{in}$ หรือท่าทางเริ่มต้นของแขนกลจะเท่ากับมุมอินพุตที่เราต้องการกำหนดให้แขนกล แต่ถ้าหากเรากำหนดค่าน้ำหนักและไบอัสเริ่มต้นเป็นอย่างอื่นจะส่งผลให้ท่าทางเริ่มต้นเป็นไปตามสมการ $q^{initial}=w^{initial}q^{in}+b^{initail}$ แทน ซึ่งค่าน้ำหนักและไบอัสเริ่มต้นนี้จะทำให้ระบบลู่เข้าเร็วขึ้นเมื่อทำให้ท่าทางเริ่มต้นของแขนกลอยู่ใกล้ตำแหน่งเป้าหมายมากกว่าท่าทางอินพุตที่ป้อน แต่จะทำให้ระบบลู่เข้าช้าลงเมื่อทำให้ท่าทางเริ่มต้นของแขนกลอยู่ห่างจากตำแหน่งเป้าหมายมากกว่าท่าทางอินพุตที่ป้อน ดังนั้นรายงานฉบับนี้จึงทำการกำหนดค่าน้ำหนักและไบอัสเริ่มต้นเป็น 1 และ 0 เพื่อให้ง่ายต่อการกำหนดท่าทางเริ่มต้นของแขนกลเท่านั้น ในบทต่อไปจะทำการทดสอบประสิทธิภาพของโครงข่ายทั้งสองเทียบกับวิธี FABRIK [11]

บทที่ 5
การทดลองและผลการทดลอง

งานวิจัยนี้ได้ทำการทดสอบอัลกอริทึมกับแขนกลจำลองในโปรแกรม MATLAB บนคอมพิวเตอร์แบบพกพา Intel Core i5-2450M @2.5 GHz RAM 4 GB โดยในงานวิจัยที่ 1 [3] ทำการทดลองกับแขนกลที่มี 3 องศาอิสระใน 3 มิติ และงานวิจัยที่ 2 [4] ทำการทดลองกับแขนกลที่มี 4 องศาอิสระใน 3 มิติ การทดลองจะแบ่งออกเป็น 2 ส่วนคือ การทดลองเพื่อดูลักษณะการลู่เข้าที่ค่าเริ่มต้นต่างกัน และทดสอบประสิทธิภาพของระบบด้วยการให้แขนกลติดตามเส้นวิถีแบบต่างๆ พร้อมทั้งทำการเปรียบเทียบผลการคำนวณและข้อดี ข้อเสียของงานวิจัยทั้ง 2 กับวิธี FABRIK [11] อีกด้วย ความยาวของท่อนแขนกลและค่าตัวแปรเริ่มต้นของการเรียนรู้แบบแพร่ย้อนกลับด้วยอัตราการเรียนรู้ไม่คงที่ (Variable learning rate backpropagation algorithm) ในงานวิจัยที่ 1 และ 2 แสดงไว้ในตารางที่ 5.1 และ 5.2 ตามลำดับ

ตารางที่ 5.1 ค่าเริ่มต้นของโครงข่ายประสาทเทียมแบบที่ 1

	x	y	z
Learning rate	0.005	0.005	0.003
Increment term	1.05	1.05	1.1
decrement term	0.3	0.3	0.3
Momentum term	0.2	0.2	0.2
Error (%)	5	5	5
Tolerance (cm)	0.0055	0.0055	0.0055
$\ell_1(cm)$	5		
$\ell_2(cm)$	14		
$\ell_3(cm)$	13		

ตารางที่ 5.2 ค่าเริ่มต้นของโครงข่ายประสาทเทียมแบบที่ 2

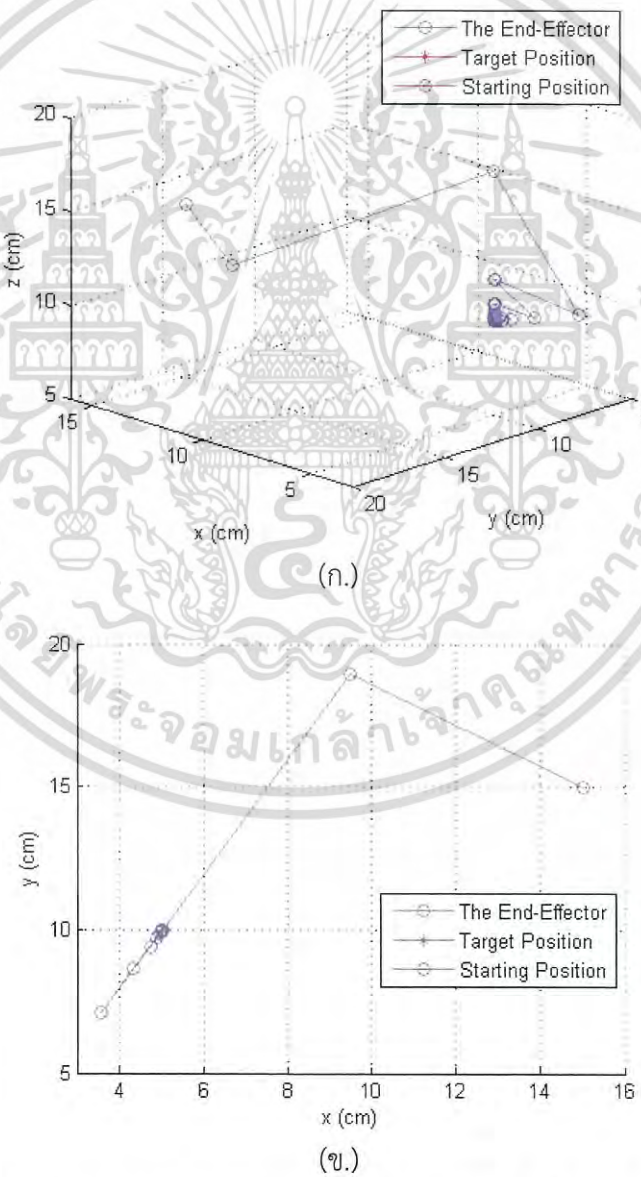
Learning rate	0.0012	$\ell_1(cm)$	5
Increment term	1.02	$\ell_2(cm)$	13
Decrement term	0.8	$\ell_3(cm)$	12
Momentum term	0.2	$\ell_4(cm)$	10
Target (cm)	0	Tolerance (cm)	0.001

5.1 เปรียบเทียบเส้นทางการเคลื่อนที่สู่ตำแหน่งเป้าหมาย

ส่วนแรกเราทำการทดลองวาดเส้นทางการเคลื่อนที่ของจุดปลายแขนกลไปยังตำแหน่งเป้าหมายใน 3 มิติว่ามีลักษณะอย่างไรพร้อมทั้งวาดกราฟเปรียบเทียบระหว่างค่าคลาดเคลื่อนจากตำแหน่งเป้าหมายแบบยุคลิดกับจำนวนรอบในการคำนวณ เมื่อสังเกตเห็นลักษณะการลู่เข้าคำตอบแล้วอาจทำให้รู้ถึงจุดเด่นจุดด้อยของระบบซึ่งสามารถนำไปแก้ไขปรับปรุงให้ดียิ่งขึ้นได้

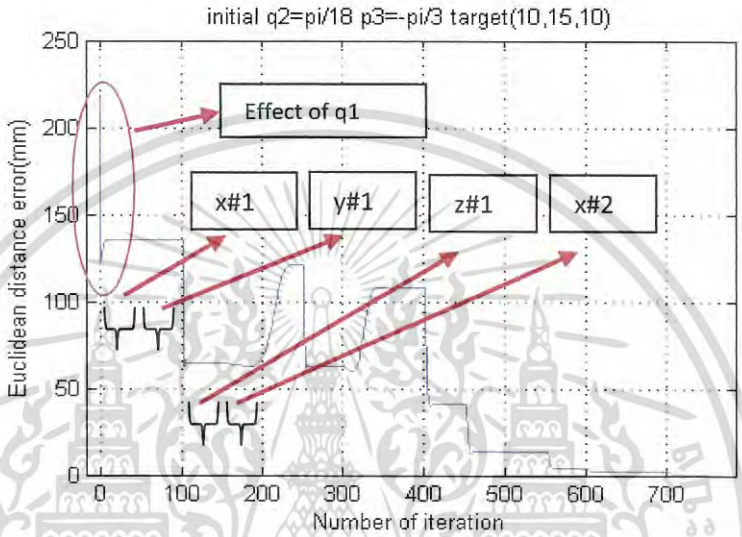
5.1.1 การลู่เข้าคำตอบของโครงข่ายประสาทเทียมแบบที่ 1

สมมติให้มุมเริ่มต้นเป็น $q_1=45$, $q_2=45$, $q_3=45$ แล้วกำหนดตำแหน่งเป้าหมายเป็น (5,10,10) เส้นทางการเคลื่อนที่แสดงในรูปที่ 5.1 โดยเส้นสีน้ำเงินคือเส้นทางการเคลื่อนที่ของจุดปลายแขนกลสู่ตำแหน่งเป้าหมาย, วงกลมสีแดงคือจุดเริ่มต้นและจุดดอกจันสีแดงคือตำแหน่งเป้าหมาย



รูปที่ 5.1 เส้นทางการเคลื่อนที่ของจุดปลายแขนกลไปยังตำแหน่งเป้าหมายใน 3 มิติ (ก.) มุมมองใน 3 มิติ (ข.) มุมมองบนระนาบ xy

จากรูปที่ 5.1 (ข.) จะเห็นได้ว่าเมื่อเราคำนวณค่า q_1 จากสมการที่ (4.1) แล้วใส่ไปเป็นค่าน้ำหนักในโครงข่ายจะทำให้ระบบทำการปรับปรุงตำแหน่ง x,y,z ในระนาบที่แขนกลหันไปทางตำแหน่งเป้าหมายเท่านั้นหรือคือปรับปรุงเฉพาะค่ามุม q_2,q_3 วิธีนี้มีข้อดีคือสามารถลดความซับซ้อนของการคำนวณได้เหมือนคำนวณหาคำตอบใน 2 มิติเท่านั้น นอกจากนี้หากทดลองวาดกราฟระยะทางคลาดเคลื่อนจากตำแหน่งเป้าหมายแบบยุคลิดเทียบกับจำนวนรอบจะพบว่าการคำนวณค่า q_1 มาใส่ยังโครงข่ายสามารถลดค่าคลาดเคลื่อนได้เป็นอย่างมากดังรูปที่ 5.2

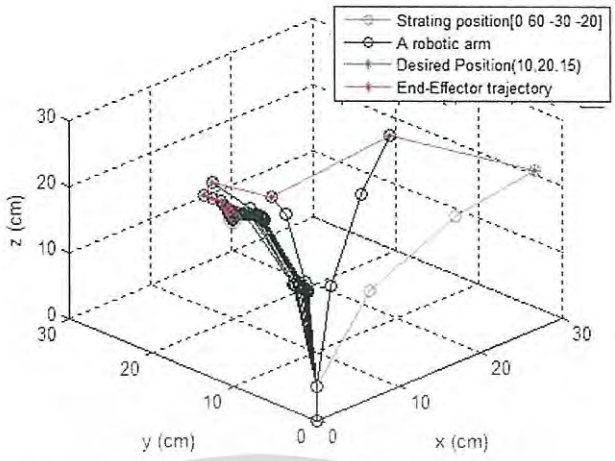


รูปที่ 5.2 ระยะทางคาดเคลื่อนจากตำแหน่งเป้าหมายแบบยุคลิดเทียบกับจำนวนรอบในการคำนวณพร้อมกับแสดงผลกระทบของค่า q_1 และการปรับปรุงโครงข่ายตำแหน่งทั้ง 3

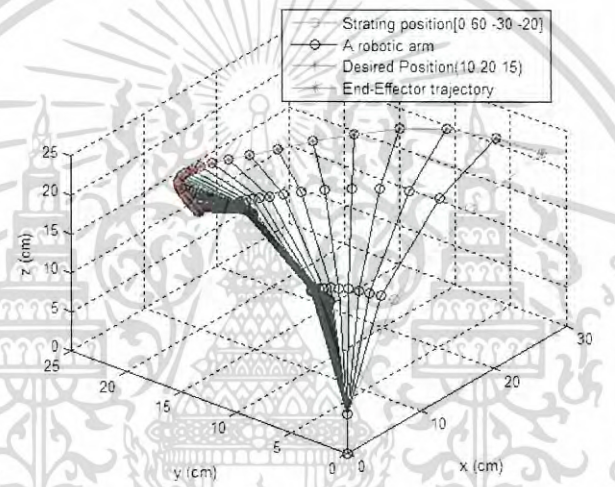
แต่วิธีนี้ก็ยังมีข้อเสียอยู่คือหากสังเกตรูปที่ 5.2 จะเห็นว่ามีบางช่วงของการคำนวณที่ระยะทางคลาดเคลื่อนเพิ่มสูงขึ้น ซึ่งเป็นไปตามที่กล่าวไว้ในหัวข้อ 4.1.2 คือเนื่องจากเราทำการปรับปรุงโครงข่ายตำแหน่งทีละโครงข่ายและไม่ได้นำค่าคลาดเคลื่อนของโครงข่ายตำแหน่งอื่นๆมาคิดร่วมด้วย

5.1.2 การลู่เข้าคำตอบของโครงข่ายประสาทเทียมแบบที่ 2

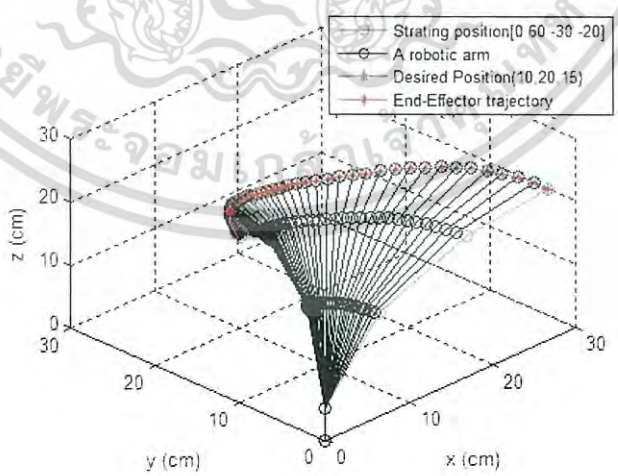
สมมติให้แขนกลมีมุมเริ่มต้นเป็น $q_1=0,\ q_2=60,\ q_3=-30,\ q_4=-20$ และตำแหน่งเป้าหมาย (10,20,15) แล้วทดลองวาดกราฟการเคลื่อนที่เข้าหาตำแหน่งเป้าหมายของแขนกลได้ดังรูปที่ 5.3 ซึ่งเส้นสีชมพูคือตำแหน่งเริ่มต้นของแขนกล, เส้นสีแดงคือเส้นทางการเคลื่อนที่ของจุดปลายแขนกล, เส้นสีดำคือลักษณะท่าทางของแขนกลเมื่อทำการปรับค่าน้ำหนักแต่ละครั้งและจุดสีน้ำเงินคือตำแหน่งเป้าหมาย ซึ่งหากเปรียบเทียบเส้นทางการเคลื่อนที่เข้าหาตำแหน่งเป้าหมายนี้กับงานวิจัยที่ 1 ในรูปที่ 5.2 จะพบว่าเส้นทางการลู่เข้านี้จะมีลักษณะวิ่งเข้าหาตำแหน่งเป้าหมายโดยตรงหรือคือค่าคลาดเคลื่อนของตำแหน่ง x,y,z จะลดลงไปพร้อมกันในทุกๆรอบของการปรับค่าน้ำหนัก ส่งผลให้ระยะคลาดเคลื่อนจากตำแหน่งเป้าหมายลดลงอย่างรวดเร็วและช่วยลดปัญหาการเพิ่มขึ้นของค่าคลาดเคลื่อนเป็นบางช่วงเหมือนงานวิจัยที่ 1 ดังแสดงในรูปที่ 5.4 (โครงข่ายแบบที่ 2 มีเอ้าพุตเป็นระยะคลาดเคลื่อนแบบยุคลิดอยู่แล้วจึงสามารถพล็อตเอ้าพุตเทียบกับจำนวนรอบการคำนวณได้เลย)



(ก.)

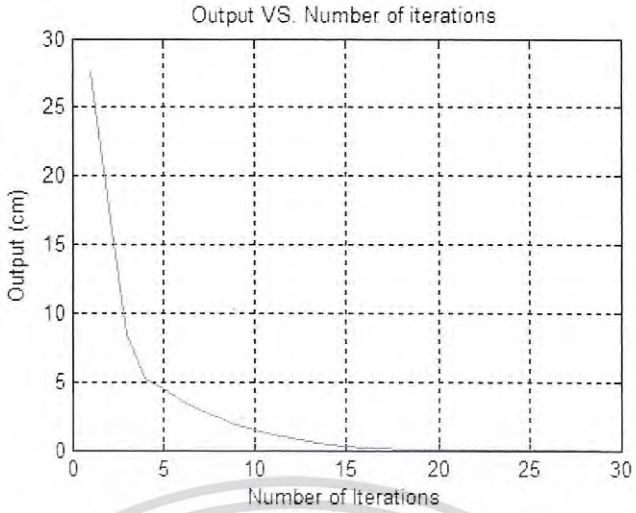


(ข.)

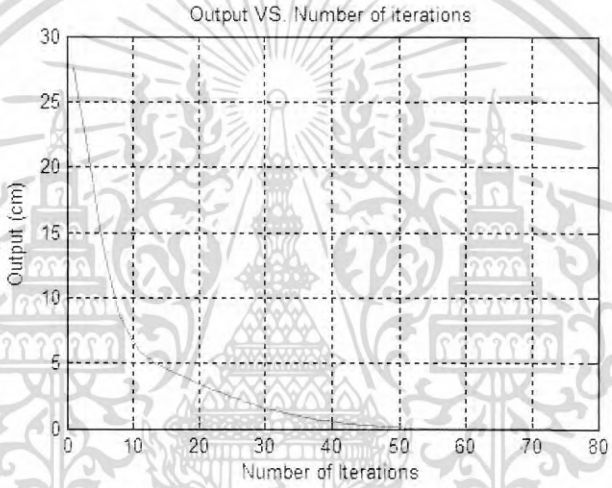


(ค.)

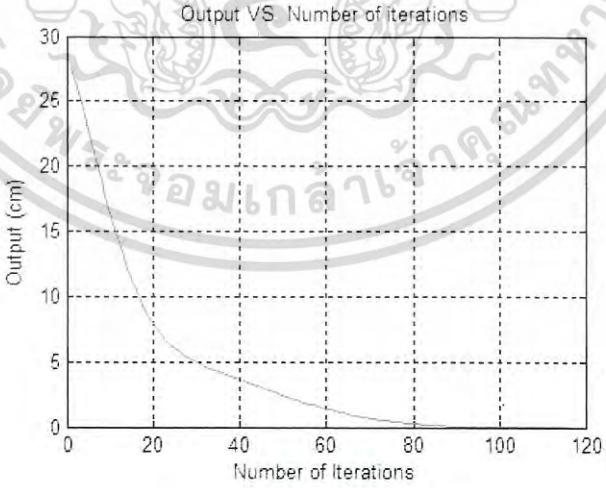
รูปที่ 5.3 เส้นทางการเคลื่อนที่สู่ตำแหน่งเป้าหมายที่ค่าอัตราการเรียนรู้ต่างกัน
(ก.) *lr*=0.001,จำนวนรอบ=29 (ข.) *lr*=0.0003,จำนวนรอบ=73 (ค.) *lr*=0.0001,จำนวนรอบ=117



(ก.)



(ข.)



(ค.)

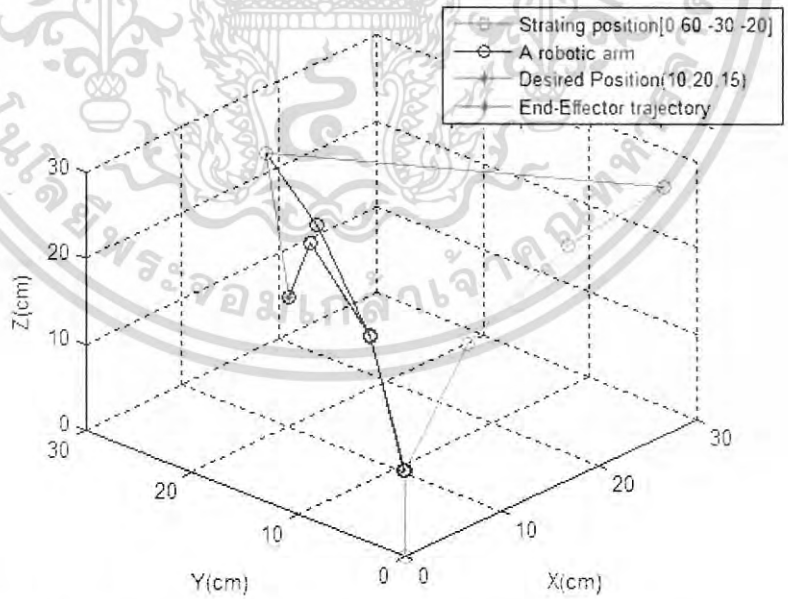
รูปที่ 5.4 การลดลงของเอ้าพุตที่ค่าการเรียนรู้ต่างกัน (ก.) *lr*=0.001,จำนวนรอบ=29 (ข.) *lr*=0.0003,จำนวนรอบ=73 (ค.) *lr*=0.0001,จำนวนรอบ=117

นอกจากโครงข่ายประสาทเทียมแบบที่ 2 จะสามารถเคลื่อนที่เข้าสู่ตำแหน่งเป้าหมายใน 3 มิติได้นั้นเรายังสามารถปรับค่าอัตราการเรียนรู้เริ่มต้น เพื่อปรับความละเอียดของการเคลื่อนที่เข้าหาตำแหน่งเป้าหมายได้อีกด้วย โดยการใช้ค่าอัตราการเรียนรู้เริ่มต้นมากแขนกลการจะสามารถเคลื่อนที่เข้าหาตำแหน่งเป้าหมายได้เร็วแต่เส้นทางการเคลื่อนที่อาจจะไม่ต่อเนื่อง แต่ถ้าหากใช้อัตราการเรียนรู้เริ่มต้นต่ำการเคลื่อนที่จะใช้เวลามากขึ้นแต่เส้นทางการเคลื่อนที่จะต่อเนื่องมากขึ้นตาม โดยเส้นทางการเคลื่อนที่เข้าหาตำแหน่งเป้าหมายที่อัตราการเรียนรู้เริ่มต้นจากมากไปน้อยถูกแสดงไว้ในรูปที่ 5.3 และ 5.4 (ก.),(ข.),(ค.) ตามลำดับ

การเคลื่อนที่เข้าสู่ตำแหน่งเป้าหมายใน 3 มิติโดยตรงในลักษณะนี้คล้ายกับมนุษย์เอื้อมมือไปหยิบจับสิ่งของ ซึ่งยังไม่มีวิธีการแก้ปัญหาจลนศาสตร์ผกผันในปัจจุบันที่สามารถเคลื่อนที่เข้าหาตำแหน่งเป้าหมายได้ในลักษณะนี้ นอกจากนั้นการเคลื่อนที่เข้าสู่ตำแหน่งเป้าหมายในลักษณะนี้อาจสามารถนำไปประยุกต์ให้แขนกลเคลื่อนที่สู่ตำแหน่งเป้าหมายโดยสามารถหลบหลีกสิ่งกีดขวางได้

5.1.3 การลู่เข้าคำตอบของวิธี FABRIK

เนื่องจากวิธี FABRIK ออกแบบการแก้ปัญหาในระนาบ เส้นทางการเคลื่อนที่เข้าหาตำแหน่งเป้าหมายของวิธีนี้จะมีลักษณะคล้ายวิธีที่ต้องการนำเสนอแบบที่ 1 นั่นคือเราต้องทำการสมมติให้แขนกลอยู่ในระนาบเดียวกับตำแหน่งเป้าหมายก่อนจึงเริ่มทำการแก้ปัญหาได้แบบปกติในระนาบนั้นๆ รูปที่ 5.5 แสดงเส้นทางการลู่เข้าตำแหน่งเป้าหมายโดยกำหนดให้จุดเริ่มต้นและตำแหน่งเป้าหมายเดียวกันกับการทดลองในรูปที่ 5.3



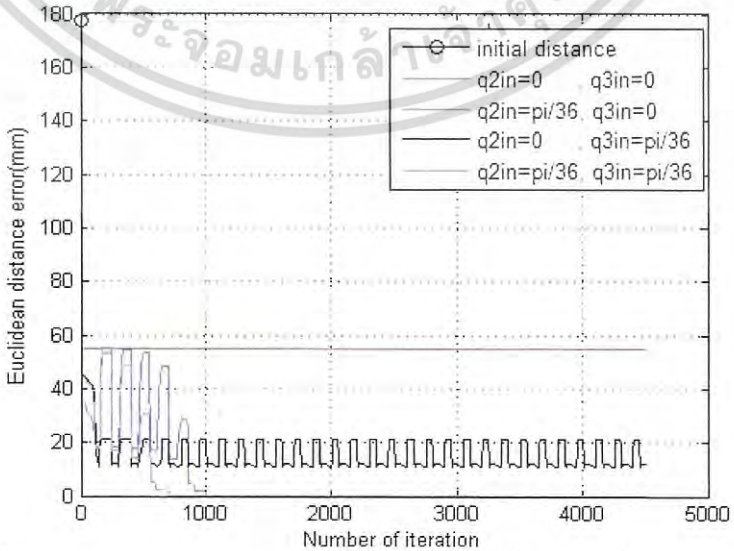
รูปที่ 5.5 เส้นทางการเคลื่อนที่สู่ตำแหน่งเป้าหมายของวิธี FABRIK

5.2 ทดสอบผลของค่าเริ่มต้นและตำแหน่งเป้าหมายต่อการลู่เข้าของคำตอบ

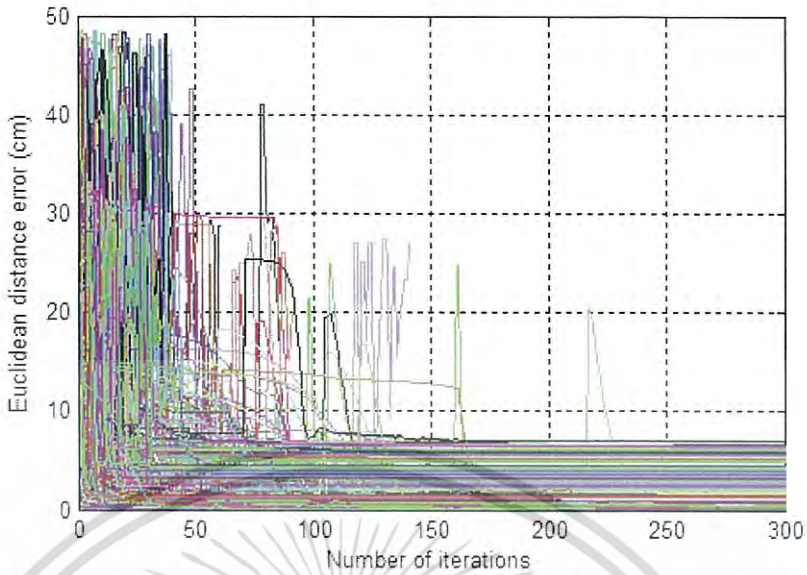
ขอบเขตของค่าเริ่มต้นเป็นสิ่งที่สำคัญมากเรื่องหนึ่งสำหรับระเบียบวิธีเชิงตัวเลข (Numerical method) เพราะค่าเริ่มต้นบางค่าอาจทำให้ระบบไม่ลู่เข้าหรือลู่เข้าช้าได้ เพราะฉะนั้นการทดลองนี้จะทำการใส่ค่ามุมเริ่มต้นเป็นศูนย์และแบบสุ่มค่าแล้ววาดกราฟระยะคลาดเคลื่อนระหว่างจุดปลายแขนกลกับตำแหน่งเป้าหมาย (โดยสมการระยะทางแบบยูคลิด) เทียบกับจำนวนรอบในการคำนวณ เพื่อดูลักษณะการลู่เข้าคำตอบแล้วนำไปพิจารณาการกำหนดขอบเขตค่ามุมเริ่มต้นที่เหมาะสม ในส่วนของค่ามุมและตำแหน่งเป้าหมายแบบสุ่มจะใช้ฟังก์ชันสุ่มค่าของโปรแกรม MATLAB สุ่มค่ามุมเริ่มต้นมา 1000 ชุด

5.2.1 ทดสอบผลของค่าเริ่มต้นและตำแหน่งเป้าหมายกับโครงข่ายประสาทเทียมแบบที่ 1

เราจะทำการทดลองการลู่เข้าเมื่อค่ามุมเริ่มต้นเป็นศูนย์ทั้งหมด และเป็นศูนย์แค่มุมใดมุมหนึ่ง ซึ่งจากการทดลองพบว่าโครงข่ายประสาทเทียมแบบที่ 1 ไม่สามารถลู่เข้าตำแหน่งเป้าหมายได้เมื่อมุมเริ่มต้นเป็นศูนย์ทั้งหมดเพราะจากสมการ (4.11) และ (4.12) เมื่อค่ามุมเริ่มต้น $q_1=0, q_2=0$ โครงข่ายจะไม่สามารถปรับค่าเอ้าพุตได้ด้วยค่าน้ำหนัก ดูได้จากกราฟสีแดงในรูปที่ 5.6 นอกจากนั้นเมื่อกำหนดให้ค่ามุมเริ่มต้นตัวใดตัวหนึ่งเป็นศูนย์เพียงตัวเดียว โครงข่ายก็จะพยายามลู่เข้าคำตอบด้วยการปรับค่าน้ำหนักของมุมที่ไม่เป็นศูนย์ โดยในรูปที่ 5.6 จากกราฟสีน้ำเงินพบว่าโครงข่ายสามารถลู่เข้าตำแหน่งเป้าหมายได้ด้วยการปรับมุม q_2 เพียงอย่างเดียวได้ แต่จากราฟเส้นสีดำโครงข่ายไม่สามารถลู่เข้าตำแหน่งเป้าหมายได้ด้วยการปรับมุม q_3 เพียงอย่างเดียวได้ ต่อมารูปที่ 5.7 แสดงการลู่เข้าตำแหน่งเป้าหมายเดียวกันที่ค่ามุมเริ่มต้นแบบสุ่ม 1000 ชุด พบว่าค่ามุมเริ่มต้นมีผลต่อการลู่เข้าในโครงข่ายแบบที่ 1 อย่างมาก โดยค่ามุมเริ่มต้นบางค่าทำให้โครงข่ายไม่สามารถลู่เข้าได้ดังที่ได้กล่าวไปก่อนหน้านี้แล้วว่าโครงข่ายแบบที่ 1 นี้ไม่มีการรับประกันการว่าโครงข่ายของตำแหน่งจุดปลายแขนกลทั้ง 3 จะถูกปรับไปในทิศทางเดียวกัน



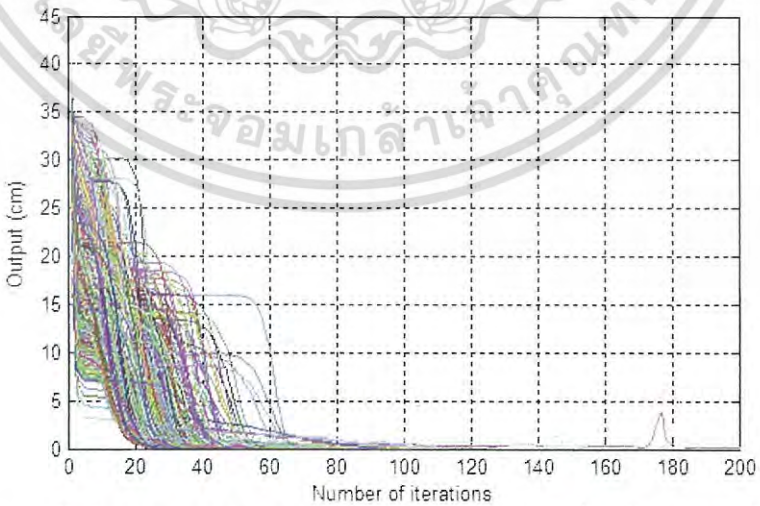
รูปที่ 5.6 การลู่เข้าคำตอบของโครงข่ายแบบที่ 1 เมื่อค่ามุมเริ่มต้นเป็นศูนย์



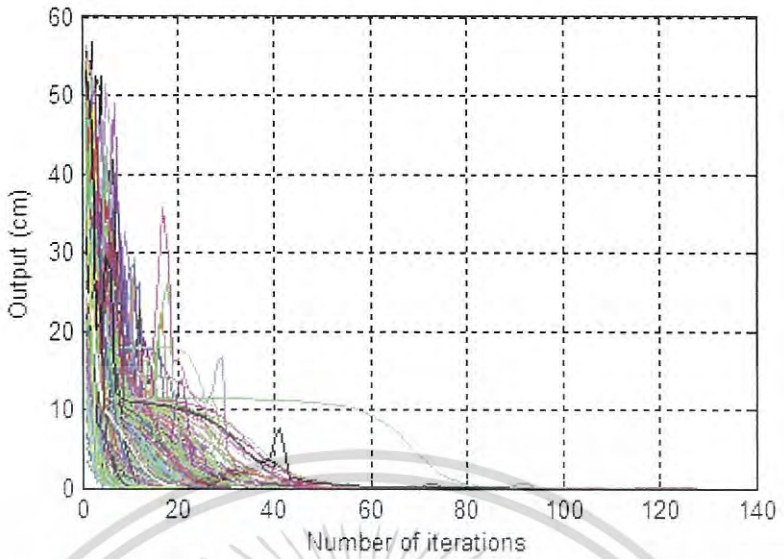
รูปที่ 5.7 การลู่เข้าคำตอบของโครงข่ายแบบที่ 1 ที่ค่ามุมเริ่มต้นแบบสุ่ม 1,000 ค่า

5.2.2 ทดสอบผลของค่าเริ่มต้นและตำแหน่งเป้าหมายกับโครงข่ายประสาทเทียมแบบที่ 2

ทำการทดลองกำหนดค่ามุมเริ่มต้นเป็นศูนย์เหมือนกับโครงข่ายแบบที่ 1 ซึ่งจากที่ได้กล่าวไว้ในหัวข้อ 4.2.3 ว่าโครงข่ายประสาทเทียมแบบที่ 2 สามารถแก้ปัญหาค่ามุมเริ่มต้นเป็นศูนย์ด้วยการเพิ่มไบอัสในชั้นซ่อนที่ 1 จึงทำให้โครงข่ายสามารถลู่เข้าสู่ตำแหน่งเป้าหมายแบบสุ่มจำนวน 1,000 ตำแหน่งได้ดังแสดงในรูปที่ 5.8 นอกจากนี้โครงข่ายประสาทเทียมแบบที่ 2 ยังสามารถลู่เข้าสู่ตำแหน่งเป้าหมาย ณ ค่ามุมเริ่มต้นที่แตกต่างกัน 1,000 ค่าอีกด้วย ค่าผลลัพธ์เฉลี่ยของเวลาและจำนวนรอบในการคำนวณถูกเปรียบเทียบกับวิธี FABRIK และแสดงไว้ในตารางที่ 5.3 และ 5.4

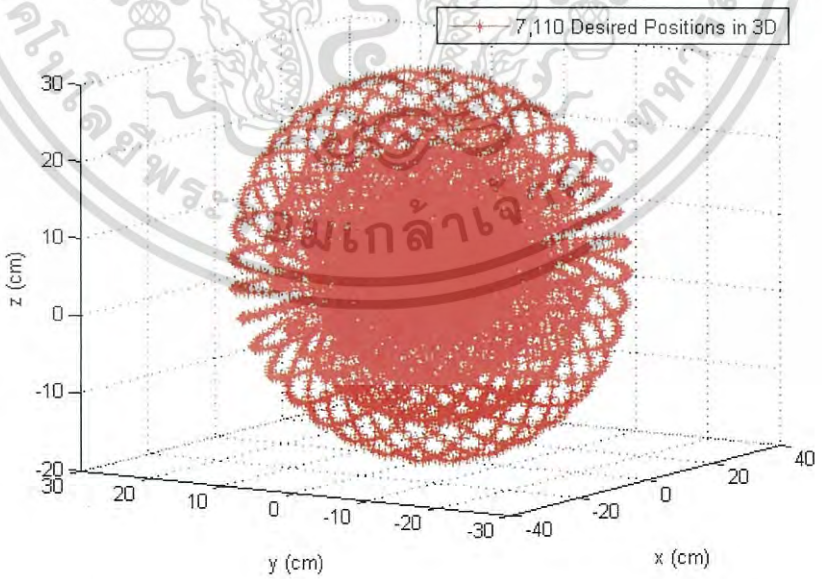


รูปที่ 5.8 การลู่เข้าคำตอบแบบสุ่ม 1,000 ค่าของโครงข่ายแบบที่ 2 ด้วยค่ามุมเริ่มต้นเป็นศูนย์



รูปที่ 5.9 การลู่เข้าคำตอบของโครงข่ายแบบที่ 2 ที่ค่ามุมเริ่มต้นแบบสุ่ม 1,000 ค่า

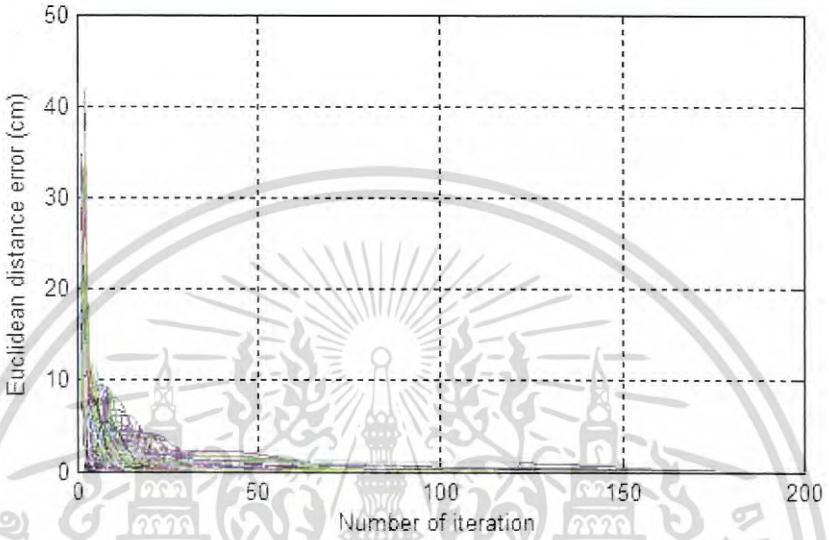
เนื่องจากโครงข่ายประสาทเทียมแบบที่ 2 สามารถลู่เข้าคำตอบได้ดี เราจึงได้ทำการทดลองเพิ่ม โดยให้โครงข่ายแบบที่ 2 ลู่เข้าตำแหน่งเป้าหมาย 7,110 จุดซึ่งครอบคลุมทั่วทั้งพื้นที่การทำงานของแขนกลเพื่อหาตำแหน่งเป้าหมายที่โครงข่ายไม่สามารถลู่เข้าได้ แต่ปรากฏว่าโครงข่ายประสาทเทียมแบบที่ 2 สามารถลู่เข้าเป้าหมายได้ทั้งหมดด้วยค่าจำนวนรอบการคำนวณเฉลี่ย 52.09 รอบด้วยค่าอัตราการเรียนรู้ 0.0019 และค่ามุมเริ่มต้นเป็น $q_{in}=[0\ 90\ 0\ 0]$ องศา



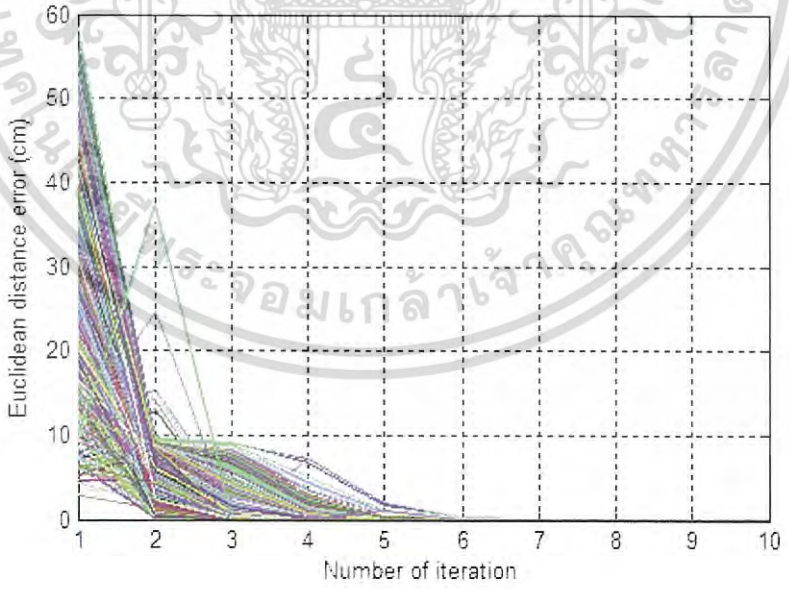
รูปที่ 5.10 ตำแหน่งเป้าหมาย 7,110 ตำแหน่งในพื้นที่ทำงานของแขนกล

5.2.3 ทดสอบผลของค่าเริ่มต้นและตำแหน่งเป้าหมายกับวิธี FABRIK

รูปที่ 5.11 แสดงการลู่เข้าสู่ตำแหน่งเป้าหมายแบบสุ่มเมื่อมุมเริ่มต้นเป็นศูนย์และรูปที่ 5.12 แสดงการลู่เข้าแบบสุ่มค่ามุมเริ่มต้น สังเกตได้ว่าการลู่เข้าแบบสุ่มค่ามุมเริ่มต้นจะใช้การคำนวณน้อยมาก โดยค่าเฉลี่ยจำนวนรอบการคำนวณจะอยู่ที่ 6-7 รอบเท่านั้น นอกจากนี้เวลาและจำนวนรอบในการคำนวณเฉลี่ยจะถูกเปรียบเทียบและแสดงไว้ในตารางที่ 5.3 และ 5.4



รูปที่ 5.11 การลู่เข้าคำตอบแบบสุ่ม 1,000 ค่าของวิธี FABRIK เมื่อมุมเริ่มต้นเป็นศูนย์



รูปที่ 5.12 การลู่เข้าคำตอบของวิธี FABRIK ที่ค่ามุมเริ่มต้นแบบสุ่ม 1,000 ค่า

ตารางที่ 5.3 ผลลัพธ์การลู่เข้าคำตอบของการสุ่มค่ามุมเริ่มต้น 1000 ค่า

ค่าคลาดเคลื่อน (mm.)	โครงข่ายประสาทเทียมแบบที่ 2		FABRIK	
	เวลาในการคำนวณ(ms.)	จำนวนรอบในการคำนวณ	เวลาในการคำนวณ(ms.)	จำนวนรอบในการคำนวณ
0.01	1.976	35.735	0.103	6.101
0.0001	2.589	46.852	0.131	8.022
0.00001	2.945	52.293	0.144	8.956

ตารางที่ 5.4 ผลลัพธ์การลู่เข้าคำตอบของการสุ่มค่าตำแหน่งเป้าหมาย 1000 ตำแหน่ง

ค่าคลาดเคลื่อน (mm.)	โครงข่ายประสาทเทียมแบบที่ 2		FABRIK	
	เวลาในการคำนวณ(ms.)	จำนวนรอบในการคำนวณ	เวลาในการคำนวณ(ms.)	จำนวนรอบในการคำนวณ
0.01	2.728	49.529	0.407	23.718
0.0001	3.261	59.496	0.549	34.068
0.00001	3.476	63.405	0.657	39.236

5.3 ทดสอบประสิทธิภาพของระบบด้วยการติดตามเส้นวิถี

เพื่อทดสอบประสิทธิภาพของวิธีที่นำเสนอ รายงานฉบับนี้ได้ทดลองกับแขนกลจำลองในโปรแกรม MATLAB โดยทดลองให้แขนกลติดตามเส้นวิถีแบบต่างๆ แล้วทำการนับจำนวนรอบและจับเวลาในการคำนวณ โดยให้แขนกลเคลื่อนที่ตามเส้นทางต่างๆดังต่อไปนี้

สมการคลื่นไซน์บนระนาบ zy ช่วง $y=[-20/3, 20/3]$

$$z = 5 + A\sin(0.3\pi y) \tag{5.1}$$

โดยที่ A คือค่าแอมพลิจูดของกราฟไซน์

สมการวงกลมบนระนาบ xy โดยที่ค่า z และรัศมี r จะเปลี่ยนค่าไปพร้อมๆกันคล้ายสมการก้นหอยใน 3 มิติ ซึ่งแต่ละจุดบนเส้นกราฟจะมีการเปลี่ยนแปลงของทุกพิกัด x,y,z

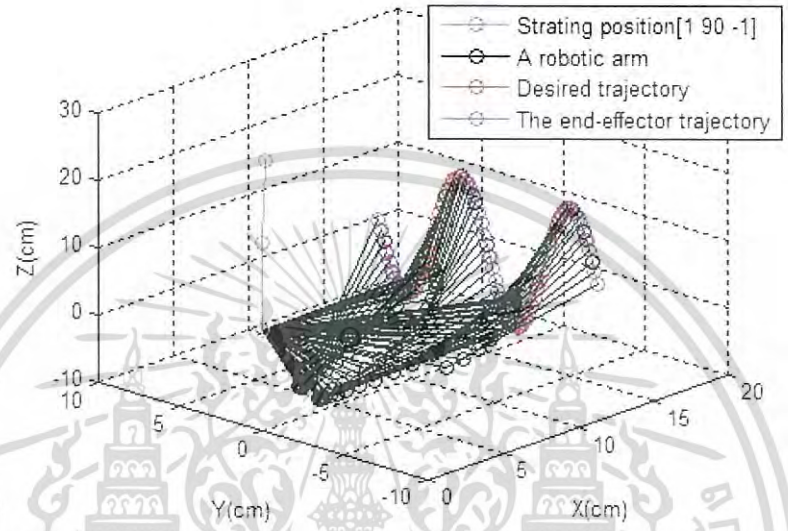
$$x^2 + y^2 = r^2 \tag{5.2}$$

ผลลัพธ์การคำนวณของทั้ง 3 วิธีจะถูกเปรียบเทียบกันในตารางที่ 5.5

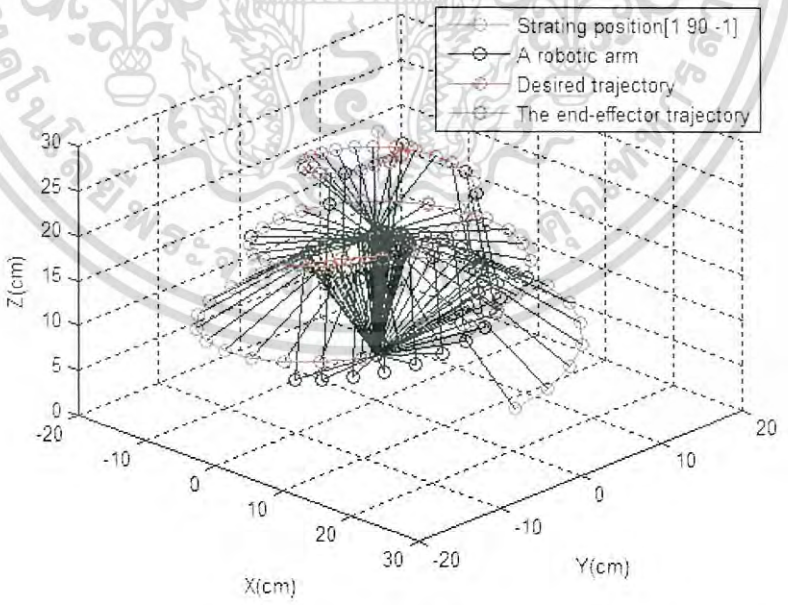
5.3.1 การเคลื่อนที่ติดตามเส้นกราฟของโครงข่ายประสาทเทียมแบบที่ 1

เนื่องจากโครงข่ายแบบที่ 1 ทำการทดลองกับแขนกลที่มี 3 องศาอิสระและมี 2 ท่อน จึงให้แขนกลเคลื่อนที่ตามกราฟไซน์และกราฟก้นหอยที่มีขนาดเล็กกว่ากราฟที่ใช้ทดสอบกับโครงข่าย

ประสาทเทียมแบบที่ 2 และวิธี FABRIK โดยกำหนดให้แอมพลิจูดเท่ากับ 10 และค่ามุมเริ่มต้น $q_1=1, q_2=90, q_3=-1$ องศาดังแสดงในรูปที่ 5.13 และให้แขนกลเคลื่อนที่ตามเส้นทางกราฟก้นหอย ดังแสดงในรูปที่ 5.14 จากตารางที่ 5.5 จะพบว่าโครงข่ายแบบที่ 1 อาจใช้เวลาในการคำนวณน้อย โครงข่ายแบบที่ 2 แต่มีบางจุดที่โครงข่ายแบบที่ 1 ลู่เข้าคำตอบได้ไม่ตรงตามค่าเป้าหมายหรือมีค่าคลาดเคลื่อนสะสมเยอะกว่าแบบที่ 2



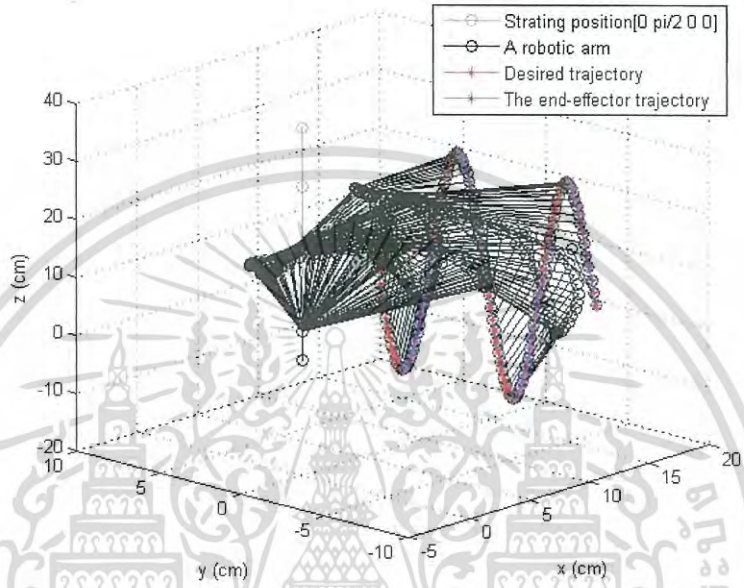
รูปที่ 5.13 โครงข่ายประสาทเทียมแบบที่ 1 เคลื่อนที่ตามเส้นทางกราฟไซน์



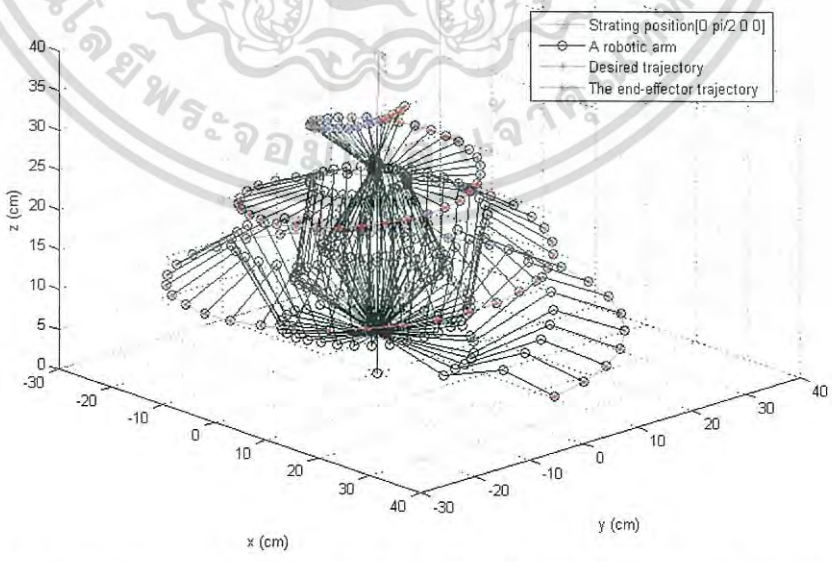
รูปที่ 5.14 โครงข่ายประสาทเทียมแบบที่ 1 เคลื่อนที่ตามเส้นทางกราฟก้นหอยใน 3 มิติ

5.3.2 การเคลื่อนที่ติดตามเส้นกราฟของโครงข่ายประสาทเทียมแบบที่ 2

โครงข่ายแบบที่ 2 ทำการทดลองกับแขนกลที่มี 4 องศาอิสระ จึงให้แขนกลเคลื่อนที่ตามกราฟไซน์โดยมีแอมพลิจูดเท่ากับ 20 และค่ามุมเริ่มต้น $q_1=0, q_2=90, q_3=0, q_4=0$ องศาดังแสดงไว้ในรูปที่ 5.15 และให้แขนกลเคลื่อนที่ติดตามกราฟก้นหอยใน 3 มิติดังแสดงในรูปที่ 5.16 ค่าเวลาและจำนวนรอบในการคำนวณดูได้ในตารางที่ 5.5



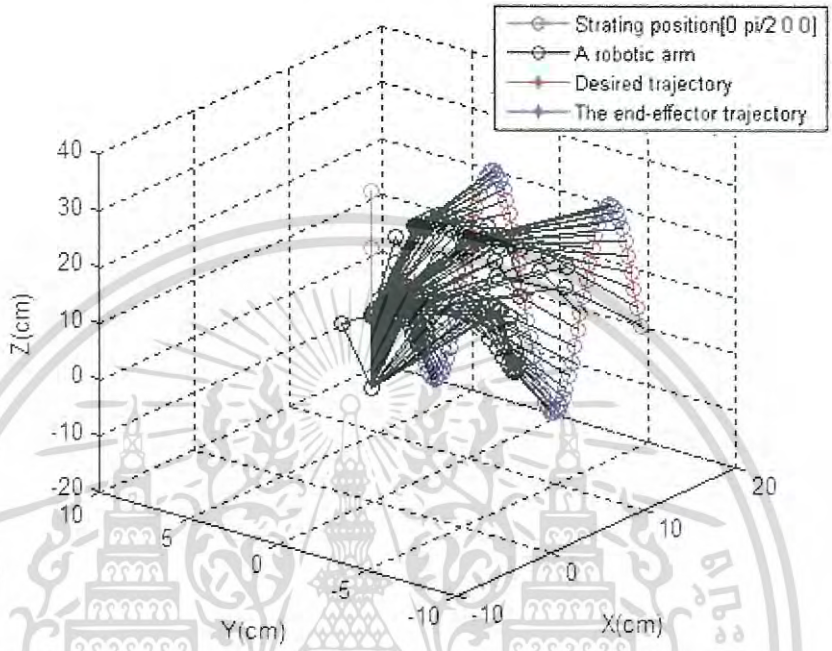
รูปที่ 5.15 โครงข่ายประสาทเทียมแบบที่ 2 เคลื่อนที่ตามเส้นทางกราฟไซน์



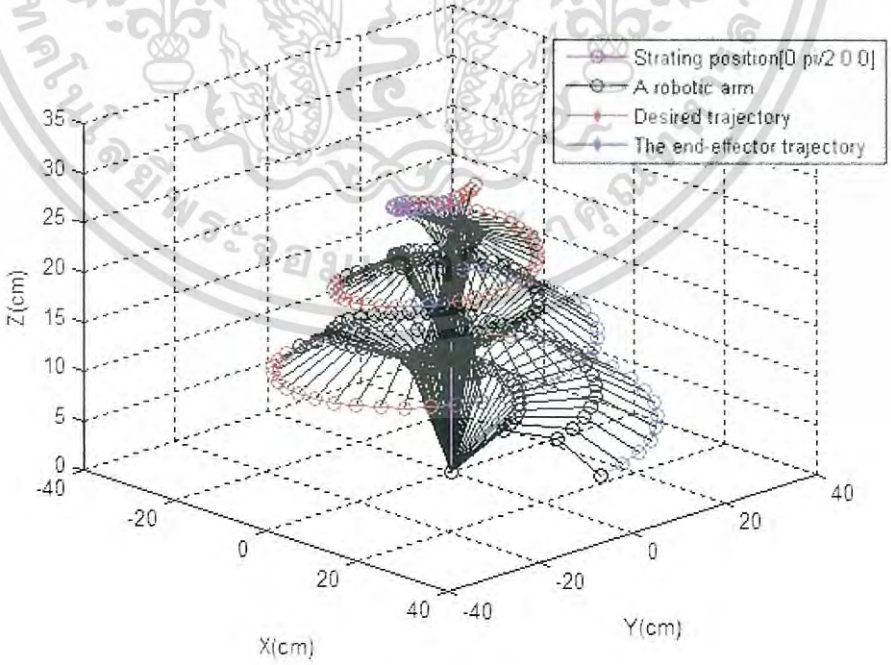
รูปที่ 5.16 โครงข่ายประสาทเทียมแบบที่ 2 เคลื่อนที่ตามเส้นทางกราฟก้นหอยใน 3 มิติ

5.3.3 การเคลื่อนที่ติดตามเส้นกราฟของวิธี FABRIK

วิธี FABRIK สามารถติดตามกราฟทั้ง 2 ชนิดดังแสดงในรูปที่ 5.17 และ 5.18 อีกทั้งยังใช้เวลาในการคำนวณน้อยที่สุดดังแสดงในตารางที่ 5.5 ในบทต่อไปเราจะสรุปผลการทดลองพร้อมทั้งกล่าวถึงจุดแตกต่างระหว่างวิธีที่นำเสนอกับวิธีอื่นๆ



รูปที่ 5.17 การเคลื่อนที่ของวิธี FABRIK ตามเส้นทางกราฟไซน์



รูปที่ 5.18 การเคลื่อนที่ของวิธี FABRIK ตามเส้นทางกราฟก้นหอยใน 3 มิติ

ตารางที่ 5.5 ผลลัพธ์การเคลื่อนที่ติดตามเส้นกราฟของทั้ง 3 วิธี

เส้นกราฟ	จำนวนจุดบนเส้นกราฟ	โครงข่ายแบบที่ 1		โครงข่ายแบบที่ 2		FABRIK	
		จำนวนรอบ	เวลาคำนวณ (s)	จำนวนรอบ	เวลาคำนวณ (s)	จำนวนรอบ	เวลาคำนวณ (s)
sine wave	40	157218	0.4306	1132	0.0639	242	0.0058
	80	303900	0.7767	2064	0.1149	486	0.0185
	120	424736	1.1194	2963	0.1654	721	0.0204
3D Spiral	40	65465	0.1385	3466	0.2131	514	0.0168
	80	139182	0.2464	6107	0.3545	1167	0.0236
	120	747399	0.5356	8488	0.4902	1884	0.0326

จากตารางวิธี FABRIK อาจใช้เวลาในการแก้ปัญหาน้อยที่สุด แต่วิธี FABRIK มีพื้นฐานในการแก้ปัญหาในระนาบ ซึ่งเมื่อนำไปใช้แก้ปัญหาในระบบ 3 มิติ วิธีนี้จำเป็นต้องคำนวณค่า q_1 ด้วยวิธีการอื่นแล้วนำไปป้อนให้กับระบบก่อนจะทำการคำนวณด้วยวิธี FABRIK ตามปกติ แต่โครงข่ายประสาทเทียมแบบที่ 2 ถูกออกแบบมาให้สามารถหาค่ามุม q_1 ได้เอง หรือออกแบบให้ลู่เข้าคำตอบใน 3 มิติเป็นพื้นฐาน ตัวแปรในการปรับเปลี่ยนจึงมีมากกว่า เวลาในการคำนวณเลยเพิ่มขึ้นตามไปด้วย แต่จากผลลัพธ์การคำนวณของโครงข่ายประสาทเทียมแบบที่ 2 ในการทดลองข้างต้น พบว่าเวลาในการหาคำตอบแต่ละครั้งเฉลี่ยอยู่ในช่วง 1 - 2 มิลลิวินาทีเท่านั้น ซึ่งผู้วิจัยคิดว่าเวลาในการคำนวณระดับนี้เร็วกว่าความสามารถในการเคลื่อนไหวของแขนกลที่ใช้งานจริงอยู่มาก การนำอัลกอริทึมนี้ไปประยุกต์ใช้งานจริงจึงไม่น่าเป็นปัญหา

บทที่ 6
สรุปผลการทดลองและข้อเสนอแนะ

วิทยานิพนธ์ฉบับนี้ต้องการแก้ปัญหาจลนศาสตร์ผกผันใน 3 มิติด้วยโครงข่ายประสาทเทียมจากสมการจลนศาสตร์ผันตรง จากการวิจัยเราได้คิดค้นโครงข่ายประสาทเทียมขึ้นมาสองแบบ แบบที่ 1 คือโครงข่ายประสาทเทียม 3 โครงข่ายที่มีอินพุตเป็นมุมข้อต่อและเอ้าพุตเป็นค่าตำแหน่ง x,y,z ของจุดปลายแขนกลดังรูปที่ 4.1 ถึง 4.3 ซึ่งสามารถเคลื่อนที่เข้าสู่เป้าหมายโดยการปรับค่าโครงข่ายตำแหน่ง x,y,z ทีละโครงข่าย [3] แบบที่ 2 คือโครงข่ายประสาทเทียมที่มีอินพุตเป็นมุมข้อต่อและเอ้าพุตเป็นค่าคลาดเคลื่อนระหว่างจุดปลายแขนกลกับตำแหน่งเป้าหมาย (ระยะทางแบบยุคลิด) ดังรูปที่ 4.9 ซึ่งสามารถเคลื่อนที่เข้าสู่ตำแหน่งเป้าหมายโดยการปรับค่าโครงข่ายตำแหน่งทั้งสามไปพร้อมๆกัน [4] รายงานฉบับนี้ได้ทำการทดลองในบทที่ 5 เพื่อทดสอบประสิทธิภาพของโครงข่ายทั้งสองพบว่า

โครงข่ายแบบที่ 1 สามารถแก้ปัญหาจลนศาสตร์ผกผันใน 3 มิติได้โดยการคำนวณหาค่ามุม q_1 ด้วยหลักตรีโกณมิติแล้วนำไปใส่ในโครงข่ายเหมือนกับการหันแขนกลไปอยู่ในระนาบเดียวกับตำแหน่งเป้าหมายจากนั้นจึงทำการปรับปรุงค่าตำแหน่งจุดปลายแขนกลทีละตำแหน่งจนลู่เข้าเป้าหมาย ซึ่งมีข้อดีคือการใส่ค่ามุม q_1 ก่อนทำให้ลดความซับซ้อนของปัญหาลงเสมือนการคำนวณใน 2 มิติดังรูปที่ 5.2 แต่มีข้อเสียคือ ในขณะที่ทำการปรับปรุงโครงข่ายตำแหน่งใดตำแหน่งหนึ่งนั้นค่าคลาดเคลื่อนของตำแหน่งอื่นๆอาจจะเพิ่มขึ้นสวนทางกันเพราะวิธีนี้ทำการปรับโครงข่ายทีละตัวโดยไม่ได้นำค่าคลาดเคลื่อนของโครงข่ายตัวอื่นๆมาคิดร่วมด้วย จึงเป็นไปได้ว่าอาจจะมีบางตำแหน่งเป้าหมายที่ไม่สามารถลู่เข้าได้ดังรูปที่ 5.7 และการทดลองที่ 5.3.1

โครงข่ายแบบที่ 2 สามารถแก้ปัญหาจลนศาสตร์ผกผันได้ โดยสามารถเคลื่อนที่เข้าหาตำแหน่งเป้าหมายใน 3 มิติได้เหมือนกับมนุษย์เอื้อมมือไปหยิบจับสิ่งของดังแสดงในรูปที่ 5.3 ในขณะที่วิธีการแก้ปัญหาจลนศาสตร์ผกผันในปัจจุบันออกแบบการแก้ปัญหาในระนาบ และหากพิจารณาการเคลื่อนที่เข้าสู่ตำแหน่งเป้าหมายใน 3 มิติแบบนี้ อาจเป็นไปได้ว่าเราสามารถควบคุมเส้นทางการเคลื่อนที่ของแขนกลให้หลบหลีกสิ่งกีดขวางใน 3 มิติได้ด้วยการเพิ่มเงื่อนไขบางอย่างให้โครงข่าย นอกจากนั้นเนื่องจากโครงข่ายประสาทเทียมแบบที่ 2 รวมโครงข่ายตำแหน่ง x,y,z เข้าด้วยกันทำให้ค่าคลาดเคลื่อนของทั้งสามตำแหน่งถูกนำมาคิดร่วมด้วยขณะที่ทำการปรับปรุงโครงข่าย จึงเป็นการรับประกันการลู่เข้าได้ทางหนึ่งดังรูปที่ 5.8, 5.9 โครงข่ายประสาทเทียมแบบที่ 2 อาจจะไม่ใช่วิธีการแก้ปัญหาจลนศาสตร์ผกผันแบบใหม่ที่คำนวณได้เร็วกว่าวิธีในปัจจุบัน แต่จากผลการทดลองพบว่าความเร็วในการลู่เข้าของโครงข่ายแบบที่ 2 นั้นอยู่ในระดับมิลลิวินาที อีกทั้งความเร็วในการลู่เข้ายังสามารถเพิ่มได้โดยเลือกใช้วิธีการเรียนรู้ใหม่กับโครงสร้างเดิม ซึ่งการเรียนรู้ของโครงข่ายประสาทเทียมก็ถูกพัฒนาและคิดค้นขึ้นใหม่อย่างต่อเนื่อง [13,14]

เอกสารอ้างอิง

[1] S. Tejomurtula, S. Kak, "Inverse kinematics in robotics using neural networks," Information Sciences, vol.116, pp.147-164, 1999.

[2] A. Guez and Z. Ahmad, "ACCELERATED CONVERGENCE IN THE INVERSE KINEMATICS VIA MULTILAYER FEEDFORWARD NETWORKS," International Joint Conference on Neural Networks, pp.341-344, 1989.

[3] P. Srisuk, A. Sento, and Y. Kitjaidure "Inverse Kinematics Solution using Neural Networks from Forward Kinematics Equations," International Conference on Knowledge and Smart Technology, pp.61-65, 2017.,

[4] P. Srisuk, A. Sento, and Y. Kitjaidure "Forward Kinematic-like Neural Network for Solving the 3D Reaching Inverse Kinematics Problems," ," The 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, in press.,

[5] Martin T. Hagan, Howard B. Demuth and Mark Beale, Neural Network Design, PWS Publishing Company. 1995.

[6] B. Daya, S. Khawandi, and M. Akoum, "Applying Neural Network Architecture for Inverse Kinematics Problem in Robotics,"J.Software Engineer & Applications, vol. 3, pp.230-239, 2010

[7] S. S. Chiddarwar, N. R. Babu, "Comparison of RBF and MLP neural networks to solve inverse kinematic problem for 6R serial robot by a fusion approach," Engineering Applications of Artificial Intelligence, vol.23, pp.1083-1092, 2010.

[8] Y. Feng, W.Yao-nan, Y. Yi-min, "Inverse Kinematics Solution for Robot Manipulator based on Neural Network under Joint Subspace," International Journal of Computers, Communications & Control, vol.7, pp.459-472, 2012.

[9] R. Köker, "A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization," Information Sciences, vol.22, pp.528-543, 2013.

[10]A. V. Duka,"Neural network based inverse kinematics solution for trajectory tracking of a robotic arm," Procedia Technology, vol. 12, pp. 20 – 27, 2014.

[11] A. Aristidou, J. Lasenby, "FABRIK: A fast, iterative solver for the Inverse Kinematics problem" Graphical Models, vol. 73, pp.243-260, 2011.

[12] L.-C. T. Wang and C. C. Chen, "A combined optimization method for solving the inverse kinematics problems of mechanical manipulators," IEEE Transactions on Robotics and Automation, vol. 7, pp. 489–499, 1991.

[13] S.J.Subavathia and T.Kathirvalavakumar, "Adaptive modified backpropagation algorithm based on differential errors," International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol.1, No.5, pp.21-34, 2011.

[14] T. Kathirvalavakumar and S. J. Subavathi, "Modified Backpropagation Algorithm With Adaptive Learning Rate Based On Differential Errors And Differential Functional Constraints," International Conference on Pattern Recognition, Informatics and Medical Engineering, pp.61-67, 2012.

ภาคผนวก
ผลงานวิจัยที่ได้รับการตีพิมพ์

[1] P. Srisuk, A. Sento, and Y. Kitjaidure "Inverse Kinematics Solution using Neural Networks from Forward Kinematics Equations," International Conference on Knowledge and Smart Technology, pp.61-65, 2017.,

[2] P. Srisuk, A. Sento, and Y. Kitjaidure "Forward Kinematic-like Neural Network for Solving the 3D Reaching Inverse Kinematics Problems," International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2017.,

Inverse Kinematics Solution using Neural Networks from Forward Kinematics Equations

Pannawit Srisuk, Adna sento, Yuttana Kitjaidure
Department of Electronics Engineering, Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang, Bangkok, 10520, Thailand
Email:pannawit.ssuk@gmail.com

Abstract—**This paper presents the inverse kinematics solution using the neural network for a robotic arm in 3-dimension. This paper creates neural networks to represent x, y and z position of the end-effector in the forward kinematics equations. The structure of the network has 4 layers; input layer, 2 hidden layers, and output layer. The input and output layers are defined as robotic arm angle and position of the end-effector, respectively. Then, the network updates the weights by the backpropagation with variable learning rate algorithm until reaching criteria that the output of the network is equal to the desired positions. Finally, the inverse kinematics solution is defined by the optimal weights of the network. To evaluate the performance algorithm, the MATLAB Program is used to demonstrate the robotic arm movement in 3-dimension. As a result, the proposed algorithm can help the robotic arm move to the desired position quickly and correctly.**

Keywords-forward kinematics; inverse kinematics; neural network; robotic arm;

I. INTRODUCTION

Inverse kinematic solution is one of the most important problems in robotic control because this solution can provide angles of a robotic arm joints from desired positions of the end-effector. However, inverse kinematics solutions have some disadvantage such as solutions may not exist or have multiple solutions and the computation will more complexity when the number of robotic arm joints increased. For this reason, many research attempts to find the new method to solve inverse kinematics problem with less complexity computation, fast and accurate. The most popular method that used to solve inverse kinematics can classify as algebraic, geometric and iterative method.

The algebraic method is the original kinematics model that mainly used Cartesian space and represents the transformation between two coordinate in Cartesian space by a translation and a rotation. In the solution the system will define Denavit & Hartenberg (DH) parameters for all joints of a robotic arm to describe a robotic arm about types of joint or length in each link. Then, multiply position of the end-effector (or desired position) by orthonormal matrices (4x4 matrices) to find all position of joints in Cartesian space and use inverse trigonometric functions to solve angles of robotic arm joints[3].

The geometric method is suitable for a less DOF robotics arm because when a robotic arm has links more than 2 this method will require many complexity trigonometry equations in computation [4]. For example, this method can solve inverse kinematics for a 2-link robotic arm in 3-dimension with less computations by assuming a 2-link robotic arm as a triangle that place on the plane then, use the law of cosines to find angles in a robot arm joints[3].

The iterative method or numerical method will calculate the model repeatedly until output reached target. This method has a lot of advantage such as converge to a single solution that depends on starting point and easy to implement the model for a high DOF robotic arm. Examples of iterative methods are CCD [11], FABRIK [12], Nero-fuzzy and Neural Network. However, the most new method has design for solving inverse kinematics base on 2-dimension [11, 12]. Now, the popular new solutions are neural network approaches that training network with data of the end-effector positions and angles of joints to create the model [5,6,7,8,9]. This method requires a lot of data, many hidden layers and neurons for the accuracy of the solution. Moreover, this method still has problems to find the target that has multiple solutions in training data. Thus, this paper proposes the method to solve inverse kinematics in 3-dimesion without using a lot of data for training and reducing neural network structure [1, 2].

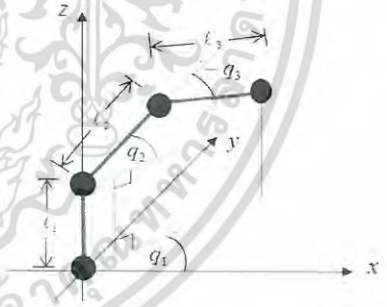


Figure 1. A 3 DOF robot arm

978-1-4673-9077-4/17/$31.00 ©2017 IEEE

II. The Proposed Neural Network Architecture for Inverse Kinematics Solution

The procedure of our proposed solution consists of 3 parts. First, finding forward kinematics equations of the end-effector. Second, using these equations to create the neural network models. Finally, updating the weight by the backpropagation with variable learning rate algorithm and using final weight as the solution of angles.

A. Forward Kinematics

The position of the end-effector in Cartesian space can be described by forward kinematics equations with trigonometry terms. For example, a 3 DOF robot arm in 3-dimension as shown in Fig. 1. The positions of x, y and z are relate to angle variables by equations as follows

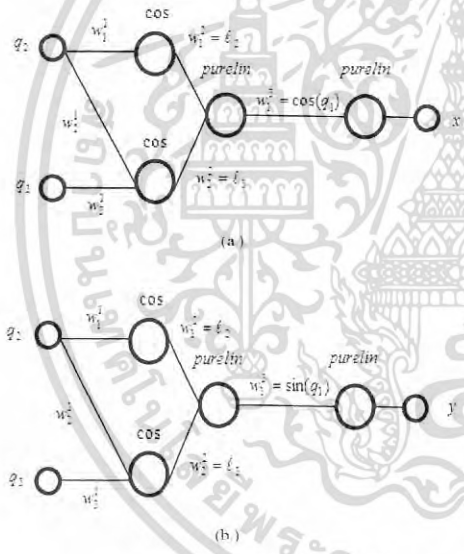
$$x_{end} = \cos(q_1)[\ell_2 \cos(q_2) + \ell_3 \cos(q_2 + q_3)] \quad (1)$$

$$y_{end} = \sin(q_1)[\ell_2 \cos(q_2) + \ell_3 \cos(q_2 + q_3)] \quad (2)$$

$$z_{end} = \ell_1 + \ell_2 \sin(q_2) + \ell_3 \sin(q_2 + q_3) \quad (3)$$

Where q, ℓ are angle variables and lengths of a robot arm link, respectively. These equations show a simple way to find the position of end-effector. Thus, this paper proposes the way to build the neural networks to represent x, y and z position of the end-effector in forward kinematics equations as shown in Fig. 2.

B. The Neural Network Model



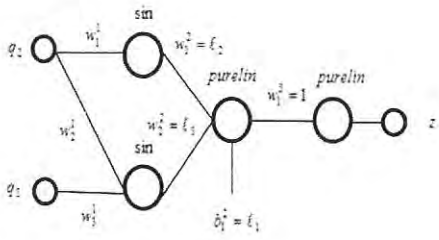


Figure 2. The neural network form forward kinematics of the end-effector (a.) x position (b.) y position (c.) z position.

The initial value of these neural network are setup as follows

- Input as angle variable (q_2, q_3) starting with random values except zero value.
- Output as x, y and z position of the end-effector
- Target is the desired positions of the end-effector
- Initial weight $w_1^1 = w_2^1 = w_3^1 = 1$, w_1^2 and w_2^2 are robotic arm links ℓ_2 and ℓ_3, respectively. $w_1^3=\cos(q_1)$ for neural x, $w_1^3=\sin(q_1)$ for neural y and $w_1^3=1$, $b_1^2 = \ell_1$ for neural z

In this model we use q_1 as a fixed weight (w_1^3) for the network of x and y position. To calculate q_1 observe the projection of a robotic arm on xy-plane and using arctan between the desired position (x_d,y_d) and the base of robot arm (x_{ref},y_{ref}).

$$q_1 = \arctan\left(\frac{y_d - y_{ref}}{x_d - x_{ref}}\right) \quad (4)$$

When q_1 is known the computation will decrease complexity the same as solving inverse kinematics problem in 2 dimensional similarly a robotic arm is placing on the planar in 3-dimension. Thus, the angle solution of network x and y will varies in the same direction.

The outputs of these networks are as follows

$$x = w_1^3[w_1^2 \cos(w_1^1 q_2) + w_2^2 \cos(w_2^1 q_2 + w_3^1 q_3)] \quad (5)$$

$$y = w_1^3[w_1^2 \cos(w_1^1 q_2) + w_2^2 \cos(w_2^1 q_2 + w_3^1 q_3)] \quad (6)$$

$$z = w_1^3[w_1^2 \sin(w_1^1 q_2) + w_2^2 \sin(w_2^1 q_2 + w_3^1 q_3) + b_1^2] \quad (7)$$

Substitutes initial weight in (5), (6) and (7). Then, these equations will become the position of the end-effector in forward kinematics equations as shown in (8), (9) and (10)

$$x = \cos(q_1)[\ell_2 \cos(w_1^1 q_2) + \ell_3 \cos(w_2^1 q_2 + w_3^1 q_3)] \quad (8)$$

$$y = \sin(q_1)[\ell_2 \cos(w_1^1 q_2) + \ell_3 \cos(w_2^1 q_2 + w_3^1 q_3)] \quad (9)$$

$$z = \ell_1 + \ell_2 \sin(w_1^1 q_2) + \ell_3 \sin(w_2^1 q_2 + w_3^1 q_3) \quad (10)$$

Now (8). (9) and (10) are the forward kinematics equations but the variable in these equations are weight (w_1^1, w_2^1, w_3^1) instead of the angles join (q_1, q_2, q_3). Thus. this neural network will update only weight in the input layer (w_1^1, w_2^1, w_3^1) as if adjust angles variable (q_2, q_3) in equations until output reached target.

C. The Operation

The proposed algorithm has many procedures as follows. First. training weight in network x by using input as random values in designed range and when the network finished updating weight the final weight updated in this network is compared with forward kinematic equations. Then. the solutions of angles are as follows

$$q_{2out} = w^1_{1final} q_{2in} \tag{11}$$

$$q_{3out} = (w^1_{2final} q_{2in} + w^1_{3final} q_{3in}) - q_{2out} \tag{12}$$

Where w_{final}, q_{in} and q_{out} are final weight values. input angles and output angles (or the solutions of angles), respectively. Second. training weight in the network y by using the solutions of angles of the network x (q_{2out}, q_{3out}) as inputs. Third. training weight in the network z by using the solutions of angles of the network y as inputs. Finally. Measure error between output of network z and desired positions. Then. repeat the process until error less than the expected value. Moreover. use the solution of the network z as the inputs of the network x in the next training weight.

III. Results

In this section. we conduct experiment about initial input angle condition. effect of q_1 and test the performance by tracking the desired trajectory.

First. Initial input value (or starting position) is very important for the iterative method. the solution may not exist or slow-convergence for some initial values. Thus. range of the initial value is important. Fig. 3 shows the characteristic of convergence when the initial values close to zero and fig. 4 shows convergence of random initial values. It is notice that when q_2=q_3=0 or q_2=0 the Euclidean distance between target and end-effector positions will not converge to the zero because when q_2=q_3=0 the output of network will fixed and the network cannot update output and when q_2=0 the network can update only q_3. Fig. 4 shows convergence of random initial value with error tolerance = 0.1 mm. This method will slow the convergence when the target is closed the boundary that a robotic arm can move. Fig. 5 shows the target at boundary is required a number of iterations more than another positions.

Second. the value of q_1 is calculated earlier and uses it as a fixed weight. This technique can instantaneously reduce a large number of errors before updating network as shown in fig. 6 and the solutions of angles of the neural x and y will varies in same direction. As a result. this method can reduce the computation by using only neural x and z or neural y and z. Fig. 7 shows when use only network x and z this method can reach the target with less number of iteration than using a full system. However. this way has limits for some of the target position. For example. let the target stay on plane x=0 the solution of q_1 is $\pi/2$ or $\pi/2$. Then. when q_1 is known the output of network x position of the end-effector is already reached the target and stop update weight whatever y positions still have error or not.

Finally. the performance is tested by letting a robotic arm track the desired trajectory. This paper creates two circle trajectories from parametric equation as follows

$$y = 10\cos t, z = 10\sin t + 10 \text{ (Circle on yz-plane) and}$$

$$x = 15\cos t, z = 15\sin t + 10 \text{ (Circle on xz-plane)}$$

In the simulation. the initial value are show in table 1. the parameters in backpropagations with variable learning rate algorithm are setup from [10] and the number of iterations in this method can measure by the number of iterations in each neural (normally is 50 iterates) multiplied by the number of times that repeat the process.

TABLE I. Initial value

	Initial value of neural		
	x	*y*	*z*
Learning rate	0.005	0.005	0.003
Increment factor	1.05	1.05	1.1
decrement factor	0.3	0.3	0.3
Momentum factor	0.2	0.2	0.2
Error (%)	5	5	5
$\ell_1(cm)$	5		
$\ell_2(cm)$	14		
$\ell_3(cm)$	13		

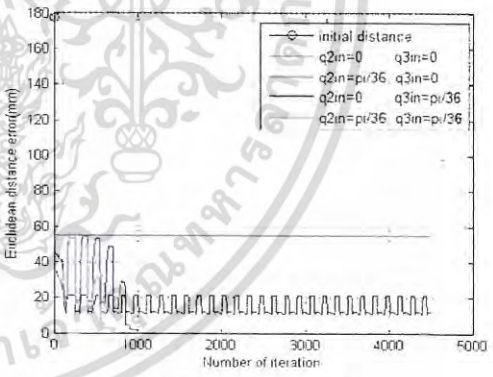


Figure 3. Effect of initial angles

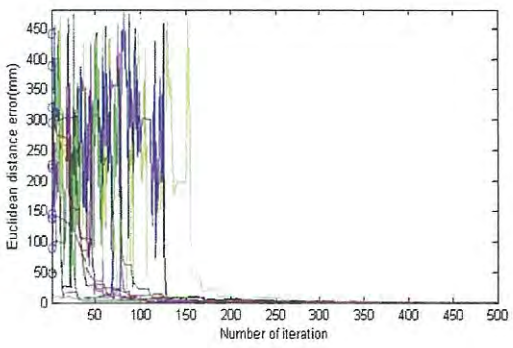


Figure 4. Convergence of random initial positions

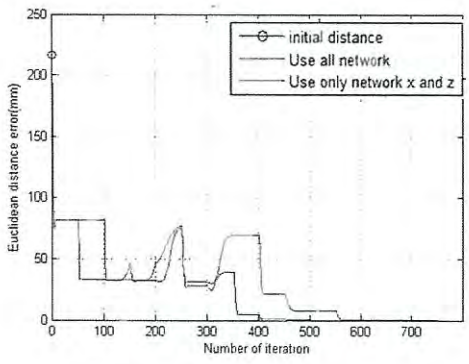


Figure 7. Reduce computation by use only neural x and z

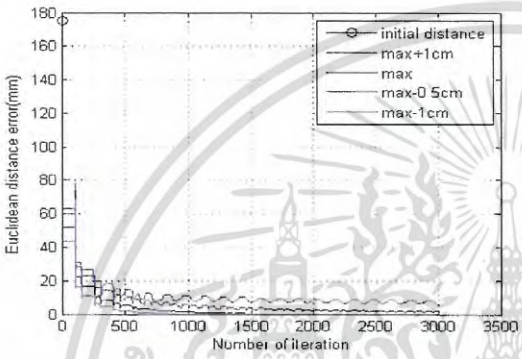


Figure 5. Convergence of the target at the boundary

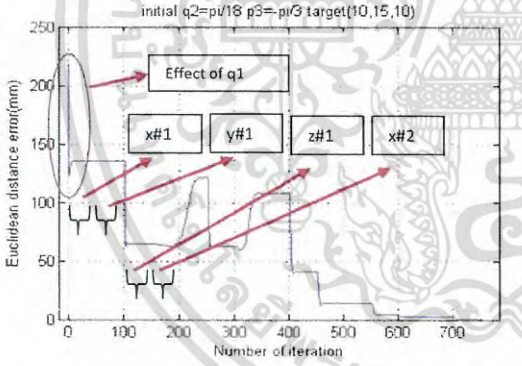


Figure 6. Euclidean error plot with 50 iterations for training weight in each neural and repeat the process 5 times.

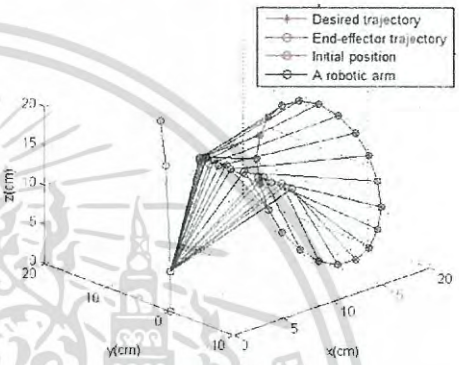


Figure 8. Circle trajectory tracking on yz-plane

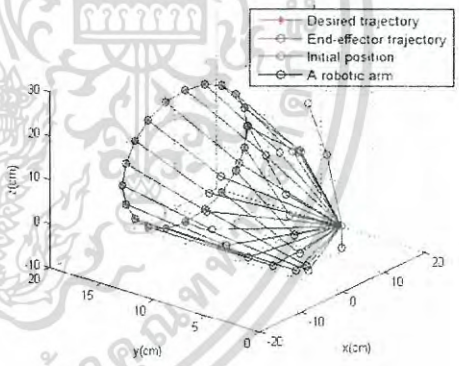


Figure 9. Circle trajectory tracking on xz-plane

IV. Conclusions

This paper can solve inverse kinematics of a 3-DOF robotic arm in 3-dimension spaces using the neural network creating from the forward kinematics equations with accuracy and the main difference between the proposed method and the other method is the proposed model design for solving inverse kinematics in 3-dimension. While, the other new model design for solving inverse kinematics based on 2-dimension such as CCD[11] and FABRIK[12]. Moreover, the proposed model easy to implement when the DOF of a robotic arm increased, the proposed model just increases only one neuron node in the first layer. So, we use less data training than [5,6,7,8,9] and the neural has small structure. Then, this method can reduce complexities in computation that can increases the speed of convergence.

In the future, we will improve the inverse kinematics algorithm based on the neural network to apply for the higher DOF of the robotic arm. We also plan to improve the learning algorithm of the neural network.

References

[1] S. Tejomurtula, S. Kak, "Inverse kinematics in robotics using neural networks," Information Sciences, vol.116, pp.147-164, 1999.

[2] S. Kieffer, V. Morellas, and M. Donath, "Neural Network Learning of the Inverse Kinematic Relationships for a Robot Arm," Robotics and Automation, pp. 2418-2425, 1991

[3] S. Kucuk, Z. Bingul, Industrial Robotics:Theory, Modelling and control , pIV pro literatur Verlag Robert Mayer-Scholz, pp.117-148, 2007.

[4] H. A. F. Mohamed, S. Yahya, M. Moghavvemi, and S. S. Yang, "A New Inverse Kinematics Method for Three Dimensional Redundant Manipulators," ICROS-SICE International Joint Conference, pp.1557-1562, 2009

[5] B. Daya, S. Khawandi, and M. Akoum, "Applying Neural Network Architecture for Inverse Kinematics Problem in Robotics,"J.Software Engineer & Applications, vol. 3, pp.230-239, 2010

[6] A. V. Duka,"Neural network based inverse kinematics solution for trajectory tracking of a robotic arm," Procedia Technology, vol. 12, pp. 20 – 27, 2014.

[7] T. Ogawa, H. Matsuura, and H. Kanada, "A Solution of Inverse Kinematics of Robot Arm Using Network Inversion," CIMCA-IAWTIC, 2005.

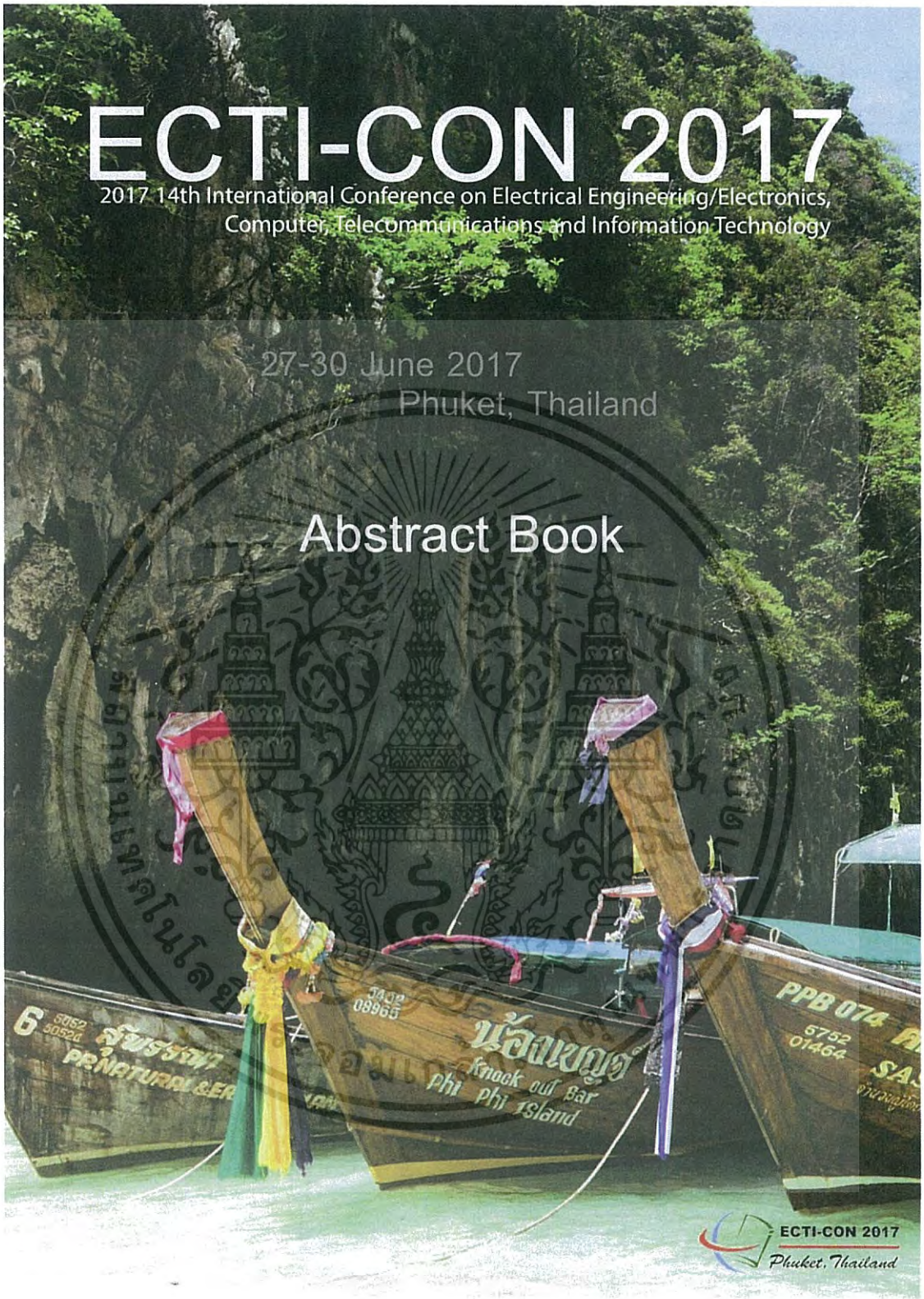
[8] S. F.M. Assal, K. Watanabe, and K. Izumi, "Neural Network Learning from Hint for the Inverse Kinematics Problem of Redundant Arm Subject to Joint Limits," Intelligent Robots and Systems, 2005.

[9] Z. Bingul, H.M. Ertunc, and C. Oysu, "Comparison of Inverse Kinematics Solutions Using Neural Network for 6R Robot Manipulator With Offset," Computational Intelligence Methods and Applications, 2005.

[10] M. T. Hagan, H. B. Demuth, M. H. Beale, and Orlando De Jesús, Neural Network Design, 2nd ed., Martin Hagan, 2014

[11] L.-C. T. Wang and C. C. Chen, "A combined optimization method for solving the inverse kinematics problems of mechanical manipulators," IEEE Transactions on Robotics and Automation, vol. 7, pp. 489–499, August 1991.

[12] A. Aristidou, J. Lasenby, "FABRIK: A fast, iterative solver for the Inverse Kinematics problem" Graphical Models, vol 73, pp.243-260, 2011



Forward Kinematic-like Neural Network for Solving the 3D Reaching Inverse Kinematics Problems

Pannawit Srisuk, Adna sento and Yuttana Kitjaidure
Department of Electronics Engineering, Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok, 10520, Thailand
E-mail: Pannawit.ssuk@gmail.com

Abstract—**This paper presents the inverse kinematic solutions based on neural networks. General neural network approaches use data of the end-effector positions as an input and angle joints as an output to train the neural network for mapping the input to the output. However, the proposed method creates the custom networks from forward kinematic equations. This special structure makes the network like a position finder with ability to automatically adjust angle joints until the end-effector reaches the desired position by backpropagation with variable learning rate algorithm. Then, the solutions of angles can be found from the final weights and bias values. Moreover, the proposed network use less number of neurons and amount of the solution space is not depend on the training data. Finally, to evaluate the performance algorithm, the MATLAB Program is used to demonstrate a 4-DOF robotic arm movement in 3-dimensional. As a result, the proposed algorithm can help a robotic arm move to the desired position (3D reaching) quickly and correctly.**

Keywords-forward kinematics; inverse kinematics; neural network; robotic arm; 3D reaching;

I. INTRODUCTION

Inverse kinematic solution is one of the most important problems in robotic control because this solution can provide angles of a robotic arm joints from desired positions of the end-effector. However, inverse kinematics solutions have some disadvantage such as solutions may not exist or have multiple solutions and the computation is more complex when the number of robotic arm joints increases. For this reason, many researches attempt to find the new method to solve inverse kinematic problem with less computation complexity, fast and accurate. The most popular method that used to solve inverse kinematics can be classified as algebraic, geometric and iterative method.

The algebraic method is the original kinematics model that mainly uses Cartesian space and represents the transformation between two coordinate in Cartesian space by translation and rotation. In the solution the system will define Denavit & Hartenberg (DH) parameters for all joints of a robotic arm to describe a robotic arm about types of joint or length in each link. Then, a position of the end-effector is multiplied (or desired position) by orthonormal matrices (4x4 matrices) to find all position of joints in Cartesian space and inverse trigonometric functions are used to solve angles of robotic arm joints[3].

The geometric method is suitable for less DOF robotic arms because when a robotic arm has links more than 2 this method will require many complex trigonometry equations in computation [4]. For example, this method can solve inverse kinematics for a 2-link robotic arm in 3-dimensional with less computations by assuming a 2-link robotic arm as a triangle that places on the plane and uses the law of cosines to find angles in the robot arm joints[3].

The iterative method uses the geometry and numerical to calculate the model repeatedly until the output reaches the target. This method has a lot of advantage such as converge to a single solution that depends on the starting point and easy to implement the model for a high DOF robotic arm. Examples of iterative methods are CCD [11], FABRIK [12], Nero-fuzzy and Neural Network [5,6,7,8,9]. However, the most of these method have been designed for solving inverse kinematics in plane [11,12] with limitation of the initial values and the target positions [5,6,7,8,9].

The popular new approaches are neural network approaches that training the network with data of the end-effector positions(input) and angles of joints(output) to mapping input to output [5,6,7,8,9]. However, searching space of this method has limited by the number of training data and the problem occurs when the training data have multiple solutions for the same end-effector position, the network has to select the optimum ones. Thus, this paper proposes the method to solve inverse kinematics in 3-dimensional using the custom neural network built from forward kinematic equations. By this way, the structure has less number of hidden nodes and layers, the searching space of this network will not depend on the training data. Moreover, the backpropagation with variable learning rate algorithm is used to guide the search direction. Thus, fast and accurate convergence can be guaranteed while the other numerical methods do not.

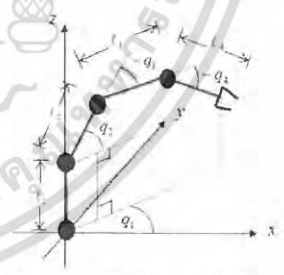


Fig. 1. A 4 DOF robotic arm in 3-dimensional

978-1-5386-0449-6/17/$31.00 ©2017 IEEE

II. The Proposed Neural Network Architecture for Inverse Kinematic Solutions

The procedure of our proposed technique consists of 3 parts. First, finding forward kinematic equations of a desired robotic arm. Second, using these equations to create the custom neural network models. Finally, training the network by the backpropagation with variable learning rate algorithm (BPVLR) and finding the solution of angles from the final weights and bias of the network.

A. Forward Kinematics

The position of the end-effector in the Cartesian space (x_{end},y_{end},z_{end}) can be described by forward kinematic equations with trigonometry terms. For example, the positions of the end-effector of a 4 DOF robotic arm with hinge joints as shown in fig. 1 are related to the angle variables by the following equations

$$x_{end} = \cos(q_1)[\ell_2\cos(q_2)+\ell_3\cos(q_2+q_3)+\ell_4\cos(q_2+q_3+q_4)] \quad (1)$$

$$y_{end} = \sin(q_1)[\ell_2\cos(q_2)+\ell_3\cos(q_2+q_3)+\ell_4\cos(q_2+q_3+q_4)] \quad (2)$$

$$z_{end} = \ell_1+\ell_2\sin(q_2)+\ell_3\sin(q_2+q_3)+\ell_4\sin(q_2+q_3+q_4) \quad (3)$$

Where q and ℓ are the angle joints and lengths of a robotic arm links, respectively. This paper aims to build the network from these equations by using cosine and sine as the transfer function. However, the nature of the neural network is summation of the inputs but (1),(2) are the product of sine and cosine forms. Thus, we have to transform them into a summation form by trigonometric formula-product identities as follows.

$$\cos(A)\cos(B) = 0.5(\cos(A+B)+\cos(A-B)) \quad (4)$$

$$\sin(A)\cos(B) = 0.5(\sin(A+B)+\sin(A-B)) \quad (5)$$

Substitutes (4), (5) in (1), (2) obtain x_{end} and y_{end} equations in sum of cosine and sine form as follows.

$$x_{end} = \begin{bmatrix} 0.5\ell_2\cos(q_1+q_2)+0.5\ell_2\cos(q_1-q_2) \\ +0.5\ell_3\cos(q_1+q_2+q_3)+0.5\ell_3\cos(q_1-q_2-q_3) \\ +0.5\ell_4\cos(q_1+q_2+q_3+q_4)+0.5\ell_4\cos(q_1-q_2-q_3-q_4) \end{bmatrix} \quad (6)$$

$$y_{end} = \begin{bmatrix} 0.5\ell_2\sin(q_1+q_2)+0.5\ell_2\sin(q_1-q_2) \\ +0.5\ell_3\sin(q_1+q_2+q_3)+0.5\ell_3\sin(q_1-q_2-q_3) \\ +0.5\ell_4\sin(q_1+q_2+q_3+q_4)+0.5\ell_4\sin(q_1-q_2-q_3-q_4) \end{bmatrix} \quad (7)$$

Now (6), (7) are more suitable to build the MLP of neural network than (1), (2).

B. The Neural Network Model

This paper proposes the technique to build the custom neural networks from forward kinematic equations that has angle variables (q_1, q_2, q_3, q_4) as the inputs shown in fig.1 while the outputs are Euclidean distance error between the end-effector (x_{end},y_{end},z_{end}) and desired position of the end-effector (x_d,y_d,z_d) as shown in (8).

$$\text{Distance Error} = \sqrt{(x_{end}-x_d)^2+(y_{end}-y_d)^2+(z_{end}-z_d)^2} \quad (8)$$

Equation (8) shows the proposed method needs to build the network from the combination of squared error in x, y and z positions from (6), (7) and (3) as shown in fig. 2.

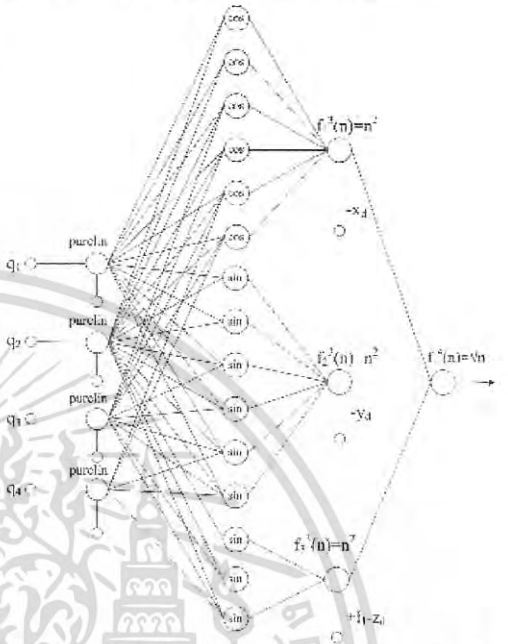


Fig. 2. the custom neural network from forward kinematics equations of a 4–DOF robotic arm.

The outputs of this network are shown as follows

$$a^4 = \sqrt{w_{1,1}^4\left(\sum_{i=1}^{6} w_{1,i}^3 a_i^2+b_1^3\right)^2+w_{1,2}^4\left(\sum_{i=1}^{6} w_{2,i}^3 a_{i+6}^2+b_2^3\right)^2+w_{1,3}^4\left(\sum_{i=1}^{3} w_{3,i}^3 a_{i+12}^2+b_3^3\right)^2} \quad (9)$$

If we let $w_{1,1}^4$ to $w_{1,3}^4$ equal 1. Then, we can substitutes (8) by (9) term by term for example bias b_1^3, b_2^3, b_3^3 are minus desired positions ($-x_d$,$-y_d$,$-z_d$+l_1), sum of product of weight and output $w_1^3a^2$, $w_2^3a^2$, $w_3^3a^2$ are x_{end}, y_{end}, z_{end}, respectively. Thus, the initial values of weight and bias are set as follows

- Input layer: the initial input angles are random values in range [0,2π].
- 1st hidden layer: the initial weight and bias are random values in range, [0,1] and [0,1], respectively. the network will update the weight to the optimum value.

- 2nd hidden layer: the weights are fix to 1,-1
- 3rd hidden layer: the weights are fix to amplitude of cosine and sine function in (6), (7) and (3), respectively. The bias in x,y,z node are set to minus the desired position.
- Output layer: the weights are fix to 1 as explained above.

Substitute initial weight and bias in (9) then, the output of the network will become Euclidean distance error as shown in (8). The next section will discuss about how to update the network.

C. The Operation

The proposed model will update weight of the network until the output reaches the target and uses the final weights and bias values as the solutions of angles. For example, let consider the net input of the first node in 3rd hidden layer (or error of x node) shown in fig.2

$$n_1^3 = 0.5\ell_2\cos(a_1^1+a_2^1)+0.5\ell_2\cos(a_1^1-a_2^1)$$
$$+0.5\ell_3\cos(a_1^1+a_2^1+a_3^1)+0.5\ell_3\cos(a_1^1-a_2^1-a_3^1)$$
$$+0.5\ell_4\cos(a_1^1+a_2^1+a_3^1+a_4^1)+0.5\ell_4\cos(a_1^1-a_2^1-a_3^1-a_4^1)+(-x_d) \quad (11)$$

It can be noticed that (11) is derived from (6) by assigned the initial weight and bias as explained earlier. Then, the outputs of input layer (a^1) become parameter q in (6). So, when the network finishes updating, the solutions of angles (q_{out}) can be found from (12)

$$\begin{bmatrix} q_{1out} \\ q_{2out} \\ q_{3out} \\ q_{4out} \end{bmatrix} = \begin{bmatrix} a_1^1 \\ a_2^1 \\ a_3^1 \\ a_4^1 \end{bmatrix} = \begin{bmatrix} w_{1,1\,final}^1 q_1 + b_{1\,final}^1 \\ w_{2,1\,final}^1 q_2 + b_{2\,final}^1 \\ w_{3,1\,final}^1 q_3 + b_{3\,final}^1 \\ w_{4,1\,final}^1 q_4 + b_{4\,final}^1 \end{bmatrix} \quad (12)$$

Where q, w_{final}, b_{final} are random initial angle, final weight and final bias value. Furthermore, the previous version of neural network proposed in [2] has some limitation that cannot generally start from zero angles. However, this model can solve the problem by putting bias term as shown in the experiments later.

III. Results

In this section, we conducted experiments to show the performance of our model under various initial input angles and desired positions and also showed trajectory tracking without angle constraints. In the simulation, the initial parameters of BPVLR [10] and a robotic arm are set as follows. A robotic arm link (cm): l_1=5, l_2=13, l_3=12, l_4=10, Learning rate = 0.0012, Increment = 1.02, Decrement = 0.8, Momentum = 0.2, target = 0 cm. the execute times and the number of iterations are measured with MATLAB codes on an Intel Core i5-2450M @ 2.5 GHz.

First, the reaching movement of the end-effector by our model and the FABRIK model for a hinge joint robotic arm is compared. Fig. 3 (a.) shows the FABRIK method needed to place a robotic arm on the same plane of desired position before solving 3D problems as show in progress from (1) to (4) but the proposed method can reach the desired position in 3-dimensional suddenly as shown in fig. 3 (b).

Second, in iterative method, the solution may not exist or slow-convergence for some initial values. However, this problem does not occur in our proposed model. Table 1 shows average execute time and number of iterations of 1000 random desired positions with zero initial and 1000 random initial values.

Finally, we let the model track the sine z=5+5sin (0.3πy) in range y=[-20/3,20/3] as shown in fig.4 and tracking 3D spiral in fig.5 with varies number of points on trajectories. The result has shown in table 2.

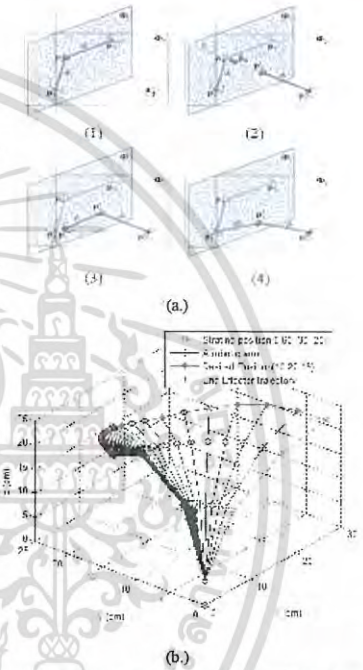


Fig. 3. The end-effector trajectory from a starting position to the desired position in 3dimensions (a.) The FABRIK model (b.) The proposed model.

TABLE I. AVERAGE RESULT FOR RANDOM INITIAL VALUES

Tol. (mm)	1000 random initial values		1000 random target positions	
	Time(s)	*No. of iterations*	*Time(s)*	*No. of iterations*
0.01	0.001976	35.735	0.002728	49.529
0.0001	0.002589	46.852	0.003261	59.496
0.00001	0.002945	52.293	0.003476	63.405

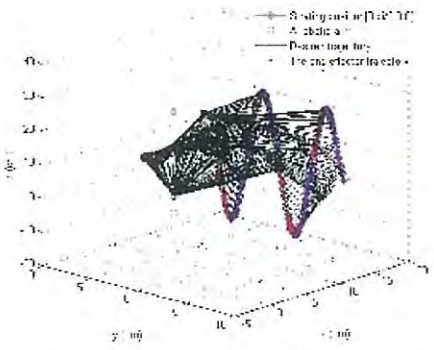


Fig. 4. A 4 DOF robotics arm tracking sin trajectory on yz-plane

Fig. 5. A 4 DOF robotics arm tracking 3D Spiral

TABLE II. RESULTS FOR TRAJECTORY TRACKING

Trajectory Type	No. of points on trajectory	Tolerance = 0.01 mm	
		No. of iterations	*Execute time(s)*
sine wave	40	1132	0.063938
	80	2064	0.114911
	120	2963	0.165440
3D Spiral	40	3466	0.213162
	80	6107	0.354575
	120	8488	0.490226

IV. CONCLUSIONS

This paper can solve inverse kinematics of a 4-DOF robotic arm in 3-dimensional spaces using the custom neural network creating from the forward kinematic equations with fast and accuracy. Moreover, the network structure uses less number of hidden layers and neuron node than [5,6,7,8,9] and the search spaces of this network are not depend on training data. Although the proposed method is not the new faster solution than the existing method but the speed of convergence can be increased by improved learning algorithm. Furthermore, the main different of our model and the other iterative methods such as FABRIK is reaching movement of the end-effector, the proposed method can reach the target in 3-dimensional like human hand to grab an object while the other can reach it in plane. In addition, due to 3D reaching ability, the proposed method seems to control the end-effector trajectory that leads to useful applications such as obstacle avoiding. In the future, we plan to apply for the higher DOF robotic arm and put this algorithm to use in real world applications.

REFERENCES

[1] S. Tejomurtula, S. Kak, "Inverse kinematics in robotics using neural networks," Information Sciences, vol.116, pp.147-164, 1999.

[2] P. Srisuk, A. Sento, and Y. Kitjaidure "Inverse Kinematics Solution using Neural Networks from Forward Kinematics Equations," International Conference on Knowledge and Smart Technology, pp.61-65, 2017., in press.

[3] S. Kucuk, Z. Bingul, Industrial Robotics:Theory, Modelling and control , pIV pro literatur Verlag Robert Mayer-Scholz, pp.117-148, 2007.

[4] H. A. F. Mohamed, S. Yahya, M. Moghavvemi, and S. S. Yang, "A New Inverse Kinematics Method for Three Dimensional Redundant Manipulators," ICROS-SICE International Joint Conference, pp.1557-1562, 2009.

[5] B. Daya, S. Khawandi, and M. Akoum, "Applying Neural Network Architecture for Inverse Kinematics Problem in Robotics,"J.Software Engineer & Applications, vol. 3, pp.230-239, 2010

[6] S. S. Chiddarwar, N. R. Babu, "Comparison of RBF and MLP neural networks to solve inverse kinematic problem for 6R serial robot by a fusion approach," Engineering Applications of Artificial Intelligence, vol.23, pp.1083-1092, 2010.

[7] Y. Feng, W.Yao-nan, Y. Yi-min, "Inverse Kinematics Solution for Robot Manipulator based on Neural Network under Joint Subspace," International Journal of Computers, Communications & Control, vol.7, pp.459-472, 2012.

[8] R. Köker, "A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization," Information Sciences, vol.22, pp.528-543, 2013.

[9] A. V. Duka,"Neural network based inverse kinematics solution for trajectory tracking of a robotic arm," Procedia Technology, vol. 12, pp. 20 – 27, 2014.

[10] M. T. Hagan, H. B. Demuth, M. H. Beale, and Orlando De Jesús, Neural Network Design, 2nd ed., pp. Martin Hagan, 2014.

[11] L.-C. T. Wang and C. C. Chen, "A combined optimization method for solving the inverse kinematics problems of mechanical manipulators," IEEE Transactions on Robotics and Automation, vol. 7, pp. 489–499, August 1991.

[12] A. Aristidou, J. Lasenby, "FABRIK: A fast, iterative solver for the Inverse Kinematics problem" Graphical Models, vol. 73, pp.243-260, 2011.

ประวัติผู้เขียน

ชื่อ-นามสกุล นายปัณณวิทย์ ศรีสุข

วัน เดือน ปีเกิด 23 กันยายน 2536 ที่กรุงเทพ

ที่อยู่ 89/183 หมู่บ้านฟ้าชมพฤกษ์ เฟส 1 ต.ลำลูกกา อ.ลำลูกกา จ.ปทุมธานี 12150

ประวัติการศึกษา 2558 วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง (เกียรตินิยมอันดับ 2)

ความชำนาญเฉพาะด้าน 1.) การสร้างโครงข่ายประสาทเทียมเลียนแบบสมการ
2.) หลักจลนศาสตร์หุ่นยนต์