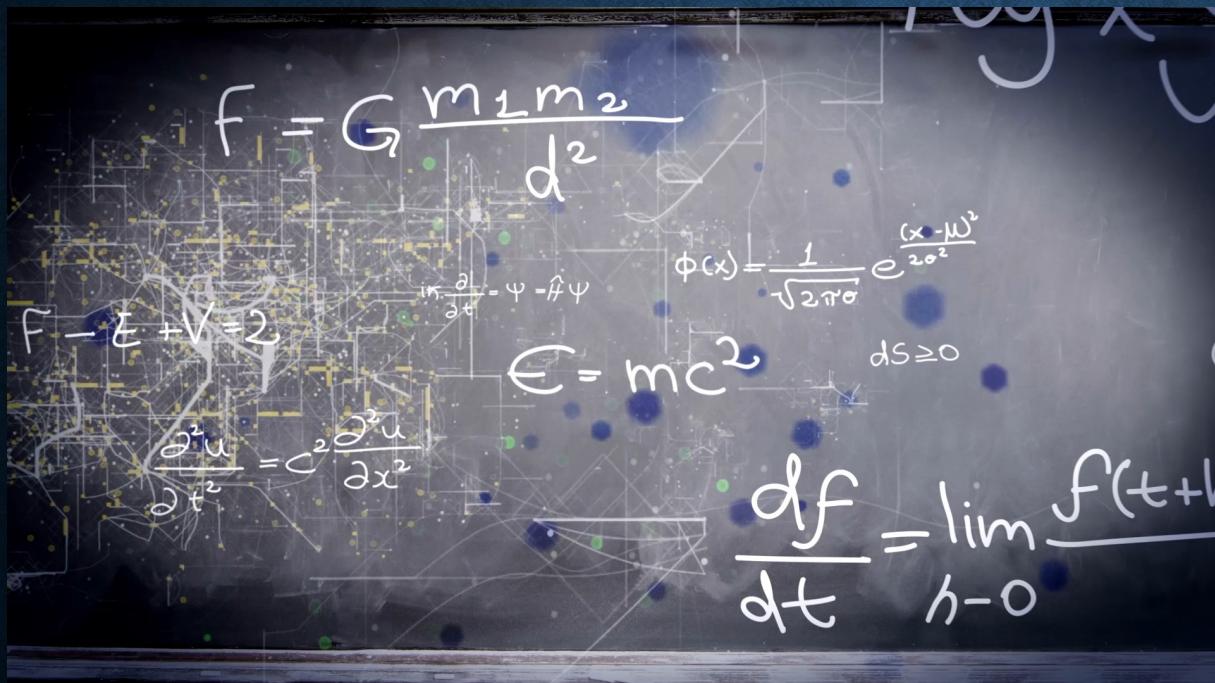


การโปรแกรมพลวัตแบบเชิงกำหนด (DETERMINISTIC DYNAMIC PROGRAMMING)



• 01208583 Robotics

Varodom Toochinda, Ph.D

Dept. of Mechanical Engineering

Kasetsart University

OUTLINE

- ประเภทของการโปรแกรมพลวัต
- หลักการของความหมายที่สุด
- ปัญหาแบบแนวอนจำกัด
- อัลกอริทึมการโปรแกรมพลวัต
- ปัญหาการควบคุมหมายที่สุดกับการโปรแกรมพลวัต
- การควบคุมแบบทำนายไมเดล
- การโปรแกรมพลวัตในระบบเวลาต่อเนื่อง

4.2 ประเภทของการโปรแกรมพลวัต

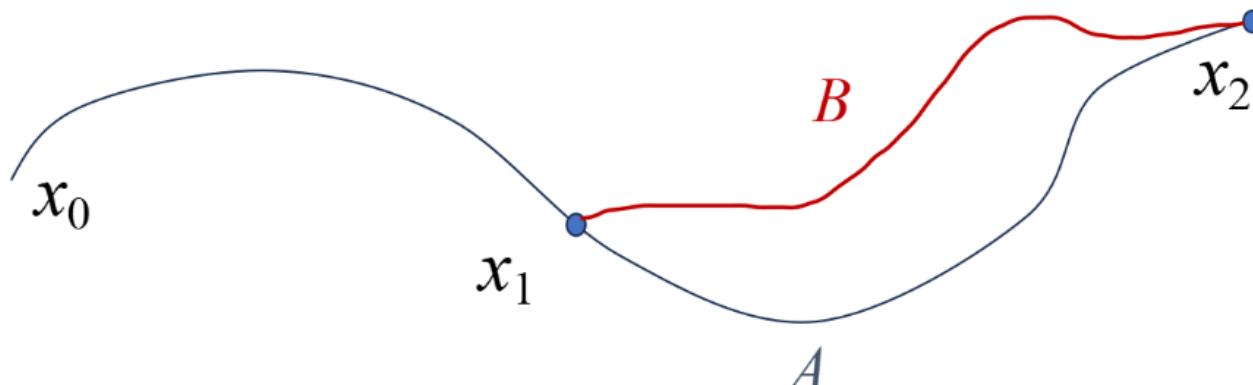
หัวใจสำคัญของปัญหา DP คือระบบพลวัตในเวลาดีสคริปท์ที่กำหนดลำดับของสถานะภายใต้อิทธิพลของตัวควบคุมหรือตัวกระทำ สามารถแยกได้เป็น 2 ประเภทหลักคือ แบบเชิงกำหนด (deterministic) เมื่อการพัฒนาของตัวแปรภายในเป็นแบบกำหนดได้แน่นอน คือได้เอาต์พุตหรือคำตอบเหมือนเดิมจากอินพุตเดียวกันทุกครั้ง หรือแบบสโตแคสติก (stochastic) เมื่อมีการรบกวนแบบสุ่มทำให้คำตอบแตกต่างกันไปในแต่ละครั้ง

นอกจากนี้อาจแบ่งประเภทของปัญหา DP ในอีกมิติหนึ่ง คือแบบแม่นตรง (exact) หรือแบบใช้การประมาณค่า (approximate) ประเด็นหลักที่แตกต่างคือแบบแม่นตรงจะต้องการโน้มเดลทางคณิตศาสตร์ ส่วนแบบการประมาณค่าจะใช้โครงข่ายประสาทเทียมหรือสถาปัตยกรรมอื่นในการลดมิติของระบบ และใช้วิธีการจำลองโน้มเดลบนคอมพิวเตอร์แทนโน้มเดลทางคณิตศาสตร์

หลักการของความหมายที่สุด

นโยบายหมายที่สุดต้องมีคุณสมบัติคือ “ไม่ว่าจะเลือกสถานะและการตัดสินใจเริ่มต้นอย่างไร การตัดสินใจครั้งต่อไปที่เหลือจะต้องยังคงเป็นนโยบายหมายที่สุดเสมอ เมื่อนับจากสถานะที่เกิดจากการตัดสินใจครั้งแรก”

จากรูปที่ 3.1 อธิบายได้ว่า สมมติว่า นโยบายในการเคลื่อนที่จากจุด x_0 ไปยังจุด x_2 โดยผ่านเส้นทาง (หรือแนววิถี) A คือนโยบายหมายที่สุดตามวัตถุประสงค์ที่กำหนด เมื่อพิจารณาปัญหาอย่างจากจุด x_1 ไปยังจุด x_2 เส้นทางที่หมายที่สุดก็ยังคงเป็นเส้นทางเดิมคือ A กล่าวคือจะไม่สามารถพบเส้นทางใหม่ B ที่หมายที่สุดกว่าเส้นทาง A ได้ เพราะมีชั้นนั้นในการหาค่าหมายที่สุดสำหรับปัญหาร่วม เราจึงเลือกเส้นทางผ่าน B แทน A



4.2.2 ปัญหาแบบแนวอนจำกัด

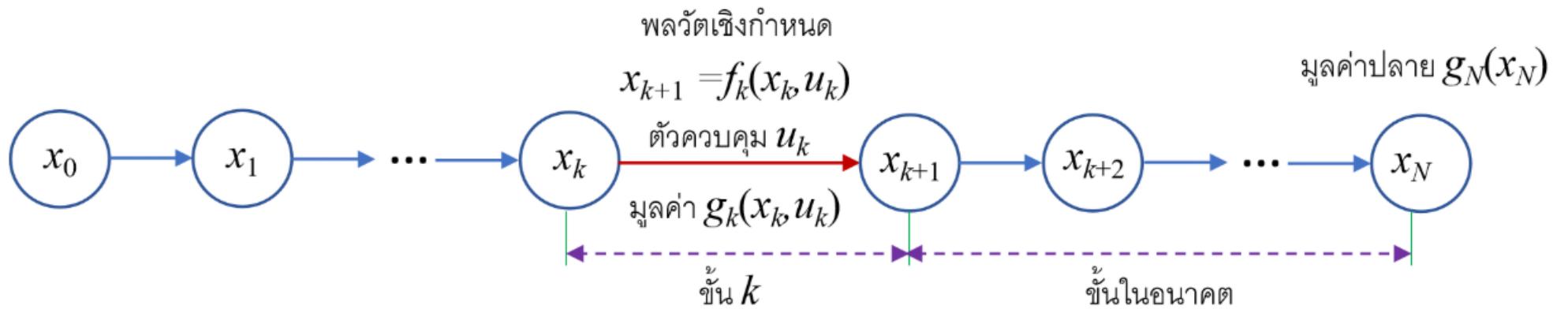
รูปแบบของปัญหาแบบแนวอนจำกัด (finite horizon) คือเมื่อระบบวิวัฒน์ไปตามขั้นเวลาจำกัด N ขั้น (stages) ดังแสดงในรูปที่ 4.1 สถานะและตัวควบคุมของระบบ ณ ขั้นเวลา k เขียนแทนด้วย x_k และ u_k ตามลำดับ โดยค่าของ u_k ณ ขั้นเวลา K ถูกเลือกจากเซต $U_k(x_k)$ ในระบบเชิงกำหนด ค่าของ x_{k+1} มิได้เป็นค่าสุ่ม แต่จะถูกกำหนดโดย x_k และ u_k สอดคล้องกับผลวัด

$$x_{k+1} = f_k(x_k, u_k), \quad k = 0, 1, \dots, N - 1 \quad (4.1)$$

นอกจากนี้จะเห็นว่าแต่ละขั้นในรูปที่ 4.1 จะมีฟังก์ชันมูลค่า $g_k(x_k, u_k)$ ที่เรียกว่าเป็นมูลค่าการบวก (additive cost) เนื่องจาก มูลค่าจะถูกสะสมตามขั้นเวลาไปจนสิ้นสุด ดังนั้นสำหรับค่าเริ่มต้น x_0 ที่กำหนด มูลค่ารวมของลำดับตัวควบคุม $\{u_0, \dots, u_{N-1}\}$ คือ

$$J(x_0; u_0, \dots, u_{N-1}) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k) \quad (4.2)$$

โดย $g_N(x_N)$ คือมูลค่าที่กำหนดให้กับส่วนปลายเมื่อกระบวนการลิ้นสุด



รูปที่ 4.1 รูปแบบปัญหาการควบคุมเหมาที่สุดแบบแนวอนจักษ์

เราต้องการหาค่าตอบตัวสุดของ (4.2) สำหรับลำดับ $\{u_0, \dots, u_{N-1}\}$ ที่สอดคล้องกับเงื่อนไขบังคับ (4.1)

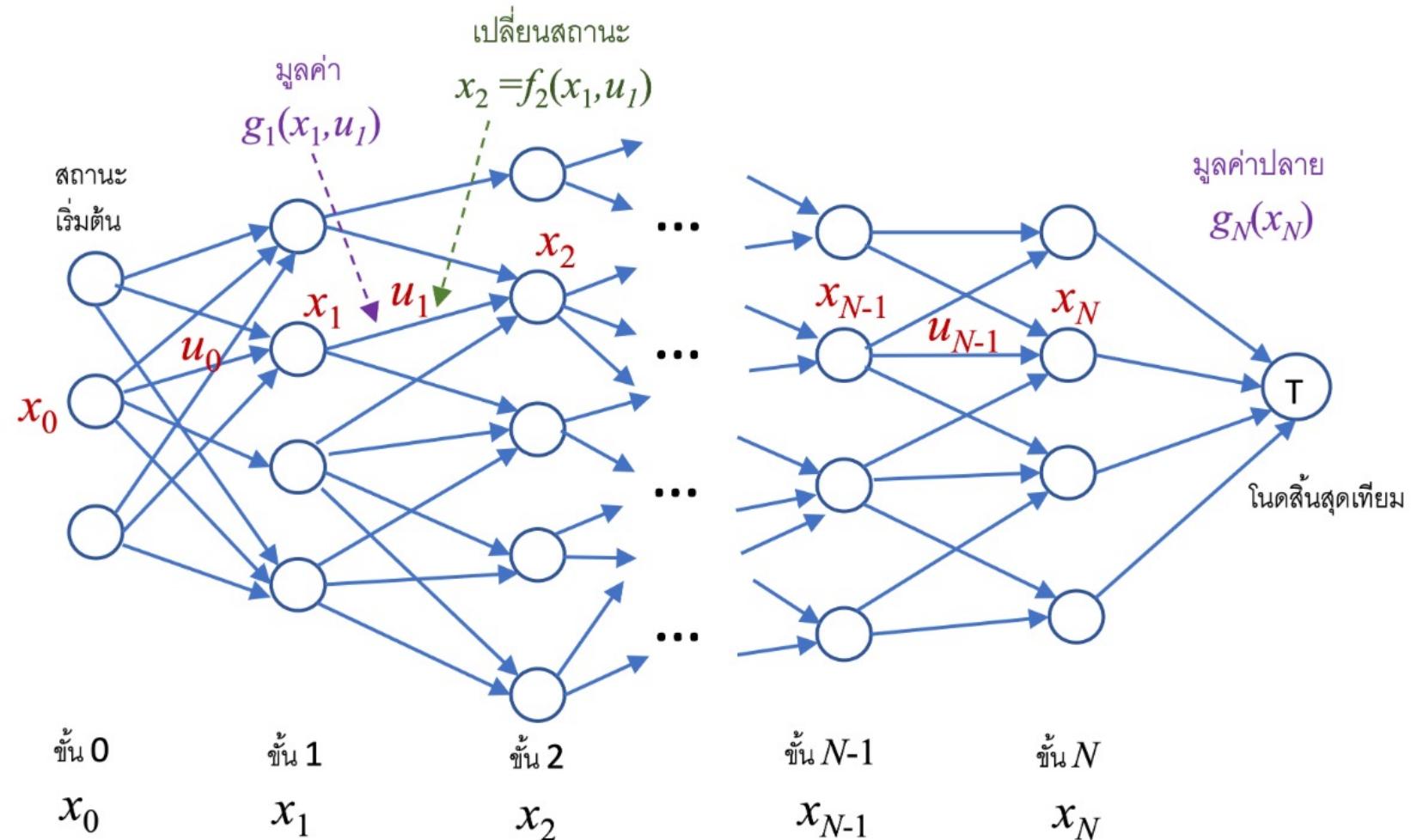
$$J^*(x_0) = \min_{\substack{u_k \in U_k(x_k) \\ k=0, \dots, N-1}} J(x_0; u_0, \dots, u_{N-1}) \quad (4.3)$$

เราต้องการหาค่าตอบตัวสุดของ (4.2) สำหรับลำดับ $\{u_0, \dots, u_{N-1}\}$ ที่สอดคล้องกับเงื่อนไขบังคับ (4.1)

$$J^*(x_0) = \min_{\substack{u_k \in U_k(x_k) \\ k=0, \dots, N-1}} J(x_0; u_0, \dots, u_{N-1}) \quad (4.3)$$

ข้อสังเกตหนึ่งสำหรับปัญหา DP ทั่วไปที่สอดคล้องกับการควบคุมเหมาะที่สุดคือ พลวัตของระบบจะวิวัฒนาไปตามขั้นเวลา ดังนั้น ตัวควบคุมในอนาคตไม่สามารถมีผลกระทบกับสถานะในอดีตได้ ตัวอย่างของแผนภาพการเปลี่ยนสถานะในปัญหา DP แบบแนวโน้มจำกัดแสดงได้ดังรูปที่ 4.2 เริ่มจากสถานะ x_0 ที่กำหนด การเปลี่ยนสถานะจะขึ้นกับพลวัต $x_{k+1} = f_k(x_k, u_k)$ เมื่อเลือกตัวควบคุม $u_k \in U_k$ ขณะที่มูลค่าในการเปลี่ยนสถานะของแต่ละขั้นถูกคำนวณโดยฟังก์ชัน $g_k(x_k, u_k)$ สำหรับขั้นสุดท้าย เราเพิ่มโนดสิ้นสุดเทียม T เข้าไปในแผนภาพ โดยเส้นทางจากสถานะ x_N ไปยังโนด T มีมูลค่าเท่ากับ $g_N(x_N)$

สังเกตว่าสำหรับสถานะ x_0 ที่กำหนด ลำดับของตัวควบคุม $\{u_0, \dots, u_{N-1}\}$ ที่เลือกทำให้สถานะเปลี่ยนตามแนววิถีหนึ่งจากแนววิถีทั้งหมดที่เป็นไปได้ และสิ้นสุดที่สถานะ x_N โนดหนึ่งในขั้น N เพื่อความเข้าใจที่ง่ายขึ้น สมมุติว่าเรามองมูลค่าของเส้นเปลี่ยนสถานะในแต่ละขั้นเป็นความยาวของเส้นนั้น จะเห็นได้ว่าการแปลงปัญหา DP เชิงกำหนดแบบแนวโน้มจำกัดก็คือการหาเส้นทางรวมที่สั้นที่สุดจากโนดเริ่มต้นสู่โนดปลายนั้นเอง โดยความยาวของเส้นทางรวมคือผลรวมของความยาวของเส้นระหว่างขั้น



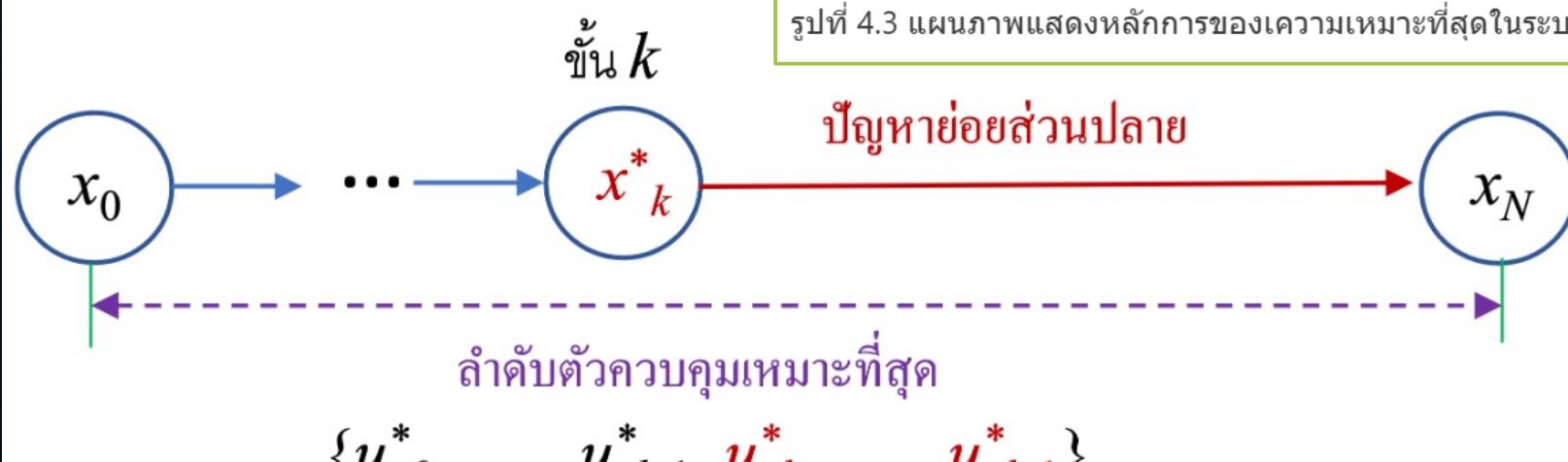
รูปที่ 4.2 แผนภาพการเปลี่ยนสถานะของปัญหา DP แบบวนวนอนจำกัด

4.3 อัลกอริทึมการโปรแกรมพลวัต

เมื่อเข้าใจรูปแบบปัญหา DP เราสามารถสร้างอัลกอริทึมสำหรับแก้ปัญหาเพื่อหาคำตอบ กล่าวโดยทั่วไปคือต้องการแก้ปัญหาการตัดสินใจโดยลำดับสำหรับ N ขั้น โดยแก้ปัญหาเป็นลำดับของปัญหาย่อยในแต่ละขั้น เป้าหมายของอัลกอริทึมคือลำดับของตัวควบคุม亥ม่าที่สุด u_0^*, \dots, u_{N-1}^* ที่ทำให้ลำดับของฟังก์ชันมูลค่า J_0^*, \dots, J_{N-1}^* มีค่าเหมาะสม(ต่ำ)ที่สุด เริ่มต้นจากฟังก์ชันมูลค่าส่วนปลายคือ $J_N^* = g_N(x_N)$ คำนวณ J_{N-1}^* โดยแก้ปัญหาการตัดสินใจขั้นเดียวที่ตัวแปร亥ม่าที่สุดคือ u_{N-1} ต่อมาใช้ J_{N-1}^* เพื่อคำนวณ J_{N-2}^* ดำเนินการเช่นนี้อย่างต่อเนื่องในการคำนวณ J_{N-3}^*, \dots, J_0^*

สังเกตว่าการคำนวณนี้มีรูปแบบย้อนหลังจากส่วนปลายมายังส่วนต้น ซึ่งอาศัยหลักการของความ亥ม่าที่สุดนั้นเอง โดยในระบบดีศรีตจะเขียนให้ชัดเจนขึ้น ดูรูปที่ 4.3 ประกอบ

รูปที่ 4.3 แผนภาพแสดงหลักการของความ亥ม่าที่สุดในระบบดีศรีต



หลักการของความเหมาะสมที่สุด (ระบบดีศกธิต)

$$x_{k+1} = f_k(x_k, u_k), \quad k = 0, 1, \dots, N-1 \quad (4.1)$$

สำหรับลำดับตัวควบคุมเหมาะสมที่สุด $\{u_0^*, \dots, u_{N-1}^*\}$ และค่าเริ่มต้น x_0 เป็นตัวกำหนดลำดับของสถานะ $\{x_1^*, \dots, x_N^*\}$ ตามสมการพลวัต (4.1) พิจารณาปัญหาย่อยที่เริ่มจาก x_k^* ณ ขั้นเวลา k ต้องการหาค่าต่ำสุดของ มูลค่ารายทาง (cost-to-go) จากขั้น k ไปยังขั้นปลาย N

$$g_k(x_k^*, u_k) + \sum_{m=k+1}^{N-1} g_m(x_m, u_m) + g_N(x_N) \quad (4.4)$$

โดยลำดับตัวควบคุม $\{u_k, \dots, u_{N-1}\}$ ในเซต $u_m \in U_m(x_m)$, $m = k, \dots, N-1$ ดังนั้นลำดับตัวควบคุมเหมาะสมที่สุดในส่วนปลายที่ถูกตัดออก $\{u_k^*, \dots, u_{N-1}^*\}$ จะเหมาะสมที่สุดสำหรับปัญหาย่อยนี้ด้วย

เราเรียกปัญหาย่อยนี้ว่า ปัญหาย่อยส่วนปลาย ที่เริ่มต้นจาก x_k^* หลักการของความเหมาที่สุดกล่าวว่า ส่วนปลายของลำดับที่เหมาที่สุดจะเหมาที่สุดสำหรับปัญหาย่อยส่วนปลาย โดยเหตุผลเดิมที่อธิบายแล้วข้างต้น หากลำดับตัวควบคุมส่วนปลายที่ถูกตัดออก $\{u_k^*, \dots, u_{N-1}^*\}$ ไม่เป็นแบบเหมาที่สุด เราต้องสามารถลดมูลค่าลงได้อีกโดยเลือกลำดับตัวควบคุมอื่นเมื่อกึ่งชั้น x_k^* ทั้งนี้เนื่องจากตัวควบคุมที่เลือกก่อนหน้านี้คือ $\{u_0^*, \dots, u_{k-1}^*\}$ มิได้จำกัดการเลือกตัวควบคุมในอนาคต

โดยหลักการของความเหมาที่สุด อัลกอริทึม DP จะเริ่มต้นจากการคำนวณค่าเหมาที่สุดของฟังก์ชันมูลค่า

$$J_N^*(x_N), J_{N-1}^*(x_{N-1}), \dots, J_0^*(x_0)$$

ตามลำดับ เริ่มต้นจาก J_N^* ย้อนหลังไปยัง J_{N-1}^*, J_{N-2}^* จนถึง J_0^* ดังนั้นเขียนเป็นโครงสร้างได้ดังนี้

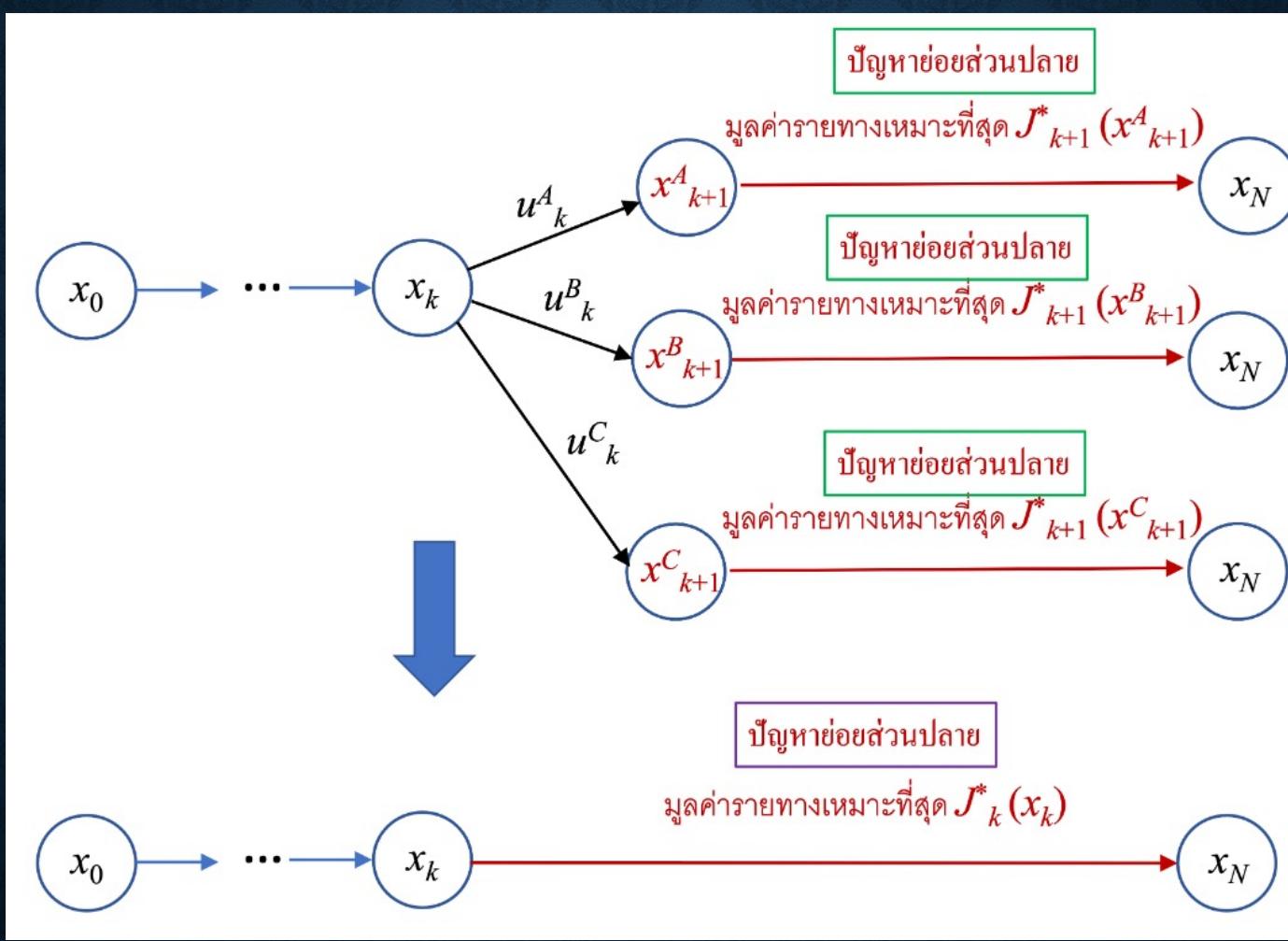
อัลกอริทึม DP สำหรับปัญหาเชิงกำหนดแบบวนวนจนจำกัด

เริ่มต้นจาก

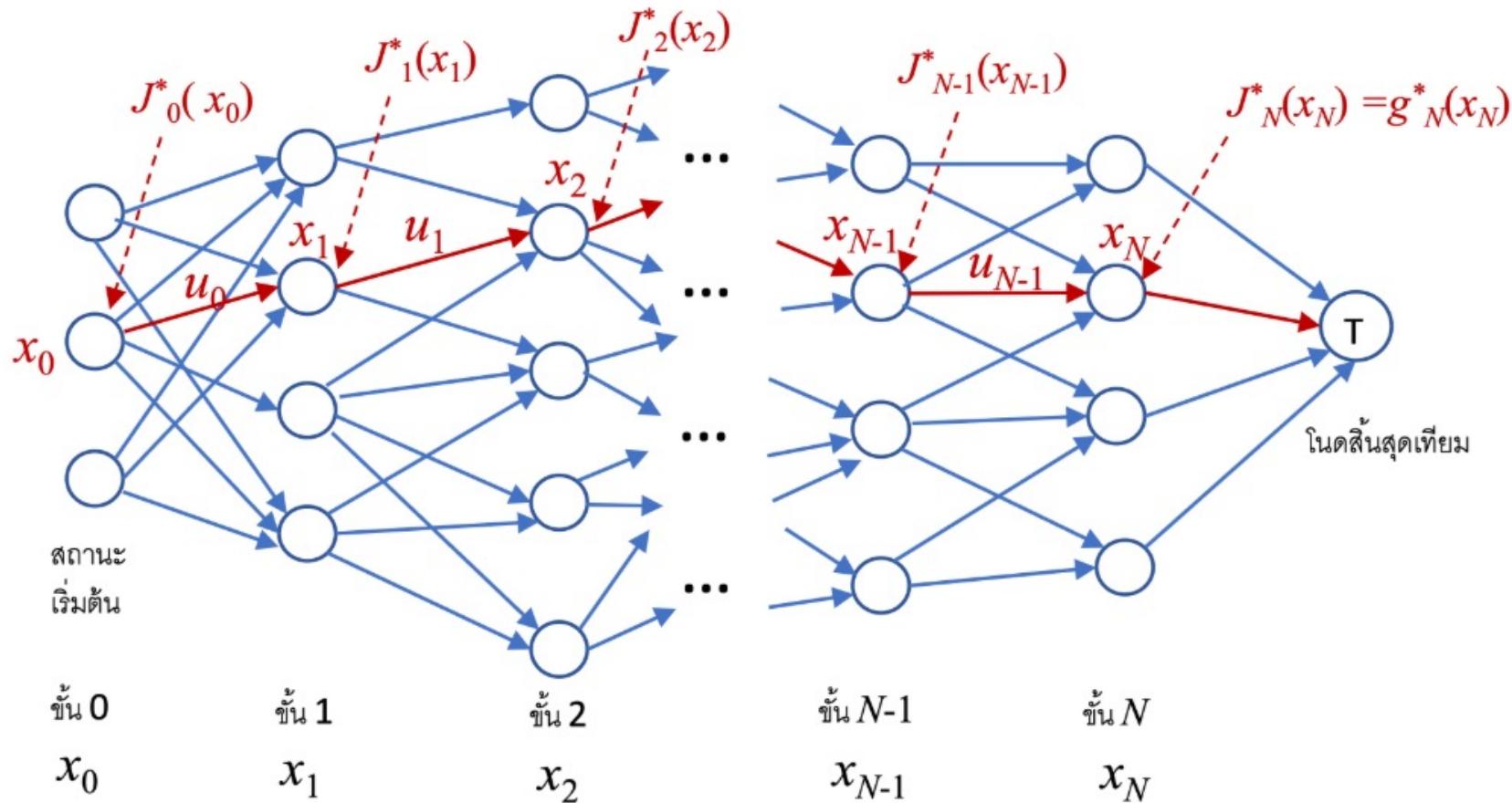
$$J_N^*(x_N) = g_N(x_N), \quad \forall x_N \quad (4.5)$$

วนซ้ำโดยลดค่า k จาก $N - 1$ ถึง 0

$$J^*(x_k) = \min_{u_k \in U_k(x_k)} [g_k(x_k, u_k) + J_{k+1}^*(f_k(x_k, u_k))], \quad \forall x_k \quad (4.6)$$



รูปที่ 4.4 มูลค่าหมายที่สุด $J_k^*(x_k)$ ของปัญหาอยู่ส่วนปลาย



รูปที่ 4.5 การคำนวณมูลค่าการไปเพماءที่สุดจากส่วนปลายย้อนมายังจุดเริ่มต้น

การหาลำดับตัวควบคุมเหมาะสมที่สุด

ในการหาลำดับของตัวควบคุมเหมาะสมที่สุด $\{u_0^*, \dots, u_{N-1}^*\}$ กำหนด

$$u_0^* = \arg \min_{u_0 \in U_0(x_0)} [g_0(x_0, u_0) + J_1^*(f_0(x_0, u_0))] \quad (4.7)$$

และ

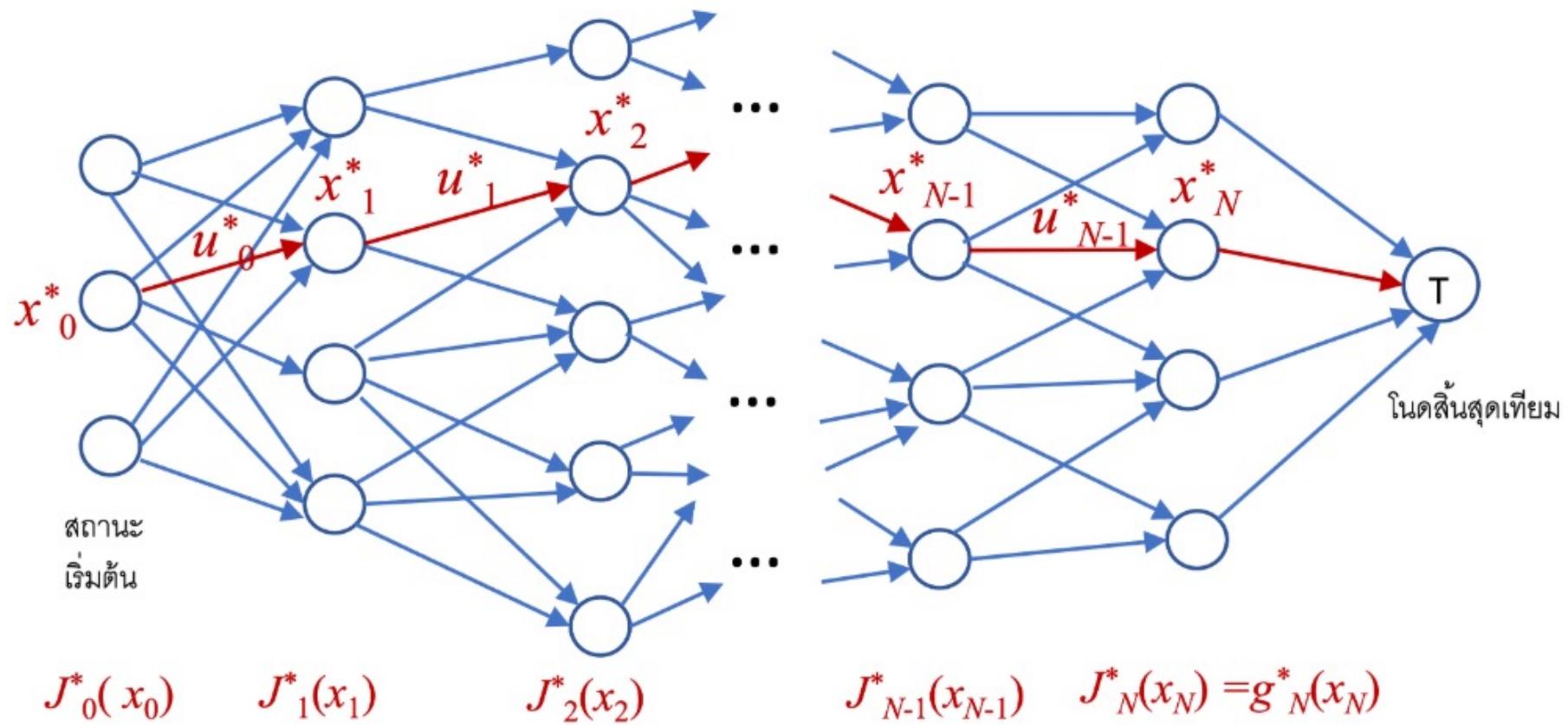
$$x_1^* = f_0(x_0, u_0^*) \quad (4.8)$$

วนซ้ำในทิศทางข้างหน้า จาก $k = 1, 2, \dots, N - 1$

$$u_k^* = \arg \min_{u_k \in U_k(x_k^*)} [g_k(x_k^*, u_k) + J_{k+1}^*(f_k(x_k^*, u_k))] \quad (4.9)$$

และ

$$x_{k+1}^* = f_k(x_k^*, u_k^*) \quad (4.10)$$



รูปที่ 4.6 การคำนวณตัวควบคุมเหมาะสมที่สุด $\{u_0^*, \dots, u_{N-1}^*\}$ ในทิศทางข้างหน้า

ผู้ที่เดยศึกษาด้าน RL อาจเดยเห็นการนำเสนอบัญหา DP อีกรูปแบบหนึ่งที่นิยมเรียกว่าฟังก์ชัน Q หรือฟังก์ชันมูลค่า-การกระทำ (action-value) โดยนิยามฟังก์ชันที่อยู่ภายในการหาค่าต่ำสุดของ (4.6)

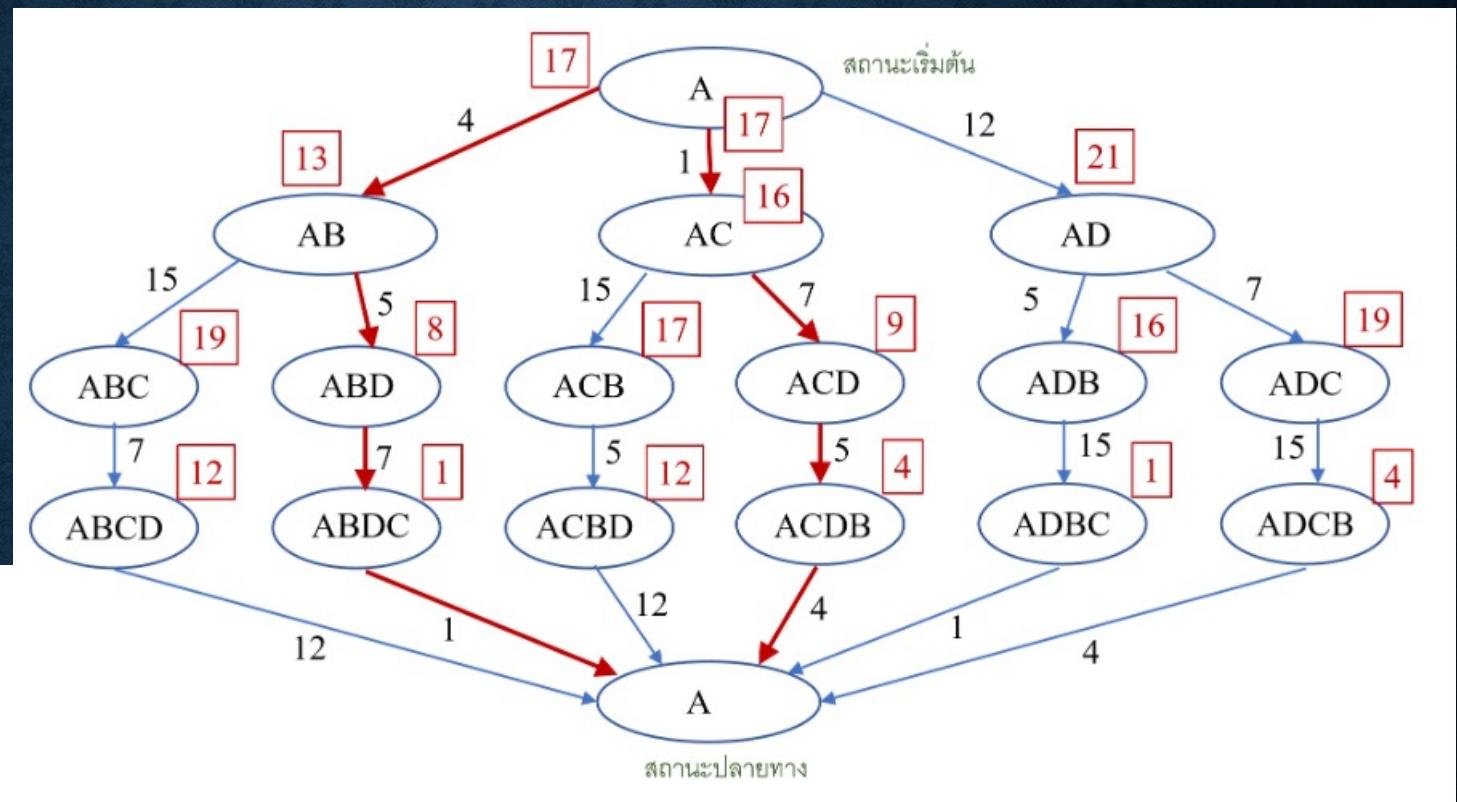
$$Q_k(x, u) = g_k(x_k, u_k) + J_{k+1}(f_k(x_k, u_k)) \quad (4.11)$$

โดยใน [4] ได้เปลี่ยนเป็นใช้อักษร S แทนเพื่อมีให้เข้ากับเมทริกซ์น้ำหนักใน LQR ในหนังสือนี้ หากมีการกล่าวถึงฟังก์ชัน Q จะใช้ฟอนต์ mathcal Q สำหรับฟังก์ชัน Q และฟอนต์ธรรมดา สำหรับเมทริกซ์น้ำหนัก และในเนื้อหาบางส่วนอาจคงอักษรของต้นกำเนิดไว้เพื่อง่ายต่อการอ้างอิง

เหตุผลของการใช้แนวทางนี้ใน RL เนื่องจากสะđวกต่อการประมาณค่าฟังก์ชันโดยวิธีการต่างๆ เช่นใช้โครงข่ายประสาทเทียน ในวิธีการใช้ฟังก์ชันมูลค่า (4.6) จะต้องเรียนรู้ทั้ง $f_k()$ และ $J_k()$ แต่ถ้าใช้ (4.11) จะเรียนรู้เพียงฟังก์ชันเดียวคือ Q กล่าวคือเป็นแนวทางที่เรียกว่าเป็นอิสระจากโมเดล (model-free)

	A	B	C	D
A		4	1	12
B	4		15	5
C	1	15		7
D	12	5	7	

เมทริกซ์ค่าใช้จ่ายการเดินทางระหว่างเมือง



ปัญหาการเดินทางของพนักงานขาย (traveling salesman problem)

4.4 แก้ปัญหาการควบคุมเหมาะสมที่สุดโดยการโปรแกรม พลวัต

เมื่อเริ่มเข้าใจรูปแบบปัญหาและอัลกอริทึม DP และ เราจะย้อนกลับมาเน้นที่สาระสำคัญของหนังสือนี้คือการควบคุมเหมาะสมที่สุด

4.4.1 ตัวควบคุม LQR

เริ่มจากปัญหา LQR ที่นิยามในบทที่ 3

$$\min_{x_{1:N}, u_{1:N-1}} J = \sum_{k=1}^{N-1} \frac{1}{2} x_k^T Q_k x_k + \frac{1}{2} u_k^T R_k u_k + \frac{1}{2} x_N^T Q_N x_N$$
$$s.t. \quad x_{k+1} = A_k x_k + B_k u_k, \quad Q_k \succcurlyeq 0, \quad R_k \succ 0 \quad (4.12)$$

การหาคำตอบของ LQR โดยอัลกอริทึม DP เริ่มจากส่วนปลาย กำหนดมูลค่าส่วนปลายตามพจน์สุดท้ายใน (4.12)

$$J_N(x) = \frac{1}{2}x^T Q_N x = \frac{1}{2}x^T P_N x \quad (4.13)$$

โดยจะใจดึงข้อมูลทริกซ์ P เหมือนกับในบทที่ 3 เพื่อจะแสดงให้เห็นต่อไปว่าได้คำตอบเหมือนกัน

ย้อนหลังหนึ่งขั้นเพื่อคำนวณ $J_{N-1}(x)$ ตาม (4.6) โดยแทนค่าพลวัตเชิงเส้นที่เป็นเงื่อนไขบังคับสมการใน (4.12)

$$J_{N-1} = \min_u \left[\frac{1}{2}x_{N-1}^T Q_N x_{N-1} + \frac{1}{2}u^T R u + J_N(Ax_{N-1} + Bu) \right] \quad (4.14)$$

$$J_{N-1} = \min_u \left[\frac{1}{2}u^T R u + \frac{1}{2}(Ax_{N-1} + Bu)^T P_N (Ax_{N-1} + Bu) \right] \quad (4.15)$$

มีรายละเอียดปลีกย่อยเล็กน้อยในขั้นการอนุพัทธ์ (4.14) เป็น (4.15) เช่นพจน์ $\frac{1}{2}x_{N-1}^T Q_N x_{N-1}$ ไม่ขึ้นกับ u จึงลงทะเบียนสำหรับการแทนค่าพลวัต $x_k = Ax_{k-1} + Bu_{k-1}$ แต่จะเห็นว่าแทนด้วย u ใน (4.15) เนื่องจาก u_{N-1} คือตัวควบคุมเหมาะสมที่สุดของ u สำหรับขั้น $N - 1$

ต้องการหาค่าตอบสำหรับพิงก์ชั้นมูลค่า (4.15) ในรูป $J_{N-1}(x)$ และยังติดอยู่ที่การหาค่าต่ำสุดของ n ดังนั้นใช้วิธีการเดิมคือหาค่าเกรเตียนต์ $\nabla_u J$ และให้เท่ากับศูนย์

$$\nabla_u J = Ru + B^T P_N (Ax_{N-1} + Bu) = 0 \quad (4.16)$$

ได้ค่าตอบสำหรับตัวควบคุมเหมาๆ ที่สุด

$$u_{N-1} = -(R + B^T P_N B)^{-1} B P_N A x_{N-1} = -K_{N-1} x_{N-1} \quad (4.17)$$

เมื่อนิยาม

$$K_{N-1} \triangleq (R + B^T P_N B)^{-1} B P_N A \quad (4.18)$$

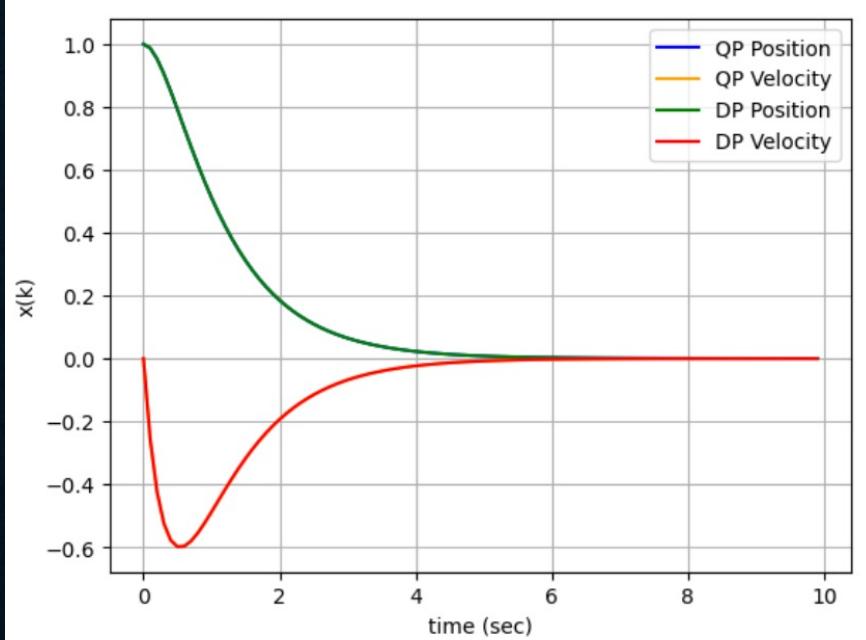
แทนค่า (4.17) กลับเข้าไปใน (4.14) จะได้เป็น

$$J_{N-1}(x) = \frac{1}{2} x^T [Q + K_{N-1}^T R K_{N-1} + (A - BK_{N-1})^T P_N (A - BK_{N-1})] x = \frac{1}{2} x^T P_{N-1} x \quad (4.19)$$

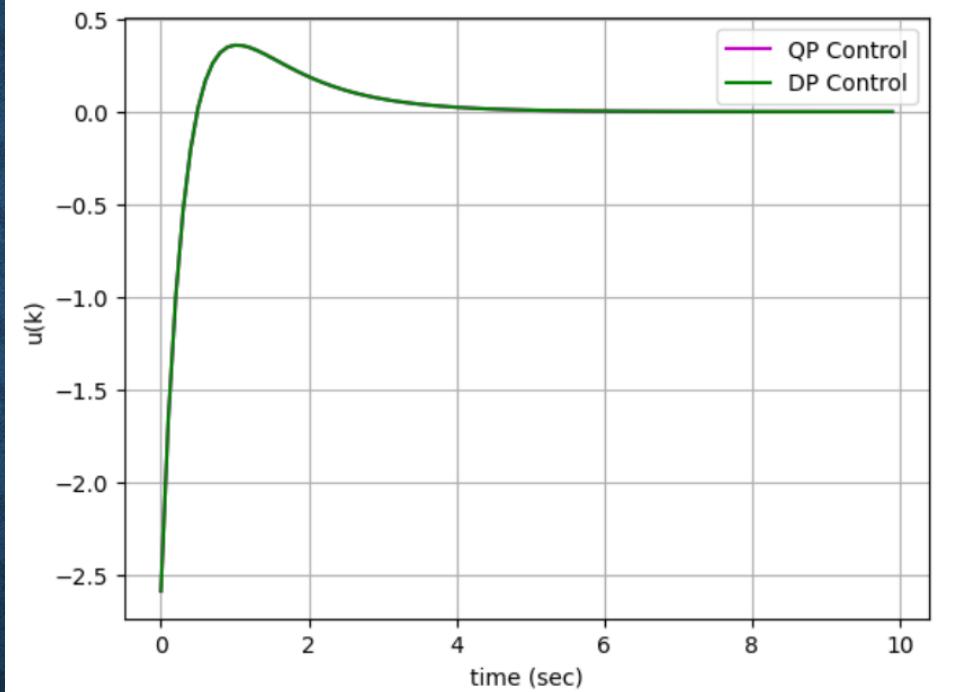
เมื่อนิยาม

$$P_{N-1} \triangleq Q + K_{N-1}^T R K_{N-1} + (A - BK_{N-1})^T P_N (A - BK_{N-1}) \quad (4.20)$$

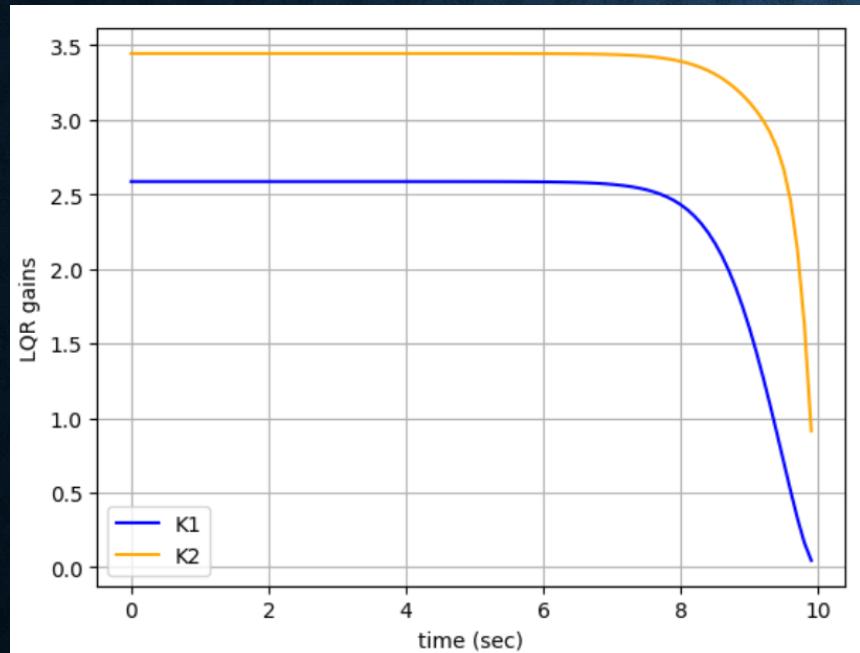
โดยการวนซ้ำย้อนหลังตามขั้นตอนนี้สำหรับเมตริกซ์ K_k และ P_k จนถึงขั้นตอน $k = 0$ เราจะได้ค่าตอบที่สอดคล้องกับวิธีการวนซ้ำrikค่าติในบทที่ 3 แม้ว่ารูปของสมการ P_k อาจดูเหมือนแตกต่างกัน แต่จากตัวอย่างต่อไปจะแสดงให้เห็นว่าให้ค่าตอบที่ตรงกัน



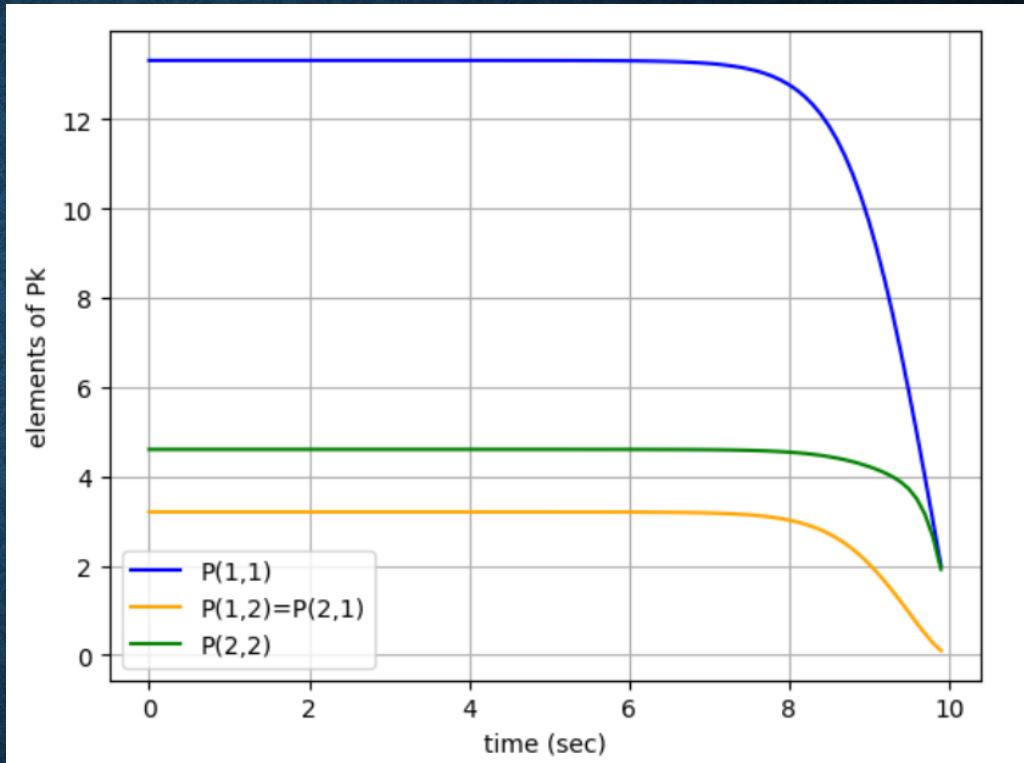
รูปที่ 4.8 ผลการเปรียบเทียบแนววิถีสถานะจากวิธี QP และ DP



รูปที่ 4.9 ผลการเปรียบเทียบแนววิถีตัวควบคุมจากวิธี QP และ DP



รูปที่ 4.10 ค่าสมาชิกของเวกเตอร์ K_k ที่คำนวณย้อนหลัง



รูปที่ 4.11 ค่าสมาชิกของเมทริกซ์ P_k ที่คำนวณย้อนหลัง

สำหรับกรณีแนวโน้มอนันต์ ใช้แพ็คเกจ Python Control Systems คำนวณหาเมทริกซ์ K ได้โดยวิธี `dlqr()` เมื่อونกับในบทที่ 3 เมื่อเปรียบเทียบกับ K_k เมื่อเข้าสู่ค่าคงที่จะพบว่าแตกต่างกันน้อยมาก

4.4.2 การควบคุมแบบทำนายโมเดล

ในเนื้อหาการควบคุมเหมาะสมที่สุดที่ผ่านมาเราได้ใช้ตัวควบคุม LQR เป็นตัวอย่างมาตรฐาน โดยเป็นตัวควบคุมพื้นฐานที่ช่วยให้เข้าใจหลักการของวิธีการต่างๆ ได้ดีขึ้น เป็นตัวควบคุมเชิงเส้นที่ใช้งานได้ดีในทางปฏิบัติแม้กับระบบไม่เป็นเชิงเส้น หากสามารถประมาณค่าเป็นระบบเชิงเส้นบริเวณจุดทำงาน

อย่างไรก็ตาม ข้อจำกัดของตัวควบคุม LQR จะเด่นขึ้นเมื่อมีเงื่อนไขบังคับสมการ เช่นการจำกัดค่าสถานะและ/หรือเอาต์พุตควบคุม ซึ่งโดยวิธี LQR มิได้ออกแบบมาเพื่อจัดการกับผลกระทบนี้ เราสามารถเข้าใจผลจากเงื่อนไขบังคับสมการได้ดีขึ้นในมุมมองของ DP

โดยวิธีการ DP เมื่อกำหนดฟังก์ชันมูลค่าขั้นของปัญหา LQR

$$g(x_k, u_k) = \frac{1}{2} x_k^T Q x_k + \frac{1}{2} u_k^T R u_k \quad (4.21)$$

หากเราทราบฟังก์ชันมูลค่ารายทาง $J(x)$ ทำให้สามารถหาคำตอบ u โดยแก้ปัญหาขั้นเดียว

$$\begin{aligned} u_k &= \arg \min_u [g(x_k, u) + J_{k+1}(f(x_k, u))] \\ &= \arg \min_u \left[\frac{1}{2} u^T R u + (Ax_k + Bu)^T P_{k+1} (Ax_k + Bu) \right] \quad (4.22) \end{aligned}$$

โดยจะทิ้งพจน์ $\frac{1}{2} x_k^T Q x_k$ ที่ไม่เป็นฟังก์ชันของ u_k เราอาจเพิ่มเงื่อนไขบังคับเพื่อจำกัดค่า u เข้า ในปัญหาขั้นเดียว แต่ย่อมมีผลทำให้คำตอบคลาดเคลื่อนไป เพราะมูลค่ารายทาง $J_{k+1}(f(x_k, u))$ ถูกคำนวณโดยไม่มีเงื่อนไขบังคับ ดังนั้นระบบนี้จะทำงานได้ไม่ดีอย่างแน่นอน

วิธีแก้คือเพิ่มจำนวนขั้นให้กับปัญหา

$$\min_{\substack{x_{1:H} \\ u_{1:H-1}}} \left[\sum_{k=1}^{H-1} \left(\frac{1}{2} x_k^T Q x_k + \frac{1}{2} u_k^T R u_k \right) + x_H^T P_H x_H \right] \quad (4.23)$$

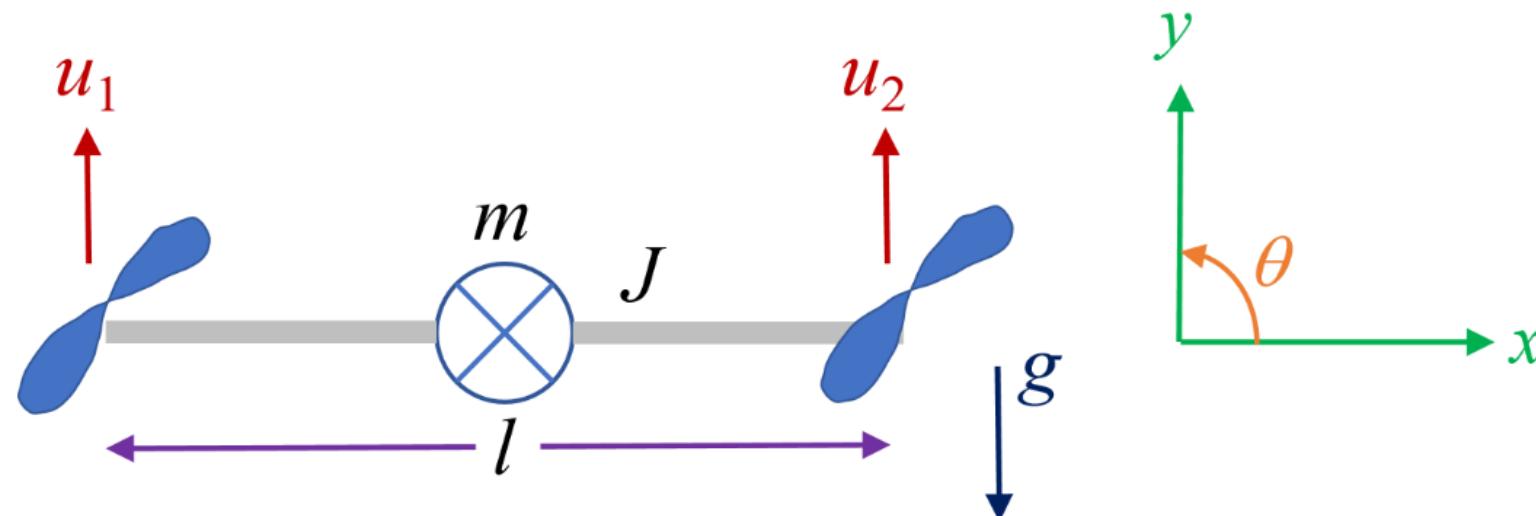
ก่อนถึงมูลค่ารายทาง $x_H^T P_H x_H$ โดย $H \ll N$ คือจำนวนแนวนอน (horizon) วิธีการนี้เรียกว่า การควบคุมแบบทำนายโนเดล (MPC: *model predictive control*) หรือ การควบคุมโดยห่าง แนวโนน (RHC: *receding horizon control*) ในหนังสือนี้จะใช้ชื่อแรกโดยเรียกตัวย่อ MPC สังเกตว่าหากไม่มีเงื่อนไขบังคับเพิ่มเติม การควบคุม MPC จะให้ผลเหมือนกับ LQR โดยไม่ขึ้นกับค่า H

แนวคิดของ MPC ในการลดผลกระทบจากการจำกัดค่าตัวแปรคือ หากค่าเหมาะสมที่สุดแบบนี้ เงื่อนไขบังคับเป็นจำนวน H ขั้น เพื่อทำให้สถานะเข้าใกล้ค่าอ้างอิงเพียงพอดونเงื่อนไขบังคับ ไม่มีผล ดังนั้นมูลค่ารายทางของ LQR จึงเป็นค่าที่ถูกต้องสำหรับในอนาคต หลักการนี้ ครอบคลุมไปถึงกรณีที่มูลค่ารายทาง $J(x)$ ได้จากการประมาณค่า ซึ่งความแม่นยำจะมีผลต่อ สมรรถนะระบบ โดยยิ่งค่าที่ประมาณได้ $\hat{J}(x)$ ใกล้เคียงกับค่าจริงมากขึ้น ก็จะสามารถใช้ค่า H ที่สั้นลงได้

เนื้อหาในหัวข้อนี้จะพิจารณาปัญหา MPC แบบคอนเวกซ์เท่านั้น ตามนิยามฟังก์ชันคอนเวกซ์ ในภาคผนวก A ทำให้สามารถหาค่าตอบได้แบบออนไลน์โดยวิธี QP

ตัวอย่าง 4.3 ในตัวอย่างนี้จะเปรียบเทียบการใช้ตัวควบคุม LQR กับ MPC กับโดรนที่นิยมเรียกว่า ควอดโรเตอร์ (quadrotor) หรือ ควอดคอปเตอร์ (quadcopter) ซึ่งโครงสร้างประกอบด้วยตัวขับ 4 ใบพัด แต่ในเบื้องต้นเพื่อให้เข้าใจง่ายขึ้นจะพิจารณา โดรนที่มี 2 ใบพัดและเคลื่อนที่ในระนาบ 2 มิติ ดังแสดงในรูปที่ 4.12 บรรยายพลวัตไม่เป็นเชิงเส้นได้ดังนี้

$$\begin{aligned} m\ddot{x} &= -(u_1 + u_2)\sin(\theta) \\ m\ddot{y} &= (u_1 + u_2)\cos(\theta) - mg \\ J\ddot{\theta} &= \frac{l}{2}(u_2 - u_1) \end{aligned} \tag{4.24}$$



รูปที่ 4.12 โดรน 2 ใบพัดในระนาบ 2 มิติ

เมื่อประมาณค่าเชิงเส้นรอบตำแหน่งที่ลอยนิ่ง (hover) จะได้เป็น

$$u_1 = u_2 = \frac{1}{2}mg \quad (4.25)$$

$$\begin{aligned}\Delta \ddot{x} &\approx -g\Delta\theta \\ \Delta \ddot{y} &\approx \frac{1}{m}(\Delta u_1 + \Delta u_2) \\ \Delta \ddot{\theta} &\approx \frac{l}{2J}(\Delta u_2 - \Delta u_1)\end{aligned} \quad (4.26)$$

เขียนเป็นสมการเมทริกซ์ได้ดังนี้

$$\begin{bmatrix} \Delta \dot{x} \\ \Delta \dot{y} \\ \Delta \dot{\theta} \\ \Delta \ddot{x} \\ \Delta \ddot{y} \\ \Delta \ddot{\theta} \end{bmatrix} = \left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \hline - & - & - & - & - & - \\ 0 & 0 & -g & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right] \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \theta \\ \Delta \dot{x} \\ \Delta \dot{y} \\ \Delta \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ - & - \\ 0 & 0 \\ \frac{1}{2J} & \frac{m}{l} \end{bmatrix} \begin{bmatrix} \Delta u_1 \\ \Delta u_2 \end{bmatrix} \quad (4.27)$$

นิยามฟังก์ชันมูลค่า MPC

$$J = \sum_{k=1}^{H-1} \left[\frac{1}{2}(x_k - x_{ref})^T Q(x_k - x_{ref}) + \frac{1}{2} \Delta u_k^T R \Delta u_k + \frac{1}{2}(x_H - x_{ref})^T P_H(x_H - x_{ref}) \right] \quad (4.28)$$

ขั้นตอนสำคัญสำหรับตัวอย่างนี้คือเขียนโค้ดเพื่อใช้แพ็กเกจ OSQP หาคำตอบเหมาะที่สุดสำหรับตัวควบคุม MPC ซึ่งต้องจัดรูปข้อมูลที่เราสร้างให้ตรงกับที่ OSQP กำหนด เราจะยังคงตามรอยวิธีการจาก [4] ซึ่งใช้รูปแบบเดิมในบทที่ 3 ในการสร้างเวกเตอร์ที่ประกอบด้วยแนววิถีของตัวควบคุมและสถานะลับกันดังเช่นในสมการ (3.25)-(3.27) แต่มีส่วนที่ต้องเพิ่มเติมเพื่อให้สอดคล้องกับรูปแบบที่ใช้โดย OSQP ดังนั้นเพื่อความเข้าใจในส่วนนี้จะขอยกย่อรูปแบบที่ใช้ในตัวหาคำตอบของ OSQP [6]

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^T P x + q^T x \\ \text{s.t. } & l \leq A x \leq u \end{aligned} \tag{4.29}$$

สังเกตว่าตัวอักษรที่ใช้สำหรับเมทริกซ์ใน (4.29) จะมีความหมายแตกต่างจากที่เราใช้อยู่ แต่เป็นตัวอักษรที่ใช้ในอาร์กิวเมนต์ของ OSQP ดังนั้นสิ่งแรกที่ต้องคำนึงคือรูปแบบของการซ้ำกันของสัญกรณ์

เบื้องต้นสังเกตพึงชันมูลค่าระหว่าง (4.28) กับ (4.29) จะเห็นว่าอยู่ในรูปแบบที่แตกต่างกัน โดยยังไม่เห็นชัดเจนว่าใน (4.28) มีพจน์ที่เป็นเชิงเส้น แต่เมื่อพิจารณาพจน์กำลังสอง $\frac{1}{2}(x_k - x_{ref})^T Q(x_k - x_{ref})$ โดยคุณครະจายแต่ละพจน์

$$\frac{1}{2}(x_k - x_{ref})^T Q(x_k - x_{ref}) = \frac{1}{2} [x_k^T Q x_k - 2x_{ref}^T Q x_k + x_{ref}^T Q x_{ref}] \tag{4.30}$$

เนื่องจากพจน์สุดท้ายเป็นค่าคงที่ไม่ขึ้นกับ x สามารถจัดทิ้งไปได้ ดังนั้น (4.30) เขียนใหม่ได้เป็น

$$\frac{1}{2}x_k^T Q x_k - x_{ref}^T Q x_k \quad (4.31)$$

ซึ่งอยู่ในรูปของ (4.29) เมื่อนิยาม $q = -(x_{ref}^T Q)^T = -Q x_{ref}$ เราสามารถใช้วิธีนี้กับทุกพจน์กำลังสองของผลต่างสถานะ $x_k - x_{ref}$ ทำให้ได้ฟังก์ชันมูลค่าในรูปที่ OSQP กำหนด

ต่อมาพิจารณาส่วนของเงื่อนไขบังคับ โดยใช้วิธีการคล้ายเดิมใน (3.28)-(3.30) บทที่ 3 เพียงแต่ต้องจัดรูปให้สอดคล้องกับ (4.29) เดิมจาก $Cz = d$ ใน (3.29) เป็น

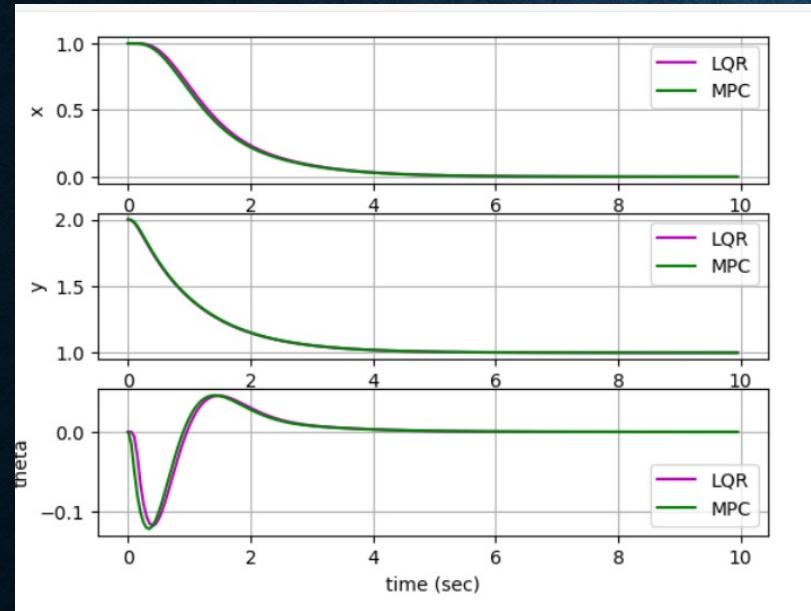
$$d \leq Cz \leq d \quad (4.32)$$

หลังจากนั้นแต่งเติมเงื่อนไขบังคับอสมการเป็นค่าจำกัดของตัวควบคุมในส่วนล่างของเมทริกซ์นี้

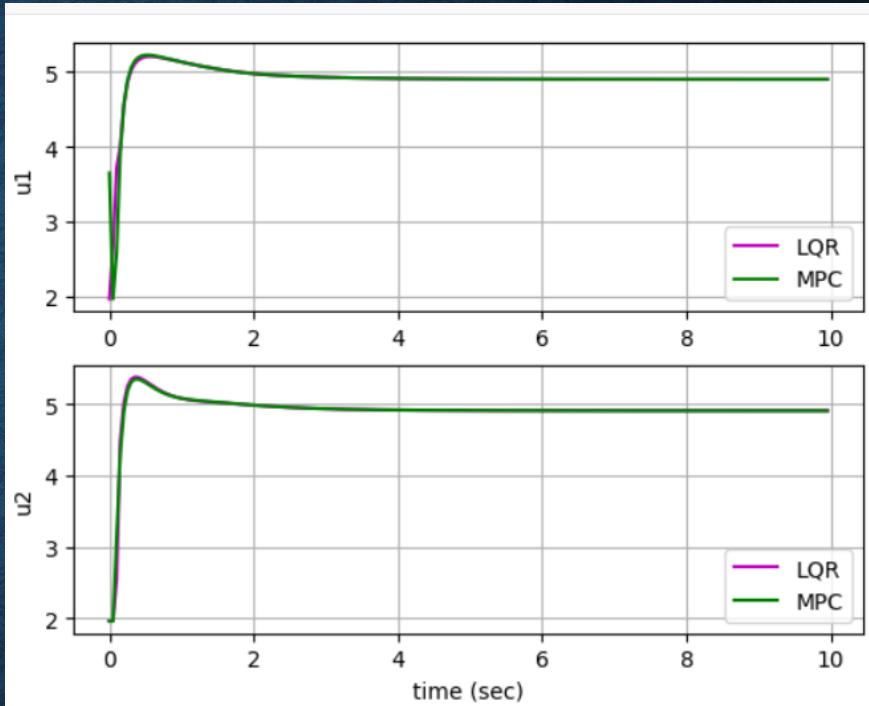
อีกประเด็นหนึ่งที่อาจทำให้สับสนได้คือสัญกรณ์ที่แตกต่างกันระหว่างเนื้อหา/โคดในหนังสือกับที่ใช้ภายใน OSCP ดังจะเห็นได้จากอาร์กิวเมนต์ของ `prob.setup()` หลังจากสร้างอ้อมเขต `prob = osqp.OSQP()` สรุปความแตกต่างของสัญกรณ์ได้ตามตาราง 3.1 ดังนี้

หนังสือ	ภายใน OSCP	รายละเอียด
H	P	พจน์กำลังสองของฟังก์ชันมูลค่า (4.29)
b	q	พจน์เชิงเส้นของฟังก์ชันมูลค่า (4.29)
D	A	เมตริกซ์ในเงื่อนไขบังคับ (4.29)
A	.	เมตริกซ์สถานะของพลวัต (4.27) ไม่เป็นข้อมูลในบล็อก I,u,A ใน OSCP
B	.	เมตริกซ์อินพุตของพลวัต (4.27) ไม่เป็นข้อมูลในบล็อกของ P,A ใน OSCP
P,Q,R	.	เมตริกซ์น้ำหนักในฟังก์ชันมูลค่า (4.28) ไม่เป็นข้อมูลในบล็อกของ P, q ใน OSCP
lb	I	ขอบเขตด้านล่างของเงื่อนไขบังคับใน (4.29)
ub	u	ขอบเขตด้านบนของเงื่อนไขบังคับใน (4.29)

ตาราง 3.1 สรุปความแตกต่างของสัญกรณ์ในเนื้อหาหนังสือและที่ใช้ภายใน OSCP



รูปที่ 4.13 เปรียบเทียบสถานะของโดรนเมื่อเลือกต่าแห่งหนึ่งเริ่มต้น $(x, y, \theta) = (1.0, 2.0, 0.0)$



รูปที่ 4.14 เปรียบเทียบตัวควบคุมของโดรนเมื่อเลือกต่าแห่งหนึ่งเริ่มต้น $(x, y, \theta) = (1.0, 2.0, 0.0)$

```
▶ Quad2Dsim(uhist1) # LQR
```

[118]

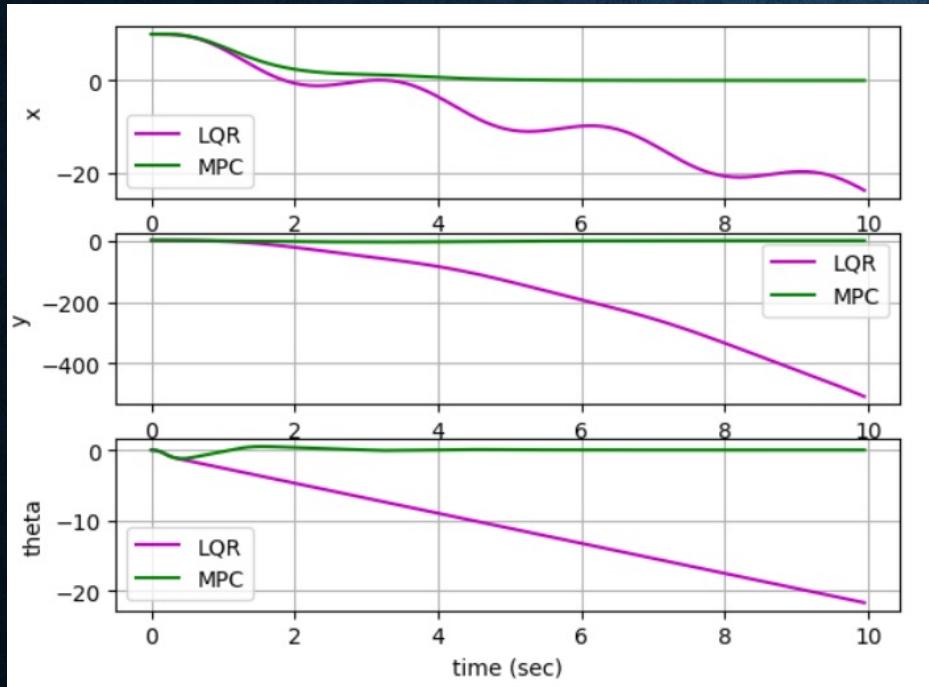
```
... simulating...
done.
generating animation...
```

...

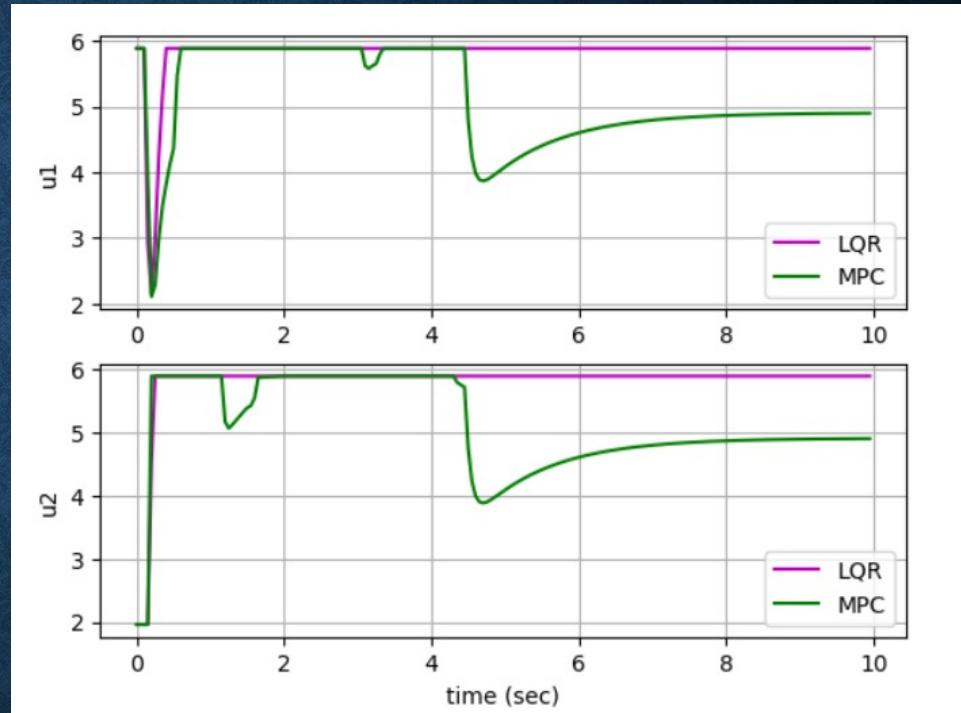
$t = 0.0$



รันเซลล์ในโน๊ตบุ๊ก chapter4.ipynb เพื่อจำลองการบินของโดรนในระนาบ 2 มิติ



รูปที่ 4.15 เปรียบเทียบสถานะของโดรนเมื่อเลือกต่าแห่งนั่งเริ่มต้น $(x, y, \theta) = (10.0, 2.0, 0.0)$



รูปที่ 4.16 เปรียบเทียบตัวควบคุมของโดรนเมื่อเลือกต่าแห่งนั่งเริ่มต้น $(x, y, \theta) = (10.0, 2.0, 0.0)$

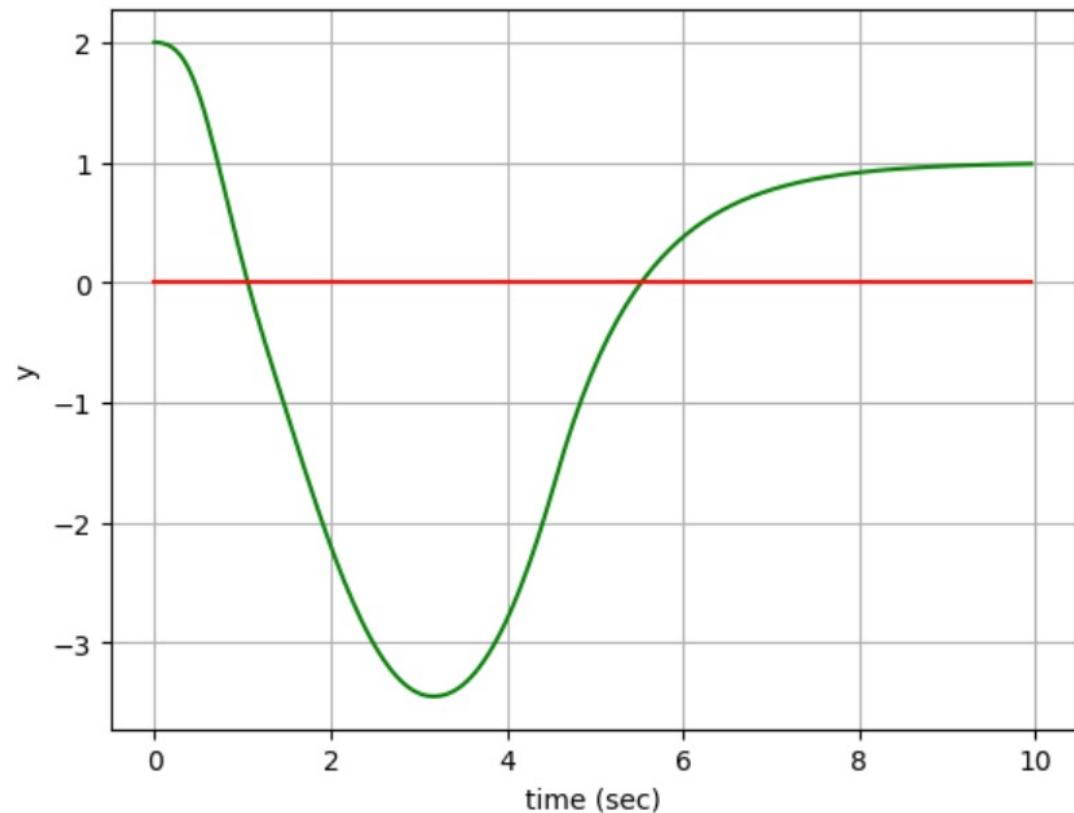
simulating...

done.

generating animation...

$t = 0.0$

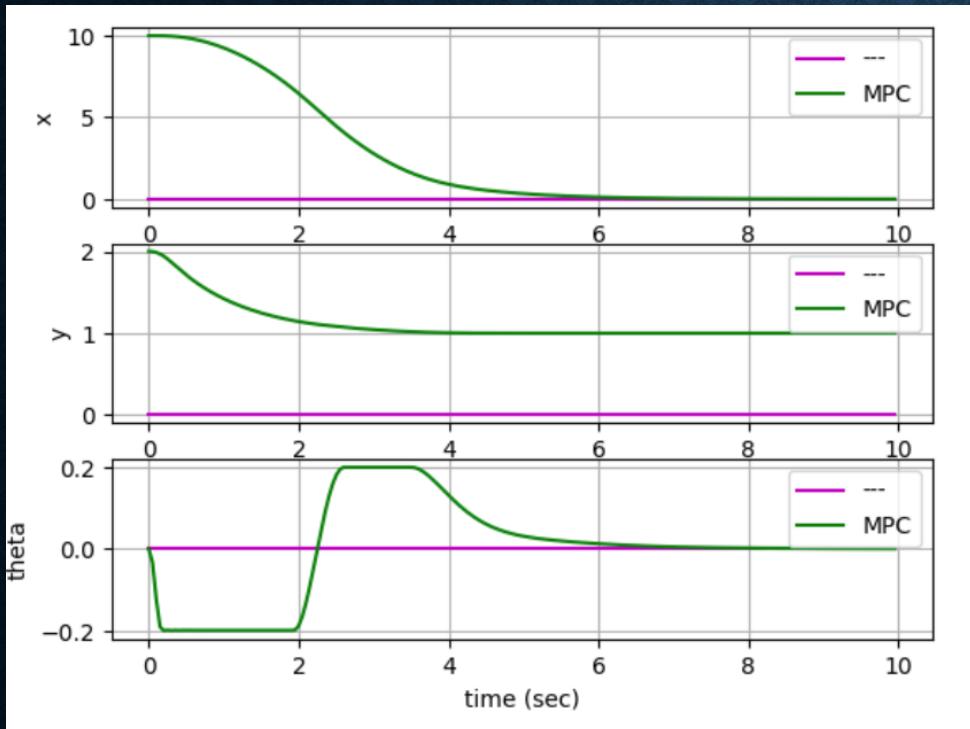
x



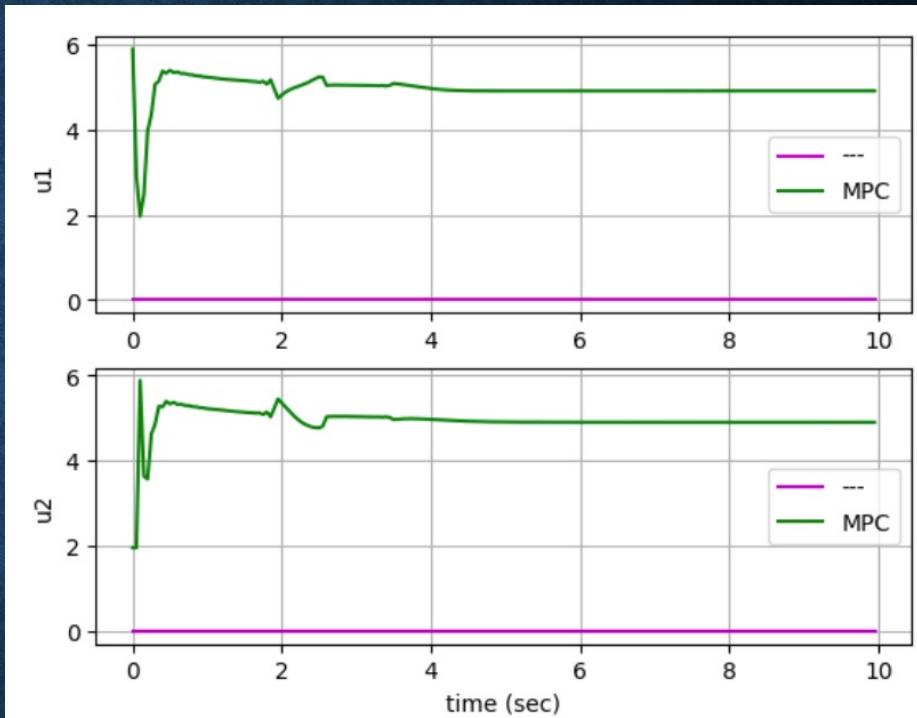
รูปที่ 4.17 สมรรถนะของโดรนในแนวแกน y อุ่งต่ำกว่าระดับพื้น

ปัญหานี้สามารถอธิบายได้จากการประมาณพลวัตไม่เป็นเชิงเส้นของโดรนโดยโมเดลเชิงเส้น ซึ่งสามารถใช้งานได้ดีเมื่อการเบี่ยงเบนมีค่าไม่น่าจะมาก แต่จากตำแหน่งเริ่มต้นที่ระยะทางในแนว x ห่างออกไปถึง 10 เมตร ทำให้ตัวควบคุมสั่งงานให้โดรนเปลี่ยนค่าองศาอย่างมากเพื่อเคลื่อนที่ในแนว x เป็นผลให้โมเดลเชิงเส้นไม่เป็นตัวแทนที่ดีของพลวัตจริงของโดรน

การใช้ตัวควบคุม MPC สามารถจัดการกับปัญหานี้ได้ เพราะเราสามารถเพิ่มเงื่อนไขบังคับอสมการตามต้องการ ในกรณีนี้ต้องการจำกัดค่าของ θ มิให้เบี่ยงเบนมากเกินกว่าที่กำหนด ในตัวอย่างนี้ตั้งไว้ $\pm 20\%$ แก้ไขส่วนตั้งค่า OSQP สำหรับ D, lb, ub ดังในเซลล์ด้านล่างนี้ โดยการสร้างเมตริกซ์ส่วนบนยังคงเดิม ในกรณีที่รันโน้ตบุกนี้ต่อเนื่องจากด้านบน เพียงรันเซลล์นี้เท่านั้น



รูปที่ 4.18 ผลตอบสนองของสถานะของโดรนจากตัวควบคุม MPC
เมื่อจำกัดค่าเบี่ยงเบน θ



รูปที่ 4.19 เอ้าต์พุตตัวควบคุม MPC เมื่อจำกัดค่าเบี่ยงเบน θ

4.5.1 สมการแฮมิลตัน-จาโคบี-เบลแมน

โดยแท้จริงแล้วสมการ (4.6),(4.7) เป็นมากกว่าอัลกอริทึม แต่ยังเป็นเงื่อนไขเพียงพอสำหรับความเหมาะสมที่สุดด้วย ถ้าเราสามารถหา J^* และ u^* ที่ทำให้ (4.6),(4.7) เป็นจริง u^* นั้นต้องเป็นตัวควบคุมเหมาะสมที่สุด

สำหรับระบบเวลาต่อเนื่อง เมื่อพิจารณา müลค่าการบวก (additive cost)

$$J = \int_0^T l_c(x(t), u(t)) dt \quad (4.33)$$

ภายใต้เงื่อนไขพลวัต

$$\dot{x} = f_c(x, u) \quad (4.34)$$

เงื่อนไขเพียงพอจะเป็นดังนี้

$$0 = \min_u \left[l_c(x, u) + \frac{\partial J^*}{\partial x} f_c(x, u) \right] \quad (4.35)$$

$$u^*(x) = \arg \min_u \left[l_c(x, u) + \frac{\partial J^*}{\partial x} f_c(x, u) \right] \quad (4.36)$$

(4.36) เรียกว่าสมการแฮมิลตัน-จาโคบี-เบลแมน (Hamilton-Jacobi-Bellman ย่อว่า HJB) [1,5]

การแสดงที่มาของ (4.36) อย่างง่ายคืออนุพัทธ์ในระบบดีสครีตและให้ลิมิตความเวลาสู่ศูนย์จากการประมาณค่าอยาเลอร์ เมื่อกำหนดความเวลา h

$$x_{k+1} = x_k + h f_c(x_k, u_k) \quad (4.37)$$

ในทำนองเดียวกัน ฟังก์ชันมูลค่าในระบบดีสครีตสามารถประมาณค่าได้จาก

$$l_d(x, u) = h l_c(x, u) \quad (4.38)$$

จากสมการ (4.6) เขียนใหม่อีกรังโดยนิยามมูลค่าขั้นเป็น $l_d(x_k, u_k)$

$$J^*(x_k) = \min_u [l_d(x_k, u_k) + J^*(f_d(x_k, u_k))] \quad (4.39)$$

เมื่อแทนค่าการประมาณค่านี้ในพจน์ด้านขวา จะได้เป็น

$$J^*(x_k) = \min_u [h l_c(x_k, u_k) + J^*(x_k + h f_c(x_k, u_k))] \quad (4.40)$$

เพื่อเข้าใจได้ง่ายง่ายสมมุติว่าพลวัตไม่เป็นฟังก์ชันของเวลา เราสามารถประมาณค่าได้เป็น

$$J^*(x_k) = \min_u \left[h l_c(x_k, u_k) + J^*(x_k) + \frac{\partial J}{\partial x} f_c(x_k, u_k) \right] \quad (4.41)$$

เนื่องจากพจน์ $J^*(x_k)$ ไม่ขึ้นกับ u สามารถนำออกนอก \min_u ได้ จะเห็นว่าพจน์ $J^*(x_k)$ ด้านซ้ายและขวาหักล้างกัน หลังจากนั้นขัดสัมประสิทธิ์ h ด้านขวาออก เหลือเป็นสมการ HJB (4.36)

បរទាន់ករណ

1. R.E. Bellman. Dynamic Programming. Princeton University Press. 1957.
2. D.P. Bertsekas. Reinforcement Learning and Optimal Control. MIT Press. 2019.
3. D.P. Bertsekas. A Course in Reinforcement Learning, 2nd ed. Athena Scientific. 2025.
4. Z. Manchester et.al. 16-745 Optimal Control & Reinforcement Learning, Course materials, Carnegie Mellon University. 2024,2025.
5. R. Tedrake. Underactuated Robotics: Algorithms for Walking, Running, Swimming, Flying, and Manipulation (Course Notes for MIT 6.832). 2023.
6. B. Stellato, G. Banjac, P. Goulart, A. Bemporad and S. Boyd.
OSQP : an operator splitting solver for quadratic programs.
Mathematical Programming computation: 12(4):637-672. 2020.