Part 1: Motor Parameter Identification

การทดลองที่ 1: Motor Parameter Identification Using Experimental Data

จุดประสงค์

1. ศึกษาพฤติกรรมพลวัตของระบบหุ่นยนต์ 1-DOF ภายใต้สัญญาณ Step, Ramp และ Chirp Input
2. ประมาณค่าพารามิเตอร์ของระบบ ได้แก่ Equivalent Inertia (J), Damping Coefficient (B), Motor Efficiency (η) และ Back-EMF Constant (Ke)
3. เปรียบเทียบผลของการติดตั้ง Gripper ต่อพารามิเตอร์ทางพลวัตของระบบ
4. สร้างแบบจำลองทางคณิตศาสตร์ของระบบทั้งกรณีมีและไม่มี Gripper

สมมติฐาน

1. การติดตั้ง Gripper ส่งผลต่อค่า Equivalent Inertia และการตอบสนองของระบบ
2. ค่าพารามิเตอร์ของระบบจะแตกต่างกันระหว่างกรณีมีและไม่มี Gripper
3. แบบจำลองจากการประมาณค่าพารามิเตอร์สามารถอธิบายพฤติกรรมของระบบจริงได้ใกล้เคียงข้อมูลการทดลอง

ตัวแปร

1. ตัวแปรต้น :

   รูปแบบสัญญาณอินพุตที่ใช้กระตุ้นระบบ

   - Step Input
   - Ramp Input
   - Chirp Input

   สถานะของระบบ

   - มี Gripper
   - ไม่มี Gripper

2. ตัวแปรตาม :

- ตำแหน่งเชิงมุมของข้อต่อ (Angular Position)
- ความเร็วเชิงมุมของข้อต่อ (Angular Velocity)
- ค่าพารามิเตอร์ที่ประมาณได้
    - Motor Damping Coefficient
    - Equivalent Inertia
    - Motor Efficiency
    - Back-EMF Constant

3. ตัวแปรควบคุม :

- หุ่นยนต์ 1-DOF ชุดเดิมตลอดการทดลอง
- อัตราทดเกียร์คงที่
- มวลและรูปทรงของแขนกลคงที่
- แรงดันไฟฟ้าที่ใช้กับระบบ
- Sampling Time
- ระยะเวลาเก็บข้อมูล 15 วินาทีต่อการทดลอง
- จำนวนการทดลองซ้ำ 3 ครั้งต่อรูปแบบสัญญาณ
- Encoder และระบบวัดผลชุดเดิม

เอกสารและงานวิจัยที่เกี่ยวข้อง

1. แบบจำลองทางคณิตศาสตร์ของมอเตอร์กระแสตรง (DC Motor Mathematical Model)

มอเตอร์กระแสตรงเป็นระบบพลวัตที่แปลงแรงดันไฟฟ้าเป็นแรงบิดเพื่อขับเคลื่อนเพลา โดยการตอบสนองของระบบขึ้นอยู่กับพารามิเตอร์สำคัญ ได้แก่ ความเฉื่อย (J), สัมประสิทธิ์แรงเสียดทานเชิงความหนืด (B), ค่าคงที่แรงเคลื่อนไฟฟ้าย้อนกลับ (Ke) และประสิทธิภาพของระบบ (η) ซึ่งส่งผลต่อความเร็ว การเร่ง และความสามารถในการควบคุมมอเตอร์โดยตรง

2. Equivalent Inertia และผลของชุดเฟืองต่อระบบ

ในการใช้งานจริง มอเตอร์จะทำงานร่วมกับชุดเฟืองและโหลดภายนอก ทำให้เกิดความเฉื่อยสมมูล (Equivalent Inertia) ของทั้งระบบ ไม่ใช่เฉพาะโรเตอร์ โดยอัตราทดเกียร์มีผลต่อการสะท้อนโมเมนต์ความเฉื่อยของโหลดกลับสู่เพลามอเตอร์ ซึ่งส่งผลต่อพฤติกรรมการตอบสนองของระบบโดยตรง.

3. System Identification

System Identification คือกระบวนการสร้างแบบจำลองทางคณิตศาสตร์จากข้อมูลอินพุตและเอาต์พุตที่วัดได้จริง เพื่อประมาณค่าพารามิเตอร์ให้ใกล้เคียงระบบจริงมากที่สุด โดยนิยมใช้ในงานหุ่นยนต์และเมคคาทรอนิกส์ เนื่องจากสามารถสร้างแบบจำลองจากข้อมูลการทดลองได้โดยไม่ต้องทราบรายละเอียดภายในของระบบทั้งหมด

4. Excitation Signal for System Identification

สัญญาณ Step, Ramp และ Chirp ใช้ศึกษาพฤติกรรมระบบในลักษณะที่แตกต่างกัน โดย Chirp Input สามารถกระตุ้นระบบได้หลายช่วงความถี่ จึงเหมาะสำหรับงาน System Identification และให้ข้อมูลของระบบได้ครบถ้วนกว่า ข้อมูลที่ได้ถูกนำมาใช้ประมาณค่าพารามิเตอร์สำคัญ ได้แก่ Equivalent Inertia, Damping Coefficient, Back-EMF Constant และ Motor Efficiency เพื่อนำไปสร้างแบบจำลอง ออกแบบตัวควบคุม และวิเคราะห์ประสิทธิภาพของระบบต่อไป

6. Model Validation

หลังจากได้ค่าพารามิเตอร์แล้ว จำเป็นต้องตรวจสอบความถูกต้องของแบบจำลองด้วยข้อมูลจากการทดลองจริง เพื่อประเมินว่าแบบจำลองสามารถอธิบายพฤติกรรมของระบบได้ดีเพียงใด การเปรียบเทียบผลการจำลองกับข้อมูลจริงเป็นขั้นตอนสำคัญก่อนนำแบบจำลองไปใช้ในการออกแบบระบบควบคุม

ขั้นตอนการดำเนินงาน

1. ติดตั้งหุ่นยนต์ 1-DOF พร้อมมอเตอร์ ชุดเฟือง แขนกล และ Encoder ให้พร้อมสำหรับการทดลอง
2. ตรวจสอบการทำงานของระบบและกำหนดสัญญาณกระตุ้นที่ใช้ในการทดลอง ได้แก่ Step Input, Ramp Input และ Chirp Input
3. ป้อนสัญญาณ Step Input ให้กับระบบ และบันทึกข้อมูลการตอบสนองเป็นเวลา 15 วินาที

4. ทำการทดลองซ้ำจำนวน 3 ครั้ง และบันทึกข้อมูลทั้งหมด
5. ดำเนินการทดลองซ้ำตามข้อ 3–4 สำหรับ Ramp Input และ Chirp Input
6. รวบรวมข้อมูลอินพุตและเอาต์พุตจากการทดลองทั้งหมด และนำเข้าสู่กระบวนการ System Identification
7. ประมาณค่าพารามิเตอร์ของระบบ ได้แก่ Equivalent Inertia (J), Damping Coefficient (B), Motor Efficiency (η) และ Back-EMF Constant (Ke)
8. เปรียบเทียบค่าพารามิเตอร์ที่ได้จากแต่ละสัญญาณกระตุ้น เพื่อนำไปใช้ในการตรวจสอบความถูกต้องของแบบจำลองในขั้นตอนถัดไป

ผลการทดลอง

1. พารามิเตอร์จาก Step Input หลัง Estimate ระหว่างมี Gripper และ ไม่มี Gripper

ตารางที่ 1 แสดงค่าพารามิเตอร์จาก Step Input หลัง Estimate (Without Gripper)

พารามิเตอร์	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	เฉลี่ย
Motor B	0.0418	0.3200	0.0442	0.1355
Motor Efficiency (η)	0.3676	0.3020	0.2996	0.3230
Equivalent Inertia (J)	0.00640	0.00561	0.00709	0.00637
Back-EMF Constant (Ke)	3.8324	3.3058	3.7971	3.6451

ตารางที่ 2 แสดงค่าพารามิเตอร์จาก Step Input หลัง Estimate (With Gripper)

พารามิเตอร์	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	เฉลี่ย
Motor B	6.26E-01	4.43E-02	4.43E-02	0.2380
Motor Efficiency (η)	0.32812	0.29936	0.29935	0.3089
Equivalent Inertia (J)	1.91E-01	7.09E-03	7.09E-03	0.0685
Back-EMF Constant (Ke)	2.6247	3.8599	3.8651	3.4499

2. พารามิเตอร์จาก Ramp Input หลัง Estimate ระหว่างมี Gripper และ ไม่มี Gripper

ตารางที่ 3 แสดงค่าพารามิเตอร์จาก Ramp Input หลัง Estimate (Without Gripper)

พารามิเตอร์	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	เฉลี่ย
Motor B	0.2000	0.2000	0.2000	0.1997
Motor Efficiency (η)	0.3008	0.3014	0.3017	0.3013
Equivalent Inertia (J)	0.1070	0.1060	0.1060	0.1062
Back-EMF Constant (Ke)	3.4430	3.4438	3.4442	3.4437

ตารางที่ 4 แสดงค่าพารามิเตอร์จาก Ramp Input หลัง Estimate (With Gripper)

พารามิเตอร์	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	เฉลี่ย
Motor B	6.29E-01	6.28E-01	2.78E-01	0.5117
Motor Efficiency (η)	0.32281	0.32611	0.29983	0.3163
Equivalent Inertia (J)	3.76E-02	1.67E-01	6.15E-03	0.0703
Back-EMF Constant (Ke)	2.629	2.6486	3.4002	2.8926

3. พารามิเตอร์จาก Chirp Input หลัง Estimate ระหว่างมี Gripper และ ไม่มี Gripper

ตารางที่ 5 แสดงค่าพารามิเตอร์จาก Chirp Input หลัง Estimate (Without Gripper)

พารามิเตอร์	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	เฉลี่ย
Motor B	0.00879	0.00809	0.00773	0.00820
Motor Efficiency (η)	0.3597	0.6978	0.6979	0.5851
Equivalent Inertia (J)	0.000000372	0.000000872	0.000000828	0.000000691
Back-EMF Constant (Ke)	4.6178	4.6238	4.6235	4.6217

ตารางที่ 6 แสดงค่าพารามิเตอร์จาก Chirp Input หลัง Estimate (With Gripper)

พารามิเตอร์	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	เฉลี่ย
Motor B	4.80E-01	4.40E-03	4.63E-01	0.3159
Motor Efficiency (η)	0.39886	0.69953	0.52835	0.5422
Equivalent Inertia (J)	1.43E-01	1.07E-06	2.30E-01	0.1241
Back-EMF Constant (Ke)	2.5438	4.0383	2.5486	3.0436

สรุปผลการทดลอง

จากการทดลองประมาณค่าพารามิเตอร์ของระบบหุ่นยนต์ 1-DOF โดยใช้สัญญาณกระตุ้น Step Input, Ramp Input และ Chirp Input ทั้งในกรณีที่ติดตั้ง Gripper และไม่ติดตั้ง Gripper สามารถประมาณค่าพารามิเตอร์ของระบบได้สำเร็จ ได้แก่ Equivalent Inertia (J), Damping Coefficient (B), Motor Efficiency (η) และ Back-EMF Constant (Ke)

ผลการทดลองแสดงให้เห็นว่าค่าพารามิเตอร์ที่ได้มีความแตกต่างกันตามลักษณะของสัญญาณกระตุ้นและสภาวะการติดตั้ง Gripper เนื่องจากแต่ละกรณีสามารถกระตุ้นพลวัตของระบบได้แตกต่างกัน อีกทั้งการติดตั้ง Gripper ยังส่งผลให้โหลดและโมเมนต์ความเฉื่อยสมมูลของระบบเปลี่ยนแปลงไป ทำให้ค่าพารามิเตอร์ที่ประมาณได้แตกต่างจากกรณีที่ไม่มี Gripper

เพื่อเพิ่มความน่าเชื่อถือของผลการทดลอง ได้ทำการทดลองซ้ำจำนวน 3 ครั้งในแต่ละกรณี และใช้ค่าเฉลี่ยของผลการประมาณค่าเป็นตัวแทนของพารามิเตอร์ในแต่ละสภาวะ ก่อนนำไปสร้างแบบจำลองทางคณิตศาสตร์และตรวจสอบความถูกต้องด้วยวิธี Cross Validation ในการทดลองถัดไป

อภิปรายผล

จากผลการประมาณค่าพารามิเตอร์พบว่าค่าที่ได้จากแต่ละสัญญาณกระตุ้นมีความแตกต่างกัน เนื่องจากสัญญาณ Step, Ramp และ Chirp สามารถกระตุ้นพลวัตของระบบในลักษณะที่แตกต่างกัน ทำให้ข้อมูลที่ใช้ในการประมาณค่ามีคุณสมบัติไม่เหมือนกัน ส่งผลต่อความแม่นยำของค่าพารามิเตอร์ที่ได้

นอกจากนี้ การติดตั้ง Gripper ส่งผลให้มวลและโมเมนต์ความเฉื่อยของระบบเพิ่มขึ้น ส่งผลให้พฤติกรรมการตอบสนองของระบบเปลี่ยนแปลงไปเมื่อเปรียบเทียบกับกรณีที่ไม่มี Gripper ดังนั้นค่าพารามิเตอร์ที่ประมาณได้จึงสะท้อนคุณลักษณะของระบบทั้งหมด ซึ่งประกอบด้วยมอเตอร์ ชุดเฟือง แขนกล และ Gripper ไม่ใช่เฉพาะตัวมอเตอร์เพียงอย่างเดียว

การทำซ้ำการทดลองจำนวน 3 ครั้งในแต่ละกรณีช่วยให้สามารถประเมินความสม่ำเสมอของผลการประมาณค่าได้ดียิ่งขึ้น โดยพิจารณาจากค่าเฉลี่ยและแนวโน้มการกระจายตัวของข้อมูล ทำให้สามารถลดผลกระทบจากสัญญาณรบกวนและความคลาดเคลื่อนที่อาจเกิดขึ้นระหว่างการเก็บข้อมูล

ผลลัพธ์ที่ได้จากการทดลองนี้จึงเป็นพื้นฐานสำคัญสำหรับการสร้างแบบจำลองทางคณิตศาสตร์ของระบบ และใช้เป็นข้อมูลสำหรับการตรวจสอบความถูกต้องของแบบจำลองด้วยวิธี Cross Validation เพื่อคัดเลือกชุดพารามิเตอร์ที่เหมาะสมที่สุดสำหรับการออกแบบระบบควบคุมในส่วนถัดไป

ข้อเสนอแนะ

1. ควรเพิ่มจำนวนรอบการทดลองมากกว่า 3 ครั้ง เพื่อเพิ่มความน่าเชื่อถือของค่าพารามิเตอร์ที่ประมาณได้
2. ควรตรวจสอบและลดสัญญาณรบกวนจาก Encoder และระบบวัดผลก่อนทำการประมาณค่า
3. ควรทดลองใช้สัญญาณกระตุ้นรูปแบบอื่นเพิ่มเติม เพื่อเปรียบเทียบประสิทธิภาพในการประมาณค่าพารามิเตอร์

อ้างอิง

1. https://www.hindawi.com/journals/mse/2012/189757/
2. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC12788085/
3. https://ieeexplore.ieee.org/document/957638/
4. https://www.mathworks.com/help/slcontrol/ug/creating-input-signals-for-estimation.html

การทดลองที่ 2: การตรวจสอบความถูกต้องและประเมินพารามิเตอร์ด้วยวิธี Cross Validation

จุดประสงค์

1. เพื่อตรวจสอบความถูกต้องของแบบจำลองที่สร้างขึ้นจากค่าพารามิเตอร์ที่ได้จากการประมาณค่า
2. เพื่อเปรียบเทียบประสิทธิภาพของแบบจำลองที่ได้จากสัญญาณ Step, Ramp และ Chirp
3. เพื่อศึกษาผลกระทบของ Gripper ต่อความแม่นยำของแบบจำลอง
4. เพื่อเลือกชุดพารามิเตอร์ที่เหมาะสมที่สุดสำหรับใช้ในการออกแบบระบบควบคุม

สมมติฐาน

1. แบบจำลองที่ได้จากสัญญาณกระตุ้นแต่ละชนิดจะให้ผลการทำนายแตกต่างกัน
2. การติดตั้ง Gripper จะส่งผลต่อค่าความคลาดเคลื่อนของแบบจำลอง
3. แบบจำลองที่มีค่า RMSE และ NRMSE ต่ำ และมีค่า R^2 สูง จะมีความเหมาะสมสำหรับใช้งานจริงมากกว่า

ตัวแปร

1. ตัวแปรต้น :

- ชุดค่าพารามิเตอร์ที่ได้จากการประมาณค่าด้วย
    - Step Input
    - Ramp Input
    - Chirp Input

สถานะของระบบ

- มี Gripper
- ไม่มี Gripper
- ชุดข้อมูลที่ใช้สำหรับ Cross Validation

2. ตัวแปรตาม :

- Root Mean Square Error
- Normalized Root Mean Square Error
- Coefficient of Determination
- ความคลาดเคลื่อนระหว่างผลการจำลองกับข้อมูลจริง

3. ตัวแปรควบคุม :

- หุ่นยนต์ 1-DOF ชุดเดิม
- แบบจำลองทางคณิตศาสตร์รูปแบบเดียวกัน
- วิธีการประมาณค่าพารามิเตอร์แบบเดียวกัน
- อัตราทดเกียร์คงที่
- แขนกลและโหลดคงที่
- Sampling Time คงที่
- ระยะเวลาของข้อมูลที่ใช้ในการประเมินผลเท่ากัน

เอกสารและงานวิจัยที่เกี่ยวข้อง

1. Cross Validation

Cross Validation เป็นกระบวนการตรวจสอบความสามารถของแบบจำลองในการทำนายข้อมูลที่ไม่เคยถูกใช้ในการสร้างแบบจำลองมาก่อน โดยมีวัตถุประสงค์เพื่อประเมินความสามารถในการนำแบบจำลองไปใช้งานจริงและลดปัญหา Overfitting

2. Root Mean Square Error

RMSE เป็นตัวชี้วัดความแตกต่างระหว่างค่าที่แบบจำลองทำนายและค่าที่วัดได้จริง โดยคำนวณจากค่าเฉลี่ยกำลังสองของความคลาดเคลื่อน RMSE ที่มีค่าน้อยแสดงว่าแบบจำลองสามารถทำนายพฤติกรรมของระบบได้ใกล้เคียงกับข้อมูลจริง

3. Normalized Root Mean Square Error

NRMSE เป็นการปรับค่า RMSE ให้อยู่ในรูปเปอร์เซ็นต์เมื่อเทียบกับช่วงของข้อมูล ทำให้สามารถเปรียบเทียบประสิทธิภาพของแบบจำลองระหว่างชุดข้อมูลที่มีขนาดแตกต่างกันได้

4. Coefficient of Determination

R^2 เป็นดัชนีที่ใช้วัดความสามารถของแบบจำลองในการอธิบายความแปรปรวนของข้อมูลจริง

ค่า R^2 เข้าใกล้ 1 หมายถึงแบบจำลองสามารถอธิบายพฤติกรรมของระบบได้ดี ในขณะที่ค่าใกล้ 0 หมายถึงแบบจำลองมีความสามารถในการอธิบายข้อมูลต่ำ

5. Model Validation for System Identification

ภายหลังจากการประมาณค่าพารามิเตอร์ จำเป็นต้องทำการตรวจสอบความถูกต้องของแบบจำลองด้วยข้อมูลชุดอื่นที่ไม่ได้ใช้ในการประมาณค่า เพื่อประเมินว่าพารามิเตอร์ที่ได้สามารถนำไปใช้อธิบายพฤติกรรมของระบบจริงได้อย่างน่าเชื่อถือ

ขั้นตอนการดำเนินงาน

1. นำค่าพารามิเตอร์ที่ได้จากการประมาณค่าในแต่ละกรณี ได้แก่ Step Input, Ramp Input และ Chirp Input มาสร้างแบบจำลองทางคณิตศาสตร์ของระบบ
2. จำลองการตอบสนองของระบบโดยใช้ค่าพารามิเตอร์ที่ได้จากแต่ละสัญญาณกระตุ้น
3. ใช้ข้อมูลจากสัญญาณ Step Input เป็นชุดข้อมูลทดสอบ (Validation Data) และเปรียบเทียบกับผลการจำลองจากพารามิเตอร์ที่ได้จาก Ramp Input และ Chirp Input
4. ใช้ข้อมูลจากสัญญาณ Ramp Input เป็นชุดข้อมูลทดสอบ และเปรียบเทียบกับผลการจำลองจากพารามิเตอร์ที่ได้จาก Step Input และ Chirp Input
5. ใช้ข้อมูลจากสัญญาณ Chirp Input เป็นชุดข้อมูลทดสอบ และเปรียบเทียบกับผลการจำลองจากพารามิเตอร์ที่ได้จาก Step Input และ Ramp Input
6. คำนวณค่าดัชนีทางสถิติสำหรับประเมินความแม่นยำของแบบจำลอง ได้แก่
    - Root Mean Square Error
    - Normalized Root Mean Square Error

- Coefficient of Determination

7. เปรียบเทียบผลการประเมินของแบบจำลองจากพารามิเตอร์แต่ละชุด
8. วิเคราะห์ความสามารถในการทำนายพฤติกรรมของระบบและความสามารถในการนำไปใช้งานจริงของแต่ละแบบจำลอง
9. เลือกชุดพารามิเตอร์ที่ให้ค่าความคลาดเคลื่อนต่ำที่สุดและมีค่า R^2 สูงที่สุด เพื่อใช้ในการออกแบบระบบควบคุมในส่วนถัดไป

ผลการทดลอง

1. พารามิเตอร์หลัง Cross Validation ด้วยสัญญาณ Step ระหว่างมี Gripper และ ไม่มี Gripper

ตารางที่ 7 แสดงค่า Cross Validation ด้วยสัญญาณ Step (Without Gripper)

พารามิเตอร์	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	เฉลี่ย
RMSE	1.4148	1.4054	1.4489	1.4230
NRMSE (%)	18.41	18.29	18.86	18.52
R^2	0.7269	0.7305	0.7135	0.7236

ตารางที่ 8 แสดงค่า Cross Validation ด้วยสัญญาณ Step (With Gripper)

พารามิเตอร์	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	เฉลี่ย
RMSE	1.07208	0.951098	0.966203	0.9965
NRMSE (%)	16.33	14.49	14.72	15.18
R^2	0.7992	0.842	0.8369	0.8260

2. พารามิเตอร์หลัง Cross Validation ด้วยสัญญาณ Ramp ระหว่างมี Gripper และ ไม่มี Gripper

ตารางที่ 9 แสดงค่า Cross Validation ด้วยสัญญาณ Ramp (Without Gripper)

พารามิเตอร์	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	เฉลี่ย
RMSE	0.5059	0.6357	0.6308	0.5908
NRMSE (%)	7.71	9.68	9.61	9.00

R^2	0.9397	0.9049	0.9063	0.9170

ตารางที่ 10 แสดงค่า Cross Validation ด้วยสัญญาณ Ramp (With Gripper)

พารามิเตอร์	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	เฉลี่ย
RMSE	0.53181	0.542448	0.455221	0.5098
NRMSE (%)	8.10	8.26	6.94	7.77
R^2	0.9189	0.9156	0.9406	0.9250

3. พารามิเตอร์หลัง Cross Validation ด้วยสัญญาณ Ramp ระหว่างมี Gripper และ ไม่มี Gripper

ตารางที่ 11 แสดงค่า Cross Validation ด้วยสัญญาณ Chirp (Without Gripper)

พารามิเตอร์	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	เฉลี่ย
RMSE	2.1222	2.2080	2.2927	2.2076
NRMSE (%)	28.58	29.73	30.87	29.73
R^2	0.8358	0.8222	0.8083	0.8221

ตารางที่ 12 แสดงค่า Cross Validation ด้วยสัญญาณ Chirp (With Gripper)

พารามิเตอร์	ครั้งที่ 1	ครั้งที่ 2	ครั้งที่ 3	เฉลี่ย
RMSE	4.165673	1.504868	4.714776	3.4618
NRMSE (%)	63.47	22.93	71.84	52.75
R^2	0.1937	0.8948	-0.0329	0.3519

สรุปผลการทดลอง

จากการตรวจสอบความถูกต้องของแบบจำลองด้วยวิธี Cross Validation โดยใช้ดัชนี RMSE, NRMSE และ R^2 พบว่าประสิทธิภาพของแบบจำลองมีความแตกต่างกันตามสัญญาณที่ใช้ในการประมาณค่าพารามิเตอร์และสภาวะการติดตั้ง Gripper

ผลการประเมินแสดงให้เห็นว่าแบบจำลองที่สร้างจากค่าพารามิเตอร์ที่ได้จาก Ramp Input ในกรณีที่ติดตั้ง Gripper ให้ผลการทำนายดีที่สุด โดยมีค่าความคลาดเคลื่อนต่ำและมีค่า R^2 สูงที่สุดเมื่อเปรียบเทียบกับกรณีอื่น แสดงให้เห็นว่าแบบจำลองสามารถอธิบายพฤติกรรมของระบบได้ใกล้เคียงกับข้อมูลจริงมากที่สุด

จากผลการประเมินเชิงสถิติและการตรวจสอบความถูกต้องของแบบจำลอง จึงเลือกใช้ชุดพารามิเตอร์ที่ได้จาก Ramp Input ในกรณีที่ติดตั้ง Gripper เป็นค่าพารามิเตอร์อ้างอิงสำหรับการสร้างแบบจำลองของระบบ และนำไปใช้ในการออกแบบระบบควบคุมใน Part ถัดไป

อภิปรายผล

ผลการทดลองแสดงให้เห็นว่าการตรวจสอบความถูกต้องของแบบจำลองด้วยวิธี Cross Validation มีความสำคัญต่อการคัดเลือกค่าพารามิเตอร์ที่เหมาะสม เนื่องจากค่าพารามิเตอร์ที่ประมาณได้จากแต่ละสัญญาณกระตุ้นไม่ได้ให้ประสิทธิภาพในการทำนายเท่ากัน แม้ว่าจะได้มาจากระบบเดียวกันก็ตาม

เมื่อพิจารณาจากค่า RMSE, NRMSE และ R^2 พบว่า Ramp Input ให้ผลการทำนายที่ดีที่สุดทั้งในด้านความแม่นยำและความเสถียรของแบบจำลอง เนื่องจากลักษณะของสัญญาณมีการเปลี่ยนแปลงอย่างต่อเนื่อง ทำให้สามารถสะท้อนพฤติกรรมของระบบได้ครอบคลุมช่วงการทำงานที่สำคัญ ส่งผลให้กระบวนการประมาณค่าพารามิเตอร์มีประสิทธิภาพมากขึ้น

นอกจากนี้ แบบจำลองที่สร้างจากข้อมูลในกรณีที่ติดตั้ง Gripper ยังให้ผลการทำนายดีกว่าหลายกรณีที่ไม่มี Gripper แสดงให้เห็นว่าการประมาณค่าพารามิเตอร์ภายใต้สภาวะที่ใกล้เคียงกับการใช้งานจริงช่วยให้แบบจำลองสามารถอธิบายพฤติกรรมของระบบได้แม่นยำยิ่งขึ้น

การใช้ตัวชี้วัดทางสถิติร่วมกับการทำ Cross Validation ช่วยลดความเสี่ยงในการเลือกค่าพารามิเตอร์ที่เหมาะสมเฉพาะกับข้อมูลชุดใดชุดหนึ่ง และช่วยให้มั่นใจได้ว่าพารามิเตอร์ที่เลือกสามารถนำไปใช้งานจริงได้อย่างมีประสิทธิภาพ ดังนั้นจึงเลือกใช้ชุดพารามิเตอร์จาก Ramp Input ในกรณีที่ติดตั้ง Gripper สำหรับการออกแบบ PID Controller, Motion Profile และ Kalman Filter ในการทดลองถัดไป

ข้อเสนอแนะ

1. ควรเพิ่มจำนวนข้อมูลที่ใช้ในการ Cross Validation เพื่อเพิ่มความน่าเชื่อถือของผลการประเมิน
2. ควรเปรียบเทียบกับวิธีการประมาณค่าพารามิเตอร์รูปแบบอื่น เพื่อยืนยันความถูกต้องของผลลัพธ์

อ้างอิง

1. https://arxiv.org/abs/1902.00683
2. https://www.mathworks.com/help/ident/

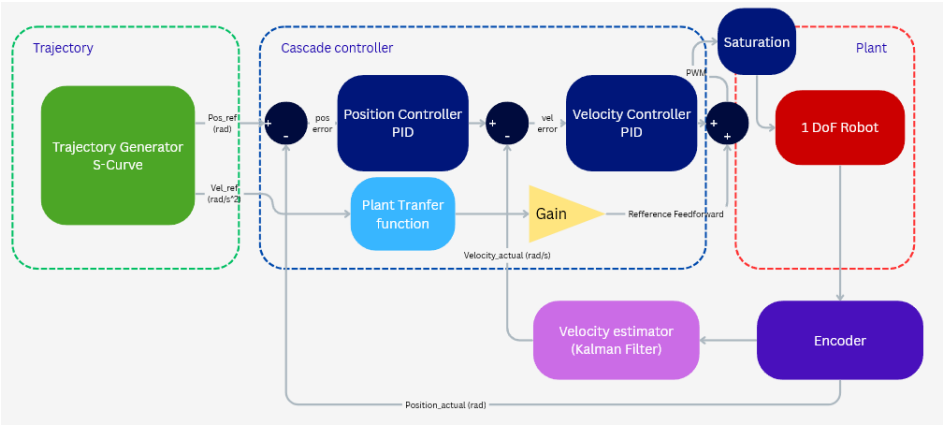
Part 2: PID Control with Feedforward and Motion Profile

จุดประสงค์

1. เพื่อออกแบบและ Implement ระบบ Cascade PID Control สำหรับแขนกล 1 องศาอิสระบนบอร์ด STM32 NUCLEO-G474RE
2. เพื่อเปรียบเทียบผลของการใส่และไม่ใส่ Reference Feedforward ในแง่ของ Tracking Error โดยใช้หลักการทางสถิติในการประเมินผล
3. เพื่อออกแบบและ Implement Disturbance Compensation โดยใช้ค่าประมาณ Disturbance Torque จาก Kalman Filter และเปรียบเทียบผลกับกรณีที่ไม่มีการชดเชย
4. เพื่อศึกษาผลกระทบของ Output Saturation ต่อ System Response และความปลอดภัยของระบบ
5. เพื่อวิเคราะห์ปัญหา Integration Windup และ Implement Anti-windup Clamping พร้อมแสดงผลลัพธ์ก่อนและหลังการแก้ไข
6. เพื่อศึกษาผลกระทบของการปรับค่า Kp, Ki, Kd ต่อพฤติกรรมของระบบและแสดงขั้นตอนการปรับพารามิเตอร์ภายใต้ Cascade Control
7. เพื่อเปรียบเทียบการเคลื่อนที่แบบ Trapezoidal Velocity Profile และ S-Curve Velocity Profile ในแง่ของ Tracking Error, Jerk และ Control Effort

เอกสารและงานวิจัยที่เกี่ยวข้อง

1. Cascade PID Control



รูปที่ 1 แสดง Cascade PID Control

2. Refference Feedforward

Reference Feedforward เป็นเทคนิคที่นำ Reference Signal จาก Trajectory Generator มาแปลงเป็น Control Signal โดยตรงโดยไม่ผ่าน Feedback Loop ทำให้ระบบสามารถชดเชย Velocity Lag ได้ล่วงหน้า

ในการทดลองนี้นำ $\dot{q_{ref}}(t)$ จาก S-Curve Trajectory มาแปลงเป็น Voltage เทียบเท่าผ่านแบบจำลองมอเตอร์ แล้วบวกโดยตรงเข้าที่ Output ของ Velocity PID

แปลง Reference Velocity จากองศาเป็นเรเดียนต่อวินาที

$$\omega_{ref(t)} = \dot{q}_{ref(t)} \cdot \frac{\pi}{180}$$

คำนวณ Feedforward Voltage จากแบบจำลองมอเตอร์

$$V_{FF}(t) = \frac{(Ke + B) \cdot \omega_{ref}(t)}{\eta}$$

แปลง Feedforward Voltage เป็น PWM Command

$$u_{FF}(t) = V_{FF}(t) \cdot \frac{u_{max} \cdot k_{scale}}{V_{\sup ply}}$$

รวม PWM จาก PID และ Feedforward พร้อม Saturation

โดยที่ $$u(t) = sat\big(u_{pid}(t) + u_{FF}(t), -u_{max}, u_{max}\big)$$

3. Disturbance Compensation

State ที่ 4 ของ Kalman Filter คือ $\widehat{\tau_d}$ ซึ่ง Model เป็น Random Walk

$$\tau_d(k+1) = \tau_d(k) + w_d(k)$$

นำค่า $\widehat{\tau_d}$ มาแปลงเป็น PWM Compensation

$$u_{comp} = \frac{\tau_d}{K_t \cdot \eta} \cdot \frac{u_{max}}{V_{\sup ply}}$$

PWM Command รวมทั้งหมด

$$u(t) = sat\big(u_{pid}(t) + u_{FF}(t) + u_{comp}(t), -u_{max}, u_{max}\big)$$

4. Integration Windup และ Anti-windup

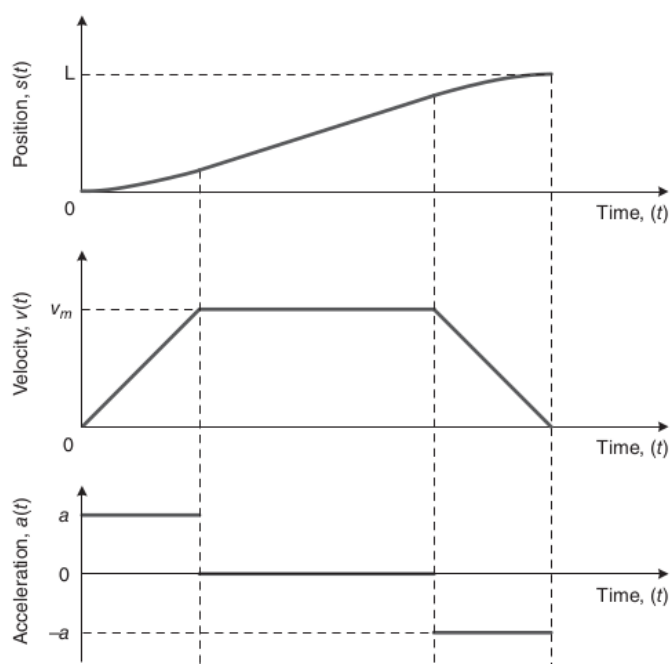
Integration Windup เกิดขึ้นเมื่อ Controller Output ถึง Saturation Limit แต่ Integral Term ยังคงสะสมต่อไป หากไม่มี Anti-windup ในช่วงที่ Output Saturate เป็นเวลา NN N Timestep แล้ว Integral จะสะสม ซึ่งทำให้เมื่อระบบกลับสู่ Operating Range เกิด Overshoot รุนแรงจาก Integral ที่มีค่ามากเกินไป วิธีแก้ — Anti-windup Clamping จำกัดค่า Integral ให้อยู่ในช่วง $[-I_{\lim it}, I_{\lim it}]$

5. Output Saturation

Output Saturation จำกัด PWM Command ให้อยู่ในช่วงที่ Hardware รองรับและปลอดภัยและเนื่องจากมอเตอร์มี Static Friction ทำให้ไม่ตอบสนองที่ PWM ต่ำมาก การเพิ่ม Deadband Compensation ชดเชยโดย Re-map PWM ออกจาก Dead Zone

6. S-Curve และ Trapezoid

6.1 Trapezoidal Velocity Profile แบ่งการเคลื่อนที่เป็น 3 Phase ได้แก่ Acceleration, Cruise และ Deceleration โดย Acceleration เปลี่ยนแบบทันทีทันใด ทำให้ Jerk มีค่าไม่จำกัด



รูปที่ 2 กราฟการเคลื่อนที่แบบ Trapezoid velocity profile

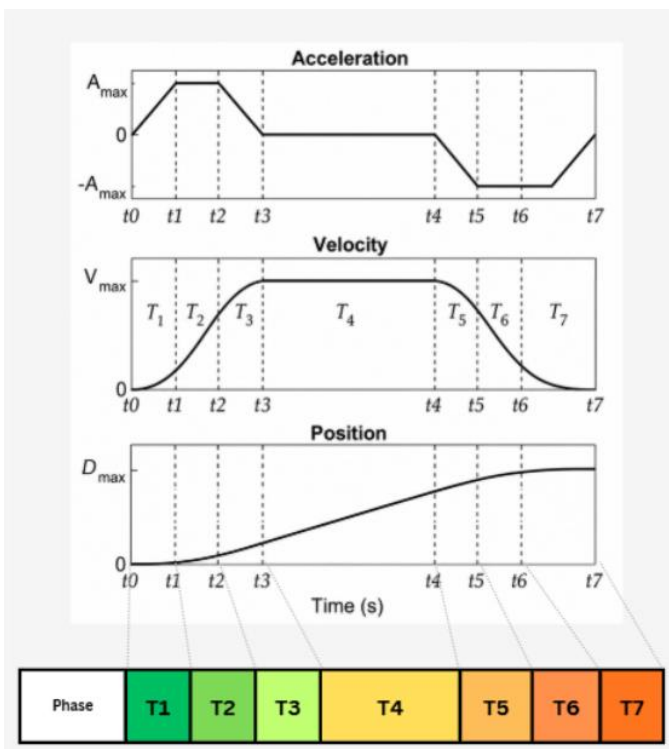
$$\dot{q}(t) = \begin{cases} a_{max} \cdot t & 0 \le t < t_1 \\ v_{max} & t_1 \le t < t_2 \\ v_{max} - a_{max}(t - t_2) & t_2 \le t \le t_f \end{cases}$$

$$\ddot{q}(t) = \begin{cases} a_{max} & 0 \le t < t_1 \\ 0 & t_1 \le t < t_2 \\ -a_{max} & t_2 \le t \le t_f \end{cases}$$

$\dddot{q}(t) \to \infty$ **ที่จุดเปลี่ยน Phase**

รูปที่ 3 สมการการเคลื่อนที่ในแต่ละช่วงของ Trapezoid velocity profile

6.2 S-Curve Velocity Profile แบ่งการเคลื่อนที่เป็น 7 Phase โดยจำกัด Jerk ไม่เกิน J_{max} ทำให้ Acceleration เปลี่ยนแบบ Linear และ Velocity เปลี่ยนแบบ Smooth



รูปที่ 4 กราฟการเคลื่อนที่แบบ S-Curve velocity profile

$+J_{max}$

Phase	T1	T7
$acc(\tau)$	$J \cdot \tau$	$-A_{max} + J \cdot \tau$
$vel(\tau)$	$v_0 + \frac{1}{2}J\tau^2$	$v_0 - A_{max}\tau + \frac{1}{2}J\tau^2$
$pos(\tau)$	$p_0 + v_0\tau + \frac{1}{6}J\tau^3$	$p_0 + v_0\tau - \frac{1}{2}A_{max}\tau^2 + \frac{1}{6}J\tau^3$

$J = 0$

Phase	T2	T6
$acc(\tau)$	A_{max}	$-A_{max}$
$vel(\tau)$	$v_0 + A_{max}\tau$	$v_0 - A_{max}\tau$
$pos(\tau)$	$p_0 + v_0\tau + \frac{1}{2}A_{max}\tau^2$	$p_0 + v_0\tau - \frac{1}{2}A_{max}\tau^2$

$-J_{max}$

Phase	T3	T5
$acc(\tau)$	$A_{max} - J\tau$	$-J \cdot \tau$
$vel(\tau)$	$v_0 + A_{max}\tau - \frac{1}{2}J\tau^2$	$v_0 - \frac{1}{2}J\tau^2$
$pos(\tau)$	$p_0 + v_0\tau + \frac{1}{2}A_{max}\tau^2 - \frac{1}{6}J\tau^3$	$p_0 + v_0\tau - \frac{1}{6}J\tau^3$

$J = 0$

Phase	T4
$acc(\tau)$	0
$vel(\tau)$	V_{max}
$pos(\tau)$	$p_0 + V_{max}\tau$

รูปที่ 5 สมการการเคลื่อนที่ในแต่ละช่วงของ S-Curve velocity profile

การทดลอง 2.1 การทดลองลองเพื่อเปรียบเทียบผลของการใส่ และไม่ใส่ Refference Feedforward

สมมติฐาน

1. การเพิ่ม Reference Feedforward จะช่วยลด Tracking Error โดยเฉพาะในช่วง Acceleration และ Deceleration เนื่องจากระบบได้รับข้อมูล Reference Velocity ล่วงหน้าโดยไม่ต้องรอให้เกิด Error ก่อน
2. ประสิทธิภาพของ Feedforward ขึ้นอยู่กับความถูกต้องของแบบจำลองมอเตอร์ที่ใช้คำนวณ หากค่า Ke, B และ η มีความคลาดเคลื่อน Feedforward อาจชดเชยได้ไม่ตรงและอาจทำให้เกิด Error เพิ่มขึ้นได้

ตัวแปร

1. ตัวแปรต้น :
    - การเปิด/ปิด Reference Feedforward (เปิด / ปิด)
2. ตัวแปรตาม
    - Position Tracking Error $epos(t) = qref(t) - qmeas(t)[deg]$
    - RMS Error [deg]
    - Mean Absolute Error, MAE [deg]
    - Maximum Error [deg]
3. ตัวแปรควบคุม
    - ระยะการเคลื่อนที่ [deg]
    - ค่าพารามิเตอร์ PID ทั้ง Outer และ Inner Loop
    - S-Curve Trajectory Parameters $(v_{max}, a_{max}, j_{max})$
    - Control Frequency (1 kHz)
    - การปิด Disturbance Compensation ตลอดการทดลองนี้

การทดลอง 2.2 การทดลองเพื่อเปรียบเทียบผลของการใส่ และไม่ใส่ Disturbance compensation

สมมติฐาน

1. การเพิ่ม Disturbance Compensation โดยใช้ค่าประมาณ Td จาก Kalman Filter จะช่วยลด Steady-State Error ได้ดีกว่าการปล่อยให้ Integral Term ชดเชยเพียงอย่างเดียว เนื่องจาก Kalman Filter ประมาณค่าได้ทันทีโดยไม่มี Delay
2. ในกรณีที่มี Disturbance คงที่หรือเปลี่ยนแปลงช้า Kalman Filter จะสามารถติดตามและประมาณค่า Td ได้อย่างแม่นยำ

ตัวแปร

1. ตัวแปรต้น :
   - การเปิด/ปิด Disturbance Compensation (เปิด / ปิด)
1. ตัวแปรตาม
   - Position Tracking Error $epos(t) = qref(t) - qmeas(t)[deg]$
   - RMS Error [deg]
   - Mean Absolute Error, MAE [deg]
   - Steady-State Error [deg]
   - ค่าประมาณ Disturbance Torque $\widehat{\tau_d}$ จาก Kalman Filter [N·m]
2. ตัวแปรควบคุม
   - ระยะการเคลื่อนที่ [deg]
   - ค่าพารามิเตอร์ PID ทั้ง Outer และ Inner Loop
   - S-Curve Trajectory Parameters $(v_{max}, a_{max}, j_{max})$
   - Control Frequency (1 kHz)
   - การเปิด Reference Feedforward คงที่ตลอดการทดลองนี้

การทดลอง 2.3 การทดลองเพื่อเปรียบเทียบการใส่ Saturation ต่อ System response และ Safety

สมมติฐาน

1. การกำหนด Saturation Limit ที่ต่ำเกินไปจะทำให้ Response ช้าลงและ Tracking Error เพิ่มขึ้น เนื่องจาก Controller ไม่สามารถส่ง PWM ที่มากพอได้
2. การกำหนด Saturation Limit ที่สูงเกินไปจะทำให้ระบบ Response เร็วขึ้นแต่เสี่ยงต่อความเสียหายของ Gear และโครงสร้างเชิงกลจากแรงบิดที่มากเกินพิกัด
3. Saturation ที่เหมาะสมจะให้ Trade-off ที่ดีระหว่าง Response Speed และความปลอดภัย

ตัวแปร

1. ตัวแปรต้น :
   - ค่า Output Saturation Limit $u_{max}[PWMunits]$
2. ตัวแปรตาม
   - Rise Time [s]
   - Overshoot [%]
   - Settling Time [s]
   - PWM Command Output $u_{max}[PWMunits]$
   - Position Tracking Error $e_{pos}(t)[deg]$
3. ตัวแปรควบคุม
   - ค่าพารามิเตอร์ PID ทั้ง Outer และ Inner Loop
   - ระยะการเคลื่อนที่ [deg]
   - S-Curve Trajectory Parameters
   - Control Frequency (1 kHz)

การทดลอง 2.4 การทดลองเพื่อให้เห็นผลของ Integration windup และแนวทางแก้ไข

สมมติฐาน

1. เมื่อไม่มี Anti-windup ระบบจะเกิด Overshoot รุนแรงหลังจาก Output Saturation เนื่องจาก Integral Term สะสมค่ามากเกินไปในช่วงที่ Output ถูก Saturate
2. การเพิ่ม Anti-windup Clamping จะช่วยลด Overshoot และ Settling Time ได้อย่างมีนัยสำคัญโดยไม่ส่งผลเสียต่อ Steady-State Error

ตัวแปร

1. ตัวแปรต้น :
   - การเปิด/ปิด Anti-windup Clamping (เปิด / ปิด)
   - ค่า Anti-windup Limit ของ Outer Loop $I_{\lim it,pos}[deg/s]$
   - ค่า Anti-windup Limit ของ Inner Loop $I_{\lim it,vel}[PWMunits]$
2. ตัวแปรตาม
   - Overshoot [%]
   - Settling Time [s]
   - ค่าสูงสุดของ Integral Term ของ Outer Loop
   - ค่าสูงสุดของ Integral Term ของ Inner Loop
3. ตัวแปรควบคุม
   - ค่าพารามิเตอร์ PID ทั้ง Outer และ Inner Loop
   - ระยะการเคลื่อนที่ [deg]
   - S-Curve Trajectory Parameters
   - Control Frequency (1 kHz)

การทดลอง 2.5 การทดลองเพื่อให้เห็นผลของการปรับ Kp, Ki, Kd ต่อพฤติกรรมของระบบ และขั้นตอนการปรับภายใต้ Cascade Control

สมมติฐาน

1. การเพิ่ม Kp จะทำให้ Response เร็วขึ้นแต่ Overshoot เพิ่มขึ้น หากสูงเกินไประบบจะ Oscillate
2. การเพิ่ม Ki จะช่วยลด Steady-State Error แต่ถ้ามากเกินไปจะทำให้เกิด Oscillation เนื่องจาก Integral Lag
3. การเพิ่ม Kd จะช่วยลด Overshoot และเพิ่ม Damping แต่ถ้ามากเกินไปจะ Amplify Noise ใน Velocity Measurement
4. การ Tune Inner Loop ให้เสถียรก่อนแล้วจึง Tune Outer Loop จะให้ผลดีกว่าการ Tune พร้อมกัน เนื่องจาก Outer Loop มองเห็น Inner Loop เป็น Plant

ตัวแปร

1. ตัวแปรต้น :
   - ค่า $K_{p,pos}$ ของ Outer Loop
   - ค่า $K_{i,pos}$ ของ Outer Loop
   - ค่า $K_{d,pos}$ ของ Outer Loop
   - ค่า $K_{p,vel}$ ของ Inner Loop
   - ค่า $K_{i,vel}$ ของ Inner Loop
   - ค่า $K_{d,vel}$ ของ Inner Loop
2. ตัวแปรตาม
   - Velocity Tracking Error ในขั้นตอน Tune Inner Loop [deg/s]
   - Position Tracking Error ในขั้นตอน Tune Outer Loop [deg]
   - Overshoot [%]
   - Settling Time [s]
   - Steady-State Error [deg]
3. ตัวแปรควบคุม

- ระยะการเคลื่อนที่ [deg]
- S-Curve Trajectory Parameters
- Feedforward และ Anti-windup คงที่ตลอดการ Tune
    - พารามิเตอร์ตัวอื่นที่ไม่ได้ปรับในรอบนั้นคงที่
    - Control Frequency (1 kHz)

การทดลอง 2.6 การทดลองเพื่อเปรียบเทียบการเคลื่อนที่แบบ Trapezoidal กับ S-curve velocity profile

สมมติฐาน

1. S-Curve Velocity Profile จะให้ค่า Jerk ที่ต่ำกว่า Trapezoidal เนื่องจากมีการจำกัด Jerk อย่างชัดเจน ส่งผลให้การเคลื่อนที่นุ่มนวลกว่าและแรงกระแทกต่อโครงสร้างน้อยกว่า
2. Trapezoidal Profile จะให้ Tracking Error ที่สูงกว่าในช่วงเปลี่ยน Phase เนื่องจาก Acceleration เปลี่ยนแปลงแบบทันทีทันใด ทำให้ Controller ตามได้ยากกว่า
3. S-Curve จะให้ Control Effort ที่สม่ำเสมอกว่า Trapezoidal เนื่องจากไม่มีการเปลี่ยน Acceleration แบบกระชาก

ตัวแปร

1. ตัวแปรต้น :

    - ประเภทของ Velocity Profile (Trapezoidal / S-Curve)

4. ตัวแปรตาม

    - Position Tracking Error $e_{pos}(t)[deg]$
    - RMS Error [deg]
    - Velocity Profile $\dot{q}_{ref(t)}[deg/s]$
    - Acceleration Profile $\ddot{q}_{ref(t)}[deg/s^2]$
    - Jerk $\dddot{q}_{ref(t)}[deg/s^3]$
    - PWM Command Output (Control Effort) $u(t)[PWMunits]$
    - Settling Time [s]

5. ตัวแปรควบคุม
    - ระยะการเคลื่อนที่ [deg]
    - v_{max} และ a_{max} คงที่ทั้งสอง Profile
    - ค่าพารามิเตอร์ PID ทั้ง Outer และ Inner Loop
    - Control Frequency (1 kHz)

ขั้นตอนการดำเนินงาน Part 2

1. ต่อวงจร Motor Driver เข้ากับบอร์ด STM32 NUCLEO-G474RE โดย TIM1 CH1 เป็น PWM Output และ GPIOB PIN4 เป็น Direction Pin
2. ต่อ Encoder เข้ากับ TIM2 ในโหมด Quadrature (TI12) ความละเอียด 8192 counts/rev
3. เขียน function Trajectory generation ทั้งแบบ Trapezoidal และ S-curve Velocity Profile ลงบน STM32Cube IDE
4. เตรียม Log ข้อมูลผ่าน simulink โดยใช้ pos_deg, ref_pos_deg, vel_deg_s, ref_vel_deg_s ผ่าน UART ที่ 100 Hz
5. เขียน function Control (robot_arm.c) โดยใช้ Interrupt ในการเรียก Inner loop ใช้ความถี่ 1000 Hz และ Outter loop ใช้ความถี่ 200 Hz
6. เขียน function Kalman
7. ทำการ Tuning Cascade Control
    a. ปิด Outer loop แล้วปรับ Kp Ki Kd ของ Inner loop พร้อมทั้ง plot ให้เห็นแนวโน้ม
        i. Tune Kp และ Kd จนได้ Velocity Response ที่เสถียรและ Response เร็วพอ
        ii. เพิ่ม Ki เพื่อลด Steady-State Velocity Error
    b. เปิด Outer loop แล้วปรับ Kp Ki Kd ของ Outter loop พร้อมทั้ง plot ให้เห็นแนวโน้ม
8. ทดลองการเปิด/ปิด Refferrence Feedforward และ Disturbance Compensation

    8.1 ตั้งค่าระยะการเคลื่อนที่ที่ 360° และ S-Curve Trajectory โดยใช้ parameter คงที่ตลอดการทดลอง

    8.2 ทดสอบทั้ง 3 กรณีดังนี้

1. ปิด Refferrence Feedforward ปิด Disturbance Compensation

2. เปิด Refferrence Feedforward ปิด Disturbance Compensation

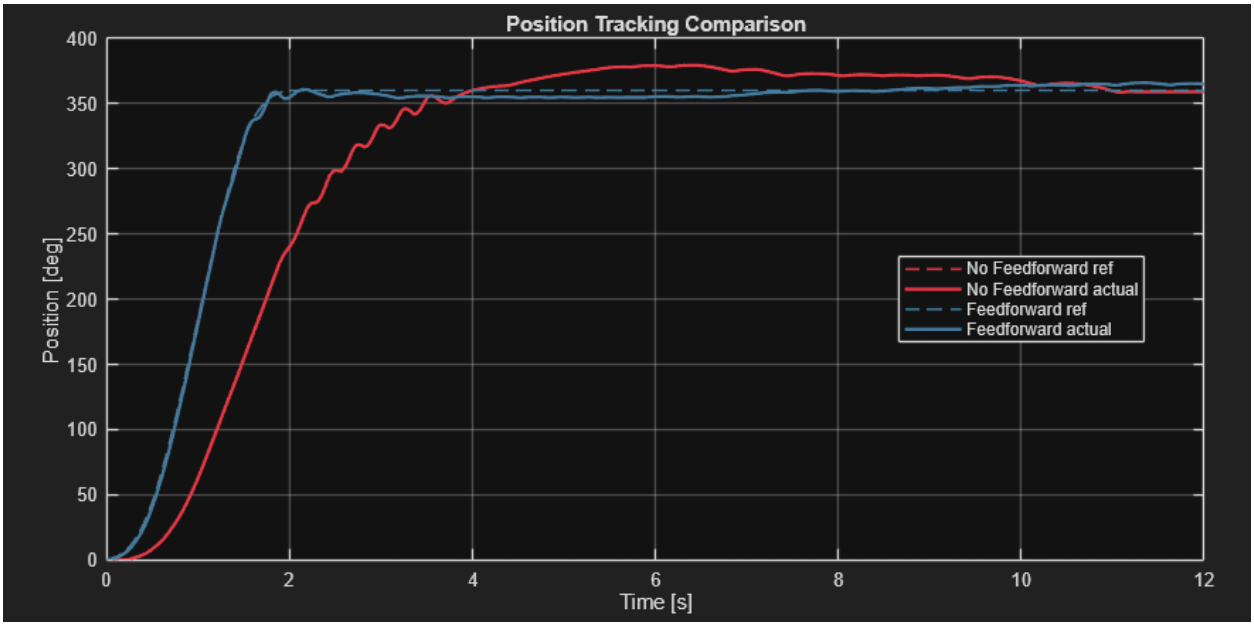
3. เปิด Refferrence Feedforward เปิด Disturbance Compensation

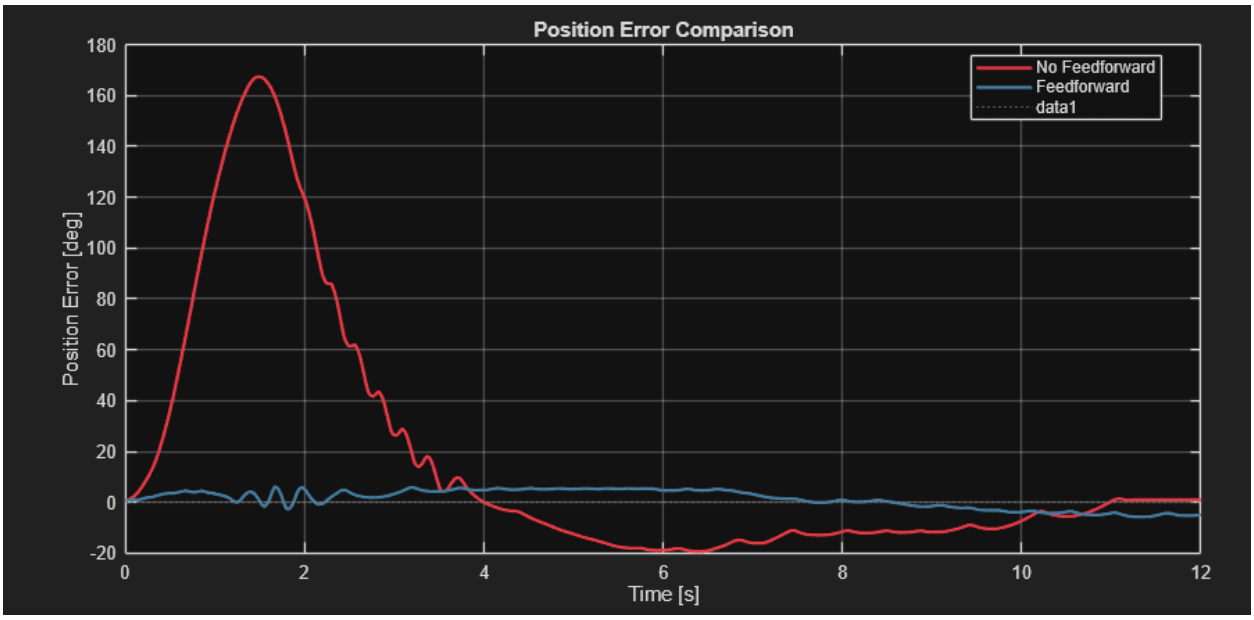
9. ทดลองการใส่/ไม่ใส่ Saturation

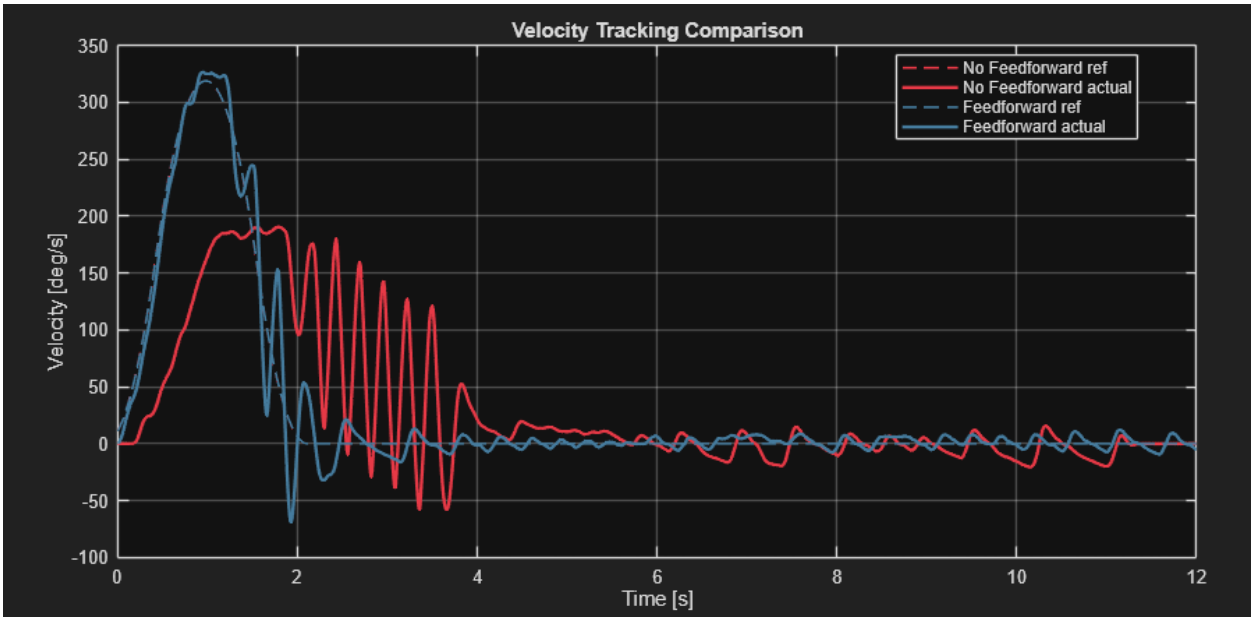
10. ทดลองผลของการเกิด Integration windup และการใช้ anti-windup ในการแก้ไข

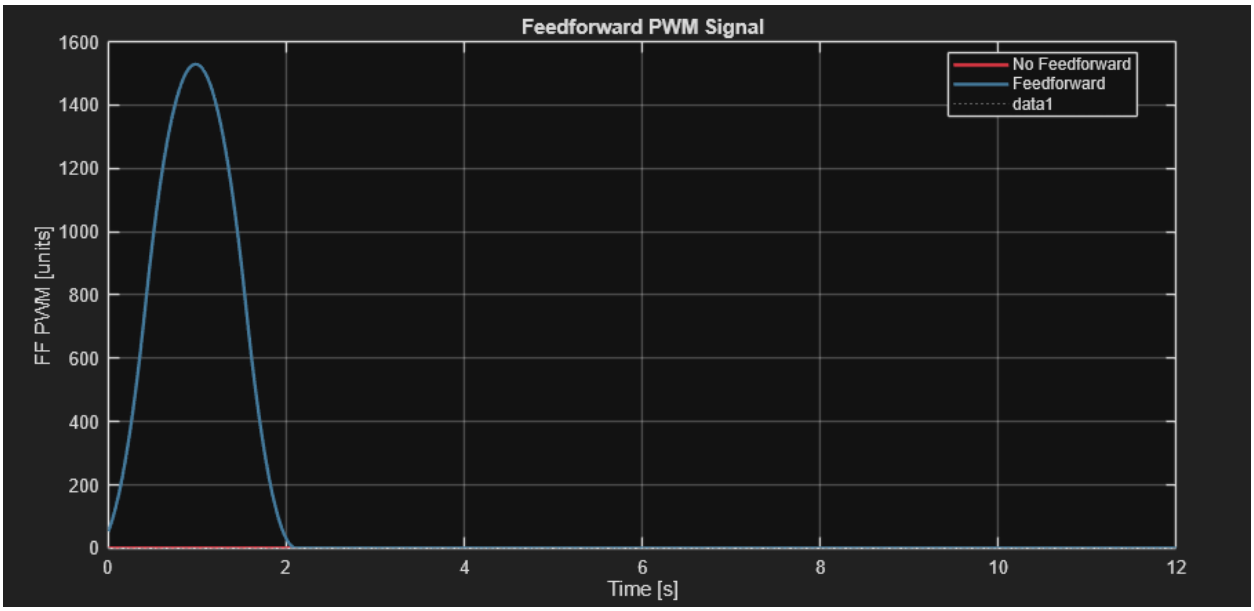
11. ทดลองเพื่อแสดงให้เห็นความแตกต่างระหว่าง Trapezoidal และ S-Curve velocity profile

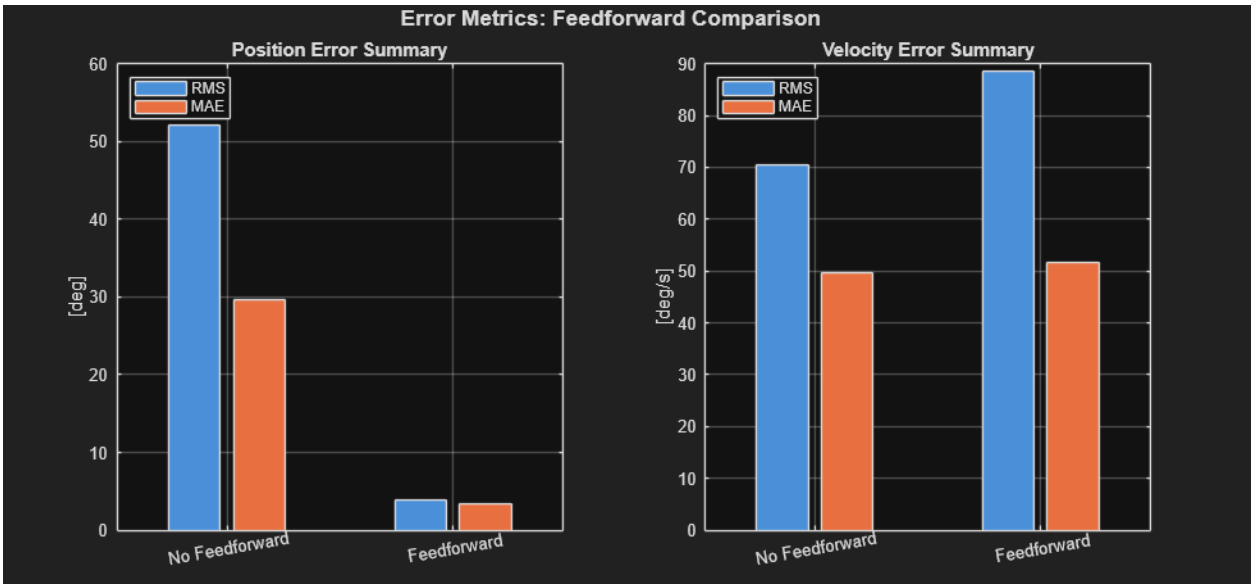
ผลการทดลองที่ 2.1




Position Error Comparison
No Feedforward
Feedforward
data1
Position Error [deg]
Time [s]
180
160
140
120
100
80
60
40
20
0
-20
0
2
4
6
8
10
12


Velocity Tracking Comparison
No Feedforward ref
No Feedforward actual
Feedforward ref
Feedforward actual
Velocity [deg/s]
Time [s]
350
300
250
200
150
100
50
0
-50
-100
0
2
4
6
8
10
12





สรุปผลการทดลองที่ 2.1

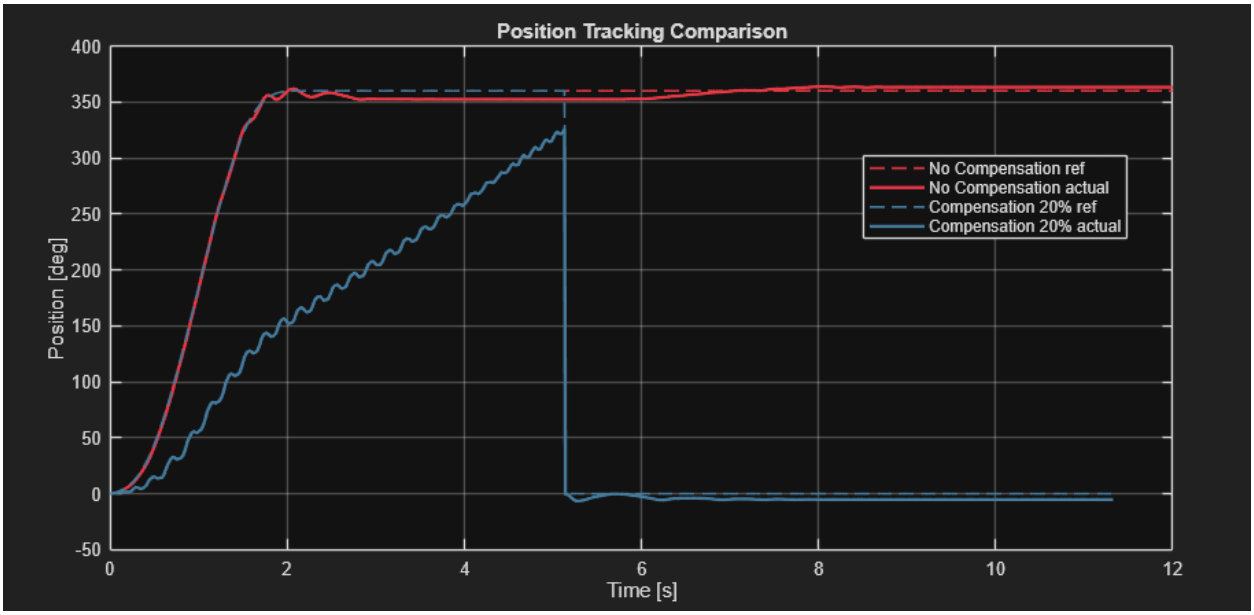
จากการเปรียบเทียบผลการทดลองระหว่างกรณีที่มีและไม่มี Reference Feedforward พบว่าการเพิ่ม Feedforward ช่วยลด Tracking Error ได้อย่างมีนัยสำคัญ โดยเฉพาะในช่วง Acceleration และ Deceleration ที่ Velocity เปลี่ยนแปลงรวดเร็ว กรณีที่ไม่มี Feedforward มี Position RMS Error สูงกว่ามาก เนื่องจาก Feedback Controller ต้องรอให้ Error เกิดขึ้นก่อนจึงจะชดเชยได้ ในขณะที่กรณีที่มี Feedforward ระบบส่ง

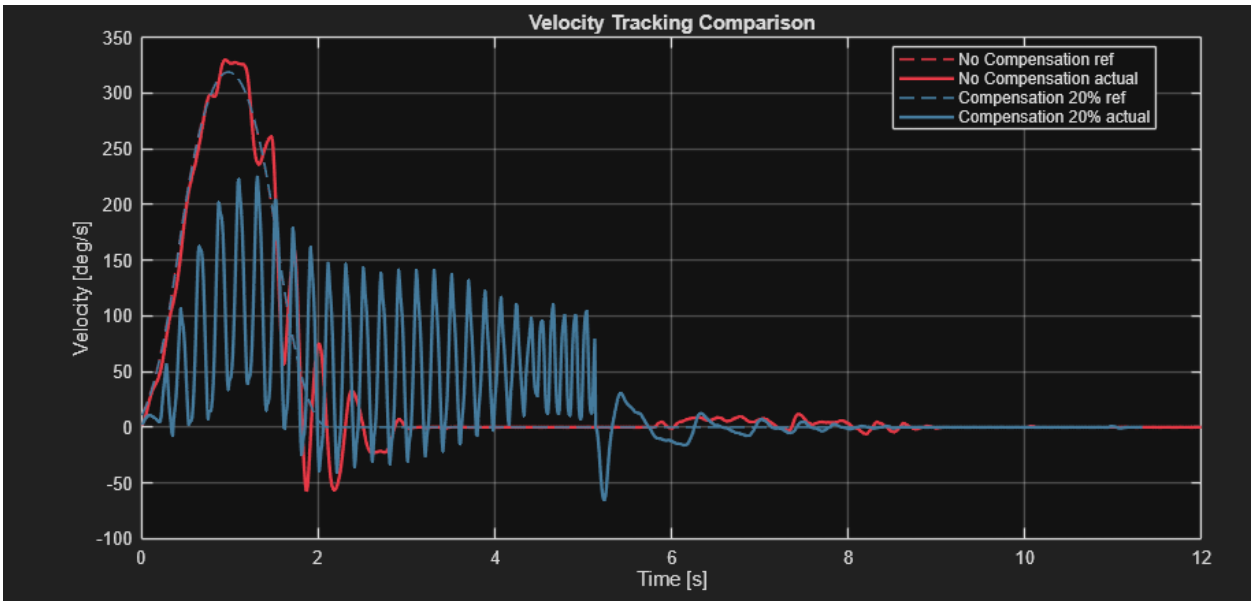
PWM ที่เหมาะสมล่วงหน้าก่อนที่ Error จะสะสม ส่งผลให้ Actual Position ตาม Reference ได้ใกล้เคียงกว่าตลอด Trajectory

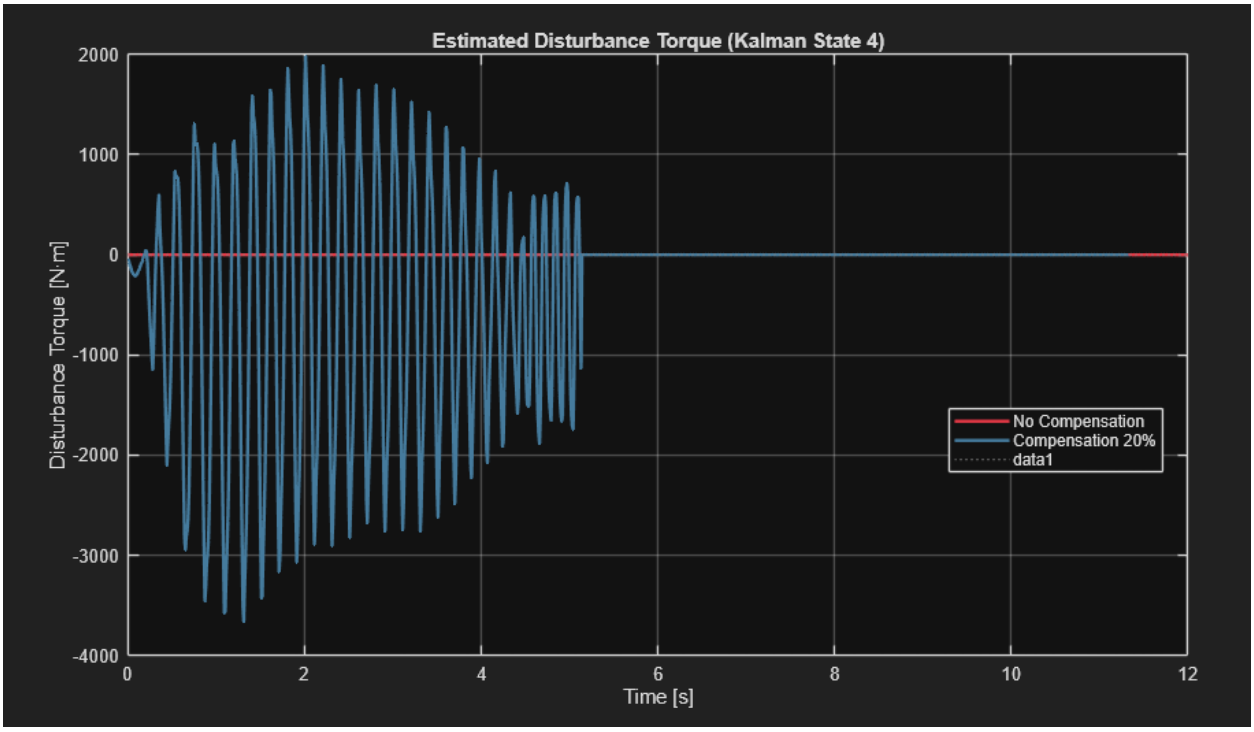
อภิปรายผลการทดลองที่ 2.1

Reference Feedforward เป็นเทคนิคที่นำ Reference Velocity จาก Trajectory Generator มาแปลงเป็น Control Signal โดยตรงโดยไม่ผ่าน Feedback Loop ทำให้ระบบสามารถชดเชย Velocity Lag ได้ล่วงหน้า เนื่องจาก Feedback Controller จะตอบสนองต่อ Error ที่เกิดขึ้นแล้วเท่านั้น ในช่วง Acceleration และ Deceleration ที่ Velocity เปลี่ยนแปลงรวดเร็ว Feedback อย่างเดียวจึงมักเกิด Tracking Error สูง การเพิ่ม Feedforward จะช่วยให้ Controller ส่ง PWM ที่เหมาะสมล่วงหน้าก่อนที่ Error จะสะสม ส่งผลให้ Tracking Error ลดลงโดยเฉพาะในช่วง Transient อย่างไรก็ตาม ประสิทธิภาพของ Feedforward ขึ้นอยู่กับความถูกต้องของค่าพารามิเตอร์มอเตอร์ที่ใช้คำนวณ หากค่า Ke, B หรือ η มีความคลาดเคลื่อน Feedforward อาจชดเชยได้ไม่ตรง และอาจเพิ่ม Error แทนที่จะลดได้

ผลการทดลองที่ 2.2




Velocity Tracking Comparison
No Compensation ref
No Compensation actual
Compensation 20% ref
Compensation 20% actual
Velocity [deg/s]
Time [s]
350
300
250
200
150
100
50
0
-50
-100
0
2
4
6
8
10
12


Estimated Disturbance Torque (Kalman State 4)
No Compensation
Compensation 20%
data1
Disturbance Torque [N·m]
Time [s]
2000
1000
0
-1000
-2000
-3000
-4000
0
2
4
6
8
10
12

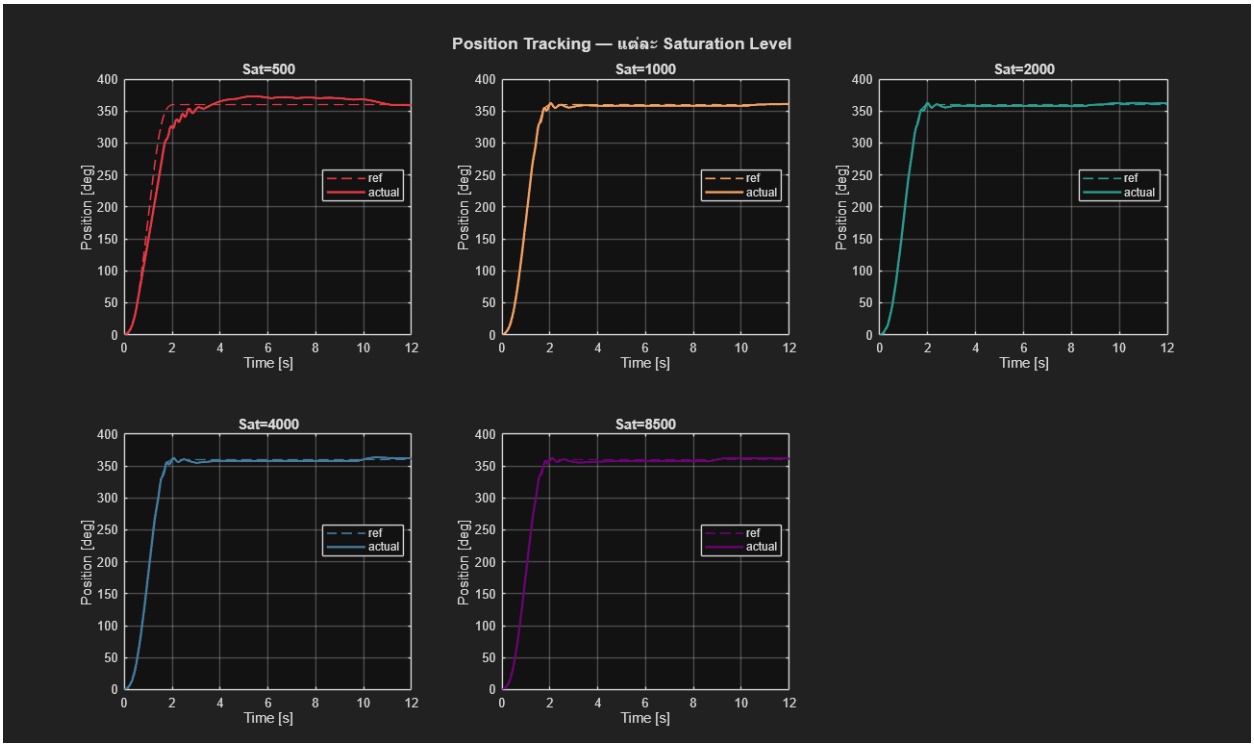
สรุปผลการทดลองที่ 2.2

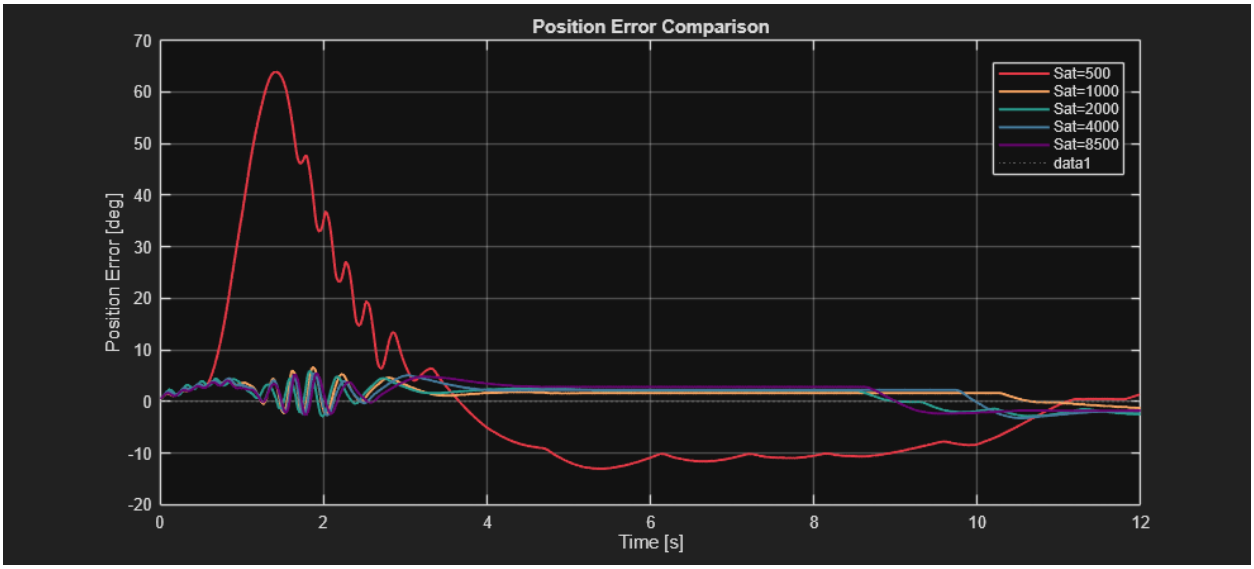
จากการทดลองพบว่าการเพิ่ม Disturbance Compensation โดยใช้ค่า **τ**^d จาก Kalman Filter State ที่ 4 ที่ระดับ 20% ส่งผลให้ Steady-State Error ลดลงได้ในกรณีที่ไม่มี Compensation อย่างไรก็ตาม ผลการทดลองแสดงให้เห็นว่าการชดเชยที่ 20% ทำให้ค่า RMS Error และ MAE ของทั้ง Position และ Velocity มีค่าสูงขึ้นเมื่อเทียบกับกรณีที่ไม่มี Compensation เนื่องจากค่า **τ**^d\hat{\tau}_d **τ**^d ที่ Kalman Filter ประมาณขึ้นมามีการ Oscillate สูงในช่วง Transient ซึ่งเมื่อนำมาบวกเข้า PWM Command โดยตรงจึงทำให้ระบบไม่เสถียรมากขึ้น สะท้อนให้เห็นว่าการเลือก Scaling Factor ของ Compensation มีผลต่อเสถียรภาพของระบบอย่างมาก

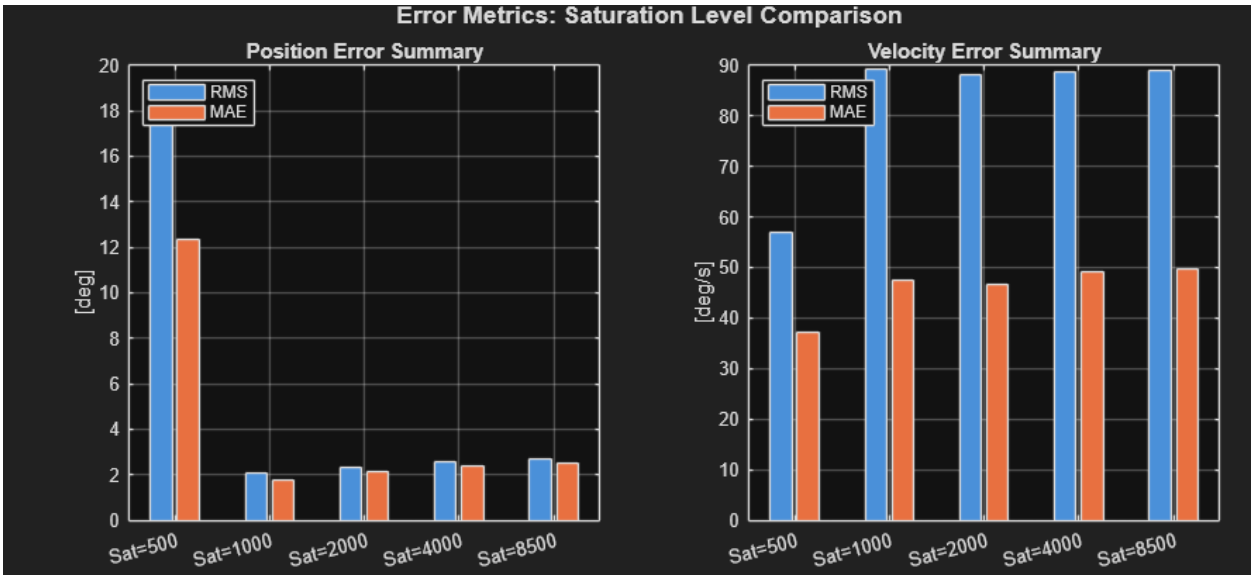
อภิปรายผลการทดลองที่ 2.2

Disturbance Compensation ในการทดลองนี้ใช้ค่าประมาณ Disturbance Torque จาก State ที่ 4 ของ Kalman Filter ซึ่งจำลองโมเมนต์บิดภาระเป็น Random Walk โดยนำค่าประมาณนี้แปลงเป็น PWM Compensation แล้วบวกเข้ากับ Output ของ Controller โดยตรง วิธีนี้มีข้อได้เปรียบเหนือการพึ่งพา Integral Term เพียงอย่างเดียว เนื่องจาก Integral Term ต้องรอให้ Error สะสมก่อนจึงจะชดเชยได้ ทำให้เกิด Delay ในการแก้ไข Steady-State Error ในทางกลับกัน Kalman Filter ประมาณค่า Disturbance Torque ได้ทันทีในแต่ละ Timestep ทำให้การชดเชยมีความรวดเร็วและแม่นยำกว่า โดยเฉพาะในกรณีที่ Disturbance มีลักษณะคงที่หรือเปลี่ยนแปลงช้า

ผลการทดลองที่ 2.3







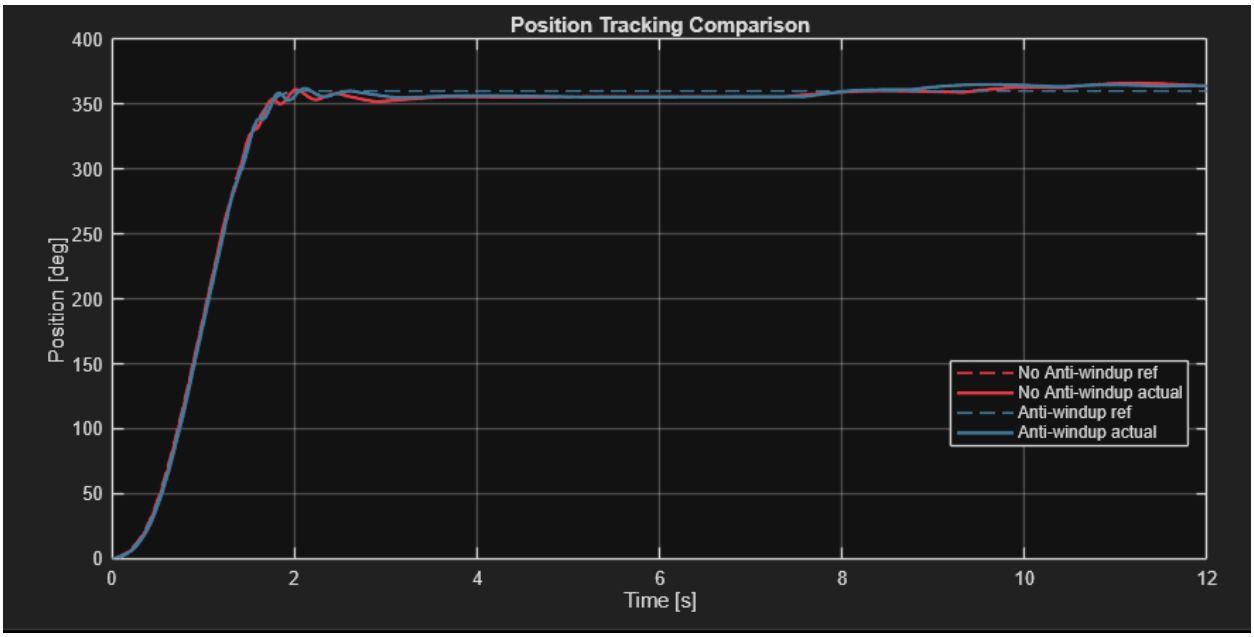
สรุปผลการทดลองที่ 2.3

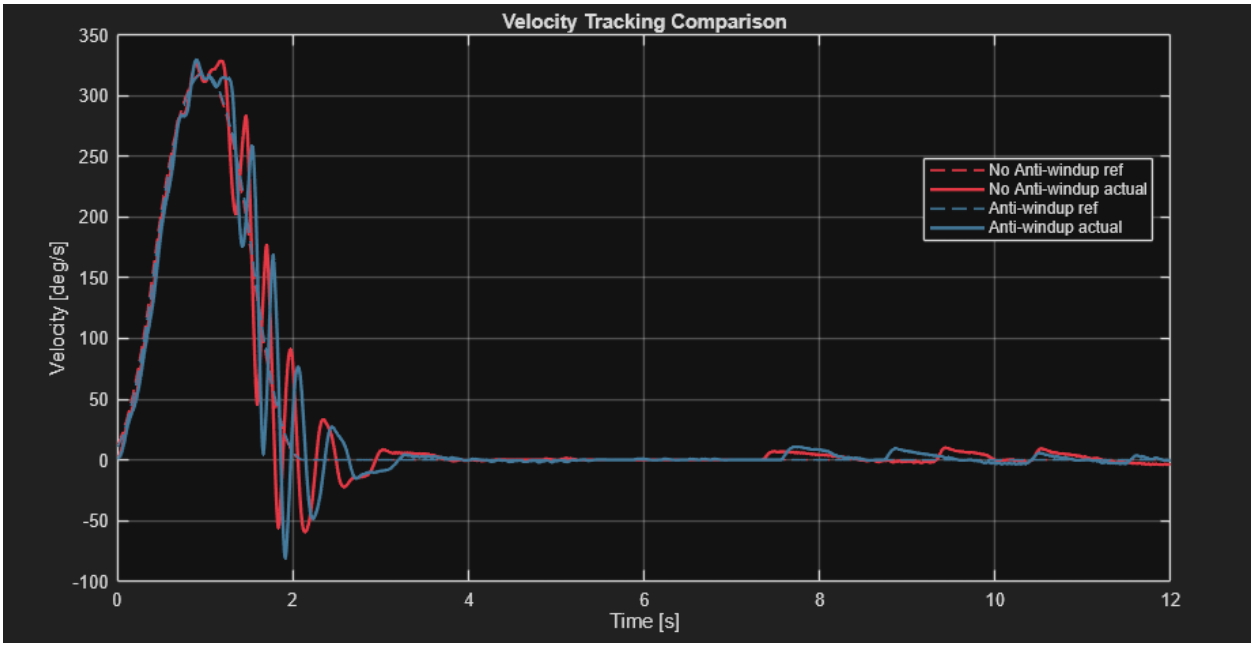
จากการทดลองที่ระดับ Saturation Limit 5 ระดับ ได้แก่ 500, 1000, 2000, 4000 และ 8500 พบว่าเมื่อ Saturation Limit ต่ำ เช่น 500 และ 1000 ระบบมี Tracking Error สูงในช่วง Acceleration เนื่องจาก Controller ไม่สามารถส่ง PWM ที่มากพอได้ ทำให้ Response ช้าและ Actual Position ตาม Reference ได้ไม่ทัน เมื่อเพิ่ม Saturation Limit ขึ้น Response เร็วขึ้นและ Tracking Error ลดลง โดยที่ระดับ 8500 ซึ่งเป็น Nominal Value ให้ผล Tracking ที่ดีที่สุดในการทดลองนี้

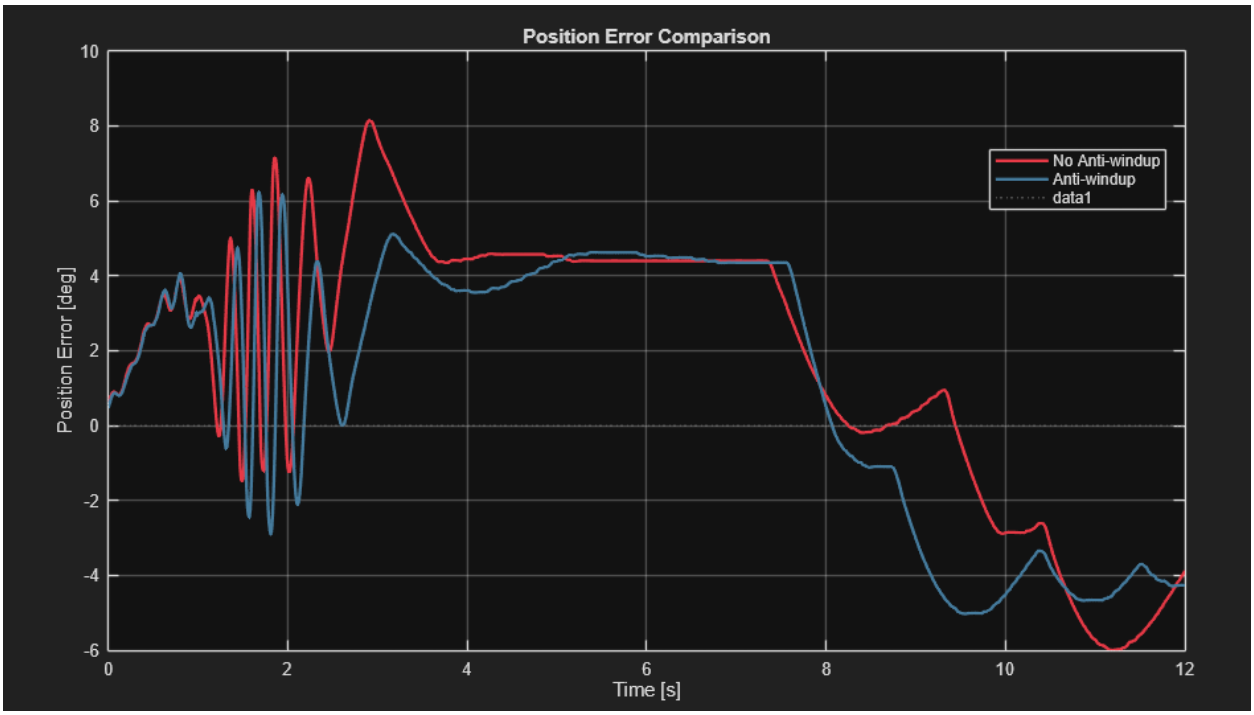
อภิปรายผลการทดลองที่ 2.3

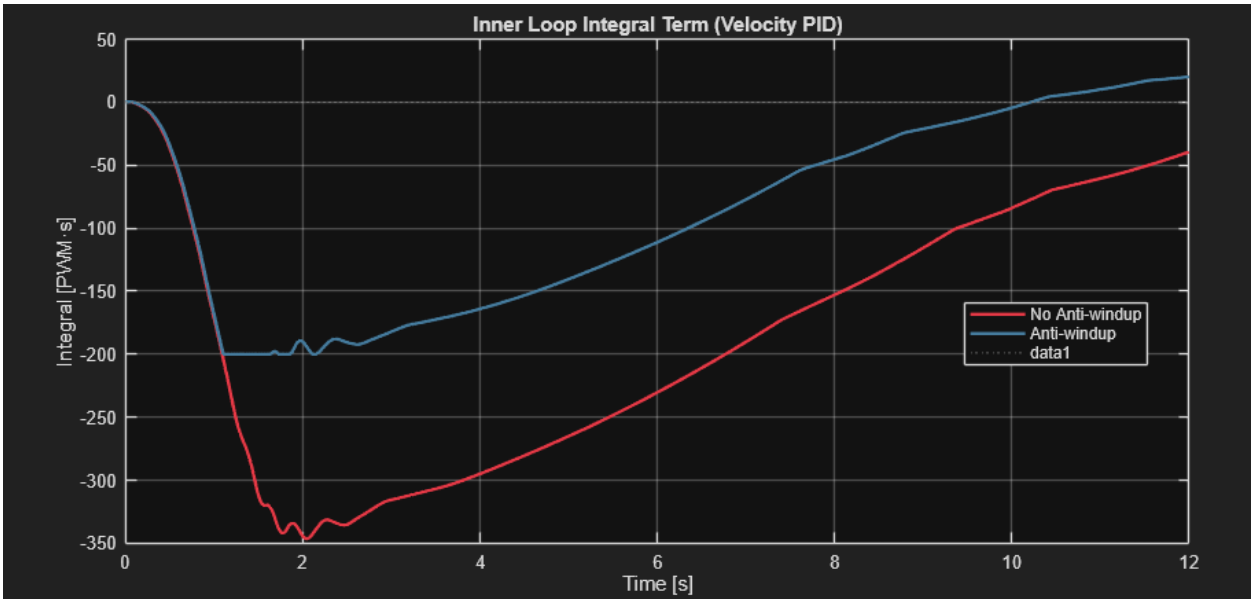
Output Saturation ทำหน้าที่จำกัด PWM Command ให้อยู่ในช่วงที่ Hardware รองรับได้อย่างปลอดภัย หาก Saturation Limit ต่ำเกินไป Controller จะไม่สามารถส่งแรงดันที่มากพอได้ ทำให้ Response ช้าลงและ Tracking Error เพิ่มขึ้น ในทางตรงข้าม หาก Saturation Limit สูงเกินไป ระบบจะตอบสนองเร็วขึ้น แต่เสี่ยงต่อความเสียหายของ Gear และโครงสร้างเชิงกล เนื่องจากแรงบิดที่เกิดขึ้นอาจเกินพิกัดที่ออกแบบไว้ นอกจากนี้มอเตอร์ยังมี Static Friction ทำให้ไม่ตอบสนองที่ PWM ต่ำมาก การเพิ่ม Deadband Compensation จึงช่วยชดเชยโดย Re-map PWM ออกจาก Dead Zone การเลือก Saturation Limit ที่เหมาะสมจึงเป็นการหา Trade-off ระหว่าง Response Speed และความปลอดภัยของระบบ

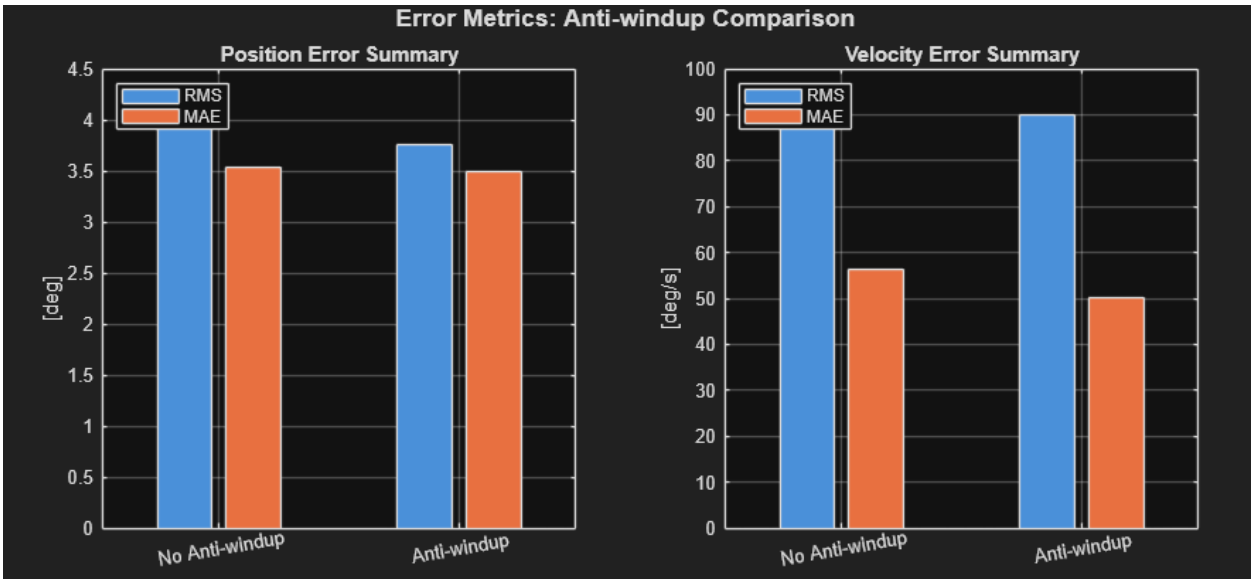
ผลการทดลองที่ 2.4






Position Error Comparison
Position Error [deg]
Time [s]
No Anti-windup
Anti-windup
data1
10
8
6
4
2
0
-2
-4
-6
0
2
4
6
8
10
12


Inner Loop Integral Term (Velocity PID)
Integral [PWM·s]
Time [s]
No Anti-windup
Anti-windup
data1
50
0
-50
-100
-150
-200
-250
-300
-350
0
2
4
6
8
10
12



สรุปผลการทดลองที่ 2.4

จากการเปรียบเทียบกรณีที่มีและไม่มี Anti-windup Clamping พบว่าทั้งสองกรณีให้ Position Tracking ที่ใกล้เคียงกันในภาพรวม อย่างไรก็ตามเมื่อพิจารณา Integral Term ของ Inner Loop พบว่ากรณีที่ไม่มี Anti-windup มีค่า Integral สะสมถึงประมาณ -350 PWM·s ในช่วง Deceleration และยังคงสะสมต่อเนื่อง ในขณะที่กรณีที่มี Anti-windup Integral Term ถูก Clamp ไว้ไม่เกินขีดจำกัดที่กำหนด ส่งผลให้ RMS Error ของ Velocity ลดลง

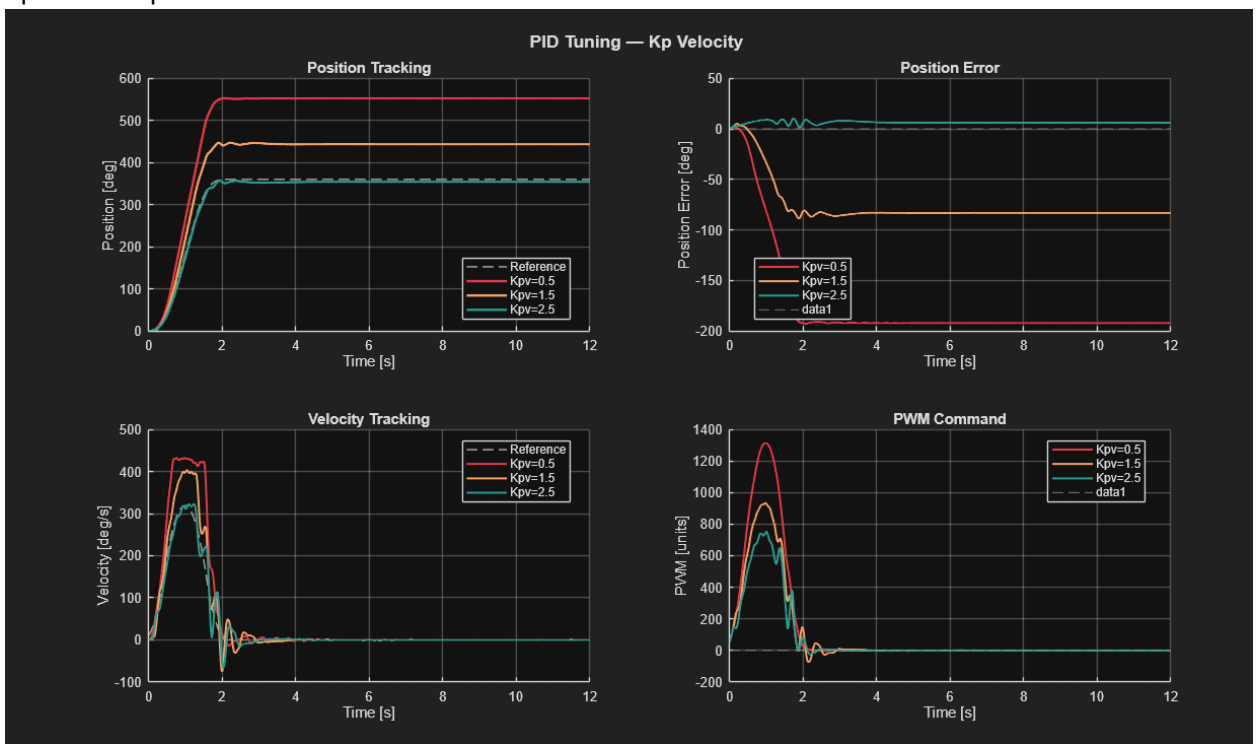
อภิปรายผลการทดลองที่ 2.4

Integration Windup เกิดขึ้นเมื่อ Controller Output ถึง Saturation Limit แต่ Integral Term ยังคงสะสมต่อไปเรื่อย ๆ เนื่องจาก Error ยังคงมีอยู่ ปัญหานี้ทำให้เมื่อระบบกลับสู่ Operating Range ปกติ Integral ที่สะสมไว้มากเกินไปจะผลักให้เกิด Overshoot รุนแรง และใช้เวลานานกว่าที่ระบบจะ Settle ลงได้ การแก้ปัญหาด้วย Anti-windup Clamping จะจำกัดค่า Integral ให้อยู่ในช่วงที่กำหนด ทำให้ Integral ไม่สะสมเกินขีดจำกัดในช่วงที่ Output Saturate ส่งผลให้ Overshoot และ Settling Time ลดลงอย่างมีนัยสำคัญ โดยไม่กระทบต่อ Steady-State Error ในระยะยาว

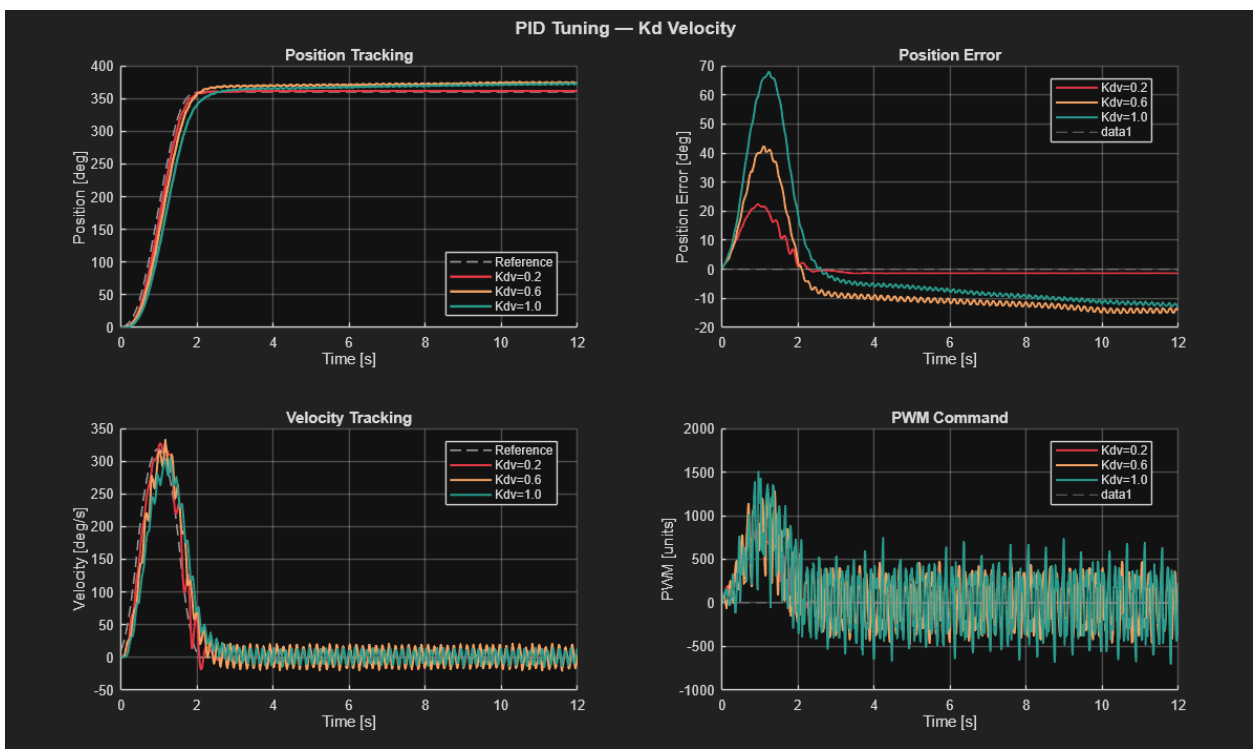
ผลการทดลองที่ 2.5

ตามลำดับการ Tune

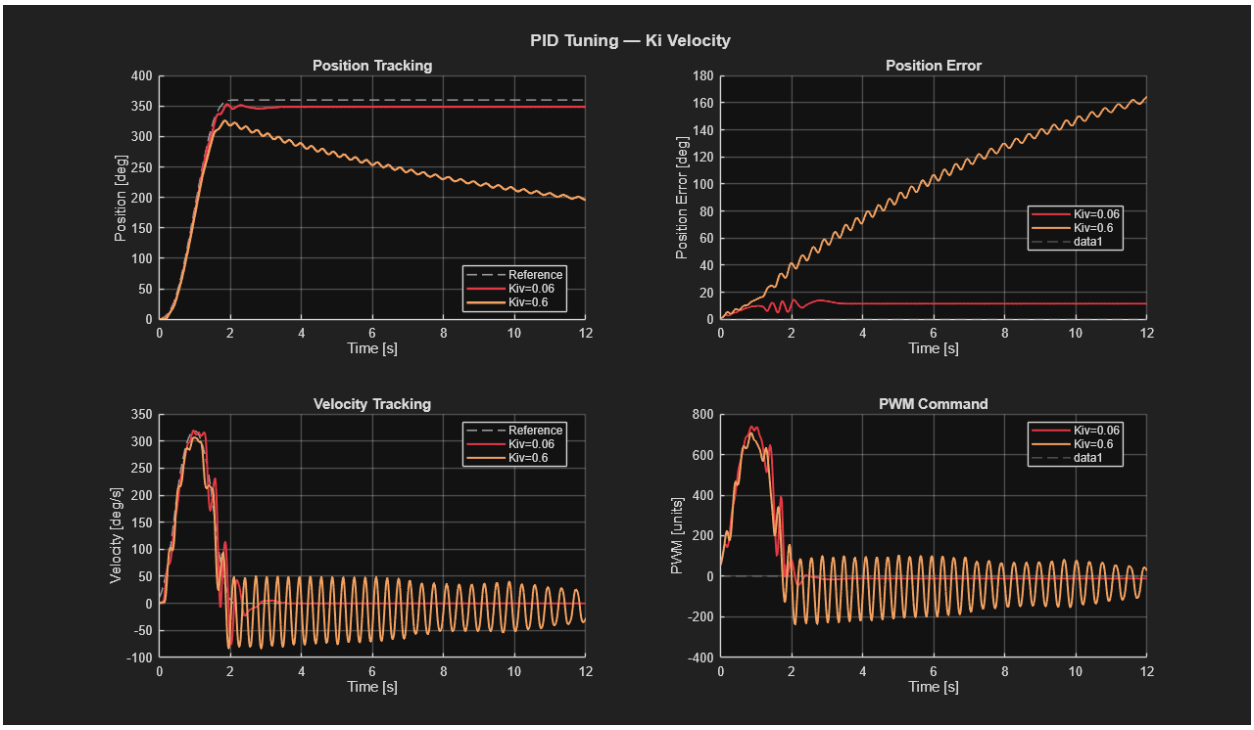
1. Kp Inner loop



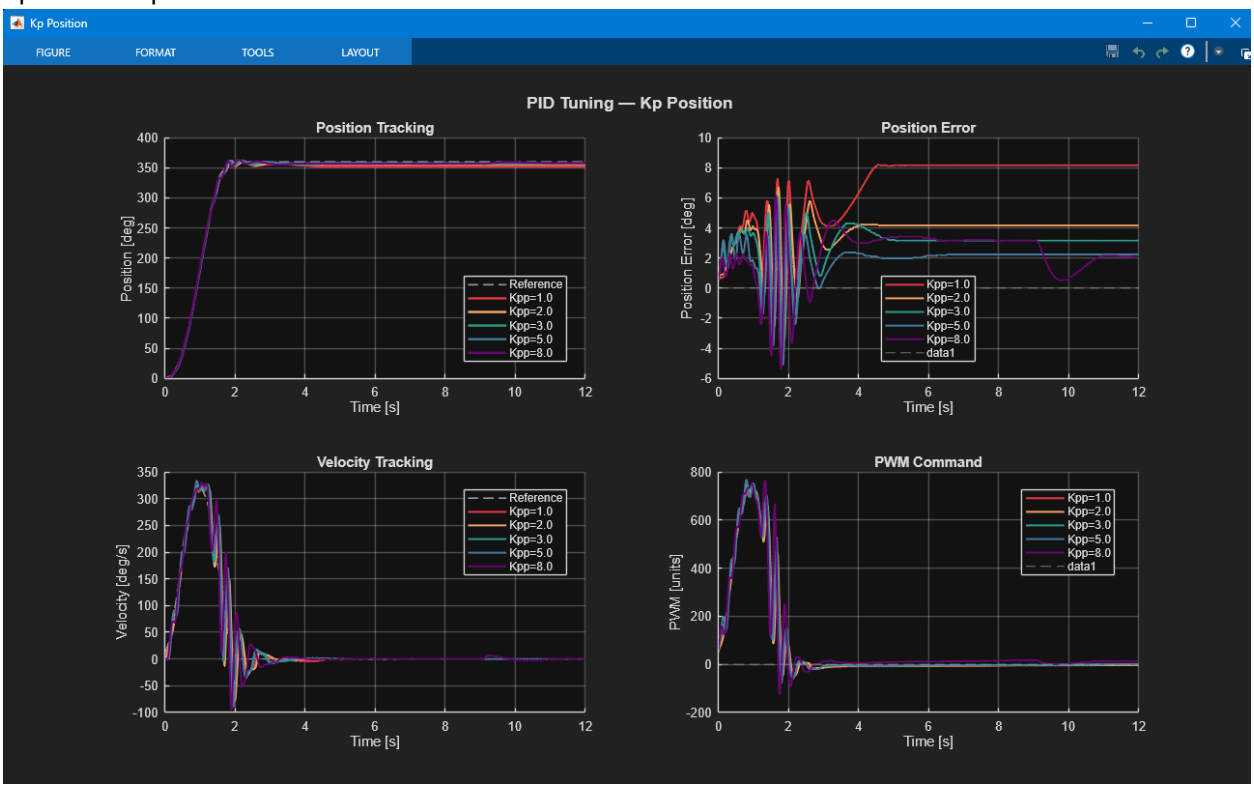
2. Kd Inner loop



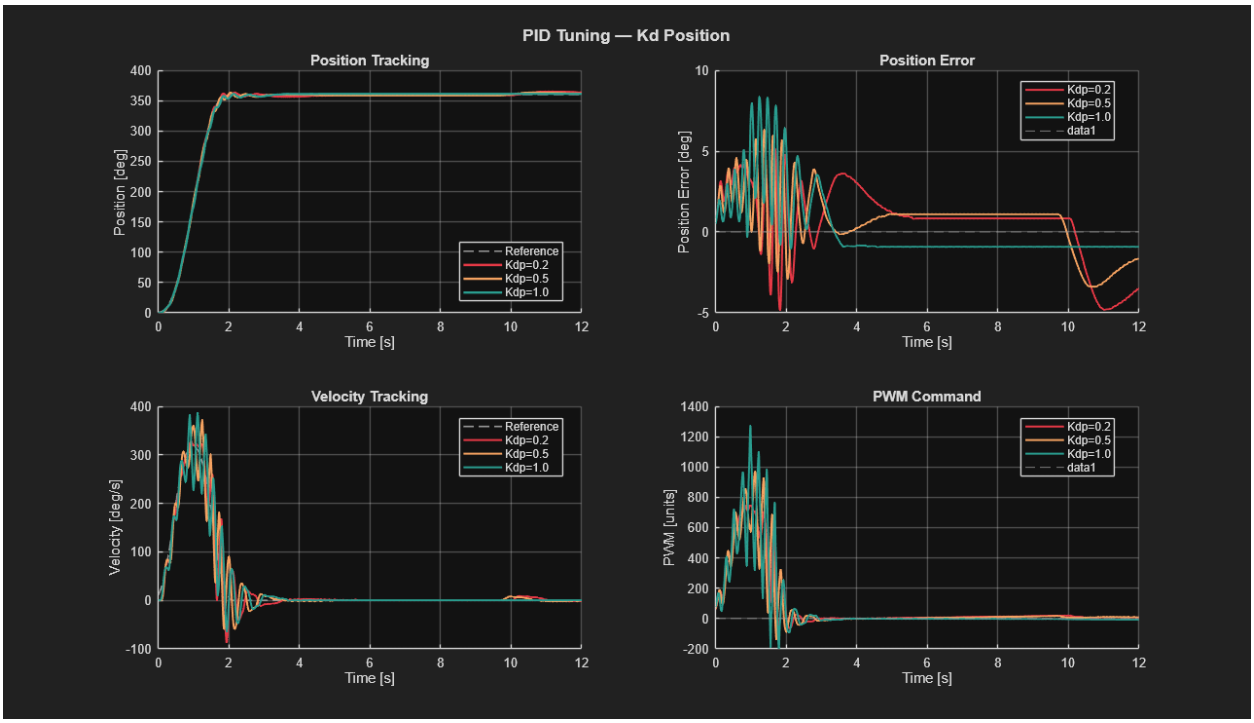
3. Ki Inner loop



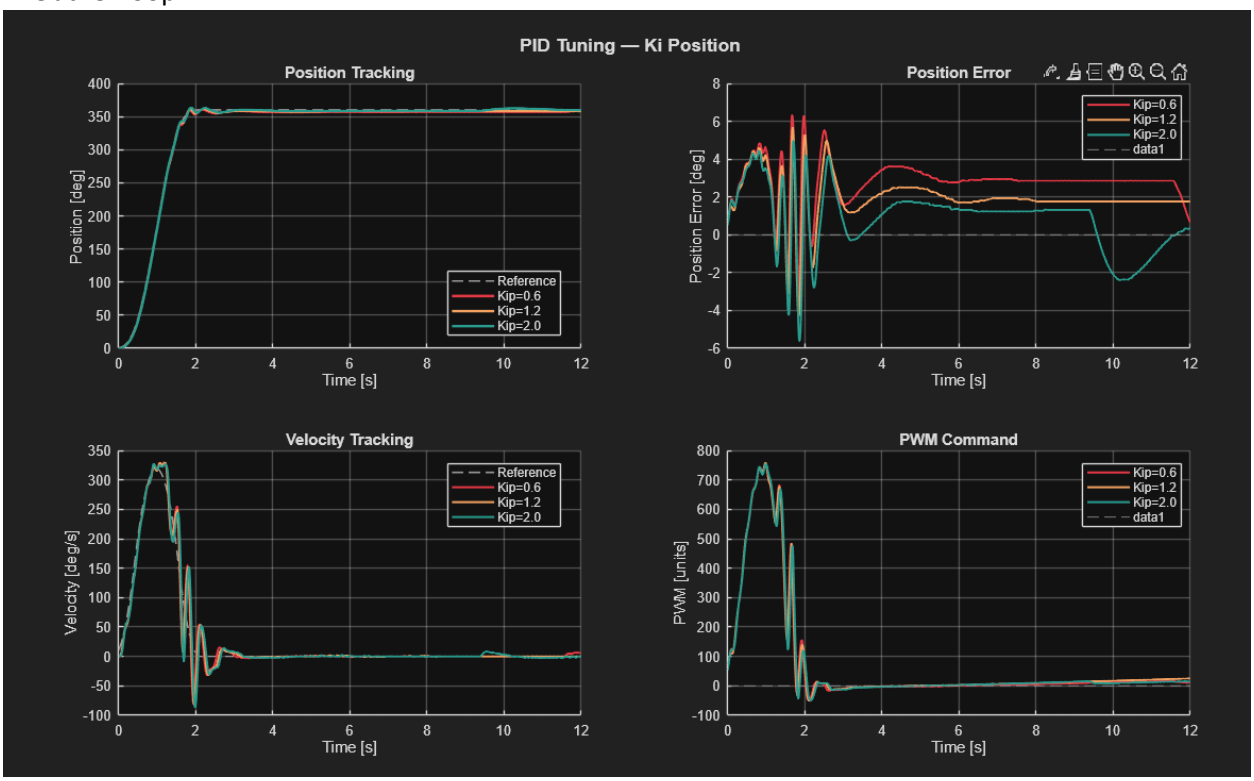
4. Kp outer loop



5. Kd Outer loop



6. Ki Outher loop



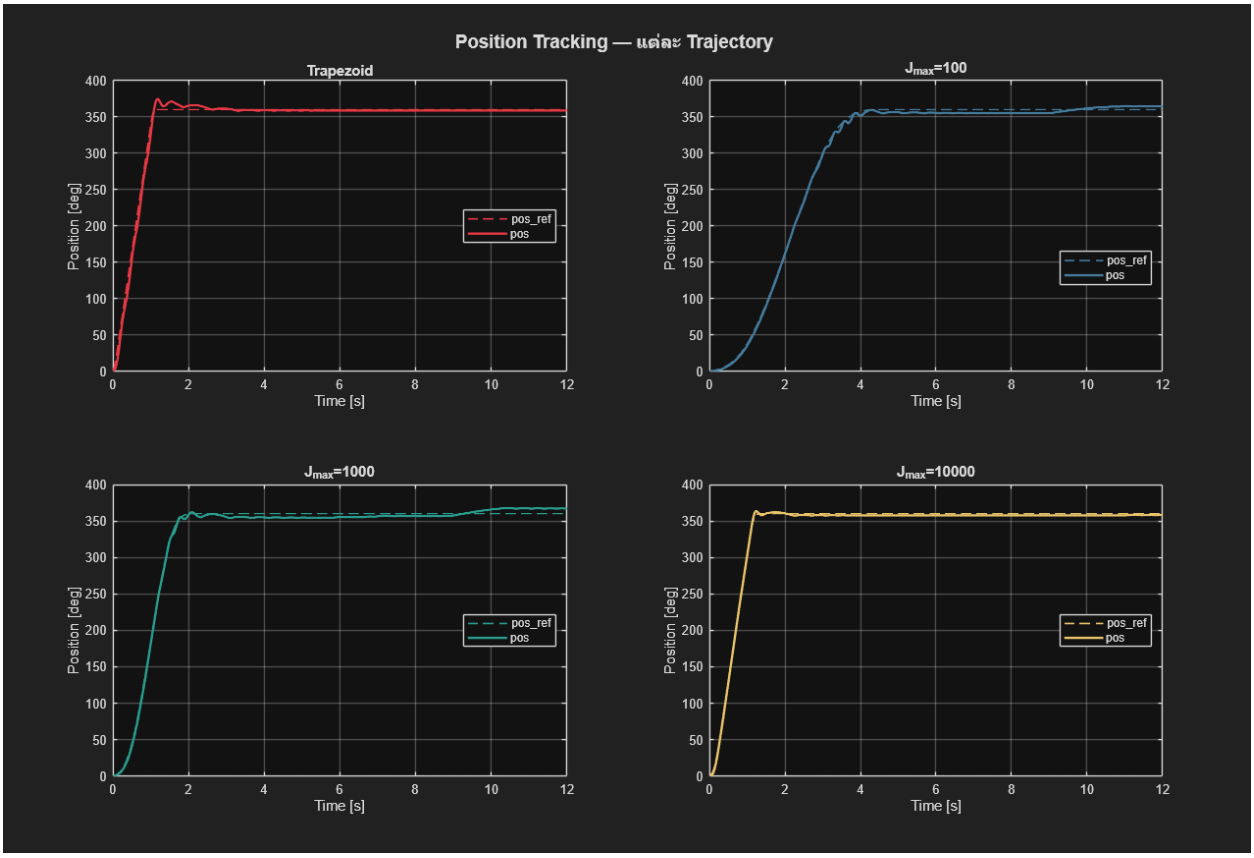
สรุปผลการทดลองที่ 2.5

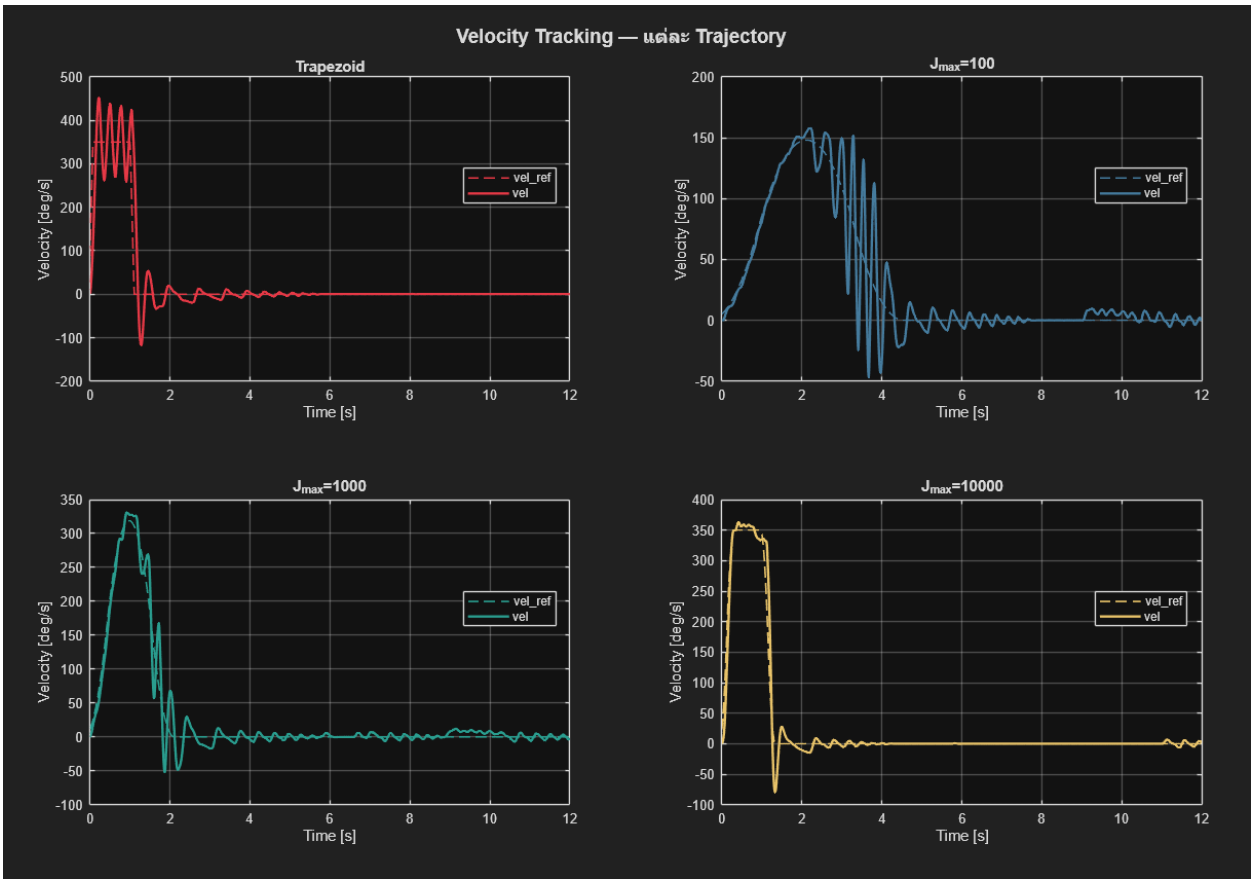
จากการ Tune พารามิเตอร์ตามลำดับ Inner Loop ก่อน Outer Loop พบว่าการเพิ่ม Kp,vel ทำให้ Response เร็วขึ้นแต่เมื่อสูงเกินไปทำให้เกิด Oscillation ใน Velocity การเพิ่ม Kd,vel ช่วยลด Oscillation แต่ทำให้ PWM Command มี Noise มากขึ้นอย่างเห็นได้ชัด การเพิ่ม Ki,velK_{i,vel} Ki,vel สูงเกินไปทำให้ระบบ Oscillate อย่างรุนแรงและ Position Error สะสมขึ้นเรื่อยๆ เมื่อเปิด Outer Loop และ Tune Kp,pos , Kd,pos, Ki,pos พบว่าระบบ Converge เขาสู่ Refferrence ได้ดีขึ้น

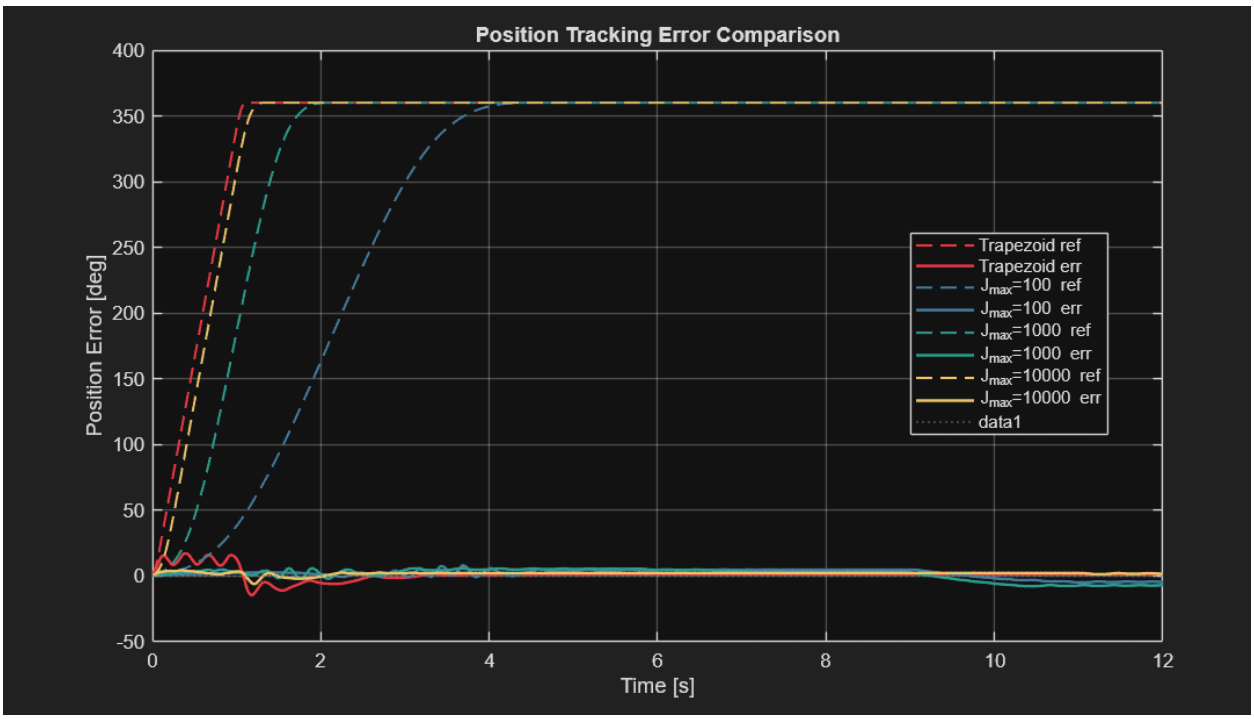
อภิปรายผลการทดลองที่ 2.5

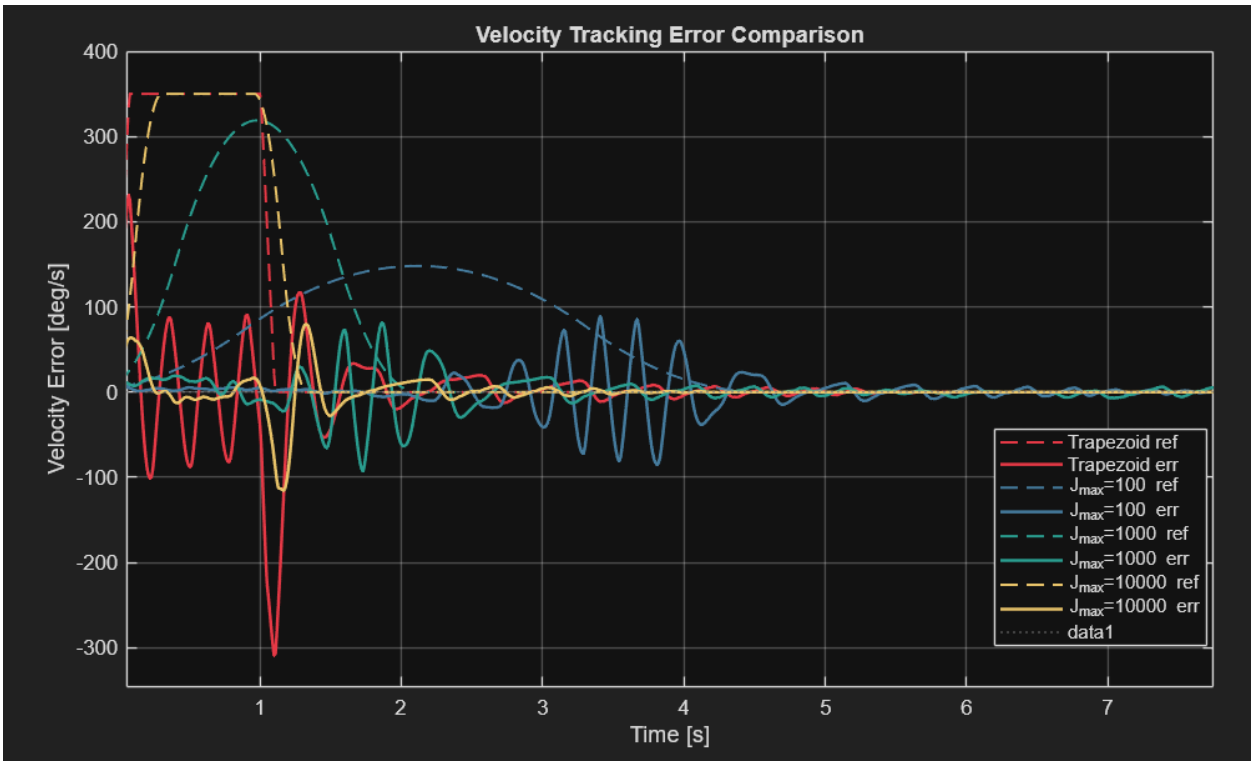
ภายใต้โครงสร้าง Cascade PID การ Tune ควรเริ่มจาก Inner Loop (Velocity) ให้เสถียรก่อน เนื่องจาก Outer Loop (Position) มองเห็น Inner Loop เป็น Plant ดังนั้นหาก Inner Loop ยังไม่เสถียร Outer Loop จะไม่สามารถทำงานได้ถูกต้อง ในแง่ของพารามิเตอร์แต่ละตัว การเพิ่ม Kp จะทำให้ Response เร็วขึ้นแต่ Overshoot เพิ่มขึ้น หากสูงเกินไปจะทำให้ระบบ Oscillate การเพิ่ม Ki ช่วยลด Steady-State Error แต่ถ้ามากเกินไปจะทำให้เกิด Oscillation จาก Integral Lag ส่วนการเพิ่ม Kd จะช่วยเพิ่ม Damping และลด Overshoot แต่หากมากเกินไปจะขยาย Noise ใน Velocity Measurement ให้รุนแรงขึ้น

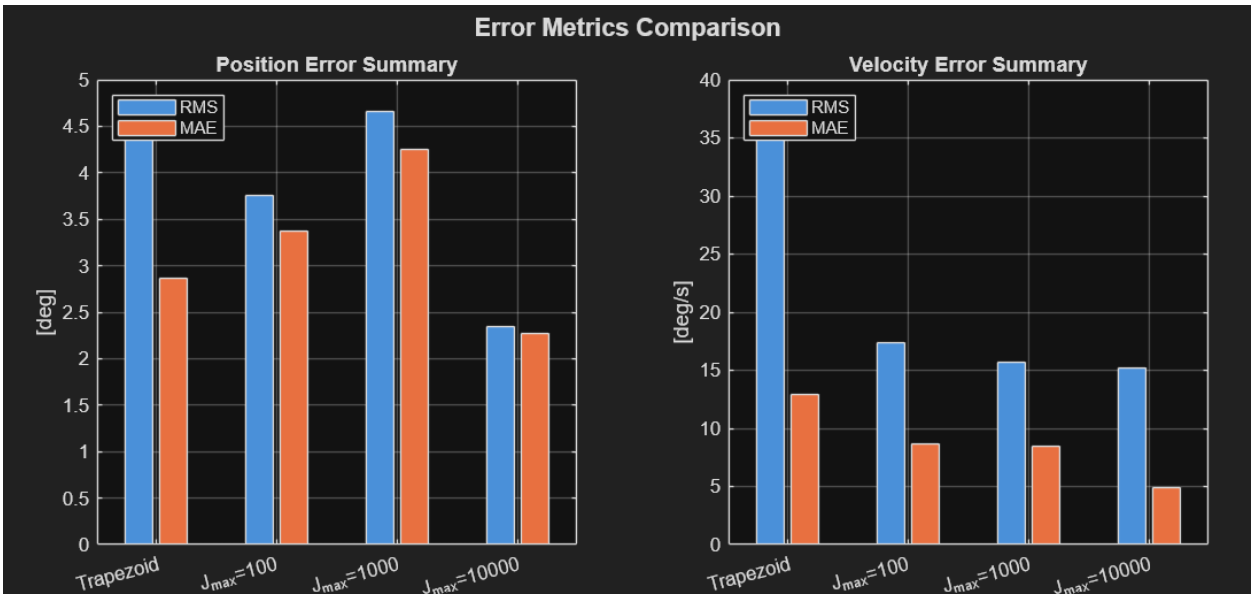
ผลการทดลองการทดลองที่ 2.6




Velocity Tracking — แต่ละ Trajectory
Trapezoid
Jmax=100
Jmax=1000
Jmax=10000
Velocity [deg/s]
Time [s]
vel_ref
vel


Position Tracking Error Comparison
Position Error [deg]
Time [s]
Trapezoid ref
Trapezoid err
Jmax=100 ref
Jmax=100 err
Jmax=1000 ref
Jmax=1000 err
Jmax=10000 ref
Jmax=10000 err
data1


Velocity Tracking Error Comparison
Velocity Error [deg/s]
Time [s]
400
300
200
100
0
-100
-200
-300
1
2
3
4
5
6
7
Trapezoid ref
Trapezoid err
Jmax=100 ref
Jmax=100 err
Jmax=1000 ref
Jmax=1000 err
Jmax=10000 ref
Jmax=10000 err
data1


Error Metrics Comparison
Position Error Summary
[deg]
RMS
MAE
Trapezoid
Jmax=100
Jmax=1000
Jmax=10000
Velocity Error Summary
[deg/s]
RMS
MAE
Trapezoid
Jmax=100
Jmax=1000
Jmax=10000

File	RMS Pos [deg]	MAE Pos [deg]	RMS Vel [d/s]	MAE Vel [d/s]
Trapezoid	4.6090	2.8619	38.4837	12.9256
J_{max}=100	3.7514	3.3769	17.3367	8.6841
J_{max}=1000	4.6634	4.2523	15.7222	8.4964
J_{max}=10000	2.3451	2.2712	15.2285	4.8543

สรุปผลการทดลองการทดลองที่ 2.6

จากการเปรียบเทียบในทางทฤษฎี Trapezoidal Velocity Profile และ S-Curve Velocity Profile ภายใต้ค่า v_{max} และ a_{max} เดียวกัน พบว่า S-Curve ให้ค่า Jerk ที่ต่ำกว่าอย่างมีนัยสำคัญ เนื่องจากมีการจำกัด Jerk ไม่เกิน J_{max} ทำให้ Acceleration เปลี่ยนแบบ Linear แทนที่จะเป็นแบบทันทีทันใดเหมือน Trapezoidal

จาก Tracking Error พบว่า Trapezoidal Profile มี RMS Error สูงกว่าในช่วงเปลี่ยน Phase ระหว่าง Acceleration กับ Cruise และ Cruise กับ Deceleration เนื่องจากการเปลี่ยน Acceleration อย่างฉับพลันทำให้ Controller ตามได้ยากกว่า ส่วน S-Curve ให้ Control Effort ที่สม่ำเสมอกว่า เนื่องจากการเปลี่ยน Acceleration อย่างราบเรียบทำให้ PWM Command เปลี่ยนแปลงน้อยกว่า

และจากการเปรียบเทียบ Trapezoidal Profile กับ S-Curve ที่ค่า *JmaxJ* ต่างกัน 3 ระดับ ได้แก่ 100, 1000 และ 10000 deg/s³ พบว่า Trapezoidal Profile มี RMS Position Error สูงที่สุดที่ 4.609 deg และ RMS Velocity Error สูงที่สุดที่ 38.484 deg/s เนื่องจาก Acceleration เปลี่ยนแปลงแบบทันทีทันใด ทำให้ Controller ตามได้ยาก S-Curve ที่ *Jmax=10000* deg/s³ ให้ RMS Position Error ต่ำที่สุดที่ 2.345 deg และ RMS Velocity Error ต่ำที่สุดที่ 15.285 deg/s ในขณะที่ S-Curve ที่ *Jmax=100* deg/s³ แม้จะมี Jerk ต่ำที่สุดแต่กลับให้ RMS Error สูงกว่า *Jmax=10000* เนื่องจากใช้เวลา Transition นานเกินไปทำให้ตาม Trajectory ได้ยากขึ้น

อภิปรายผลการทดลองที่ 2.6

ความแตกต่างของ Jerk ระหว่างสอง Profile ส่งผลโดยตรงต่อการสั่นสะเทือนของโครงสร้างแขนกล Trapezoidal ที่มี Jerk ไม่จำกัดที่จุดเปลี่ยน Phase ทำให้เกิด Impulsive Force ต่อชุดเฟือง ซึ่งในระยะยาวจะ

ส่งผลต่ออายุการใช้งานของอุปกรณ์เชิงกล S-Curve จึงเหมาะสมกว่าสำหรับงานที่ต้องการความแม่นยำสูง และมีข้อกำหนด Settling Time ≤ 0.5 วินาที ตาม Criteria ของการทดลองนี้

อย่างไรก็ตาม S-Curve ใช้เวลาเดินทางนานกว่า Trapezoidal เล็กน้อยเมื่อใช้ j_{max} เดียวกัน เนื่องจากต้องใช้เวลาเพิ่มในช่วง Jerk Phase ทั้ง 6 ช่วง (T1-T3 และ T5-T7) ดังนั้นการเลือก j_{max} ที่เหมาะสมจึงเป็นการ Trade-off ระหว่าง Smoothness กับ Travel Time

Reference Feedforward ที่ใช้ร่วมกับ S-Curve ให้ประโยชน์มากกว่าเมื่อใช้กับ Trapezoidal เนื่องจาก Feedforward คำนวณจาก $\dot{q}_{ref}(t)$ ที่เปลี่ยนแบบ Smooth ส่งผลให้ $u_{FF}(t)$ เปลี่ยนแบบ Smooth ด้วย ลดการกระโดดของ PWM Command ที่จุดเปลี่ยน Phase

ข้อเสนอแนะ

อ้างอิง

Part 3: Kalman Filter Design and State Estimation

การทดลองที่ 1 การออกแบบและพัฒนา Discrete Kalman Filter สำหรับการประมาณค่า State ของระบบ DC Motor บน STM32G474RE

จุดประสงค์

1. เพื่อสร้าง State-Space Model ของ DC Motor ใน Discrete Domain โดยมีอย่างน้อย 4 States พร้อมอธิบายความหมายทางกายภาพของแต่ละ State
2. เพื่อออกแบบและพัฒนา Discrete Kalman Filter (KF) พร้อม Matrix Operation เป็น Library บน STM32G474RE
3. เพื่อทำการกำหนดค่า Matrix Q และ R และวิเคราะห์ผลกระทบต่อ Trade-off ระหว่าง Responsiveness และ Smoothness ของ Filter
4. เพื่อตรวจสอบความสมเหตุสมผลของ State ที่ Kalman Filter ประมาณขึ้นมาโดยเปรียบเทียบกับการคำนวณโดยวิธีอื่น
5. เพื่อนำเสนอผลการทดลองด้วย Visualization ที่เหมาะสม พร้อมการวิเคราะห์เชิงสถิติ

สมมติฐาน

1. ระบบ DC Motor สามารถจำลองด้วย State-Space Model ใน Discrete Domain ได้อย่างแม่นยำ โดยใช้ค่า Inertia (J) และ Damping (B) ที่ได้จากการทดลองใน Lab 1
2. Noise ที่เกิดขึ้นในระบบ (ทั้ง Process Noise และ Measurement Noise) มีลักษณะเป็น Gaussian White Noise ซึ่งเป็นสมมติฐานพื้นฐานของ Kalman Filter
3. การปรับค่า Q (Process Noise Covariance) และ R (Measurement Noise Covariance) จะส่งผลต่อพฤติกรรมของ Filter ในลักษณะที่สามารถวัด และวิเคราะห์ได้เชิงปริมาณ
4. State ที่ Kalman Filter ประมาณขึ้นมาโดยไม่ได้วัดโดยตรง เช่น ความเร็วเชิงมุม จะมีความสอดคล้องกับการคำนวณด้วยวิธี Finite Difference จากสัญญาณ Encoder

ตัวแปร

1. ตัวแปรต้น :

- ความแตกต่างของค่า Matrix Q และ R ที่ใช้ใน Kalman Filter

2. ตัวแปรตาม :

- พฤติกรรมของ Kalman Filter ได้แก่ ความเสถียรของสัญญาณ และความเร็วในการตอบสนองต่อการเปลี่ยนแปลง
- ค่า Kalman Gain ที่เปลี่ยนแปลงตามค่า Q และ R
- พฤติกรรมของแบบจำลองทางคณิตศาสตร์
- ความสอดคล้องภายในของ Filter วัดด้วย NIS และ Whiteness ของ Innovation

3. ตัวแปรควบคุม :

ค่าคงที่ของมอเตอร์ที่ได้จาก Lab 1 (Ts = 0.005 s, Encoder AMT102 โหมด X4 = 8192 CPR)

พารามิเตอร์	ไม่มี Gripper	มี Gripper	หน่วย
Rm	1.802616747	1.802616747	Ohm
Lm	0.001093312315	0.001093312315	H
B	0.1996633	0.2380463	N·s/m
Eff	0.3012833	0.3089433	—
J	0.1061867	0.0685224	kg·m²
Ke	3.4436667	3.4499000	V·s/rad
Km	1.037519372	1.037519372	N·m/A

ตารางที่ 13 แสดงค่า Motor Parameter ที่ได้จากการเก็บค่าสองรูปแบบ

เอกสารและงานวิจัยที่เกี่ยวข้อง

1. **Kalman Filter**

Kalman Filter (KF) คืออัลกอริทึมที่ Rudolf E. Kalman คิดค้นในปี ค.ศ. 1960 เพื่อประมาณค่า State ของระบบ Dynamic ที่มี Noise จากสัญญาณวัดที่ไม่สมบูรณ์ KF ทำงาน 2 ขั้นตอนสลับกัน ได้แก่ (1) Prediction Step ใช้โมเดลทางคณิตศาสตร์ของระบบทำนาย State ล่วงหน้า และ (2) Correction Step ปรับแก้ค่า State โดยถ่วงน้ำหนักระหว่างค่าทำนายและค่าที่วัดได้ ผ่านตัวแปร Kalman Gain (K) ซึ่งคำนวณจากอัตราส่วนระหว่างความไม่แน่นอนของโมเดล (Q) และความไม่แน่นอนของ Sensor (R)

2. **State Variable**

State Variable คือตัวแปรที่เมื่อกำหนดค่าเริ่มต้นและ Input ของระบบแล้ว สามารถอธิบายพฤติกรรมของ ระบบได้ครบถ้วนในทุกช่วงเวลา สำหรับ DC Motor ในการทดลองนี้กำหนด State Vector เป็น $x = [\theta, \omega, i, m_L]^{\mathrm{T}}$ (ลำดับนี้ตรงกับเมทริกซ์ A ที่ใช้จริง) ประกอบด้วย

- θ — ตำแหน่งเชิงมุม (rad): วัดได้โดยตรงจาก Encoder
- $\omega = \dot{\theta}$ — ความเร็วเชิงมุม (rad/s): ประมาณค่าโดย KF (ไม่ได้วัดตรง)
- i — กระแสไฟฟ้า Armature (A): สัมพันธ์โดยตรงกับแรงบิดของมอเตอร์ ประมาณค่าโดย KF
- m_L — โมเมนต์บิดภาระ (N·m): แรงรบกวนที่ไม่ทราบค่า จำลองแบบ Random Walk ช่วยให้ Filter ทนทานต่อแรงรบกวนภายนอก

3. **Matrix Q (Process Noise Covariance)**

เป็น symmetric positive semi-definite matrix ขนาด 4×4 ที่แสดงระดับความไม่แน่นอนของ process model แต่ละ element บน diagonal แทนความไม่แน่นอนของแต่ละ State ตามลำดับ ได้แก่ Q_θ (ตำแหน่ง), Q_ω (ความเร็ว), Q_i (กระแส) และ Q_{mL} (โมเมนต์บิดภาระ) ค่า Q ที่สูงหมายความว่าเราไม่เชื่อถือสมการ State-Space ของ State นั้น ซึ่งอาจเกิดจากการประมาณค่าพารามิเตอร์ (เช่น J, B, Ke, Km) คลาดเคลื่อน มี nonlinearity ในระบบจริง หรือมีแรงรบกวนภายนอกที่โมเดลไม่ได้จำลองไว้ ทำให้ KF หันไปพึ่งพา measurement มากขึ้น ในทางกลับกัน Q ที่ต่ำหมายความว่าเราเชื่อว่า mathematical model แม่นยำมาก

4. **R (Measurement Noise Covariance)**

แสดงถึงระดับความไม่แน่นอนของการวัดจากเซนเซอร์ สำหรับระบบนี้ R เป็น scalar (ขนาด 1×1) เพราะมี output เดียวคือ displacement จาก Encoder AMT102 ค่า R ที่สูงสะท้อนว่าเซนเซอร์มี noise มากหรือเราไม่เชื่อถือการวัด ส่งผลให้ Kalman Filter พึ่งพา process model มากขึ้นและให้ผลลัพธ์ที่ smooth กว่า แต่ตอบสนองต่อการเปลี่ยนแปลงจริงช้าลง ค่า R สามารถประมาณได้จากการวัด variance ของสัญญาณเซนเซอร์ขณะระบบอยู่นิ่ง

5. **Measurement noise**

ความไม่แน่นอนที่เกิดจากตัวเซนเซอร์เอง เช่น quantization error, thermal noise สำหรับ Encoder AMT102 แหล่ง noise หลักคือ Quantization จากความละเอียดเชิงมุมที่จำกัด ซึ่งในทางปฏิบัติสามารถประมาณค่า R ได้จากการเก็บข้อมูลเซนเซอร์ขณะระบบนิ่งแล้วคำนวณหา variance

6. **Process noise**

ความไม่แน่นอนที่เกิดจาก mathematical model ของระบบ ซึ่งอาจเกิดจากการประมาณค่าพารามิเตอร์ผิดพลาด แรงภายนอกที่ไม่ได้จำลอง หรือ nonlinearity ของระบบจริง โดย Kalman Filter จำลอง process noise ว่าเป็น Gaussian white noise ที่มี covariance เท่ากับ Q

7. **ผลของอัตราส่วน Q/R ต่อ Kalman Gain**

Kalman Gain K กำหนดว่า Kalman Filter จะให้น้ำหนักกับ measurement หรือ model prediction มากกว่ากัน เมื่อ Q/R สูง Kalman Gain จะสูง ทำให้ filter เชื่อ sensor มากขึ้นและตอบสนองเร็ว แต่ noise ที่ผ่านมามากขึ้น เมื่อ Q/R ต่ำ Kalman Gain จะต่ำ ทำให้ filter เชื่อ model มากขึ้นและสัญญาณ smooth กว่า แต่อาจ lag หรือไม่ทันต่อการเปลี่ยนแปลงจริง

ขั้นตอนการดำเนินงาน

การทดลองที่ 1 การทดลองเพื่อหาผลกระทบของการปรับค่า Matrix Q และ R

1. เขียน Library สำหรับอ่าน Encoder ใน STM32CubeIDE
2. คำนวณหาสมการ Kalman และใส่ลงใน STM32CubeIDE

Continuous-time State-Space Model ของ DC Motor (4 States)

อ้างอิงจาก kalman-filter.com โมเดล DC Motor ถูกแยกออกเป็นสองส่วน ได้แก่ ส่วนไฟฟ้าและส่วนกลศาสตร์ จากวงจรไฟฟ้าของมอเตอร์ได้สมการ mesh ดังนี้

จากกฎแรงดันของคิร์ชฮอฟฟ์ (Kirchhoff's Voltage Law) จะได้สมการลูป (Mesh Equation) ของขดลวดอาร์เมเจอร์

$$V_s = L\frac{di}{dt} + Ri + V_e$$

แรงดันไฟฟ้าต้านกลับ (*Back-EMF Voltage:* V_e) มีความสัมพันธ์เชิงเส้นกับความเร็วรอบ

$$V_e = K_e\dot{\theta}$$

เมื่อย้ายข้างเพื่อหาอนุพันธ์ของกระแสไฟฟ้า (i) จะได้สมการเชิงอนุพันธ์อันดับหนึ่ง

$$i = \frac{1}{L}V_s - \frac{R}{L}i - \frac{K_e}{L}\dot{\theta}$$

สำหรับส่วนกลศาสตร์ใช้สมการ Newton ของการหมุนรอบแกน

$$J\ddot{\theta} = m_m - m_R - m_L$$

เมื่อแทนค่าและย้ายข้างเพื่อหาค่าความเร่งเชิงมุม $\left(\ddot{\theta}\right)$ จะได้สมการเชิงอนุพันธ์อันดับสอง

$$\ddot{\theta} = \frac{K_T}{J}i - \frac{b}{J}\dot{\theta} - \frac{1}{J}m_L$$

เมื่อนิยาม State Vector เป็น $\boldsymbol{x} = [\theta, \omega, i, m_L]^{\mathrm{T}}$ และจัดในรูป State-Space ได้ดังนี้

$$\dot{x}(t) = Ax(t) + Bu(t) + Gz(t)$$

$$y(t) = Cx(t) + v(t)$$

โดยที่ System Matrix A, Input Matrix B, Disturbance Matrix G และ Output Matrix C มีค่าดังนี้

$$x(t) = \begin{bmatrix} \theta \\ \dot{\theta} \\ i \\ m_L \end{bmatrix} \text{(state order: } \theta,\ \omega,\ i,\ m_L\text{)}$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{b}{J} & \frac{K_T}{J} & -\frac{1}{J} \\ 0 & -\frac{K_e}{L} & -\frac{R}{L} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \\ 0 \end{bmatrix}, u(t) = V_s$$

$$G = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

เนื่องจากระบบวัดได้เพียงค่าตำแหน่งเชิงมุม (θ) ผ่านเซนเซอร์ เมทริกซ์การวัด C จึงเป็น

$$C = [1 \quad 0 \quad 0 \quad 0]$$

โดย State ที่ 3 (mL) จำลองว่าการเปลี่ยนแปลงของโมเมนต์บิดภาร $\frac{d(m_L)}{dt} = z(t)$ เป็น Gaussian Noise ทำให้ mL เปลี่ยนแบบ Random Walk ซึ่งเหมาะสมเมื่อโมเมนต์บิดภายนอกไม่เป็นศูนย์และไม่ทราบค่าแน่ชัด

Discrete-time System Model และการ Discretize ด้วย ZOH

เนื่องจาก STM32G474RE ทำงานใน Discrete Domain จึงต้องแปลงโมเดลด้วยวิธี Zero-Order Hold (ZOH) ที่ Sample Time Ts โดยใช้สูตร Matrix Exponential

$$A_d = e^{AT_s}$$

$$B_d = \left(\int_0^{T_s} e^{A\tau} d\tau\right) B$$

ในทางปฏิบัติบน STM32 คำนวณค่า Ad และ Bd บน Python/MATLAB ก่อน แล้วฝังเป็น Hard-coded Matrix ใน Firmware เพื่อหลีกเลี่ยงการคำนวณ Matrix Exponential ใน Real-time โมเดล Discrete ที่ได้มีรูปแบบ

$$x(k) = A_d x(k-1) + B_d u(k-1) + w(k-1)$$

$$y(k) = Cx(k) + v(k)$$

สมการ Discrete Kalman Filter (4 States)

Kalman Filter ทำงาน 2 ขั้นตอนสลับกันทุก Sampling Period ดังนี้

ขั้นที่ 1 Prediction Step (Time Update)

ใช้โมเดลของระบบทำนาย State และ Error Covariance ล่วงหน้า โดยใช้ค่าจาก Timestep ก่อนหน้า

$$\widehat{x^-}(k) = A_d \hat{x}(k-1) + B_d u(k-1) \quad \ldots (1)$$

$$P^-(k) = A_d P(k-1) A_d^T + Q \quad \ldots (2)$$

โดยที่ $\widehat{x^-}(k)$ คือ Prior State Estimate (4x1), $P^-(k)$ คือ Prior Error Covariance (4x4), Q คือ Process Noise Covariance (4x4)

ขั้นที่ 2 Correction Step (Measurement Update)

เมื่อได้รับค่าการวัดจาก Encoder ให้คำนวณ Kalman Gain แล้วปรับแต่ง State Estimate

$$K(k) = P^{-}(k)C^{T}(CP^{-}(k)C^{T} + R)^{-1} \quad \dots (3)$$

$$\hat{x}(k) = \widehat{x^{-}}(k) + K(k)\big(y(k) - C\widehat{x^{-}}(k)\big) \quad \dots (4)$$

$$P(k) = (I - K(k)C)P^{-}(k) \quad \dots (5)$$

โดยที่ $K(k)$ คือ Kalman Gain (4x1 เนื่องจาก C มี 1 row), $y(k)$ คือค่าที่อ่านได้จาก Encoder (scalar), $y(k) - C\widehat{x^{-}}(k)$ คือ Innovation หรือ Residual, I คือ Identity Matrix ขนาด 4x4

การตรวจสอบ Observability

ก่อนนำโมเดลไปใช้จริงต้องตรวจสอบว่าระบบ Observable หรือไม่ โดยคำนวณ Observability Matrix O แล้วตรวจสอบ rank

$$\mathcal{O} = \begin{bmatrix} C \\ CA_d \\ CA_d^2 \\ CA_d^3 \end{bmatrix}$$

$$\text{rank}(\mathcal{O}) = 4$$

หาก rank$(\mathcal{O}) = 4$ แสดงว่าระบบ Fully Observable กล่าวคือสามารถประมาณค่า State ทั้ง 4 ได้จากการวัดตำแหน่งเพียงอย่างเดียว ซึ่งเป็นเงื่อนไขจำเป็นสำหรับการใช้ Kalman Filter อย่างถูกต้อง

3. ติดตั้งหุ่นให้พร้อมสำหรับการทดลอง
4. ทดลองรันหุ่นเพื่ออ่านค่า Angular velocity ที่ได้จากการคำนวณองศาที่อ่านได้จาก Encoder โดยการปรับค่า Matrix Q และ R ตามลำดับ ดังตาราง

ตารางที่ 14 แสดงค่าที่ใช้ในการทดลองเพื่อหา Matrix Q และ R

ครั้งที่	Q	R
1	diag$(10^{-8}, 10^{-4}, 10^{-3}, 10^{-1})$	4.90229×10^{-6}
2	diag$(10^{-8}, 10^{-4}, 10^{-3}, 10^{-1})$	4.90229×10^{-8}
3	diag$(10^{-8}, 10^{-4}, 10^{-3}, 10^{-1})$	4.90229×10^{-10}
4	diag$(10^{-10}, 10^{-6}, 10^{-5}, 10^{-3})$	4.90229×10^{-8}
5	diag$(10^{-8}, 10^{-4}, 10^{-3}, 10^{-1})$	4.90229×10^{-8}
6	diag$(10^{-6}, 10^{-2}, 10^{-1}, 10^{1})$	4.90229×10^{-8}

เนื่องจากฮาร์ดแวร์แขนกลจริงมีเอาต์พุตที่วัดค่าได้เพียงตัวแปรเดียว คือ ตำแหน่งเชิงมุม ผ่าน Encoder รุ่น AMT102 โดยไม่มีเซนเซอร์กระแสไฟฟ้า เมทริกซ์ R จึงลดรูปทางคณิตศาสตร์จากเมทริกซ์สเปซมิติเหลือเพียงค่า

สเกลาร์ (Scalar ขนาด 1×1) ในการทดลองนี้ ทางทีมได้เลือกกำหนดรูปแบบการนับพัลส์ของ Encoder ในไมโครคอนโทรลเลอร์ STM32 เป็น โหมด X4 (Quadrature Decoding) เพื่ออ่านสัญญาณจากทุกขอบขาขึ้นและขาลงของทั้ง Channel A และ B ซึ่งจะเพิ่มความละเอียดจากสเปกเดิม 2,048 พัลส์ต่อรอบ เป็น 8,192 CPR ความคลาดเคลื่อนขั้นต่ำสุดเชิงทฤษฎีของการแปลงสัญญาณดิจิทัลตัวนี้ เรียกว่า Quantization Noise ซึ่งมีสูตรคำนวณตามสถิติของระเบียบดิจิทัลดังนี้

$$R_{theoretical} = \frac{\Delta^2}{12}$$

โดยที่ Δ คือ ความละเอียดขั้นต่ำสุดในหน่วยเรเดียนที่เซนเซอร์สามารถตรวจจับได้ใน 1 steppulse

$$\Delta = \frac{2\pi}{8192} \approx 0.0007669904 \text{ rad}$$

เมื่อนำค่าความละเอียดสเต็ปนี้ไปแทนค่าเข้าสู่สมการ *Quantization Noise* จะได้ว่า

$$R_{theoretical} = \frac{(0.0007669904)^2}{12} \approx 4.90229 \times 10^{-8}$$

ทางคณะผู้จัดทำจึงเลือกใช้ค่า

$$R = 4.90229 \times 10^{-8}$$

และเมทริกซ์ *Q* มีขนาด 4×4 ตามสเปซเวกเตอร์ 4 สถานะของระบบ

$$x = [\theta \quad \omega \quad i_a \quad \tau_d]^T$$

เมทริกซ์นี้ทำหน้าที่ระบุระดับความไม่มั่นใจในแบบจำลองคณิตศาสตร์ ยิ่งค่าในสเตตใดมีค่า *Q* สูง ตัวกรอง Kalman Filter จะยิ่งปล่อยให้สมการคณิตศาสตร์ของสเตตนั้นคลาดเคลื่อนได้มากขึ้น และหันไปพึ่งพาการคำนวณปรับแต่งจากเซนเซอร์จริงแทน

เนื่องจากค่า Process Noise ไม่สามารถใช้เซนเซอร์ใดๆ วัดได้โดยตรง จึงต้องกำหนดค่าเริ่มต้นบนแนวทแยงมุม จากการวิเคราะห์ธรรมชาติทางฟิสิกส์ของระบบแขนกลจริง ดังนี้

จากการวิเคราะห์ธรรมชาติของระบบแขนกล ค่า *Q* บน *diagonal* ถูกกำหนดดังนี้ โดยอ้างอิงจากขนาดและหน่วยของแต่ละ *State* ตามลำดับ ได้แก่ ตำแหน่งเชิงมุม (*rad)* ความเร็วเชิงมุม (*rad/s)* โมเมนต์บิดภาร (*Nm)* และกระแสไฟฟ้า (*A)*

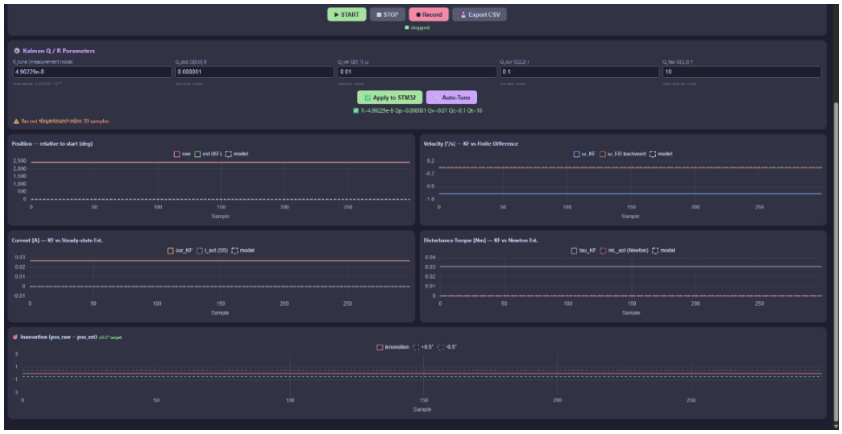
$$Q = \begin{bmatrix} Q_\theta & 0 & 0 & 0 \\ 0 & Q_\omega & 0 & 0 \\ 0 & 0 & Q_{\mathrm{mL}} & 0 \\ 0 & 0 & 0 & Q_{\mathrm{i}} \end{bmatrix}$$

โดยค่าเริ่มต้นที่เลือกไว้สำหรับการทดลอง คือ $Q = diag(10^{-8}, 10^{-4}, 10^{-3}, 10^{-1})$

$$Q = \begin{bmatrix} 10^{-8} & 0 & 0 & 0 \\ 0 & 10^{-4} & 0 & 0 \\ 0 & 0 & 10^{-3} & 0 \\ 0 & 0 & 0 & 10^{-1} \end{bmatrix}$$

ซึ่งสะท้อนว่าเราไม่แน่ใจในการจำลองโมเมนต์บิดภาร (x_3) และกระแส (x_4) มากกว่าตำแหน่งและความเร็ว

5. STM32 ถูกตั้งค่าให้ส่งข้อมูลออกทาง UART ทุก Sampling Period ในรูปแบบ CSV โดย Sampling Rate ของ Kalman Filter Loop กำหนดไว้ที่ 1000 Hz และ Q/R Matrix ถูกเขียนให้ปรับได้จากภายนอกผ่าน Command UART โดยไม่ต้อง Flash ใหม่ทุกครั้ง
6. พัฒนาเว็บแอปพลิเคชันโดยใช้ Web Serial API เพื่อรับข้อมูลจาก STM32 โดยตรงผ่าน USB โดยมีฟังก์ชันหลักดังนี้



รูปที่ 6 แสดงหน้าเว็บที่ใช้สำหรับเก็บข้อมูล

6.1 Kalman Q/R Parameters (แถบบนสุด)

เว็บรับค่า Q/R ได้ 5 ตัว ได้แก่ R_tune, Q_pos, Q_vel, Q_cur, Q_tau แล้วกด Apply to STM32 เพื่อส่งค่าไปอัปเดต KF บน STM32 โดยตรงโดยไม่ต้อง Flash ใหม่ ทำให้ทดลองเปลี่ยน Q/R ได้เร็วและบันทึกผลแต่ละชุดได้ทันที

6.2 ข้อมูลที่แสดงในเว็บ

กราฟในเว็บ	State ที่ตรวจสอบ	วิธีเทียบ
Velocity — KF vs Finite Difference	ω (state ที่ 2)	ω_{FD} Backward
Current — KF vs Steady-state Est.	i (state ที่ 3)	$i_{est} = \frac{(V_s - K_e \cdot \omega)}{R_m}$
Disturbance Torque — KF vs Newton Est.	m_L (state ที่ 4)	Newton eq.
Innovation (pos_raw – pos_est)	ตรวจสอบ KF health	ควรอยู่ใน ±0.5°
Position — raw vs est(KF) vs model	θ (state ที่ 1)	วัดโดยตรง

ตารางที่ 15 แสดงข้อมูลของกราฟที่แสดงผลในเว็บ

Loop	ความถี่	เหตุผล
Kalman Update + Encoder	1 kHz	ต้องตรงกับ Ts ที่ใช้ discretize Ad,Bd (1ms)
Inner Velocity Loop	1 kHz	loop เร็วสุด — ตอบสนอง dynamics มอเตอร์
Outer Position Loop	100 Hz	ช้ากว่า inner 10× (cascade standard)
Trajectory Loop	1 kHz	gen ref position/velocity ต่อเนื่อง
Data Streaming (UART)	100 Hz	พอสำหรับ visualization ไม่บล็อก ISR

ตารางที่ 16 แสดงความถี่ของแต่ละ loop

1) เหตุผลที่เลือก Kalman update = 1 kHz

- Ad, Bd ถูกคำนวณที่ Ts = 1 ms ถ้า update rate ไม่ตรง Ts โมเดลจะผิด (state propagation ผิด)
- electrical pole ของมอเตอร์ L/R ≈ 0.6 ms ต้อง sample ≥ 1 kHz เพื่อจับ current dynamics (Nyquist)

2) ผลกระทบหากเลือกความถี่ไม่เหมาะสม

- ถ้าช้าเกินไป (< 1 kHz) discretization mismatch, มอง current dynamics ไม่ทัน, เกิด aliasing

- ถ้าเร็วเกินไป FD ขยาย quantization noise มากขึ้น (Δ/Ts ใหญ่), CPU overload

3) Interrupt Strategy & Real-time Constraint
   - TIM7 ⟶ hardware timer interrupt 1 kHz (priority สูงสุด) ⟶ รัน Encoder read + KF Predict + KF Update + control + streaming
   - UART RX ⟶ interrupt-driven (priority ต่ำกว่า) ⟶ parse คำสั่งจูน Q/R โดยไม่บล็อก control loop
   - Real-time constraint งานทั้งหมดใน ISR 1 kHz ต้องเสร็จภายใน 1 ms KF เป็น 4×4 matrix ops (~3 × 64 multiply-add) + update ~30 ops ⟶ ใช้เวลา << 1ms บน Cortex-M4F @ 170 MHz
   - ใช้ FPU (hardware float, -mfpu=fpv4-sp-d16 -mfloat-abi=hard) matrix ops เป็น float32 คำนวณเร็วใน hardware ไม่ต้อง software emulation
   - Streaming แยกออกทุก 10 ticks (100 Hz) เพื่อไม่ให้ UART transmit บล็อก 1 kHz loop

6.3 Innovation Plot (ส่วนล่างสุด)

$$Innovation = z[k] - H \cdot \hat{x}^{-}[k]$$

คือความแตกต่างระหว่าง Measurement จริงกับที่ KF ทำนาย ถ้า KF ทำงานถูกต้อง Innovation ควรมีลักษณะเป็น White Noise รอบค่าศูนย์ และอยู่ภายใน Bound ±0.5° ที่กำหนด โดยที่ค่า Innovation อยู่ใน ±0.5° ตลอด จะถือว่า KF ทำงานปกติ Q/R เหมาะสม หาก Innovation ออกนอก Bound บ่อย คือ Q ต่ำเกินไป KF ไม่ responsive พอ และถ้า Innovation มี Pattern ไม่ random คือ Model ผิด หรือ R ต่ำเกินไป

6.4 ขั้นตอนการใช้งานเว็บ

1) กำหนดค่า Q/R ชุดที่ต้องการในช่องด้านบน กด Apply to STM32
2) กด START เพื่อเริ่มรับข้อมูล จากนั้นกด Record เพื่อเริ่ม Log
3) สั่งให้ Motor เคลื่อนที่ตาม Trajectory มาตรฐานเดิมทุกรอบ (เพื่อให้เปรียบเทียบได้ Fair)
4) สังเกต Innovation Plot ขณะทดลอง ถ้าออกนอก Bound ให้ปรับ Q/R แล้วทำซ้ำ
   - กำหนด R จาก Quantization Noise และตรวจซ้ำด้วย static variance แล้วยึดค่า R ไว้

- ออกแบบ Trajectory ที่มีครบทุก State (Persistent Excitation) ผสม chirp/multi-sine + step ทั้งสองทิศ + ramp ความเร็วคงที่ + ช่วงเปลี่ยนภาระ และใช้ trajectory เดียวกันกับทุกการตั้งค่า
- เปลี่ยนค่า Q ไปเรื่อย ๆ ตามตาราง โดยปรับแบบ joint/coordinate-descent เพราะพฤติกรรมขึ้นกับอัตราส่วน Q/R จากนั้นคำนวณ NIS เฉลี่ย และ Whiteness ของ Innovation สำหรับแต่ละค่า
- เก็บเฉพาะค่าที่ NIS อยู่ในแถบ Chi-square และ Innovation ผ่าน Whiteness test คือ Consistent
- ในกลุ่มที่ Consistent เลือกค่าที่ให้ Smoothness ดีที่สุดภายใต้ Lag ที่ยอมรับได้ (ตามข้อกำหนดของระบบควบคุม)
- Validate ค่าที่เลือกบน Trajectory ชุดใหม่ (hold-out) ต้องยัง Consistent อยู่ เพื่อกันการ overfit

5) เมื่อครบ กด Export CSV บันทึกพร้อม Label ชุด Q/R
6) ทำซ้ำข้อ 1–5 สำหรับ Q/R ทุกชุดที่ต้องการเปรียบเทียบ

7. บันทึกผลการทดลองที่ได้ เพื่อนำไปหาค่า Matrix Q และ R ที่เหมาะสมในการใช้กับ Kalman Filter
8. นำค่าที่เก็บได้จากการทดลองมาเปรียบเทียบ และหาค่าความคลาดเคลื่อนจาก Raw data ที่เก็บได้

การทดลองที่ 2 เปรียบเทียบค่า state ที่คำนวณด้วยวิธีอื่น กับ state ที่ Kalman Filter estimate ขึ้นมาให้

1. คำนวณ $\boldsymbol{\omega}$ เพื่อเอาไป plot เทียบกับ KF output

   สมการ Backward Difference

$$\omega_{FD}[k] = \frac{\theta[k] - \theta[k-1]}{T_s}$$

   ต้องการให้ smooth ขึ้น ใช้ Central Difference

$$\omega_{FD}[k] = \frac{\theta[k+1] - \theta[k-1]}{2T_s}$$

2. $\boldsymbol{i}$ (กระแสไฟฟ้า) ประมาณจากสมการวงจร Steady-state

$$i_{est}[k] = \frac{V_s[k] - K_e \cdot \omega_{FD}[k]}{R_m}$$

ใช้ค่าจากระบบที่มีอยู่แล้วทั้งหมด คือ V_s ที่ STM32 ส่งออก + ω_{FD} ที่คำนวณจาก Encoder + K_e และ R_m จาก Lab 1 แล้วเอา i_{est} ไปเทียบกับ i_{KF}

3. m_L (Load Torque) ประมาณจากสมการ Newton

$$m_{L,est}[k] = K_m \cdot i_{est}[k] - J \cdot \alpha[k] - B \cdot \omega_{FD}[k]$$

โดย $\alpha[k] = \frac{\omega_{FD}[k] - \omega_{FD}[k-1]}{T_s}$ คือความเร่งเชิงมุมจาก Finite Difference อีกชั้น แล้วเทียบกับ mL_{KF} (state ที่ 3)

ผลการทดลอง

1) R Sweep (Measurement Noise)

ล็อก Q = diag(1e-8, 1e-4, 1e-3, 1e-1) แล้วเปลี่ยนค่า R ไปเรื่อย ๆ

R	mean(NIS)	white	สรุป
4.90229e-8 (ทฤษฎี)	0.0001	✗	เล็กเกินไป
4.90229e-7	0.50	✓	ดีที่สุด
4.90229e-6	0.049	✓	ใหญ่ไป
4.90229e-5	0.005	✗	ใหญ่ไป

ตารางที่ 17 แสดงค่าที่ได้จากการทดลอง R Sweep

R = 4.90229×10^{-7} ให้ NIS ใกล้ 1 ที่สุด (0.50) และเป็น white noise ค่านี้สูงกว่าทฤษฎี quantization noise ($\Delta^2/12 = 4.90229 \times 10^{-8}$) ประมาณ 10 เท่า เพราะ encoder จริงมี noise จากการสั่นสะเทือน (mechanical vibration, bearing play) มากกว่าแค่ quantization

2) Q Sweep (ล็อก R = 4.90229e-7)

เปลี่ยนค่าแต่ละ diagonal ของ Q ทีละตัว เลือกค่าที่ NIS ใกล้ 1 และ white

ตารางที่ 18 แสดงค่าที่ได้จากการทดลอง Q Sweep

พารามิเตอร์	ค่าที่เลือก	mean(NIS)	white
$Q_{pos}\ (\theta)$	1e-10	0.53	—
$Q_{vel}\ (\omega)$	1e-4	0.64	✓
$Q_{cur}\ (i)$	1e-4	0.49	✓
$Q_{tau}\ (m_L)$	1e-2	0.51	✓

- Q_vel มีผลต่อ NIS ชัดเจน 1e-4 ให้ NIS สูงสุด (0.64)
- Q_cur แทบไม่มีผล (NIS ~0.49 ทุกค่า) เพราะ current ไม่มี measurement ตรง จูนด้วย NIS ไม่ได้ ใช้ค่า physical แทน
- NIS plateau ที่ ~0.5 filter conservative เล็กน้อย (under-confident 2×) แต่ปลอดภัย ไม่ diverge

3) ผลการทดสอบความสอดคล้อง (NIS / Whiteness)

สำหรับแต่ละการตั้งค่า บันทึก NIS เฉลี่ย, แถบ Chi-square, ผลผ่าน/ไม่ผ่าน Whiteness, Smoothness และ Lag

การตั้งค่า (Q, R)	NIS เฉลี่ย	Whiteness	Consistent	สรุปพฤติกรรมของ Filter
R Sweep: R = 4.90229e-8 (ค่าทฤษฎี)	0.0001	ไม่ผ่าน	✗	R เล็กเกินไป
R Sweep: R = 4.90229e-7	0.50	ผ่าน	✓	ดีที่สุด (NIS ใกล้ 1 ที่สุด)
R Sweep: R = 4.90229e-6	0.049	ผ่าน	✗	R ใหญ่เกินไป
R Sweep: R = 4.90229e-5	0.005	ไม่ผ่าน	✗	R ใหญ่เกินไป
Q Sweep: Q_vel = 1e-4	0.64	ผ่าน	✓	มีผลต่อ NIS ชัดเจนที่สุด
Q Sweep: Q_tau = 1e-2	0.51	ผ่าน	✓	ให้ผลเป็น White noise

ค่า Q และ R ที่เลือกใช้ คือ

- R: 4.90229e-7
- Q_{pos}: 1e-10
- Q_{vel}: 1e-4
- Q_{cur}: 1e-4
- Q_{tau}: 1e-2

4) สถิติเปรียบเทียบ KF vs Finite Difference

คำนวณ ω_{FD} จาก θ ที่ log มา (Backward และ Central) แล้วเปรียบเทียบเชิงสถิติกับ ω_{KF}

1. ค่าความคลาดเคลื่อนรากที่สองเฉลี่ย (Root Mean Square Error: RMS)

$$\text{RMS} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(\omega_{KF}[k] - \omega_{FD}[k])^2}$$

2. สัมประสิทธิ์สหสัมพันธ์ (Correlation Coefficient: r)

$$r = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2\sum(y-\bar{y})^2}}$$

3. ค่าความคลาดเคลื่อนเฉลี่ย (Mean Error: $\bar{e}$)

$$\bar{e} = \frac{1}{N}\sum_{k=1}^{N}(\omega_{KF}[k] - \omega_{FD}[k])$$

เกณฑ์: ถ้า r > 0.95 ถือว่า KF track ได้สมเหตุสมผล / KF ควรมี Std ต่ำกว่า FD (smoother) / Mean Error ใกล้ 0 แสดงว่าไม่มี Systematic Bias

ตัวชี้วัด	Kalman Filter	Finite Difference
RMS Error (rad/s)	6.45	-
Std Dev (rad/s)	38.6	38.0
Mean Error (rad/s)	0.26	-
Pearson Correlation	0.986	-
Max Deviation (rad/s)	17.3	-

ตารางที่ 19 แสดงค่า error ที่วัดได้จาก KF vs Finite Difference

เนื่องจากระบบไม่มี Ground Truth ของ State จริง การเลือก Q, R โดยใช้เกณฑ์ ทำให้ค่าประมาณ ตรงกับค่าที่วัดได้มากที่สุด (RMSE ต่ำสุด) เป็นเกณฑ์ที่ผิด เพราะมันให้ความสำคัญกับ Filter ที่แทบไม่กรอง noise เมื่อ R เข้าใกล้ 0 ค่าประมาณจะลอกค่าวัดที่มี noise มาทั้งหมด แนวทางที่ถูกต้องคือจูนให้ Filter สอดคล้องกับสมมติฐาน ซึ่งตรวจสอบได้จาก Innovation เพียงอย่างเดียว โดยไม่ต้องมีค่าจริง

Innovation และ Covariance ที่ Filter ทำนายไว้

$$\nu(k) = y(k) - C\widehat{x^-}(k)$$

$$S(k) = CP^-(k)C^T + R$$

Filter ที่จูนถูกต้อง Innovation $\nu(k)$ จะต้อง มีค่าเฉลี่ยเป็นศูนย์ เป็น White Noise และมี Covariance เท่ากับ $S(k)$ ตรวจสอบด้วย 2 การทดสอบ

1. NIS (Normalized Innovation Squared)

$$\text{NIS}(k) = \nu^T(k)S^{-1}(k)\nu(k)$$

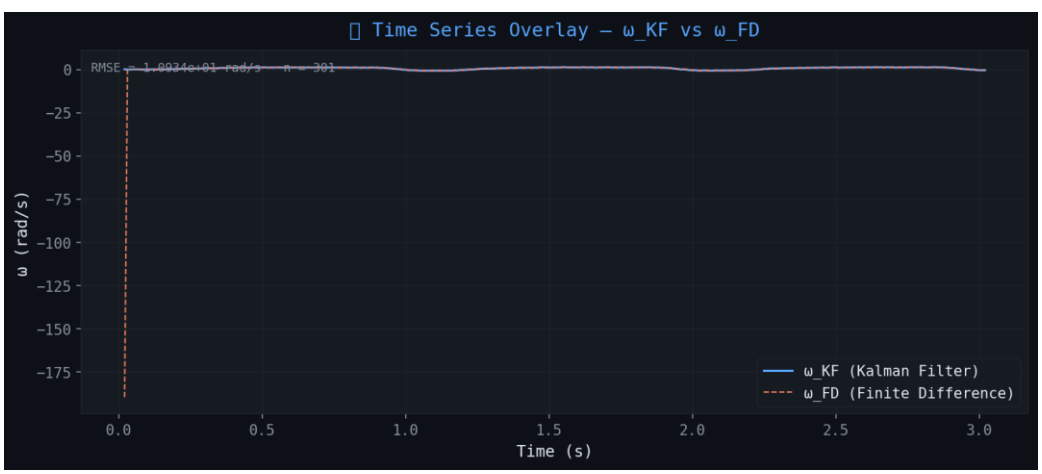
NIS เฉลี่ยตามเวลาควรอยู่ในแถบความเชื่อมั่น Chi-square ที่มีจุดศูนย์กลางที่ m (จำนวนช่องสัญญาณที่วัด) ถ้า Q เล็กเกินไป NIS จะสูงกว่า m (Filter มั่นใจเกินจริง) ถ้า Q ใหญ่เกินไป NIS จะต่ำกว่า m (Filter เฉื่อย)

$$\frac{1}{N}\sum_{k=1}^{N} \text{NIS}(k) \in \left[\frac{\chi^2_{\alpha/2}(Nm)}{N}, \frac{\chi^2_{1-\alpha/2}(Nm)}{N}\right]$$

2. Whiteness การทดสอบ Autocorrelation (Anderson Test)

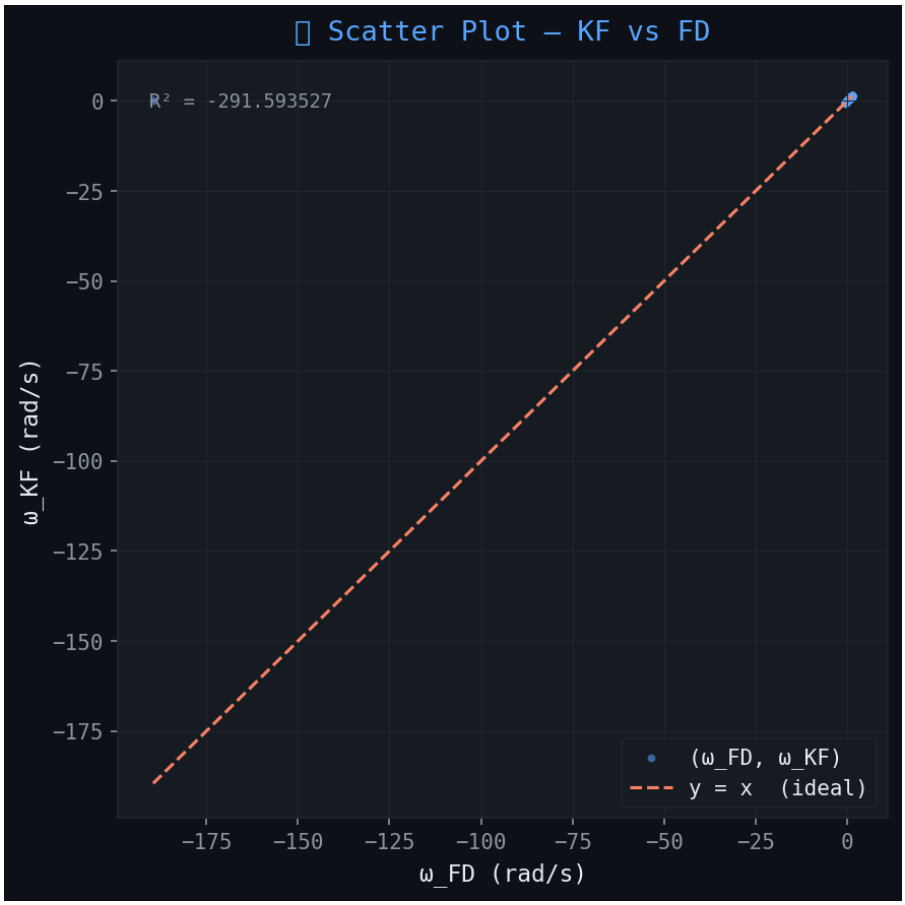
Autocorrelation ของ Innovation ที่ lag ≠ 0 ต้องอยู่ในแถบ ±1.96/√N หากยังมี correlation เหลืออยู่ แปลว่าโมเดลหรือค่า Q ยังไม่ถูกต้อง

$$\hat{\rho}(\tau) = \frac{\sum_k \nu(k)\nu(k+\tau)}{\sum_k \nu(k)^2}, \qquad |\hat{\rho}(\tau)| \le \frac{1.96}{\sqrt{N}}$$



รูปที่ 7 กราฟ Time Series Overlay

รูปที่ 8 กราฟ Residual Plot (Error vs Time)



รูปที่ 9 กราฟ Scatter Plot KF vs FD

สรุปผลการทดลอง

การออกแบบและจูน Discrete Kalman Filter (KF) แบบ 4-state สำหรับ DC Motor สามารถหาค่า Q และ R ที่ทำให้ Filter มี Consistent โดยเลือกใช้ค่า R ที่ 4.90229e-7 และเมทริกซ์ Q ตามลำดับคือ diag(1e-10, 1e-4, 1e-4, 1e-2)

การตรวจสอบด้วยวิธี Innovation / NIS Consistency Test ยืนยันว่าค่าที่เลือกทำให้ Innovation มีลักษณะเป็น White noise และมีค่า NIS เฉลี่ยที่ประมาณ 0.50 ถึง 0.64 ซึ่งบ่งชี้ว่าโครงสร้างแบบจำลอง (Ad, Bd) ถูกต้อง ในส่วนของการเปรียบเทียบความเร็วเชิงมุม (ω) ระหว่าง KF กับวิธี Finite Difference (FD) พบว่ามีความสอดคล้องกันสูงมาก โดยมีค่า Pearson Correlation r = 0.986 และมี Mean Error เพียง 0.26 °/s ซึ่งยืนยันว่า KF ประมาณค่าได้สมเหตุสมผลและไม่มี Systematic bias

อภิปรายผล

การประมาณค่าความเร็วเชิงมุม (ω) ของตัวกรอง Kalman Filter (KF) แสดงให้เห็นถึงประสิทธิภาพในการติดตามสัญญาณที่ดีเยี่ยมเมื่อเปรียบเทียบกับวิธี Finite Difference (FD) โดยผลการวิเคราะห์เชิงสถิติพบว่ามีค่าสัมประสิทธิ์สหสัมพันธ์ของเพียร์สัน (Pearson Correlation Coefficient: r) สูงถึง 0.986 ซึ่งสูงกว่าเกณฑ์มาตรฐานที่ตั้งไว้ที่ 0.95 อย่างมีนัยสำคัญ นอกจากนี้ ตัวกรองยังมีค่าความคลาดเคลื่อนรากที่สองเฉลี่ย (Root Mean Square Error: RMS Error) เพียง 6.45 °/s และมีค่าความคลาดเคลื่อนเฉลี่ย (Mean Error หรือ bias) ที่เข้าใกล้ศูนย์เพียง 0.26 °/s ซึ่งเป็นสิ่งยืนยันว่าการประมาณค่าสถานะมีความสมเหตุสมผลสูง และไม่มี Systematic Bias เกิดขึ้นในกระบวนการคำนวณ

เมื่อพิจารณาในแง่ความเรียบของสัญญาณ พบว่าตัวกรอง Kalman Filter ให้ผลลัพธ์ที่มีความ Smooth กว่าการประมาณค่าด้วยวิธี Finite Difference อย่างเห็นได้ชัด เนื่องจากในเชิงทฤษฎี การหาอนุพันธ์อันดับหนึ่งด้วยวิธี Finite Difference จะขยายสัญญาณรบกวนจาก Quantization Noise ของ Encoder ที่เกิดจากความละเอียดเชิงมุมที่จำกัดผ่านการหารด้วยคาบเวลาการสุ่ม (T_s) ส่งผลให้สัญญาณความเร็วที่ได้จากวิธี Finite Difference มีความผันผวนและมีสัญญาณรบกวนปะปนอยู่มาก ในทางตรงกันข้าม อัลกอริทึม Kalman Filter จะใช้ Process Model ในการ Prediction Step แล้วจึงนำค่าการวัดตำแหน่งเชิงมุมจากเซนเซอร์จริงมาทำการ Correction Step จึงทำให้สามารถกรองสัญญาณรบกวนได้อย่างมีประสิทธิภาพและให้ผลลัพธ์ที่ราบเรียบกว่ามาก

ในช่วง Transient Period ระบบอาจเกิด Phase Lag ตามหลังค่าความเร็วจากการประมาณด้วยวิธี Finite Difference ได้เล็กน้อย ซึ่งมีความสอดคล้องกับผลการทดลองที่ค่า NIS เฉลี่ยเกิดการคงตัวอยู่ที่ประมาณ 0.5 แทนที่จะเป็น 1.0 พอดี บ่งชี้ว่า Filter มีความมั่นใจในแบบจำลองน้อยกว่าความเป็นจริงประมาณ 2 เท่า แต่พฤติกรรมในลักษณะนี้ก็ถือว่ายอมรับได้ และส่งผลดีในแง่ของความเสถียร และความปลอดภัยของระบบควบคุม โดยช่วยป้องกันไม่ให้ตัวกรองเกิด Diverge ของสัญญาณประมาณค่าสถานะ

สำหรับความสมเหตุสมผลในภาพรวมของค่าสถานะความเร็วเชิงมุมที่ Filter ประมาณขึ้นมา สามารถยืนยันความถูกต้องได้จากการตรวจสอบค่าสัญญาณ Innovation ที่แสดงพฤติกรรมเป็น White Noise โดยไม่มี Autocorrelation ซึ่งเป็นหลักฐานสำคัญที่ชี้ว่าโครงสร้างแบบจำลองสเปซสถานะในโดเมนไม่ต่อเนื่อง (A_d, B_d) ที่ได้จากการแปลงแบบจำลองทางฟิสิกส์ของมอเตอร์นั้นมีความถูกต้องแม่นยำ นอกจากนี้ การที่ค่าความแปรปรวนสัญญาณรบกวนจากการวัดจริง ($R = 4.90229 \times 10^{-7}$) มีค่าสูงกว่าค่าทางทฤษฎีของสัญญาณรบกวนดิจิทัล ($R_{theoretical} = 4.90229 \times 10^{-8}$) ประมาณ 10 เท่า ยังช่วยสะท้อนปัจจัยทางกายภาพจริงของระบบฮาร์ดแวร์ที่มีสัญญาณรบกวนจาก Mechanical Vibration และระยะคลอนของ Bearing เพิ่มเติมเข้ามานอกเหนือจากขีดจำกัดของเซนเซอร์ ทั้งนี้ แม้ว่าวิธี Finite Difference จะไม่ใช่ Ground Truth เนื่องจากเป็นการคำนวณจากเซนเซอร์ที่มีสัญญาณรบกวนร่วมด้วยเช่นกัน แต่การนำมาใช้ในลักษณะ Sanity Check ควบคู่ไปกับเกณฑ์ NIS ก็ถือเป็นแนวทางที่ถูกต้อง และมีความน่าเชื่อถือสูงในการรับรองความถูกต้องของสถานะความเร็วเชิงมุม

ข้อเสนอแนะ

- ควรเก็บค่าของ Motor Parameter โดยอ้างอิงรูปแบบระบบจากระบบที่จะใช้จริง โดยที่ดูถึงความเป็นไปได้ของค่า Parameter ที่ทำการ Estimate ออกมาว่ามีความสมเหตุสมผลหรือไม่
- ควรทดสอบที่ความเร็วหลายระดับ (เช่น 1, 2, 3 rad/s) เพื่อยืนยันว่า KF ทำงานได้ดีในทุก Operating Point
- หาก m_L หรือ i ต้องการตรวจสอบเพิ่มเติม ต้องใช้ Sensor เพิ่มเติม คือ

  $m_L \longrightarrow$ ประมาณจาก Torque Sensor หรือ $Current \times Km - J \times \alpha$

  $i \longrightarrow$ ประมาณจากสมการ $i_{est} = \frac{(Vs - Ke \times \omega)}{Rm}$ (Steady-state approx)

อ้างอิง

1. Welch, G. & Bishop, G. - An Introduction to the Kalman Filter (UNC Chapel Hill)

2. Kalman, R.E. (1960) - A New Approach to Linear Filtering and Prediction Problems

3. Wikipedia - Kalman Filter

4. STM32G474RE Reference Manual - STMicroelectronics

5. MATLAB Documentation - Discrete Kalman Filter Design

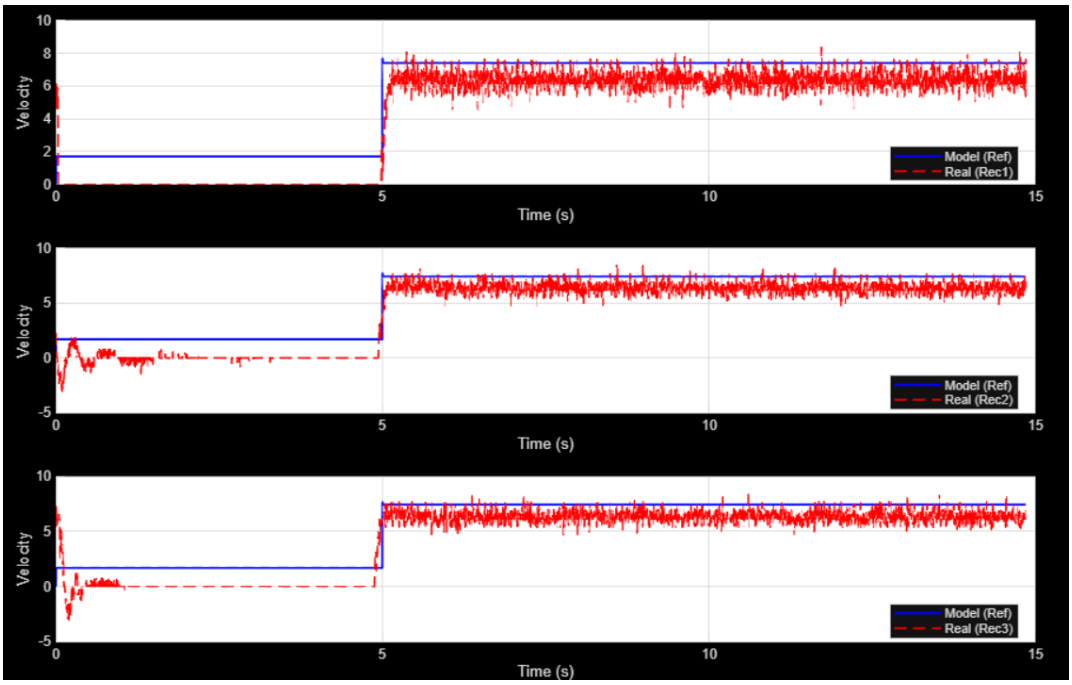
6. scipy.signal.cont2discrete - Zero-Order Hold Discretization

7. Simon, D. (2006) - Optimal State Estimation: Kalman, H∞, and Nonlinear Approaches

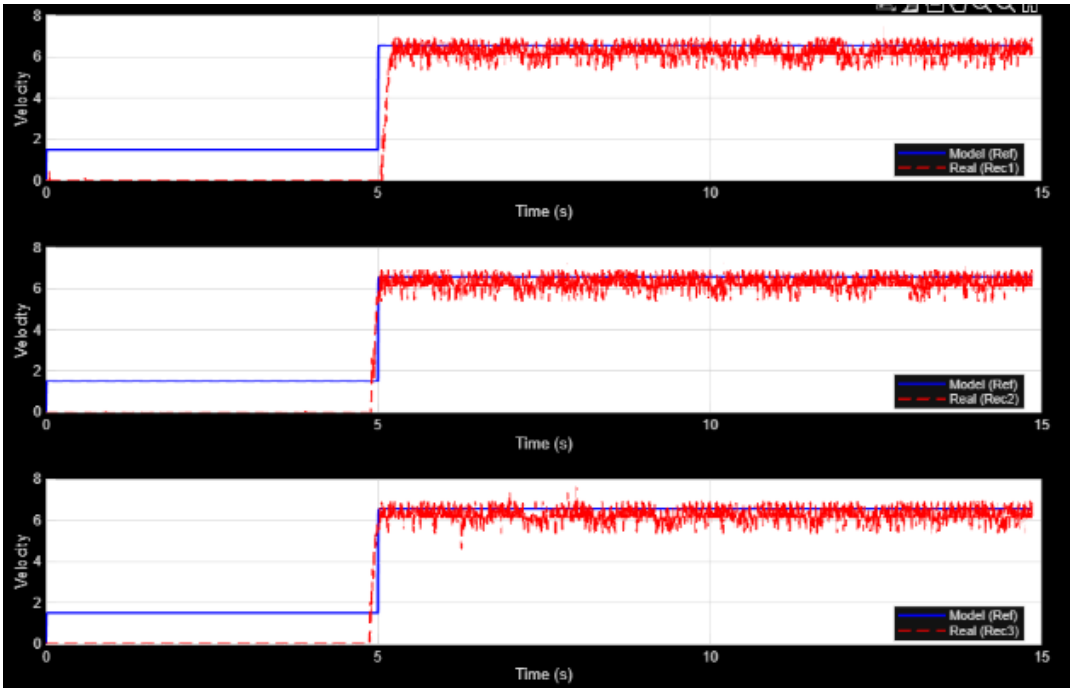
8. Faragher, R. (2012) - Understanding the Basis of the Kalman Filter (IEEE Signal Processing)

9. Brown, R.G. & Hwang, P.Y.C. - Introduction to Random Signals and Applied Kalman Filtering

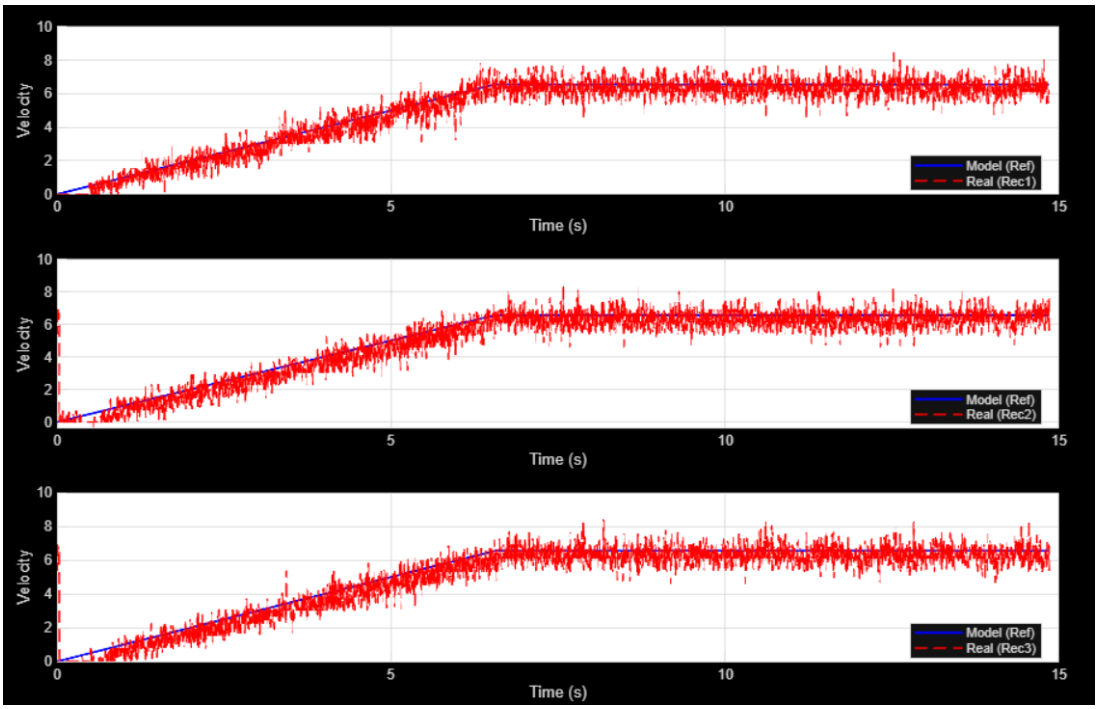
10. https://kalman-filter.com/estimating-the-state-of-a-dc-motor/



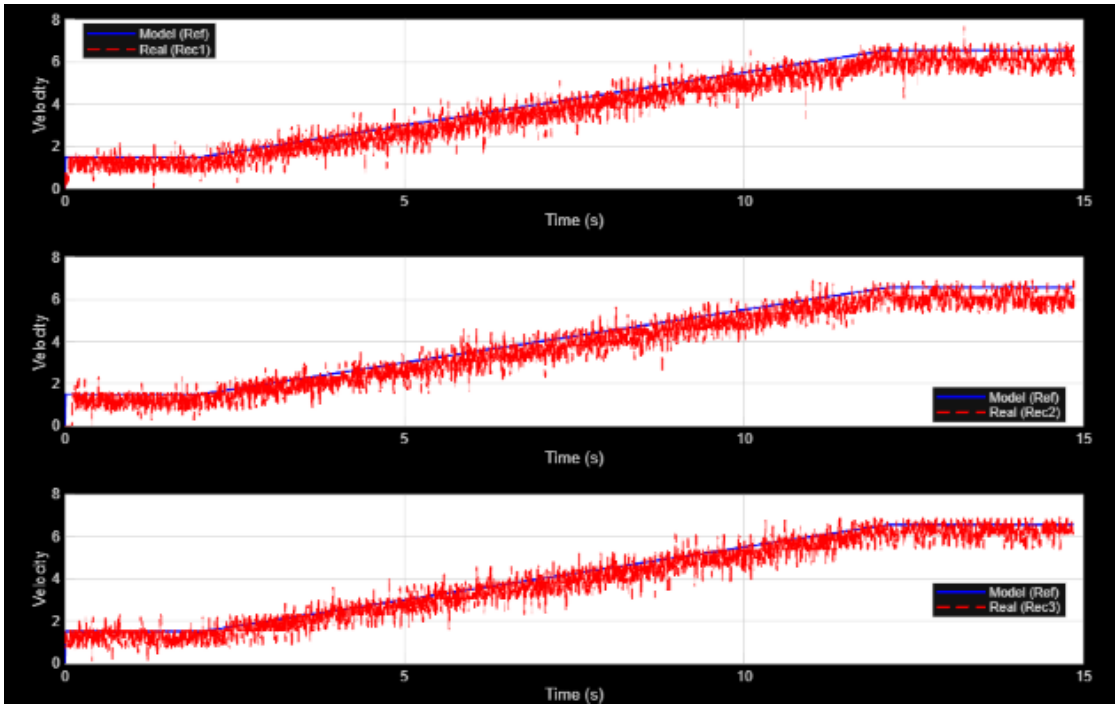
รูปที่ 10 แสดงค่า Cross Validation ด้วยสัญญาณ Step (Without Gripper)



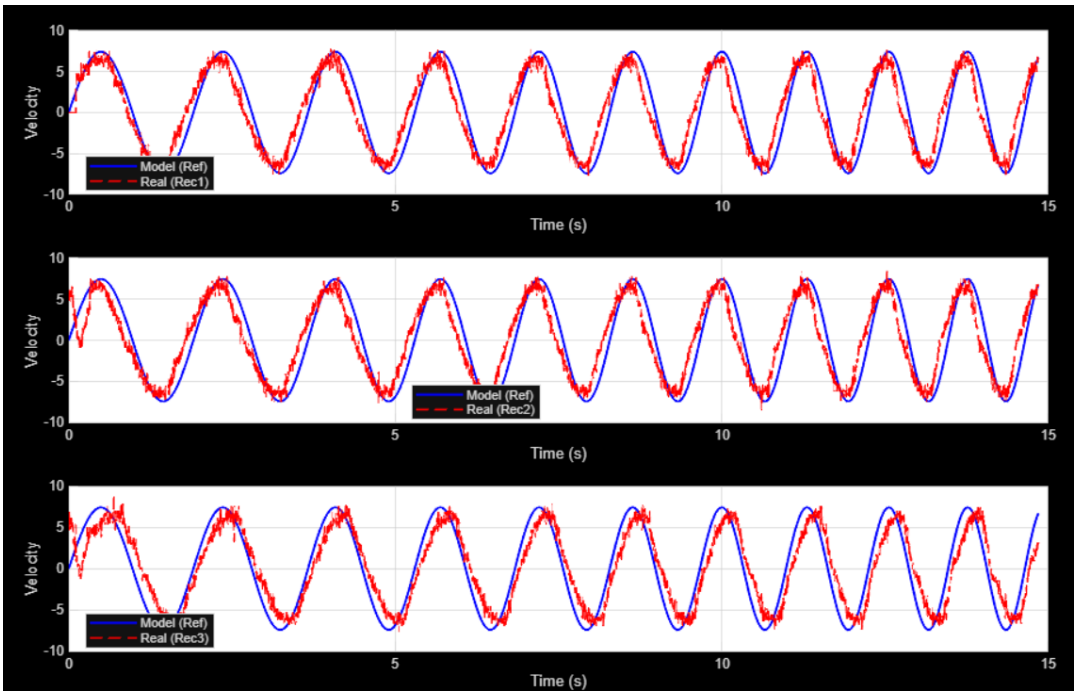
รูปที่ 11 แสดงค่า Cross Validation ด้วยสัญญาณ Step (With Gripper)



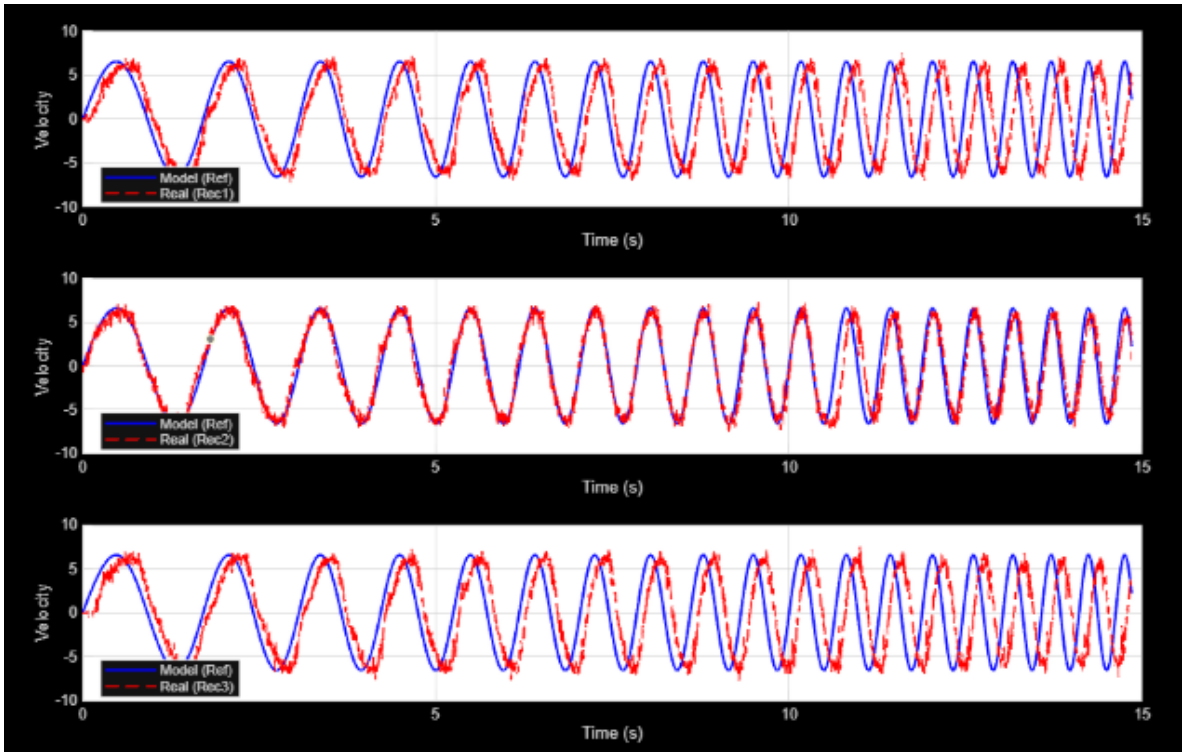
รูปที่ 12 แสดงค่า Cross Validation ด้วยสัญญาณ Ramp (Without Gripper)



รูปที่ 13 แสดงค่า Cross Validation ด้วยสัญญาณ Ramp (With Gripper)



รูปที่ 14 แสดงค่า Cross Validation ด้วยสัญญาณ Chirp (Without Gripper)



รูปที่ 15 แสดงค่า Cross Validation ด้วยสัญญาณ Chirp (With Gripper)

ตอบ Criteria

LAB 1

1. สามารถออกแบบการทดลองและหา Motor Parameter ได้อย่างถูกต้องและน่าเชื่อถือ

   จากการทดลองที่ 1: Motor Parameter Identification Using Experimental Data" รายงานแสดงขั้นตอนการทดลองโดยใช้สัญญาณ Step, Ramp และ Chirp Input ทำซ้ำ 3 ครั้งต่อสัญญาณ ใน 2 กรณีคือมีและไม่มี Gripper ผลการทดลองแสดงใน ตารางที่ 1–6 (หน้า 4–6)

2. สามารถวิเคราะห์ และคำนวณที่มาของระบบหุ่นยนต์ ออกแบบการทดลอง และสามารถประมาณค่า Damp ของหุ่นยนต์ได้

   จากเนื้อหาตารางที่ 1–6 (หน้า 4–6) แสดงค่า Motor B (Damping Coefficient) ที่ประมาณได้จากทุกสัญญาณกระตุ้น พร้อม cross-check ระหว่างกรณีมีและไม่มี Gripper

3. สามารถวิเคราะห์ และคำนวณที่มาของระบบหุ่นยนต์ ออกแบบการทดลองและสามารถประมาณค่า Inertia รอบแกนหมุนของหุ่นยนต์ได้

   จากเนื้อหาตารางที่ 1–6 (หน้า 4–6) แสดงค่า Equivalent Inertia (J) ที่ประมาณได้จากทุกสัญญาณกระตุ้น

4. สามารถเปรียบเทียบและเลือก Motor Parameter ที่เหมาะสมที่สุดโดยใช้ หลักการทางสถิติที่เหมาะสม (เช่น Mean, Std, Confidence Interval, Residual Analysis) พร้อมแสดง Cross Validation และนำเสนอผลด้วย Visualization ที่เหมาะสมกับลักษณะข้อมูล

   จากเนื้อหาส่วนสรุปผลการทดลอง (หน้า 6) และอภิปรายผล (หน้า 6–7) รายงานใช้ค่าเฉลี่ย 3 ครั้งเป็นตัวแทนพารามิเตอร์ และระบุว่าจะนำไป Cross Validation ในขั้นตอนถัดไป / Cross Validation แสดงใน ตารางที่ 7–12 (หน้า 12–13, การทดลองที่ 2)

LAB 2

1. แสดงผลเปรียบเทียบกับกรณีไม่มี Feedforward ในแง่ของ Tracking Error โดยใช้ หลักการทางสถิติ (เช่น RMS Error, Mean Absolute Error) ในการประเมินผล

   จากเนื้อหาส่วน การทดลอง 2.1

2. สามารถออกแบบและ Implement Disturbance Compensation ได้ โดยอธิบายได้ว่าปริมาณใดที่เลือกชดเชย เหตุผลที่เลือก และวิธีการประมาณค่าปริมาณนั้น

จากเนื้อหาส่วน

3. สามารถออกแบบการทดลองให้เห็นผลกระทบจากการปรับค่า Kp, Ki, Kd ต่อพฤติกรรมของระบบได้ครบถ้วน

จากเนื้อหาส่วน

4. สามารถวิเคราะห์ อธิบาย และแก้ไขปัญหา Integration Windup ได้ พร้อมแสดงผลลัพธ์ก่อน/หลังการแก้ไข

จากเนื้อหาส่วน

5. สามารถออกแบบและ Implement Saturation ใน Controller Output ได้อย่างถูกต้อง พร้อมอธิบายเหตุผลในการกำหนดขอบเขต Saturation และวิเคราะห์ผลกระทบที่มีต่อ System Response และ Safety

จากเนื้อหาส่วน

6. สามารถออกแบบและ Implement Cascade Control โดยใช้ PID control บน STM32 ได้อย่างถูกต้อง

จากเนื้อหาส่วน

7. สามารถแสดงขั้นตอนการปรับ Parameter (Kp, Ki, Kd) ภายใต้ Cascade Control ได้

จากเนื้อหาส่วน

8. สามารถออกแบบและ Implement Trapezoidal Velocity Profile ได้อย่างถูกต้องเขียนเป็น Library ได้

จากเนื้อหาส่วน

9. สามารถออกแบบการเคลื่อนที่แบบ S-Curve บน STM32 ได้

จากเนื้อหาส่วน

10. สามารถอธิบายความแตกต่างจาก Trapezoidal ในแง่ของ Jerk และ Smoothness ของ Motion

จากเนื้อหาส่วน

11. สามารถเปรียบเทียบผลลัพธ์ของ Trapezoidal และ S-Curve ในเชิง Tracking Error, Control Effort

จากเนื้อหาส่วน

12. สามารถออกแบบและใช้งาน Reference Feedforward ได้อย่างถูกต้อง

จากเนื้อหาส่วน

LAB 3

1. สามารถเขียน State-Space Model ของ DC Motor โดยใช้ Inertial และ damping ของระบบ ใน Discrete Domain ได้อย่างถูกต้อง โดยมี อย่างน้อย 4 States พร้อมอธิบายความหมายทางกายภาพของแต่ละ State

   จากเนื้อหาส่วน Part 3: Kalman Filter Design การทดลองที่ 1 (หน้า 32–35) รายงานสร้าง State-Space Model 4 States (θ, ω, m_L, i) พร้อมอธิบายที่มาจาก Kirchhoff's Voltage Law และสมการกลศาสตร์

2. สามารถอธิบายเหตุผลในการเลือก State ทั้ง 4 ขึ้นไป และอธิบายได้ว่า State ที่เพิ่มขึ้นมานั้นมีบทบาทต่อประสิทธิภาพของ Filter อย่างไร

   จากเนื้อหาส่วนจุดประสงค์ข้อ 1 (หน้า 32) และสมมติฐาน (หน้า 32) รายงานอธิบาย State ที่ 4 (τ_d — Disturbance Torque) เป็น Random Walk Model เพื่อประมาณค่า Disturbance สำหรับ Compensation

3. สามารถเขียนโปรแกรม Discrete Kalman Filter (≥4 States) พร้อม Matrix Operation เป็น Library บน STM32G474RE ได้อย่างถูกต้อง

   จากเนื้อหาหน้า 32 รายงานระบุการ Implement Kalman Filter พร้อม Matrix Operation เป็น Library บน STM32G474RE

4. สามารถแสดงขั้นตอนการกำหนด Matrix Q และ R พร้อมอธิบายความหมายทางกายภาพของแต่ละ Element และแสดงผลกระทบของการปรับ Q/R ต่อ Trade-off ระหว่าง Responsiveness กับ Smoothness โดยใช้หลักการทางสถิติในการประเมิน

   จากเนื้อหาส่วน ผลการทดลอง R Sweep และ Q Sweep (หน้า 44–45) รายงานแสดงตารางที่ 17 ผล R Sweep และตารางที่ 18 ผล Q Sweep พร้อมใช้ NIS (Normalized Innovation Squared) และ Whiteness Test (Anderson Test) ในการประเมิน พร้อมอธิบายว่า R = 4.90229×10^{-7} ให้ NIS ใกล้ 1 ที่สุด

5. สามารถวิเคราะห์ว่า State ที่ไม่ได้วัดโดยตรงที่ Kalman Filter ประมาณขึ้นมานั้นมีความสมเหตุสมผลหรือไม่ โดยเปรียบเทียบกับค่าที่คำนวณด้วยวิธีอื่น และนำเสนอผลด้วย Visualization ที่เหมาะสม

จากตารางที่ 15 ข้อมูลของกราฟที่แสดงผลในเว็บ (หน้า 41) รายงานระบุการตรวจสอบ State ω (KF vs Finite Difference), i (KF vs Steady-state Est.), m_L(KF vs Newton Eq.) พร้อมกราฟในรูปที่ 7 (หน้า 47)

Integrate

1. สามารถออกแบบและนำเสนอ Block Diagram ของระบบโดยรวม ได้อย่างครบถ้วน โดย Diagram ต้องแสดง Trajectory Generator, Feedforward, PID (Cascade), Kalman Filter, Disturbance Compensation, Saturation และ Sensor Feedback พร้อมบุทิศทาง Signal Flow ได้อย่างชัดเจน

   จากรูปที่ 1 แสดง Cascade PID Control (หน้า 15) Block Diagram แสดงองค์ประกอบหลักครบ ได้แก่ Trajectory Generator (S-Curve), Position PID, Velocity PID, Plant Transfer Function, Reference Feedforward, Velocity Estimator (Kalman Filter), Saturation และ Encoder

2. สามารถกำหนดและอธิบาย Control Frequency ของแต่ละ Loop ได้อย่างสมเหตุสมผล (เช่น Trajectory Loop, Outer Position Loop, Inner Velocity Loop, Kalman Filter Update Rate) พร้อมอธิบายเหตุผลในการเลือกความถี่และผลกระทบหากเลือกความถี่ไม่เหมาะสม

   จากตารางที่ 16 แสดงความถี่ของแต่ละ Loop (หน้า 41) รายงานระบุ Kalman Update + Encoder = 1 kHz, Inner Velocity Loop = 1 kHz, Outer Position Loop = 100 Hz, Trajectory Loop = 1 kHz, Data Streaming = 100 Hz พร้อมอธิบายเหตุผล

3. สามารถสรุปและวิเคราะห์ผลการทดลองทั้งหมดได้อย่างน่าเชื่อถือ โดยใช้หลักการทางสถิติที่เหมาะสม ในการเปรียบเทียบผลระหว่าง Configuration ต่างๆ รวมถึงวิเคราะห์ข้อจำกัดของระบบและสาเหตุของความคลาดเคลื่อนที่เหลืออยู่

   จากเนื้อหาส่วนอภิปรายผลของแต่ละ Part — Part 1 (หน้า 6–7), Part 2 (หน้า 30), Part 3 (หน้า 50) มีการวิเคราะห์ข้อจำกัด และสาเหตุของความคลาดเคลื่อนในแต่ละ Part

4. สามารถเลือกกรูปแบบ Visualization ที่เหมาะสมกับข้อมูลแต่ละประเภท ระบุหน่วยใน SI ได้ครบถ้วน

จากเนื้อหาส่วน Part 3 รูปที่ 7 (หน้า 47) แสดงกราฟ Time Series Overlay ($\boldsymbol{\omega_{KF}}$ vs $\boldsymbol{\omega_{FD}}$) ระบุหน่วย rad/s บนแกน Y และ Time (s) บนแกน X ครบถ้วน