โปรแกรมออกแบบ
กรุ๊ฟดีเลย์อิควอไลเซอร์

GROUP DELAY EQUALIZER PROGRAM

โดย

วารินทร์ วีระสัย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต
สาขาเทคโนโลยีอิเล็กทรอนิกส์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2536

หัวข้อปริญญานิพนธ์ โปรแกรมออกแบบกรุ๊ฟดีเลย์อิควอไลเซอร์ (GROUP DELAY EQULIZER PROGRAM)

โดย

นายวารินทร์ วีระสัย เลขประจำตัว 35102024

อาจารย์ที่ปรึกษา อ. ไพศาล สิทธิโยภาสกุล

อ. อรรถสิทธิ หล่าสกุล

ภาควิชา เทคนิคอุตสาหกรรม

ปีการศึกษา 2536

คณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต

คณะกรรรมการสอบปริญญานิพนธ์

____________________ ประธานกรมการ

( )

____________________ กรรมการ

( )

____________________ กรรมการ

( )

____________________ กรรมการ

( )

____________________ กรรมการ

โปรแกรมออกแบบกรุ๊ฟดีเลย์อิควอไลเซอร์

GROUP DELAY EQUALIZER PROGRAM

โดย วารินทร์ วีระสัย 35102024

อาจารย์ที่ปรึกษา อ.ไพศาล สิทธิโยภาสกุล

อ. อรรถสิทธิ์ หล่าสกุล

บทคัดย่อ

โปรแกรมออกแบบกรุ๊ฟดีเลย์อิควอไลเซอร์ ใช้ในการออกแบบวงจรและแก้ไขความผิดเพี้ยนทางเวลา คือทำให้กรุ๊ฟดีเลย์มีค่าคงที่ให้มากที่สุด โดยอาศัยหลักการบังคับตำแหน่งโพลของทรานส์เฟอร์ฟังก์ชั่นที่ถูกประมาณขึ้น ในปริญญานิพนธ์นี้ยังได้พิจารณาผลตอบสนองในขอบข่ายของเวลาของวงจรที่ประมาณขึ้นด้วย

ABSTRACT

Group delay egualizer program is used to designed the delay egualizer eircuit it's mean to maximaly constant group delay. The approximated by applying the constraints on the poles of the function. In addition , time domain response of the approximate circuit is considered.

สารบัญ

บทที่ 1
บทนำ

ในการติดต่อสื่อสารโทรคมนาคม เรามีความจำเป็นที่จะต้องมีการชดเชยทั้งทางเฟสและทางขนาดของสัญญาณ โดยทำให้เฟสที่ได้เป็นเชิงเส้นให้มากที่สุดตามที่ต้องการในช่วงทรานส์มิสชันแบนด์ (TRANSMISSION BAND) หรืออีกนัยหนึ่งก็คือทำให้กรุ๊ฟดีเลย์ (GROUP DELAY) คงที่นั่นเอง ในปริญญานิพนธ์นี้ เป็นโปรแกรมในการประมาณฟังก์ชันของวงจรดีเลย์อิควอไลเซอร์เพื่อให้ได้ค่ากรุ๊ฟดีเลย์คงที่

การอิควอไลซ์กรุ๊ฟดีเลย์ ในที่นี้จะใช้กระบวนการอิเทอเรท (ITERATE) ซึ่งต้องมีการประมาณค่าเริ่มต้นให้ก่อน และโดยการใช้วิธีการของนิวตัน - ราฟสัน จะได้สมการเชิงเส้นออกมา เมื่อทำการแก้สมการเชิงเส้นนี้จะได้คำตอบซึ่งนำไปแทนค่าเพื่อหาค่าผลรวมของกรุ๊ฟดีเลย์ แล้วทำซ้ำไปเรื่อย ๆ จนกระทั่งได้ค่ากรุ๊ฟดีเลย์คงที่

เน็ทเวิร์กฟังก์ชันที่จะนำมาใช้ จะต้องเป็นวงจรที่ยอมให้สัญญาณผ่านได้หมดทุกความถี่ ดังนั้นจึงใช้วงจร ALL-PASS NETWORK มาทำการแก้ค่ากรุ๊ฟดีเลย์

บทที่2

ออล - พาส ทรานส์เฟอร์ ฟังก์ชัน

(ALL - PASS TRANSFER FUNCTION)

การออกแบบดีเลย์อิควอไลเซอร์ กำหนดให้ทรานส์เฟอร์ฟังก์ชันของวงจร ALL - PASS เป็น

$$H(S) = \frac{p(-s)}{P(s)} \qquad (1)$$

โดยที่ P(S) เป็นเฮอร์วิทซ์โพลิโนเมียล(HURWITZ POLYNOMIAL) คือ โพลิโนเมียลที่อยู่ในรูปความถี่เชิงซ้อน ($s = \sigma + j\omega$) สัมประสิทธิ์ค่าจริงจะมีอยู่ในครึ่งซ้ายของระนาบ S-PLANE เท่านั้น เมื่อเขียนให้อยู่ในแกนของความถี่ (REAL FREQUENCY AXIS) จะได้

$$H(j\omega) = Me^{j\theta} = \frac{P(-j\omega)}{P(j\omega)} \qquad (2)$$

เมื่อ M และ θ เป็นขนาดและเฟสตามลำดับ
โดยที่

$$|P(-j\omega)| = |P(j\omega)|$$

และ

$$\arg P(-j\omega) = -\arg P(j\omega)$$

ดังนั้น จากสมการ (2) จะได้ M = 1 และ $\theta = -2\arg P(j\omega)$

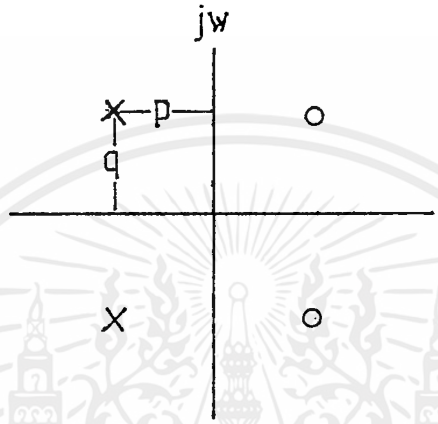
โพลิโนเมียล P(S) สามารถเขียนในรูปควอดราติก แฟคเตอร์ (QUADRATIC FACTOR) ได้คือ

$$s^2 + as + b$$

ดังนั้น สมการ (2) จะเขียนได้ใหม่เป็น

$$H(s) = \frac{s^2 - as + b}{s^2 + as + b} \qquad (3)$$

ทรานส์ฟังก์ชัน H(S) ในสมการ (3) ซึ่งอยู่ในรูปความถี่เชิงซ้อนสามารถเขียนตำแหน่งของโพลและซีโร่ใน S -PLANE ได้ ดังแสดงในรูปที่ 1



รูปที่ 1 แสดงตำแหน่งโพลและซีโร่ของ ALL-PASS TRANSFER FUNCTION

จากทรานส์เฟอร์ฟังก์ชันในสมการ (3) สามารถเขียนได้ใหม่เป็น

$$H(s) = \frac{s^2 - 2ps + p^2 + q^2}{s^2 + 2ps + p^2 + q^2} = \frac{(s-p-jq)}{(s+p-jq)} \bullet \frac{(s-p+jq)}{(s+p+jq)} \qquad (4)$$

ค่าของเฟส จะได้

$$\theta = -2\left[\ \arctan\frac{\omega - q}{p} + \arctan\frac{\omega + q}{p}\ \right] \qquad (5)$$

จากสมการ (5) โดยการดิฟเฟอเรนทิเอทเทียบกับ ω จะได้สมการของกรุ๊ฟดีเลย์คือ

$$Tg = \frac{d\theta}{d\omega} = \frac{2p}{p^2 + (\omega - q)^2} + \frac{2p}{p^2 + (\omega + q)^2} \qquad (6)$$

จากสมการ (6) จะเห็นได้ว่า ค่ากรุ๊ฟดีเลย์ขึ้นอยู่กับระยะห่างของโพลหรือซีโร่ใน S-PLANE คือตัวแปร p และ q ดังนั้นการประมาณเน็ทเวิร์กฟังก์ชันเพื่อแก้ไขความผิดเพี้ยนทางเวลานั้นก็คือ *การบังคับหาตำแหน่งของโพลหรือซีโร่ที่ดีที่สุดในวงจรดีเลย์อิควอไลเซอร์*

อนุพันธ์ย่อย (PARTIAL DERIVATIVES) ของ T_g เมื่อเทียบกับ p และ q จะได้ดังนี้

$$\frac{\partial tg}{\partial p} = 2\left[\frac{(\omega - q)^2 - p^2}{(p^2 + (\omega - q)^2)^2} + \frac{(\omega + q)^2 - p^2}{(p^2 + (\omega + q)^2)^2}\right] \qquad (7)$$

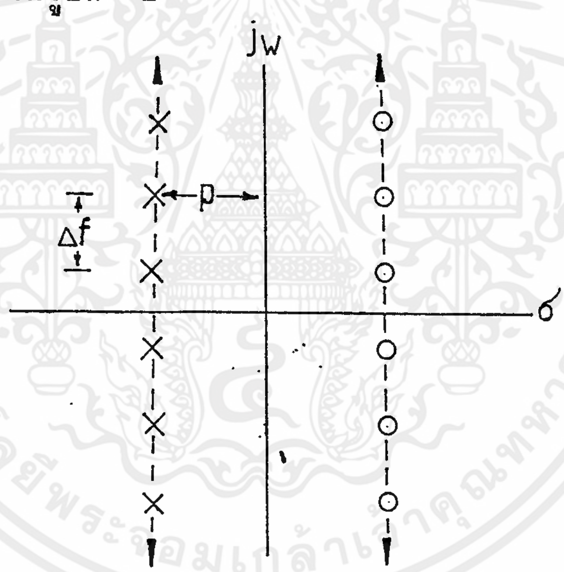
$$\frac{\partial tg}{\partial q} = 4p\left[\frac{\omega - q}{(p^2 + (\omega - q)^2)^2} + \frac{\omega + q}{(p^2 + (\omega + q)^2)^2}\right] \qquad (8)$$

ซึ่งเราจะนำสมการ (7),(8) นี้ไปใช้ต่อไป

บทที่ 3
การประมาณค่าเริ่มต้น
(FIRST APPROXIMATIONS)

ดังที่ได้กล่าวไว้ในบทนำว่า โปรแกรมต้องการค่าเริ่มต้น เพื่อใช้ในการอิเทอเรท ซึ่งวิธีการประมาณค่าเริมต้นนั้นมีหลายวีธี เช่น Template method แต่โดยการใช้ " กฎกำปั้นทุบดิน (rule of thumb)" ก็สามารถจะได้ค่าเริ่มต้นที่ใช้ได้แล้ว ซึ่งในปริญญานิพนธ์นี้ก็ใช้วิธีนี้

การประมาณค่าเริ่มต้นของโพลและซีโร่ของวงจร ALL-PASS นั้นจะใช้วิธีการกำหนดค่าของโพลและซีโร่กระจายขนานไปกับแกน Imajinary ใน S-PLANE ดังแสดงในรูปที่ 2



รูปที่ 2 แสดงการกำหนดค่าเริ่มต้นของโพลและซีโร่
ค่ากรุ๊พดีเลย์ของรูปนี้ จะมีค่าเฉลี่ยคงที่และมีความราบเรียบตลอด

ช่วงระยะห่างระหว่างโพลและซีโร่นั้น สามารถกำหนดได้จากช่วงแบนด์วิธที่ต้องการกับจำนวนเซคชั่นของวงจร ALL-PASS อันดับ 2 คือ

$$\Delta f = \frac{\text{bandwidth of freqency}}{\text{number of section} - 0.5} \tag{9}$$

เราสามารถแสดงให้เห็นความสัมพันธ์ระหว่าง p,Δf ผลรวมของค่าคลาดเคลื่อนของกรุ๊ปดีเลย์ (ε) ได้

$$p = \frac{\Delta f}{2\pi} \ln \frac{4}{\Delta f \varepsilon} \qquad (10)$$

เมื่อ p และ Δf มีหน่วยเป็น c/s และ ε มีหน่วยเป็น sec ค่า q จะหาได้จากค่า Δf ซึ่งพิจารณาจากรูปที่ 2 จะได้

$$q_i = \Delta f \left[i - \tfrac{1}{2} \right] \qquad (11)$$

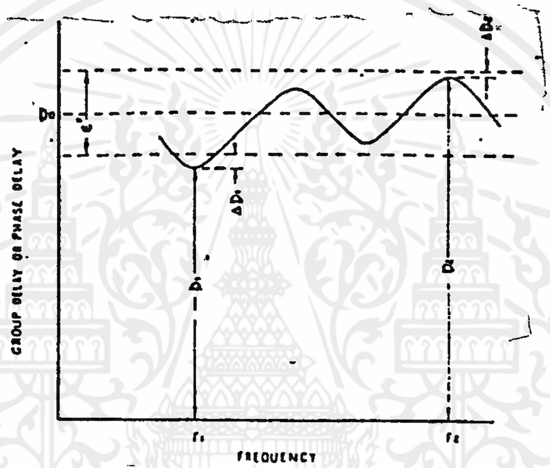
โดยที่ i เป็นลำดับที่ของวงจร ALL-PASS แต่ละเซคชัน

จากสมการ (9),(10),(11) จะได้ค่าเริ่มต้น เพื่อใช้ในการอิเทอเรทหาค่าตอบต่อไป ซึ่งถ้าค่าเริ่มต้นนี้มีค่าใกล้เคียงกับค่าโพลหรือซีโร่ที่ต้องการ ก็จะทำให้จำนวนครั้งในการอิเทอเรชันน้อยลง

บทที่ 4

ทฤษฎีในการออกแบบ

เพื่อที่จะให้ได้กรุ๊ฟดีเลย์คงที่ จะต้องทำให้ทางเดินกรุ๊ปดีเลย์ของวงจร all-pass มีผลต่อวงจรที่ต้องการจะแก้ นั่นคือ กำหนดให้ D เป็นผลรวมขอลกรุ๊ฟดีเลย์ทั้งหมด คือของวงจรดีเลย์อิควอไลเซอร์ และวงจรที่ต้องการแก้ไขความผิดเพี้ยนทางเวลา และมีค่าเฉลี่ยคงที่เป็น D_o ด้วยจุดสูงสุดและต่ำสุดเปลี่ยนแปลงจากค่านี้เท่ากับ $\pm \varepsilon/2$ เมื่อ ε คือ ผลรวมของค่าคลาดเคลื่อนของกรุ๊ฟดีเลย์ (delay variation) ตัวอย่างของลักษณะกรุ๊ปดีเลย์ที่ยังไม่คงที่เป็นดังรูปที่ 3



รูปที่ 3

ค่าผลรวมของกรุ๊ฟดีเลย์ (D) มีค่าต่ำสุดเท่ากับ D_1 ที่ความถี่ f_1 และค่าสูงสุดเท่ากับ D_2 ที่ความถี่ f_2 ดังนั้น เพื่อจะให้ได้ค่ากรุ๊ปดีเลย์ที่มีค่าคลาดเคลื่อนมากที่สุด $\pm \varepsilon/2$ จากค่าเฉลี่ย D_o เราต้องเพิ่มค่า D ที่ D_1 เท่ากับ และที่ f_2 เท่ากับ ΔD_2 นั่นคือ

ค่าต่ำสุด $$\Delta D_1 + D_1 + \varepsilon/2 = D_0 \qquad (12)$$

ค่าสูงสุด $$\Delta D_1 + D_1 - \varepsilon/2 = D_0 \qquad (13)$$

ที่ความถี่ใด ๆ สมมุติเป็น j จะได้

ค่าต่ำสุด $\Delta D_j + D_j + \varepsilon/2 = D_0$ (14)

ค่าสูงสุด $\Delta D_j + D_j - \varepsilon/2 = D_0$ (15)

ผลรวมของกรุ๊พดีเลย์ทั้งหมด D จะขึ้นอยู่กับโคออดิเนทของ (p,q) ในแต่ละส่วนของทรานส์เฟอร์ฟังก์ชัน ALL-PASS อันดับสอง การเปลี่ยนแปลงค่าใด ๆ ของ D ที่ช่วงความถี่ต่าง ๆ ขึ้นอยู่กับการเปลี่ยนค่า $(\Delta p,\Delta q)$ ในโคออดิเนทที่เหมาะสม ตัวอย่างที่สมมุติให้ (p_0,q_0) เป็นรากเริ่มต้นของสมการ

$$\Delta D(p,q) = 0$$

ที่เป็นศูนย์ เพราะเราต้องการไม่ให้มีค่าคลาดเคลื่อนเลย จึงเทียบกับศูนย์ ถ้า $(p_0+\Delta p , q_0+\Delta q)$ เป็นรากของระบบสมการแล้วจะได้

$$\Delta D (p_0+\Delta p , q_0+\Delta q) = 0$$

ฟังก์ชัน ΔD สามารถหาอนุพันธ์ได้เมื่อกระจายออกเป็นอนุกรมโดยใช้เป็นอนุกรมเทย์เลอร์หลายตัวแปรจะได้

$$\Delta D_0 + \Delta p \frac{\partial \Delta D}{\partial p_0} + \Delta q \frac{\partial \Delta D}{\partial q_0} + \dots\dots\dots = 0$$

เมื่อไม่คิดเทอมที่มีอนุพันธ์อันดับสองและสูงกว่าแล้วจะได้ระบบสมการเชิงเส้น

$$\Delta p \frac{\partial \Delta D}{\partial p_0} + \Delta q \frac{\partial \Delta D}{\partial q_0} = -\Delta D_0$$

ดังนั้นรากที่ประมาณได้ใหม่คือ

$$p_1 = p_0 + \Delta p$$

$$q_1 = q_0 + \Delta q$$

ถ้าต้องการหารากที่ดียิ่งขึ้น ต้องดำเนินการกระทำซ้ำหลายหน จนกระทั่งถูกต้อง ก็คือทำการเปลี่ยนค่า $\Delta p,\Delta q$ ไป แล้วนำมาแทนค่าได้ p_1, q_1 ใหม่และทำจน ΔD เป็นศูนย์

จากหลักการนี้ เราสามารถนำเอาวิธีนิวตัน-ราฟสัน มาใช้แก้สมการ 2n ตัวแปร m สมการของวงจรดีเลย์อิควอไลเซอร์ หลักการของวิธีนิวตัน-ราฟสันก็คือ ใช้เทย์เลอร์กระจายแต่ละสมการออกเป็นอนุกรม เมื่อตัดเทอมที่มีอันดับสูงกว่าหนึ่งออกแล้ว จะเป็นการหารากของสมการเชิงเส้นนั่นเอง ก็คือ ที่ความถี่ใด ๆ ผลต่างของกรุ๊พดีเลย์ ΔD_j จะสัมพันธ์กับการเปลี่ยนแปลงของ $\Delta p_i , \Delta q_i$ ดังสมการ

$$\begin{aligned} \Delta D_i = {} & \left[\frac{\partial D}{\partial p_1}\right]_{f_i} \Delta p_1 + \dots\dots + \left[\frac{\partial D}{\partial p_i}\right]_{f_i} \Delta p_i + \dots\dots \\ & + \left[\frac{\partial D}{\partial p_n}\right]_{f_i} \Delta p_n + \dots\dots + \left[\frac{\partial D}{\partial q_1}\right]_{f_i} \Delta q_1 + \dots\dots \\ & + \left[\frac{\partial D}{\partial q_i}\right]_{f_i} \Delta q_i + \dots\dots + \left[\frac{\partial D}{\partial q_n}\right]_{f_i} \Delta q_n \end{aligned} \tag{16}$$

จะเห็นว่ามีตัวแปร 2n ตัวแปรคือ $p_1 \dots p_n , q_1 \dots q_n$ ดังนั้นต้องมีจำนวนสมการถึง $m = 2n$ สมการจึงจะแก้สมการได้ ดังนั้นสมการ (16) สามารถเขียนได้ใหม่ในรูปเมทริกส์ คือ

$$\begin{vmatrix} \frac{\partial D_1}{\partial p_1} + \frac{\partial D_1}{\partial p_i} + \dots + \frac{\partial D_1}{\partial p_n} + \frac{\partial D_1}{\partial q_1} + \frac{\partial D_1}{\partial q_i} + \dots + \frac{\partial D_1}{\partial q_n} \\ \frac{\partial D_j}{\partial p_1} + \frac{\partial D_j}{\partial p_i} + \dots + \frac{\partial D_j}{\partial p_n} + \frac{\partial D_j}{\partial q_1} + \frac{\partial D_j}{\partial q_i} + \dots + \frac{\partial D_j}{\partial q_n} \\ \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \\ \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \quad \cdot \\ \frac{\partial D_m}{\partial p_1} + \frac{\partial D_m}{\partial p_i} + \dots + \frac{\partial D_m}{\partial p_n} + \frac{\partial D_m}{\partial q_1} + \frac{\partial D_m}{\partial q_i} + \dots + \frac{\partial D_m}{\partial q_n} \end{vmatrix} \begin{vmatrix} \Delta p_1 \\ \Delta p_i \\ \cdot \\ \cdot \\ \Delta p_n \\ \Delta q_1 \\ \Delta q_i \\ \cdot \\ \cdot \\ \Delta q_n \end{vmatrix} = \begin{vmatrix} \Delta D_1 \\ \Delta D_j \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \Delta D_m \end{vmatrix} \tag{17}$$

เมื่อ n คือจำนวนคู่ (p,q) และ $\frac{\partial D_j}{p_i}, \frac{\partial D_j}{q_i}$ เป็นการหาอนุพันธ์ของ D เทียบกับ pi และ qi ที่ความถี่ j ซึ่งสามารถหาได้คือ

$$\frac{\partial D_j}{\partial p_i} = 2\left[\frac{(\omega_j - q_i)^2 - p_i^2}{[p_i^2 + (\omega_j - q_i)^2]^2} + \frac{(\omega_j + q_i)^2 - p_i^2}{[p_i^2 + (\omega_j + q_i)^2]^2}\right] \tag{18}$$

$$\frac{\partial D_j}{\partial p_i} = 4p_i\left[\frac{\omega_j - q_i}{[p_i^2 + (\omega_j - q_i)^2]^2} - \frac{\omega_j + q_i}{[p_i^2 + (\omega_j + q_i)^2]^2}\right] \tag{19}$$

สมการ (17) เป็นสมการเชิงเส้นที่จะหาค่าของ $\Delta p_i, \Delta q_i$ เพื่อปรับปรุงค่าของ pi และ qi ใหม่ แล้วนำไปแทนค่าเพื่อหาค่าผลรวมของกรุ๊ฟดีเลย์ และจะได้ค่า ΔD_jน้อยลงเรื่อยๆ โดยจะกระทำซ้ำๆ กันจนกระทั่ง ΔD_j มีค่าเป็นศูนย์ นั่นคือ

ค่าต่ำสุด $D_j + \varepsilon/2 = D_0$ (20)

ค่าสูงสุด $D_j - \varepsilon/2 = D_0$ (21)

ในทางปฏิบัติ กระบวนการจะกระทำจนกว่าจะได้

ค่าต่ำสุด $|D_j + \varepsilon/2 - D_0| \leq \text{Tol}$ (22)

ค่าสูงสุด $|D_j - \varepsilon/2 - D_0| \leq \text{Tol}$ (23)

เมื่อ Tol เป็นค่าตรวจสอบการลู่เข้าที่ยอมให้ได้ (Tolerence)

จากเซ็ตของสมการที่ผ่านมา จะเห็นได้ว่ามีจำนวนของตัวแปรทั้งหมดเท่ากับ 2n + 3 เมื่อ n เป็นจำนวนเน็ทเวิร์กของวงจร ALL-PASS อันดับสอง ซึ่งมีตัวแปรคือ p_i และ q_i (ก็คือ 2n) ตัวแปรที่เหลือคือ ε, D_0 และช่วงแบนด์วิธของความถี่ที่ใช้ ซึ่งพิจารณาเป็นอิสระจากตัวแปรตัวอื่น เมื่อเรากำหนดค่าแบนด์วิธ ก็จะเหลือตัวแปรเท่ากับ 2n+2 และถ้าไม่กำหนดแบนด์วิธ

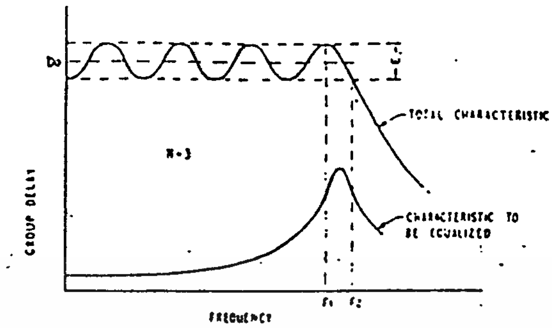
แต่กำหนด ε ก็จะเหลือตัวแปร 2n+2 เหมือนกัน และเนื่องจากการแก้ค่ากรุ๊ฟดีเลย์ จะทำให้ได้ค่ากรุ๊ฟดีเลย์ที่มีลักษณะเป็นริปเปิลที่เท่า ๆ กัน คือมีค่าเฉลี่ยที่ D_o ยอดของริปเปิลอยู่ที่ $\pm \varepsilon/2$ ซึ่งจะมีจำนวนของริปเปิลเท่ากับจำนวนตัวแปรคือ 2n+2 ด้วย ซึ่งจะพิจารณาได้จากตัวอย่างอิควอไลซ์วงจร LOW-HIGH PASS ต่อไปนี้

การอิควอไลซ์ LOW - HIGH PASS

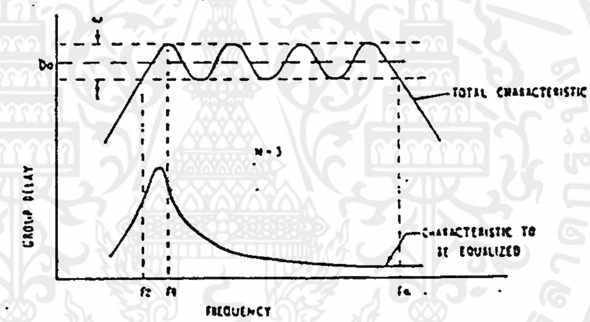
รูปที่ 4(a) คือผลที่ได้จากการอิควอไลซ์ค่ากรุ๊ฟดีเลย์ เมื่อ n = 4 จะเห็นได้ว่ามีจุดของริปเปิลจำนวน 9 จุด เพราะมีตัวแปร 2n+2= 10 ตัว

รูปที่ 4(b) และ 4(c) เป็นผลจากการแก้ไขค่ากรุ๊ฟดีเลย์ของวงจร low-pass และ high-pass ตามลำดับ จากรูป เราต้องการที่จะทำการอิควอไลซ์ถึงแค่ความถี่ f1 เท่านั้น ดังนั้น จะมีริปเปิล 8 จุด เมื่อใช้ n = 3

รูป 4(a) การจำลองค่ากรุ๊ฟดีเลย์คงที่



รูป 4(b) การอิควอไลซ์วงจร Low-Pass



รูป 4(c) การอิควอลไลซ์วงจร High-Pass

บทที่ 5
ขั้นตอนการออกแบบ
(The Approximation Procedure)

การออกแบบโปรแกรมสามารถแบ่งได้เป็น 3 ส่วนใหญ่ ๆ คือ

1. หาค่าสูงสุดและต่ำสุดของผลรวมกรุ๊ฟดีเลย์
2. ทดสอบว่าคงที่หรือยัง โดยเทียบกับค่าในสมการ (22), (23)
3. คำนวนค่าการเปลี่ยนแปลงของ $(\Delta p, \Delta q)$

ค่าสูงสุดและต่ำสุด ของผลรวมกรุ๊ฟดีเลย์ จะได้จากการคำนวณค่ากรุ๊ฟดีเลย์ โดยการตัดช่วงปิดมาช่วงหนึ่งในย่านความถี่ที่เราต้องการ ต่อจากนั้นก็คำนวณหาค่าความถี่ (ซึ่งมีหลายค่า) โดยการดิฟเฟอเรนทิเอทว่ามีค่าเป็นศูนย์หรือมีการเปลี่ยนเครื่องหมาย ก็จะได้ค่าสูงสุดและต่ำสุด

ในการทดสอบความราบเรียบของกรุ๊ฟดีเลย์ จะใช้สมการ (22), (23) คือ

ค่าต่ำสุด $|D_j + \varepsilon/2 - D_0| \leq \text{Tol}$

ค่าสูงสุด $|D_j - \varepsilon/2 - D_0| \leq \text{Tol}$

โดยทั่วไป ค่า ε และ D_0 จะยังไม่ทราบในตอนเริ่มต้นการอิเทอเรชั่น ดังนั้นจะข้ามขั้นตอนการทดสอบค่าต่ำสุด, สูงสุด ในการอิเทอเรทครั้งแรก ค่า Tol สามารถที่จะกำหนดลงไปได้ การเปลี่ยนแปลงของ $(\Delta p, \Delta q)$ จะได้จากการแก้สมการเชิงเส้น (17)

สามารถเขียนโฟล์วชาร์ทได้ดังรูปที่ 5

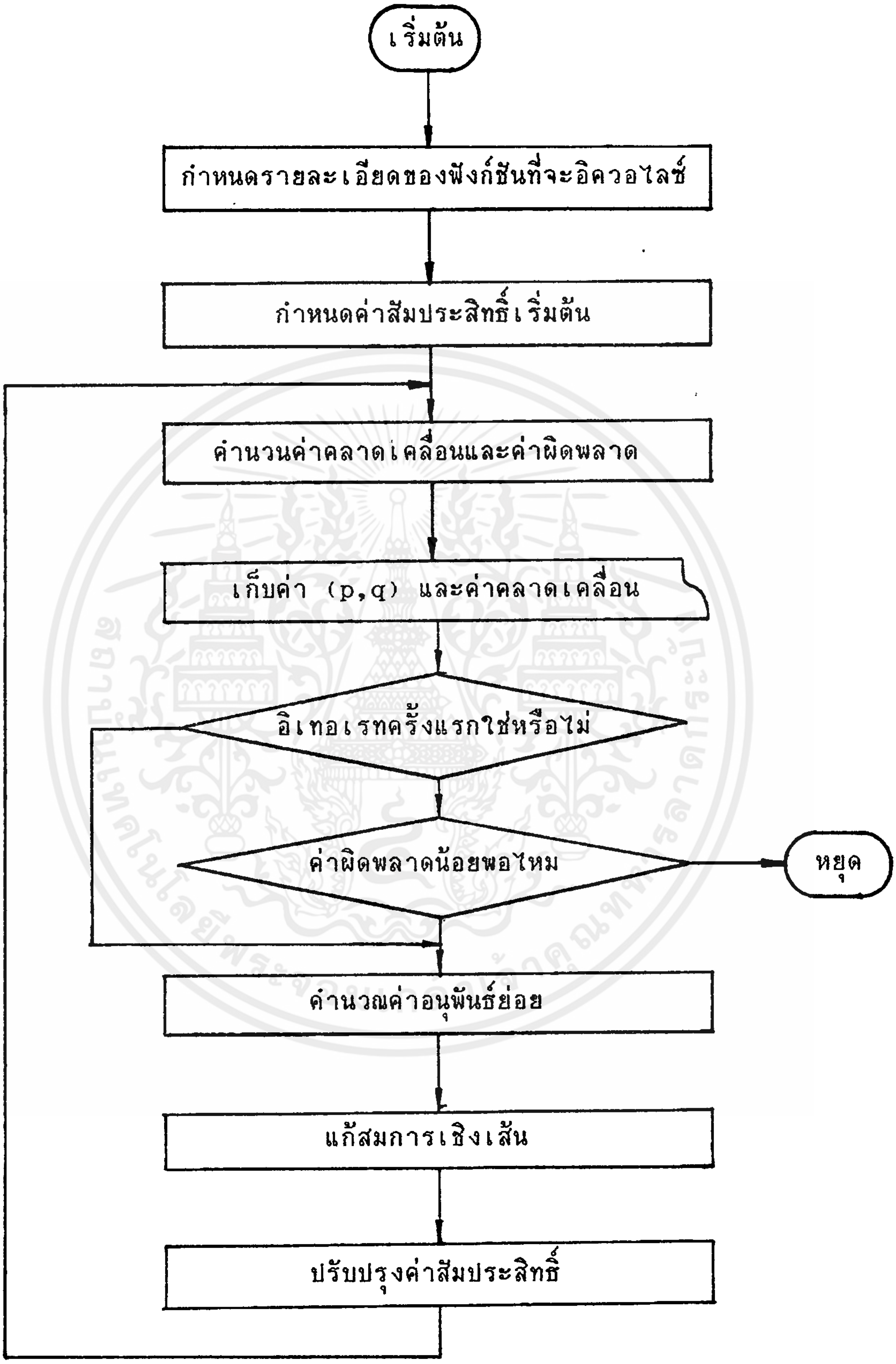
แต่ละบล็อกจะอธิบายได้ดังนี้คือ

- กำหนดรายละเอียดของฟังก์ชันที่จะอิควอไลซ์

เป็นการกำหนดสมการกรุ๊ฟดีเลย์ของฟังก์ชันที่ต้องการจะแก้ไข โดยผู้ใช้ต้องคำนวณมาเอง และกำหนดค่าเฉลี่ยของกรุ๊ฟดีเลย์ (D_0) ค่าคลาดเคลื่อนสูงสุดของกรุ๊ฟดีเลย์ (ε) และค่าตรวจสอบการลู่เข้าที่ยอมให้มีได้ (Tol)

- กำหนดค่าสัมประสิทธิ์เริ่มต้น

ก็คือค่า p,q เริ่มต้น จะได้จากสมการ (9), (10), (11)



รูปที่ 5 โฟล์วชาร์ทของโปรแกรมแก้ไขค่ากรุ๊พดีเลย์

ซึ่งจะพิจารณาได้จากตัวอย่างการใช้งานต่อไป

- คำนวณค่าคลาดเคลื่อนและค่าผิดพลาด

ค่าคลาดเคลื่อนคือค่า ΔD ซึ่งจะนำไปใช้ในการแก้สมการเชิงเส้น ส่วนค่าผิดพลาดคือ ค่าสูงสุดและต่ำสุดของกรุ๊ฟดีเลย์ (max, min) โดยใช้ฟังก์ชัน DifF, EvalvateFirstDeriv, และโพรซีเดอร์ MinMax กับฟังก์ชัน TestforRoot

- เก็บค่า p,q ค่าคลาดเคลื่อน

เป็นการเก็บตำแหน่งของโพลและซีโร่ ซึ่งจะเป็นค่าตอบของโปรแกรมนี้นั่นเอง

- อิเทอเรทครั้งแรกใช่หรือไม่

จะเช็คว่าเป็นการอิเทอเรทครั้งแรกหรือไม่ ถ้าใช่จะข้ามการทดสอบค่าผิดพลาด ดังที่กล่าวมาแล้วตอนต้น

- ค่าผิดพลาดน้อยพอหรือไม่

เป็นการเช็คค่าสูงสุดและต่ำสุดของกรุ๊ฟดีเลย์ว่าอยู่ในช่วงที่ยอมให้มีได้ (Tol) หรือยัง คือเทียบกับสมการ (22), (23) ถ้าได้แล้วก็จะจบโปรแกรม

- คำนวณอนุพันธ์ย่อย

เป็นการคำนวณค่าอนุพันธ์ตามสมการ (18),(19) โดยนำค่า w,a,b มาคำนวณ

- แก้สมการเชิงเส้น

เป็นการคำนวณเพื่อหาค่า $\Delta p_i, \Delta q_i$ ตามสมการ (17) ซึ่งต้องนำค่า ΔD และค่าอนุพันธ์ย่อยมาใช้ โดยใช้โพรซีเดอร์ Partial-Pivoting, EROmulAdd

- ปรับปรุงค่าสัมประสิทธิ์

เป็นการนำค่า p,q จากบล็อก "แก้สมการเชิงเส้น" มาเพิ่มค่าให้กับ p,q เดิม

การทำงานของโปรแกรมเพื่อหาค่าผิดพลาด (max,min) จะมี 2 ช่วงคือ ในแรกกระทำการคำนวณค่าสัมประสิทธิ์ของฟังก์ชันจากค่าสัมประสิทธิ์เริ่มต้นเพื่อให้ได้ค่าเฉลี่ยของกรุ๊ฟดีเลย์ใกล้เคียงกับที่กำหนดไว้ หลังจากนั้นจึงจะคำนวณหาจุดสูงสุดและต่ำสุดของผลรวมกรุ๊ฟดีเลย์ทั้งหมด และปรับปรุงค่าสัมประสิทธิ์ของฟังก์ชันใหม่ เพื่อให้ได้ค่าคลาดเคลื่อนน้อยที่สุด

รายละเอียดของโปรแกรมแสดงไว้ในภาคผนวก โดยที่

- function EquF

คิดค่ากรุ๊ฟดีเลย์ที่ความถี่ต่าง ๆ โดยคิดค่าของวงจรที่จะแก้ แล้วนำมารวมกับค่าดีเลย์ของวงจร All-pass ทุกชุดที่ใช้ ในตอนแรกรับค่าความถี่เป็น ω เพื่อคำนวณตามสมการกรุ๊ฟดีเลย์ได้ แล้วจึงนำค่านี้มาหารด้วย 2π ก็จะได้ค่าความถี่เป็น f (จาก $\omega = 2\pi f$)

- function DifF

เป็นฟังก์ชันที่ทำการดิฟเฟอเรนทิเอท เพื่อใช้ในการหาค่าสูงสุดและต่ำสุดโดยฟังก์ชันจะให้ค่าออกมาเป็นตัวเลข ถ้าค่าที่ป้อนให้เป็นค่าสูงสุดหรือต่ำสุดก็จะได้ค่าเป็นศูนย์

- Procedure MinMax

เป็นโพรซีเดอร์ที่ใช้หาค่ารากของกรุ๊ฟดีเลย์ โดยใช้วิธีแบ่งครึ่งช่วง (bisection) ซึ่งกำหนดจำนวนครั้งในการแบ่งไว้สูงสุด 500 ครั้ง และมีตัวแปร Error เป็นตัวเช็คค่าราก

- Procedure Partial-Pivoting

เป็นโพรซีเดอร์ที่ใช้หาผลเฉลยของระบบสมการเชิงเส้น [สมการ (17)] โดยใช้วิธีกำจัดของเกาส์ (Gaussian elemination) ค่าที่ได้จะเป็นค่าสัมประสิทธิ์เพื่อนำไปปรับปรุงต่อไป

- Procedure EROmulAdd

เป็นโพรซีเดอร์ที่ทำหน้าที่ดำเนินการเปลี่ยนแถวเชิงธาตุมูล (elementary row operation) เพื่อใช้ในวิธีกำจัดของเกาส์

บทที่ 6

การใช้งานโปรแกรมออกแบบกรุ๊ฟดีเลย์อิควอไลเซอร์

1. ใส่ค่าสมการกรุ๊ฟดีเลย์ของวงจรที่ต้องการจะแก้ความผิดเพี้ยนทางเวลา โดยทดลองใส่ค่าบวกหรือลบ แล้วดูผลว่าได้รูปกราฟของกรุ๊ฟดีเลย์ เป็นคอมพลีเมนท์กับวงจรหรือไม่ โดยใส่ในฟังก์ชัน EquF

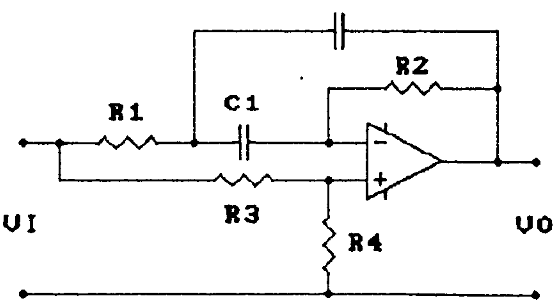
2. ใส่ค่าเฉลี่ยของกรุ๊ฟดีเลย์ (Do) ซึ่งควรจะให้มีค่ามากกว่าค่าเฉลี่ยกรุ๊ฟดีเลย์ของวงจรที่ต้องการจะแก้ ใส่ค่าคลาดเคลื่อน และค่าตรวจสอบการลู่เข้าที่ยอมให้มีได้ในเมนโปรแกรม

3. ใส่ค่าสัมประสิทธิ์เริ่มต้นในเมนโปรแกรม โดยคำนวณจากสมการที่ (9), (10) และ (11) พร้อมกันนี้ต้องใส่จำนวนของตัวแปร p, q ในตัวแปรชื่อ Dimen ด้วย

4. โปรแกรมจะให้คำตอบเป็นค่าระยะห่างของโพลหรือซีโร่ กับแกน Imajinary (p) และแกน Real (q) ซึ่งจะนำค่านี้ไปหาค่าอุปกรณ์ของวงจรต่อไป

5. วงจรดีเลย์อิควอไลเซอร์ เป็นวงจร All-pass อันดับที่ 2 ซึ่งสามารถสร้างขึ้นได้โดยใช้วงจรแบบพาสซีฟหรือแบบแอกทีฟ ในการทดลอง จะใช้วงจรแบบแอกทีฟอาร์ซี ดังในรูปที่ 6 ซึ่งสามารถเขียนเป็นโวลท์เตอทรานส์เฟอร์ฟังก์ชันได้คือ

$$\frac{V_o}{V_i} = \frac{R_4}{R_3 + R_4}\,\frac{S^2 - \frac{2}{R_2 C} S + \frac{1}{R_1 R_2 C^2}}{S^2 + \frac{2}{R_2 C} S + \frac{1}{R_1 R_2 C^2}} \qquad (24)$$



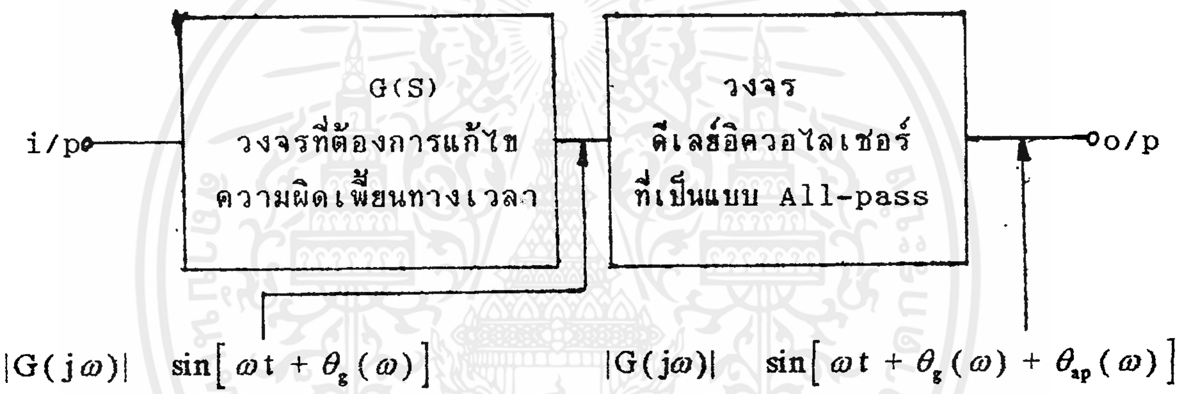
รูปที่ 6 วงจร All-pass อันดับ 2

จากค่าโพลและซีโร่ที่ได้ในข้อ 4 เมื่อแทนค่าลงในสมการที่ (4) คือ

$$H(S) = \frac{s^2 - 2ps + p^2 + q^2}{s^2 + 2ps + p^2 + q^2}$$

และเปรียบเทียบสัมประสิทธิ์ที่ได้กับสมการที่ (24) จะได้ค่าอุปกรณ์ R, C แต่ละตัวของวงจร All-pass ซึ่งอาจจะต้องทำการสเกลทางขนาดและความถี่ เพื่อให้ได้ค่าอุปกรณ์ที่มีขาย

6. การแก้ไขความผิดเพี้ยนทางเวลากระทำได้โดยการนำเอาวงจรดีเลย์อิควอไลเซอร์แบบ All-pass นี้ มาต่อคาสเคด (cascade) กับวงจรที่ต้องการแก้ไขความผิดเพี้ยนทางเวลาตามในรูปที่ 7

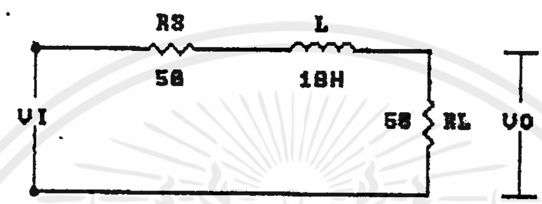


รูปที่ 7 แสดงการนำดีเลย์อิควอไลเซอร์ไปใช้งาน

ตัวอย่างการใช้งานโปรแกรมออกแบบกรุ๊ฟดีเลย์อิควอไลเซอร์

ตัวอย่างที่ 1

วงจร RL low pass filter ดังรูปที่ 8



รูปที่ 8 RL-LPF

มีทรานส์เฟอร์ฟังก์ชั่นคือ

$$\frac{Vo}{Vi} = \frac{RL}{RS+RL+SL}$$

$$= \frac{RL}{RS+RL} \bullet \frac{\omega c}{S+\omega c}$$

เมื่อ $\omega c = \dfrac{RS+RL}{L}$

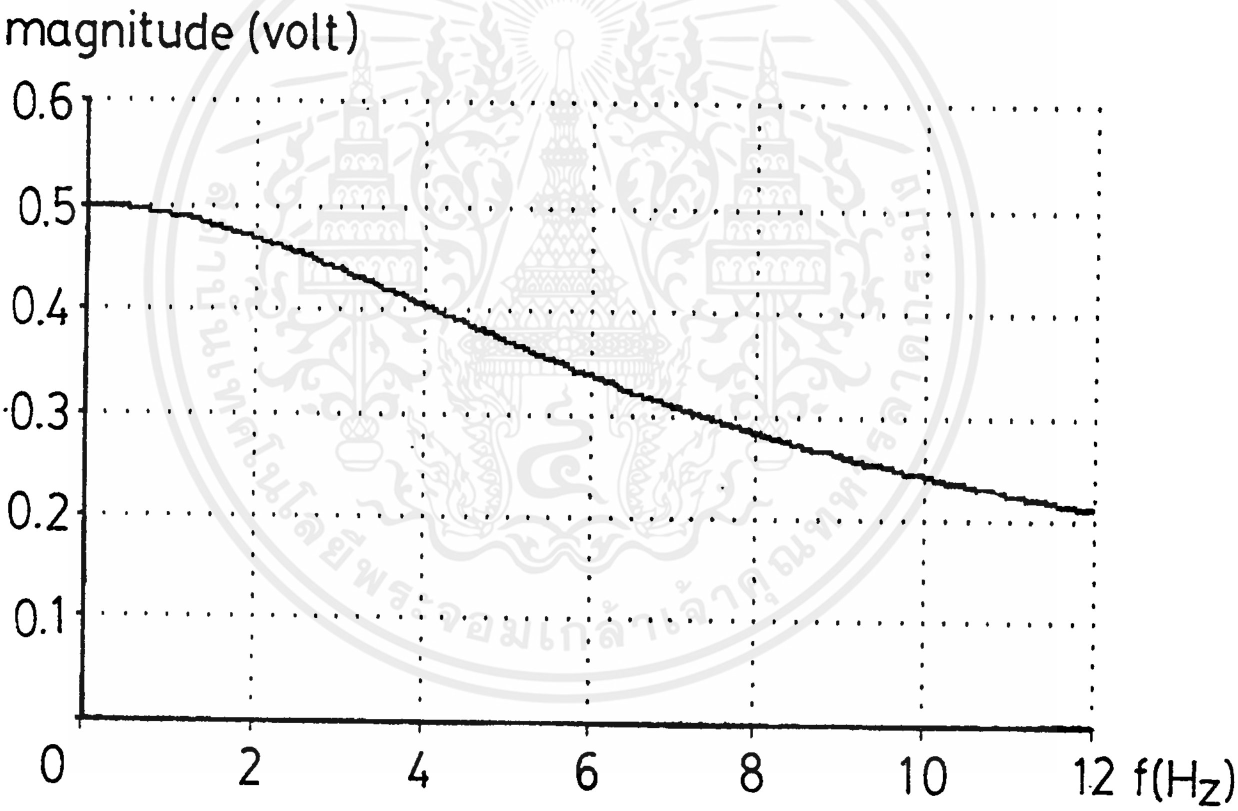
เมื่อแทนค่าจะได้

$$\frac{Vo}{Vi} = \frac{50}{100+j18\,\omega}$$

และจะได้สมการกรุ๊ฟดีเลย์

$$Td = \frac{\omega c}{\omega^2 c + \omega^2} = \frac{5.56}{(5.56)^2 + \omega^2}$$

เมื่อพล็อตผลตอบสนองทางขนาดต่อความถี่ของวงจรออกมา จะมีลักษณะดังรูปที่ 9 จะเห็นว่าวงจรมีช่วงพาสแบนด์ (pass band) ประมาณ 6 H_z ดังนั้น จะทำการอิควอไลซ์ค่ากรุ๊ฟดีเลย์ในช่วง 0-6 H_z เท่านั้น



รูปที่ 9 ผลตอบสนองทางขนาดของวงจร RL-LPF

ซึ่งในช่วงดังกล่าว กรุ๊ฟดีเลย์ยังมีค่าไม่คงที่ คือมีค่าคลาดเคลื่อนประมาณ 96 mS โดยมีค่าลดลงเรื่อยๆ ตั้งแต่ 2 ถึง 6 H_z ดูรูปที่ 10 ประกอบ ดังนั้น จะใช้วงจร All-pass order 2 จำนวน 5 เซคชั่นมาทำการอิควอไลซ์ และจากสมการ (9) จะได้

$$\Delta f = \frac{6}{4.5} = 1.3333$$

ถ้ากำหนดให้ค่าคลาดเคลื่อน (ε) มีค่าประมาณ 5 mS จากสมการ (10) จะได้ระยะห่างของตำแหน่งโพลหรือซีโร่กับแกน real คือ

$$P = \frac{1.3333}{2\pi} = \ln\frac{4}{1.3333 \times 0.005} = 1.3575$$

และจากสมการ (11) จะได้ระยะห่างของตำแหน่งโพลหรือซีโร่ เซคชั่นต่างๆกับแกน imajinary คือ

$$\begin{aligned}
q1 &= (1.3333)(0.5) = 0.6667 \\
q2 &= (1.3333)(1.5) = 2.0000 \\
q3 &= (1.3333)(2.5) = 3.3333 \\
q4 &= (1.3333)(3.5) = 4.6666 \\
q5 &= (1.3333)(4.5) = 6.0000
\end{aligned}$$

ซึ่งค่า (p,q) เหล่านี้ จะนำไปเป็นค่าเริ่มต้นให้แก่โปรแกรมนำไปอิเทอเรท นอกจากนี้ ในโปรแกรมยังต้องกำหนด

-สมการกรุ๊ฟดีเลย์ของวงจรที่ต้องการจะอิควอไลซ์ โดยต้องทดลองใส่ทั้งค่าบวกและลบดู ว่าสมการใดทำให้เกิดลักษณะกรุ๊ฟดีเลย์ที่เป็น
คอมพลีเมนท์ (complement) กับวงจรที่จะแก้ ในที่นี้เป็นบวกและใส่ในฟังก์ชั่น EquF

-ค่าเฉลี่ยของกรุ๊ฟดีเลย์ (Do) ในที่นี้ให้มีค่าเท่ากับ 0.17 S

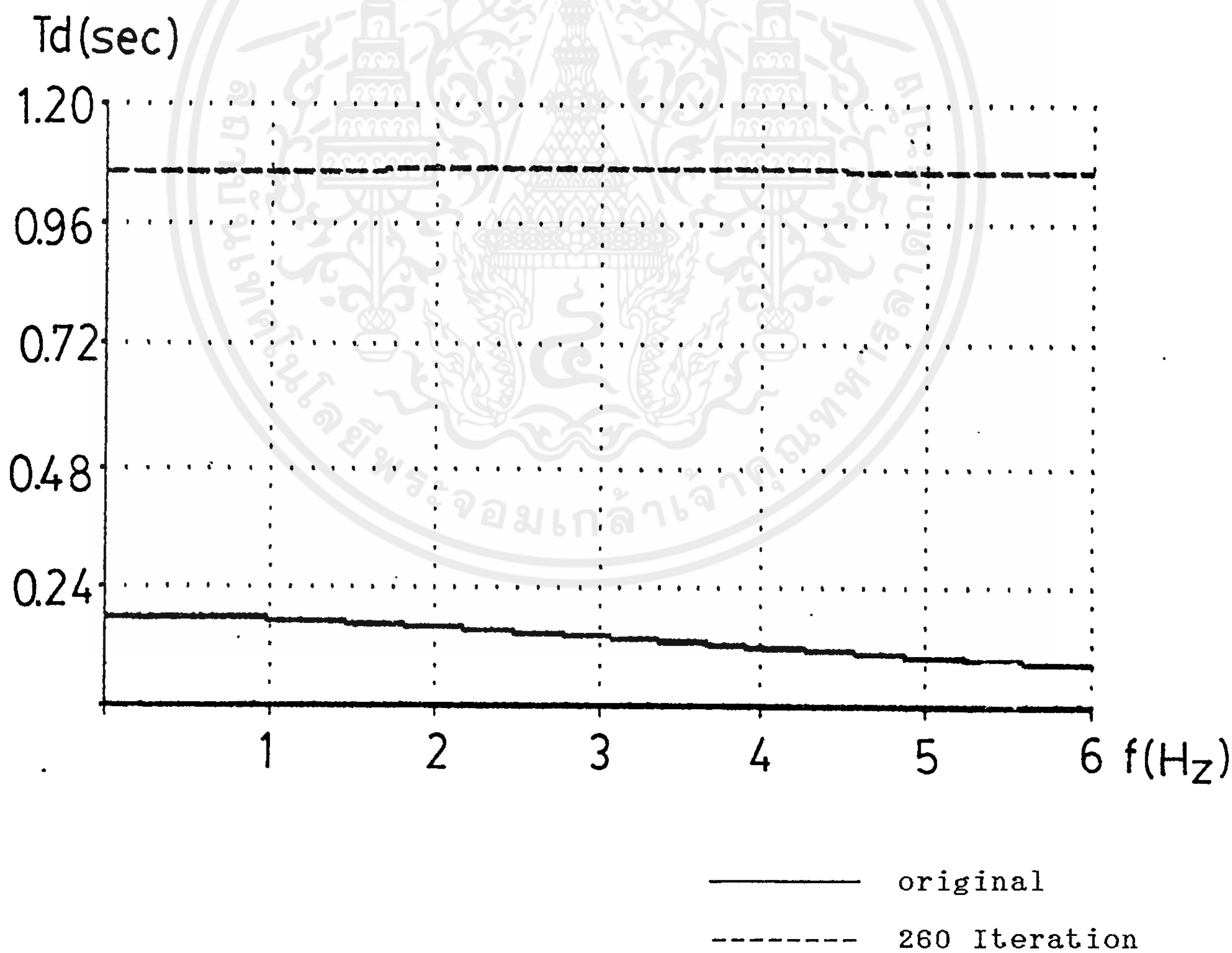
-ค่าคลาดเคลื่อนของกรุ๊ฟดีเลย์ (ในโปรแกรมใช้ตัว e) ในที่นี้ทดลองจาก 0.005 s ลงมาเรื่อยๆ จนได้ 0.0009 s ถ้าค่ายิ่งน้อยก็จะยิ่งได้กรุ๊ฟดีเลย์ที่ราบเรียบยิ่งขึ้น แต่ก็ต้องดูว่าโปรแกรมสามารถทำได้หรือไม่

-จำนวนตัวแปรโคออดิเนท (p,q) (Dimen) ในที่นี้ Dimen = 10

หลังจากการอิเทอเรทโดยคอมพิวเตอร์ ซึ่งจะใช้จำนวนครั้งในการอิเทอเรทช่วงแรก 157 ครั้ง และช่วงหลังรวมเท่ากับ 260 ครั้ง จะได้อิควอดิเนทของ (p,q) ชุดต่างๆ คือ

p1	=	5.72733	q1	=	3.62393
p2	=	5.44326	q2	=	8.64379
p3	=	5.19243	q3	=	13.58603
p4	=	4.72356	q4	=	18.27942
p5	=	3.51310	q5	=	22.59609

เมื่อนำค่าโคออดิเนทเหล่านี้ มาแทนค่าลงในสมการของวงจร All-pass order 2 (สมการ (4)) เซตชั้นต่างๆ แล้วต่อคาสเคด (coscade) กับวงจรที่จะอิควอไลซ์ พล็อทกราฟลักษณะกรุ๊ฟดีเลย์ออกมา จะได้ดังรูปที่ 10



รูปที่ 10 แสดงผลจากการอิควอไลซ์กรุ๊ฟดีเลย์

จะเห็นว่ากรุ๊ฟดีเลย์คงที่ในช่วงแบนด์วิธ 6 Hz และมีความคลาดเคลื่อนประมาณ 5 mS

การสร้างวงจรดีเลย์อิควอไลเซอร์ ทำได้โดยการนำค่าโคออดิเนท(p,q) ที่ได้ ไปแทนค่าลงในสมการของ All-pass order 2 (สมการ (4)) แล้วนำไปเทียบสัมประสิทธิ์กับทรานส์เฟอร์ฟังก์ชั่นของวงจร All-pass แบบแอกทีฟอาร์ซี (สมการ (24)) แล้วทำการสเกลทั้งทางขนาดและความถี่ ซึ่งจะได้ค่าของอุปกรณ์แต่ละตัวในวงจร All-pass order 2 แต่ละชุดคือ

Section 1

R_1 = 198 Ω , R_2 = 277 Ω , R_3 = 2 KΩ , R_4 = 700 Ω

C = 10 μF

Section 2

R_1 = 83 Ω , R_2 = 292 Ω , R_3 = 1.6 KΩ , R_4 = 1.4 KΩ

C = 10 μF

Section 3

R_1 = 39 Ω , R_2 = 306 Ω , R_3 = 1.6 KΩ , R_4 = 6.39 KΩ

C = 10 μF

Section 4

R_1 = 42 Ω , R_2 = 673 Ω , R_3 = 1.6 KΩ , R_4 = 6.39 KΩ

C = 5 μF

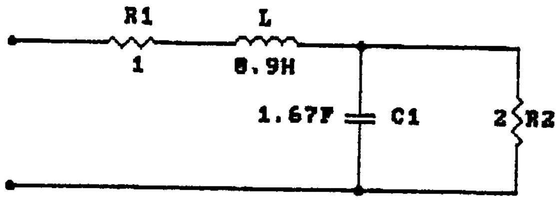
Section 5

R_1 = 35 Ω , R_2 = 1.51 KΩ , R_3 = 1.5 KΩ , R_4 = 15.88 KΩ

C = 3 μF

ตัวอย่างที่ 2

วงจร low pass filter แบบ Butterworth ดังแสดงในรูปที่ 11



รูปที่ 11 วงจร butterworth filter

มีโวลท์เตจทรานส์เฟอร์ฟังก์ชั่นคือ

$$H(S) = \frac{2}{3.0065^2 + 4.245 + 3}$$

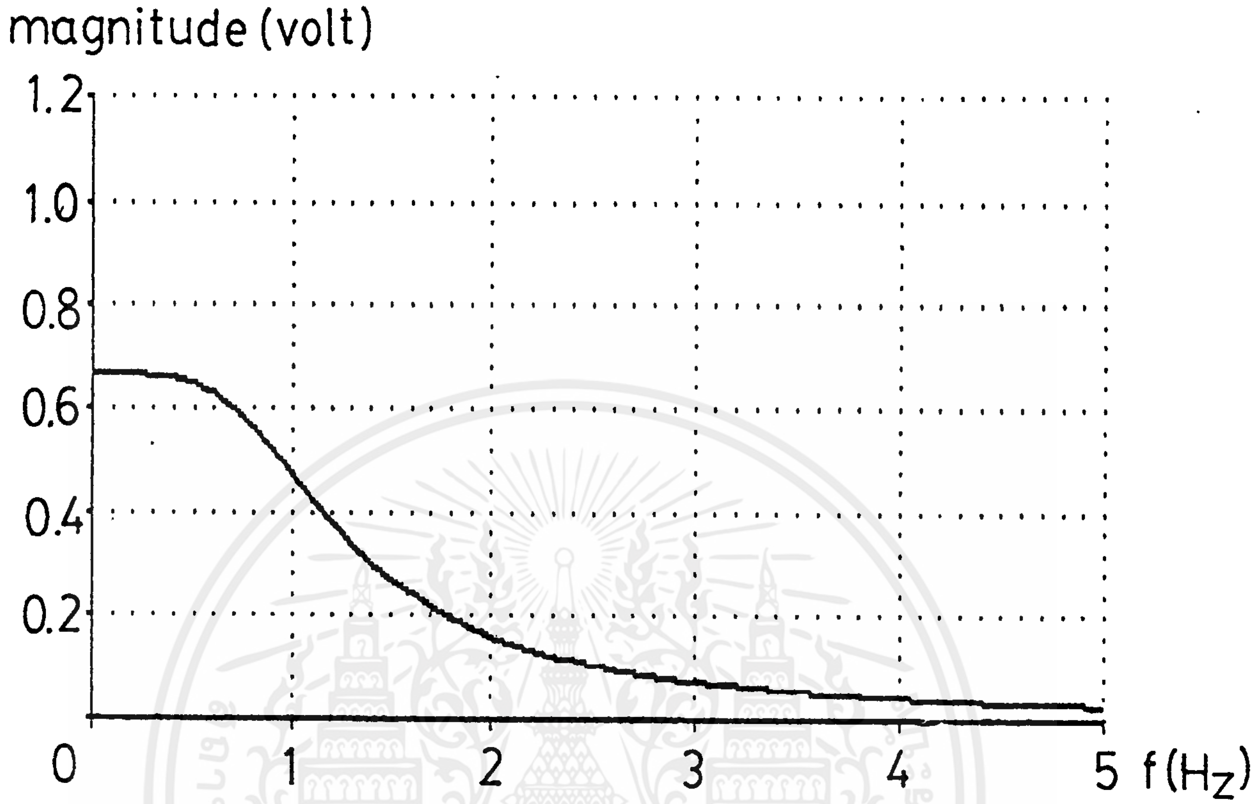
และสมการกรุ๊ฟดีเลย์เป็น

$$Td = \frac{12.746\ \omega^2 + 12.72}{9.036\ \omega^4 - 0.058\ \omega^2 + 9}$$

วงจรมีผลตอบสนองทางขนาดต่อความถี่ดังรูปที่ 12

จากรูปจะเห็นว่า วงจรมี pass band ในช่วง 0-1 Hz ดังนั้นจะทำการอิควอไลซกรุ๊ฟดีเลย์ในช่วงนี้เท่านั้น ซึ่งในช่วงนี้ยังมีความคลาดเคลื่อนอยู่ประมาณ 296 mS ดังในรูปที่ 13

เพื่อให้ได้ค่าคลาดเคลื่อนที่น้อย จะใช้จำนวนเซคชั่นของวงจร All-pass order 2 เท่ากับ 3 และค่าคลาดเคลื่อนมีค่าไม่เกิน 10 mS จะได้สัมประสิทธิ์เริ่มต้นคือ



รูปที่ 12 แสดงผลตอบสนองทางขนาดของ butterworth filter

$$\Delta f = \frac{1}{2.5} = 0.4$$

$$p = \frac{0.4}{2\pi} \ln \frac{4}{0.4 \times 0.01} = 0.4397$$

$$q1 = (0.4)(0.5) = 0.2$$

$$q2 = (0.4)(0.5) = 0.6$$

$$q3 = (0.4)(2.5) = 1.0$$

และป้อนค่าเหล่านี้ในโปรแกรม

-สมการกรุ๊ฟดีเลย์ของวงจรที่ต้องการจะอิควอไลซ์ เป็นบวก

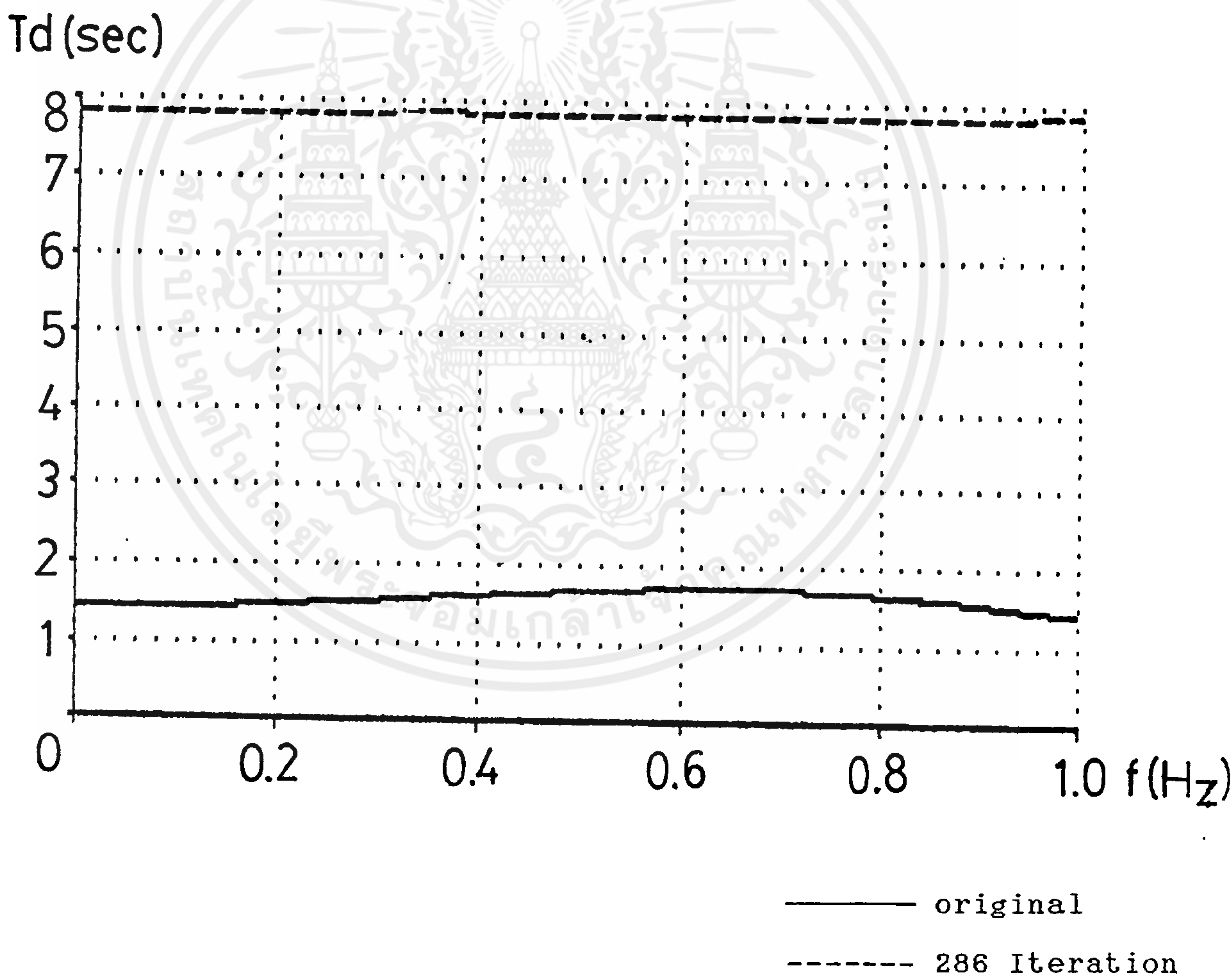
-ค่าเฉลี่ยกรุ๊ฟดีเลย์ เท่ากับ 1.25 s

-ค่าคลาดเคลื่อนกรุ๊ฟดีเลย์ เท่ากับ 1 mS

-จำนวนสัมประสิทธิ์ (p,q) เท่ากับ 6

เมื่อทำการอิเทอเรท ซึ่งจะใช้จำนวนครั้งในการอิเทอเรทช่วงแรกเท่ากับ 170 ครั้ง และช่วงหลังรวมเท่ากับ 286 ครั้ง ผลลัพธ์ที่ได้ จะให้กรุ๊ฟดีเลย์คงที่ในช่วง 1 Hz และมีค่าคลาดเคลื่อนประมาณ 8.6 mS ดังแสดงในรูปที่ 13 และได้โคออดิเนทของ (p,q) เซคชั่นต่างๆ คือ

p1	=	0.83492	q1	=	0.27830
p2	=	0.70721	q2	=	1.18607
p3	=	0.51208	q3	=	1.72562



รูปที่ 13 แสดงผลจากการอิควอไลซ์เทียบกับก่อนอิควอไลซ์

จากค่าโพลและซีโร่ที่ได้จากโปรแกรม เมื่อแทนค่าลงในสมการ (4) และเปรียบเทียบสัมประสิทธิ์ที่ได้กับสมการ (24) แล้วทำการสเกลทั้งทางขนาดและความถี่ ซึ่งจะได้ค่าของอุปกรณ์แต่ละตัวของวงจร All-pass อันดับที่ 2 แต่ละชุดคือ

Section 1

R_1 = 1.71 Ω , R_2 = 1.9 KΩ , R_3 = 2 KΩ , R_4 = 555 Ω

C = 10 μF

Section 2

R_1 = 594 Ω , R_2 = 2.25 KΩ , R_3 = 1.6 KΩ , R_4 = 1.51 KΩ

C = 10 μF

Section 3

R_1 = 251 Ω , R_2 = 3.11 KΩ , R_3 = 1.5 KΩ , R_4 = 4.63 KΩ

C = 1C μF

ตัวอย่างที่ 3

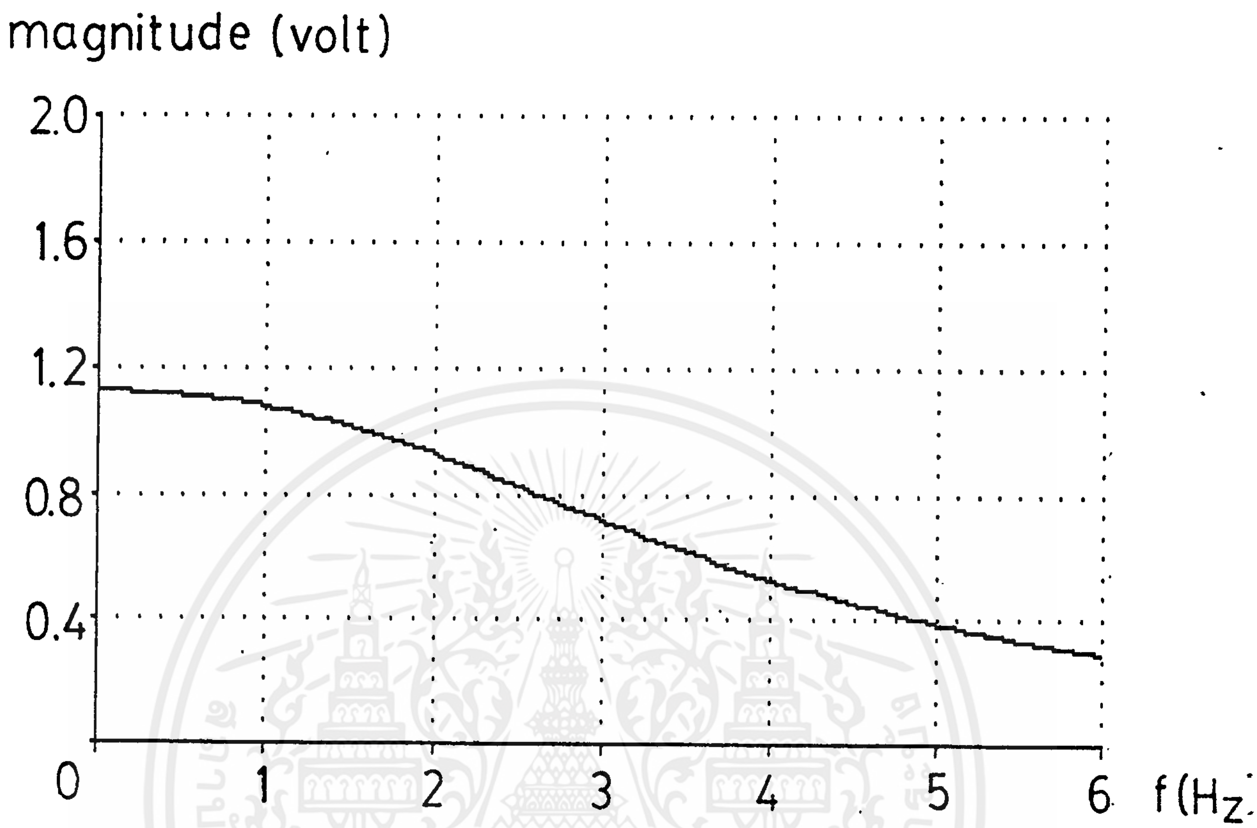
ทดลองกำหนดวงจรที่มีทรานส์เฟอร์ฟังก์ชันเป็น

$$H(S) = \frac{(10.6935-\omega^2)-j6.01480}{(10.6935-\omega^2)+j6.04108}$$

ซึ่งจะได้ฟังก์ชันของกรุ๊ปดีเลย์คือ

$$Td = \frac{6.04108}{9.1236+(\omega-1.2529)^2} + \frac{6.04108}{9.1236+(\omega+1.2529)^2}$$

วงจรนี้ยังมีค่ากรุ๊ปดีเลย์ไม่คงที่ในช่วง 0-6 Hz มีค่าประมาณ 840 mS ดังรูปที่ 13



รูปที่ 13 แสดงค่ากรุ๊ฟดีเลย์ที่ไม่คงที่

เราจะทำการปรับค่ากรุ๊ฟดีเลย์ให้คงที่ในช่วงแบนด์วิธ 6 Hz และใช้วงจร All-pass จำนวน 5 เซคชัน ซึ่งพิจารณาจากพื้นที่ที่ต้องทำการอิควอไลซ์ดังนั้น จะได้ ค่าระยะห่างระหว่างโพลหรือซีโร่ คือ

$$\Delta f = \frac{6}{4.5} = 1.3333$$

กำหนด ค่าคลาดเคลื่อน (ε) มีค่าเป็น 5 ms จะได้ระยะห่างโพลกับแกน imajinary เป็น

$$p = \frac{1.3333}{2\pi} \ln \frac{4}{(1.3333)(0.005)}$$
$$= 1.3575$$

และจะได้ระยะห่างระหว่างโพลกับแกน real ของเซคชันต่างๆคือ

$$q1 = (1.3333)(0.5) = 0.6667$$
$$q2 = (1.3333)(1.5) = 2.0000$$
$$q3 = (1.3333)(2.5) = 3.3333$$
$$q4 = (1.3333)(3.5) = 4.6667$$
$$q5 = (1.3333)(4.5) = 6.0000$$

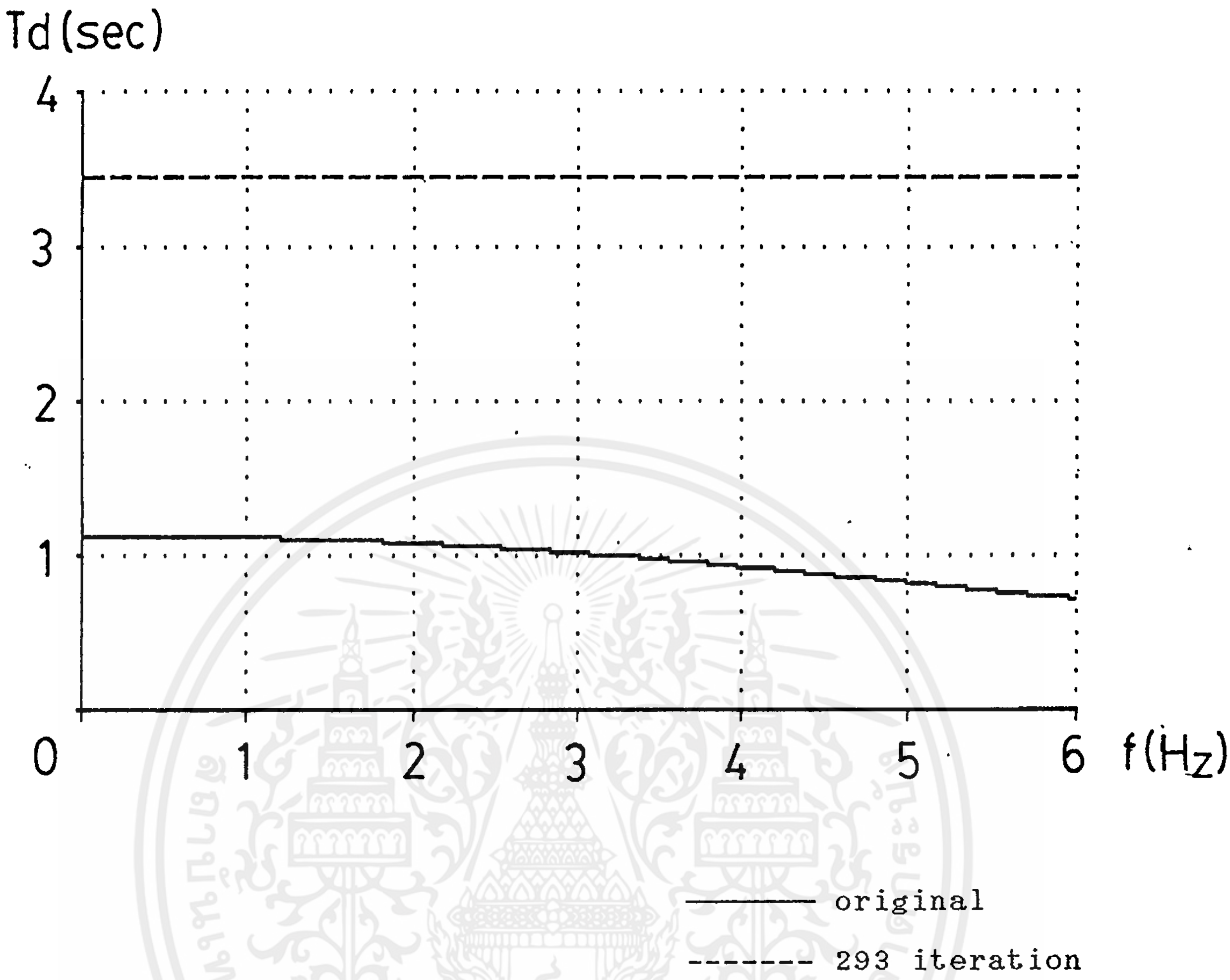
ดังนั้นจะได้ค่าตำแหน่งของโพล (p,q) ของเซคชันต่าง ๆ คือ (1.3575, 0.6667) , (1.3575,1.9999) ,(1.3575,3.3333) ,(1.3575,4.6666), (1.3575,5.9999) ซึ่งค่านี้จะเป็นค่าเริ่มต้นให้โปรแกรมนำไปคำนวนต่อไป นอกจากนี้ยังกำหนด

- ค่าเฉลี่ยของกรุ๊ฟดีเลย์ (Do) ในที่นี้ให้มีค่าเท่ากับ 0.55 s
- ค่าคลาดเคลื่อนของกรุ๊ฟดีเลย์ (E) ซึ่งในที่นี้ทดลองจาก 0.0055 ไป แต่ในโปรแกรมใช้ค่า 0.0009 s ซึ่งค่านี้ยิ่งน้อยก็จะยิ่งได้ผลของกรุ๊ฟดีเลย์ที่ราบเรียบยิ่งขึ้น แต่ก็ต้องดูว่าโปรแกรมสามารถทำได้หรือไม่
- สมการกรุ๊ฟดีเลย์ของวงจรที่ต้องการจะแก้ ก็คือสมการที่ผ่านมา

ผลที่ได้คือ อิเทอเรทช่วงแรก 182 ครั้ง และช่วงหลังรวมเท่ากับ 293 ครั้ง และตำแหน่ง (p,q) ของเซคชันต่าง ๆ คือ

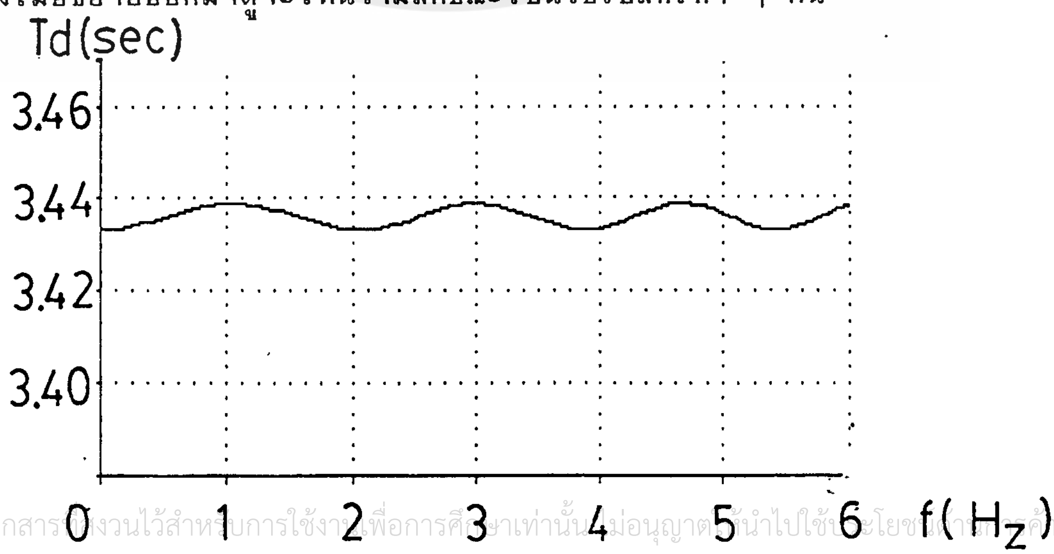
$$p1 = 2.81671 \qquad q1 = 1.19108$$
$$p2 = 2.20909 \qquad q2 = 3.18929$$
$$p3 = 1.97361 \qquad q3 = 4.82457$$
$$p4 = 1.71919 \qquad q4 = 6.32644$$
$$p5 = 1.22616 \qquad q5 = 7.69616$$

เมื่อนำค่า (p,q) เหล่านี้ มาแทนค่าลงในสมการของวงจร All-pass order 2 เซคชันต่าง ๆ นำมาต่อคาสเคดกับวงจรที่ต้องการจะแก้ เมื่อพล๊อตกราฟลักษณะกรุ๊ฟดีเลย์ออกมาจะได้ดังรูปที่ 14



รูปที่ 14 แสดงผลจากการอิควอไลซ์กรุ๊ฟดีเลย์

ผลที่ได้ จะมีค่าคงที่ในช่วง 6 H_z และมีค่าคลาดเคลื่อนประมาณ 5.6 mS ซึ่งเมื่อขยายออกมาดูจะเห็นว่ามีลักษณะเป็นริปเปิลที่เท่า ๆ กัน



รูปที่ 15 แสดงว่าริปเปิลของกรุ๊ฟดีเลย์ที่ผ่านการอิควอไลซ์

เมื่อลองใช้ 3 เซคชัน จะได้

$$\Delta f = \frac{6}{2.5} = 2.4 \quad MH_z$$

$$p = \frac{2.4}{2\pi} = \frac{4}{(2.4 \times 0.005)} = 2.2189$$

$$q1 = 2.4(0.5) = 12$$

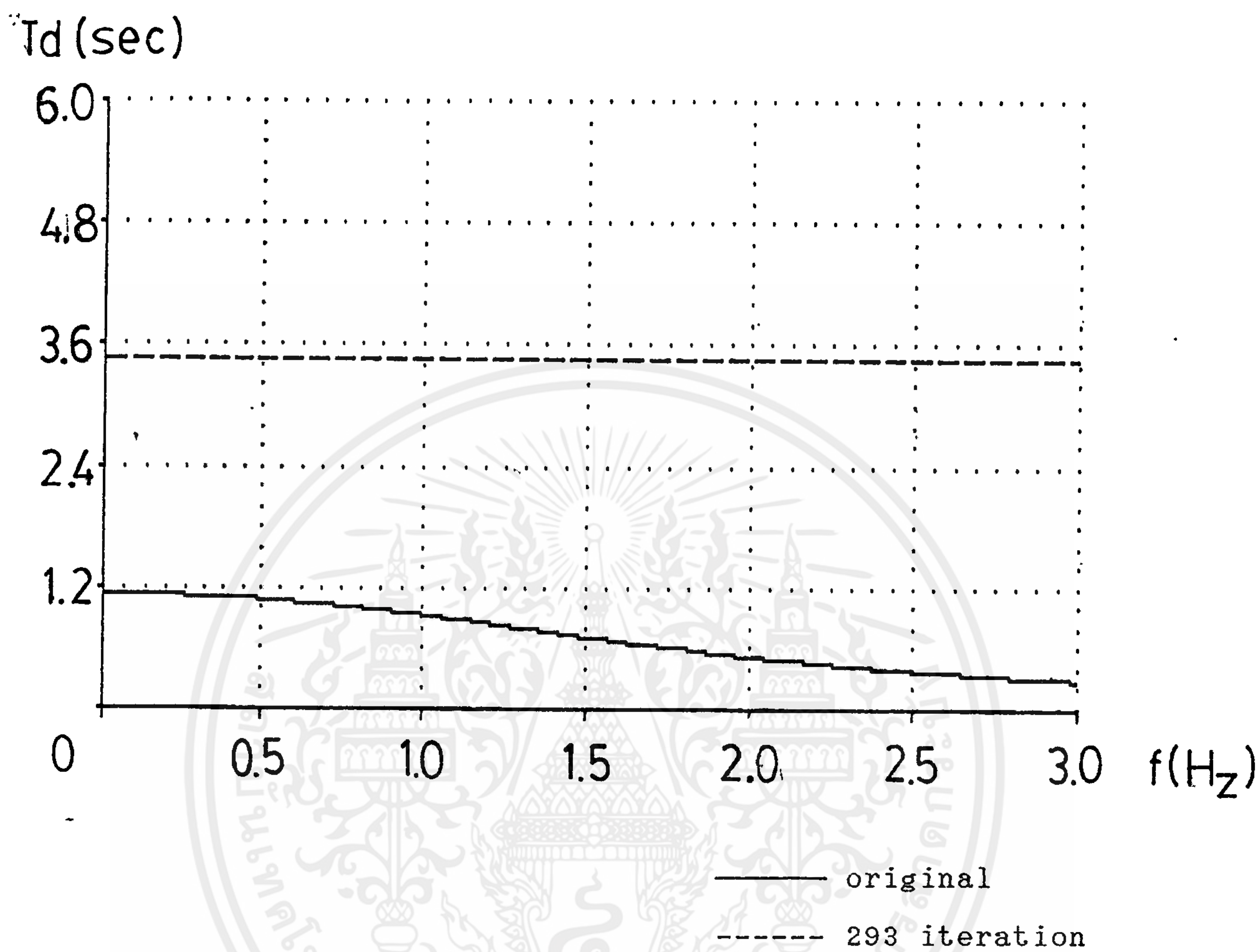
$$q2 = 2.4(1.5) = 3.6$$

$$q3 = 2.4(2.5) = 6.0$$

ในโปรแกรมใช้ค่า Do = 0.55 s ε = 0.0009 s จะได้อิเทอเรชันช่วงแรกเท่ากับ 182 ครั้ง และช่วงหลังรวมเท่ากับ 293 ครั้ง และค่า (p,q) เซคชันต่าง ๆ คือ

p1 = 2.37608	q1 = 1.03292
p2 = 1.85723	q2 = 2.83091
p3 = 1.26986	q3 = 4.24189

ผลที่ได้จะได้ค่าคงที่ในช่วง 0-2.5 Hz เท่านั้น ดังรูปที่ 16



รูปที่ 16 แสดงค่ากรุ๊ฟดีเลย์เมื่อใช้วงจร All-pass 3 เซคชัน

ภาคผนวก ก

การสเกลสิ่ง

(Scaling)

การสเกลสิ่งแบ่งเป็น 2 แบบคือ

1. การสเกลทางขนาด (Magnitude Scaling)
2. การสเกลทางความถี่ (Frequency Scaling)

1. การสเกลทางขนาด

พิจารณาอิมพีแดนซ์ของพาสซีฟอิลิเมนท์

$$Z_R = R$$

$$Z_L = j\omega L$$

$$Z_C = \frac{1}{j\omega C} \qquad (ก.1)$$

จาก (ก.1) หาขนาดของอิมพีแดนซ์แต่ละตัวได้

$$|Z_R| = R$$

$$|Z_L| = \omega L$$

$$|Z_C| = \frac{1}{\omega C} \qquad (ก.2)$$

จาก (ก.2) ถ้าต้องการเพิ่มหรือลดขนาด เราจะสเกลขนาดด้วยค่าคงที่ค่าหนึ่งเท่ากับ Km คือ

$$km\,|Z_R| = km\;R$$

$$km\,|Z_L| = km\;\omega L$$

$$km\,|Z_C| = \frac{km}{\omega C} = \frac{1}{\omega(\frac{c}{km})}$$

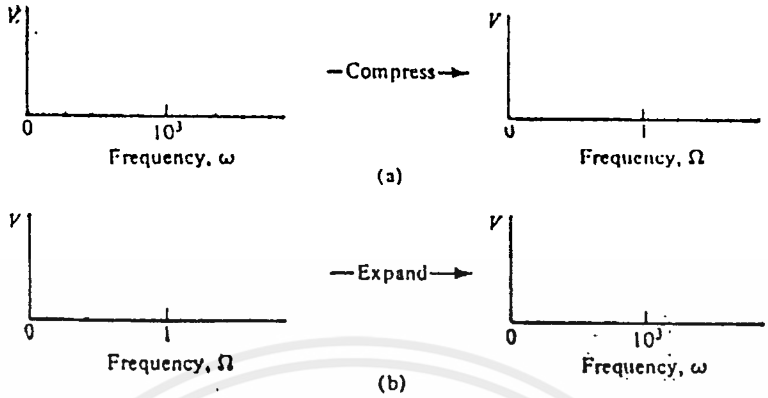
ดังนั้นจะได้ค่าของอิลิเมนท์ใหม่แต่ละตัว คือ

$$Rnew = Km\;Rold$$

$$Lnew = Km\;Lold$$

$$Cnew = \frac{1}{km}\,Cold$$

2. การสเกลทางความถี่



รูปที่ ก.1

จาก

$$|Z_R| = R$$

$$|Z_L| = \omega L$$

$$|Z_C| = \frac{1}{\omega C}$$

ค่า Z_R ไม่สามารถทำการสเกลทางความถี่ได้ เพราะไม่ขึ้นกับความถี่

$$Rnew = Rold$$

$$|Z_L| = \omega L = (\frac{L}{Kf}) = (Kf\,\omega)\ Lnew$$

เมื่อ

$$Lnew = \frac{1}{Kf}\ Load$$

จะเห็นว่า ค่า Z_L ยังมีค่าเท่าเดิม แต่เราสามารถใช้ L ค่าใหม่ได้

$$|Z_C| = \frac{1}{\omega C} = \frac{1}{(Kf\,\omega)(\frac{C}{Kf})} = \frac{1}{(Kf\,\omega)(Cnew)}$$

เมื่อ

$$Cnew = \frac{1}{Kf}\ Cold$$

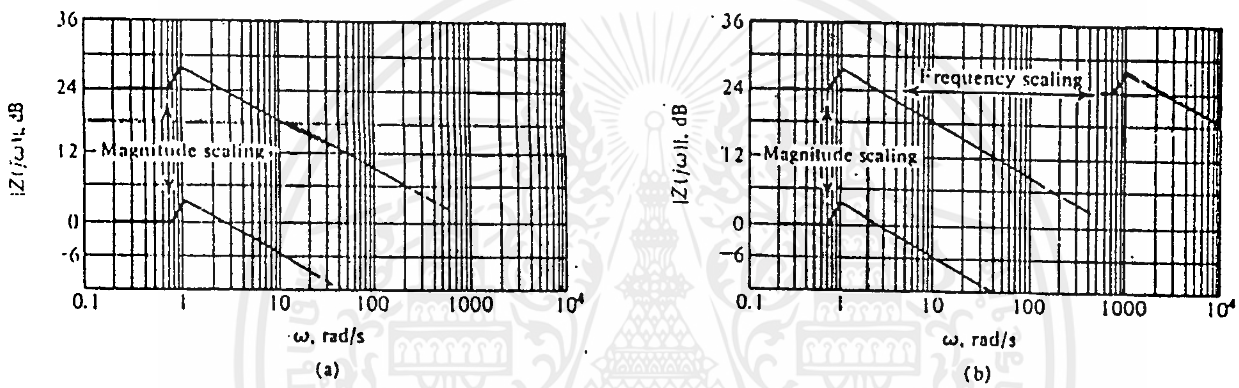
ถ้าทำการสเกลทั้งทางขนาดและความถี่พร้อมกัน จะได้

$$\text{Rnew} = \text{Km Rold}$$

$$\text{Lnew} = \frac{\text{Km}}{\text{Kf}} \text{Lold}$$

$$\text{Cnew} = \frac{1}{\text{Km Kf}} \text{Cold}$$

ดูค่าผลตอบสนอง (response) ที่ได้จากการสเกลในรูป ก.2



ตัวอย่าง วงจรอนุกรม RLC วงจรหนึ่ง มีค่า R=0.1 ohm, L=1H, C=1F จงทำการสเกลทั้งทางขนาดและความถี่เพื่อหาค่าอิลิเมนท์แต่ละตัวใหม่

<u>วิธีทำ</u> เริ่มจาก ให้ Cnew = 1 nF

นั่นคือ $\text{Km} = 10^5$, $\text{Kf} = 10^6$

$$\text{Cnew} = \frac{1}{\text{Km Kf}} \text{Cold}$$

$$= \frac{1}{10^3 \times 10^6} \bullet 1$$

$$= 1 \text{ nF}$$

$$Lnew = \frac{Km}{Kf} Lold$$
$$= \frac{10^3}{10^6} \bullet 1$$
$$= 1 \text{ mH}$$

$$Rnew = Km\ Rold$$
$$= 10^3 \times 0.1$$
$$= 100$$

การหาค่า Km , Kf จะต้องทดลองดูค่าที่หาได้ของอิลิเมนท์แต่ละตัวด้วย

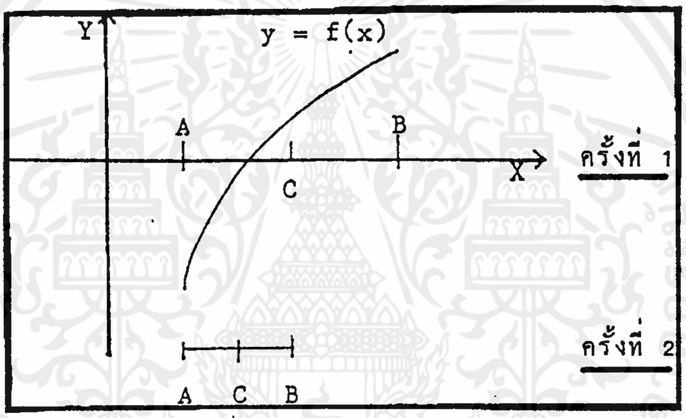
ภาคผนวก ข

รากของสมการ f(X)=0

การพิจารณาฟังก์ชั่นที่ต่อเนื่อง f(X) ว่า f(X)=0 มีรากบนช่วง (a,b) หรือไม่ มีหลายวิธีการ แต่ในที่นี้จะเสนอเพียง ๒ วิธี เท่านั้นคือ วิธีแบ่งครึ่งช่วง และวิธีของนิวตัน-ราฟสัน

1. วิธีแบ่งครึ่งช่วง (Half inteval method)

ถ้า y=f(x) เป็นฟังก์ชั่นที่มีความต่อเนื่องบนช่วงต้องหาช่องมา (a,b)และ f(x)=0 มีรากบนช่วง (a,b) และกราฟของ y=f(x) มีการเปลี่ยนเครื่องหมายช่วง (a,b) เรามีวิธีพิจารณารากของสมการ f(x)=0 ดังนี้



รูปที่ ข.1

<u>ขั้นตอน</u> 1. เพราะว่า f(x) ต่อเนื่องบนช่วง (a,b) และมีการเปลี่ยนเครื่องหมายบนช่วง (a,b) ดังนั้นต้องมี A และ B บนช่วง (a,b) ที่ทำให้ A < B และ f(A)f(B) <= 0 ถ้า F(A)F(B) = 0 แสดงว่า A หรือ B เป็นรากของสมการ

<u>ขั้นตอน</u> 2. ถ้า f(A)f(B) < 0 ให้ C = (a+b)/2 จะได้ว่า f(C) = 0 หรือ f(C)f(A) < 0 หรือ f(C)f(B) < 0

<u>กรณี 1</u> ถ้า f(C) = 0 แสดงว่า C เป็นรากของสมการ

<u>กรณีที่ 2</u> ถ้า f(A)f(C) < 0 แสดงว่า รากของสมการอยู่บนช่วง [A,C] เลือกให้ B = C แล้วกลับไปทำตามขั้นตอน ข.

<u>กรณีที่ 3</u> ถ้า f(C)f(B) < 0 แสดงว่า รากของสมการอยู่บนช่วง [C,B] เลือกให้ A = C แล้วกลับไปทำตามขั้นตอน ข.

ด้วยการคำนวณตามขั้นตอนที่กล่าวจะได้ว่าช่วง [A,B] จะเล็กลงเรื่อยๆ และคลุมรากของสมการ $f(x) = 0$ การพิจารณาค่ารากของ $f(x)=0$ อาจทำดังนี้

1. ช่วงที่คลุมรากของสมการเล็กเพียงพอหรือ
2. เมื่อ $C = (A+B)/2$ ค่าของ $f(C)$ มีค่าเข้าใกล้ศูนย์เพียงพอ

<u>ตัวอย่าง1.2.1</u> กำหนดให้ $f(x) = x^2 - 2$ บนช่วง [0,2] จงหารากของสมการ $f(x) = 0$ ที่ทำให้ $\left|f(x)\right| < 0.00005$

<u>วิธีทำ ครั้งที่ 1</u> ให้ A = 0 และ B = 2 จะได้ $C = (A+B)/2 = 1$

$F(1) = -1$

$F(0)F(1) = (-2)(-1) = 2 > 0$

$F(1)F(2) = (-1)(2) = -2 < 0$

ดังนั้นช่วงที่คลุมรากสมการคือ [1,2]

<u>ครั้งที่ 2</u> ให้ A = 1 และ B = 2 จะได้ $C = (A+B)/2 = 1.5$

$f(1.5) = 0.25$

$f(1)f(1.5) = (-1)(0.25) < 0$

$f(1.5)f(2) = (0.25)(2) > 0$

ดังนั้นช่วงที่คลุมรากสมการคือ [1,1.5]

<u>ครั้งที่ ๓</u> ให้ A = 1 และ B = 1.5 จะได้ $C = (A+B)/2 = 1.25$

$f(1.25) = -0.4375$

$f(1)f(1.25) = (-1)(-0.4375) > 0$

$f(1.25)f(1.5) = (-0.4375)(0.25) < 0$

ดังนั้นช่วงที่คลุมรากสมการคือ [1.25,1.5]

เมื่อคำนวณต่อไปสรุปเป็นตารางให้ดังนี้

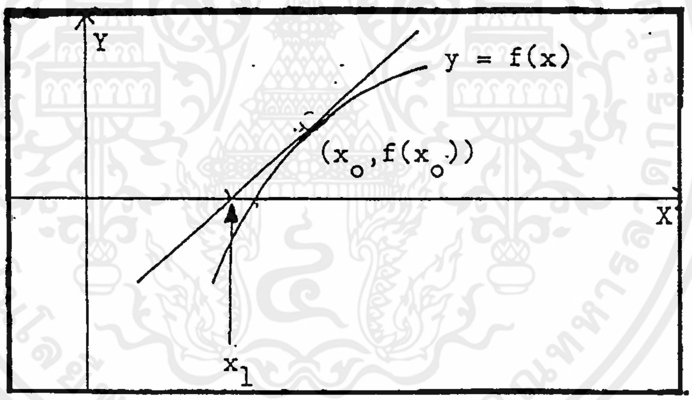
n	A	B	C	f(C)
1	0.000000	2.000000	1.000000	-1.000000
2	1.000000	2.000000	1.500000	0.2500000
3	1.000000	1.500000	1.250000	-0.437500
4	1,250000	1.500000	1,375000	-0.109375
5	1.375000	1.500000	1.473500	0.0664063
6	1.375000	1.437500	1.406250	-0.0224609
7	1.406250	1.437500	1.421875	0.0217285
8	1.406250	1.421875	1.414063	-0.0004272

9	1,414063	1,421875	1.417969	0.0106354
10	1.414063	1.417969	1.416016	0.0051003
11	1.414063	1.416016	1,415039	0.0023355
12	1.414063	1.415039	1.414151	0.009539
13	1.414036	1.414551	1.414307	0.0002632
14	1.414063	1.414307	1.414185	-0.0000820
15	1.414185	1.414307	1.414246	0.0000906
16	1.414185	1.414246	1.414251	0.0000043

สรุปรากของสมการคือ $x = 1.414215$

2. วิธีนิวตัน-ราฟสัน (The Newton-Raphson method)

ถ้าฟังก์ชั่น $f(x)$ เป็นฟังก์ชั่นที่อนุพันธ์สามารถหาค่าได้และสมการ $f(x) = 0$ มีรากเราสามารถหารากของสมการ $f(x) = 0$ ดังนี้



รูปที่ ข.๒

เลือก x_o เป็นค่าเริ่มต้นในการประมาณรากของสมการ ต่อไปลากเส้นสัมผัสเส้นโค้ง $y = f(x)$ ได้สมการเส้นสัมผัสเส้นโค้ง $y = f(x)$ ที่จุด $(x_o, f(x_o))$ เป็น

$$y - f(x_o) = f'(x_o)(x - x_o)$$

จุดตัดของเส้นสัมผัสกับแกน x คือจุด $(x_1, 0)$ เมื่อ

$$0 - f(x_o) = f'(x_o)(x_1 - x_o)$$

ดังนั้น $$x_1 = x_o - f(x_o)/f'(x_o)$$

ต่อไปลากเส้นสัมผัสเส้นโค้ง $y = f(x)$ ที่จุด $(x_1, f(x_1))$ จะได้สมการเส้นสัมผัสเป็น

$$y - f(x_1) = f'(x_1)(x - x_1)$$

จุดตัดของเส้นสัมผัสกับแกน x คือจุด $(x_2,0)$ เมื่อ

$$0 - f(x_1) = f'(x_1)(x_2 - x_2)$$

ดังนั้น $$x_2 = x_1 - f(x_1)/f'(x_1)$$

ในทำนองเดียวกันเราจะได้ว่า

$$x_3 = x_2 - f(x_2)/f'(x_2)$$

$$x_4 = x_3 - f(x_3)/f'(x_3)$$

และ $$x_{n+1} = x_n - f(x_n)/f'(x_n), n = 0,1..$$

ในกรณีที่ x_n ลู่เข้าจะได้ว่าลิมิต x_n เป็นรากของสมการ $f(x) = 0$

ในทางปฏิบัติ เราอาจหยุดการคำนวณ เมื่อได้ x_n ดังนี้

1. $|f(x_n)|$ มีค่าน้อยเพียงพอ เช่น $|f(x_n)| < 0.000005$

หรือ 2. เปอร์เซนต์ค่าคลาดเคลื่อนสัมพันธ์ $|(x_n - x_{n-1})/x_n|*100$ มีค่าน้อยเพียงพอ

เช่น $|(x_n - x_n - 1)/x_n|*100 < 0.05$

<u>ตัวอย่าง 1.4.1</u> จงประมาณค่ารากของสมการ $f(x) = x^2-2 = 0$ บนช่วง [0,2] โดยกำหนดให้เปอร์เซนต์ค่คลาดเคลื่อนสัมพัทธ์มีค่าน้อยกว่า 0.05

<u>วิธีทำ</u> $f(x) = x^2-2$

$f'(x) = 2x$

เลือก $x_o = 1$

$x_1 = x_o - f(x_o)/f'(x_o)$

$= 1 - f(1)/f,(1)$

$= 1 - (-1)/2$

$= 1.5$

เปอร์เซนต์คลาดเคลื่อนสัมพัทธ์เท่ากับ $|(x_1-x_o)/x_1|*100 = 33.33$

ต่อไป $x_2 = x_1 - f(x_1)/f'(x_1)$

$= 1.5 - f(1.5)/f'(1.5)$

$- 1.5 - (0.25)/3$

$= 1.416667$

เปอร์เซนต์ค่าคลาดเคลื่อนสัมพันธ์เท่ากับ $\left|\frac{x_2 - x_1}{x_2}\right| 100 = 5.88$

เมื่อคำนวณต่อไปเราสรุปเป็นตารางได้ดังนี้

n	x	f(x)	ค่าคลาดเคลื่อน (x)
1	1.000000	-1.0000000	33.3333400
2	1.500000	0.2500000	5.8823560
3	1.416667	0.0069444	0.1733076
4	1.414216	0.0000060	0.0001517

สรุปได้ว่ารากของสมการคือ x = 1.414216 #

ปัญหาและข้อเสนอแนะบางประการเกี่ยวกับการประมาณรากของสมการของ f(x) = 0 โดยวิธีของนิวตัน-ราฟสัน

1. การเลือกค่าเริ่มต้น x_o อาจทำให้ขั้นตอนการหาค่า x_n บางขั้นตอนมีค่า x_n ที่ทำให้ $f'(x_n) = 0$ ได้เช่น จากตัวอย่าง 1.4.1 ถ้าเลือก $x_o = 0$

จะได้ว่า $f(x_o) = f(0) = -2$

แต่ $f'(x_o) = f'(0) = 0$

ทำให้เราไม่สามารถหาค่า x_1 ได้

วิธีแก้ปัญหาข้างต้นคือ เลือก x_o เป็นค่าใหม่ให้เหมาะสม ก็จะทำให้ได้ค่าประมาณรากของสมการ

2. การเลือกค่าเริ่มต้น x_o อาจทำให้การได้ค่า x_n ที่สอดคล้องเงื่อนไขทั้งค่าคลาดเคลื่อนสัมพัทธ์น้อยกว่าค่าที่กำหนดให้ และค่า $|f(x_n)|$ น้อยกว่าค่าที่กำหนดให้นั้น ค่าของ x_n ที่ได้อาจจะอยู่นอกโดเมนของฟังก์ชั่นที่กำหนดให้ก็ได้ ตัวอย่างเช่น จากตัวอย่าง 1.4.3 ถ้าเราเลือก $x_o = 0$ จะทำให้ได้ผลดังนี้

n	x	f(x)	ค่าคลาดเคลื่อน (x)
1	12.500000	703.1250000	100.0000000
2	9.844192	189.3405000	26.9784200
3	8.378905	43.0858200	17.4878100
4	7.785074	5.8336110	7.6278150
5	7.675545	0.1828919	1.4269870
6	7.671883	0.0002136	0.0477404
7	7.671879	-0.0000153	0.0000559

นั่นคือ $x = 7.671879$ เป็นรากของสมการ $x^3 - 8x^2 - 4x + 50 = 0$ แต่อยู่นอก โดเมนของฟังก์ชั่น f
วิธีแก้ปัญหาคือ เลือกค่า x_o ให้ใกล้ค่ารากจริงมากที่สุด โดยการประมาณจากกราฟของสมการ $y = f(x)$

3. การหาสูตรของ $f'(x)$ ในบางครั้งหากสูตรของ $f'(x)$ หาได้ยากเราสามารถประมาณค่า $f'(x)$ ได้โดย

$$f'(x) = \frac{f(x+h) - f(x)}{h}$$

โดยใช้ค่า h ที่เหมาะสม เช่น h = 0.1,0.01,0.05 เช่นจากตัวอย่าง 1.4.2

$$f(x) = e^{-x} - x$$

$$f'(x) = \frac{f(x+0.01) - f(x)}{0.01}$$

และเลือก $x_o = 1$ จะได้ผลสรุปเป็นตารางได้ดังนี้

n	x	f(x)	ค่าคลาดเคลื่อน (x)
1	0.537264	0.0470814	86.1283600
2	0.567035	0.0001697	5.2503610
3	0.567144	-0.0000004	0.0191275
4	0.567143	0.0000001	0.0000420

4. การเลือกค่า x^o ที่ไม่เหมาะสมอาจทำให้กว่าที่จะได้ราก ที่ต้องการต้องใช้เวลาในการคำนวณหลายครั้ง เช่น อาจตัวอย่าง 1.4.3 เมื่อเลือก $x_o = 0.5$ เราต้องคำนวณถึง x_o จึงจะได้ x_n ที่สอดคล้องเงื่อนไขว่าเปอร์เซนต์ค่าคลาดเคลื่อนสัมพัทธ์น้อยกว่า 0.0005 ดังตารางต่อไปนี้

n	x	f(x)	ค่าคลาดเคลื่อน (x)
1	4.600000	-40.3440000	89.1304400
2	1.742776	24.0240400	163.9468000
3	2.797729	-1.9105800	37.7074900
4	2.722158	0.0018120	2.7761550
5	2.722229	0.0000000	0.0026275
6	2.722229	0.0000000	0.0000000

แต่ถ้าเราเลือก $x_o = 2.5$ เราคำนวณเพียง x_3 เท่านั้น

n	x	f(x)	ค่าคลาดเคลื่อน (x)
1	2.722772	-0.0137634	8.1818210
2	2.722229	0.0000038	0.0199687
3	2.722229	0.0000000	0.0000088

ตัวอย่าง 1.4.4

จงประมาณรากของสมการ $f(x) = 0.1 + 0.51x - \sin x = 0$ โดยกำหนดค่าเริ่มต้นเป็น $x_o = 0$ และให้ค่าประมาณ x_n ที่หาได้มีเปอร์เซนต์ค่าคลาดเคลื่อนสัมพัทธ์น้อยกว่า 0.0005 และ $|f(x_n)| < 0.0000005$

วิธีทำ $f(x) = 0.1+0.51x-\sin x$

$x_o = 0$

แบบที่1 ใช้สูตรแม่นตรงของ $f'(x)$

$f'(x) = 0.51-\cos x$

ผลการคำนวณสรุปเป็นตารางได้ดังนี้

n	x	f(x)	ค่าคลาดเคลื่อน (x)
1	0.204082	0.0014137	100.0000000
2	0.207094	0.0000010	1.4547200
3	0.207096	0.0000000	0.0009858
4	0.207096	0.0000000	0.0000000

แบบที่2 ใช้สูตรประมาณค่าของ $f'(x)$ ได้

$$f'(x) = \frac{f(x + 0.01) - f(x)}{0.01}$$

ผลการคำนวณสรุปเป็นตารางได้ดังนี้

n	x	f(x)	ค่าคลาดเคลื่อน (x)
1	0.2.4088	0.0014105	100.0000000
2	0.207101	-0.0000022	1.4545950
3	0.207096	0.0000000	0.0022305
4	0.207096	0.0000000	0.0000000

โดยการคำนวณทั้ง 2 แบบจะได้ราก $x = 0.207096$ #

การหารากของสมการ $f(x) = 0$ สำหรับบางฟังก์ชั่น $f(x)$ อาจพบปัญหาบางประการ เช่น

$$f(x) = x^3 - 5x^2 + 7x - 3$$

จะไม่มีการเปลี่ยนเครื่องหมายบนช่วง [0,2] เพราะว่า $f(x) = 0$ ทุกค่า x บนช่วง [0,2] ดังนั้นเราไม่สามารถหารากของสมการ

$$f(x) = x^3 - 5x^2 + 7x - 3 = 0$$

บนช่วง [0,2] โดยวิธีแบ่งครึ่งช่วง และวิธีตำแหน่งผิด
สำหรับฟังก์ชั่น $f(x)$ บางฟังก์ชั่น $f(x) = 0$ อาจจะมีรากซ้ำได้เช่น

$$f(x) = x^3 - 5x^2 + 7x - 3$$
$$= (x - 3)(x - 1)(x - 1) = 0$$

มีราก 1 ซ้ำกัน 2 ตัว และรากเป็น 3 หนึ่งตัว

$$f(x) = x^4 - 6x^3 + 12x^2 - 10x+3$$
$$= (x-3)(x-1)(x-1)(x-1) = 0$$

มีราก 1 ซ้ำกัน 3 ตัว และรากเป็น 3 หนึ่งตัว
สำหรับฟังก์ชั่นที่เราคาดว่า $f(x) = 0$ จะมีรากซ้ำนั้น เราอาจจะประยุกต์สูตรของนิวตัน-ราฟสัน ในการหาค่าดังนี้
จากสมการ $f(x) = 0$
และ $f(x), f'(x)$ และ $f''(x)$ มีความต่อเนื่อง จะได้ว่า

$$u(x) = f(x)/f'(x) \quad \text{เมื่อ } f'(x) \neq 0$$

รากของสมการ $f(x) = 0$ และ $u(x) = 0$ จะเป็นตัวเดียวกัน
ดังนั้นการหารากของสมการ $u(x) = 0$ เราสามารถหาได้โดยใช้สูตร

$$x_{n+1} = x_n - u(x_n)/u'(x_n) \quad ,n = 0,1,2....$$

เมื่อ x_0 เป็นค่าเริ่มเป็นค่าเริ่มต้นที่เหมาะสม

การหารากของสมการ $f(x) = 0$ โดยอาศัยวิธีการหารากของ $u(x) = 0$ นั้น นอกจากใช้ได้ในกรณีที่ $f(x) = 0$ มีรากซ้ำกันแล้วยังสามารถทำให้ลดจำนวนครั้งของการคำนวณด้วย นั่นคือ การลู่เข้าของ x_n สู่รากของ $f(x) = 0$ จะเร็วขึ้น
ตัวอย่างเช่น การหารากของ

$$f(x) = x^3 - 5x^2 + 7x - 3 = 0 \text{ บนช่วง } [0,2]$$

เลือกค่าเริ่มต้น $x_0 = 0$ โดยให้ $\left|f(x_n)\right| = 0.0000001$
โดยวิธีใช้สูตร

$$x_{n+1} = x_n - f(x_n)/f^2(x_n)$$

จะสรุปเป็นตารางได้ดังนี้

n	x	f(x)
1	0.428571	-0.83965010
2	0.685714	-0.22859480
3	0.832865	-0.06053639
4	0.913329	-0.01567483
5	0.955784	-0.00399637
6	0.977654	-0.00025368
7	0.988769	-0.00025368
8	0.994369	-0.00006390
9	0.997194	-0.00001597
10	0.998614	-0.00000381
11	0.999301	-0.00000095
12	0.999642	-0.00000000

โดยวิธีใช้สูตร

$$x_{n+1} = u(x_n)/u^2(x_n)$$

เมื่อ $u(x) = f(x)/f^2(x)$

$$= (x^3 - 5x + 7x - 3)/(3x^2 - 10x + 7)$$

เราจะคำนวณได้ดังนี้

n	x	f(x)
1	1.105263	0.05413624
2	1.003079	0.00155255
3	0.999954	-0.00257732
4	0.999907	0.00000000

ผลจากการคำนวณจะได้ว่าการลู่เข้าสู่รากของสมการของ

$$x_{n+1} = x_n - u(x_n)/u^2(x_n)$$

จะเร็วกว่าสูตร

$$x_{n+1} = x_n - f(x_n)/f^2(x_n)$$

ภาคผนวก ค
ผลเฉลยของระบบสมการเชิงเส้น

ในบทนี้ เราศึกษาเกี่ยวกับวิธีที่รัดกุมเพื่อใช้ในการคำนวณหาผลเฉลยของระบบสมการเชิงเส้น n ตัวแปร n สมการ

$$
\begin{aligned}
-0.002x + 4.000y + 4.000z &= 7.998 \\
-2.000x + 2.906y - 5.387z &= -4.481 \\
3.000x - 4.031y - 3.112z &= -4.143
\end{aligned}
$$

ซึ่งมีเมตริกซ์แต่งเติม [A:b] เป็น

$$
\left[\begin{array}{ccc|c}
-0.002 & + 4.000 & + 4.000 & 7.998 \\
-2.000 & + 2.906 & - 5.387 & -4.481 \\
3.000 & - 4.031 & - 3.112 & -4.143
\end{array}\right]
$$

โดยการคำนวณและปัดเศษเพื่อให้เหลือตัวเลขนัยสำคัญ 4 ตัว

วิธีกำจัดของเกาส์ (Gaussian elemination)

วิธีกำจัดของเกาส์เป็นวิธีการหาผลเฉลยของระบบสมการเชิงเส้นที่คิดโดยนักคณิตศาสตร์ชาวเยอรมัน ชื่อ เกาส์, คาร์ล ฟรีดริค (Gauss, Carl friedrich) มีขั้นตอนต่างๆ สำหรับระบบสมการ n สมการ n ตัวแปร Ax = b ดังนี้

1. สร้างเมทริกแต่งเติม [A:b]
2. ทำการสลับแถว (ถ้าจำเป็น) เพื่อให้ได้เมทริกซ์ที่มีคุณสมบัติว่าสมาชิกที่ตำแหน่งแถวที่ 1 หลักที่ 1 มีค่าสัมบรูณ์มากที่สุดในหลักที่ 1 นั้น และถ้าค่านั้นไม่เท่ากับศูนย์เราจะคำนวณต่อไป
3. ทำการดำเนินการเปลี่ยนแถวเชิงธาตุมูลกับเมทริกซ์ที่ได้ เพื่อให้สมาชิกในหลักที่ 1 ของแถวที่ 2,3,4,...,n เป็นศูนย์ทุกตัว ได้เมทริกซ์ $[A:b]^{(1)}$

$$\left[\begin{array}{cccccc|c} a_{11}^{(2)} & a_{12}^{(2)} & a_{13}^{(2)} & \ldots & a_{1n}^{(2)} & & b_1^{(2)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \ldots & a_{2n}^{(2)} & & b_2^{(2)} \\ 0 & 0 & a_{33}^{(2)} & \ldots & a_{3n}^{(2)} & & b_3^{(2)} \\ . & . & . & & . & & . \\ . & . & . & & . & & . \\ . & . & . & & . & & . \\ 0 & 0 & a_{n3}^{(2)} & \ldots & a_{nn}^{(2)} & & b_n^{(2)} \end{array}\right]$$

ในทำนองเดียวกันเมื่อทำต่อไปและ $a_{ii}^{(i)} \# 0$, $i = 1,2,\ldots,n$
เราจะได้เมทริกซ์ที่สมมูลแถวกับเมทริกซ์ของระบบสมการเป็น

$$\left[\begin{array}{cccccc|c} a_{11}^{(n)} & a_{12}^{(n)} & a_{13}^{(n)} & \ldots & a_{1n}^{(n)} & & b_1^{(n)} \\ 0 & a_{22}^{(n)} & a_{23}^{(n)} & \ldots & a_{2n}^{(n)} & & b_2^{(n)} \\ 0 & 0 & a_{33}^{(n)} & \ldots & a_{3n}^{(n)} & & b_3^{(n)} \\ . & . & . & & . & & . \\ . & . & . & & . & & . \\ . & . & . & & . & & . \\ 0 & 0 & a_{n3}^{(n)} & \ldots & a_{nn}^{(n)} & & b_n^{(n)} \end{array}\right]$$

โดยการแทนค่าย้อนกลับจะได้

$$x_n = b_n^{(n)}/a_{nn}^{(n)}$$

$$x_{n-1} = (b_{n-1}^{(n)} - a_{(n-1)n}^{(n)}\, x_n) \;/\; a_{(n-1)(n-1)}^{(n)}$$

$$\vdots$$

$$x_2 = (b_2^{(n)} - a_{23}^{(n)}x_3 - a_{24}^{(n)}x_4 - \ldots - a_{2n}^{(n)}x_n)/a_{22}^{(n)}$$

$$x_1 = (b_1^{(n)} - a_{12}^{(n)}x_2 - a_{13}^{(n)}x_3 - \ldots - a_{1n}^{(n)}x_n)/a_{11}^{(n)}$$

<u>หมายเหตุ</u> 1. $a_{11}^{(1)}, a_{22}^{(2)}, a_{33}^{(3)}, \ldots, a_{nn}^{(n)}$ ในแต่ละขั้นตอนที่ได้ เรียกว่า สมาชิกหลัก (pivot element)

2. $a_{ii}^{(i)}$ บางตัวเป็นศูนย์เราจะหยุดการคำนวณ เพราะว่าระบบสมการ [A:b] อาจมีหรือไม่มีผลเฉลยก็ได้

<u>ตัวอย่าง</u> จงหาผลเฉลยของระบบสมการ

$$2x + y - 2z = 8$$
$$3x + 2y - 4z = 15$$
$$5x + 4y - z = 1$$

โดยวิธีกำจัดของเกาส์

<u>วิธีทำ</u>

$$[A:b] = \left[\begin{array}{ccc|c} 2 & 1 & -2 & 8 \\ 3 & 2 & -4 & 15 \\ 5 & 4 & -1 & 1 \end{array}\right]$$

$R_1 \langle\text{--}\rangle R_3$

$$\left[\begin{array}{ccc|c} 5 & 4 & -1 & 1 \\ 3 & 2 & -4 & 15 \\ 2 & 1 & -2 & 8 \end{array}\right]$$

$R_2 - (3/5)R_1$

$$\left[\begin{array}{ccc|c} 5 & 4 & -1 & 1 \\ 0 & -0.4 & -3.4 & 14.4 \\ 2 & 1 & -2 & 8 \end{array}\right]$$

$R_3 - (2/5)R_1$

$$\left[\begin{array}{ccc|c} 5 & 4 & -1 & 1 \\ 0 & -0.4 & -3.4 & 14.4 \\ 0 & -0.6 & -1.6 & 7.6 \end{array}\right]$$

$R_2 \longleftrightarrow R_3$

$$\left[\begin{array}{ccc|c} 5 & 4 & -1 & 1 \\ 0 & -0.6 & -1.6 & 7.6 \\ 0 & -0.4 & -3.4 & 14.4 \end{array}\right]$$

$R_3 - (-0.4/-0.6)R_2$

$$\left[\begin{array}{ccc|c} 5 & 4 & -1 & 1 \\ 0 & -0.6 & -1.6 & 7.6 \\ 0 & 0 & -2.33333 & 9.33333 \end{array}\right]$$

โดยการแทนค่าย้อนกลับจะได้ว่าผลเฉลยคือ

$$z = 9.33333/(-2.33333) = -4$$

$$y = (7.6 + (1.6)(-4))/(-0.6) = -2$$

$$x = (1 - (-1)(-4) - 4(-2))/5 = 1$$

ภาคผนวก ง

รายละเอียดของโปรแกรม

```
{**********************************************************}
{              This program is complied by               }
{                     turbo-87.com                         }
{**********************************************************}

program Delay_Equalizer;

 const ArraySize = 60;        { Maximum number of equations }
 type complex     = record
                      Re,Im : real;
                     end;
 CompVector = array[0..ArraySize] of complex;
 IntVector  = array[0..ArraySize] of integer;
 RealVector = array[1..ArraySize] of real;
 matrix = array[1..ArraySize] of RealVector;


 var Coefficients                : matrix;
     Constants,Solution,p,q,w    : RealVector;
     data,d1,d2                  : complex;
     e,D0,D,Tol,min,max,mm       : Real;
     Dimen,Row,Column,i,iter     : integer;
     Error                       : byte;
     IterM                       : boolean;
 label loop,exitd;

 function EquF(w:real):real;
```

```
 var D  : real;
     i  : integer;
 begin
  { Input group delay equation to be equalized }

{ Example #1 RL-LPF Wc = 5.56 Hz }
{  D := (5.56/(sqr(5.56)+sqr(w))); }

{ Example #2  low pass butterworth }
{
   D :=  -((12.746*w*w)+12.72)/((9.036*w*w*w*w)
          -(0.058*w*w)+9);
}

{ Example #3 }

   D := (6.04108/(9.1236+sqr(w-1.2529))
        +6.04108/(9.1236+sqr(w+1.2529)));

for i := 1 to (Dimen shr 1) do
     D := D + ( ( 2*p[i] ) / ( sqr(p[i])+sqr(w-q[i]) ) +
                ( 2*p[i] ) / ( sqr(p[i])+sqr(w+q[i]) ) );
   EquF := D/(2*pi);
 end;

function DifF(x: real):real;
 const Tolerance = 1e-10;
 type  vector = array[1..100] of real;
 var   Term,Iter,TwoToTheIterMinus2,Extrap  : integer;
       DeltaX,FourToTheExtrapMinus1         : real;
```

```
    OldEstimate,NewEstimate        : vector;

 function EvaluateFirstDeriv(X      : real;
                           DeltaX : real):real;
 var LeftPoint,RightPoint : real;
 begin
   LeftPoint := EquF(X - DeltaX);
   RightPoint := EquF(X + DeltaX);
   EvaluateFirstDeriv := (RightPoint - LeftPoint)
                         /(2 * DeltaX);
 end;

begin
    if ABS(X) < Tolerance then
      DeltaX := Sqrt(Tolerance)
    else
      DeltaX := ABS(X * Sqrt(Tolerance));
    OldEstimate[1] := EvaluateFirstDeriv(X,DeltaX);
    Iter := 1;
    TwoToTheIterMinus2 := 1;
    repeat
      Iter := Succ(Iter);
      DeltaX := DeltaX / 2;
      NewEstimate[1] := EvaluateFirstDeriv(X,DeltaX);
      TwoToTheIterMinus2 := TwoToTheIterMinus2 * 2;
      FourToTheExtrapMinus1 := 1;
      for Extrap := 2 to Iter do
       begin
        FourToTheExtrapMinus1 := FourToTheExtrapMinus1*4;
        NewEstimate[Extrap] :=
         (FourToTheExtrapMinus1 * NewEstimate[Extrap - 1]
```

```
                - OldEstimate[Extrap - 1])
                   / (FourToTheExtrapMinus1 - 1);
          end;
         OldEstimate := NewEstimate;
       until (ABS(NewEstimate[Iter - 1] - NewEstimate[Iter])
              <=ABS(Tolerance * NewEstimate[Iter])) or
                (ABS(DeltaX) < Tolerance);
       DifF := NewEstimate[Iter];
   end;

   procedure MinMax(LeftEndPoint   : real;
                    RightEndPoint : real;
                var Answer        : real;
                var Error         : byte);
    const NearlyZero = 1E-10;
    var Tol,yLeft,yRight,MidPoint,yMidPoint : real;
        Iter, MaxIter                       : integer;
        Found                               : boolean;

    function TestForRoot(X,OldX,Y,Tol : real) : boolean;
    begin
     TestForRoot := (ABS(Y) <= NearlyZero) or (ABS(X - OldX)
                      < ABS(OldX * Tol));
    end;

   begin
     Maxiter := 500;
     Tol := 1E-10;
     Error := 0;
     Found := false;
     yLeft := DifF(LeftEndpoint);
```

```
yRight := DifF(RightEndpoint);
if ABS(yLeft) <= NearlyZero then
begin
  Answer := LeftEndpoint;
  Found := true;
end;
if ABS(yRight) <= NearlyZero then
begin
  Answer := RightEndpoint;
  Found := true;
end;
if not Found then   { Test for errors }
begin
  if yLeft * yRight > 0 then
    Error := 2;
  if Tol <= 0 then
    Error := 3;
  if MaxIter < 0 then
    Error := 4;
end;
if (Error = 0) and (Found = false) then
begin
 Iter := 0;
 yLeft := DifF(LeftEndPoint);
 while not(Found) and (Iter < MaxIter) do
  begin
   Iter := Succ(Iter);
   MidPoint := (LeftEndpoint + RightEndpoint) / 2;
   yMidPoint := DifF(MidPoint);
   Found  :=  TestForRoot(MidPoint, LeftEndpoint,
                       yMidPoint,Tol);
```

```
    if (yLeft * yMidPoint) < 0 then
       RightEndpoint := MidPoint
    else
      begin
        LeftEndpoint := MidPoint;
        yLeft := yMidPoint;
      end;
   end;
  Answer := MidPoint;
  if Iter >= MaxIter then
    Error := 1;
 end;
end;

procedure Partial_Pivoting(var Dimen          : integer;
                               var Coefficients : matrix;
                               Constants      : RealVector;
                           var Solution       : RealVector;
                           var Error          : byte);
const NearlyZero = 1E-10;
var Sum,Multiplier,Dummy             : real;
    Term,Row,ReferenceRow,PivotRow   : integer;
    DummyRow                         : RealVector;

 procedure EROmultAdd(Multiplier    : real;
                      Dimen         : integer;
                  var ReferenceRow : RealVector;
                  var ChangingRow  : RealVector);
 var Term : integer;
 begin
   for Term := 1 to Dimen do
```

```
    ChangingRow[Term] := ChangingRow[Term]
                          + Multiplier*ReferenceRow[Term];
  end;

begin
  Error := 0;
  if Dimen < 1 then
    Error := 1
  else
    if Dimen = 1 then
      if ABS(Coefficients[1, 1]) < NearlyZero then
        Error := 2
      else
        Solution[1] := Constants[1] / Coefficients[1, 1];
  if Dimen > 1 then
  begin
    { Make Coefficients matrix upper triangular }
    ReferenceRow := 0;
    while (Error = 0) and (ReferenceRow < Dimen - 1) do
    begin
      ReferenceRow := Succ(ReferenceRow);
      { Find row with largest element in this column }
      { and switch this row with the ReferenceRow    }
      PivotRow := ReferenceRow;
      for Row := ReferenceRow + 1 to Dimen do
        if ABS(Coefficients[Row, ReferenceRow]) >
           ABS(Coefficients[PivotRow, ReferenceRow]) then
          PivotRow := Row;
      if PivotRow <> ReferenceRow then
       begin
        DummyRow := Coefficients[PivotRow];
```

```
    Coefficients[PivotRow]:=Coefficients[ReferenceRow];
      Coefficients[ReferenceRow] := DummyRow;
      Dummy := Constants[PivotRow];
      Constants[PivotRow] := Constants[ReferenceRow];
      Constants[ReferenceRow] := Dummy;
    end
  else
{ If the diagonal element is zero, no solution exists }
    if ABS(Coefficients[ReferenceRow, ReferenceRow])
       < NearlyZero then
      Error := 4;        { No solution }
  if Error = 0 then
    for Row := ReferenceRow + 1 to Dimen do
   { Make the ReferenceRow element of these rows zero}
      if ABS(Coefficients[Row, ReferenceRow])
           > NearlyZero then
      begin
       Multiplier := -Coefficients[Row, ReferenceRow]/
                  Coefficients[ReferenceRow,ReferenceRow];
        EROmultAdd(Multiplier, Dimen,
                   Coefficients[ReferenceRow],
                   Coefficients[Row]);
        Constants[Row] := Constants[Row] + Multiplier
                                 * Constants[ReferenceRow];
      end;
 end;
 if ABS(Coefficients[Dimen, Dimen]) < NearlyZero then
   Error := 5;      { No solution }
 if Error = 0 then
 begin
   Term := Dimen;
```

```
      while Term >= 1 do
      begin
        Sum := 0;
        for Row := Term + 1 to Dimen do
          Sum :=Sum+Coefficients[Term, Row]*Solution[Row];
        Solution[Term] := (Constants[Term] - Sum) /
                                    Coefficients[Term, Term];
        Term := Pred(Term);
      end;
    end;
  end;
end;

{************** MAIN ************************}

begin
  clrscr;
  Tol := 1E-10;
  iter := -1;
  IterM := false;
  FillChar(Coefficients, SizeOf(Coefficients), 0);
  FillChar(Constants, SizeOf(Constants), 0);
  Dimen := 10;
  e := 0.0009;
  DO := 0.5;

{ Example #1 ,Example #3 }

  p[1] := 1.3575;  q[1] := 0.6667;
  p[2] := 1.3575;  q[2] := 2.0000;
  p[3] := 1.3575;  q[3] := 3.3333;
```

```
  p[4] := 1.3575;  q[4] := 4.6667;
  p[5] := 1.3575;  q[5] := 6;

{ Example #3 -> 3 section }
{
  p[1] := 2.2189;  q[1] := 1.2;
  p[2] := 2.2189;  q[2] := 3.6;
  p[3] := 2.2189;  q[3] := 6.0;
}
{ Example #2 low pass butterworth }
{
  p[1] := 0.4397;  q[1] := 0.2;
  p[2] := 0.4397;  q[2] := 0.6;
  p[3] := 0.4397;  q[3] := 1.0;
}

loop:
  iter := iter+1;
  gotoxy(25,1);  writeln('Iterate ',iter); writeln;
  for i := 1 to (Dimen shr 1) do
    Writeln('p',i,'=',p[i]:13:10,'q':17,i,'=',q[i]:13:10);
  writeln;

  min := 0;
  max := 0;
  for Row := 1 to Dimen do
   begin
    if odd(Row) then
     begin                   (* Minimum *)
      if Row=1 then
       begin
```

```
    if not IterM then
      w[Row] := 0
    else
     begin
      w[Row] := q[1]/3;
      if DifF(w[Row])>=0 then
        MinMax(0,w[Row],mm,Error)
      else
        MinMax(w[Row],q[1],mm,Error);
      if Error=0 then w[Row] := mm;
     end;
   end
  else
   begin
    w[Row] := (q[(Row shr 1)]+q[(Row shr 1)+1])/2;
    if IterM then
     begin
      if DifF(w[Row])>=0 then
        MinMax(q[(Row shr 1)],w[Row],mm,Error)
      else
        MinMax(w[Row],q[(Row shr 1)+1],mm,Error);
      if Error=0 then w[Row] := mm;
     end;
   end;
 end
else
 begin                (* Maximum *)
  w[Row] := q[Row shr 1];
  if IterM then
   begin
    if DifF(w[Row])<=0 then
```

```
      MinMax((q[(Row shr 1)-1]+q[(Row shr 1)])/2,w[Row]
                 ,mm,Error)
    else
      MinMax(w[Row],(q[(Row shr 1)]+q[(Row shr 1)+1])/2
                       ,mm,Error);
    if Error=0 then w[Row] := mm;
   end;
  end;
 D := EquF(w[Row]);
 if odd(Row) then
  begin
   if (Row=1) and not IterM then
    begin
     Constants[Row] := D0-D;
     if min<abs(D-D0) then  min := abs(D-D0);
    end
   else
    begin
     Constants[Row] := D0-D-e/2;
     if min<abs(D-D0+e/2) then  min := abs(D-D0+e/2);
    end;
  end
 else
  begin
   Constants[Row] := D0-D+e/2;
   if max<abs(D-D0-e/2) then  max := abs(D-D0-e/2);
  end;
 Writeln('w',Row:2,' = ',w[Row],'delta D':15,' = ',
                 Constants[Row]);
end;
 WRITELN;
```

```
 WRITELN('MIN = ',min,'MAX = ':15,max);
if (min<Tol) and (max<Tol) then
begin
 if IterM then
   goto exitd
 else
  begin
   gotoxy(1,1);  write('First[',iter,']');
   IterM := true;
   read;
  end;
end;

for Row := 1 to Dimen do
 begin
  for Column := 1 to Dimen do
   begin
     if Column<=(Dimen shr 1) then
      begin
       i := Column;
       Coefficients[Row, Column] :=
        2*( ( sqr(w[Row]-q[i]) - sqr(p[i]) )
            / sqr( sqr(p[i]) + sqr(w[Row]-q[i]) )
          + ( sqr(w[Row]+q[i]) - sqr(p[i]) )
            / sqr( sqr(p[i]) + sqr(w[Row]+q[i]) ) );
      end
     else
      begin
       i := Column - (Dimen shr 1);
       Coefficients[Row, Column] :=
        4*p[i]*( ( w[Row]-q[i] ) / sqr( sqr(p[i]) +
```

```
                sqr(w[Row]-q[i]) ) - ( w[Row]+q[i] ) /
                sqr( sqr(p[i]) + sqr(w[Row]+q[i]) ) );
           end;
        end;
      end;
     Partial_Pivoting(Dimen,Coefficients,Constants,Solution
                             ,Error);
     if Error=0 then
      begin
       for i := 1 to (Dimen shr 1) do
        begin
          p[i] := p[i]+Solution[i];
          q[i] := q[i]+Solution[(Dimen shr 1)+i];
        end;
      end
     else
      begin
       Writeln;
       Write('***** Error *****');

       WRITELN('ERROR = ':17,Error);
       WRITELN('ABS(Coefficients[Dimen, Dimen]) = '
               ,ABS(Coefficients[Dimen, Dimen]));

       read;
      end;
     goto loop;
    exitd:
   end.
```

กิติกรรมประกาศ

ผู้จัดทำ ขอขอบคุณ อ. ไพศาล สิทธิโยภาสกุล, อ. อรรถสิทธิ์ หล่าสกุล, ผศ. ดร. กนก เจนจิระพงศ์เวช ที่ได้กรุณาให้คำปรึกษาแนะนำทั้งด้านเนื้อหาและการเขียนโปรแกรม และขอขอบคุณเพื่อน ๆ ทุกคนที่ได้ให้ความช่วยเหลือตลอดการทำปริญญานิพนธ์ ซึ่งมิได้นำมากล่าวไว้ในที่นี้

บรรณานุกรม

1. ดร.จเร สุรวัฒน์ปัญญา, การคำนวณเชิงตัวเลขด้วย BASIC, พิมพ์ครั้งที่ 1 ซีเอ็ดยูเคชัน, กรุงเทพมหานคร, 2531
2. ผศ.ดำรงค์ ทิพย์โยธา, ระเบียบวิธีการคำนวณตัวกำหนดและเมตริกซ์, โรง-พิมพ์จุฬาลงกรณ์มหาวิทยาลัย, กรุงเทพมหานคร, 2534
3. บุญเลิศ เอี่ยมทัศนา, เรียนรู้ภาษาปาสคาลด้วยเทอร์โบปาสคาล 4.0-5.0, ซีเอ็ดยูเคชัน, กรุงเทพมหานคร, 2532
4. มนตรี พจนารถลาวัณย์, การเขียนโปรแกรมคอมพิวเตอร์ด้วยเทอร์โบปาสคาล ซีเอ็ดยูเคชัน, กรุงเทพมหานคร, 2533
5. บุญเลิศ เอี่ยมทัศนา, คู่มือเทอร์โบปาสคาล รุ่น 4.0-5.0, ซีเอ็ดยูเคชัน, กรุงเทพมหานคร, 2532
6. มงคล อัศวโกวิทภรณ์, การเขียนโปรแกรมกราฟิคส์, พิมพ์ครั้งที่ 1, ดวงกมลสมัย จำกัด, กรุงเทพมหานคร, 2534
7. Franklin F.kuo, Network Analysis And Synthesis, Second Edition, John Wiley & Sons Inc., Singapore, 1976
8. A.J. Gibbs, The Attainment of T chebycheff Approximations to Prescribed Phase Delay and Group Delay Characteristics with the Aid of an Electronic Computer, Preceedings I.R.E.E., Australia, 1966