บทที่ 2 ระเบียบวิธีวิจัย

2.1. ระบบที่ถูกใช้ในการควบคุม

โดยทั่วไปแล้วระบบต่างๆที่เราเจอนั้นเป็นระบบที่ไม่เป็นเชิงเส้น ซึ่งมีความซับซ้อนในการพิจารณาและยากกับการออกแบบตัวควบคุมที่จะนำมาใช้ในการควบคุมระบบ ดังนั้นเพื่อให้ง่ายกับการพิจารณา เราสามารถทำการประมาณระบบต่างๆให้เป็นแบบเชิงเส้นในช่วงของจุดทำงานที่เราสนใจได้ โดยในการออกแบบระบบควบคุมที่จะกล่าวถึงต่อไปนั้น ก็จะอ้างอิงถึงระบบเชิงเส้นที่จะอธิบายในหัวข้อนี้

2.1.1. ระบบเชิงเส้นที่ไม่แปรผันตามเวลา (Linear time invariant system)

ดังที่ได้กล่าวมาข้างต้นแล้วว่าระบบ feedback control system โดยทั่วไปแล้วจะถูกประมาณให้เป็นเชิงเส้นเพื่อจะได้นำไปใช้ในการออกแบบการควบคุมระบบได้ง่าย ในหัวข้อนี้เราจะพิจารณาระบบเชิงเส้นที่ไม่มีการเปลี่ยนแปลงตามเวลา หรือ Linear time invariant system โดยสามารถเขียนเป็นสมการในรูปแบบของ state space ได้ดังนี้

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k+1) &= Cx(k+1) \end{aligned} \tag{1}$$

โดย $x(k)$ คือ state variable ที่ time step k ส่วน $u(k)$ คือ input trajectory ที่ time step k ซึ่งเป็นสัญญาณที่เราต้องการ track ตาม และในที่นี้ $y(k)$ คือ output trajectory ที่ time step k ซึ่งเป็นสัญญาณ output ของระบบที่เราต้องการให้มีค่าเท่ากับ $u(k)$ โดยถ้าเราต้องการเขียนความสัมพันธ์ระหว่าง input และ output โดยให้อยู่ในรูปแบบ matrix equation นั้น เราสามารถเขียนได้เป็น

$$\underline{y} = \overline{A}x(0) + P\underline{u} \tag{2}$$

$$\underline{y} = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(p) \end{bmatrix}, \quad \underline{u} = \begin{bmatrix} u(0) \\ u(1) \\ \vdots \\ u(p-1) \end{bmatrix}, \quad P = \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{p-1}B & CA^{p-2}B & \cdots & CB \end{bmatrix}, \quad \overline{A} = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^p \end{bmatrix}$$

$x(0)$ คือค่าของ initial state ที่เราต้องทำการสมมติขึ้นมา โดยทั่วไปแล้ว เราจะกำหนดให้ค่านี้มีค่าเป็นศูนย์เพื่อง่ายแก่การพิจารณา ซึ่งถ้าเทอมนี้มีค่าเป็นศูนย์ จะเห็นได้ว่าความสัมพันธ์ของ input และ output นั้นถูกกำหนดค่าโดยเมตริกซ์ P ซึ่งมีค่า CB ใน main diagonal line และ CAB, CA^2B, ... ใน lower diagonal ตามลำดับ ค่าต่างๆในเมตริกซ์ P นี้เป็นค่าที่เรียกกันว่า Markov parameter ของระบบ ซึ่งเป็นค่าที่เราต้องทำการหาขึ้นมาก่อนที่จะเริ่มทำการควบคุมระบบ

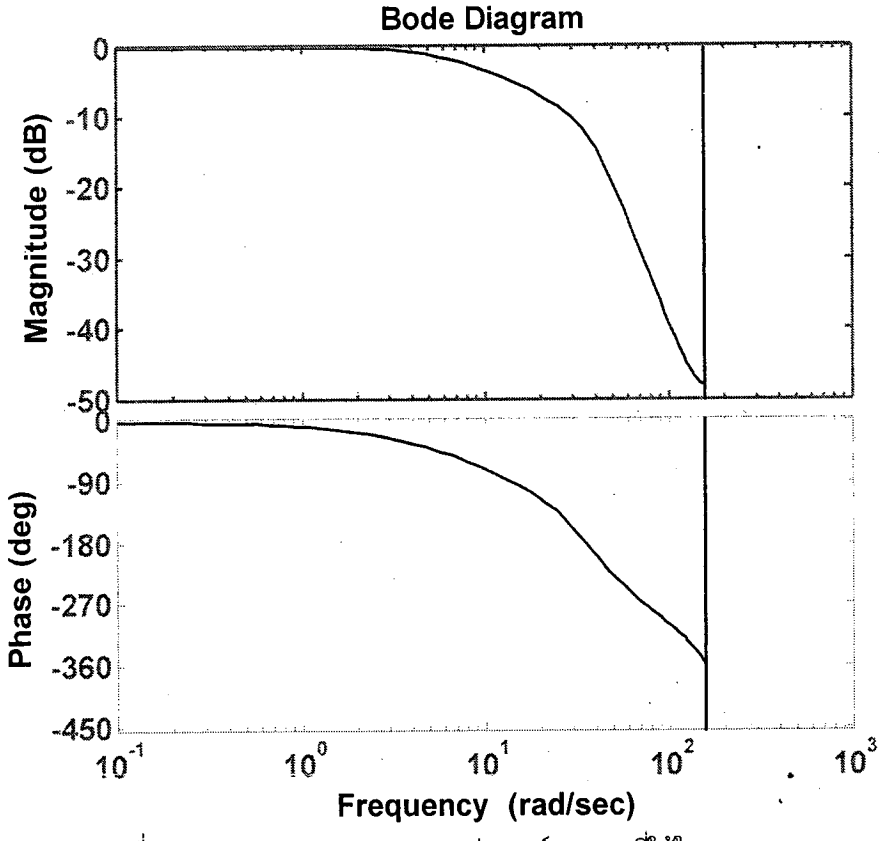
จะเห็นได้ว่าระบบนี้ถูกสร้างขึ้นโดย input จะเริ่มตั้งแต่ time step ที่ศูนย์ ซึ่งจะทำให้เกิด output ที่ time step ที่หนึ่ง ลักษณะดังกล่าวเกิดจากกระบวนการการทำ discretization ของระบบจาก continuous time ไปเป็น discrete time โดยใช้ Zero Order Hold (ZOH) ซึ่งกระบวนการดังกล่าวจะทำให้เกิด time delay เท่ากับหนึ่งเสมอในระบบ discrete time ใดๆที่มีการใช้ ZOH เนื่องจากระบบ continuous time ใดๆมักจะไม่มี zero หรือถ้ามี zero จำนวน zero ก็มักจะมีค่าน้อยกว่าจำนวน pole อยู่ 1 เสมอ ดังนั้น extra pole 1 ตัวที่เกิดขึ้นใน discrete time system ก็คือ image ของ extra zero ใน continuous time system นั่นเอง [1]

2.1.2. ระบบหุ่นยนต์แขนกล

ระบบที่เราจะใช้ในการทำ numerical simulation ในงานวิจัยนี้ จะเป็นโมเดลหุ่นยนต์แขนกลของบริษัท NASA Langley Research Center โดยเราจะพิจารณาเพียงแค่ 1 ข้อต่อของหุ่นยนต์ สมการทางคณิตศาสตร์ หรือ Transfer function ของหนึ่งข้อต่อนี้สามารถประมาณให้อยู่ในรูป 3^{rd} order model ได้เป็น

$$G(s)=\left(\frac{a}{s+a}\right)\left(\frac{\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}\right) \tag{3}$$

ซึ่งค่า parameter ต่างๆมีค่าเป็น $a=8.8,\ \zeta=0.5,\ \omega_n=37$ rad/sec เมื่อจะนำระบบนี้ไปใช้ต่อ เราจะทำการแปลงระบบให้อยู่ในรูปของ Z-domain โดยผ่าน A/D (ZOH) ในที่นี้เราจะใช้ค่า sampling time ซึ่งเป็นค่า default เท่ากับ 1/50 second รูปที่ 2.1 แสดงขนาดและเฟสของระบบหุ่นยนต์แขนกลใน discrete time domain



รูปที่ 2.1 ขนาดและเฟสของระบบหุ่นยนต์แขนกลที่ใช้ในการทดลอง

2.2 ทฤษฎีพื้นฐานของระบบควบคุมจากการทำซ้ำ

ระบบควบคุมแบบเรียนรู้จากการทำซ้ำเริ่มมีการนำมาใช้ในปี ค.ศ. 1978 โดยได้รับแรงดลใจมาจากการใช้งานหุ่นยนต์ที่ทำงานแบบซ้ำไปซ้ำมาในภาคอุตสาหกรรม [2] แต่เนื่องจากผลงานที่ตีพิมพ์นี้เผยแพร่เป็นภาษาญี่ปุ่นเพียงอย่างเดียว แนวความคิดจึงไม่ได้กระจายไปอย่างแพร่หลายนัก จนกระทั่งในปี ค.ศ. 1984 ได้มีการศึกษางานวิจัยทางด้านนี้อย่างจริงๆจังๆจากบุคคลต่างๆในแต่ละมุมโลก [3-6] ซึ่งได้รับแรงดลใจจากการใช้งานในรูปแบบการทำซ้ำของงานประเภทที่แตกต่างกันออกไป

ในระบบการควบคุมแบบป้อนกลับ (Feedback Control System) จะมีค่าผิดพลาดเกิดขึ้นสำหรับทุกๆสัญญาณอินพุทที่ป้อนเข้าไป โดยถ้าป้อนสัญญาณอินพุทค่าเดิมเข้าไปหลายๆรอบ ก็จะเกิดค่าผิดพลาดซ้ำเดิมขึ้นทุกๆรอบ ดังนั้นในการใช้งานในรูปแบบของคำสั่งที่มีการทำซ้ำ ทฤษฎีการควบคุมแบบเรียนรู้จากการทำซ้ำสามารถถูกนำมาใช้ เพื่อลดค่าผิดพลาดที่เกิดขึ้นในแต่ละรอบของการทำงาน ให้มีค่าลดลงจนลู่เข้าสู่ศูนย์ (diverge to zero) ซึ่งระบบการควบคุมแบบเรียนรู้จากการทำซ้ำจะมีอยู่สองประเภทคือ ระบบควบคุมแบบเรียนรู้ที่ทำแบบต่อเนื่อง (Repetitive Control System) และระบบควบคุมแบบเรียนรู้ที่มีการย้อนกลับไปสู่ภาวะเริ่มต้น (Learning Control System) โดยระบบการเรียนรู้ทั้งสองประเภทนี้แตกต่างกันเพียงเงื่อนไขเริ่มต้น (initial conditions) ในการทำงานของระบบเท่านั้น ซึ่งระบบควบคุมแบบเรียนรู้ที่ทำแบบต่อเนื่องนั้นจะทำขบวนการในรอบถัดไปต่อเนื่องกันไปอย่างไม่หยุด โดยจะไม่มีการกำหนดให้เงื่อนไขเริ่มต้นใหม่หลังจากที่

ขบวนการเสร็จสิ้นลงในแต่ละรอบ ส่วนระบบควบคุมแบบเรียนรู้ที่มีการย้อนกลับไปสู่ภาวะเริ่มต้นจะมีการกำหนดเงื่อนไขเริ่มต้นที่เหมือนเดิมทุกครั้งก่อนที่จะมีการทำซ้ำครั้งถัดไป

ในระบบควบคุมแบบเรียนรู้ทั้งสองประเภทนั้น จะมีหลักการทำงานคล้ายๆกัน กล่าวคือ จะมีการจดจำค่าผิดพลาดที่เกิดขึ้นในการทำซ้ำครั้งปัจจุบัน เพื่อนำมาปรับค่าสัญญาณที่จะป้อนให้กับระบบในการทำงานรอบถัดไป สัญญาณที่ป้อนให้ระบบในรอบถัดไปนั้น จะเป็นสัญญาณที่จะไปกำจัดค่าผิดพลาดที่เกิดขึ้นให้มีค่าลดลงจากรอบเดิม จนกระทั่งค่าผิดพลาดที่ได้ลู่เข้าสู่ศูนย์เมื่อมีการทำซ้ำหลายๆครั้ง

2.2.1. การควบคุมแบบเรียนรู้ที่ทำแบบต่อเนื่อง (Repetitive Control)

การควบคุมแบบ Repetitive control นั้นถูกวิจัยกันอย่างแพร่หลายเพื่อนำไปใช้กับงานประยุกต์หลายๆด้าน เช่นการควบคุมการเขียน/อ่านข้อมูลของฮาร์ดดิสก์ไดร์ฟ สัญญาณที่สั่งให้หัวเขียนหัวอ่านเคลื่อนที่ไปยังแทรคข้อมูลที่ต้องการนั้นเป็นสัญญาณที่มีลักษณะเป็นคาบ และถ้าเราสามารถทำให้หัวเขียนหัวอ่านเคลื่อนที่ด้วยความผิดพลาดน้อยมากๆ เราก็สามารถที่จะเพิ่มขนาดความจุของฮาร์ดดิสก์ได้อีกด้วย

จากที่ได้กล่าวมาข้างต้นว่าการเรียนรู้จากการทำซ้ำเป็นการนำเอาสัญญาณ control input จากการทำซ้ำครั้งก่อนมาปรับปรุงโดยจะดูจากค่าความผิดพลาดจากการทำซ้ำครั้งที่แล้ว ดังนั้นรูปแบบอย่างง่ายของการควบคุมแบบ Repetitive control สามารถเขียนได้เป็น

$$u(k+p) = u(k) + \phi e(k) \tag{4}$$

p ในที่นี้คือจำนวน time steps ในหนึ่งรอบการทำซ้ำ จากสมการจะเห็นได้ว่า control input ในการทำงานรอบถัดไปจะมีค่าเป็น control input ในรอบการทำงานที่แล้ว รวมกับเทอมของค่าผิดพลาดจากการทำงานรอบที่แล้ว ถ้าเราเขียนสมการนี้ให้อยู่ในรูป Z-transform จะได้

$$U(z) = \left[\frac{\phi}{z^p - 1}\right] E(z) \tag{5}$$

และเราสามารถเขียนความสัมพันธ์ของค่าผิดพลาดจากการทำซ้ำครั้งหนึ่งไปยังการทำซ้ำอีกครั้งหนึ่งได้เป็น

$$z^p E(z) = \left[1 - F(z)G(z)\right] E(z) \tag{6}$$

จากสมการจะเห็นได้ว่าถ้าเทอม $1 - F(z)G(z)$ มีขนาดน้อยกว่าหนึ่งที่ทุกๆความถี่แล้ว ค่าความผิดพลาดในการทำซ้ำครั้งถัดๆไปจะมีค่าลดลงอย่างต่อเนื่อง ซึ่งเงื่อนไขนี้เป็นเงื่อนไขของ necessary และ sufficient conditions ของ stability ใน Repetitive control system ซึ่งเราสามารถพิจารณา stability ของระบบได้

ง่ายๆโดยดูจาก polar plot ของ $F(e^{i\omega T})G(e^{i\omega T})$ ว่าอยู่ข้างในวงกลมหนึ่งหน่วยที่มีจุดศูนย์กลางอยู่ที่ตำแหน่ง (1,0) หรือไม่

2.2.2. การควบคุมแบบเรียนรู้ที่มีการย้อนกลับไปสู่ภาวะเริ่มต้น (Iterative Learning Control)

ระบบที่มีการทำซ้ำและมีการตั้งค่าเริ่มต้นใหม่ในแต่ละขบวนการทำซ้ำนั้น จะถูกพบเห็นได้บ่อยๆในขบวนการผลิตต่างๆในโรงงาน เช่น หุ่นยนต์จับวางสิ่งของจาก station หนึ่ง ไปยังอีก station หนึ่ง ซึ่งลักษณะของงานจะเป็นแบบจับแล้ววาง (pick and place) อย่างนี้ไปเรื่อยๆ โดยในแต่ละขบวนการทำซ้ำนั้นจะมีค่าความคลาดเคลื่อน (error) เกิดขึ้นอันเนื่องมาจากปัจจัยหลายอย่าง เช่น แรงโน้มถ่วงของโลกที่กระทำต่อแขนกลในแต่ละข้อต่อ หรือ แรงที่กระทำในแต่ละข้อต่อ (centrifugal toques) ซึ่งสิ่งต่างๆเหล่านี้ ล้วนแล้วแต่เป็นสิ่งรบกวนที่เกิดขึ้นภายในระบบ (disturbance) และเมื่อเราสั่งงานให้ระบบมีการทำงานแบบซ้ำไปซ้ำมา ค่าความคลาดเคลื่อนต่างๆที่เกิดขึ้นก็จะมีค่าที่เท่ากันในแต่ละขบวนการทำซ้ำ หรืออาจกล่าวได้ว่า ถ้าเรามองระบบที่เราสนใจอยู่ให้เป็นกล่องๆหนึ่ง เมื่อเราป้อน trajectory input เข้าไป เราก็จะได้ค่า output ออกมาค่าหนึ่ง และเมื่อเราทำการป้อน trajectory input ที่เหมือนเดิมเข้าไปอีก แน่นอนว่าค่าของ output ที่จะได้ก็จะมีค่าเท่าเดิม ซึ่งค่าของ output ที่เกิดขึ้นเหมือนเดิมในทุกๆการทำซ้ำนั้น จะก่อให้เกิดค่าของ error ที่เหมือนเดิมขึ้นในทุกๆการทำซ้ำขึ้นด้วย

จากการที่เรารู้ว่ามนุษย์สามารถเลียนรู้ได้จากสิ่งที่ผิดพลาดในอดีต ดังนั้น ILC จึงเลียนแบบพฤติกรรมของมนุษย์ กล่าวคือ ILC จะนำค่าของ error ในขบวนการทำซ้ำครั้งปัจจุบันมาปรับปรุงค่าของ trajectory input ที่จะป้อนเข้าระบบในขบวนการทำซ้ำครั้งถัดไป โดยเราสามารถเขียนเป็นสมการได้ดังต่อไปนี้

$$u_{j+1}(k) = u_j(k) + Le_j(k) \tag{7}$$

ในที่นี้ subscript j คือจำนวนครั้งของขบวนการทำซ้ำ ดังนั้น $u_{j+1}(k)$ จึงหมายถึง control input repetition ที่ $j+1$ time step ที่ k ซึ่งจะมีค่าเท่ากับ control input repetition ที่ j time step ที่ k รวมกับค่าของ learning matrix L คูณกับ error repetition ที่ j time step ที่ k ลักษณะขบวนการทำซ้ำจะเกิดขึ้นเรื่อยๆแบบ recursive โดยถ้า $j=0$ หมายถึง ระบบยังไม่มีการใช้ Learning controller ดังนั้น trajectory input จึงถูกป้อนเข้าไปยัง feedback control system โดยตรง โดยจะไม่มีการเปลี่ยนแปลงสัญญาณก่อนเข้าไปในตัวระบบเลย แต่เมื่อ j เริ่มมีค่ามากกว่าศูนย์แล้ว Learning controller ก็จะเริ่มทำงานโดยจะนำ trajectory input และ error ของการทำซ้ำครั้งก่อนมาใช้คำนวณหา control input ที่จะป้อนเข้าไปในระบบ feedback control ใหม่เหมือนในสมการที่ (7)
เมื่อทำการเปลี่ยนแปลงสัญญาณที่จะป้อนเข้าไปใน feedback control system เรียบร้อยแล้ว ต่อมาเราจะนำสัญญาณ u_{j+1} ที่คำนวณได้ป้อนเข้าสู่ระบบ ซึ่งจะเป็นไปตามสมการที่ (8)

$$\underline{y}_j = \overline{A}x(0) + P\underline{u}_j \tag{8}$$

$$\underline{y}_j = \begin{bmatrix} y_j(1) \\ y_j(2) \\ \vdots \\ y_j(p) \end{bmatrix}, \ \underline{u}_j = \begin{bmatrix} u_j(0) \\ u_j(1) \\ \vdots \\ u_j(p-1) \end{bmatrix}, \ P = \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{p-1}B & CA^{p-2}B & \cdots & CB \end{bmatrix}, \ \overline{A} = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^p \end{bmatrix}$$

โดย $\underline{y}_j$ คือ output ของระบบ repetition ที่ j ซึ่งเขียนอยู่ในรูปของ column vector และ P คือ matrix ที่ประกอบไปด้วย markov parameter ของระบบ ที่อธิบายความสัมพันธ์ของ control input และ output ทั้งนี้เราพิจารณาจากระบบที่เป็นแบบ Linear Time Invariant (LTI) system และ p คือจำนวน time step ทั้งหมดในขบวนการทำซ้ำหนึ่งครั้ง ในสมการที่ (4) นี้ เราจะสมมติว่า ระบบมี intial condition $x(0)$ เป็นศูนย์

จากสมการที่ (7) และ (8) จะเห็นได้ว่า ILC จะทำการปรับสัญญาณ control input u_j ไปทุกๆ iteration ดังนั้นสัญญาณที่เราต้องการให้ output ทำการ track ตามนั้นจะถูกเรียกใหม่ว่าเป็น y_d ซึ่งเป็น trajectory input ที่เราต้องการ track ตาม โดยในแต่ละ iteration นั้น y_d จะมีลักษณะของสัญญาณเหมือนกันเสมอ และใน iteration ที่ศูนย์ หรือ $j = 0$ หมายถึง ระบบยังไม่มีการใช้ Learning controller ดังนั้น y_d จึงมีค่าเท่ากับ u ซึ่งเราสามารถเขียนสมการ error propagation จากการทำซ้ำครั้งหนึ่งไปสู่การทำซ้ำอีกครั้งหนึ่งได้ดังสมการ

$$\underline{e}_{j+1} = [I - PL]\underline{e}_j \tag{9}$$

$$\underline{e}_j = \begin{bmatrix} e_j(1) \\ e_j(2) \\ \vdots \\ e_j(p) \end{bmatrix} = \begin{bmatrix} y_d(p) - y(p) \\ y_d(p) - y(p) \\ \vdots \\ y_d(p) - y(p) \end{bmatrix}$$

I ในที่นี้คือ Identity matrix จะเห็นได้ว่าถ้าเราต้องการให้ error ในแต่ละ iteration ลดลงเรื่อยๆจนมีค่าเข้าใกล้ศูนย์ (converge to zero) เมื่อ $j \to \infty$ เทอม $[I - PL]$ ต้องมีค่า eigenvalue ทุกๆค่าที่น้อยกว่า 1 หรือกล่าวคือ

$$\max_i |\lambda_i(I - PL)| < 1 \tag{10}$$

ซึ่ง condition ในสมการที่ (10) เป็น necessary condition ของ stability of ILC และบ่อยครั้งที่ ILC law ที่ใช้นั้นจะทำให้เกิด poor transient response ซึ่งเราสามารถเขียน sufficient stability condition ได้เป็น

$$\max_i \sigma_i(I - PL) < 1 \tag{11}$$

ดังนั้นระบบใดๆที่สอดคล้องกับสมการที่ (11) จะทำให้เกิดผลตอบสนองที่ลดค่าลงอย่างต่อเนื่อง (monotonically decay)

2.3. วิธีพื้นฐานของการออกแบบตัวควบคุมแบบเรียนรู้ที่มีการย้อนกลับไปสู่ภาวะเริ่มต้น

จากการที่เราทราบว่าระบบมีสมการความสัมพันธ์ระหว่าง input และ output เป็นอย่างไรแล้วนั้น ต่อมาเราจะนำเมตริกซ์ P นี้มาออกแบบตัวควบคุม (controller) ที่มีความเหมาะสมกับระบบนั้นต่อไป โดยทั่วไปแล้วการออกแบบตัวควบคุมในระบบ ILC มีหลายรูปแบบ แต่ในหัวข้อนี้จะขอกล่าวถึง learning control law ที่เป็นพื้นฐานที่สำคัญและง่ายแต่การนำเอาไปใช้งาน โดยจะแยกออกเป็นประเภทต่างๆดังนี้

2.3.1. Contraction Mapping ILC Law

Contraction mapping law เป็นการปรับค่า control input ให้ minimize ค่า Euclidean norm ของ error ทั้งหมดในขบวนการทำซ้ำหนึ่งครั้ง โดยในที่นี้เราสามารเขียนความสัมพันธ์ของ controller ได้ดังนี้

$$L = \phi P^T \tag{12}$$

โดยค่า ϕ คือค่า gain ที่เราสามารถปรับค่าได้ และเมตริกซ์ P คือเมตริกซ์ของระบบที่เราประมาณหาค่าได้จากการใช้ Linearization ดังที่ได้กล่าวมาแล้ว

2.3.2. Partial Isometry ILC Law

รูปแบบของ learning control law สำหรับ Partial isometry นี้จะคล้ายกับ contraction mapping law โดยถ้าเป็น contraction mapping law แล้ว $L = \phi P^T = \phi VSU^T$ ซึ่ง $P = USV^T$ เป็น singular value decomposition ของระบบ ใน partial isometry law นั้น เราจะไม่พิจารณาถึงค่าต่างๆใน singular value แต่จะพิจารณาเพียงแค่

$$L = \phi VU^T \tag{13}$$

2.3.3. Quadratic Cost ILC Law

ใน Quadratic cost ILC law นี้ จะเป็นการ minimize ทั้งเทอมของ Euclidean norm ของ error และเทอมของ control input ตลอดทั้ง trajectory โดยมี Q เป็น weight ของเทอม error และ R เป็น weight ของเทอม control input ซึ่งมีสมการเป็น

$$L = \left(P^T Q P + R\right)^{-1} P^T Q \tag{14}$$

2.4. วิธีการออกแบบตัวควบคุมแบบเรียนรู้ที่มีการย้อนกลับไปสู่ภาวะเริ่มต้นจากข้อมูลเชิงความถี่ของระบบ

ในหัวข้อนี้จะกล่าวถึงการออกแบบตัวควบคุมแบบ learning control โดยอาศัยข้อมูลใน frequency domain ของระบบมาใช้ในการออกแบบ [7]

2.4.1. Optimization Control Law

การออกแบบ control law โดยการใช้ Optimization ในรูปแบบนี้ เราจะออกแบบ Repetitive control law ขึ้นมาก่อนแล้วทำการเปลี่ยนกฎควบคุมให้อยู่ในรูปของ learning control law โดยขั้นตอนการออกแบบ repetitive control law นั้น เริ่มจากการพิจารณา stability condition สำหรับ RC system ดังที่ได้กล่าวมาข้างต้นแล้ว กล่าวคือ ถ้าเราเขียนความสัมพันธ์ของ error ใน frequency domain แล้ว เราจะได้

$$z^{p}E(z) = [1 - F(z)G(z)]E(z) \qquad (15)$$

ซึ่งเทอมทางซ้ายมือแสดงถึงค่าของ error ใน repetition ถัดไป จะมีค่าเท่ากับเทอมทางขวามือคือ $[1 - F(z)G(z)]$ คูณกับ error ใน repetition ปัจจุบัน ซึ่งถ้าเราสามารถทำให้เทอม $[1 - F(z)G(z)]$ มีขนาดเข้าใกล้ศูนย์ (หรือเท่ากับศูนย์) ในทุกๆความถี่ ค่าของ error ใน repetition ถัดไปก็จะมีค่าเข้าใกล้ศูนย์ทันที ทั้งนี้ $F(z)$ คือ repetitive controller ที่ต้องทำการออกแบบขึ้นมา และ $G(z)$ คือ transfer function ของระบบ feedback control system
ดังจะเห็นได้จากสมการที่ (15) เราควรออกแบบ $F(z)$ เพื่อทำให้เทอม $[1 - F(z)G(z)]$ มีขนาดน้อยที่สุด หรือกล่าวคือ เราทำการตั้ง Objective function ในการทำ minimization ใน Frequency domain โดย Objective function นี้คือ

$$\text{Min}\left\{\left[1 - F(e^{i\omega_j})G(e^{i\omega_j})\right]^{*} W_j \left[1 - F(e^{i\omega_j})G(e^{i\omega_j})\right]\right\} \forall j \qquad (16)$$

โดย $F(e^{i\omega}) = a_1 e^{-mi\omega} + a_2 e^{(-m+1)i\omega} + \ldots + a_n e^{(n-m)i\omega}$ ทั้งนี้ n คือจำนวน gain ทั้งหมดที่ใช้ใน controller และ $m-1$ คือจำนวนของ non-causal gain ที่ใช้

ขั้นตอนการออกแบบ repetitive controller นั้น เราจะเริ่มจากการกำหนดจำนวน gain และ non-causal gain ที่จะใช้ใน controller ก่อน หลังจากนั้นจึงแทนสมการของ controller เข้าไปในสมการที่ (16) อีกทีหนึ่ง เพื่อหา coefficient $a_1, a_2, \ldots, a_n$ โดยหลังจากได้ coefficient ต่างๆเสร็จแล้ว เราจึงมาจัดรูปใหม่ให้ใช้กับระบบ ILC

วิธีการเปลี่ยนรูป repetitive controller ให้อยู่ในรูปของ learning controller นั้น หรือแปลงจาก $F(z) \rightarrow L$ เราจะเห็นได้ว่า repetitive controller เขียนอยู่ในรูปของ transfer function ใน frequency domain แต่ learning controller นั้น จะถูกเขียนอยู่ในรูปของ matrix ใน time domain ซึ่งมีขนาดเท่ากับ

จำนวน time step ในหนึ่ง iteration หรือ p วิธีการแปลงนั้น เราจะจับ coefficient ต่างๆ หรือ $a_1, a_2, \ldots, a_n$ ใส่เข้าไปใน matrix L ดังแสดง

$$L = \begin{bmatrix} a_{m-2} & \cdots & a_1 & 0 \\ \vdots & a_{m-2} & \cdots & a_1 \\ a_n & \cdots & \ddots & \vdots \\ 0 & a_n & \cdots & a_{m-2} \end{bmatrix}_{p \times p} \tag{17}$$

จะเห็นได้ว่าค่า gain $a_1, a_2, \ldots, a_n$ ถูกนำมาวางเป็น pattern ในเมตริกซ์ L ซึ่งเราก็จะได้ learning matrix จากการออกแบบ repetitive control law ดังสมการที่ (17)

2.4.2. Optimization Control Law with minimal gain size

จากการออกแบบ optimization control law ในหัวข้อที่แล้ว เราไม่สามารถบอกได้ว่าขนาดของค่า gain $a_1, a_2, \ldots, a_n$ ต่างๆนั้นจะมีค่ามากเท่าใด ซึ่งเราไม่ต้องการค่า gain ที่มีค่าสูงมากๆเนื่องจากถ้าค่า gain สูงขึ้นมาก จะทำให้สัญญาณ control input ที่ป้อนเข้าระบบจะมีค่าสูงมากตามไปด้วย ซึ่งระบบบางระบบนั้นไม่สามารถสร้างสัญญาณ control input ที่มีขนาดสูงตามต้องการได้ จึงทำให้เราไม่สามารถควบคุมระบบจริงได้จากการใช้ค่า gain ของ controller ที่มีค่าสูงๆนั่นเอง ดังนั้นเราจึงควรเพิ่ม objective function อีกเทอมหนึ่งเข้าไปได้เพื่อทำการ minimize ขนาดของ gain ด้วย cost function ใหม่สามารถถูกเขียนได้เป็น

$$\text{Minimize} \left\{ \left[1 - F(e^{i\omega})G(e^{i\omega})\right]^* \left[1 - F(e^{i\omega})G(e^{i\omega})\right] + V\left(a_1^2 + a_2^2 + \ldots + a_n^2\right) \right\} \tag{18}$$

จะเห็นได้ว่า cost function ที่เพิ่มเข้าไป จะทำการลดขนาดของ gain ให้ต่ำที่สุด พร้อมกับทำให้เทอม $\left\{ \left[1 - F(e^{i\omega})G(e^{i\omega})\right]^* \left[1 - F(e^{i\omega})G(e^{i\omega})\right] \right\}$ มีขนาดเล็กที่สุดด้วย ซึ่งเมื่อเราทำ optimization เพื่อให้ได้ repetitive controller ออกมาแล้ว เราจึงนำมาแปลงให้เป็น learning control matrix เหมือนที่กล่าวในหัวข้อที่แล้ว

2.4.3. Column cut-off control Law

ในบาง applications เราอาจไม่จำเป็นต้องพิจารณาค่าความผิดพลาดใน time step แรกๆ เนื่องจากเราอาจต้องการค่าความถูกต้องที่สูงเมื่อระบบเข้าสู่สภาวะ steady-state เรียบร้อยแล้ว ซึ่งถ้าใช้ learning control ใน application ดังกล่าวนั้น ค่าความผิดพลาดใน time step แรกๆ จะถูกคงค่าไว้เท่าเดิมเนื่องจากไม่มีการเรียนรู้เกิดขึ้น แต่ค่าความผิดพลาดใน time step อื่นๆจะมีค่าลดลงตามขบวนการของ learning control ดังนั้นในการออกแบบ control law ถ้าเราไม่ต้องการพิจารณาค่าความผิดพลาดใน time step ไหน ก็ให้ตัดค่าใน column นั้นทิ้งออกไป ยกตัวอย่างเช่น ถ้าเรามีการใช้ Optimization control law ที่มี learning matrix ดังที่ได้กล่าวมาคือ

$$L=\begin{bmatrix} a_{m-2} & \cdots & a_1 & 0 \\ \vdots & a_{m-2} & \cdots & a_1 \\ a_n & \cdots & \ddots & \vdots \\ 0 & a_n & \cdots & a_{m-2} \end{bmatrix}_{p\times p} \tag{19}$$

และเราไม่ต้องการพิจารณาค่าความผิดพลาดใน time step ที่ 1 เราจึงตัดค่าต่างๆใน column ที่ 1 ทิ้งไปซะ ซึ่งก็จะได้ learning matrix ใหม่ขึ้นมาคือ

$$L=\begin{bmatrix} a_{m-3} & \cdots & a_1 & 0 \\ a_{m-2} & \ddots & \cdots & a_1 \\ \vdots & \cdots & \ddots & \vdots \\ a_n & \cdots & \cdots & a_{m-3} \end{bmatrix}_{p\times p-1} \tag{20}$$

จะเห็นได้ว่า learning matrix ใหม่ที่ได้มานั้นจะมี dimension เป็น $p\times p-1$

2.5. วิธีการออกแบบตัวควบคุมแบบเรียนรู้ที่มีการย้อนกลับไปสู่ภาวะเริ่มต้นโดยวิธีการปรับเกนให้เกิดการเรียนรู้เร็วที่สุด

ในการออกแบบเมตริกซ์การเรียนรู้หรือ Learning matrix นั้น ในบางครั้งถ้าเมตริกซ์การเรียนรู้ที่ได้ออกแบบมานั้นไม่มีประสิทธิภาพ กล่าวคือเมตริกซ์การเรียนรู้ที่ได้ออกแบบมานั้นไม่มีความเสถียรภาพหรือมีความสามารถในการเรียนรู้ที่ต่ำ ดังนั้นเราสามารถปรับค่าเกนในเมตริกซ์การเรียนรู้บางตัวเพื่อทำให้ระบบมีเสถียรภาพและมีความสามารถในการเรียนรู้ที่สูงขึ้นได้ [8] โดยตำแหน่งเกนในเมตริกซ์การเรียนรู้ที่เราจะทำการปรับนั้นสามารถพิจารณาได้จากค่า sensitivity function ซึ่งเป็นตัวชี้วัดว่าตำแหน่งเกนที่ตำแหน่งใดควรจะถูกปรับค่า และเราจะใช้ขบวนการปรับค่าแบบ steepest descent เพื่อจะปรับค่าเกนที่จะทำให้เกิดค่า maximum singular value ของเมตริกซ์ $(I-PL)$ ที่ต่ำที่สุด ดังจะอธิบายในหัวข้อถัดไป

2.5.1. Sensitivity function

ในหัวข้อนี้จะกล่าวถึงการเปลี่ยนแปลงของ maximum singular value ของเมตริกซ์ $(I-PL)$ ซึ่งเป็นสมการ sufficient condition สำหรับ stability ของ ILC เทียบกับการเปลี่ยนแปลงค่าของเกนในเมตริกซ์การเรียนรู้ (Learning matrix) L โดยถ้าเรามีเมตริกซ์การเรียนรู้ดังสมการ

$$L=\begin{bmatrix} \varepsilon_{11} & \varepsilon_{12} & \cdots & \varepsilon_{1n} \\ \varepsilon_{21} & \varepsilon_{22} & \cdots & \varepsilon_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \varepsilon_{n1} & \varepsilon_{n2} & \cdots & \varepsilon_{nn} \end{bmatrix} \tag{21}$$

และอัตราการเปลี่ยนแปลงของเมตริกซ์การเรียนรู้เทียบกับ ε_{11} มีค่าเป็น

$$\frac{\partial L}{\partial \varepsilon_{11}} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{22}$$

ในทำนองเดียวกัน ถ้าเราหาเมตริกซ์ $\frac{\partial L}{\partial \varepsilon_{ij}}$ เราก็จะได้เมตริกซ์ที่มี element แถวที่ i หลักที่ j มีค่าเป็น 1 ส่วน element ในหลักและแถวอื่นๆมีค่าเป็นศูนย์ ในการหาค่า singular value ของเมตริกซ์ $(I-PL)$ นั้น เราสามารถหาได้จากการเขียนให้อยู่ในรูป decomposition

$$(I-PL) = USV^T \tag{23}$$

โดย $V^TV = U^TU = I$ และ S เป็น diagonal matrix ที่มีค่า singular value เรียงค่าจากมากไปหาน้อยอยู่บน main diagonal line ดังนั้นถ้าเราต้องการพิจารณาการเปลี่ยนแปลงของ maximum singular value เทียบกับค่าเกนในเมตริกซ์การเรียนรู้ เราสามารถเขียนได้เป็น

$$\frac{\partial \sigma_{\max}}{\partial \varepsilon_{ij}} = U^T \frac{\partial (I-PL)}{\partial \varepsilon_{ij}} V \tag{24}$$

$\sigma_{\max}$ คือค่า maximum singular value ของเมตริกซ์ $(I-PL)$ ทั้งนี้เรานิยามคำจำกัดความของ sensitivity หรือความไวในการเปลี่ยนค่าของ maximum singular value เทียบกับค่าเกนในเมตริกซ์การเรียนรู้ได้เป็น $\frac{\partial \sigma_{\max}^2}{\partial \varepsilon_{ij}}$ โดยเรารู้ว่า

$$\begin{aligned} (I-PL)^T(I-PL) &= (I-PL)(I-PL)^T \\ &= USV^TVSU^T \\ &= US^2U^T \end{aligned} \tag{25}$$

ดังนั้น

$$\begin{aligned} S^2 &= \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & 0 & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_n^2 \end{bmatrix}. \\ &= U^T \Gamma U \end{aligned} \tag{26}$$

โดยที่ $\Gamma = (I-PL)^T(I-PL)$ และ $\sigma_1^2 \geq \sigma_2^2 \geq \ldots \geq \sigma_n^2$ ดังนั้นเมื่อเราต้องการหาค่า sensitivity เราสามารถทำได้ดังนี้

$$\frac{\partial \sigma_{\max}^2}{\partial \varepsilon_{ij}} = u_1^T \frac{\partial \Gamma}{\partial \varepsilon_{ij}} u_1$$
$$\frac{\partial \sigma_{\max}^2}{\partial \varepsilon_{ij}} = u_1^T \frac{\partial \left[(I - PL)^T (I - PL) \right]}{\partial \varepsilon_{ij}} u_1 \qquad (27)$$
$$\frac{\partial \sigma_{\max}^2}{\partial \varepsilon_{ij}} = u_1^T \left[(I - PL)\left(-P\frac{\partial L}{\partial \varepsilon_{ij}} \right)^T + \left(-P\frac{\partial L}{\partial \varepsilon_{ij}} \right)(I - PL)^T \right] u_1$$

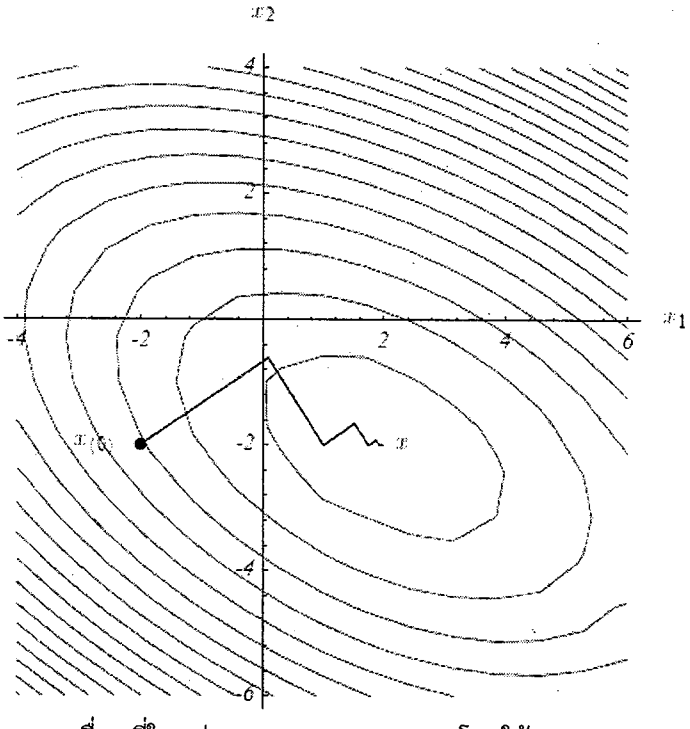
โดยที่ u_1 คือค่าในหลักที่ 1 ของเมตริกซ์ U และค่า $\frac{\partial L}{\partial \varepsilon_{ij}}$ ก็คือเมตริกซ์ที่มี element เท่ากับ 1 ที่หลักที่ i แถวที่ j ดังกล่าวมาแล้วข้างต้น

2.5.2. Steepest descent method

ขบวนการการทำ optimization เพื่อหาค่าต่ำสุดหรือสูงสุดนั้น วิธีการหนึ่งที่ถูกใช้กันอย่างกว้างขวางคือ steepest descent method วิธีการนี้จะมีการทำซ้ำหลายๆรอบ (iteration) เพื่อหาค่า optimal ออกมา ซึ่งในการทำซ้ำแต่ละรอบนั้น จะเป็นการ update ค่าของ parameter ที่เราต้องการทำ optimization โดยมีสมการเป็น

$$x(k+1) = x(k) + s(k)\, D(k) \qquad (28)$$

ตัวแปร $x(k)$ คือค่าที่เราต้องหาค่าต่ำสุด/สูงสุดใน iteration ที่ k (ในงานวิจัยนี้ขอพูดแค่ค่าต่ำสุดเนื่องจากเราทำการ minimize cost function เพียงอย่างเดียว) ส่วนตัวแปร $s(k)$ คือ step size ซึ่งเป็นขนาดของขั้นที่จะทำการกระโดดไปในแต่ละ iteration และ $D(k)$ คือ direction ที่จะทำการกระโดด ซึ่งปกติเราจะกำหนดให้ direction ของการกระโดดเป็นทิศตรงข้ามกับ Gradient ของ function ณ จุด $x(k)$ ซึ่ง Gradient ของ function นั้นจะมีลักษณะตั้งฉากกับเส้นของ function ดังนั้นเวลาทำการ update ค่าในแต่ iteration direction $D(k)$ จะตั้งฉากกันไปเรื่อยๆ ดังแสดงในรูปที่ 2.2



รูปที่ 2.2 ทิศทางการเคลื่อนที่ในแต่ละ iteration update โดยใช้ Steepest descent: ที่มา [9]

ในการเลือกขนาดของ step size นั้น ก่อนอื่นเราต้องทำการกำหนด initial step size ขึ้นมาก่อน โดยอาจจะเป็นค่าน้อยๆก่อนก็ได้ แล้วจึงทำการเปรียบเทียบดูว่าถ้าค่า $x(k+1)$ มีค่าต่ำกว่า $x(k)$ แล้ว เราจะทำการเพิ่มขนาดของ step size ขึ้นสองเท่า เนื่องจากเรากระโดดมาถูกทิศ และเราต้องการไปยังจุด minimal ให้เร็วที่สุด เราจึง double ขนาดของ step size แต่ในทางตรงกันข้าม ถ้า $x(k+1)$ มีค่าสูงกว่า $x(k)$ นั่นคือเรากระโดดไปในทิศที่ไม่ถูกต้อง เราจึงต้องทำการกลับทิศเพื่อกระโดดกลับไปยังจุดก่อนหน้า (จุดสุดท้ายที่ค่า $x(k+1)$ มีค่าต่ำกว่า $x(k)$) แล้วจึงทำการลดขนาดของ step size ลงครึ่งหนึ่งแล้วกระโดดต่อ

ในงานวิจัยนี้ เราต้องการหาค่า coefficient ของ learning matrix ที่จะทำให้เกิดค่า $\max_i \sigma_i(I-PL)<1$ ต่ำที่สุด ดังนั้นเราจะใช้ steepest descent หาค่า coefficient ของ learning matrix โดยกำหนดให้มี direction เป็นทิศทางที่เกิดค่า sensitivity สูงสุดในแต่ละ iteration

2.6. การออกแบบตัวควบคุมแบบเรียนรู้ที่มีการย้อนกลับไปสู่ภาวะเริ่มต้นโดยวิธีการหาค่าเฉลี่ย

โดยทั่วไปแล้ว โมเดลที่ได้จากการประมาณของระบบจริงจะมีความผิดพลาดอยู่มาก ดังนั้นเราจึงควรที่จะออกแบบ controller เพื่อที่จะรองรับการประมาณที่ผิดพลาดไป โดยในหัวข้อนี้เสนอวิธีการในการออกแบบตัวควบคุมแบบเรียนรู้ที่มีการย้อนกลับไปสู่ภาวะเริ่มต้นโดยวิธีการหาค่าเฉลี่ย โดยเริ่มจากการสร้างจำนวนโมเดลที่ประมาณมาทั้งหมด M โมเดล (M เป็นค่าคงที่ใดๆ) ซึ่งใน M โมเดลที่สร้างขึ้นมานั้นมีค่าของตัวแปรของระบบไม่แน่นอน มีค่าแตกต่างกันออกไปอยู่ในช่วงๆหนึ่ง จากนั้นจึงเลือกออกแบบ controller จำนวน M ตัว ที่เหมาะสมกับแต่ละโมเดลที่สร้างขึ้นมา ในที่นี้เราต้องการออกแบบตัวควบคุมเพียง 1 ตัวเท่านั้น ที่ทำให้เกิดค่า expected value ของ cost function ทั้ง M ฟังก์ชัน มีค่าต่ำที่สุด หรือกล่าวคือ เราจะนำเอา controller ที่ได้ออกแบบมาทั้งหมด M ตัวมาทำการหาค่าเฉลี่ย ตามทฤษฎีของ stochastic model ที่กล่าวว่าค่า expected value ของ function จำนวนทั้งหมด M ฟังก์ชัน จะมีค่าประมาณเท่ากับค่าเฉลี่ยของ M function นั้น เมื่อจำนวน M หรือ sample มีค่ามากๆ นั่นเอง ในที่นี้จะขอกล่าวถึงเทคนิคออกแบบ learning control law จากการหาค่าเฉลี่ยโดยวิธีพื้นฐานของการออกแบบ ILC ดังต่อไปนี้

2.6.1. Averaging Contraction Mapping ILC Law

จากสมการที่ (12) เมื่อมีการออกแบบตัวควบคุมแบบ contraction mapping ตัวควบคุมที่ใช้จะมีค่าเท่ากับค่าเกนคูณกับ transpose ของ เมตริกซ์ P หรือกล่าวคือ $L=\phi P^T$ โดยถ้าโมเดลของระบบที่เราทำการพิจารณาอยู่นั้นมีจำนวนทั้งหมด M โมเดล เราสามารถหาค่าเฉลี่ยจากการใช้ contraction mapping control law ได้ดังต่อไปนี้

$$L=\frac{1}{M}\sum_{i=1}^{M}L_i=\frac{\phi}{M}\sum_{i=1}^{M}P_i^T \tag{29}$$

2.6.2. Averaging Partial Isometry ILC Law

ในทำนองเดียวกันกับสมการที่ (13) การออกแบบ partial isometry ILC law จากการหาค่าเฉลี่ยสามารถถูกแสดงได้เป็น

$$L=\frac{1}{M}\sum_{i=1}^{M}L_i=\frac{\phi}{M}\sum_{i=1}^{M}V_iU_i^T \tag{30}$$

2.6.3. Averaging Quadratic Cost ILC Law

วิธีการออกแบบ quadratic cost ILC law จากการหาค่าเฉลี่ยจะคล้ายกับสมการที่ (14) กล่าวคือ

$$L=\left(\sum_{i=1}^{M}P_i^TQP_i+MR\right)^{-1}\sum_{i=1}^{M}P_i^TQ \tag{31}$$