บทที่ 2

การแก้ปัญหาออปติไมเซชัน

ปัญหาการทำงานบางประการของระบบไฟฟ้ากำลังนั้นมีความซับซ้อนสูง เนื่องจากในการจ่ายกำลังงานไฟฟ้าจากระบบผลิตไปยังโหลดนั้น ต้องมีการควบคุมระบบให้มีเสถียรภาพ ความมั่นคง ความเชื่อถือได้ รูปแบบของปัญหาที่พบจึงมีลักษณะไม่เป็นเชิงเส้นทำให้มีการนำเทคนิคการแก้ปัญหาแบบไม่เป็นเชิงเส้นมาใช้ วิธีการแก้ปัญหาแบบดั้งเดิมนั้นจะใช้การวิเคราะห์ทางคณิตศาสตร์เป็นหลัก โดยพิจารณาจากสมมติฐานที่ว่าปัญหาต้องจำลองได้ด้วยสมการทางคณิตศาสตร์ที่ชัดเจน มีความต่อเนื่องและสามารถหาอนุพันธ์ได้อย่างน้อยหนึ่งอันดับ ด้วยวิธีการนี้การแก้ปัญหาสามารถทำได้อย่างรวดเร็ว

การแก้ปัญหาออปติไมเซชันนั้นโดยปกติจะแปลงปัญหาให้เป็นปัญหาการหาค่าต่ำสุดหรือสูงสุดโดยแทนปัญหาด้วยฟังก์ชันวัตถุประสงค์ (Objective function) และอาจมีเงื่อนไขบังคับ (Constraints) ในการแก้ปัญหา จากวิธีแบบดั้งเดิมนั้นอาศัยหลักการของอนุพันธ์อันดับที่หนึ่งของฟังก์ชันวัตถุประสงค์เท่ากับศูนย์ ดังนั้นจึงไม่อาจจะรับประกันได้ว่าจุดคำตอบที่ค้นพบนั้นเป็นจุดต่ำสุดโดยรวม (Global minimizer) ประกอบกับการพัฒนาวิธีการตามธรรมชาติเพื่อนำมาใช้ในการแก้ปัญหาที่ให้ผลตอบที่ดีกว่าและความสามารถในการโปรแกรมคอมพิวเตอร์ที่สะดวกรวดเร็วขึ้น ดังนั้นจึงมีการประยุกต์ใช้กระบวนการวิวัฒนาการอย่างแพร่หลาย

2.1 การแก้ปัญหาออปติไมเซชันด้วยวิธีการวิเคราะห์ทางคณิตศาสตร์

ปัญหาออปติไมเซชันปกติเป็นปัญหาเกี่ยวกับการหาค่าต่ำสุดของฟังก์ชันวัตถุประสงค์ f(X) เทียบกับตัวแปรควบคุม $X = [X_1\ X_2 \ldots X_N]^T$ โดยมีรูปแบบของปัญหาเป็นดังนี้

$$\underset{x}{Minimize}\ f(X) \tag{2.1}$$

ภายใต้เงื่อนไขของความเหมาะสม (Optimality conditions) ถ้ากำหนดให้จุดคำตอบที่เหมาะสมของปัญหาแทนได้ด้วย $\dot{X}$ และจากการกระจายฟังก์ชัน f(X) ด้วยอนุกรมของเทย์เลอร์ [1] จะได้สมการดังนี้

$$f(\dot{X}+p)=f(\dot{X})+\nabla f(\dot{X})^T p+\frac{1}{2}p^T\nabla^2 f(\xi)p, \tag{2.2}$$

โดยที่ p ต้องไม่เป็นเวกเตอร์ศูนย์, ξ เป็นค่าระหว่างจุด $X=\dot{X}+p$ และ $\dot{X}$ จากเงื่อนไข First-order necessary condition [1] จะได้ความสัมพันธ์ ดังสมการ

$$\nabla f(\dot{X})=0 \tag{2.3}$$

ตามสมการที่ 2.3 นี้เป็นเงื่อนไขที่จำเป็นในการบ่งชี้ว่าจุดที่พิจารณาเป็นจุดต่ำสุด แต่จุดที่สอดคล้องกับเงื่อนไขนี้ไม่ได้มีเฉพาะจุดต่ำสุดเท่านั้น ดังนั้นเงื่อนไขนี้จึงยังไม่เพียงพอที่จะระบุจุดต่ำสุดได้ ในการบ่งชี้จุดต่ำสุดอย่างชัดเจนนั้นต้องการเงื่อนไขที่เพียงพอ Second-order sufficient condition [1] ประกอบการพิจารณา ดังนี้

เมื่อแทนสมการที่ 2.3 ลงในสมการที่ 2.2 จะได้สมการ

$$f(X)=f(\dot{X}+p)=f(\dot{X})+\frac{1}{2}p^T\nabla^2 f(\xi)p \tag{2.4}$$

จากคุณสมบัติของจุดต่ำสุด $f(\dot{X}+p)>f(\dot{X})$ ทำให้พจน์ที่สองทางด้านขวามือในสมการที่ 2.4 ต้องมีค่ามากกว่าศูนย์จึงจะสอดคล้องกับเงื่อนไขนี้ นั่นคือ ถ้า $\nabla^2 f(\dot{X})$ เป็น psdf (Positive semi-definite matrix) แล้วจะทำให้ $v^T\nabla^2 f(\dot{X})v \geq 0$ *สำหรับบางค่าของ* v และจะเป็นจริงสำหรับ $v^T\nabla^2 f(\xi)v \geq 0$ ด้วย ถ้า $\left\|\xi-\dot{x}\right\|$ มีค่าน้อย ๆ

ดังนั้นถ้าเลือก p ให้เหมาะสมแล้ว ค่า ξ จะมีค่าเข้าใกล้ $\dot{X}$ มากเพียงพอที่จะรับประกันได้ว่า $f(\dot{X})<f(X)$ ซึ่งเงื่อนไขนี้เรียกว่า Second-order sufficient condition

2.1.1 การแก้ปัญหาออปติไมเซชันของฟังก์ชันไม่เป็นเชิงเส้นโดยไม่มีเงื่อนไขบังคับ

ปัญหาออปติไมเซชันโดยไม่มีเงื่อนไขบังคับเป็นการหาค่าต่ำสุดของฟังก์ชันวัตถุประสงค์เทียบกับตัวแปรควบคุมใด ๆ วิธีการที่ใช้แก้ปัญหาออปติไมเซชันโดยไม่มีเงื่อนไขอาศัยหลักของแคลคูลัสแบบหลายตัวแปร ดังนี้

ถ้ากำหนดปัญหาออปติไมเซชันแบบไม่มีเงื่อนไขตามสมการที่ 2.1 เงื่อนไขที่จำเป็นในการแก้ปัญหาคือ First-order necessary condition ดังสมการ 2.5 และ 2.6 ซึ่งจะเรียกว่า Gradient vector

$$\frac{\partial f}{\partial X_i} = 0 \qquad ; i = 1,2,...,N \tag{2.5}$$

หรือ

$$\nabla f = (\frac{\partial f}{\partial X_1}, \frac{\partial f}{\partial X_2}, ..., \frac{\partial f}{\partial X_N}) = 0 \tag{2.6}$$

ในการแก้ปัญหาจะมีการนำเอาวิธีการค้นหา (Search method) มาใช้เพื่อวิเคราะห์ในกรณีที่ความไม่เป็นเชิงเส้นของฟังก์ชันวัตถุประสงค์มีค่าสูง โดยในที่นี้จะนำเสนอ Gradient search และ Newton's search method ตามลำดับ ดังนี้

ตามวิธีการของ Gradient Search นั้นเมื่อกำหนดค่าเริ่มต้นให้กับตัวแปรควบคุม X จากนั้นทำการคำนวณค่า Gradient ของฟังก์ชันตามสมการที่ 2.6 เพื่อนำมาสร้างจุดคำตอบในรอบการคำนวณครั้งต่อไปตามสมการที่ 2.7

$$X_{k+1} = X_k + \alpha \nabla f(X_k) \tag{2.7}$$

ค่า α คำนวณได้จากการทำ Line search subprogram ที่มีรูปแบบปัญหาดังนี้

$$\underset{\alpha>0}{\text{Minimize}}\ F(\alpha) \equiv f(X_k + \alpha \nabla f(X_k)) \tag{2.8}$$

วิธีการ Gradient search นี้ เลือก search direction หรือ เวกเตอร์ p ตามสมการที่ 2.9

$$p_k = \nabla f(X_k) \tag{2.9}$$

ดังนั้น ในสมการที่ 2.8 เป็นการแก้ปัญหาย่อยเพื่อหาช่วงก้าวที่เหมาะสม (Step length) α ซึ่งในหลาย ๆ กรณี กระบวนการนี้อาจใช้เวลาในการคำนวณมากเกินความจำเป็น ทำให้การแก้ปัญหาไม่มีประสิทธิภาพเท่าที่ควร ทั้งนี้จุดมุ่งหมายของการหาช่วงก้าวที่เหมาะสมนี้เพื่อรับประกันว่าในทุก ๆ รอบของการคำนวณตามกระบวนการ Search ที่ใช้ นั้น ค่าของฟังก์ชันวัตถุประสงค์จะมีค่าลดลง

ในส่วนของ Newton's search method นั้น จะใช้การวิเคราะห์จากการกระจายอนุกรมของเทย์เลอร์เป็นหลัก แต่จะใช้การประมาณถึงพจน์ของอนุพันธ์อันดับที่สองเท่านั้น ดังสมการที่ 2.10

$$f(X_k+p_k)\cong f(X_k)+\nabla f(X_k)^T p_k+\frac{1}{2}p_k^T\nabla^2 f(X_k)p_k \qquad (2.10)$$

คำนวณค่า Gradient ของสมการที่ 2.10 เทียบกับ p_k จะได้ความสัมพันธ์ตามสมการที่ 2.11 ดังนี้

$$\nabla f(X_k+p_k)=\nabla f(X_k)+\nabla^2 f(X_k)p_k \qquad (2.11)$$

โดยวิธีการนี้คาดหวังว่าจุดคำตอบต่อไปที่ค้นพบคือจุดที่สอดคล้องกับเงื่อนไข $\nabla f(X_{k+1})=0$ ดังนั้นจะได้สมการดังนี้

$$\nabla f(X_k)+\nabla^2 f(X_k)p_k=0 \qquad (2.12)$$

$$p_k=-\left[\nabla^2 f(X_k)\right]^{-1}\nabla f(X_k) \qquad (2.13)$$

ปกติวิธีนี้ใช้ค่า $\alpha = 1$ และเป็นวิธีที่มีอัตราการลู่เข้าที่รวดเร็วมาก แต่มีข้อเสียคือ อาจประสบปัญหาในการลู่เข้าของคำตอบ ดังนั้นจึงมีการปรับปรุงเพื่อรับประกันการลู่เข้า โดยพิจารณาจากสมการที่ 2.10 จะพบว่าค่าของฟังก์ชันวัตถุประสงค์จะมีค่าลดลงในรอบการคำนวณถัดไปเมื่อ

$$p_k^T\nabla f(X_k)<0 \qquad (2.14)$$

แทนค่า p_k จากสมการที่ 2.13 ในสมการที่ 2.14 จะได้ดังนี้

$$\begin{aligned} &p_k^T\nabla f(X_k)=-\nabla f(X_k)^T\left[\nabla^2 f(X_k)\right]^{-1}\nabla f(X_k)<0 \text{ หรือ}\\ &\nabla f(X_k)^T\left[\nabla^2 f(X_k)\right]^{-1}\nabla f(X_k)>0 \end{aligned} \qquad (2.15)$$

เพื่อให้สอดคล้องกับสมการที่ 2.15 ดังนั้น $\left[\nabla^2 f(X_k)\right]^{-1}$ ต้องเป็น pdf (Positive definite matrix) โดยเราจะเรียก $\left[\nabla^2 f(X_k)\right]^{-1}$ ว่าเป็น Hessian matrix ในกรณีที่คำนวณ Hessian matrix ได้ผลออกมาเป็น indefinite จะทำให้กระบวนการไม่ลู่เข้า ดังนั้น ต้องปรับปรุงโดยทำการประมาณค่า Hessian matrix ด้วย pdf matrix ที่เหมาะสมดังต่อไปนี้

ถ้า $\nabla^2 f(X_k)$ เป็น pdf สามารถแยกตัวประกอบของเมตริกซ์ได้ดังนี้

$$\nabla^2 f(X_k) = LDL^T \tag{2.16}$$

โดยที่ D แทน Diagonal matrix และ $\nabla^2 f(X_k)$ จะไม่เป็น pdf เมื่อ $d_{ii} \leq 0$ ซึ่งเมื่อเกิดเหตุการณ์นี้ขึ้นต้องแทนค่า d_{ii} ด้วยค่าที่เป็นบวกบางค่า อาจใช้ $|d_{ii}|$ หรือ ค่าที่เป็นบวกเล็ก ๆ บางค่าก็ได้ หรือใช้การบวกด้วย Diagonal matrix E เข้ากับค่าเดิมดังสมการที่ 2.17

$$\nabla^2 f(X_k) + E = LDL^T \tag{2.17}$$

เมื่อกำหนด Diagonal matrix ดังนี้

$$D = \begin{bmatrix} d_{11} & 0 & \cdots & 0 \\ 0 & d_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_{MM} \end{bmatrix}$$

2.1.2 การแก้ปัญหาออปติไมเซชันของฟังก์ชันไม่เป็นเชิงเส้นโดยมีเงื่อนไขบังคับสมการ

ปัญหาออปติไมเซชันของฟังก์ชันไม่เป็นเชิงเส้นโดยมีเงื่อนไขบังคับสมการนี้ เกิดขึ้นเมื่อตัวแปรควบคุมที่ใช้ไม่สามารถเปลี่ยนแปลงค่าได้อย่างอิสระ เช่น ถ้าต้องการจัดสรรกำลังผลิตของเครื่องกำเนิดไฟฟ้าจำนวน 3 เครื่อง เพื่อจ่ายให้โหลดขนาด 300 MW โดยเครื่องกำเนิดไฟฟ้าแต่ละเครื่องมีฟังก์ชันค่าเชื้อเพลิงเป็น $F_1(P_{G1})$, $F_2(P_{G2})$ และ $F_3(P_{G3})$ ตามลำดับ โดยเป้าหมายของการจัดสรรกำลังผลิตนั้นเพื่อทำให้ต้นทุนการผลิตต่ำที่สุด ดังนั้นฟังก์ชันวัตถุประสงค์จึงมีค่าเท่ากับผลรวมของฟังก์ชันค่าเชื้อเพลิงทั้งสามฟังก์ชัน แต่มีเงื่อนไขบังคับสมการ โดยกำลังผลิตทั้งหมดต้องเท่ากับโหลด ดังนี้

$$\begin{aligned} &\text{Minimize} \quad F_T = F_1(P_{G1}) + F_2(P_{G2}) + F_3(P_{G3}) \\ &\text{Subject to} \quad P_{G1} + P_{G2} + P_{G3} = 300 \end{aligned} \tag{2.18}$$

การแก้ปัญหาในรูปแบบนี้นั้น มีขั้นตอนบางอย่างที่ต้องเพิ่มเข้ามา และเพื่อให้การแก้ปัญหาอยู่ในรูปแบบทั่วไป ในที่นี้ได้กำหนดรูปแบบปัญหาออปติไมเซชันของฟังก์ชันไม่เป็นเชิงเส้นโดยมีเงื่อนไขบังคับสมการเป็นดังสมการที่ 2.19

$$\begin{aligned} &\text{Minimize} \quad f(X) \\ &\text{Subject to} \quad g_i(X) = 0 \quad ; i = 1,2,\ldots Eq \end{aligned} \tag{2.19}$$

เมื่อ Eq คือ จำนวนเงื่อนไขบังคับสมการ, $g_i(X)$ คือ เงื่อนไขบังคับสมการ

วิธีการแก้ปัญหาที่ใช้กันอย่างแพร่หลายก็คือ การแปลงปัญหาออปติไมเซชันแบบมีเงื่อนไขให้เป็นปัญหาแบบไม่มีเงื่อนไขโดยใช้ฟังก์ชันของลากรองซ์ (Lagrange) โดยที่เงื่อนไขบังคับสมการต้องมีตัวคูณของลากรองซ์คูณอยู่ แล้วนำไปบวกเข้ากับฟังก์ชันวัตถุประสงค์ดังสมการที่ 2.20

$$L = f(X) + \sum_{i=1}^{Eq} \lambda_i g_i(X) \tag{2.20}$$

จากสมการที่ 2.20 นี้ ปัญหาจะอยู่ในรูปแบบที่ไม่มีเงื่อนไข แต่จำนวนตัวแปรจะเพิ่มเข้ามาเท่ากับจำนวนของเงื่อนไขบังคับแบบสมการ นั่นคือ จำนวนของตัวแปร X บวกด้วย จำนวนของ λ และจาก Optimality conditions จะได้ความสัมพันธ์ตามสมการต่อไปนี้

$$\frac{\partial L(X,\lambda)}{\partial X_i} = \frac{\partial f(X)}{\partial X_i} + \sum_{i=1}^{Eq} \lambda_i \frac{\partial g_i(X)}{\partial X_i} \tag{2.21}$$

$$\frac{\partial L(X,\lambda)}{\partial \lambda_i} = g_i(X) = 0 \tag{2.22}$$

2.1.3 การแก้ปัญหาออปติไมเซชันของฟังก์ชันไม่เป็นเชิงเส้นโดยมีเงื่อนไขบังคับอสมการ

ปัญหาในลักษณะนี้นั้น โดยปกติเงื่อนไขบังคับอสมการจะอยู่ในรูปขอบเขตของตัวแปรควบคุม เช่น ขอบเขตของกำลังผลิตจากเครื่องกำเนิดไฟฟ้าแต่ละตัว เป็นต้น รูปแบบปัญหาออปติไมเซชันของฟังก์ชันไม่เป็นเชิงเส้นโดยมีเงื่อนไขบังคับอสมการเป็นดังสมการที่ 2.23

$$\begin{aligned} &\text{Minimize} \quad f(X) \\ &\text{Subject to} \quad g_i(X) = 0 \quad ; i = 1,2,\ldots Eq \\ &\qquad\qquad h_j(X) \le 0 \quad ; j = 1,2,\ldots, Iq \end{aligned} \tag{2.23}$$

เมื่อ Iq คือ จำนวนเงื่อนไขบังคับอสมการ, $h_j(X)$ คือ เงื่อนไขบังคับอสมการ

ฟังก์ชันของลากรองซ์ที่ได้จากการแก้ปัญหานี้มีค่าดังสมการที่ 2.24

$$L = f(X) + \sum_{i=1}^{Eq} \lambda_i g_i(X) + \sum_{j=1}^{Iq} \mu_j h_j(X) \tag{2.24}$$

จากสมการที่ 2.24 นี้ ปัญหาจะอยู่ในรูปแบบที่ไม่มีเงื่อนไข แต่จำนวนตัวแปรจะเพิ่มเข้ามาเท่ากับจำนวนของเงื่อนไขบังคับแบบสมการรวมกับจำนวนของเงื่อนไขบังคับแบบอสมการ นั่นคือ จำนวนของตัวแปร X บวกด้วย จำนวนของ λ บวกด้วย จำนวนของ μ และจาก Optimality conditions จะได้ความสัมพันธ์ตามสมการต่อไปนี้

$$\frac{\partial L(X,\lambda)}{\partial X_k} = \frac{\partial f(X)}{\partial X_k} + \sum_{i=1}^{Eq} \lambda_i \frac{\partial g_i(X)}{\partial X_k} + \sum_{j=1}^{Iq} \mu_j \frac{\partial h_j(X)}{\partial X_k} \tag{2.25}$$

$$\frac{\partial L(X,\lambda)}{\partial \lambda_i} = g_i(X) = 0 \tag{2.26}$$

$$\frac{\partial L(X,\lambda)}{\partial X_k} = h_j(X) \le 0 \tag{2.27}$$

$$\mu_j h_j(X) = 0 \;\&\; \mu_j \ge 0 \tag{2.28}$$

จากสมการ 2.27 เงื่อนไขบังคับแบบอสมการจะ inactive เมื่อ ผลของสมการ 2.27 มีค่าไม่เท่ากับศูนย์ และจะทำให้ $\mu_j = 0$ แต่ถ้าสมการที่ 2.27 เท่ากับศูนย์เงื่อนไขบังคับอสมการจะ active และ $\mu_j \ge 0$

2.1.4 Sequential quadratic programming (SQP)

การแก้ปัญหาออปติไมเซชันแบบมีเงื่อนไขบังคับนั้น สามารถแปลงปัญหาให้อยู่ในรูปปัญหาออปติไมเซชันแบบไม่มีเงื่อนไขบังคับได้ โดยใช้ฟังก์ชันของลากรองซ์ และจากการประยุกต์สมการของ Kuhn-Tucker [1] ในการแก้ปัญหา SQP เป็นอัลกอริทึมที่มีประสิทธิภาพและถูกนำมาใช้แก้ปัญหาออปติไมเซชันของฟังก์ชันไม่เป็นเชิงเส้นอย่างแพร่หลายในปัจจุบัน วิธีการนี้เป็นการที่คล้ายคลึงวิธีของนิวตัน (Newton' method) โดยอาศัยหลักการปรับปรุงค่า Hessian matrix ที่เกิดจากฟังก์ชันของลากรองซ์ด้วยวิธี Quasi-Newton [1] และการแก้ปัญหาย่อยแบบ Quadratic (Quadratic programming sub-problem) ในแต่ละรอบของการคำนวณ ดังนั้นจึงนิยมเรียกวิธีนี้ว่า Sequential Quadratic Programming, Iterative Quadratic Programming, Recursive Quadratic

Programming, Successive Quadratic Programming, หรือ Constrained Variable Metric methods [2] โดยรายละเอียดของวิธีการนี้สามารถสรุปได้ดังต่อไปนี้

ถ้ากำหนดรูปแบบของปัญหาดังนี้

Minimize	$f(X)$		(2.29)
Subject to	$g_i(X) = 0$	$i = 1,2,\ldots,Eq$	(2.30)
	$h_j(X) \le 0$	$j = 1,2,\ldots,Iq$	(2.31)
	$X^{min} \le X \le X^{max}$		(2.32)

โดยที่ X เป็นเวกเตอร์ของตัวแปรควบคุม $X \in \Re^n$

$f(X)$ เป็นฟังก์ชันวัตถุประสงค์ค่าจริง $f(X): \Re^n \rightarrow \Re$

$g_i(X)$ เป็นเวกเตอร์ที่ให้ค่าฟังก์ชันของเงื่อนไขบังคับสมการ

$g_i(X): \Re^n \rightarrow \Re^{Eq}$

$h_j(X)$ เป็นเวกเตอร์ที่ให้ค่าฟังก์ชันของเงื่อนไขบังคับอสมการ

$h_j(X): \Re^n \rightarrow \Re^{Iq}$

X^{min}, X^{max} แทนค่าต่ำสุดและค่าสูงสุดของตัวแปรควบคุมตามลำดับ

Eq, Iq แทนจำนวนของเงื่อนไขบังคับสมการและอสมการตามลำดับ

ในแต่ละรอบของการแก้ปัญหาจะใช้การแก้ปัญหาย่อย QP (QP subproblem) ซึ่งมีรูปแบบการแก้ปัญหาโดยการพิจารณาเงื่อนบังคับแบบไม่เป็นเชิงเส้นให้เป็นเชิงเส้น (Linearizing the nonlinear constraints) จะได้รูปแบบปัญหาดังนี้

$$\underset{d \in \Re^n}{Minimize}\ \frac{1}{2} d^T H_k d + \nabla f(x_k)^T d \tag{2.33}$$

$$\nabla g_i(x_k)^T d + g_i(x_k) = 0 \quad ; i = 1,2,\ldots,Eq \tag{2.34}$$

$$\nabla h_j(x_k)^T d + h_j(x_k) \le 0 \quad ; j = 1,2,\ldots,Iq \tag{2.35}$$

การแก้ปัญหาย่อยนี้ใช้การสร้างจุดคำตอบใหม่โดยเพื่อใช้ในรอบการคำนวณถัดไปดังนี้ $x_{k+1} = x_k + \alpha_k d_k$ โดย α_k แทนช่วงก้าวที่เหมาะสม (step length) ที่คำนวณได้จากกระบวนการ Line search เพื่อรับประกันว่าในรอบถัดไปค่าของ Merit function [2] จะต้องมีค่าลดลงอย่างเพียงพอ H_k เป็น positive definite matrix แทนการประมาณค่า Hessian matrix จากฟังก์ชันของลากรองซ์ และมีการปรับปรุงตามวิธีการ Quasi-Newton ในทุก ๆ รอบของการคำนวณ

ในการปรับปรุง Hessian matrix เพื่อให้ได้ positive definite matrix สามารถคำนวณได้ดังนี้

$$
\begin{aligned}
H_{k+1} &= H_k + \frac{q_k q_k^T}{q_k^T s_k} - \frac{H_k^T H_k}{s_k^T H_k s_k} \\
s_k &= x_{k+1} - x_k \\
q_k &= \nabla f(x_{k+1}) + \sum_{i=1}^{Eq} \lambda_i \nabla g_i(x_{k+1}) + \sum_{j=1}^{Iq} \mu_j \nabla h_j(x_{k+1}) \\
&\quad - \left(\nabla f(x_k) + \sum_{i=1}^{Eq} \lambda_i \nabla g_i(x_k) + \sum_{j=1}^{Iq} \mu_j \nabla h_j(x_k) \right)
\end{aligned}
\tag{2.36}
$$

ในการหาช่วงก้าวที่เหมาะสมนั้น α_k ต้องไม่มีค่าเป็นลบและทำให้ค่าของ Merit function $\psi(x)$ ลดลงในแต่ละรอบการคำนวณ ดังนี้

$$
\psi(x) = f(x) + \sum_{i=1}^{Eq} r_i * g_i(x) + \sum_{j=1}^{Iq} r_j * \max\{0, h_j(x)\} \tag{2.37}
$$

โดยที่ r_i แทนค่าสัมประสิทธิ์การปรับโทษคำนวณได้ดังนี้

$$
(r_{k+1})_i = \underset{i}{Max}\left\{ \lambda_i, \frac{1}{2}((r_k)_i + \lambda_i) \right\} \qquad ; i = 1,2,\ldots,Eq+Iq \tag{2.38}
$$

การคำนวณค่าสัมประสิทธิ์การปรับโทษในขณะเริ่มต้น คำนวณได้ตามสมการต่อไปนี้

$$
r_i = \frac{\|f(X)\|}{\|g_i(X)\|} \quad \text{หรือ} \quad r_j = \frac{\|f(X)\|}{\|h_j(X)\|} \tag{2.39}
$$

2.2 การแก้ปัญหาออปติไมเซชันด้วยวิธีการคำนวณเชิงวิวัฒนาการ

ปัญญาประดิษฐ์(Artificial intelligence) [3,4] เป็นกระบวนการแก้ปัญหาโดยอาศัยการจำลองผลอย่างชาญฉลาด ซึ่งปกติจะเลียนแบบพฤติกรรมการแก้ปัญหาของมนุษย์เป็นส่วนใหญ่ เช่น ระบบผู้เชี่ยวชาญ(Expert system) ระบบฟัซซี่(Fuzzy system) ระบบโครงข่ายประสาทเทียม (Artificial neural network) เจนเนติกอัลกอริทึม(Genetic algorithms) การโปรแกรมเชิงวิวัฒนาการ (Evolutionary programming) และ Simulated annealing(SA) เป็นต้น ความฉลาดเป็นคุณสมบัติที่โดดเด่นของมนุษย์ สามารถทำให้เกิดการเรียนรู้ การทำความเข้าใจ และการค้นหาวิธีแก้ปัญหา การคำนวณเชิงวิวัฒนาการ(Evolutionary computation) จัดเป็นปัญญาประดิษฐ์ชนิดหนึ่ง ซึ่งประกอบด้วย เจนเนติกอัลกอริทึม การโปรแกรมเชิงวิวัฒนาการ และกลยุทธ์วิวัฒนาการ(Evolution strategies) กระบวนการนี้เป็นการเลียนแบบวิวัฒนาการของสิ่งมีชีวิตตามธรรมชาติโดยอาศัยกฎการคัดเลือกตามธรรมชาติเป็นหลัก ในแต่ละรุ่นของการถ่ายทอดผู้ถูกคัดเลือกจะถ่ายทอดและปรับปรุงคุณสมบัติไปสู่รุ่นถัดไปเพื่อสร้างสมาชิกที่ดีที่สุดต่อไป

2.2.1 กระบวนการวิวัฒนาการตามธรรมชาติ (Natural evolution)

วิวัฒนาการของสิ่งมีชีวิตสามารถอธิบายได้ด้วยกระบวนการทางสถิติที่กระทำกันระหว่างสมาชิกภายในกลุ่มประชากรเดียวกัน ประกอบด้วยกระบวนการรีโปรดักชัน(Reproduction), มิวเตชัน(Mutation), การแข่งขัน(Competition) และการคัดเลือก(Selection) องค์ประกอบของสิ่งมีชีวิตสามารถพิจารณาในลักษณะของฟีโนไทป์และจีโนไทป์ได้ ซึ่งแสดงพฤติกรรมที่ปรากฏออกมาและลักษณะของสิ่งมีชีวิตในระดับยีน

ถ้าจำลองคุณลักษณะของจีโนไทป์และฟีโนไทป์เป็นปริภูมิสถานะ(State spaces) P และ G ตามลำดับ และกำหนดฟังก์ชันการถ่ายโอน 4 ฟังก์ชันนิยามดังนี้

$$f_1: I \times G \rightarrow P,$$
$$f_2: P \rightarrow P,$$
$$f_3: P \rightarrow G,$$
$$f_4: G \rightarrow G.$$

ฟังก์ชัน f_1 เป็นการถ่ายโอนสมาชิก $g_1 \in G$ ไปยัง $p_1 \in P$ โดยกระบวนการนี้จะถูกกระทบจากสิ่งแวดล้อม ซึ่งแทนด้วย I ฟังก์ชัน f_2 (การคัดเลือก) เป็นการถ่ายโอนสมาชิก p_1 ไปยัง p_2 ใน

ปริภูมิเดียวกัน ฟังก์ชัน f_3 เป็นผลกระทบจากฟังก์ชัน f_2 ต่อปริภูมิ G ฟังก์ชัน f_4 (มิวเตชันและรีโปรดักชัน) เป็นการถ่ายโอน $g_2 \in G$ ไปยังจุด $g'_3 \in G$ ซึ่งถือว่าเป็นการเปลี่ยนแปลงทางยีนทำให้เกิดสมาชิกใหม่ในกลุ่มประชากร กระบวนการที่มีลำดับตามฟังก์ชันทั้งสี่ฟังก์ชันนี้ถือว่าเป็นกระบวนการวิวัฒนาการตามธรรมชาติที่เกิดขึ้นหนึ่งรุ่น

กระบวนการคัดเลือกตามธรรมชาติเป็นการบังคับในทางวิวัฒนาการเพื่อให้สมาชิกในกลุ่มประชากรมีพฤติกรรมที่เหมาะสมในการอยู่รอดในสภาวะแวดล้อมที่บังคับ(Environmental constraints) ดังนั้นกระบวนการคัดเลือกนี้จึงนำไปสู่การประเมินความอยู่รอดของสมาชิกในกลุ่มประชากรที่เรียกว่า "ความเหมาะสม(Fitness)" ซึ่งค่าความเหมาะสมนี้เป็นการแสดงถึงความสามารถในการอยู่รอดและการแพร่พันธุ์ของสมาชิกในสภาพแวดล้อมที่กำหนดและไม่สามารถที่จะวัดค่าได้โดยตรง แต่จะขึ้นอยู่กับสภาวะแวดล้อม ลักษณะนิเวศวิทยา และโครงสร้างของตัวสมาชิกในกลุ่มประชากรเอง

กระบวนการวิวัฒนาการของสิ่งมีชีวิตเป็นการปรับปรุงและพัฒนาพฤติกรรมของสมาชิกในกลุ่มประชากรที่เหมาะสมให้เกิดการถ่ายทอดสู่รุ่นถูกหลาน ทำให้เกิดสมาชิกที่มีความเหมาะสมที่สุดที่จะอยู่รอดในสภาวะแวดล้อมเหล่านั้นได้ จากแนวคิดนี้ได้มีการเลียนแบบและจำลองเอากระบวนการวิวัฒนาการนี้มาใช้แก้ปัญหาทางวิศวกรรม เพื่อสร้างระบบที่สามารถปรับปรุงและพัฒนาผลเฉลยตามกระบวนการที่กล่าวมาข้างต้นเพื่อให้ได้ผลเฉลยที่เหมาะสมที่สุดในการแก้ปัญหาต่อไป

2.2.2 การทำ ออปติไมเซชันโดยใช้กระบวนการวิวัฒนาการ (Evolutionary optimization)

กระบวนการวิวัฒนาการเป็นการสร้างสิ่งมีชีวิตหรือสมาชิกในกลุ่มประชากรให้มีความซับซ้อนมากยิ่งขึ้นโดยได้รับอิทธิพลมาจากสิ่งแวดล้อมด้วยความน่าจะเป็นที่แตกต่างกัน ในการคัดเลือกตามธรรมชาติจะมีสมาชิกในกลุ่มประชากรจำนวนหนึ่งที่มีความเหมาะสมที่สุดเท่านั้นที่สามารถอยู่รอดเพื่อถ่ายทอดคุณลักษณะไปสู่รุ่นถัดไปได้

เบรเมอร์แมน (Bremermann 1958) [3] ได้ทำการทดลองเพื่อนำไปสู่ข้อสรุปที่ว่า หลักการวิวัฒนาการเป็นวิธีการที่มีประสิทธิภาพในการทำความเข้าใจ การเรียนรู้ของปัญหาเพื่อค้นหาวิธีการแก้ปัญหาที่ดีที่สุดต่อไป ซึ่งเป็นกระบวนในการทำออปติไมเซชันนั่นเอง การทดลองใช้การเข้ารหัสเลขฐานสองในการจำลองโครงสร้างของยีนส์เพื่อแทนสมาชิกในกลุ่มประชากรและทำการทดสอบค่าความเหมาะสมของสมาชิกเพื่อนำไปใช้ในกระบวนการคัดเลือกสมาชิกที่อยู่รอดต่อไป จากการทดลองพบว่ากระบวนการเกิดมิวเตชัน (Mutation) ที่เหมาะสมนั้นต้องให้มีการเปลี่ยนแปลงเพียงหนึ่งบิทต่อสมาชิกหนึ่งตัวในแต่ละรุ่นเท่านั้น

ในปี ค.ศ. 1962 เบรเมอร์แมนทำการทดลองเพื่อขยายผลจากเดิมโดยพิจารณาปัญหาการหาค่าต่ำสุดของฟังก์ชันค่าจริง $f(x_1,x_2,\ldots,x_n)$ โดยที่ $x_i \in R$ การทดลองเลือกใช้ฟังก์ชันเชิงเส้นเพื่อให้สะดวกต่อการวิเคราะห์กระบวนการวิวัฒนาการ โดยปัญหาที่พิจารณาอยู่ในรูป Minimize $\| Ax-b \|_2$ ทั้งนี้หากใช้เฉพาะการเกิดมิวเตชันแล้ว กระบวนการวิวัฒนาการจะสามารถนำไปสู่จุดต่ำสุดที่ต้องการได้อย่างรวดเร็ว นอกจากนี้เบรเมอร์แมนได้แนะนำว่าในกระบวนการเกิดมิวเตชันอาจใช้ความน่าจะเป็นร่วมในการพิจารณาได้ แต่จะทำให้กระบวนการค้นหาจุดต่ำสุดนั้นช้าลง ต่อมาเบรเมอร์แมนและรอจสัน (Bremermann and Rogson) [3] ได้ทำการทดลองโดยการใช้การแปลงแบบอื่นในการสร้างสมาชิกใหม่ของกลุ่มประชากร เช่น การครอสโอเวอร์ (Crossover) แต่ผลที่ได้จากการทดลองไม่ดีกว่าการทดลองเดิม

ในปี ค.ศ. 1967 บาร์ริเซลลี (Barricelli) [3] ได้ทำการทดลองกระบวนการวิวัฒนาการโดยใช้มิวเตชันและครอสโอเวอร์เพื่อเปรียบเทียบกัน ผลที่ได้พบว่าการทำครอสโอเวอร์นั้นไม่ได้ทำให้ความเร็วของการทำออปติไมเซชันเพิ่มขึ้น

2.2.3 การโปรแกรมเชิงวิวัฒนาการ (Evolutionary programming)

ฟอเจล (Fogel) [3] ได้พัฒนาปัญญาประดิษฐ์โดยใช้การจำลองกระบวนการวิวัฒนาการที่เรียกว่า การโปรแกรมเชิงวิวัฒนาการ ที่มีพื้นฐานมาจากการสร้างระบบที่มีพฤติกรรมที่ฉลาด โดยสามารถประเมินและทำนายสภาวะแวดล้อม ตลอดจนการให้การตอบสนองที่เหมาะสม ในปี ค.ศ. 1962-1965 ฟอเจลได้เริ่มต้นทดลองกับระบบที่มีลักษณะไม่ต่อเนื่องที่เรียกว่า Finite state machines และได้พัฒนาอย่างต่อเนื่องจนกระทั่งปี ค.ศ. 1991 ฟอเจลได้เสนอโครงร่างของการโปรแกรมเชิงวิวัฒนาการดังต่อไปนี้ เริ่มต้นด้วยการสุ่มค่าประชากรเริ่มต้นของกลุ่มประชากรพร้อมทั้งคำนวณค่าความเหมาะสมตามหลักความน่าจะเป็น แล้วทำการสร้าง Offspring แบบสุ่มโดยใช้การทำมิวเตชันของประชากรเริ่มต้น ใช้ค่าความเหมาะสมในการคัดเลือกสมาชิกที่จะอยู่รอดในรุ่นถัดไป กระบวนการทำออปติไมเซชันโดยการโปรแกรมเชิงวิวัฒนาการมีขั้นตอนดังนี้

1) *Vector representation* : ในการทำ Optimization ของฟังก์ชันค่าจริง โดยกำหนดฟังก์ชันวัตถุประสงค์เป็น $f(\bar{p}) : \Re^n \rightarrow \Re$ โดยเวกเตอร์ $\bar{p}$ เป็นสมาชิกของกลุ่มประชากร (Population) ที่จะถูกพัฒนาต่อไป
2) *Initialization* : ประชากรเริ่มต้น p_i ; i=1,2,...,N_p ถูกเลือกแบบสุ่มจาก Feasible ranges โดยใช้การสุ่มแบบ Uniform random
3) *Creation of offspring* : p_i' จะถูกสร้างจาก p_i โดยที่ $p'_{i,j} = p_{i,j} + N(0,\sigma_j^2)$ ซึ่งจะเรียกการดำเนินการนี้ว่า Mutation โดยที่ $\sigma_j = \beta \frac{f_{pi}}{f_{\min}}(p_{j,\max} - p_{j,\min})$

4) *Competition & Selection* : สร้าง Competing pool จาก p_i และ p_i' สมาชิกทั้งหมดใน Competing pool จะมีจำนวนเป็น 2 เท่าของประชากรเริ่มต้น จากนั้นจะทำการเลือกสมาชิกที่มีค่า fitness score สูงสุดหรือต่ำสุด(ขึ้นอยู่กับว่าเป็นการหาค่าสูงสุดหรือต่ำสุดของฟังก์ชัน) จำนวน N_p ให้อยู่รอดเพื่อใช้เป็นประชากรเริ่มต้นใน generation ถัดไป

5) *Stopping rule* : กระบวนการในการสร้าง generation ใหม่นั้น จะดำเนินการไปจนกระทั่งไม่สามารถที่จะลดหรือเพิ่มค่า fitness score ได้อีกแล้ว หรือเมื่อจำนวน generation ถึงค่าสูงสุดที่ตั้งไว้

การคำนวณค่า fitness score (W_p) คำนวณได้ ดังนี้

สุ่มค่าตัวแปร u_1,u_2 โดยการสุ่มแบบ Uniform random โดยที่ $u_1,u_2 \in [0,1]$ คำนวณค่า r จากสมการ $r = [2N_pu_2+1]$

$$W_{P_i} = \sum_{t=1}^{N_P} w_t$$

โดยที่ $W_t = 1$, ถ้า $u_1 < \dfrac{f_{pi}}{f_{pr} + f_{pi}}$

$W_t = 0$, ที่ค่าอื่น ๆ

2.2.4 กลยุทธ์วิวัฒนาการ (Evolution strategies)

ในปี ค.ศ. 1965 Rechenberg และ Schwefel [3] ได้ทำการเลียนแบบกระบวนการวิวัฒนาการโดยเน้นการแก้ปัญหาออปติไมเซชันของฟังก์ชันค่าจริงที่มีความต่อเนื่องโดยมีรูปแบบที่คล้ายกับการโปรแกรมเชิงวิวัฒนาการดังนี้

1) *Vector representation* : ในการทำ Optimization ของฟังก์ชันค่าจริง โดยกำหนดฟังก์ชันวัตถุประสงค์เป็น $f(\vec{p})$: $\Re^n \rightarrow \Re$ โดยเวกเตอร์ $\vec{p}$ เป็นสมาชิกของกลุ่มประชากร (Population) ที่จะถูกพัฒนาต่อไป

2) *Initialization* : ประชากรเริ่มต้น p_i ; i=1,2,...,N_p ถูกเลือกแบบสุ่มจาก Feasible ranges โดยใช้การสุ่มแบบ Uniform random

3) *Creation of offspring* : p_i' จะถูกสร้างจาก p_i โดยที่ $p'_{i\,j} = p_{i\,j} + N(0,\sigma_j^2)$ ซึ่งจะเรียกการดำเนินการนี้ว่า Mutation โดยที่ $\sigma_j = \beta \dfrac{f_{pi}}{f_{\min}}(p_{j,\max} - p_{j,\min})$

4) *Competition & Selection* : สร้าง Competing pool จาก p_i และ p_i' สมาชิกทั้งหมดใน Competing pool จะมีจำนวนเป็น 2 เท่าของประชากรเริ่มต้น จากนั้นจะทำการเลือก

สมาชิกที่มีค่า fitness score สูงสุดหรือต่ำสุด(ขึ้นอยู่กับว่าเป็นการหาค่าสูงสุดหรือต่ำสุดของฟังก์ชัน) จำนวน N_p ให้อยู่รอดเพื่อใช้เป็นประชากรเริ่มต้นใน generation ถัดไป

5) *Stopping rule* : กระบวนการในการสร้าง generation ใหม่นั้น จะดำเนินการไปจนกระทั่งไม่สามารถที่จะลดหรือเพิ่มค่า fitness score ได้อีกแล้ว หรือเมื่อจำนวน generation ถึงค่าสูงสุดที่ตั้งไว้

การคำนวณค่าความเหมาะสมของสมาชิกในกลุ่มประชากรนั้นสามารถคำนวณได้จากฟังก์ชันวัตถุประสงค์โดยตรง

2.2.5 เจนเนติกอัลกอริทึม (Genetic algorithms) [3-5]

เจนเนติกอัลกอริทึมเป็นวิธีการที่จำลองแบบกระบวนการวิวัฒนาการในธรรมชาติเช่นเดียวกับ การโปรแกรมเชิงวิวัฒนาการและกลยุทธ์วิวัฒนาการ มีหลักการพื้นฐานคือ พยายามพัฒนาประชากรของผลเฉลยเพื่อให้มีความเหมาะสมต่อปัญหาที่กำลังพิจารณา โดยใช้กฎการคัดเลือกตามธรรมชาติซึ่งอาศัยค่าความเหมาะสม (Fitness score) เป็นตัวตัดสิน ดังนี้

1) ประชากรเริ่มต้นจะถูกสร้างขึ้นโดยวิธีการสุ่มในรูปของสตริง (ปกติใช้ระบบเลขฐานสอง) ซึ่งจะมีลักษณะคล้ายกับแบบจำลองโครโมโซมของสิ่งมีชีวิต สมาชิกทุกตัวในกลุ่มประชากรเริ่มต้นนี้จะถูกแปลงเป็นค่าจริงเพื่อใช้วัดความเหมาะสมในการอยู่รอดและถ่ายทอดสู่รุ่นการถ่ายทอดถัดไป (ปกติใช้ฟังก์ชันวัตถุประสงค์และเรียกค่านี้ว่าค่าความเหมาะสม)
2) กระบวนการสร้างรุ่นการถ่ายทอดจะเริ่มด้วยการจับคู่ของสตริงเพื่อสร้าง Offspring การจับคู่นี้จะใช้กระบวนการสุ่มโดยอาศัยความน่าจะเป็น สมาชิกที่มีค่าความเหมาะสมสูงจะมีโอกาสที่จะถูกเลือกเพื่อสร้างรุ่นการถ่ายทอดได้มากกว่าและหลายครั้งกว่าสมาชิกที่มีค่าความเหมาะสมต่ำ กระบวนการสร้างรุ่นการถ่ายทอดนี้ปกติจะใช้การดำเนินการใน 2 ลักษณะคือ ครอสโอเวอร์และมิวเตชัน [3,5]
   - ครอสโอเวอร์เป็นกระบวนการแลกเปลี่ยนชิ้นส่วนระหว่างคู่ของสตริงที่ถูกเลือก
   - มิวเตชันเป็นการเปลี่ยนแปลงค่าของสตริงที่ถูกเลือกในบางตำแหน่งแบบสุ่ม

   เจนเนติกอัลกอริทึมจะใช้ครอสโอเวอร์เป็นตัวดำเนินการหลักและมิวเตชันถือเป็นตัวดำเนินการที่มีความสำคัญน้อยกว่าครอสโอเวอร์ ดังนั้นรุ่นการถ่ายทอดที่เกิดขึ้นส่วนใหญ่จะถูกสร้างขึ้นมาจากครอสโอเวอร์ ซึ่งแตกต่างจากการโปรแกรมเชิงวิวัฒนาการและกลยุทธ์วิวัฒนาการที่จะใช้มิวเตชันเป็นตัวดำเนินการหลักและเป็นตัวดำเนินการเพียงตัวเดียวที่ใช้

3) รุ่นการถ่ายทอดที่ถูกสร้างขึ้นทั้งหมดจะมีจำนวนเท่ากับประชากรเริ่มต้นและจะแทนที่ประชากรเริ่มต้นทั้งหมด และด้วยการสร้างรุ่นการถ่ายทอดหลาย ๆ รุ่น จะทำให้ประชากรที่ถูกสร้างขึ้นใหม่นี้มีค่าคู่เข้าสู่จุดคำตอบที่เหมาะสมของปัญหา (Near-optimum solution) โดยจำนวนรุ่นการถ่ายทอดที่ต้องใช้นั้นจะขึ้นอยู่กับความซับซ้อนของปัญหา

2.3 สรุป

ในบทนี้กล่าวถึงการแก้ปัญหาออปติไมเซชันซึ่งเป็นวิธีการที่ใช้เลือกจุดทำงานที่เหมาะสมของระบบพลวัต ในที่นี้ได้นำเสนอวิธีการแก้ปัญหาออปติไมเซชันด้วยวิธีการวิเคราะห์ทางคณิตศาสตร์ กับวิธีการคำนวณเชิงวิวัฒนาการ ซึ่งทั้งสองวิธีมีข้อดีและข้อด้อยที่แตกต่างกัน โดยที่วิธีการคำนวณทางคณิตศาสตร์นั้นใช้เวลาคำนวณเร็วกว่า แต่จุดคำตอบที่ได้จากวิธีการคำนวณเชิงวิวัฒนาการจะดีกว่า ในที่นี้ได้นำเสนอถึงแนวคิด ตลอดจนวิธีการแก้ปัญหาโดยใช้วิธีการดังกล่าวมาพอสังเขป

การคำนวณทางคณิตศาสตร์ที่นำเสนอนั้นมีหลายวิธี แต่วิธีที่จะใช้ในวิทยานิพนธ์ฉบับนี้คือ Sequential quadratic programming และการคำนวณเชิงวิวัฒนาการนั้นได้นำเสนอ การโปรแกรมวิวัฒนาการ กลยุทธ์วิวัฒนาการ และเจนเนติกอัลกอริทึม แต่วิธีการที่เลือกใช้ในการแก้ปัญหาออปติไมเซชันในวิทยานิพนธ์นี้คือ กลยุทธ์วิวัฒนาการ