1 บทนำ

คงไม่อาจปฏิเสธได้ว่าการศึกษาด้านระบบควบคุมเป็นงานที่ท้าทายอย่างยิ่ง โดยเฉพาะสำหรับผู้เริ่มต้น เหตุผลหลักคือทฤษฎีและหลักการอ้างอิงกับคณิตศาสตร์ในหลายหัวข้อ เช่น พีชณิตเชิงเส้น ตัวแปรเชิงซ้อน แคลคูลัส จนถึงการวิเคราะห์ขั้นสูงสำหรับระบบไม่เป็นเชิงเส้น เช่นทฤษฎีเลียพูนอฟ นอกจากนั้นแนวทางของระบบควบคุมยังมีการแตกแขนงออกไปอย่างกว้างขวาง และมีการผสมผสานกับวิธีการในหลากหลายสาขา เช่นโครงข่ายประสาทเทียม (neural networks) ตรรกศาสตร์คลุมเครือ (fuzzy logic) หรือการเรียนรู้ของเครื่อง ในปัจจุบันยังมีผู้นำเสนอแนวทางใหม่อยู่อย่างต่อเนื่อง

1.1 ต้นกำเนิดของการควบคุมเหมาะที่สุด

แม้ว่ามนุษย์จะได้คิดค้นการควบคุมป้อนกลับมาเป็นเวลานานตั้งแต่ยุคกรีกโบราณก่อนคริสต์ศักราช แต่ความก้าวหน้าจะเด่นชัดเมื่อมนุษย์พัฒนาศาสตราวุธเพื่อเอาชนะฝ่ายตรงข้าม โดยเฉพาะในช่วงสงครามโลกครั้งที่ 2 ที่เป็นจุดเริ่มต้นของการออกแบบตัวควบคุมแบบคลาสสิก (classical control) โดยการแปลงระบบพลวัตระบบเชิงเส้นให้อยู่ในรูปฟังก์ชันถ่ายโอน และอาศัยการพล็อตแผนภาพโบเด (Bode plot) ของระบบวงเปิดลงบนกระดาษกราฟ เป็นการวิเคราะห์และออกแบบในโดเมนความถี่ การออกแบบในลักษณะนี้มีชื่อเรียกว่า *การจัดสัณฐานวงรอบ (loopshaping)* [1] จนกระทั่งประมาณช่วงทศวรรษที่ 60 เมื่อเริ่มมีการใช้คอมพิวเตอร์ช่วยในการคำนวณ ซึ่งตัวประมวลผลในยุคเริ่มต้นถนัดการคำนวณข้อมูลในรูปของเมทริกซ์และเวกเตอร์ การวิเคราะห์และออกแบบจึงเปลี่ยนเป็นรูปของเมทริกซ์ในโดเมนเวลาเรียกว่า *ตัวแทนปริภูมิสถานะ (state space representation)* [2] ซึ่งในช่วงนี้เองเป็นจุดเริ่มต้นของการควบคุมเหมาะที่สุด (optimal control)

> มีการออกแบบและสังเคราะห์ตัวควบคุมหลายวิธีที่ใช้ชื่อว่า การจัดสัณฐานวงรอบ เช่น QFT (Quantitative Feedback Theory) หรือ H_∞ ในบริบทนี้อ้างถึงการจัดสัณฐานวงรอบของฟังก์ชันถ่ายโอนวงเปิดโดยอาศัยการจัดรูปแผนภาพโบเด ซึ่งเป็นการออกแบบตัวควบคุมในยุคเริ่มต้น

หากต้องการสรุปในประโยคเดียว การควบคุมเหมาะที่สุดคือแขนงหนึ่งของทฤษฎีระบบควบคุมที่เกี่ยวข้องกับการหาตัวควบคุมสำหรับระบบพลวัตที่ต้องการโดยหาค่าเหมาะที่สุดของฟังก์ชันวัตถุประสงค์ ภายใต้เงื่อนไขบังคับที่กำหนด การหาค่าเหมาะที่สุดมีการประยุกต์ใช้งานในหลายสาขาทั้งด้านวิทยาศาสตร์ วิศวกรรมศาสตร์ การวิจัยเชิงปฏิบัติการ เศรษฐศาสตร์ และสาขาอื่นที่สามารถจัดรูปโจทย์ปัญหาให้สอดคล้องกับวิธีการหาคำตอบนี้

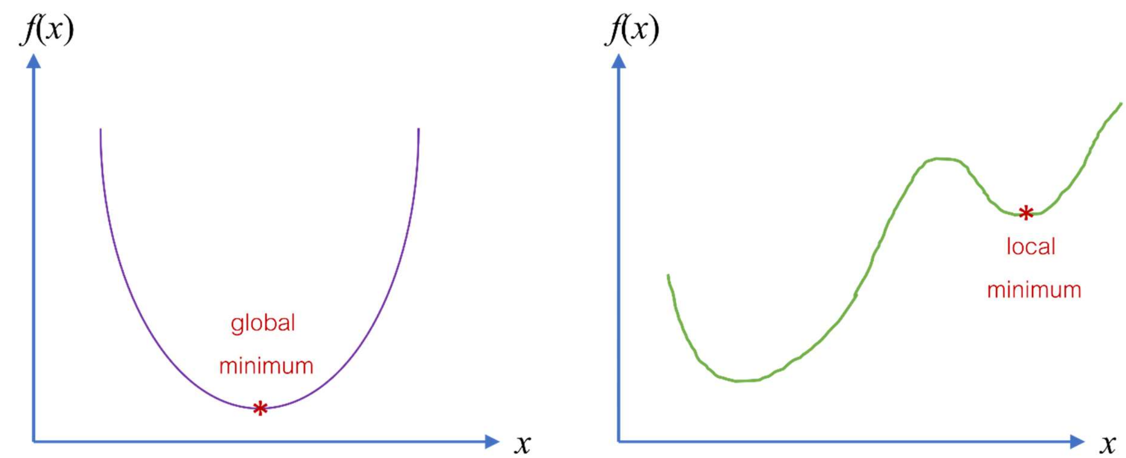
ในช่วงเริ่มต้นที่คอมพิวเตอร์ยังไม่มีสมรรถนะสูงเท่าปัจจุบัน การออกแบบตัวควบคุมจะเป็นลักษณะออฟไลน์ คือกำหนดฟังก์ชันวัตถุประสงค์จากความต้องการของโจทย์ปัญหา ที่ประกอบด้วยพจน์ของตัวแปรสถานะและตัวแปรควบคุม เช่นต้องการขับเคลื่อนยานพาหนะให้ถึงที่หมายเร็วที่สุดโดยใช้เชื้อเพลิงน้อยสุด หลังจากนั้นใช้คอมพิวเตอร์หาคำตอบที่เหมาะที่สุดได้เป็นตัวควบคุมคงที่เพื่อนำไปใช้ในระบบป้อนกลับ วิธีการนี้มีชื่อเรียกว่า *การควบคุมกำลังสองเชิงเส้น (linear quadratic control)* ซึ่งในบทความหรือหนังสือหลายเล่มจะใช้ชื่อ *LQR: Linear Quadratic Regulator* หรือในกรณีที่มีการประมาณค่าสถานะของพลานต์จะขยายเป็นวิธี *การควบคุมกำลังสองเชิงเส้นแบบเกาส์เซียน (LQG: Linear Quadratic Gaussian)*

ในการสังเคราะห์ตัวควบคุมสมัยใหม่อาจเรียก LQG ว่า H_2 เพื่อใช้กรอบการออกแบบร่วมกับ H_∞

โดยหลักการแล้วการหาคำตอบของ LQR, LQG จะมีการวนรอบเพื่อหาค่าเหมาะที่สุดของอัตราขยาย สามารถจัดได้เป็นกรณีเฉพาะของการหาค่าเหมาะที่สุดแบบคอนเวกซ์ (convex optimization) [3] แต่ในอดีตการแก้ปัญหาในลักษณะออนไลน์ คือหาคำตอบทุกช่วงเวลาการสุ่มของตัวควบคุมมีการใช้งานน้อย เนื่องจากถูกจำกัดโดยสมรรถนะของคอมพิวเตอร์ในเวลานั้นโดยมีการใช้กับระบบที่มีแบนด์วิดท์ต่ำเช่นการควบคุมกระบวนการ (process control)

ตัวอย่างหนึ่งคือ *การควบคุมแบบทำนายโมเดล (MPC: Model Predictive Control)* [4] ที่มีการนำเสนอตั้งแต่ประมาณปี ค.ศ. 1970 แต่ยังไม่มีการใช้งานแพร่หลายในยุคนั้น การหาคำตอบเหมาะที่สุดของ MPC โดยตัวประมวลผลสมรรถนะต่ำอาจต้องการคาบเวลานานหลายนาที แต่โดยสมรรถนะคอมพิวเตอร์ในปัจจุบัน (โดยเฉพาะเมื่อใช้ตัวประมวลผลเช่น GPU จำนวนมาก) อัลกอริทึม MPC สามารถใช้ในระบบที่ตอบสนองอย่างรวดเร็วเช่นการควบคุมหุ่นยนต์

อย่างไรก็ตาม การหาคำตอบเหมาะที่สุดในงานทั่วไปมิได้ถูกจำกัดเฉพาะฟังก์ชันแบบคอนเวกซ์ที่สามารถได้คำตอบรวดเร็ว และคำตอบเป็นค่าต่ำสุดแบบวงกว้าง (global minimum) ในกรณีที่ฟังก์ชันไม่เป็นคอนเวกซ์ เวลาในการคำนวณไม่สามารถคาดเดาได้ และคำตอบอาจติดอยู่กับค่าต่ำสุดเฉพาะที่ (local minimum) กราฟ 2 มิติในรูปที่ 1.1 แสดงค่าต่ำสุดแบบวงกว้างของฟังก์ชันคอนเวกซ์ และค่าต่ำสุดเฉพาะที่ของฟังก์ชันที่ไม่เป็นคอนเวกซ์



รูปที่ 1.1 ค่าต่ำสุดแบบวงกว้างและค่าต่ำสุดเฉพาะที่

ดังนั้นจึงเป็นถือเป็นประเด็นสำคัญในการพิจารณาฟังก์ชันวัตถุประสงค์และเงื่อนไขบังคับทั้งหมดโดยละเอียด เพื่อจัดรูปและเลือกตัวแก้ปัญหา (solver) ที่เหมาะสม และตัดสินว่าสามารถหาคำตอบได้ภายในเวลาที่กำหนดหรือไม่

1.2 ปัญหาการหาค่าเหมาะที่สุด

ในส่วนนี้จะกล่าวถึงพื้นฐานและหลักการทั่วไปของการหาค่าเหมาะที่สุดโดยยังไม่ลงลึกทางด้านระบบควบคุม ดังได้กล่าวแล้วว่าปัญหาการหาค่าเหมาะที่สุดสามารถประยุกต์ใช้งานได้อย่างกว้างขวาง แม้กระทั่งในระบบเดียวกันเช่นหุ่นยนต์หรือโดรน อาจใช้การหาค่าเหมาะที่สุดสำหรับตัวควบคุมและการวางแผนเส้นทางเดิน ให้ทำงานร่วมกันในลักษณะซ้อนวงกันก็ได้ ด้านเศรษฐศาสตร์มีการใช้งานด้านการลงทุน การจัดการความเสี่ยง หรือระบบลอจิสติกในงานอุตสาหกรรม ใน [3] ได้กล่าวถึงโจทย์ปัญหาในหลากหลายสาขาที่สามารถจัดรูปเป็นการหาค่าเหมาะที่สุดได้

1.2.1 หลักการของความเหมาะที่สุด

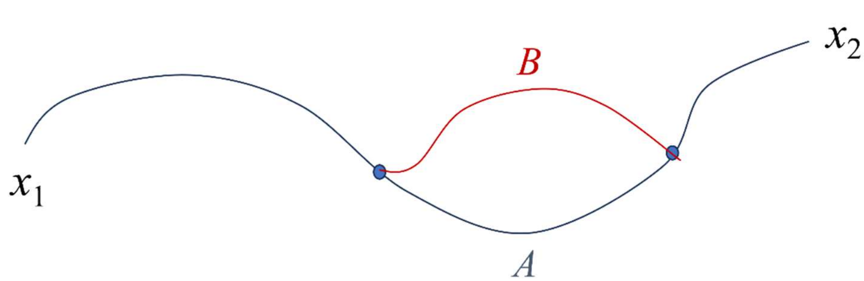
หนึ่งในผลงานเริ่มต้นที่สร้างพื้นฐานให้กับการหาค่าเหมาะที่สุดและการเรียนรู้เสริมกำลัง คือหนังสือที่เขียนโดยเบลแมนในหัวข้อ การโปรแกรมพลวัต (dynamic programming) [5] ที่ช่วยให้สามารถจัดรูปปัญหาเพื่อสร้างอัลกอริทึมสำหรับประมวลผลโดยคอมพิวเตอร์ได้ กล่าวโดยสรุปได้เป็นการแบ่งปัญหาที่ซับซ้อนเป็นโครงสร้างซ้อนในของปัญหาย่อย (subproblems) ที่สามารถหาคำตอบได้ง่ายขึ้นโดยการเรียกซ้ำ (recursive) ผ่านความสัมพันธ์ที่เรียกตามชื่อของผู้ให้กำเนิดว่า *สมการของเบลแมน (Bellman equation)* นอกจากนั้นเบลแมนยังได้มีส่วนสำคัญในทฤษฎีควบคุมเวลาต่อเนื่องในสมการอนุพันธ์ย่อย (partial differential equation) ที่เรียกชื่อว่าสมการ HJB: Hamilton-Jacobi_bellman [6]

เบลแมนได้นำเสนอ *หลักการของความเหมาะที่สุด (principle of optimality)* ทำให้สามารถแบ่งแยกปัญหาการตัดสินใจรวมเป็นปัญหาย่อยได้ นิยามใน [5] กล่าวไว้ดังนี้

นโยบายเหมาะที่สุดต้องมีคุณสมบัติคือ ไม่ว่าจะเลือกสถานะและการตัดสินใจเริ่มต้นอย่างไร การตัดสินใจครั้งต่อไปที่เหลือจะต้องยังคงเป็นนโยบายเหมาะที่สุดเสมอ เมื่อนับจากสถานะที่เกิดจากการตัดสินใจครั้งแรก

เบลแมนใช้คำว่า “นโยบาย” แทนการกระทำ หรือการตัดสินใจ ต่อมาถูกใช้ในบทความเกี่ยวกับการเรียนรู้เสริมกำลัง สำหรับในสาขาระบบควบคุมนโยบายก็คือเอาต์พุตของตัวควบคุม

นิยามนี้อาจมีลักษณะเป็นนามธรรม จะอธิบายให้เข้าใจได้ง่ายขึ้นดังในรูปที่ 1.2 สมมุติว่านโยบายในการเคลื่อนที่จากจุด x_1 ไปยังจุด x_2 โดยผ่านเส้นทาง A คือนโยบายเหมาะที่สุดตามวัตถุประสงค์ที่ต้องการ เช่นใช้เวลาน้อยสุด ดังนั้นจะไม่สามารถพบเส้นทางลัด B ที่ทำให้



รูปที่ 1.2 หลักการของความเหมาะที่สุด

การเดินทาง เร็วกว่าเส้นทางที่ผ่าน A ได้ มิฉะนั้นเราจะไม่สามารถเรียกเส้นทางที่ผ่าน A ว่าเป็นนโยบายเหมาะที่สุด

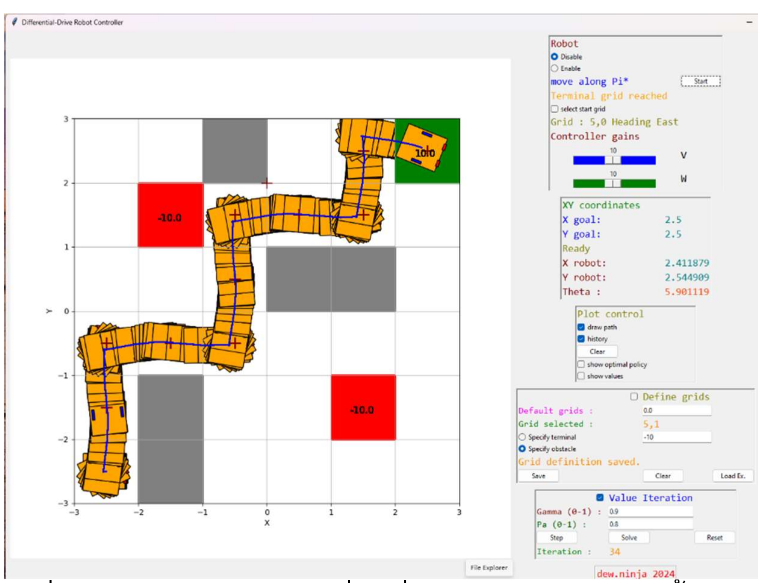
หลักการของความเหมาะที่สุดเป็นหัวใจสำคัญที่ทำให้สามารถใช้วิธีการโปรแกรมพลวัตได้ ตัวอย่างเช่นในการแก้ปัญหาย่อยจากส่วนท้ายมายังต้น ต้องมั่นใจว่านโยบายส่วนท้ายต้องเป็นแบบเหมาะที่สุดโดยไม่ขึ้นกับการตัดสินใจในส่วนต้น

ตัวอย่าง 1.1 ในบทที่ 8 ของหนังสือ [7] ได้แสดงตัวอย่างขั้นพื้นฐานของการโปรแกรมพลวัตในการวางแผนเส้นทางหุ่นยนต์เคลื่อนที่บนกริดเวิลด์โดยวิธีการที่เรียกว่า *การวนซ้ำมูลค่า (value iteration)* อาศัยสมการของเบลแมน

$$v(s) = r(s) + \gamma \max_a \sum_{s'} p(s'|s,a)v(s') \tag{1.1}$$

โดย $r(s)$ แทนรางวัลของสถานะ s ค่า γ คือตัวแปรประกอบส่วนลดในช่วง 0 – 1 และ $p(s'|s,a)$ คือความน่าจะเป็นในการเลือกการกระทำ a เพื่อเปลี่ยนสถานะจาก s เป็น s'

นิยามกริดเป้าหมาย กริดที่มีมูลค่าลบ (แทนพื้นที่อันตรายที่ไม่ต้องการให้หุ่นยนต์เข้าถึง) และกริดสิ่งกีดขวาง กำหนดค่าเริ่มต้นใดๆ (เช่นค่าศูนย์) ให้กับสถานะคือแต่ละตารางบนกริดเวิลด์ เลือกค่า γ และ $p(s'|s,a)$ และใช้วิธีการวนซ้ำมูลค่าที่จะทำให้มูลค่าลู่เข้าสู่ค่าคำตอบนโยบายเหมาะที่สุดได้จากทิศทางการเคลื่อนที่เพื่อได้รางวัลสูงสุด รูปที่ 1.3 แสดงตัวอย่างหนึ่งของหุ่นยนต์ที่เคลื่อนที่ตามนโยบายเหมาะที่สุดจากตำแหน่งเริ่มต้นสู่เป้าหมาย



รูปที่ 1.2 การวางแผนการเคลื่อนที่หุ่นยนต์โดยวิธีการวนซ้ำมูลค่า

ประเด็นหนึ่งที่น่าสนใจในการวางแผนเส้นทางเดินหุ่นยนต์จากตัวอย่าง 1.1 คือเมื่อแปรค่าพารามิเตอร์ γ และ $p(s'|s,a)$ จะสามารถเปลี่ยนแปลงนโยบายเหมาะที่สุด เสมือนว่าหุ่นยนต์มีความฉลาดที่จะตัดสินใจขึ้นกับอัตราการลดลงของรางวัลและความแม่นยำในการเคลื่อนที่

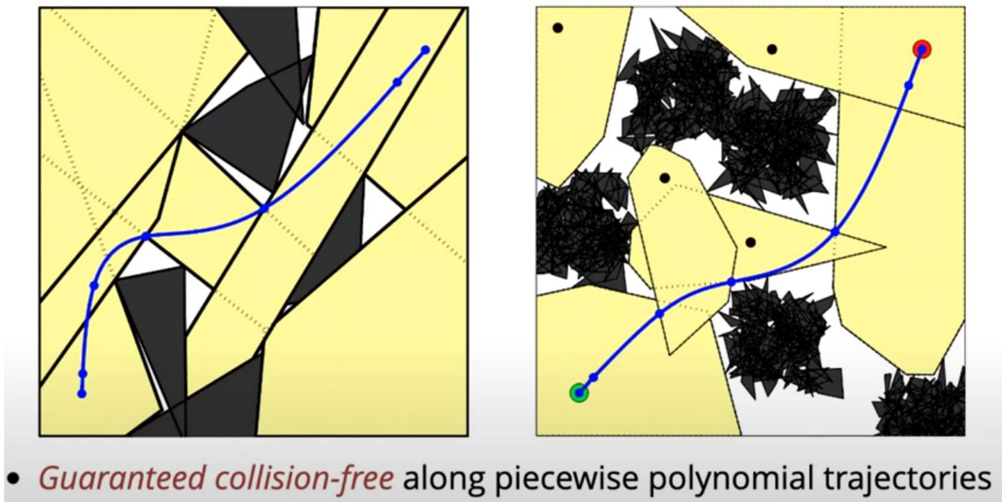
ในหนังสือหลายเล่มเช่น [8] ใช้ตัวอย่างการแก้ปัญหากริดเวิลด์เป็นพื้นฐานสู่เนื้อหาหลักของการเรียนรู้เสริมกำลัง (reinforcement learning) เมื่อเพิ่มความไม่แน่นอนแบบสโทแคสติก นอกจากนั้นพบว่าการหาคำตอบของปัญหากริดเวิลด์จะขึ้นกับจำนวนของสถานะคือกริด เมื่อมีจำนวนมากขึ้น การคำนวณจะเพิ่มมากขึ้นตามจนประสบปัญหาที่เบลแมนเรียกว่า คำสาปของมิติ (curse of dimensionality)

ถึงจุดนี้ต้องการชี้ประเด็นหนึ่งที่แสดงความสัมพันธ์ระหว่างการควบคุมเหมาะที่สุดและการเรียนรู้เสริมกำลัง ซึ่งทำให้ทั้งสองมารวมอยู่ในชื่อหนังสือเล่มนี้ได้ ในปัญหาการหาค่าเหมาะที่สุดที่กล่าวถึงตั้งแต่ต้นเราต้องการหาคำตอบที่ทำให้ฟังก์ชันวัตถุประสงค์มีค่าน้อยสุด แต่ในสมการเบลแมน (1.1) เป็นตรงข้ามกันคือ นโยบายเหมาะที่สุดของหุ่นยนต์สอดคล้องกับเส้นทางที่ทำให้ได้รางวัลสูงสุด สิ่งที่เหมือนเป็นตรงข้ามกันนี้แท้จริงแล้วอยู่บนพื้นฐานหรือหลักการเดียวกัน

คำกล่าวติดตลกที่ผู้เขียนชอบใช้คือผู้ศึกษาการเรียนรู้เสริมกำลังคือผู้มองโลกในแง่ดี (optimistic) ส่วนผู้ศึกษาการควบคุมเหมาะที่สุดคือผู้มองโลกในแง่ร้าย (pessimistic) 😊

ตัวอย่าง 1.2 วิธีการกริดเวิลด์ในตัวอย่าง 1.1 เป็นเหมือนโจทย์ปัญหาของเด็กเล่นเพื่อให้เข้าใจการโปรแกรมพลวัตและปูทางสู่การเรียนรู้เสริมกำลัง ข้อด้อยนอกเหนือจากคำสาปของมิติคือหุ่นยนต์ถูกบังคับให้เคลื่อนที่ใน 4 ทิศทางตั้งฉาก คือเหนือ ตะวันออก ใต้ ตะวันตกเท่านั้น ซึ่งเป็นข้อจำกัดสำหรับหุ่นยนต์หรือโดรนที่ต้องการเคลื่อนที่หลบหลีกสิ่งกีดขวางอย่างรวดเร็ว

วิธีการหนึ่งที่ถูกนำเสนอจากห้องปฏิบัติการด้านหุ่นยนต์ของ MIT และกำลังได้รับความสนใจอย่างมากในปัจจุบันเรียกว่า *กราฟของเซตคอนเวกซ์ (GCS : Graph of Convex Set)* [9] ซึ่งสามารถใช้ในการวางแผนเส้นทางเดินของหุ่นยนต์ อธิบายโดยย่อคือแบ่งพื้นที่ว่างที่หุ่นยนต์สามารถเคลื่อนที่จากตำแหน่งเริ่มต้นไปยังเป้าหมายโดยไม่ชนสิ่งกีดขวางเป็นเซตคอนเวกซ์ย่อยดังแสดงในรูปที่ 1.3 โดยคุณสมบัติของเซตคอนเวกซ์คือเมื่อกำหนดจุด 2 จุดใดๆ ในเซต เส้นตรงระหว่างสองจุดนี้จะต้องอยู่ในเซต ดังนั้นโดยหลักการรับประกันได้ว่าทุกเวลาที่หุ่นยนต์อยู่ในเซตคอนเวกซ์จะไม่ซ้อนทับกับสิ่งกีดขวาง



รูปที่ 1.3 การวางแผนเส้นทางโดยวิธี GCS

[https://youtu.be/KSCC7mVJzaw?si=1rH8ARNq0B48Qduu]

ในรูป 1.3 พิจารณาวัตถุเคลื่อนที่เป็นจุดเดียว ในการใช้งานทางปฏิบัติคงต้องพิจารณาปัจจัยอื่นเช่น รูปทรงของวัตถุ เช่นหากมีส่วนปีกของอากาศยานที่แผ่กว้างออกไปจนออกนอกพื้นที่คอนเวกซ์

นอกจากนั้น การสร้างพื้นที่ย่อยเป็นเซตคอนเวกซ์ช่วยให้คำนวณหาคำตอบเหมาะที่สุดได้เร็ว ทำให้การวางแผนเส้นทางเดินนี้สามารถทำได้แบบเรียลไทม์ เป็นการประยุกต์ใช้การหาค่าเหมาะที่สุดในการวางแผนเส้นทางเดินที่สามารถใช้งานได้จริงในทางปฏิบัติ

1.2.2 การจัดรูปปัญหาทางคณิตศาสตร์

หากต้องการจัดรูปปัญหาการหาค่าเหมาะที่สุดที่ครอบคลุมทุกกรณี เพื่อหาคำตอบโดยวิธีทางคณิตศาสตร์ สามารถเขียนได้เป็นดังนี้

$$\begin{aligned} \min_{x} \quad & f(x) \\ s.t. \quad & c(x) \le b \end{aligned} \tag{1.2}$$

โดย min แทนการหาค่าน้อยสุด และ s.t. ย่อมาจาก subject to หมายถึง ขึ้นอยู่กับ หรือภายใต้เงื่อนไขบังคับ ในที่นี้ $x=(x_1,\ldots,x_n)$ คือ *ตัวแปรที่ต้องการหาค่าเหมาะที่สุด (optimization variables)* $f:\mathrm{R}^n \to \mathrm{R}$ คือ *ฟังก์ชันวัตถุประสงค์ (objective function)* และ $c:\mathrm{R}^n \to \mathrm{R}^m$ คือ *ฟังก์ชันเงื่อนไขบังคับ (constraint function)* และเวกเตอร์ค่าคงที่ b คือค่าจำกัดหรือขอบเขต เวกเตอร์ x^* เรียกว่าเหมาะที่สุด (optimal) หรือเป็นคำตอบของ (1.2) ถ้าทำให้ค่าวัตถุประสงค์มี

ค่าน้อยที่สุดในเวกเตอร์ทั้งหมดที่สอดคล้องกับเงื่อนไขบังคับ หากไม่สามารถหาค่าของ x^* ได้เรียกว่าเป็นปัญหาที่เป็นไปไม่ได้ (infeasible)

> นิยามและสัญนิยมที่ใช้ในแต่ละบทความ/หนังสืออาจมีความแตกต่างกันในรายละเอียด เช่นอาจแยกแต่ละฟังก์ชันเงื่อนไขบังคับ โดยเฉพาะกรณีที่มีทั้งสมการและอสมการ หรือเขียนเป็น $c(x) \le 0$ ซึ่งได้จากการย้าย b ใน (1.2) ไปทางด้านซ้าย หรือ $c(x) \ge 0$ สมมูลกับ $-c(x) \le 0$

ในกรณีทั่วไปที่ $f(x)$ และ $c(x)$ เป็นฟังก์ชันใดๆ การหาคำตอบอาจมีความซับซ้อนและไม่สามารถรับประกันได้ว่าจะได้คำตอบภายในเวลาที่กำหนด เราสนใจเป็นกรณีพิเศษกับปัญหาการหาค่าเหมาะที่สุดแบบคอนเวกซ์ ที่มีฟังก์ชันวัตถุประสงค์และเงื่อนไขบังคับเป็นแบบคอนเวกซ์สอดคล้องกับอสมการ

$$f(\alpha x + \beta y) \le \alpha f(x) + \beta f(y) \tag{1.3}$$

สำหรับทุกค่า $x, y \in \mathrm{R}^n$ และ $\alpha, \beta \in \mathrm{R}$ โดย $\alpha + \beta = 1,\ \alpha \ge 0,\ \beta \ge 0$ สังเกตว่าฟังก์ชันคอนเวกซ์จะครอบคลุมฟังก์ชันเชิงเส้นและสัมพรรคด้วย

1.3 เครื่องมือซอฟต์แวร์

ในหนังสือนี้จะใช้เครื่องมือซอฟต์แวร์แบบโอเพนซอร์ส ที่ผู้ใช้ทั่วไปสามารถดาวน์โหลดมาติดตั้งบนเครื่องคอมพิวเตอร์ของตัวเอง จะเน้นภาษาไพทอนเป็นหลักเนื่องจากมีผู้ใช้จำนวนมากและมีชุมชน (community) ขนาดใหญ่ที่สามารถสืบค้นข้อมูลได้ง่าย ภาษาใหม่ที่น่าสนใจคือจูเลียมีจุดเด่นเช่นสมรรถนะการคำนวณ ความสะดวกในการหาอนุพันธ์อัตโนมัติ (automatic differentiation) แต่ยังมีชุมชนขนาดเล็กเมื่อเทียบกับไพทอน

ในการใช้งานไพทอนมักต้องติดตั้งแพ็กเกจช่วยในการคำนวณเฉพาะงาน เช่น JAX [10] ช่วยการหาอนุพันธ์อัตโนมัติ CVXPY [11] สำหรับปัญหาการหาค่าเหมาะที่สุดแบบคอนเวกซ์ หรือ Drake [12] ที่เน้นการใช้งานด้านหุ่นยนต์ หากมีการใช้งานแพ็กเกจอื่นจะกล่าวถึงในส่วนนั้น เนื่องจากซอฟต์แวร์บางตัวเช่น Drake ไม่สามารถใช้งานได้หรือทำงานช้าบนระบบคลาวด์ เช่น Colab ดังนั้นควรติดตั้งไพทอนและจูปิเตอร์บนเครื่องคอมพิวเตอร์ของเรา หรือใช้ Docker ซึ่งเป็นวิธีการแนะนำสำหรับผู้เริ่มต้น

ตัวอย่าง 1.3 ในตัวอย่างนี้จะสาธิตการใช้แพ็กเกจย่อย MathematicalProgram ในเครื่องมือซอฟต์แวร์ Drake หาคำตอบเหมาะที่สุดของปัญหาโปรแกรมกำลังสอง (quadratic program) ซึ่งต่อไปจะเรียกย่อว่า QP รูปแบบปัญหาโดยทั่วไปคือ

$$\begin{aligned} \min_{x} \quad & \frac{1}{2} x^T Q x + q^T x + k \\ s.t. \quad & Ax \le b, \\ & Cx = d \end{aligned} \tag{1.4}$$

จะเห็นว่าฟังก์ชันมูลค่าเป็นฟังก์ชันกำลังสองของ x คือตัวแปรในการตัดสินใจ k เป็นค่าคงที่ที่สามารถละทิ้งได้เนื่องจากไม่มีผลกับคำตอบของการหาค่าเหมาะที่สุด Q คือเมทริกซ์สมมาตรและหากเป็นแบบกึ่งบวกแน่นอน จะเรียกว่าปัญหา QP แบบคอนเวกซ์ เงื่อนไขบังคับทั้งหมดเป็นแบบเชิงเส้น

> ในหนังสือนี้มักใช้คำว่าเชิงเส้น (linear) ซึ่งเป็นศัพท์ไทยที่คุ้นเคยมากกว่าสัมพรรค (affine) แม้ว่าอาจไม่ถูกใจนักคณิตศาสตร์นัก นอกจากว่าเป็นกรณีเฉพาะที่ความแตกต่างมีความสำคัญ

พิจารณาโจทย์ QP พื้นฐานดังนี้ [13]

$$\begin{aligned} \min_{x} \quad & x_0^2 + x_1^2 + x_2^2 \\ s.t. \quad & \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} \le \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}, \\ & \begin{bmatrix} 2 & 3 & 1 \\ 5 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{aligned} \tag{1.5}$$

เริ่มโดยนำเข้าแพ็กเกจที่ต้องการ

```
# python libraries
import numpy as np
from pydrake.all import MathematicalProgram, Solve, eq, ge, le
```

สร้างวัตถุ `prog` จาก `MathematicalProgram()` กำหนดจำนวนตัวแปรตัดสินใจ `x` คือ 3 เพิ่มฟังก์ชันมูลค่า และเงื่อนไขบังคับ หลังจากนั้นหาคำตอบโดยใช้ `Solve(prog)`

```
prog = MathematicalProgram()
x = prog.NewContinuousVariables(3)

prog.AddCost(x.dot(x))
prog.AddConstraint(eq(np.array([[2, 3, 1], [5, 1, 0]]).dot(x), [1, 1]))
prog.AddConstraint(le(x, 2 * np.ones(3)))

result = Solve(prog)

# Get the solution
if result.is_success():
    print("Solution: " + str(result.GetSolution()))
```

```
Solution: [0.15897436 0.20512821 0.06666667]
```

1.4 สรุปท้ายบท

เนื้อหาในบทนี้อธิบายแนวคิดของการควบคุมเหมาะที่สุดที่มีความสัมพันธ์กับการเรียนรู้เสริมกำลัง หลักการพื้นฐานและการจัดรูปปัญหา และแนะนำเครื่องมือซอฟต์แวร์ที่สามารถใช้ในการหาคำตอบ

(เพิ่มเติมโครงสร้างของหนังสือในภายหลัง)

บรรณานุกรม

1. K.J. Åström and R. M. Murray. Feedback Systems: An Introduction for Scientists and Engineers, 2nd ed. Princeton University Press. 2020. https://fbswiki.org

2. R.E. Kalman. *A new approach to linear filtering and prediction problems*. Transactions of the ASME, Journal of Basic Engineering, 82:34–45, 1960.

3. S.P Boyd and L. Vandenberghe, Convex Optimization. Cambridge University Press. 2004. https://web.stanford.edu/~boyd/cvxbook/

4. M. Morari and J.H. Lee, *Model predictive control: past, present, and future.* Computers & Chemical Engineering, 23: 667—682, 1999.

5. R.E. Bellman. Dynamic Programming. Princeton University Press. 1957.

6. J. Yong and X.Y. Zhou. *Dynamic Programming and HJB Equations*. Stochastic Controls : Hamiltonian Systems and HJB Equations. Springer. pp. 157 – 215. 1999.

7. วโรดม ตู้จินดา. การโปรแกรมไพทอนสำหรับงานควบคุมและฝังตัว. ดิว นินจา. 2567.

https://github.com/dewdotninja/py4conemb

8. R.S. Sutton and A.G. Barto. Reinforcement Learning: An Introduction. 2nd ed. MIT Press. 2020. http://incompleteideas.net/book/the-book-2nd.html

9. T. Marcucci, J. Umengerger, P.A. Pablo and R. Tedrake. *Shortest Paths in Graphs of Convex Sets*. SIAM Journal on Optimization, 34(1): 507 – 532. 2024.

10. R. Frostig, M.J.Johnson and C. Leary. *Compiling machine learning programs via high-level tracing*. MLsys: 1-3. 2018.

11. S. Diamond and S.P.Boyd. *Python-embedded modeling language for convex optimization*. Journal of Machine Learning Research. 83(17):1-5. 2016.

12. R. Tedrake. *Drake: Model-based design in the age of robotics and machine learning.* Toyota Research Institute. 2021.

13. R. Tedrake. Robotic Manipulation: Perception, Planning, and Control. Course Notes for MIT 6.421. 2024. http://manipulation.mit.edu

โจทย์ปัญหา

1-1 ใช้ Drake แก้ปัญหา QP ตามโจทย์ดังนี้

$$
\begin{aligned}
&\min_{x} && 2x_0^2 + x_1^2 + 5x_2^2 \\
&s.t. && |x| \le \begin{bmatrix} 0.65 \\ 0.65 \\ 0.65 \end{bmatrix},
\end{aligned}
$$

$$
\begin{bmatrix} 1 & 4 & 3 \\ 2 & 7 & 8 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \end{bmatrix} \qquad \text{(P1.1)}
$$