บทที่ 3

อัลกอริทึมที่สำคัญในโปรแกรม

ในบทนี้เป็นการนำเสนอเนื้อหาของอัลกอริทึมที่สำคัญ ซึ่งใช้ในการวิเคราะห์และออกแบบตามทฤษฎีควบคุมทั้งแบบคลาสสิก แบบสมัยใหม่ และแบบดิจิตัล

3.1 อัลกอริทึมสำหรับการคำนวณพื้นฐาน

1. การหาอันดับ (Rank) ของเมตริกซ์ ใช้การทำ Gaussian Elimination แบบ Partial Pivoting โดยมีขั้นตอนดังนี้

1.1 กำหนดให้ j = 1

1.2 หาตำแหน่งแถวของสมาชิกที่มีค่ามากที่สุดในคอลัมน์ j

1.3 สลับแถวที่ j กับแถวซึ่งมีค่าสมาชิกที่มากที่สุด

1.4 กำหนดให้ i = 1

1.5 คำนวณ m = a_{ij}/a_{jj} แล้วกำหนดให้ a_{ij} = 0

1.6 ลบสมาชิกในแถวที่ต่ำกว่าแถวที่ j ออกด้วย m*ค่าในแถวที่ j

1.7 เพิ่มค่า j ด้วย 1 แล้วตรวจสอบว่า j มีค่าเกิน n-1 หรือไม่ ถ้ายังไม่เกินให้ย้อนกลับไปทำขั้น 1.2 ซ้ำ

1.8 ตรวจสอบแนวแทยงมุมของเมตริกซ์ ถ้าเมตริกซ์มีอันดับเท่ากับ n สมาชิกในแนวแทยงมุมหลังจากผ่านการทำ Gaussian Elimination จะมีครบทุกตำแหน่ง

2. การหารากของโพลิโนเมียลกำลัง n อาศัยหลักการของ Bairstow ในการดึงตัวร่วมในรูปของโพลิโนเมียลกำลังสองออกจากสมการที่ต้องการหาเรื่อยๆจนหมดสมการ จากนั้นจึงหารากของโพลิโนเมียลกำลังสองที่ดึงออกมาได้ทีละคู่จนครบ โดยเมื่อกำหนดรูปแบบของโพลิโนเมียลให้อยู่ในรูป

$$a_0 s^n + a_1 s^{n-1} + a_2 s^{n-2} + \ldots + a_{n-1} s + a_n \qquad (3\text{-}1)$$

ขั้นตอนในการคำนวณหารากของสมการสามารถสรุปเป็นขั้นตอนได้ดังนี้

2.1 กำหนดค่าเริ่มต้น u และ v ให้เท่ากับ 0

2.2 คำนวณ $b_1,b_2,...,b_n$ จากสมการ

$$b_k = a_k - b_{k-1}u - b_k - 2v \quad ; \quad k = 2,3,\ldots,n \qquad (3\text{-}2)$$

โดยกำหนดให้ $b_0 = 1$ และ $b_1 = a_1-u$

2.3 คำนวณ $c_1,c_2,...,c_n$ จากสมการ

$$c_k = b_k - c_{k-1}u - c_{k-2}v \quad ; \quad k = 2,3,\ldots,n-1 \qquad (3\text{-}3)$$

โดยกำหนดให้ $c_0 = 1$ และ $c_1 = b_1-u$

2.4 คำนวน Δu และ Δv จากสมการ

$$\Delta u = \frac{\begin{vmatrix} b_n & c_{n-2} \\ b_{n-1} & c_{n-3} \end{vmatrix}}{\begin{vmatrix} c_{n-1} & c_{n-2} \\ c_{n-2} & c_{n-3} \end{vmatrix}} \qquad (3\text{-}4)$$

$$\Delta v = \frac{\begin{vmatrix} c_{n-1} & b_n \\ c_{n-2} & b_{n-1} \end{vmatrix}}{\begin{vmatrix} c_{n-1} & c_{n-2} \\ c_{n-2} & c_{n-3} \end{vmatrix}} \qquad (3\text{-}5)$$

2.5 เพิ่มค่า u และ v ด้วย Δu และ Δv ตามลำดับ

$$\begin{aligned} u_{i+1} &= u_i + \Delta u_i \\ v_{i+1} &= v_i + \Delta v_i \end{aligned} \qquad (3\text{-}6)$$

2.6 ย้อนกลับไปขั้นที่ 2.2 แล้วคำนวณจนกว่า Δu และ Δv มีค่าเข้าใกล้ศูนย์ ในระดับที่ยอมรับได้ หากอัลกอริทึมไม่สามารถลู่เข้าหาคำตอบได้ ภายในการคำนวณ 50 วงรอบ ให้เปลี่ยนค่า u1 เป็น a_{n-1}/a_{n-2} และ v1 เป็น a_n/a_{n-2}

2.7 คำนวณค่ารากจากสมการ

$$s_1, s_2 = \frac{-u \pm \sqrt{u^2 - 4v}}{2} \qquad (3\text{-}7)$$

2.8 หักรากที่สามารถแยกได้ออกมาจากโพลิโนเมียล โดยอาศัยสมการ

$$
\begin{aligned}
b_0 &= 1 \\
b_1 &= a_1 - u \\
b_2 &= a_2 - b_1 u - v \\
b_3 &= a_3 - b_2 u - b_1 v \\
b_4 &= a_4 - b_3 u - b_2 v \qquad (3\text{-}8) \\
&\cdots\cdots\cdots\cdots\cdots\cdots \\
b_k &= a_k - b_{k-1} u - b_{k-2} v \\
&\cdots\cdots\cdots\cdots\cdots\cdots \\
b_{n-2} &= a_{n-2} - b_{n-3} u - b_{n-4} v
\end{aligned}
$$

แทนโพลิโนเมียลด้วยโพลิโนเมียลใหม่ที่หักส่วนของรากแล้วย้อนกลับไปทำขั้นที่ 2.1 จนกว่าสามารถหารากได้ครบทุกตัว

3. การหา Eigenvalue ของเมตริกซ์ หากใช้การคำนวณจากการหารากของสมการ det[sI - A] จะเสียเวลามาก ดังนั้นจึงใช้เทคนิคที่สามารถหา Eigenvalue ออกมาได้โดยตรง ซึ่งเทคนิคที่เลือกใช้ในการวิจัยครั้งนี้ คือ อัลกอริทึมแบบ QR ซึ่งเป็นการแปลงรูปเมตริกซ์แบบ Orthogonal ให้เมตริกซ์ใหม่อยู่ในรูป Schur ซึ่งเป็นเมตริกซ์ที่มีสมาชิกเฉพาะชีกบนขวา โดยอาจจะมีสมาชิกในแนวขนานกับแนวแทยงมุมแต่ต่ำกว่าหนึ่งขั้นได้ สมาชิกในแนว 3 แถวแทยงมุมของเมตริกซ์แบบ Schur มีความสัมพันธ์กับค่า Eigenvalue ของระบบซึ่งสามารถใช้คำนวณ Eigenvalue ออกมาได้ เมื่อกำหนดให้เมตริกซ์ที่เราสนใจเป็นเมตริกซ์จัตุรัสมิติ n x n รายละเอียดการพิสูจน์อัลกอริทึมนี้สามารถศึกษาได้จาก Buchanan and Turner (1992) โดยอัลกอริทึมในวิทยานิพนธ์นี้ได้มาจากชุดคำสั่งของ Eispack ขั้นตอนในการหา Eigenvalue ด้วยวิธี QR สามารถสรุปได้ดังนี้

3.1 เปลี่ยนเมตริกซ์ให้อยู่ในรูป Hessenberg

3.2 กำหนด j = 1 และเมตริกซ์ H เท่ากับเมตริกซ์ที่แปลงจากเมตริกซ์ A เป็นรูป Hessenberg

3.3 ถ้า $h_{j+1,j}$ ไม่เท่ากับศูนย์ ให้ทำตามขั้นตอนดังนี้

3.3.1 คำนวณ $\beta = \dfrac{h_{jj}}{h_{j+1,j}}$

3.3.2 คำนวณ $s = \dfrac{1}{\sqrt{1+\beta^2}}$; $c = \beta s$

3.3.3 กำหนดให้ ξ = แถวที่ j ของเมตริกซ์ H และ = แถวที่ j+1 ของเมตริกซ์ H

3.3.4 กำหนดให้ k = ค่ามากที่สุดระหว่าง 1 และ j-2

3.3.5 คำนวณ $h_{jk} = c\xi_k + s\eta_k$ และ $h_{j+1,k} = -s\xi_k + c\eta_k$

3.3.6 คำนวณ k = k+1 หากค่า k น้อยกว่า n ให้ย้อนกลับไปทำขั้นที่ 3.3.5

3.3.7 กำหนดให้ $h_{j+1,j} = 0$

3.3.8 กำหนด ξ = คอลัมน์ที่ j ของเมตริกซ์ H และ = คอลัมน์ที่ j+1 ของเมตริกซ์ H

3.3.9 กำหนดให้ k = 1

3.3.10 คำนวณ $h_{jk} = c\xi_k + s\eta_k$ และ $h_{k,j+1} = -s\xi_k + c\eta_k$

3.3.11 คำนวณ k = k+1 หากค่า k น้อยกว่า j+2 หรือ n ให้ย้อนกลับไปทำขั้นที่ 3.3.10

3.4 คำนวณ j = j+1 หาก j มีค่าน้อยกว่า n-1 ให้ย้อนกลับไปทำขั้น 2.3

3.5 กำหนดให้สมาชิกใต้แนวแทยงมุมแถวที่สองของเมตริกซ์ H เท่ากับ 0

3.6 คำนวณหาค่า Eigenvalue ของเมตริกซ์ จากสมาชิกในแนวแทยงมุมทั้ง 3 แถว โดยแบ่งออกเป็น 2 กรณีดังนี้

3.6.1 ถ้าขนาดของ $h_{k+1,k}$ มีค่าใกล้เคียงศูนย์มาก ให้ถือว่า Eigenvalue ที่ตำแหน่งนั้นเท่ากับค่าจำนวนจริง h_{kk}

3.6.2 ถ้าขนาดของ $h_{k+1,k}$ มีค่าไม่ใกล้ค่าศูนย์ ให้หา Eigenvalue จากสมการ

$$\lambda^2 - (h_{kk} + h_{k+1,k+1})\lambda + h_{kk}h_{k+1,k+1} - h_{k+1,k}h_{k,k+1} = 0 \qquad (3\text{-}9)$$

4. การหาคำตอบของสมการด้วยวิธี Bisection เป็นเทคนิคแบบ Numerical Search สำหรับการหาคำตอบของสมการ โดยใช้หลักการตรวจสอบค่าของสมการที่ตำแหน่งขอบเขตของค่าตัวแปรทั้งสองด้าน และที่ตำแหน่งระหว่างกลางของขอบเขตการค้นหา แล้วใช้การเปลี่ยนค่าตำแหน่งขอบเขตของตัวแปรตามแนวโน้มของค่าตัวแปรทั้งสาม ดังนั้นค่าตำแหน่งขอบเขตของตัวแปรจะมีการเปลี่ยนแปลงไปจนกว่าจะสามารถหาคำตอบของสมการได้ สำหรับการวิเคราะห์เกี่ยวกับเทคนิค Bisection นี้ สามารถศึกษาได้จาก Buchanan and Turner (1992) สำหรับขั้นตอนการคำนวณมีดังนี้

4.1 คำนวณค่ากลางจาก (ค่าเริ่มต้น + ค่าสิ้นสุด)/2

4.2 คำนวณ f(ค่าเริ่มต้น)*f(ค่ากลาง)

4.2.1 ถ้าผลคูณมีค่ามากกว่าศูนย์ กำหนดค่าเริ่มต้นเท่ากับค่ากลาง

4.2.2 ถ้าผลคูณมีค่าน้อยกว่าศูนย์ กำหนดค่าสิ้นสุดเท่ากับค่ากลาง

4.3 ทำการคำนวณขั้นตอนที่ 4.1 และ 4.2 ซ้ำไปเรื่อยๆ จนกว่าความแตกต่างระหว่างค่าเริ่มต้นและค่าสิ้นสุดอยู่ในช่วงที่ยอมรับได้

5. การหาคำตอบของสมการด้วยวิธี Secant ในบางสมการ เราไม่สามารถกำหนดขอบเขตของการหาคำตอบได้ตั้งแต่ช่วงเริ่มต้นของการคำนวณ ดังนั้นจึงจำเป็นต้องใช้อัลกอริทึมที่สามารถไล่หาคำตอบได้เอง โดยในวิทยานิพนธ์ฉบับนี้ เลือกวิธีของ Secant ซึ่งสามารถไล่หาคำตอบจากสมการ โดยไม่ต้องกำหนดขอบเขตในการหาคำตอบ สำหรับหลักการและรายละเอียดของวิธี Secant นี้สามารถศึกษาได้จาก Hoffman (1992)

5.1 กำหนดค่า x0 และ x1 ซึ่งเป็นค่าเริ่มต้น 2 ค่าสำหรับการหาคำตอบของสมการ

5.2 คำนวณหาค่าตัวแปรในลูปต่อไปจากสมการ

$$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})} \qquad (3\text{-}10)$$

5.3 วนลูปคำนวณจนกว่าความแตกต่างระหว่างตัวแปรในลูปมีค่าอยู่ในช่วงที่ยอมรับได้

6. การแปลงรูปเมตริกซ์ให้อยู่ในรูป Hessenberg เมตริกซ์ที่อยู่ในรูป Hessenberg มีความสำคัญเป็นอย่างยิ่งในการคำนวณที่เกี่ยวกับ Eigenvalue ของเมตริกซ์ โดยขั้นตอนของการแปลงเมตริกซ์ใดๆมาสู่รูป Hessenberg สามารถทำได้ในหลายวิธี แต่ในวิทยานิพนธ์ฉบับนี้ เลือกวิธีของ QR ที่สามารถศึกษาได้จาก Chapra and Canale (1988) โดยขั้นตอนในการคำนวณมีดังนี้

6.1 กำหนดค่า j = 1

6.2 หา s_j จากสมการ

$$s_j = \sqrt{\sum_{k=j+1}^{n} \left(a_{kj}^{(j-1)}\right)^2} \qquad (3\text{-}11)$$

โดย $a_{ik}^{(j)}$ คือ สมาชิกของเมตริกซ์ที่ตำแหน่งแถว i คอลัมน์ k ในการคำนวณขั้นที่ j

6.3 ถ้า s_j ไม่เท่ากับศูนย์ ให้ทำตามขั้นตอนดังนี้

6.3.1 หาค่า α และ v จากสมการ

$$_j = \frac{1}{\left(s_j + \left|a^{(j)}_{j+1,j}\right|\right)s_j} \qquad (3\text{-}12)$$

$$\mathbf{P}_j = \mathbf{I} - \alpha_j \mathbf{v}^{(j)}\mathbf{v}^{(j)T} \qquad (3\text{-}13)$$

6.3.2 คำนวณ $\mathbf{z} = \alpha\mathbf{v}$

6.3.3 คำนวณ $\mathbf{x} = \mathbf{Az}$ และ $\mathbf{y} = \mathbf{A}^T\mathbf{z}$

6.3.4 คำนวณ $\mu = \frac{\alpha\,\mathbf{v}^T\mathbf{x}}{2}$

6.3.5 คำนวณ $\mathbf{z} = \mu\mathbf{v}$

6.3.6 คำนวณ $\mathbf{p} = \mathbf{y} - \mathbf{z}$ และ $\mathbf{q} = \mathbf{x} - \mathbf{z}$

6.3.7 คำนวณ $\mathbf{A} = \mathbf{A} - \mathbf{vp}^T - \mathbf{qv}^T$

6.4 ให้ j = j+1 แล้วตรวจสอบว่าเกิน n-2 หรือยัง หากยังไม่เกินให้ย้อนกลับไปทำตามข้อ 6.2 ใหม่

7. การหาค่า Determinant และ Inverse ของเมตริกซ์ สำหรับการคำนวณค่าทั้งสองนี้ ใช้การทำ Guassian Elimination with Partial Pivoting ในการคำนวณหา โดยสามารถศึกษารายละเอียดได้จาก Hoffman (1992)

7.1 แปลงรูปเมตริกซ์ให้อยู่ในรูป Upper Triangular โดยการทำ Guassian Elimination

7.2 คำนวณ Determinant จากสมาชิกในแนวแทยงมุมของเมตริกซ์

7.3 แปลงรูปเมตริกซ์ให้เป็นเมตริกซ์ Identity โดยใช้การทำ Guassian Elimination

7.4 นำขั้นตอนและค่าต่างๆที่กระทำต่อเมตริกซ์ในข้อ 7.1 และ 7.3 ไปกระทำต่อเมตริกซ์ Identity ซึ่งได้ผลลัพธ์เป็น Inverse ของเมตริกซ์

3.2 อัลกอริทึมสำหรับการวิเคราะห์ระบบควบคุม

1. การหารากของสมการลักษณะ และการหา Eigenvalue ของระบบ ในความเป็นจริงแล้ว การวิเคราะห์ทั้ง 2 แบบนี้เป็นการวิเคราะห์เดียวกันคือ การหารากของสมการซึ่งสามารถบอกถึงลักษณะของผลตอบที่จะได้จากระบบ ซึ่งโดยทั่วไปจะเป็นค่าเดียวกันยกเว้นในบางกรณี โดยในทฤษฎีควบคุมแบบคลาสสิกจะเป็นการหารากจากเทอมที่เป็นส่วนของ

ฟังก์ชันถ่ายโอน ซึ่งก็คือสมการลักษณะ ในขณะที่ทฤษฎีควบคุมสมัยใหม่เป็นการหารากของสมการ det[sI - A] = 0 อัลกอริทึมสำหรับคำนวณหารากของสมการทั้ง 2 แบบนี้แตกต่างกันไป

1.1 การหารากจากสมการลักษณะ ใช้เทคนิคการหารากของสมการโพลิโนเมียลกำลัง n

1.2 การหา Eigenvalue ของระบบ ใช้เทคนิคสำหรับหาค่า Eigenvalue ของเมตริกซ์ A

2. การตรวจสอบ Controllability และ Observability ของระบบ การตรวจสอบ Controllabilty และ Observabilty ของระบบ จัดเป็นเทคนิคการวิเคราะห์ในทฤษฎีควบคุมสมัยใหม่ที่มีความสำคัญมาก เนื่องจากคุณสมบัติทั้งสองสามารถบอกถึงความเป็นไปได้ของการใช้สัญญาณเข้าเพื่อควบคุมการทำงานของระบบ และความเป็นไปได้ในการสังเคราะห์ตัวแปรสถานะของระบบจากผลตอบตามลำดับ การตรวจสอบคุณสมบัติทั้งสองนี้ขึ้นกับการตรวจสอบอันดับของเมตริกซ์ที่มีมิติ n x nm เป็นหลัก ซึ่งจากการประมาณปริมาณของการคำนวณที่ต้องใช้ในการตรวจสอบ อันดับของเมตริกซ์โดยใช้วิธีเชิงเลข ได้ว่าวิธีตรวจสอบเมตริกซ์ซึ่งมีมิติ n x nm นี้ หาก m ไม่เท่ากับ 1 ใช้การคำนวณ $11n^3$ ซึ่งจะเสียเวลามากในระบบใหญ่ๆ

แต่จากหลักการว่าอันดับของเมตริกซ์ MM^T ซึ่งมีมิติ n x n มีค่าเท่ากับอันดับของเมตริกซ์ M ซึ่งมีมิติ n x nm ซึ่งปรากฏอยู่ใน Datta (1984) ปริมาณการคำนวณที่ต้องใช้ในการตรวจสอบตามเทคนิคนี้มีค่าเท่ากับ $7n^3$ เท่านั้น ดังนั้นในวิทยานิพนธ์นี้จึงใช้เทคนิคนี้ในการการตรวจสอบอันดับของเมตริกซ์ ขั้นตอนของอัลกอริทึมสามารถสรุปได้ดังนี้

2.1 ถ้ากำหนด Controllability Matrix หรือ Observability Matrix มา ให้ตรวจสอบมิติของเมตริกซ์ว่ามีจำนวนคอลัมน์มากกว่าจำนวนแถวหรือไม่ ถ้าจำนวนคอลัมน์มากกว่าให้หาเมตริกซ์ M จากสมการ

$$M = WW^T \qquad (3\text{-}14)$$

จากนั้นข้ามไปจากหัวข้อ 2.3 เพื่อหาอันดับของเมตริกซ์

2.2 ถ้าให้ตรวจสอบ Controllability จากคู่เมตริกซ์ A,B ให้ใช้สมการเพื่อหา M ดังนี้

$$M = \sum_{i=0}^{n-1} A^i BB^T (A^T)^i \qquad (3\text{-}15)$$

ในกรณีต้องการตรวจสอบ Observability จากคู่เมตริกซ์ A,C ให้ใช้เมตริกซ์ A^T แทน A และใช้เมตริกซ์ C^T แทน B

2.3 ตรวจสอบอันดับของเมตริกซ์ด้วยวิธี Gaussian Elimination แบบ Partial Pivoting

3. การหา Transfer Function Matrix จากสมการสถานะ Transfer Function Matrix เป็นเมตริกซ์ที่บอกถึงความสัมพันธ์ระหว่างสัญญาณออกและสัญญาณเข้าของระบบ ซึ่งในการวิเคราะห์หรือออกแบบจะมีอยู่หลายกรณีที่ต้องการข้อมูลอยู่ในรูปของ Transfer Function Matrix สำหรับอัลกอริทึมในวิทยานิพนธ์ถูกนำเสนอโดย Perry,Sun and Berger (1988) โดยแบ่งการคำนวณเป็น 2 ขั้นตอน คือ การคำนวณหาค่าสัมประสิทธิ์ของสมการลักษณะจากเมตริกซ์ที่ใช้ในการตรวจสอบ Controllability จากนั้นจึงหาสัมประสิทธิ์ของเทอมเศษจากการแปลงให้อยู่ในรูป Phase-variable Canonical สมการสถานะของระบบที่เราสนใจ สามารถเขียนได้ดังนี้

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} \end{aligned} \qquad (3\text{-}16)$$

เมื่อ x คือเวกเตอร์ตัวแปรสถานะมิติ n u คือเวกเตอร์สัญญาณควบคุมมิติ m y คือเวกเตอร์ผลตอบของระบบมิติ p คือ Transfer Function Matrix สามารถหาได้จากสมการ

$$\mathbf{H}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} \qquad (3\text{-}17)$$

โดยสมาชิกแต่ละตัวใน Transfer Function Matrix อยู่ในรูป

$$H_{ij}(s) = \frac{p_1 + p_2 s + \ldots + p_m s^m}{q_1 + q_2 s + \ldots + q_n s^n} \qquad (3\text{-}18)$$

การหาสัมประสิทธิ์ของสมการลักษณะ หาได้จากสมการ

$$q = -A^n b \qquad (3\text{-}19)$$

เมื่อ $q = [q_1\ q_2\ \ldots\ q_n]^T$ สำหรับ b เป็นเวกเตอร์ที่เลือกจากเมตริกซ์ B โดยพิจารณาว่าสัญญาณควบคุมใดที่ทำให้ระบบ Controllable หลังจากหาค่าสัมประสิทธิ์ของสมการ

ลักษณะได้ จึงคำนวณหาสัมประสิทธิ์ของเทอมที่เป็นเศษ โดยอาศัยแปลงเมตริกซ์ให้อยู่ใน Phase-variable Canonical โดยใช้สมการ

$$\begin{aligned} M_i &= \begin{bmatrix} m_1 & m_2 & \cdots & m_n \end{bmatrix} \\ m_n &= b \\ m_j &= \Lambda m_{j+1} + q_{j+1} b_i \quad ; j = (n-1)\ldots 1 \end{aligned} \tag{3-20}$$

จากนั้นจึงคำนวณหาสมาชิกต่างๆของ Transfer Function Matrix จากสมการ

$$H_i(s) = \frac{\left[CM_i S \right]}{q(s)} \quad ; S = \begin{bmatrix} 1 \\ s \\ \vdots \\ s^{n-1} \end{bmatrix} \tag{3-21}$$

4. การหาค่า Gain Margin และ Phase Margin ค่า Gain Margin และ Phase Margin จัดเป็นพารามิเตอร์ที่สำคัญในระบบควบคุม เนื่องจากค่าทั้งสองนี้สามารถใช้แสดงถึงเสถียรภาพแบบสัมพันธ์ได้ ในการคำนวณหาค่า Gain Margin นั้นทำได้ ในหลายวิธีเช่น อาศัย Routh table เป็นต้น กรณีของ Phase Margin นั้นมักนิยมใช้วิธีแบบกราฟฟิกเป็นหลัก คือ การหาจากกราฟผลตอบเชิงความถี่แบบต่างๆ เช่น Nyquist plot Bode plot เป็นต้น ในวิทยานิพนธ์ฉบับนี้ อาศัยเทคนิคการหาคำตอบแบบ Numerical Search คือ วิธี Secant ในการหาค่า Gain Margin และ Phase Margin เมื่อกำหนดให้ $F(j\omega) = \angle G(j\omega) - 180^\circ$ ในกรณีของการหา Gain Margin ซึ่งได้ผลเป็นเป็นความถี่ ณ ตำแหน่งที่เกิด Phase-crossover แล้วนำความถี่ที่ได้ไปแทนค่าหาค่าขนาดของ $1/G(j\omega)H(j\omega)$ ซึ่งก็คือ Gain Margin สำหรับการหา Phase Margin อาศัยการหาค่าความถี่ ณ ตำแหน่งที่เกิด Gain-crossover แล้วนำผลไปแทนค่า $\angle G(j\omega)H(j\omega) + 180^\circ$ จึงได้เป็น Phase Margin ขั้นตอนสำหรับการหา Gain Margin และ Phase Margin จึงสรุปได้ดังนี้

4.1 ใช้เทคนิคการค้นหาแบบ Secant หาค่าความถี่ ω ซึ่งทำให้เงื่อนไข $\angle G(j\omega) - 180^\circ$ เท่ากับศูนย์เป็นจริง

4.2 นำค่าความถี่ที่ได้ไปแทนใน $1/\left|G(j\omega)H(j\omega)\right|$ ซึ่งได้เป็นค่า Gain Margin

4.3 หาค่าความถี่ ω ซึ่งทำให้เงื่อนไข $\left|G(j\omega)H(j\omega)\right|$ เท่ากับหนึ่งเป็นจริง โดยใช้เทคนิคการค้นหาแบบ Secant

4.4 นำค่าความถี่ที่ได้ไปแทนใน $\angle G(j\omega)H(j\omega) + 180^\circ$ เพื่อเป็นค่า Phase Margin

3.3 <u>อัลกอริทึมที่ใช้ในการออกแบบระบบควบคุม</u>

1. การหาข้อมูลสำหรับการทดสอบแบบวงปิด ความถี่ในการแกว่งของผลตอบเชิงเวลาของระบบ จัดเป็นข้อมูลที่สำคัญสำหรับใช้ปรับค่าพารามิเตอร์ของตัวควบคุมแบบ PID ด้วยวิธีการทดสอบแบบวงปิด เนื่องจากในบางระบบยอมให้เกิดการแกว่งไม่ได้ เพราะอาจทำให้เกิดความเสียหายแก่อุปกรณ์ในระบบได้ ดังนั้นโปรแกรมจึงทำการคำนวณค่าความถี่ของการแกว่งเอง เพื่อไม่ต้องไปทดลองกับระบบจริง เมื่อพิจารณาความสัมพันธ์ระหว่างค่าความถี่การแกว่งกับผลตอบในเชิงความถี่ของระบบ พบว่าค่าความถี่ของการแกว่งสามารถหาได้จากความถี่ ณ ตำแหน่งที่เกิด Phase-crossover ดังนั้นการหาความถี่ ณ ตำแหน่งที่เกิด Phase-crossover จึงใช้เงื่อนไข

$$\angle G(j\omega)H(j\omega) = -180^\circ \qquad (3\text{-}22)$$

สำหรับข้อมูลอีกตัวของการคำนวณค่าพารามิเตอร์ของตัวควบคุม PID โดยวิธีแบบวงปิด คือ อัตราขยายที่ทำให้เกิดการแกว่งซึ่งมีค่าเท่ากับค่า Gain Margin ของระบบ ซึ่งหาจากค่าความถี่ที่เกิด Phase-crossover สำหรับขั้นตอนในการหาข้อมูลสำหรับการทดสอบแบบวงปิดสามารถสรุปได้ดังนี้

1.1 หาค่าความถี่ ซึ่งตรงกับเงื่อนไขในสมการ (3-22) โดยใช้การค้นหาคำตอบด้วยวิธี Secant

1.2 หาค่า Gain Margin

2. การออกแบบตัวชดเชยทั้งสามแบบ เทคนิคการออกแบบตัวชดเชยแบบล้ำหน้าและล้าหลัง เป็นการออกแบบในเชิงความถี่เท่านั้น โดยข้อกำหนดของการออกแบบจะอยู่ในรูปของ Phase Margin และขนาดของความผิดพลาดในสภาวะอยู่ตัวของระบบ ขั้นตอนคร่าวๆของการออกแบบสามารถศึกษาได้จาก (Kuo, 1993; Golten and Verwer, 1991; Stefani and others, 1993) อัลกอริทึมที่ใช้ในการออกแบบตัวชดเชยแบบล้ำหน้าและล้าหลังมีขั้นตอนดังนี้

2.1 ตรวจสอบว่ามีความผิดพลาดในสภาวะอยู่ตัวหรือไม่ ถ้าความผิดพลาดในสภาวะอยู่ตัวมีค่าเป็นศูนย์ให้ข้ามไปทำขั้นตอน 2.2

2.1.1 รับข้อมูลคือขนาดของความผิดพลาดในสภาวะอยู่ตัวที่ต้องการ

2.1.2 วิเคราะห์ความเป็นไปได้ของการใช้อัตราขยาย เพื่อเปลี่ยนขนาดของความผิดพลาดในสภาวะอยู่ตัวของระบบ หากไม่สามารถเปลี่ยนขนาดของความผิดพลาดด้วยการใช้อัตราขยายได้ให้แจ้งต่อผู้ใช้ แล้วจบการทำงาน

2.1.3 คำนวณอัตราขยาย ซึ่งทำให้ความผิดพลาดในสภาวะอยู่ตัวมีค่าเท่ากับขนาดที่กำหนดมาจากผู้ใช้ในหัวข้อ 2.1

2.2 รับข้อมูลคือ Phase Margin (PM) และประเภทของตัวชดเชยที่จะทำการออกแบบ

2.3 หา phase margin (PM_o) ของระบบเดิม

2.4 เปรียบเทียบ PM_o กับค่า PM ว่าใกล้เคียงพอที่จะยอมรับได้หรือไม่ โดยหากค่าทั้งสองใกล้เคียงจนพอยอมรับได้ ให้จบการทำงาน

2.5 แยกการทำงานออกเป็น 3 กรณีตามชนิดของตัวชดเชย

2.5.1 ตัวชดเชยแบบล้ำหน้า ให้ทำตามขั้นตอนต่อไปนี้

2.5.1.1 กำหนดให้ ϕ_m = PM - PM_o + E โดย E คือมุมที่เผื่อไว้ และมีค่าเริ่มต้นคือ 5°

2.5.1.2 คำนวณ α จาก $\alpha = \frac{1-\sin\phi_m}{1+\sin\phi_m}$

2.5.1.3 ตรวจสอบเพื่อหาความถี่ ω_m ซึ่งทำให้ระบบเปิดมีอัตราขยายเท่ากับ $0\log\alpha$

2.5.1.4 คำนวณ $\omega_1 = \omega_m\sqrt{\alpha}$ และ $\omega_2 = \frac{\omega_m}{\sqrt{\alpha}}$ แล้วแทนค่าทั้งสองที่คำนวณได้ไปหาฟังก์ชันถ่ายโอนของตัวชดเชยแบบล้ำหน้าคือ

$$G_c(s) = \frac{\left(1+\frac{s}{\omega_1}\right)}{\left(1+\frac{s}{\omega_2}\right)} \qquad (3\text{-}23)$$

2.5.2 ตัวชดเชยแบบล้าหลัง ให้ทำตามขั้นตอนต่อไปนี้

2.5.2.1 กำหนดให้ ϕ_m = PM - PM_o + E โดย E คือมุมที่เผื่อไว้ และมีค่าเริ่มต้นคือ 5°

2.5.2.2 ตรวจสอบหาความถี่ ω_m ซึ่งทำให้ค่ามุมเฟสของระบบเปิด เท่ากับ $180^\circ - \phi_m$

2.5.2.3 คำนวณอัตราขยาย G ที่ความถี่ ω_m แล้วแทนลงในสูตร $20\log\beta = G$ จากนั้นจึงคำนวณหาค่า β ออกมา

2.5.2.4 คำนวณ $\omega_2 = \omega_m/10$ และ $\omega_1 = \omega_2/\beta$ เพื่อหาฟังก์ชันถ่ายโอนของตัวชดเชยแบบล้าหลังคือ

$$G_c(s) = \frac{\left(1+\frac{s}{\omega_1}\right)}{\left(1+\frac{s}{\omega_2}\right)} \qquad (3\text{-}24)$$

2.5.3 ตัวชดเชยแบบล้ำหน้า-ล้าหลัง ให้ทำตามขั้นตอนต่อไปนี้

2.5.3.1 กำหนดให้ $\phi_m = PM - PM_o + E$ โดย E คือมุมที่เผื่อไว้ และมีค่าเริ่มต้นคือ 5°

2.5.3.2 คำนวณ α จากสมการ $\alpha = \frac{1-\sin\phi_m}{1+\sin\phi_m}$

2.5.3.3 ตรวจสอบหา ω_m ซึ่งทำให้อัตราขยายของระบบเปิดเท่ากับ $10\log\alpha$

2.5.3.4 คำนวณหาค่าพารามิเตอร์ $T = \frac{1}{\omega_m\sqrt{\alpha}}$, $\omega_4 = 1/T$, $\omega_3 = 1/(\alpha T)$, $\omega_2 = 0.1\omega_3$ และ $\omega_1 = \omega_2/2$ แล้วแทนในฟังก์ชันถ่ายโอนของตัวชดเชยแบบล้ำหน้าล้าหลัง ได้ว่า

$$G_c(s) = \frac{\left(1+\frac{s}{\omega_2}\right)\left(1+\frac{s}{\omega_3}\right)}{\left(1+\frac{s}{\omega_1}\right)\left(1+\frac{s}{\omega_4}\right)} \qquad (3\text{-}25)$$

2.6 หา Phase Margin (PM_n) ของระบบที่ต่อตัวชดเชยแล้ว

2.7 เปรียบเทียบค่าของ PM_n กับ PM ว่าใกล้เคียงพอที่จะยอมรับได้หรือไม่ หากยอมรับได้ให้แสดงพารามิเตอร์ของตัวชดเชยที่คำนวณได้ แล้วจบการทำงาน ถ้ายอมรับไม่ได้ให้เปลี่ยนค่า E แล้วย้อนกลับไปทำขั้นตอน 2.5 โดยถ้า $PM_n < PM$ ให้เพิ่มค่า E แต่ถ้า $PM_n > PM$ ให้ลดค่า E

3. การออกแบบโดยเทคนิคการกำหนดตำแหน่งเสา การออกแบบโดยเทคนิคการกำหนดตำแหน่ง Pole เป็นเทคนิคการออกแบบพื้นฐานที่มีประสิทธิภาพในระดับพอสมควร หากว่ามีสัญญาณควบคุมเพียงสัญญาณเดียว เทคนิคการออกแบบนี้จะสามารถคำนวณหาค่าอัตราขยายป้อนกลับได้หลายวิธี โดยแต่ละวิธีให้ผลออกมาเช่นเดียวกัน สำหรับวิธีที่ใช้ในกรณีสัญญาณควบคุมเดียว เช่น การแทนค่าในสูตรของ Ackermann เป็นต้น แต่หากมีสัญญาณควบคุมมากกว่าสัญญาณเดียวแล้ว การคำนวณหาค่าอัตราขยายจะทำได้ค่อนข้างลำบาก และอัตราขยายที่คำนวณได้ จะมีค่าที่ไม่แน่นอนแล้วแต่วิธีที่ใช้ในการคำนวณ อัลกอริทึมซึ่งใช้ในวิทยานิพนธ์นำเสนอโดย Patel and Misra (1984) โดยใช้หลักการของการแปลงรูปสมการสถานะไปอยู่ในรูปของ Unreduced Upper Hessenberg หลังจากแปลง เมตริกซ์เสร็จแล้วจึงใช้การแปลงให้เมตริกซ์อยู่ในรูป Real Schur แล้วจึงคำนวณหาเมตริกซ์ของอัตราขยายป้อนกลับ สำหรับขั้นตอนในการคำนวณ มีดังนี้

3.1 ทำการแปลงคู่เมตริกซ์ [A,b] ให้อยู่ในรูป Unreduced Upper Hessenberg [F,g] ตามขั้นตอนดังนี้

3.1.1 กำหนดให้ i = 0 แล้วทำการแปลงแบบ Householder เพื่อแปลงเมตริกซ์ b ให้อยู่ในรูป

$$T_0^T b = \left[g_1^{(0)}\ 0 \cdots 0\right] = g_0 \qquad (3\text{-}26)$$

3.1.2 หาเมตริกซ์ $F_0 = T_0^T A T_0$ และ T = T_0

3.1.3 ถ้า i ≠ n-2 ให้แบ่งเมตริกซ์ F_i ให้อยู่ในรูป

$$F_i = \begin{bmatrix} f_{11}^{(i)} & f_{12}^{(i)} & \cdots & f_{1,i+1}^{(i)} & f^{T(i)}_{1,i+2} \\ f_{21}^{(i)} & f_{22}^{(i)} & \cdots & f_{2,i+1}^{(i)} & f^{T(i)}_{2,i+2} \\ 0 & f_{32}^{(i)} & \cdots & f_{3,i+1}^{(i)} & f^{T(i)}_{3,i+2} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & f_{i+1,i+1}^{(i)} & f^{T(i)}_{i+1,i+2} \\ 0 & 0 & \cdots & f_{i+2,i+1}^{(i)} & F^{T(i)}_{i+2,i+2} \end{bmatrix} \qquad (3\text{-}27)$$

3.1.4 ตั้งค่า i = i + 1 และ $b_i = f_{i+1,j}^{i-1}$

3.1.5 ถ้า $b_i \neq 0$ ให้ทำการแปลงแบบ Householder เพื่อแปลงเมตริกซ์ b_i ให้อยู่ในรูป

$$T_i^T b_i = \left[f_{i+1,j}^{(i)}\ \ 0\ \cdots\ 0\right]^T \qquad (3\text{-}28)$$

3.1.6 คำนวณ

$$F_i = \begin{bmatrix} I_i & 0 \\ 0 & T_i^T \end{bmatrix} F_{i-1} \begin{bmatrix} I_i & 0 \\ 0 & T_i \end{bmatrix} \qquad (3\text{-}29)$$

$$T = \begin{bmatrix} I_i & 0 \\ 0 & T_i \end{bmatrix} \qquad (3\text{-}30)$$

ย้อนกลับไปทำขั้นตอนที่ 3.1.3

3.1.7 ถ้า $f^{(i)}_{i+2,i+1} = 0$ กำหนดให้ $\mu = n-1$ หรือ $\mu = n$ แล้วกลับไปทำ

3.1.8 ตั้งค่า $\mu = i$ และตั้งค่า $F = F_i$ และ $G = G_0$

3.2 ในกรณีให้เมตริกซ์ B เป็นแบบหลายคอลัมน์ ให้ทำตามขั้นตอนดังนี้

3.2.1 ตั้ง $i = 1$ $A_1 = A$ และ b_1 เป็นคอลัมน์แรกของเมตริกซ์ B

3.2.2 ใช้เทคนิคในข้อ 3.1 เพื่อเปลี่ยนรูปคู่เมตริกซ์ $[A_1,b_1]$ ให้อยู่ในรูป Unreduced Upper Hessenberg แล้วนำเมตริกซ์ T_1 และค่า μ_1 ที่คำนวณได้มาเก็บ

3.2.3 คำนวณ $F_1 = T_1^T A T_1$ $G_1 = T_1^T B$ $V = T_1$ $n_c = \mu_1$

3.2.4 ถ้า $n_c \neq n$ ให้แบ่ง F_i และ G_i ให้อยู่ในรูป

$$F_i = \begin{bmatrix} F^{(i)}_{11} & F^{(i)}_{12} & \cdots & F^{(i)}_{1,i} & F^{(i)}_{1,i+1} \\ 0 & F^{(i)}_{22} & \cdots & F^{(i)}_{2,i} & F^{(i)}_{2,i+1} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & F^{(i)}_{i,i} & F^{(i)}_{i,i+1} \\ 0 & 0 & \cdots & 0 & F^{(i)}_{i+1,i+1} \end{bmatrix}$$

$$G_i = \begin{bmatrix} g^{(i)}_{11} & g^{(i)}_{12} & \cdots & g^{(i)}_{1i} & g^{(i)}_{1,i+1} & \cdots & g^{(i)}_{1,m} \\ 0 & g^{(i)}_{22} & \cdots & g^{(i)}_{2i} & g^{(i)}_{1,i+2} & \cdots & g^{(i)}_{2,m} \\ \vdots & \vdots & & \vdots & \vdots & & \\ 0 & 0 & \cdots & g^{(i)}_{ii} & 0 & \cdots & g^{(i)}_{i,m} \\ 0 & 0 & \cdots & 0 & 0 & \cdots & g^{(i)}_{i+1,m} \end{bmatrix} \qquad (3\text{-}31)$$

3.2.5 กำหนดให้ค่า $i = i+1$ $A_i = F^{(i-1)}_{i,i}$ และ $b_i = g^{(i-1)}_{ii}$

3.2.6 ใช้เทคนิคในข้อ 3.1 เพื่อเปลี่ยนรูปคู่เมตริกซ์ $[A_i,b_i]$ ให้อยู่ในรูป Unreduced Upper Hessenberg แล้วนำเมตริกซ์ T_i และค่า μ_i ที่คำนวณได้มาเก็บ

3.2.7 คำนวณ

$$F_i = \begin{bmatrix} I_{n_c} & 0 \\ 0 & T_i^T \end{bmatrix} F_{i-1} \begin{bmatrix} I_{n_c} & 0 \\ 0 & T_i \end{bmatrix} \quad (3\text{-}32)$$

$$G_i = \begin{bmatrix} I_{n_c} & 0 \\ 0 & T_i^T \end{bmatrix} G_{i-1} \quad (3\text{-}33)$$

$$= V \begin{bmatrix} I_{n_c} & 0 \\ 0 & T_i \end{bmatrix} \quad (3\text{-}34)$$

3.2.8 ตั้ง F = F_i G = G_i และ ρ = I

3.3 กำหนดให้ F_1 = F , g_1 = g , N = Identity Matrix และ k = k_0 = 0

3.4 การกำหนดตำแหน่งเสา ซึ่งเป็นค่าจริงสามารถทำได้ด้วยขั้นตอนดังนี้

3.4.1 หาเมตริกซ์ P_i ที่เป็น Orthogonal Matrix ซึ่งทำให้

$$f_i^T P_i = \pm \|f_i\|_2 e_n^T \quad (3\text{-}35)$$

เมื่อกำหนดให้ f_i = [0 0 ... $f_{n,n-1}$ f_{nn}-λ]^T และ $\|f_i\|_2 = \sqrt{f_i^T f_i}$ และ e_n^T = [0 0 ... 1]

3.4.2 คำนวณ $\overline{F}_i^T = P_i^T F_i P_i$ และ $\overline{g}_i = P_i^T g_i$

3.4.3 เปลี่ยน $\overline{F}_i^T$ ให้เป็นเมตริกซ์ H_i ซึ่งอยู่ในรูป Hessenberg โดยใช้การแปลงแบบ Orthogonal ในขณะเดียวกันก็ใช้เมตริกซ์ U_i ที่ได้จากการแปลงไปคูณกับเมตริกซ์ g_i

$$\begin{aligned} U_i^T \overline{F}_i^T U_i &= H_i \\ g_{i+1} &= U_i^T g_i \end{aligned} \quad (3\text{-}36)$$

3.4.4 คำนวณค่าเวกเตอร์ k ซึ่งทำให้สมาชิก (i+1,i) ของเมตริกซ์ $H_i - g_{i+1}k_i$ มีค่าเป็นศูนย์

3.4.5 คำนวณ $N_i = P_i U_i$, $N = NN_i$ และ $k = k + k_i N^T$

3.4.6 คำนวณ $F_{i+1} = H_i - g_{i+1}k_i$

3.4.7 ย้อนกลับไปทำการกำหนดค่าตำแหน่งเสาจนครบ

3.5 การกำหนดค่าตำแหน่งเสา ซึ่งเป็นค่าเชิงซ้อน ใช้ขั้นตอนดังนี้

3.5.1 หาเมตริกซ์ P_i ที่เป็น Orthogonal Matrix ซึ่งทำให้

$$f_i^T P_i = \pm \|f_i\|_2 e_n^T \qquad (3\text{-}37)$$

เมื่อกำหนดให้ $f_i = [0\ 0\ \ldots\ f_{n,n-2}\ f_{n,n-1}\ f_{nn}-\lambda]^T$ และ $\|f_i\|_2 = \sqrt{f_i^T f_i}$ และ $e_n^T = [0\ 0\ \ldots\ 1]$

3.5.2 คำนวณ $\overline{F}_i^T = P_i^T F_i P_i$ และ $\overline{g}_i = P_i^T g_i$

3.5.3 เปลี่ยน $\overline{F}_i^T$ ให้เป็นเมตริกซ์ H_i ซึ่งใกล้เคียงรูป Hessenberg โดยใช้การแปลงแบบ Orthogonal สำหรับตำแหน่งซึ่งไม่สามารถกำจัดด้วยวิธีการแปลงแบบ Orthogonal คือ ตำแหน่ง (3,1) ในขณะเดียวกันก็ใช้เมตริกซ์ U_i ที่ได้จากการแปลงไปคูณกับเมตริกซ์ g_i

$$\begin{aligned} U_i^T \overline{F}_i^T U_i &= H_i \\ g_{i+2} &= U_i^T g_i \end{aligned} \qquad (3\text{-}38)$$

3.5.4 คำนวณค่าเวกเตอร์ k ซึ่งทำให้สมาชิกที่ตำแหน่ง (i+1,i) และ (i+2,i+1) ของเมตริกซ์ $H_i - g_{i+2}k_i$ มีค่าเป็นศูนย์

3.5.5 คำนวณ $N_i = P_i U_i$, $N = NN_i$ และ $k = k + k_i N^T$

3.5.6 คำนวณ $F_{i+2} = H_i - g_{i+2}k_i$

3.5.7 ย้อนกลับไปทำการกำหนดค่าตำแหน่ง Pole จนครบ

4. การแก้สมการ ARE โดยใช้เทคนิค Schur สมการ ARE (Algebraic Riccati Equation) เป็นสมการที่มีความสำคัญมากในการออกแบบระบบควบคุม เพราะสมการนี้มีใช้อยู่ในการออกแบบหลายอย่าง ตัวอย่างของการประยุกต์ใช้สมการนี้ เช่น การหาค่าเมตริกซ์อัตราขยายป้อนกลับโดยเทคนิคเล็งเลิศ (Optimal) เทคนิคการแก้สมการนี้ที่ใช้กันทั่วไป คือ การใช้วิธีเชิงเลข เช่น Conjugate Gradient ในการแก้หาคำตอบของสมการ แต่การใช้วิธีเชิงเลขในการแก้หาคำตอบของสมการ ARE ยังมีจุดอ่อนอยู่ที่ความเร็วในการคำนวณยังช้าอยู่ สำหรับอัลกอริทึมที่ใช้ในวิทยานิพนธ์ ใช้การแปลงรูปของเมตริกซ์ Hamiltonian ให้อยู่ในรูป Real Schur แล้วจึงอาศัยการจัดตำแหน่งของ Eigenvalue ให้เรียงตามขนาด จากนั้นจึงใช้การแก้สมการเพื่อหาคำตอบ เทคนิคการแก้สมการ ARE โดยใช้การแปลงรูปแบบ Schur มีจุดเด่นอยู่ที่การแก้สมการจะสามารถหาคำตอบออกมาได้โดยตรง ด้วยการแก้สมการเพียงครั้งเดียว ทำให้มีความเร็วมากกว่าการใช้วิธีเชิงเลขในการค้นหาคำตอบ สมการ ARE โดยทั่วไปสามารถเขียนได้ในรูปแบบดังนี้

$$A^T P + PA - PSP + Q = 0 \qquad (3\text{-}39)$$

เมตริกซ์ทั้งหมดมีมิติ n x n โดยเมตริกซ์ S และ Q เป็นเมตริกซ์แบบสมมาตร และเป็น Positive Definite ในการคำนวณจะสมมุติว่าคู่เมตริกซ์ [A,B] สามารถทำให้มีเสถียรภาพได้ (Stabilizable) และคู่เมตริกซ์ [A,C] สามารถตรวจวัดได้ (Detectable) ภายใต้ข้อสมมุติทั้งสองข้อนี้ ได้ว่าสมการจะมีคำตอบซึ่งเป็น Positive Semidefinite และมีเพียงคำตอบเดียว โดยเมื่อพิจารณาเมตริกซ์ Hamiltonian

$$H = \begin{bmatrix} A & -S \\ -Q & -A^T \end{bmatrix} \qquad (3\text{-}40)$$

จากข้อสมมุติทั้งสองข้อจะรับประกันได้ว่า H จะไม่มีเสาที่มีค่าเชิงซ้อนอย่างเดียว แต่จะเป็นค่าสังยุกต์เชิงซ้อน ดังนั้นจึงสามารถหาเมตริกซ์ U สำหรับทำการแปลงรูปแบบ Orthogonal เพื่อแปลงให้เมตริกซ์ H อยู่ในรูป Real Schur ได้

$$T = U^T H U = \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix} \qquad (3\text{-}41)$$

หลังจากนั้นจึงใช้การแปลงรูปเมตริกซ์แบบ Orthogonal เพื่อเปลี่ยนตำแหน่งของสมาชิกในแนวแทยงมุม 3 แถวของเมตริกซ์ T_{11} และ T_{22} ซึ่งหมายถึง Eigenvalue ของเมตริกซ์ทั้งสองให้มีส่วนจริงเป็นลบและเป็นบวกตามลำดับ หลังจากแปลงรูปเมตริกซ์เสร็จจะได้เมตริกซ์ U ซึ่งเมื่อแบ่งออกเป็นเมตริกซ์ที่มีมิติ n x n จะได้

$$\mathrm{U} = \begin{bmatrix} \mathrm{U}_{11} & \mathrm{U}_{12} \\ \mathrm{U}_{21} & \mathrm{U}_{22} \end{bmatrix} \qquad (3\text{-}42)$$

จากนั้นจึงหาคำตอบของสมการ ARE ได้จาก

$$X = U_{21} U_{11}^{-1} \qquad (3\text{-}43)$$

สำหรับอัลกอริทึมที่ใช้ในการคำนวณขั้นตอนต่าง ๆ เหล่านี้ สามารถศึกษาได้จาก (Arnold and Laub, 1984; Petkov and other, 1991; Stewart, 1976) สำหรับขั้นตอนการคำนวณสามารถสรุปได้ดังนี้

4.1 สร้างเมตริกซ์ H ตามสมการ (3-40)

4.2 ทำการแปลงแบบ Orthogonal เพื่อเปลี่ยน H ให้เป็นเมตริกซ์ T ซึ่งอยู่ในรูป Real Schur

4.3 ทำการสลับตำแหน่งของ Eigenvalue ใน T_{11} ของเมตริกซ์ T มีเฉพาะ Eigenvalue ที่มีค่าจริงเป็นค่าลบ

4.4 แก้สมการ $U_{11}^T P = U_{21}^T$ เพื่อหาค่าเมตริกซ์ P

2.4 อัลกอริทึมสำหรับการจำลองผลตอบ

1. การจำลองผลตอบเชิงเวลาของระบบ ผลตอบเชิงเวลาของระบบสามารถใช้ในการเปรียบเทียบประสิทธิภาพของระบบได้โดยตรง แต่เนื่องจากความสัมพันธ์ระหว่างปริมาณต่าง ๆของผลตอบเชิงเวลากับข้อมูลของระบบในรูปแบบต่าง ๆเช่น ฟังก์ชันถ่ายโอน ไม่สามารถหาออกมาเป็นสมการที่แน่นอนได้ ดังนั้นหากต้องการใช้ผลตอบเชิงเวลาเป็นเกณฑ์วัดประสิทธิภาพของระบบควบคุม ก็ต้องใช้การจำลอง (Simulation) เพื่อสร้างกราฟของผลตอบเชิงเวลาขึ้นแล้วจึงใช้การวัดค่าต่าง ๆจากกราฟเพื่อใช้ในการพิจารณาประสิทธิภาพ เทคนิคการจำลองผลตอบเชิงเวลาที่มีใช้กันอยู่ในปัจจุบันมีอยู่มากมาย แต่ในวิทยานิพนธ์นี้เลือกใช้เทคนิค Runge-Kutta (order 4) ซึ่งเป็นเทคนิคที่ใช้กันอยู่ทั่วไปในการแก้สมการอนุพันธ์ การจำลองผลตอบแบบ Runge-Kutta นี้สามารถใช้ได้ในกรณีที่ฟังก์ชันที่ต้องการจำลองอยู่ในรูปสมการอนุพันธ์เท่านั้น ดังนั้นในกรณีข้อมูลของระบบอยู่ในรูปของฟังก์ชันถ่ายโอนจึงต้องเปลี่ยนให้อยู่ในรูปของสมการอนุพันธ์ก่อน โดยปกติแล้ว ข้อมูลของระบบมักมีตัวแปรมากกว่าหนึ่งขึ้นไป ดังนั้นในการจำลองผลจึงไม่ใช้เทคนิค Runge-Kutta โดยตรง แต่มีการประยุกต์ให้ใช้กับข้อมูลในลักษณะของเมตริกซ์ ซึ่งจะพบว่ารูปของสมการสถานะเหมาะสมสำหรับการใช้ในกรณีนี้พอดี ดังนั้นขั้นตอนแรกของการจำลองผลตอบเชิงเวลา คือการเปลี่ยนรูปข้อมูลของระบบให้อยู่ในรูปของสมการสถานะเสียก่อน หลังจากนั้นจึงเป็นการใช้เทคนิค Runge-Kutta ในการหาคำตอบของสมการสถานะ สำหรับขั้นตอนต่าง ๆของการจำลองผลตอบเชิงเวลามีดังนี้

1.1 ตรวจสอบว่าข้อมูลของระบบอยู่ในรูปของสมการสถานะหรือไม่ ถ้าอยู่ในรูปของสมการสถานะแล้ว ให้ข้ามไปทำขั้นตอน 2

1.1.1 เปลี่ยนรูปแบบของข้อมูลที่เป็นฟังก์ชันถ่ายโอน

$$G(s) = \frac{b_{m+1}s^m + b_m s^{m-1} + \ldots + b_2 s + b_1}{s^n + a_n s^{n-1} + \ldots + a_2 s + a_1} \qquad (3\text{-}44)$$

ให้อยู่ในรูปของสมการสถานะ โดยรูปแบบของสมการสถานะที่เปลี่ยนได้ง่าย คือ รูปของ Phase-variable Canonical

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & & & 0 & 1 \\ -a_1 & -a_2 & \cdots & -a_{n-1} & -a_n \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \mathbf{u} \qquad (3\text{-}45)$$

$$\mathbf{y} = \begin{bmatrix} b_1 & b_2 & \cdots & b_m & 0 \end{bmatrix} x$$

1.2 รับข้อมูลคือ เวลาเริ่มต้น a เวลาสิ้นสุด b สภาวะเริ่มต้น x_0 จำนวนขั้นของการจำลองผล N และชนิดของสัญญาณทดสอบ u(t)

1.3 กำหนด F(z,t) = Az + Bu(t) , h = (b - a)/N , t = a , w(t) = x_0 และ z = x_0

1.4 ทำตามขั้นตอนต่อไปนี้ N ครั้ง

1.4.1 คำนวณ K_1 = hF(z,t)

1.4.2 คำนวณ K_2 = hF(z+K_1/2,t+h/2)

1.4.3 คำนวณ K_3 = hF(z+K_2/2,t+h/2)

1.4.4 คำนวณ K_4 = hF(z+K_3/2,t+h)

4.5 คำนวณ z = z + (K_1+2K_2+2K_3+K_4)/6

4.6 คำนวณ t = t + h

4.7 w(t) = z

2. การสร้างกราฟของผลตอบเชิงความถี่ทั้ง 3 แบบ กราฟของผลตอบเชิงความถี่ จัดเป็นเครื่องมือช่วยในการวิเคราะห์และออกแบบที่ดีมาก โดยเฉพาะในกรณีทฤษฎีควบคุมแบบคลาสสิก เนื่องจากการใช้ผลตอบเชิงเวลาเป็นเกณฑ์ของประสิทธิภาพยังมีจุดอ่อนอยู่ที่ไม่สามารถวิเคราะห์หาค่าต่าง ๆได้จากฟังก์ชันถ่ายโอน แต่ในการใช้ผลตอบเชิงความถี่จะไม่มีปัญหานี้ เพราะโดยพื้นฐานแล้วตัวแปรเชิงซ้อน s มีความเกี่ยวข้องโดยตรงกับตัวแปรความถี่ นอกจากนี้เทคนิคการคำนวณหาค่าต่าง ๆในการวิเคราะห์และออกแบบเชิง

ความถี่ได้พัฒนามาจนเป็นขั้นตอนที่ค่อนข้างแน่นอนแล้ว จึงง่ายต่อการใช้งาน เทคนิคการสร้างกราฟทั้งสามแบบมีรากฐานอยู่ที่การคำนวณค่าของฟังก์ชันถ่ายโอนที่ความถี่ต่าง ๆ ซึ่งโดยทั่วไปจะออกมาเป็นค่าเชิงซ้อน จากนั้นจึงนำไปแปลงรูปให้เหมาะกับการสร้างกราฟแบบต่าง ๆ สำหรับอัลกอริทึมที่ใช้คำนวณค่าเชิงซ้อนที่ความถี่ต่าง ๆของฟังก์ชันถ่ายโอนจะเป็นการคำนวณในกรณีฟังก์ชันถ่ายโอนเป็นแบบโพลิโนเมียล โดยใช้หลักการตัวกระทำเชิงซ้อน (Imaginary Operator, j) เมื่อถูกยกกำลังจะมีค่าที่ซ้ำกันเป็นช่วง ๆ ดังนั้นในขั้นตอนของการคำนวณจึงมีเฉพาะการบวกลบและการยกกำลังเท่านั้น ทำให้ประหยัดเวลาที่ใช้ในการคำนวณลงได้ สำหรับขั้นตอนของการคำนวณมีดังนี้

2.1 รับข้อมูลคือ ขอบเขตของการคำนวณ [a,b] จำนวนขั้นของการคำนวณ N และชนิดของกราฟที่ต้องการสร้าง

2.2 คำนวณ $h = (b - a)/n$ และกำหนดให้ $\omega = a$

2.3 คำนวณหาค่าเชิงซ้อนของเทอมที่เป็นส่วนโดยทำตามขั้นตอนต่อไปนี้

2.3.1 กำหนดให้ $i = 1$ $\beta = 0$ α = สัมประสิทธิ์ของเทอมซึ่งเป็นค่าคงที่

2.3.2 พิจารณาตัวแปร s ซึ่งมีกำลัง i ว่ามีสัมประสิทธิ์เป็นศูนย์หรือไม่ ถ้าสัมประสิทธิ์เป็นศูนย์ให้ข้ามไปทำขั้น 2.3.3 แต่ถ้าไม่เป็นศูนย์ให้พิจารณาว่า i เมื่อหารด้วย 4 แล้วเหลือเศษเท่าไร แล้วแยกทำตามขั้นตอนต่อไปนี้

2.3.2.1 เศษเป็นศูนย์ $\alpha = \alpha +$ (ค่าสัมประสิทธิ์ x ω^i)

2.3.2.2 เศษเป็นหนึ่ง $\beta = \beta +$ (ค่าสัมประสิทธิ์ x ω^i)

2.3.2.3 เศษเป็นสอง $\alpha = \alpha -$ (ค่าสัมประสิทธิ์ x ω^i)

2.3.2.4 เศษเป็นสาม $\beta = \beta -$ (ค่าสัมประสิทธิ์ x ω^i)

2.3.3 $i = i + 1$

2.3.4 หากว่ากำลังสูงสุดของเทอมที่เป็นส่วนน้อยกว่า i ให้ย้อนไปทำขั้น 2.3.2

2.3.5 Denominator = $\alpha + j\beta$

2.4 คำนวณหาค่าเชิงซ้อนของเทอมที่เป็นเศษ โดยวิธีเดียวกับในขั้นที่ 3 จากนั้นเก็บผลไว้ในตัวแปรชื่อ Numerator

2.5 $G(j\omega)$ = Numerator/Denominator

2.7 คำนวณ $\omega = \omega + h$ โดยถ้า $\omega < b$ ให้ย้อนไปทำขั้น 2.3

2.6 แปลงรูปข้อมูลค่าเชิงซ้อนเมื่อเปลี่ยนค่าความถี่ไปเรื่อยๆ ซึ่งเก็บไว้ใน $G(j\omega)$ ให้อยู่ในรูปที่เหมาะสำหรับ การสร้างกราฟตามชนิดของกราฟที่กำหนดมา เช่น กราฟแบบ Bode Plot ก็แปลงค่าเชิงซ้อนให้เป็นอัตราขยายกับเฟส เป็นต้น

3. การสร้างกราฟ Root Loci กราฟ Root Loci เป็นกราฟที่แสดงแนวทางการเปลี่ยนตำแหน่งเสาต่างๆ ของระบบวงปิดเมื่อมีการเปลี่ยนแปลงค่าพารามิเตอร์บางตัวของระบบไปเรื่อยๆ ในอัลกอริทึมนี้ ค่าพารามิเตอร์ซึ่งเปลี่ยนไปเรื่อยๆคือ อัตราขยาย K ซึ่งต่ออนุกรมเข้ากับกระบวนการ การคำนวณแนวทางการเปลี่ยนตำแหน่งของ pole นี้ หากใช้การแก้สมการเพื่อหารากทั้งหมดของสมการลักษณะ โดยเปลี่ยนค่า K ไปเรื่อยๆ พบว่าต้องใช้เวลาในการคำนวณมาก ดังนั้นเทคนิคที่ใช้ในโปรแกรม คือ การใช้หลักของ Continution ในการหาแนวการเดินของตำแหน่งเสาแยกจากกัน ซึ่งนำเสนอโดย Pan and Chao (1978)

$$G(s)H(s) = K\frac{P(s)}{Q(s)} = \frac{K(s-z_1)(s-z_2)\ldots(s-z_m)}{(s-p_1)(s-p_2)\ldots(s-p_n)} \qquad (3\text{-}46)$$

โดย G(s)H(s) คือฟังก์ชันถ่ายโอนของระบบเปิด K คืออัตราขยาย และ $m < n$ จากสมการ (3-46) ได้ว่ากราฟ Root Loci สร้างได้จากสมการ

$$g(s,K) = Q(s) + KP(s) = 0 \qquad (3\text{-}47)$$

แทนการแก้หารากของสมการ (3-46) โดยตรงสำหรับ K แต่ละค่าที่เปลี่ยนไปเรื่อยๆ สมการที่ใช้เป็นอัลกอริทึมอยู่ในรูป

$$\begin{aligned} \frac{dg(s,K)}{dt} &= -g(s(t),K(t)) \quad ; \quad g(s(0),K(0)) = 0 \\ \frac{dK}{dt} &= 1 \quad ; \quad K(0) = K_0 \end{aligned} \qquad (3\text{-}48)$$

โดย $s(0) = s_0$ เป็นรากของสมการเมื่อ $K = K_0$ และโดยอาศัยความสัมพันธ์ระหว่างตัวแปรสามารถเขียนสมการ (3-48) ใหม่ได้ดังนี้

$$\frac{ds}{dt} = -\frac{\left[g+\frac{dg}{dK}\right]}{dg/ds} \quad ; \quad s(0)=s_0$$
$$\frac{dK}{dt} = 1 \quad ; \quad K(0)=K_0 \qquad (3\text{-}49)$$

เมื่อเขียนอีกรูปหนึ่งคือ

$$\frac{ds}{dt} = \frac{\left[Q(s)+KP(s)\right]+P(s)}{Q(s)+KP(s)} \quad ; \quad s(0)=s_0$$
$$\frac{dK}{dt} = 1 \quad ; \quad K(0)=K_0 \qquad (3\text{-}50)$$

จากสมการ (3-50) โดยใช้เทคนิคการแก้สมการอนุพันธ์เชิงเส้นก็จะสามารถหาแนวทางการเดินของตำแหน่งเสาแต่ละแนวได้ ด้วยการแทนค่าเริ่มต้นของตำแหน่งรากลงในสมการ สำหรับเทคนิคการแก้สมการอนุพันธ์ที่นำมาใช้ในอัลกอริทึมนี้คือเทคนิคของ Euler ซึ่งจะได้ว่า

$$s_{k+1} = s_k - h\frac{Q(s_k)+K_kP(s_k)+P(s_k)}{Q(s_k)+K_kP(s_k)}$$
$$K_{k+1} = K_k + h \qquad (3\text{-}51)$$

ดังนั้นจึงสามารถสรุปขั้นตอนต่าง ๆของการสร้างกราฟ Root Loci ได้ดังนี้

3.1 รับข้อมูลคือ ขอบเขตการเปลี่ยนแปลงของ K คือ [a,b] และจำนวนขั้นของการจำลองกราฟ N

3.2 คำนวณ h = (b - a)/N และตำแหน่งเสาทั้งหมดของฟังก์ชันถ่ายโอนเมื่อ K = a

3.3 เลือกเสาที่ใช้ในการหาแนวทางเดินมา 1 ค่า โดยเสาที่เลือกต้องไม่ซ้ำกับเสาที่เคยหาไปแล้ว จากนั้นกำหนดให้ s = ตำแหน่งเสาที่เลือก

3.3.1 ทำตามขั้นตอนต่อไปนี้ N ครั้ง

3.3.1.1 คำนวณตามสมการ (3-51) โดย s_0 = s

3.3.1.2 แสดงตำแหน่งเสาที่คำนวณได้ออกทางกราฟ Root Loci

3.4. ตรวจสอบว่าหาแนวทางเดินของ Pole ครบแล้วหรือไม่ ถ้ายังไม่ครบให้ย้อนกลับไปทำในขั้นที่ 3.3