สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

ปีการศึกษา 2541

การรักษาเสถียรภาพระบบไฟฟ้ากำลังโดยใช้เจเนติกอัลกอริทึม

Power System Stability using Genetic Algorithm

โดย

นายคงเกียรติ กุลกลางดอน

นายถวิล ภูมิฐาน

นายทรงศักดิ์ ม่วงงาม

อาจารย์ที่ปรึกษา

รศ.มณฑล ลีลาจินดาไกรฤกษ์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เลขหมู่.....
เลขทะเบียน 34169
วัน, เดือน, ปี - 6 ต.ค. 2542

เอกสารนี้ใช้เพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2541

ภาควิชาวิศวกรรมไฟฟ้า

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การรักษาเสถียรภาพระบบไฟฟ้ากำลังโดยใช้เจเนติกอัลกอริทึม

ผู้จัดทำ

1. นายคงเกียรติ กุลกลางดอน
2. นายถวิล ภูมิฐาน
3. นายทรงศักดิ์ ม่วงงาม

_____อาจารย์ที่ปรึกษา

(รศ.มณฑล ลีลาจินดาไกรฤกษ์)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การรักษาเสถียรภาพระบบไฟฟ้ากำลังโดยเจเนติกอัลกอริทึม

นายคงเกียรติ กุลกลางดอน
นายถวิล ภูมิฐาน
นายทรงศักดิ์ ม่วงงาม
รศ.มณฑล ลีลาจินดาไกรฤกษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2541

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เสนอวิธีการออกแบบตัวควบคุมชนิดไฮบริด เพื่อปรับปรุงเสถียรภาพระบบไฟฟ้ากำลังโดยใช้เจเนติกอัลกอริทึม เจเนติกอัลกอริทึมเป็นวิธีการจำลองวิวัฒนาการธรรมชาติทางชีววิทยา โดยนำตัวแปรของอุปกรณ์ควบคุมต่างๆ จัดให้อยู่ในรูปโครงสร้างโครโมโซมแล้วนำเข้าสู่กระบวนการคัดเลือกและกระบวนการทางพันธุกรรม โครโมโซมที่เหมาะสมจะคัดเลือกจากโครโมโซมที่มีค่าความเหมาะสมสูงสุดและถูกนำไปใช้เป็นรูปแบบของตัวรักษาเสถียรภาพระบบไฟฟ้ากำลัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

POWER SYSTEM STABILITY USING GENETIC ALGORITHM

Kongkiat Gulglangdon

Tawin Phoomthan

Songsak Muang-ngam

Assoc.Prof.Monthol Leelajindakrairerk Advisor

1998

Abstract

This project presents a method to design Hybrid-type controller to improving power system stability using Genetic Algorithm. The Genetic Algorithm to emulate the natural biological evolution. In this application, the tuning parameters from the controllers are coded into the chromosome structure. The best elitist chromosome are selected from a high fitness chromosome which is used to be the uniform of power system stabilizers.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 1
บทนำ

1.1 ความเป็นมาของโครงงาน

ปัจจุบันการรักษาเสถียรภาพของระบบไฟฟ้ากำลังได้มีการนำเอาวิธีการควบคุมและอุปกรณ์ควบคุมแบบต่างๆ มาใช้อยู่หลายวิธีซึ่งแต่ละอุปกรณ์ควบคุมจะมีวิธีการ รูปแบบ และข้อจำกัดที่เป็นลักษณะเฉพาะตัว ในระบบโรงจักรที่ทำการผลิตและส่งจ่ายกำลังไฟฟ้า มีการใช้อุปกรณ์ควบคุมอยู่อย่างเพื่อทำการรักษาและปรับปรุงเสถียรภาพระบบไฟฟ้ากำลังให้ดีเสมอ เช่น ตัวรักษาเสถียรภาพระบบไฟฟ้ากำลัง(PSS) ที่ใช้กับเครื่องกำเนิดไฟฟ้าแบบซิงโครนัส หรือการนำเอาทฤษฎีฟัซซี่มาใช้ในการหาสัญญาณควบคุมให้กับอุปกรณ์ควบคุมต่างๆ การใช้อุปกรณ์ชดเชยกำลังไฟฟ้าเสมือนแบบสถิตย์(SVC) ในการควบคุมกับสายส่งระยะไกล เป็นต้น

จากตัวอย่างอุปกรณ์ควบคุมข้างต้นนี้ แต่ละอุปกรณ์จะมีตัวแปรที่ปรับเปลี่ยนค่าได้(Tuning Parameters) ในการตั้งระดับการควบคุม การหาค่าตัวแปรเหล่านี้ แต่เดิมได้จากการทดสอบทุกค่าในช่วงของค่าที่จะเป็นไปได้ ซึ่งจะต้องใช้เวลาในการหาค่าที่เหมาะสมที่สุดเป็นเวลานาน และค่าของตัวแปรนั้นๆ จะใช้ได้กับระบบๆ หนึ่งเท่านั้น เมื่อระบบมีการเปลี่ยนแปลงอุปกรณ์ควบคุมใหม่ก็จำเป็นที่จะต้องมีการหาค่าของตัวแปรที่เหมาะสมใหม่ ทำให้เกิดความยุ่งยากมากขึ้น

วิธีการทางเจเนติกอัลกอริทึม เป็นวิธีการหนึ่งที่ใช้หาค่าที่เหมาะสมที่สุดสำหรับตัวแปรของแต่ละอุปกรณ์ควบคุม โดยใช้เวลาน้อยมากเมื่อเทียบกับแบบเดิมและค่าที่เหมาะสมที่สุดของตัวแปรที่หาได้นั้นยังสามารถทำให้ระบบไฟฟ้ากำลังมีเสถียรภาพได้อย่างดี อีกทั้งวิธีการทางเจเนติกอัลกอริทึมมีความหลากหลายในการนำไปประยุกต์ใช้ จึงน่าจะเป็นวิธีการที่เหมาะสมในการใช้งานต่อไปในอนาคต

1.2 วัตถุประสงค์ของโครงงาน

การนำเอาเจเนติกอัลกอริทึมมาช่วยในการออกแบบหาค่าที่เหมาะสมที่สุดสำหรับตัวแปรได้นำมาทดลองกับแบบจำลองของอุปกรณ์ควบคุมแบบไฮบริด ซึ่งประกอบด้วยการทำงานร่วมกันของตัวรักษาเสถียรภาพระบบไฟฟ้ากำลัง(Power System Stabilizer:PSS) ตัวควบคุมแบบฟัซซี่ (Fuzzy Control:FL) ซึ่งอาศัยข้อมูลที่ได้รับจากอุปกรณ์ควบคุมแบบพีดี(Proportional Derivative:PD) และอุปกรณ์กลไกการตัดสินแบบฟัซซี่(Fuzzy Judgement:FJ) โดยมีวัตถุประสงค์ที่สำคัญคือ

1.เพื่อการพัฒนาและออกแบบหาค่าที่เหมาะสมที่สุดสำหรับตัวแปรของอุปกรณ์ควบคุมแบบต่างๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.เพื่อศึกษาการรักษาเสถียรภาพของระบบไฟฟ้ากำลัง ภายใต้สภาวะการรบกวนแบบเล็กน้อยและการรบกวนแบบรุนแรง

1.3 แนวคิดที่ใช้ในโครงงาน

เจเนติกอัลกอริทึมเป็นวิธีการที่จำลองกระบวนการวิวัฒนาการทางธรรมชาติ คือ การคัดเลือกทางธรรมชาติและอาศัยพื้นฐานความคิดทางพันธุกรรมในการถ่ายทอดลักษณะต่างๆ ไปยังลูกหลานเพื่อนำมาใช้หาคำตอบที่ใกล้เคียงหรือดีที่สุดของปัญหาหนึ่ง เจเนติกอัลกอริทึมเป็นวิธีการหาคำตอบโดยพิจารณาและดำเนินการจากตัวแปรหรือตัวแปรต่างๆ ของปัญหาซึ่งถูกนำมาจัดให้อยู่ในรูปโครงสร้างของโครโมโซมตามที่กำหนดเพื่อคัดเลือกโครโมโซมคำตอบที่ดีที่สุด สำหรับสร้างวิวัฒนาการคำตอบให้ดีขึ้น โดยแลกเปลี่ยนหรือแปลงค่าตัวแปรต่างๆ ระหว่างโครโมโซมที่ถูกคัดเลือก อันจะทำให้คำตอบถูกปรับปรุงให้ดีขึ้น

การใช้เจเนติกอัลกอริทึมในการออกแบบหาค่าที่เหมาะสมที่สุดของตัวแปรสำหรับอุปกรณ์ควบคุมเพื่อการรักษาและปรับปรุงระบบไฟฟ้ากำลังนั้น ทำโดยการนำเอาตัวแปรที่สำคัญจากอุปกรณ์ควบคุมต่างๆ มาจัดให้อยู่ในรูปโครงสร้างโครโมโซมที่กำหนดขึ้น แล้วปฏิบัติตามกระบวนการทางพันธุกรรมได้แก่ การคัดเลือกโครโมโซมที่เหมาะสม การครอสโอเวอร์ การมิวเทชัน การรีโพรดักชันและการอินเวอร์ชัน

1.4 ขอบเขตของโครงงาน

โครงงานนี้มีของเขตในการเขียนโปรแกรมคอมพิวเตอร์ เพื่อจำลองระบบไฟฟ้ากำลังของเครื่องกำเนิดไฟฟ้าชิงโครนัสเดี่ยวต่อกับบัสอนันต์ โดยใช้อุปกรณ์ในการรักษาเสถียรภาพระบบไฟฟ้ากำลังแบบไฮบริด โดยโปรแกรมที่ออกแบบจะใช้วิธีการทางเจเนติกอัลกอริทึมในการหาค่าที่เหมาะสมที่สุดของตัวแปรสำหรับอุปกรณ์ควบคุมแบบต่างๆ โดยใช้ภาษาซีในการเขียนโปรแกรม

1.5 ขั้นตอนการศึกษา

ขั้นตอนการศึกษาต่างๆ สามารถแบ่งเป็นหัวข้อได้ดังนี้

1. ศึกษาแบบจำลองซึ่งประกอบด้วย
    1.1 ระบบเครื่องกำเนิดไฟฟ้าเดี่ยวต่อกับบัสอนันต์
    1.2 ตัวควบคุมเครื่องกำเนิดไฟฟ้า ได้แก่ AVR, GOV และ PSS ชนิดต่างๆ
    1.3 ตัวควบคุมแบบไฮบริด ซึ่งประกอบด้วย ΔP-PSS, FL, PD และ FJ
2. ศึกษาทฤษฎีพื้นฐานของการควบคุมแบบฟัซซี่ และการประยุกต์ใช้งานฟัซซี่ในส่วนที่เกี่ยวข้องกับโครงงาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. ศึกษาทฤษฎีพื้นฐานของเจเนติกอัลกอริทึม และการประยุกต์ใช้งานของเจเนติกอัลกอริทึมในส่วนโครงงาน

4. เขียนโปรแกรมเจเนติกอัลกอริทึม และประยุกต์ใช้งานกับแบบจำลองดังต่อไปนี้

4.1 แบบจำลองที่ควบคุมด้วยตัวควบคุมแบบไฮบริด ที่ป้อนสัญญาณควบคุมให้ AVR

4.2 แบบจำลองที่ควบคุมด้วย $\Delta\omega$-PSS ที่ป้อนสัญญาณควบคุมให้ AVR

4.3 แบบจำลองที่ควบคุมด้วย ΔP-PSS ที่ป้อนสัญญาณควบคุมให้ AVR

4.4 แบบจำลองที่ควบคุมด้วย SVC ชนิดฟัซซี่

4.5 เป็นกรณีศึกษาเพิ่มเติมโดยใช้แบบจำลองที่ควบคุมด้วยการทำงานร่วมกันของ $\Delta\omega$-PSS และ SVC ชนิดฟัซซี่

5. บันทึกผลการทดลองจากการควบคุมด้วยวิธีการต่างๆ

6. สรุปผลการทดลอง การทำวิจารณ์ เปรียบเทียบข้อดีข้อเสียและจัดทำข้อเสนอแนะ

1.6 ประโยชน์ที่คาดว่าจะได้รับจากการศึกษา

1. การใช้เจเนติกอัลกอริทึม ทำให้ลดเวลาในการหาค่าตัวแปรที่เหมาะสมสูงสุด ซึ่งแต่เดิมวิธีการทดสอบทุกค่าของทุกตัวแปรทำให้ใช้เวลานานและมีความยุ่งยาก อีกทั้งเป็นการประหยัดงบประมาณที่ใช้ทำการศึกษาเพราะได้ทำการจำลองด้วยเครื่องคอมพิวเตอร์ส่วนบุคคล(Personal Computer)
2. ในการออกแบบตัวควบคุมแบบใหม่ ตัวแปรควบคุมที่ให้ผลการตอบสนองที่ดีที่สุด คือ สิ่งที่เราไม่ทราบคำตอบมาก่อน บางครั้งการคำนวณด้วยสมการทางคณิตศาสตร์แบบเดิมก็เป็นสิ่งที่ยุ่งยากเจเนติกอัลกอริทึมก็เป็นอีกวิธีการหนึ่งที่สามารถแก้ปัญหานี้ได้
3. การจำลองบนคอมพิวเตอร์ทำให้สามารถทราบปัญหาเกี่ยวกับเสถียรภาพได้ล่วงหน้า และสามารถทำการทดสอบวิธีควบคุมที่สามารถแก้ไขปัญหาเสถียรภาพได้อย่างมีประสิทธิภาพ

1.7 คำจำกัดความที่ใช้ในการศึกษา

1.7.1 สัญลักษณ์

δ = มุมถ่ายโอนกำลัง(Power angle : rad)

θ_m = มุมทางกลของโรเตอร์(Mechanical angle of Rotor : rad)

ω = ความเร็ว(Speed : rad/s)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ω_0 = ความเร็วซิงโครนัส(Synchronus speed : rad/s)

ω_m = ความเร็วเพลาโรเตอร์(Rotor shaft velosity : rad/s)

σ = เพอร์มาเนนท์ครูป(Permanent droop)

ϕ_d = ฟลักซ์แม่เหล็กในแกน d (d-axis flux)

ϕ_q = ฟลักซ์แม่เหล็กในแกน q (q-axis flux)

ϕ_f = ฟลักซ์แม่เหล็กในขดลวดฟิลด์(field flux)

B = ซัสเซบแตนซ์(susceptance)

D = สัมประสิทธิ์แดมปิ้ง(Damping coefficient)

i_d = กระแสในแกน d (d-axis current)

i_q = กระแสในแกน q (q-axis current)

I = โมเมนท์ความเฉื่อย(Moment of inertia)

J = ดัชนีการทำงาน (Performance index)

K_A = เกนของเอกไซเตอร์(Exciter gain)

K_f = เกนของลูปป้อนกลับสำหรับควบคุมแรงดัน(voltage control feedback loop gain)

K_s = เกนของตัวชดเชยกำลังไฟฟ้าแบบสถิตย์ (SVC gain)

M = ค่าคงที่ความเฉื่อย(inertia constant : pu-s^2 / elec.degree)

P_e = กำลังไฟฟ้าเอาท์พุท(Electrical power output : pu)

P_i = กำลังไฟฟ้าอินพุท(Electrical power input : pu)

P_0 = กำลังกลเอาท์พุท(Mechanical power output ; pu)

T_a = ทอร์กเร่ง(Accelerating torque ; pu)

T_A = ค่าคงตัวเวลาเอกไซเตอร์(exciter time constant : s)

T_c = ค่าคงตัวเวลาตัวชดเชยกำลังไฟฟ้าแบบสถิตย์(SVC time constant : s)

T_{d0}' = ค่าคงตัวเวลาเปิดวงจรขดลวดฟิลด์(open circuit time constant of the field)

T_e = ทอร์กไฟฟ้าเอาท์พุท(electrical torque output : pu)

T_i = ทอร์กไฟฟ้าอินพุท(electrical torque : pu)

T_f = ค่าคงตัวเวลาป้อนกลับควบคุมแรงดัน(voltage control feedback loop time constant : s)

T_g = ค่าคงตัวเวลาเกท(gate time constant :s)

T_m = ค่าคงตัวเวลาวงจรการวัด (measuring circuit time constant : s)

T_r = ค่าคงตัวเวลาวงจรรีเซต(reset circuit time constant : s)

T_t = ค่าคงตัวเวลาเทอร์ไบน์(turbine time constant : s)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

T_v = ค่าคงตัวเวลาวงจรวัดของตัวชดเชยกำลังไฟฟ้าแบบสถิตย์(measuring circuit time constant of SVC : s)

V_c = แรงดันที่ตัวชดเชยกำลังไฟฟ้าแบบสถิตย์(SVC voltage)

V_d = แรงดันในแกน d (d-axis voltage)

V_q = แรงดันในแกน q (q-axis voltage)

V_f = แรงดันขดลวดฟิลด์(field voltage)

V_0 = แรงดันที่บัสอนันต์(Infinite bus voltage)

V_t = แรงดันที่ขั้วเครื่องกำเนิดไฟฟ้า(Terminal voltage)

X_d = ซิงโครนัสรีแอกแตนซ์ในแกน d(Synchronous reactance d-axis : pu)

X_q = ซิงโครนัสรีแอกแตนซ์ในแกน q(Synchronous reactance q-axis : pu)

X_d' = ทรานเซียนท์รีแอกแตนซ์ในแกน d (Transient reactance d-axis : pu)

X_q' = ทรานเซียนท์รีแอกแตนซ์ในแกน q (Transient reactance q-axis : pu)

1.7.2 คำย่อ

AVR = ตัวควบคุมแรงดันไฟฟ้าอัตโนมัติ (Automatic Voltage Regulator)

GA = เจเนติกอัลกอริทึม (Genetic Algorithm)

GOV = ตัวควบคุมความเร็วเครื่องกำเนิดไฟฟ้า (Gorvernor)

PSS = ตัวควบคุมเสถียรภาพระบบไฟฟ้ากำลัง (Power System Stabilizer)

SVC = ตัวชดเชยกำลังไฟฟ้าเสมือนแบบสถิตย์ (Static Var Compensator)

SSR = การเลือกแบบสุ่มโดยมีการแทนที่ (Stochastic Sampling with Replacement)

SSPR = การเลือกแบบสุ่มโดยมีการแทนที่บางส่วน (Stochastic Sampling with Partial Replacement)

SUS = การเลือกแบบยูนิเวอร์ซอลโดยสุ่ม (Stochastic Universal Sampling)

TCR = รีแอกเตอร์ควบคุมแบบไทริสเตอร์ (Thyristor Control Reactor)

TSC = ตัาเก็บประจุสวิทซ์แบบไทริสเตอร์ (Thyristor Switching Capacitor)

FL = อุปกรณ์ควบคุมแบบฟัซซี่(Fuzzy Controller)

FJ = กลไกการตัดสินแบบฟัซซี่(Fuzzy Judgement)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
ความรู้พื้นฐานเกี่ยวกับเจเนติกอัลกอริทึม

ในปัจจุบันนี้ปัญหาที่ต้องการคำตอบที่ดีที่สุด(Optimal Solution) ทางวิทยาศาสตร์ วิศวกรรมศาสตร์ คอมพิวเตอร์ หรือในการทำงานต่างๆที่เกิดขึ้นมากมายนั้น สามารถหาคำตอบได้หลายๆวิธี ซึ่งแตกต่างกันไปตามชนิดของปัญหา ความคิด เทคนิค วิธีการวิเคราะห์ปัญหานั้นๆ และความแพร่หลายในการพัฒนาศักยภาพของคอมพิวเตอร์ให้รู้จักเรียนรู้เพื่อช่วยหาคำตอบหรือช่วยตัดสินคำตอบในขั้นต้นมีมากขึ้น โดยปัจจุบันนี้นักวิทยาศาสตร์ได้เริ่มนำความรู้เกี่ยวกับทฤษฎีหรือกฎเกณฑ์ทางธรรมชาติมาช่วยในการศึกษาวิจัย เช่น นิวรอลเน็ตเวิร์ค (Neural Network) ฟัซซี่ลอจิก (Fuzzy Logic) เป็นต้น เจเนติกอัลกอริทึมเป็นอีกวิธีการหนึ่งที่จำลองรูปแบบวิธีการทางชีววิทยา ในการให้กำเนิดประชากรรุ่นใหม่หรือขยายเผ่าพันธุ์ในรุ่นลูก รุ่นหลานต่อไป ซึ่งอาศัยพื้นฐานความคิดของวิวัฒนาการทางธรรมชาติถ่ายทอดลักษณะต่างๆ ทางพันธุกรรม โดยปฏิบัติตามกระบวนการทางพันธุศาสตร์ เพื่อใช้ในการหาคำตอบที่ดีที่สุดหรือใกล้เคียงที่สุดของปัญหาโดยคอมพิวเตอร์

2.1 เจเนติกอัลกอริทึมเบื้องต้น

ปี ค.ศ. 1975 John Holland เริ่มสนใจศึกษาในทฤษฎีวิวัฒนาการทางธรรมชาติ (Natural Evolution) ในการกำเนิดประชากร (Population) สิ่งมีชีวิตในรุ่นต่อๆไป โดยกระบวนการธรรมชาติทางชีววิทยาประกอบด้วยการคัดเลือกทางธรรมชาติ (Natural Selection) คือสิ่งมีชีวิตใดแข็งแรงกว่าย่อมมีโอกาสอยู่รอดได้มากกว่านั่นหมายถึงการมีโครโมโซมซึ่งประกอบด้วยยีนส์ต่างๆ ที่มีลักษณะที่ดี นั้นมีโอกาสอยู่รอดได้มากกว่า โครโมโซมที่สามารถอยู่รอดได้ก็จะถูกถ่ายทอดยีนส์ที่มีลักษณะที่ดีเหล่านั้นไปยังลูกหลานได้มากกว่าเช่นกัน และกระบวนการทางพันธุศาสตร์ (Genetic Operation) คือการกำเนิดโครโมโซมใหม่โดยการผสมพันธุ์เพื่อถ่ายทอดยีนส์จากการครอสโอเวอร์ หรือกลายพันธุ์จากมิวเทชัน

จากความเชื่อในวิวัฒนาการทางธรรมชาติ ที่แสดงถึงลักษณะที่เป็นอยู่ของสิ่งมีชีวิตโดยการถ่ายทอดลักษณะต่างๆบนโครโมโซมนั้นมีคุณสมบัติทั่วไปที่ยอมรับกันคือ

(1) วิวัฒนาการเป็นผลที่เกิดขึ้นเนื่องจากการเปลี่ยนแปลงทางโครโมโซม ที่เป็นอยู่ซึ่งแสดงลักษณะของสิ่งมีชีวิตนั้นๆ

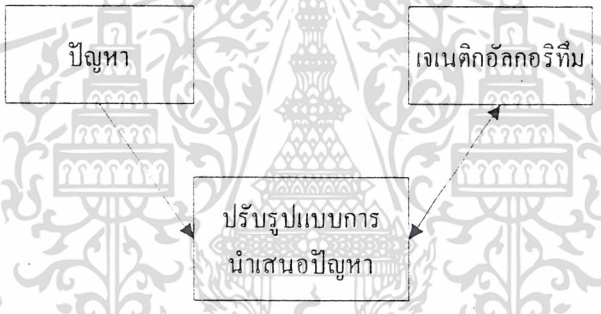
(2) ธรรมชาติทางการคัดเลือกมีความสัมพันธ์กับโครโมโซมที่แสดงถึงประสิทธิภาพของโครงสร้างที่ดี ที่คัดเลือกเพื่อถ่ายทอดส่วนของโครงสร้างที่ดี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(3) การถ่ายทอดในลักษณะที่เกิดวิวัฒนาการนั้น โครโมโซมพ่อ-แม่ มีการแลกเปลี่ยนส่วนโครงสร้างกันเพื่อสร้างโครโมโซมลูก และเหตุผลที่ทำให้เกิดโครโมโซมที่แตกต่างออกไปคือกระบวนการผ่าเหล่า

(4) วิวัฒนาการทางธรรมชาติมิได้เป็นสิ่งที่เกิดจากการจดจำ แต่เป็นกระบวนการที่เกิดจากโครงสร้างต่างๆ ในโครโมโซมที่เหมาะสมกับสภาพแวดล้อมที่เกิดขึ้นในขณะนั้น

Holland คิดว่าแนวความคิดจากคุณสมบัติเหล่านี้ น่าจะนำมาปรับปรุงใช้กับคอมพิวเตอร์ให้ช่วยแก้ปัญหาที่ยุ่งยากต่างๆ ในการหาคำตอบที่ดีที่สุดหรือใกล้เคียงที่สุด เขาจึงได้ทำการวิจัยโดยจำลองแบบเพื่อทดลองกับปัญหาแบบต่างๆ โดยมีจุดมุ่งหมายที่จะศึกษาระบบปรับปรุงการประมวลผลของ (self adaptive process) และเพื่อสร้างโปรแกรมระบุผู้เชี่ยวชาญ (artificial system software) เพื่อแก้ปัญหา โดยอาศัยแนวความคิดของระบบทางธรรมชาติ และค้นพบวิธีใหม่ซึ่งเรียกว่า เจเนติกอัลกอริทึม (Genetic Algorithm :GA)



รูปที่ 2.1 หลักการเบื้องต้นของ GA

รูปที่ 2.1 แสดงหลักการเบื้องต้นในการใช้ GA แก้ปัญหา โดยจะต้องมีการปรับปรุงรูปแบบปัญหาในการนำเสนอ GA ในลักษณะที่เหมาะสม เพราะ GA เป็นวิธีการค้นหาคำตอบโดยอาศัยวิธีการเลียนแบบการคัดเลือกทางธรรมชาติ และธรรมชาติทางพันธุกรรมโดยการรวมกันหรือสลับเปลี่ยนตัวแปรต่างๆ อันเป็นองค์ประกอบโครงสร้างของปัญหาที่ให้คำตอบที่ต้องการ ซึ่งอาศัยหลักการสุ่ม เพื่อปรับปรุงความสามารถในการค้นหาคำตอบที่ดีขึ้น การค้นหาคำตอบจากรุ่นหนึ่งไปยังรุ่นถัดไปตามวิวัฒนาการทางธรรมชาตินั้นคำตอบในรุ่นใหม่เกิดขึ้นจากการสร้างความสัมพันธ์ของโครงสร้างต่างๆ ที่ประกอบด้วยค่าตัวแปรที่เหมาะสมดีในรุ่นก่อน ดังนั้นจึงทำให้ได้คำตอบที่ขึ้น จะเห็นได้ว่าวิธีการพื้นฐานของ GA เป็นแบบการสุ่ม แต่มีหลักการและประสิทธิภาพจากการเดาคำตอบใหม่จากสถิติคำตอบเดิมที่ดี ซึ่งแตกต่างจากวิธีทั่วไปคือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(1) GA ค้นหาคำตอบภายใต้โครงสร้างของปัญหาอันเกิดจากการกำหนดรหัส (coding) รูปแบบโครงสร้างจากกลุ่มตัวแปรต่างๆ ของปัญหานั้น ไม่ใช่ค้นหาคำตอบจากกลุ่มของตัวแปรนั้น

(2) GA ค้นหาคำตอบจากการพิจารณาจากประชากรคำตอบ หรือกลุ่มคำตอบ ไม่ใช่พิจารณาจากคำตอบใดคำตอบหนึ่ง

(3) ค้นหาคำตอบจากผลลัพธ์ของกลุ่มค่าตัวแปรที่ฟังก์ชันเป้าหมายของปัญหา

(4) GA ค้นหาคำตอบโดยอาศัยการถ่วงน้ำหนักความเหมาะสมของแต่ละคำตอบจากกลุ่มคำตอบนั้นๆ

2.2 ฟังก์ชันเป้าหมายกับฟังก์ชันความเหมาะสม

การหาคำตอบที่ดีที่สุดของปัญหาของ GA มีพื้นฐานอยู่บนผลลัพธ์จากการหาคำตอบที่ผ่านมา วิธีการของ GA จะไม่พิจารณาจากขั้นตอนของการแก้ปัญหา แต่จะพิจารณาโดยตัดสินว่าคำตอบใหม่ที่ได้รับดีขึ้นหรือไม่ หรือเป็นคำตอบที่ใกล้เคียงคำตอบที่ต้องการหรือไม่ จากฟังก์ชันเป้าหมาย (Object Function : f) เนื่องจากแต่ละปัญหาจะสามารถกำหนดฟังก์ชันเป้าหมายซึ่งเป็นฟังก์ชันที่แสดงความสัมพันธ์ของแต่ละตัวแปร เงื่อนไข หรือข้อกำหนดต่างๆ ของปัญหานั้นๆ ที่ระบุคำตอบใดคำตอบหนึ่งที่สามารถเป็นไปได้ ณ ค่าตัวแปร เงื่อนไข หรือข้อกำหนดดังกล่าว สำหรับฟังก์ชันเหมาะสม (Fitness Function : F) เป็นฟังก์ชันที่กำหนดความเหมาะสม(fitness) แต่ละโครโมโซมเปรียบเสมือนค่าฟ ามสามารถในการอยู่รอดของแต่ละโครโมโซมและฟังก์ชันที่กำหนดโอกาส หรือสัดส่วนที่แต่ละโครโมโซมเหมาะสมที่จะถูกคัดเลือกมากน้อยเพียงใด นั่นคือฟังก์ชันความเหมาะสมจะเป็นฟังก์ชันที่แสดงถึงค่าคำตอบที่เกิดขึ้นจากชุดตัวแปรของปัญหาของโครโมโซมนั้นดีเพียงใด โดยทั่วไปแล้วเรามักใช้ฟังก์ชันเป้าหมายเป็นฟังก์ชันความเหมาะสม หรืออาจใช้ฟังก์ชันเป้าหมายที่ถูกปรับให้เหมาะสมกับการนำเสนอ GA เป็นฟังก์ชันความเหมาะสมก็ได้

2.3 รูปแบบโครโมโซม

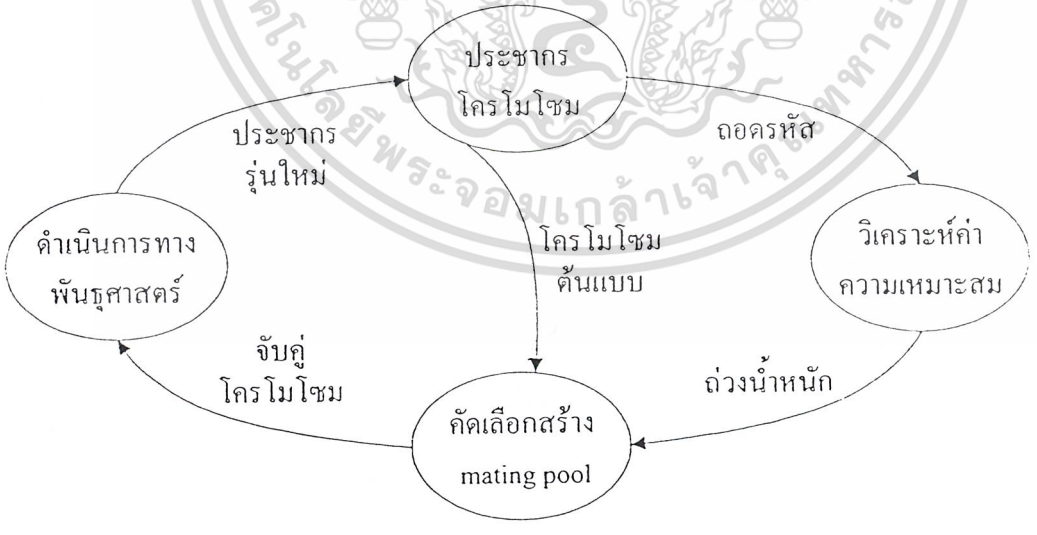
เราทราบกันแล้วว่าวิวัฒนาการทางธรรมชาติชีววิทยานั้นเป็นความเปลี่ยนแปลงต่างๆ ของสิ่งมีชีวิตเกิดขึ้นในโครโมโซม ดังนั้นจุดเริ่มต้นของการจำลองแบบทางธรรมชาติของ GA เพื่อใช้แก้ปัญหา จึงเริ่มจากการมองปัญหาเทียบเท่ากับโครโมโซมชนิดหนึ่ง ประกอบด้วยยีนส์ลักษณะต่างๆ ซึ่งหมายถึงลำดับข้อมูลต่างๆ ที่แปลความหมายแล้วให้ค่าคำตอบของปัญหาค่าหนึ่ง การมองภาพยีนส์ของ GA ให้เสมือนยีนส์ทางพันธุกรรมที่แสดงความหมายหรือเป็นตัวแทนคำตอบใดคำตอบหนึ่ง หรือลักษณะใดลักษณะหนึ่งทางพันธุกรรม ในทางพันธุศาสตร์นั้นยีนส์เป็นตัวแสดงลักษณะที่อยู่รอดในสภาพแวดล้อมขณะนั้น สำหรับ GA นั้นยีนส์เป็นตัวแสดงค่าคำตอบของปัญหาที่แปรผันไปตามประยุกต์ใช้งาน ซึ่งโดยทั่วไปยีนส์หมายถึงตัวแปร ตัวแปร เงื่อนไข หรือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อกำหนดต่างๆ ที่เป็นองค์ประกอบของปัญหา ดังนั้นการกำหนดรูปแบบโครโมโซมของแต่ละปัญหาโดยการแปลงตัวแปร ตัวแปร เงื่อนไข หรือข้อกำหนดต่างๆ ให้อยู่ในลำดับของยีนส์บนโครโมโซมหรือเรียกว่าสตริง (string) อันประกอบด้วยบิท (bit) หรือเรียกว่าอักขระ (character) ซึ่งลักษณะต่างๆ ที่เป็นไปได้ของแต่ละยีนส์คือค่าของบิท (bit Value) หรือค่าตัวแปร ตัวแปรต่างๆ ที่เป็นไปได้ และรูปแบบของค่าบิทที่จัดเรียงบนโครโมโซมคือ ยีโนไทป์ (genotype) ที่แสดงถึงค่าตัวแปร ตัวแปรต่างๆ ที่เป็นไปได้ชุดหนึ่งหรือฟีโนไทป์ (phenotype)นั่นเอง การกำหนดรูปแบบโครโมโซมของปัญหาให้เป็นตามธรรมชาติ โดยกำหนดรหัสในรูปแบบตัวเลขหรือตัวอักษรในช่วงจำกัดค่าตัวแปรหรือตัวแปร และประกอบรวมกันในจำนวนยีนส์หรือความยาวของโครโมโซมที่คงที่ เช่น หากต้องการหาค่าสูงสุดทางของฟังก์ชันทางคณิตศาสตร์ $y = x^2$ ที่ x เป็นจำนวนเต็มในช่วง [0,31] แล้ว วิธีการของ GA ในการแก้ปัญหาโดยกำหนดรูปแบบของโครโมโซมจากการกำหนดรหัสของตัวแปร x เป็นตัวแปรไบนารี 0 หรือ 1 จำนวน 5 ตำแหน่ง ซึ่ง x จะมีค่าตั้งแต่ 00000 ถึง 11111 เป็นค่า 0 ถึง 31 ตามต้องการเป็นต้น

2.4 วัฏจักรเจเนติกอัลกอริทึม

เมื่อกำหนดรูปแบบโครโมโซมและฟังก์ชันความเหมาะสมของปัญหาแล้ว GA จะสามารถประมวลผลหาคำตอบของปัญหาได้ โดยสร้างวิวัฒนาการกลุ่มคำตอบในรุ่นต่อไปตามวัฏจักรการทำงานของ GA (Genetic Algorithm Cycle) ดังรูปที่ 2.2 ซึ่งมี 4 ขั้นตอน คือ



รูปที่ 2.2 วัฏจักรเจเนติกอัลกอริทึม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(1) สร้างประชากรโครโมโซมรุ่นเก่าตามรูปแบบโครโมโซมที่กำหนดไว้ โดยประชากรต้นกำหนด (Initial Population) เกิดจากการสร้างชุดโครโมโซมต้นกำเนิด จากการสุ่มค่าแต่ละบิทของแต่ละโครโมโซม

(2) วิเคราะห์ค่าความเหมาะสมแต่ละโครโมโซมโดยถอดรหัสค่าตัวแปร ตัวแปรต่างๆ ของแต่ละบิทในโครโมโซม และคำนวณค่าความเหมาะสมจากค่าฟังก์ชันความเหมาะสมที่กำหนดไว้

(3) สร้าง mating pool คือชุดโครโมโซมต้นแบบหรือชุดโครโมโซมพ่อ-แม่ ที่สามารถอยู่รอดเป็นต้นแบบ ซึ่งอาศัยการจำลองการคัดเลือกทางธรรมชาติ โดยพิจารณาถ่วงน้ำหนักจากค่าความเหมาะสมของแต่ละโครโมโซมหากโครโมโซมใดดีมีค่าความเหมาะสมมากก็จะมีโอกาสถูกคัดเลือกเป็นต้นแบบมาก

(4) ดำเนินการทางพันธุศาสตร์โดยสุ่มจับคู่โครโมโซมต้นแบบใน mating pool เพื่อสร้างประชากรโครโมโซมรุ่นใหม่ ซึ่งตัวดำเนินการทางพันธุศาสตร์ประกอบด้วย ครอสโอเวอร์ โดยแลกเปลี่ยนค่าบิทบางส่วนของโครโมโซมซึ่งกันและกัน หรือมิวเทชัน โดยสุ่มเปลี่ยนค่าบิทบางบิทของโครโมโซม เป็นต้น

ค้นหาคำตอบของ GA จะประมวลผลซ้ำตามวัฏจักร GA จนกว่าจะได้คำตอบที่พอใจตามกฎเกณฑ์ที่ตั้งไว้ หรือในระยะเวลาตามจำนวนรุ่นที่ดำเนินการที่ต้องการ ซึ่งแสดงอัลกอริทึมการทำงานของ GA ดังนี้

อัลกอริทึม GA

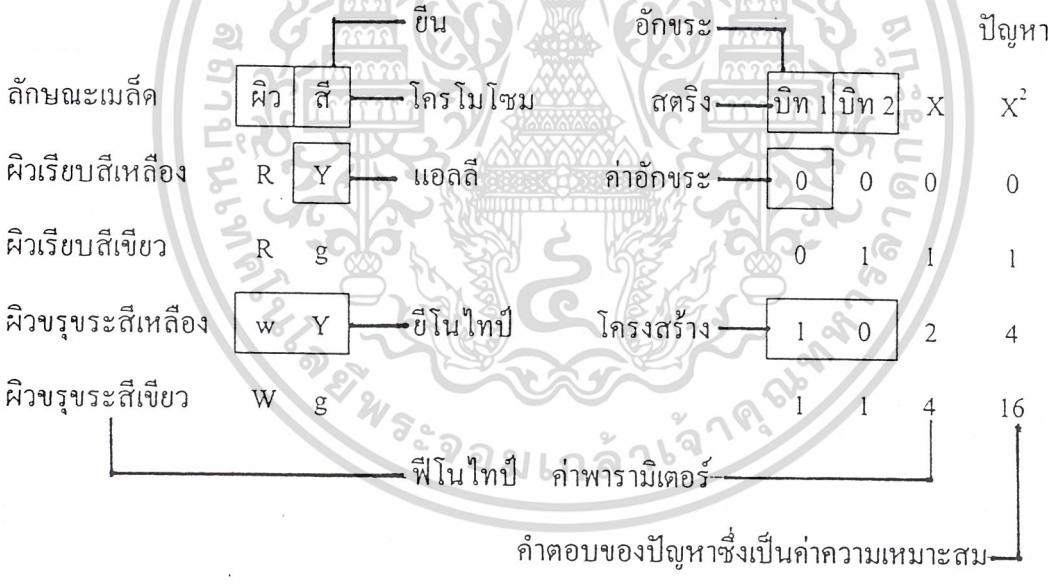
```
BEGIN
    t : = 0;
    // สร้างประชากรโครโมโซมต้นกำเนิดโดยการสุ่ม
    Initpopulation P(t);
    // วิเคราะห์ค่าความเหมาะสมของแต่ละโครโมโซมประชากรต้นกำเนิด
    Evaluate P(t);
    // ตรวจสอบเงื่อนไขความพอใจ (เช่น เวลา,ค่าความเหมาะสม เป็นต้น)
    while not terminate
    begin
        t : t+1;
        // คัดเลือกโครโมโซมต้นแบบจากกลุ่มประชากรรุ่นก่อน
        P'(t) : = Selectparents P(t-1);
        // แลกเปลี่ยนส่วนยีนส์ภายในโครโมโซมต้นแบบ
        Recombine P'(t);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        // มิวเทชันโครโมโซมต้นแบบ
        Mutate P′(t);
        // วิเคราะห์ค่าความเหมาะสมของประชากรรุ่นใหม่
        Evaluate P′(t);
        // ประชากรรุ่นใหม่กลายเป็นประชากรรุ่นเก่าต่อไป
        P(t) := P′(t);
    end;
END.
```

2.5 พันธุศาสตร์ทางชีววิทยา กับ เจเนติกอัลกอริทึม

เพื่อเปรียบเทียบลักษณะทางโครงสร้างทางพันธุศาสตร์กับเจเนติกอัลกอริทึม แล้วเรากล่าวโดยสรุปได้คือ ในทางพันธุศาสตร์ แต่ละโครโมโซม ประกอบด้วยหน่วยเก็บลักษณะ หรือยีนส์ ซึ่งเก็บค่าแสดงลักษณะ หรือแอลลี และแต่ละแบบของชุดยีนส์เรียกว่ายีโนไทป์ ซึ่งแสดงลักษณะภายนอกที่ปรากฏเรียกว่าฟีโนไทป์ ดังรูปที่ 2.3-a



(a) ลักษณะทางพันธุศาสตร์ของโครโมโซมควบคุมลักษณะของเมล็ดถั่ว ซึ่งมียีนส์ลักษณะผิวของเมล็ดคือ มีลักษณะเรียบ (R) หรือขรุขระ (w) และยีนส์ลักษณะสีของเมล็ดคือ มีสีเหลือง (Y) หรือสีเขียว (g)

(b) ลักษณะทางจีเนติก อัลกอริทึมของปัญหาในการหาค่าสูงสุดของฟังก์ชัน $f(x) = x^2$ ซึ่ง x มีค่าอยู่ในช่วง [0,4] โดยพารามิเตอร์ x จะถูกแปลงให้อยู่ในรูปไบนารีสตริง

รูปที่ 2.3 รายละเอียดทางพันธุศาสตร์ กับ เจเนติกอัลกอริทึม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำหรับในทางเจเนติกอัลกอริทึม ตัวแปรหรือตัวแปรของปัญหาจะถูกแปลงให้อยู่ในรูปของสตริง ซึ่งเรียกว่าโครโมโซม ประกอบด้วยอักขระ หรือบิท แต่ละตำแหน่งของโครโมโซมจะเก็บค่าอักขระหรือค่าบิท ที่แสดงถึงโครงสร้างของแต่ละโครโมโซมที่ค่าตัวแปรหรือตัวแปรของปัญหาแตกต่างกัน และเป็นตัวกำหนดค่าความเหมาะสมตามฟังก์ชันความเหมาะสมของแต่ละปัญหา ดังที่ 2.3-b ซึ่งสรุปความหมายเปรียบเทียบคำศัพท์ ที่ใช้พันธุศาสตร์กับทางเจเนติกอัลกอริทึม ดังตารางที่ 2.1

Natural Genetic	Genetic Algorithm
Chromosome	string
Gene	character,bit
Allele	character value,bit value
Locus	string position
Genotype	structure
Phenotype	a decode structure

ตารางที่ 2.1 คำศัพท์ทางพันธุศาสตร์กับทางเจเนติกอัลกอริทึม

2.6 เจเนติกอัลกอริทึมแบบง่าย

GA ในยุคเริ่มแรกของ Holland นั่นคือ เจเนติกอัลกอริทึมแบบง่าย (Simple Genetic Algorithm : SGA) ซึ่งมีขั้นตอนพื้นฐานที่มีกระบวนการไม่มากนักและง่ายในการศึกษาความเข้าใจแต่ละขั้นตอนการทำงานของ GA เพื่อแก้ปัญหาในการหาคำตอบ แบ่งออกเป็น 2 ส่วนคือ ขั้นเตรียมการและขั้นการทำงาน

สำหรับในส่วนของขั้นเตรียมการนั้นเป็นส่วนของการปรับปรุงรูปแบบของ ปัญหาให้เหมาะสมสำหรับการนำเสนอ GA เพื่อใช้ในการแก้ปัญหานั้นๆ ประกอบด้วย

2.6.1 กำหนดฟังก์ชันความเหมาะสม เพื่อความสะดวกและง่ายต่อความเข้าใจในขั้นตอนการทำงานต่างๆ จะกำหนดตัวอย่างปัญหาสำหรับอธิบายรายละเอียดการหาคำตอบของ SGA คือ ปัญหาการหาค่าสูงสุดของฟังก์ชัน $y = x^2$ ที่ x มีค่าระหว่างจำนวนเต็ม I[0,31] ดังนั้น

ตัวอย่าง : ฟังก์ชันเป้าหมาย คือ $f = x^2$

และกำหนดให้ฟังก์ชันความเหมาะสม คือ $F = x^2$

ซึ่งคำตอบที่ดีที่สุด คือ ค่า x ที่มีความเหมาะสมสูงสุด (MAX(F))

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6.2 กำหนดรูปแบบโครโมโซม รูปแบบของโครโมโซม SGA นั้นเป็นแบบไบนารี โดยค่าตัวแปรหรือตัวแปรของปัญหาจะถูกแปลงให้อยู่ในรูปของไบนารีโครโมโซม คือประกอบด้วยบิทที่มีค่าเป็น 0 หรือ 1 ซึ่งเป็นค่าในตัวเลขฐานสอง และมีความยาว (Chromosome Length : lchrom) ตามแต่จะกำหนด ซึ่งแสดงด้วยสัญลักษณ์ได้ดังนี้

B_1	B_2	B_3		B_{lchrom}

ซึ่ง $B_i \in I\ [0, 1]$

ตัวอย่าง : วิธีการเข้ารหัสแบบไบนารีโดยแปลงค่าตัวแปร x ให้อยู่ในไบนารีบิท 5 บิท (chrom = 5) ดังนั้นโครโมโซมของปัญหาจะมีค่าอยู่ในช่วง 00000 ถึง 11111 ซึ่งเมื่อถอดรหัสแล้วจะทำให้ x มีค่าอยู่ในช่วง 0 ถึง 31 ตามที่ต้องการ

ในส่วนของรายระเอียดขั้นตอนการทำงานของ SGA จะเป็นขั้นตอนพื้นฐานเบื้องต้นแบบง่ายประกอบด้วย

1. ประชากรรุ่นเก่า (Old Population) เป็นชุดโครโมโซมที่ถูกคัดเลือกไปเป็นต้นแบบสำหรับสร้างประชากรรุ่นใหม่ (New Population) ในวิวัฒนาการรุ่น (generation : gen) ต่อไป โดยประชากรเริ่มต้นที่ gen = 0 จะถูกสร้างขึ้นโดยการสุ่มตามจำนวนโครโมโซมในแต่ละรุ่น (Population Size : popsize) ที่กำหนด

ตัวอย่าง :

ลำดับ	โครโมโซม
1	01110
2	11001
3	01000
4	10011

ชุดโครโมโซมในรุ่นเริ่มต้นนี้เป็นชุดโครโมโซมที่กำหนดให้ในแต่ละรุ่นประกอบด้วย 4 โครโมโซม ซึ่งแต่ละโครโมโซมเกิดจากการสุ่มค่าไบนารี 0 หรือ 1 จำนวน 5 ครั้ง

2. วิเคราะห์ค่าความเหมาะสม เป็นขั้นตอนการถอดรหัสจากรูปแบบโครโมโซมที่กำหนดไว้ เพื่อคำนวณค่าความเหมาะสมตามฟังก์ชันความเหมาะสมของปัญหาของแต่ละโครโมโซม ในที่นี้ฟังก์ชันเป้าหมายหรือฟังก์ชันความเหมาะสม คือ $F = x^2$ ดังนั้นการวิเคราะห์ค่าความเหมาะสมของ SGA โดยถอดรหัสเลขฐาน 2 ของแต่ละโครโมโซมเป็นค่าตัวแปร x และคำนวณค่าความเหมาะสมคือค่า x^2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวอย่าง :

ลำดับ	โครโมโซม	x	ค่าความเหมาะสม
1	01110	18	196
2	11001	25	625
3	01000	8	64
4	10011	19	361

ชุดโครโมโซมในรุ่นเริ่มต้นมีค่าความเหมาะสมเป็น 196,625, 64 และ 361 ตามลำดับ

3. การคัดเลือก เป็นขั้นตอนที่จำลองแบบการคัดเลือกทางธรรมชาติเพื่อสร้าง mating pool โดยการเลือกชุดโครโมโซมรุ่นเก่าให้เป็นโครโมโซมต้นแบบหรือโครโมโซมพ่อ-แม่ เพื่อใช้สร้างโครโมโซมลูกเป็นรุ่นต่อไป สำหรับการคัดเลือกของ SGA เป็นแบบอ้างอิงค่าความเหมาะสม (Fitness-based Selection) โดยพิจารณาค่าความเหมาะสมเป็นตัวตัดสินว่า โครโมโซมใดในรุ่นเก่ามีโอกาสจะถูกเลือกเป็นโครโมโซมพ่อ-แม่มากน้อยเพียงใด โครโมโซมที่มีความเหมาะสมที่ดีจะถูกกำหนดน้ำหนักค่าความน่าจะเป็นที่จะถูกเลือกแต่ละครั้งสูง การกำหนดค่าความน่าจะเป็นที่จะถูกเลือกต่อการสุ่มเลือกแต่ละครั้ง (Probability of Selected Value : pselect) ของแต่ละโครโมโซมโดยกำหนดจากค่าความเหมาะสม เทียบกับผลรวมของค่าความเหมาะสมทั้งหมด ดังสมการที่ 2.1

$$pselect_i = \frac{F_i}{\sum F} \qquad (2.1)$$

ซึ่งสามารถคำนวณค่าที่คาดหวังว่าจะสุ่มได้ (Expected Value : E) ของแต่ละโครโมโซมในแต่ละรุ่นดังสมการที่ 2.2

$$E_i = pselect_i * popsize = \frac{F_i}{F} \qquad (2.2)$$

สำหรับวิธีการสุ่มโครโมโซมต้นแบบของ SGA เป็นแบบจำลองการหมุนวงล้อถ่วงน้ำหนัก (Roullet Wheel : RW) ซึ่งกำหนดค่าแต่ละช่วงของวงล้อนั้นตามค่าความน่าจะเป็นที่จะสุ่มได้ในแต่ละครั้งของแต่ละโครโมโซม ซึ่งมีวิธีการดังนี้

(1) หาค่าความเหมาะสมของแต่ละโครโมโซม

(2) หาค่าความน่าจะเป็นที่จะสุ่มได้ในแต่ละครั้งในแต่ละโครโมโซม

(3) หาความถี่สะสม (q) ของความน่าจะเป็นของแต่ละโครโมโซมดังสมการที่ 2.3

$$q_i = \sum_{j=1}^{i} pselect_j \qquad (2.3)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(4) สร้างเลขสุ่มจำนวนจริง (r) ที่มีค่าอยู่ในช่วง [0.0,1.0]

(5) เลือกโครโมโซมลำดับที่ r ซึ่ง r มีค่าอยู่ระหว่าง q_{i-1} และ q_i

ตัวอย่าง :

ลำดับ	โครโมโซม	x	ค่าความเหมาะสม (F)	ค่าความน่าจะเป็น ($pselect_i$)	จำนวนที่คาดหวัง (E_i)	จำนวนที่สุ่มได้จากRW
1	01110	14	196	0.157	0.628	1
2	11001	25	625	0.502	2.008	2
3	01000	8	64	0.051	0.204	0
4	10011	19	361	0.290	1.160	1
	รวม		1246	1.000	4.000	
	ค่าเฉลี่ย		312	0.250	1.000	
	ค่าสูงสุด		625	0.502	2.008	

ตัวอย่างการกำหนดค่าความน่าจะเป็น โดยกำหนดจากค่าความเหมาะสม เทียบกับผลรวมของค่าความเหมาะสมทั้งหมด จะเห็นได้ว่าในการคัดเลือกโครโมโซมต้นแบบจาก 4 โครโมโซมนี้ โอกาสที่จะสุ่มได้โครโมโซมลำดับที่ 1 ต่อการสุ่มแต่ละครั้งเท่ากับ 0.157 และโอกาสที่จะสุ่มได้โครโมโซมลำดับที่ 2,3,4 ต่อการสุ่มแต่ละครั้งเท่ากับ 0.502 ,0.051 และ 0.290 ตามลำดับ และจำนวนโครโมโซมต้นแบบที่สุ่มได้โดยการจำลองหมุนวงล้อดังนี้

ลำดับโครโมโซม	1	2	3	4
ค่าความเหมาะสม (F)	196	625	64	361
ค่าความน่าจะเป็นที่จะสุ่มได้แต่ละครั้ง ($pselect_i$)	0.157	0.502	0.051	0.290
ความถี่สะสมค่าความน่าจะเป็น (q_i)	0.157	0.659	0.710	1.000
สร้างเลขสุ่มในการหมุนวงล้อแต่ละครั้ง (r)	0.333	0.844	0.456	0.128
ลำดับโครโมโซมที่ถูกเลือก ($q_{i-1} \leq r \leq q_i$)	2	4	2	1

ซึ่งจำนวนที่สุ่มได้เป็นโครโมโซมต้นแบบใน mating pool ของแต่ละโครโมโซมเป็น 1,2,0 และ 1 ตามลำดับ จะเห็นได้ว่าโครโมโซมลำดับที่ 2 มีค่าความเหมาะสมสูงที่สุดจะมีโอกาสถูกคัดเลือกในจำนวนที่มากที่สุด ส่วนโครโมโซมลำดับที่ 3 มีค่าความเหมาะสมต่ำมากจึงมีโอกาสที่จะไม่ถูกคัดเลือกเลย

ห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. ดำเนินการทางพันธุศาสตร์ เป็นขั้นตอนที่จำลองแบบธรรมชาติทางพันธุกรรม ซึ่งตัวดำเนินการทางพันธุศาสตร์ของ SGA คือ ครอสโอเวอร์และมิวเทชัน ซึ่งมีรายระเอียดดังนี้

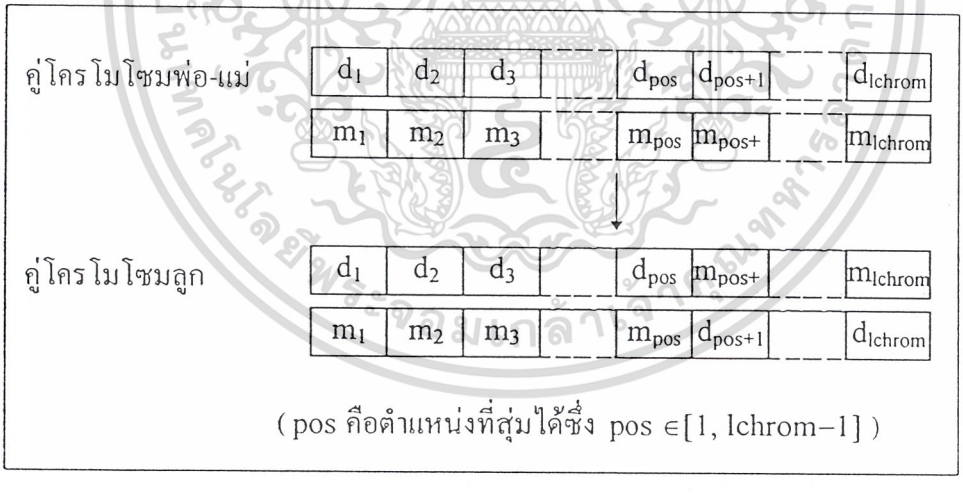
4.1 ครอสโอเวอร์ เป็นตัวดำเนินการในการแลกเปลี่ยนส่วนของโครโมโซมพ่อ-แม่ ตามการกำหนดอัตราความน่าจะเป็นของการครอสโอเวอร์ (Probability of Crossover : P_c) เพื่อสร้างชุดโครโมโซมใหม่หรือโครโมโซมลูก มีขั้นตอนการทำงานคือ

ขั้นตอนแรก : สุ่มจับคู่โครโมโซมพ่อ-แม่ใน mating pool ที่สร้างไว้จากการคัดเลือก

ขั้นตอนสอง : สร้างเลขสุ่มจำนวนจริง (r) มีค่าอยู่ในช่วง [0.0 , 1.0] โดยถ้า $r \leq P_c$ แล้ว โครโมโซมพ่อ-แม่นั้นจึงมีการครอสโอเวอร์

ขั้นตอนสาม : ครอสโอเวอร์โดยแลกเปลี่ยนส่วนของคู่โครโมโซมพ่อ-แม่นั้น ซึ่งการครอสโอเวอร์ของ SGA เป็นการครอสโอเวอร์แบบ 1 จุด (One-point Crossover) แสดงดังรูปที่ 2.4 ดังนี้

- สุ่มเลือกตำแหน่ง pos เป็นตำแหน่งที่จะครอสโอเวอร์ ซึ่ง pos มีค่าอยู่ในช่วง [1, lchrom-1]
- แลกเปลี่ยนค่าในแต่ละบิทของคู่โครโมโซมพ่อ-แม่ตั้งแต่ตำแหน่งที่ pos+1 ถึง lchrom ซึ่งจะทำให้เกิดโครโมโซมลูกใหม่ 2 โครโมโซม



รูปที่ 2.4 ครอสโอเวอร์แบบ 1 จุด

จำนวนการครอสโอเวอร์ในแต่ละรุ่นดำเนินการขึ้นอยู่กับการกำหนดค่า P_c ซึ่งแตกต่างกันในแต่ละปัญหา เช่น ถ้าจำนวนประชากรแต่ละรุ่น popsize เท่ากับ 30 โครโมโซม และกำหนดให้ $P_c = 0.6$ แล้วจำนวนการครอสโอเวอร์แต่ละรุ่นเท่ากับ $P_c*(popsize/2) = 0.6*(30/2) = 9$ ครั้ง (การครอสโอเวอร์ 1 ครั้งเกิดจากโครโมโซม 2 โครโมโซม)

ลิขสิทธิ์มหาวิทยาลัยเชียงใหม่
Copyright© by Chiang Mai University
All rights reserved
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

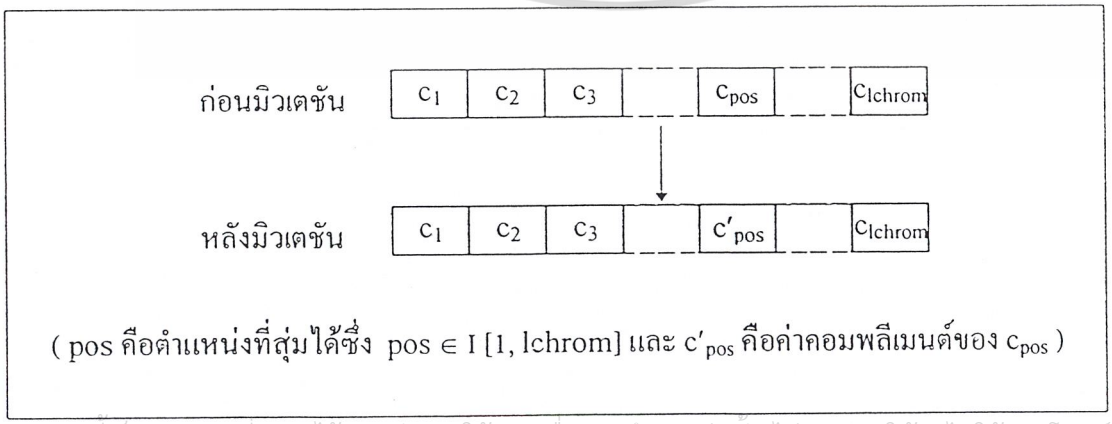
สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

ตัวอย่าง : กำหนด $P_c = 0.5$ โครโมโซมพ่อ-แม่ใน mating pool จากการคัดเลือกครอสโอเวอร์ดังนี้

ลำดับที่คัดเลือก	mating pool	สุ่มจับคู่พ่อ-แม่	เลขสุ่ม (r)	ก่อนครอสโอเวอร์	สุ่มตำแหน่ง (pos)	หลังครอสโอเวอร์	x	ค่าความเหมาะสม (F)	ลำดับโครโมโซมลูก
2	1 1 0 0 1	1,2	0.321	0 1 1 1 0	2	0 1 0 0 1	9	81	1
4	1 0 0 1 1		≤ 0.5	1 1 0 0 1		1 1 1 1 0	30	*900*	2
2	1 1 0 0 1	2,4	0.654	ไม่ครอสโอเวอร์		1 1 0 0 1	25	625	3
1	0 1 1 1 0		> 0.5			1 0 0 1 1	19	361	4
	รวม							1967	
	ค่าเฉลี่ย							492	
	ค่าสูงสุด							900	

จากการสุ่มจับคู่โครโมโซมพ่อ-แม่ใน mating pool ได้โครโมโซมลำดับที่ 1 คู่ลำดับที่ 2 และลำดับที่ 2 จับคู่ลำดับที่ 4 แต่เฉพาะโครโมโซมคู่แรกจะเกิดครอสโอเวอร์ เนื่องจากเลขสุ่ม $r \leq 0.5$ ตามอัตราครอสโอเวอร์ที่กำหนด โดยตำแหน่งในการครอสโอเวอร์ที่สุ่มได้คือ pos = 2 จะเห็นได้ว่าโครโมโซมลำดับที่ 2 ที่เกิดขึ้นหลังจากครอสโอเวอร์มีค่าความเหมาะสมดีขึ้นกว่าโครโมโซมพ่อ-แม่ทั้งหมดในรุ่นก่อนเป็น 900 ซึ่งแสดงให้เห็นถึงการจำลองแบบกระบวนการครอสโอเวอร์ตามธรรมชาติทางพันธุศาสตร์ของ SGA ช่วยสร้างคำตอบที่ดีขึ้น

4.2 มิวเทชัน เป็นตัวดำเนินการผ่าเหล่าตัวหนึ่งที่อาจช่วยให้โครโมโซม มีค่าความเหมาะสมดีขึ้นหลังจากครอสโอเวอร์ โดยกลับค่าของบิทเป็นค่าใหม่ในตำแหน่งบิทที่สุ่มได้ ตามอัตราความน่าจะเป็นของการมิวเทชันในแต่บิท (Probability of Mutation : P_m) ที่กำหนด สำหรับการมิวเทชันของ SGA นั้นเป็นแบบไบนารีมิวเทชัน (Binary Mutation) โดยกลับค่าบิทเป็นค่าคอมพลีเมนต์คือจาก 0 เป็น 1 หรือจาก 1 เป็น 0 ดังรูปที่ 2.5



(pos คือตำแหน่งที่สุ่มได้ซึ่ง pos ∈ I [1, lchrom] และ c'_{pos} คือค่าคอมพลีเมนต์ของ c_{pos})

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 2.5 ไบนารีมิวเทชัน

34169

จำนวนการมิวเทชันในแต่ละรุ่นขึ้นอยู่กับการกำหนดค่า P_m ซึ่งแตกต่างกันในแต่ละปัญหา เช่น ถ้าจำนวนประชากรในแต่ละรุ่น popsize เท่ากับ 30 โครโมโซม ซึ่งแต่ละโครโมโซมประกอบด้วย 5 บิท และกำหนดให้ $P_m = 0.02$ แล้ว จำนวนการมิวเทชันในแต่ละรุ่นเท่ากับ P_m*popsize*lchrom = 0.02*30*5 = 3 บิท

ตัวอย่าง : กำหนด $P_m = 0.1$ ดำเนินการมิวเทชันที่ได้จากการครอสโอเวอร์ดังนี้

ลำดับ	ก่อน มิวเตชัน	เลขสุ่ม (r)					หลัง มิวเตชัน					x	ค่าความเหมาะสม (F)
1	0 1 0 0 1	0.581	0.346	0.062	0.785	0.401	0	1	1	0	1	13	169
2	1 1 1 1 0	0.829	0.534	0.947	0.308	0.277	1	1	1	1	0	30	900
3	1 1 0 0 1	0.398	0.646	0.494	0.765	0.029	0	1	0	0	0	24	576
4	1 0 0 1 1	0.175	0.335	0.837	0.577	0.308	1	1	1	1	0	19	361
	รวม												2006
	ค่าเฉลี่ย												*502*
	ค่าสูงสุด												900

จากการสุ่มตำแหน่งที่จะมิวเทชันโดยสร้างเลขสุ่ม r ของแต่ละ ตำแหน่งบิทในแต่ละโครโมโซมแล้ว ตำแหน่งบิทที่ 3 ของโครโมโซมลำดับที่ 1 และตำแหน่งบิทที่ 4 ของโครโมโซมลำดับที่ 3 เป็นตำแหน่งที่ $r \leq 0.1$ ตามอัตราการที่กำหนดจึงจะเกิดการมิวเทชัน ทำให้โครโมโซมมีค่าความเหมาะสมจาก 81 และ 625 เป็น 169 และ 576 ตามลำดับ จะเห็นได้ว่ามิวเทชันเป็นตัวดำเนินการที่อาจทำให้โครโมโซมมีค่าความเหมาะสมสูงขึ้นหรือลดลงได้ แต่อย่างไรก็ตามทำให้ค่าความเหมาะสมดีขึ้นจาก 492 เป็น 502 แสดงถึงการหาคำตอบของ SGA โดยส่วนมากดีขึ้น และคาสมสำคัญของการหาคำตอบของ GA นั้นเป็นความต้องการได้คำตอบจากคำตอบที่ดีขึ้นเกิดขึ้น ซึ่งจะมีโอกาสอยู่รอดเพื่อถ่ายทอดส่วนที่ดีต่อไป

5. ประชากรรุ่นใหม่ เป็นชุดโครโมโซมลูกที่เกิดจากขั้นตอนของวิวัฒนาการต่างๆ ทั้งหมด ซึ่งประชากรรุ่นใหม่ทั้งหมดที่เกิดขึ้น จะถูกถ่ายทอดไปเป็นประชากรรุ่นเก่าสำหรับวิวัฒนาการรุ่นถัดไป ซึ่งเรียกวิวัฒนาการแบบนี้ว่า การถ่ายทอดแบบทั่วไปหรือรีโพรดักชันแบบทั่วไป(General Reproduction) กระบวนการต่างๆ จะถูกปฏิบัติซ้ำๆ จนกระทั่งถึงรุ่นที่มากที่สุด(max generation: max gen)ที่ต้องการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7 การปรับปรุงเจเนติกอัลกอริทึม

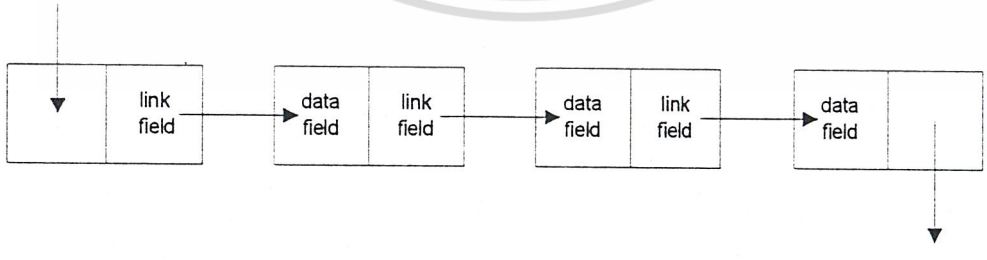
กลไกของ GA ไม่ได้ถูกควบคุมด้วยสมการดิฟเฟอเรนเชียลหรือมีพฤติกรรมเหมือนกับฟังก์ชันชนิดต่อเนื่องทั่วไป กลไกของ GA มีความสามารถที่เป็นเอกลักษณะเฉพาะตัวในการออฟติไมซ์ระบบที่มีความซับซ้อน และไม่สามารถใช้วิธีการคำนวณด้วยสมการคณิตศาสตร์แบบปกติได้ การใช้ GA แบบมาตรฐานอาจไม่มีความยืดหยุ่นในการนำไปใช้งานจริง เนื่องจากในทางวิศวกรรมมักมีข้อกำหนดต่างๆมากมายสำหรับการออกแบบ ปัญหานี้จะเห็นเด่นชัดมาก หากปัญหาที่ทำการแก้ไขมีความซับซ้อน มีข้อจำกัดต่างๆมากมาย หรือมีลักษณะการทำงานแบบมัลติทาส์กกิ้ง (multi tasking) เนื้อหาในบทนี้จึงได้แนะนำวิธีการต่างๆในการปรับปรุง GA ที่จำเป็น

2.7.1 การแทนโครโมโซม

การเข้ารหัสโครโมโซมอาจมีได้หลายวิธีขึ้นกับลักษณะของปัญหาที่ทำการแก้ไข วิธีการเข้ารหัสแบบไบนารีเป็นวิธีการหนึ่งที่ใช้กันอย่างกว้างขวาง เนื่องจากว่ามีความง่ายและสามารถติดตามผลได้ง่าย แต่มีผลการวิจัยที่ได้ทดลองจะพบว่า การเข้ารหัสแบบเกรย์ (Gray code) ให้ผลทำงานที่ดีกว่าการเข้ารหัสแบบไบนารีอยู่เล็กน้อย

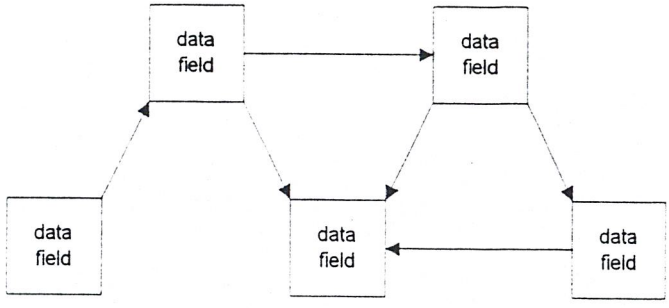
ปัจจุบัน การจัดการกับโครโมโซมโดยใช้ค่าจริง ได้รับความสนใจมากขึ้น ซึ่งการแทนด้วย floating point จะให้ผลลัพธ์ที่รวดเร็วกว่าวิธีอื่น นอกจากนี้ประสิทธิภาพในการทำงานของ GA ยังขึ้นอยู่กับ ความสามารถของตัวดำเนินการเจเนติกพิเศษ แต่อย่างไรก็ตามในบางกรณีการใช้รหัสค่าจริงอาจไม่ได้ผลดีเท่าที่ควรก็ได้

สำหรับปัญหาที่มีการค้นหาในลักษณะมีลำดับ มักใช้การแทนโครโมโซมที่พิจารณาจากลำดับ การแทนโครโมโซมด้วยวิธีนี้แบบง่ายที่สุดคือ ลิงค์ลิสแบบเชิงเส้น ซึ่งแสดงในรูปที่ 2.6



รูปที่ 2.6 ลิงค์ลิสแบบเชิงเส้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.7 การแทนโครโมโซมแบบกราฟ

ในกรณีที่ความยาวโครโมโซมมีการเปลี่ยนแปลงได้ อาจใช้แทนโครโมโซมแบบกราฟดังในรูปที่ 2.7 ข้อดีของการเข้ารหัสโดยพิจารณาจากลำดับเมื่อเปรียบเทียบกับการเข้ารหัสแบบตัวอักษร คือ สามารถหลีกเลี่ยงคำตอบที่ไม่ถูกต้องที่เกิดจากการครอสโอเวอร์ได้

ในบางกรณี อาจมีการใช้ดัชนีเป็นโครโมโซมแทนการใช้ค่าจริง ในลักษณะของ look-up table ซึ่งวิธีนี้จะมีประโยชน์สำหรับการคัดเลือกที่ไม่เป็นเชิงเส้น

จากที่กล่าวมาทั้งหมด จะพบว่าการปรับปรุงวิธีการแทนโครโมโซมมีมากมายหลายวิธี ซึ่งการเลือกใช้วิธีใดนั้นจะต้องเลือกให้เหมาะสมกับแต่ละปัญหา เพราะการเลือกรูปแบบที่เหมาะสมจะช่วยเพิ่มประสิทธิภาพและลดภาระในการคำนวณเมื่อมีการนำไปใช้งานจริง

2.7.2 สเกลลิ่งแบบเชิงเส้น

ค่าฟิทเนส f_i ของแต่ละโครโมโซมที่ทำสเกลลิ่ง จะมีค่าความสัมพันธ์อย่างเป็นเชิงเส้นกับค่าฟิทเนสเดิม f_i' ดังนี้

$$f_i = af_i' + b \qquad (2.4)$$

โดยที่ a และ b ถูกเลือกค่าที่ให้ ค่าเฉลี่ยฟิทเนสเดิมเท่ากับค่าเฉลี่ยฟิทเนสใหม่หลังจากทำสเกลลิ่ง และค่าฟิทเนสในการทำสเกลสูงสุดถูกกำหนดจากค่าคงที่ที่คูณกับค่าเฉลี่ยฟิทเนส

วิธีการทำสเกลลิ่งแบบเชิงเส้นนี้จะช่วยลดผลกระทบจากการลู่เข้าก่อนกำหนด เนื่องจากบางโครโมโซมมีค่าฟิทเนสสูงกว่าปกติจากโครโมโซมอื่นๆ แต่อย่างไรก็ตามจะต้องหลีกเลี่ยงการเกิดค่าฟิทเนสที่อาจมีค่าเป็นลบจากการใช้สเกลลิ่งวิธีนี้ ดังนั้น การเลือกค่า a และ b จึงต้องพิจารณาย่านของค่าฟิทเนสเดิมด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.7.3 สเกลลิ่งด้วยกฎยกกำลัง

การทำสเกลลิ่งวิธีนี้ ทำได้โดยการยกกำลังฟังก์ชันวัตถุประสงค์ O_i

$$f_i = o_i^k \qquad (2.5)$$

เมื่อ k มีค่าขึ้นอยู่กับปัญหาที่ต้องการแก้ไขหรืออาจมีค่าเปลี่ยนแปลงค่าได้ขณะทำการรัน

2.7.4 แรงค์กิ้ง

วิธีค่าฟิทเนสไม่ได้มีความสัมพันธ์โดยตรงกับค่าของฟังก์ชันวัตถุประสงค์ แต่จะเป็นการจัดลำดับ (rank) ของค่าฟังก์ชันวัตถุประสงค์

การใช้วิธีการนี้จะช่วยหลีกเลี่ยงการเกิดการลู่เข้าก่อนกำหนด และช่วยเพิ่มความเร็วในการค้นหาคำตอบเมื่อประชากรใกล้คอนเวอร์เจนซ์ (convergence)

2.7.5 วิธีการคัดเลือก

การคัดเลือกเป็นกระบวนการที่ใช้ในการเลือกสมาชิกแต่ละตัวในประชากรสำหรับกระบวนการรีโพรดักชั่น โครโมโซมลูกหลานที่ดีเกิดขึ้นได้จากกลไกการคัดเลือกโครโมโซมพ่อแม่ที่ดี ดังนั้นจึงมีมาตรการการวัดการทำงานของอัลกอริทึมการคัดเลือกดังนี้

- ความลำเอียง (Bias) คือ ค่าสัมบูรณ์ของความแตกต่างระหว่างค่าจริงกับความน่าจะเป็นที่คาดหวังไว้สำหรับการคัดเลือกของสมาชิกแต่ละตัว ความลำเอียงจะมีค่าเป็นศูนย์เมื่อความน่าจะเป็นในการถูกเลือกของสมาชิกแต่ละตัวเท่ากับค่าความน่าจะเป็นที่คาดหวังๆไว้

- การกระจาย (Speed) คือ ย่านของจำนวนทดลองที่เป็นไปได้ที่สมาชิกตัวหนึ่งจะถูกเลือก ถ้า ตัวอย่างเช่น g(i) คือ จำนวนทดสอบของสมาชิกลำดับที่ i การกระจายต่ำสุด คือ การกระจายที่น้อยที่สุดที่ยินยอมให้มีความลำเอียงเป็นศูนย์ตามทฤษฎี

$$g(i) \in \{\lfloor et(i) \rfloor, \lceil et(i) \rceil\} \qquad (2.6)$$

เมื่อ $et(i)$ คือ จำนวนทดสอบที่คาดหวังไว้ของสมาชิกแต่ละตัวที่ i, $\lfloor et(i) \rfloor$ คือ ขอบเขตล่าง และ คือ $\lceil et(i) \rceil$ขอบเขตบน ดังนั้น การกระจายจะเป็นตัววัดความเที่ยงของวิธีการคัดเลือก

- ประสิทธิภาพ จะขึ้นอยู่กับความซุบซ้อนของเวลาทั้งหมดสำหรับอัลกอริทึม

วิธีการคัดเลือกที่ดีควรจะมีความลำเอียงเป็นศูนย์ มีการกระจายต่ำ และไม่เพิ่มความซับซ้อนของเวลาในการประมวลผล GA มากเกินไป

เทคนิคการคัดเลือกต่างๆมักจะใช้กลไกของวงล้อรูเลท วิธีการคัดเลือกแบบวงล้อรูเลท

เอกสารนี้สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อย่างง่าย คือ การเลือกแบบสุ่มโดยมีการแทนที่(SSR : Stochastic Sampling with Replacement) วิธีนี้ขนาดเซกเมนท์และความน่าจะเป็นในการคัดเลือกจะคงที่ตลอดช่วงการคัดเลือก SSR มีแนวโน้มให้ความลำเอียงเป็นศูนย์แต่มีแนวโน้มที่เพิ่มการกระจายซึ่งมีไม่จำกัด

การเลือกแบบสุ่มโดยวิธีการแทนที่บางส่วน(SSR : Stochastic Sampling with Partial Replacement) เป็นวิธีการที่มีการปรับขนาดเซกเมนท์ของโครโมโซมถ้าโครโมโซมนั้นมีการถูกเลือก ในแต่ละครั้งที่โครโมโซมถูกเลือกจะมีการลดขนาดเซกเมนท์ด้วยแฟกเตอร์ค่าหนึ่ง ถ้าขนาดเซกเมนท์ของโครโมโซมมีค่าเป็นลบ ขนาดเซกเมนท์ก็จะถูกกำหนดเป็นศูนย์ วิธีนี้จะกำหนดขอบเขตบนของการกระจาย $\lceil et(i) \rceil$ แต่ขอบเขตล่างจะมีค่าศูนย์และมีความลำเอียงที่สูงขึ้น วิธีการคัดเลือกแบบวงล้อรูเลทมีความซับซ้อนของเวลาเป็น NlogN เมื่อ N คือ ขนาดของประชากร

การคัดเลือกแบบยูนิเวอร์ซอลโดยสุ่ม (SUS : Stochastic Universal Sampling) เป็นวิธีที่มีการกระจายต่ำ ความลำเอียงเป็นศูนย์ และความซับซ้อนของเวลาเป็น N วิธีคัดเลือกแบบ SUS จะใช้ตัวชี้ N ตัวที่มีระยะห่างเท่ากันในการเลือกประชากรในครั้งเดียว ประชากรจะถูกสลับแบบสุ่มและจะสุ่มตัวเลขค่าหนึ่งในช่วง $[0,F_{SUM}/N]$ เป็น ptr ดังนั้นตัวทั้ง N ตัวจะมีค่าเป็น $[ptr,ptr+F_{SUM}/N,\ldots,ptr+(N-1)F_{SUM}/N]$ โครโมโซมที่ถูกเลือกเป็นโครโมโซมที่ครอบคลุมพื้นที่ในตำแหน่งของตัวชี้

วิธีการคัดเลือกแบบวงล้อรูเลทเป็นวิธีการที่ใช้แพร่หลายมากที่สุดโดยอาศัยกลไกการคัดเลือกตามสัดส่วน ขั้นตอนต่างๆสรุปไว้ในตารางที่ 2.2

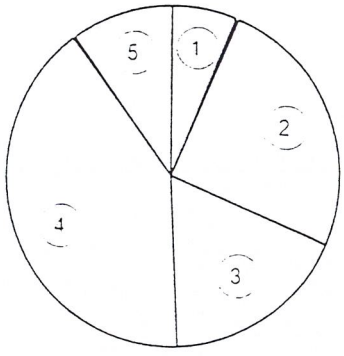
ตารางที่ 2.2 ขั้นตอนการคัดเลือกวงล้อรูเลท

-บวกค่าฟิทเนสของสมาชิกทุกตัว (F_{SUM})
-สร้างตัวเลขโดยสุ่ม (n) ระหว่าง 0 และ F_{SUM}
-กลับไปเริ่มที่สมาชิกตัวแรก แล้วบวกค่าฟิทเนสของสมาชิกถัดไป จนกระทั้งผลรวมฟิทเนสที่ได้มีค่ามากกว่าหรือเท่ากับ F_{SUM}

ตัวอย่างเช่น ในรูปที่ 2.8 เส้นรอบวงล้อรูเลท คือ F_{SUM} ของโครโมโซม 5 ตัว โครโมโซมที่ 4 มีค่าฟิทเนสสูงสุดจึงครอบครองเนื้อที่ในวงล้อรูเลทมากที่สุด ในขณะที่โครโมโซมที่ 1 มีค่าฟิทเนสต่ำสุด จึงครอบครองพื้นที่ในวงล้อรูเลทต่ำสุด ในการเลือกโครโมโซม จะเริ่มจากการสร้างตัวเลขสุ่มในช่วง $[0,F_{SUM}]$ เนื้อที่ของโครโมโซมตัวใดที่ครอบคลุมค่าเลขสุ่มนี้จะถูกเลือก

วงรอบการพัฒนาจะเกิดซ้ำไปเรื่อยๆ จนกระทั่งเงื่อนไขการหยุดที่กำหนดเป็นจริง ซึ่งเงื่อนไขการหยุดนี้อาจเป็น จำนวนเจนเนอเรชันสูงสุด ความเบี่ยงเบนของสมาชิกระหว่างเจนเนอเรชัน หรือค่าฟิทเนสที่มีการกำหนดไว้ล่วงหน้า

ลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ห้ามมิให้ทำซ้ำ ดัดแปลง เผยแพร่ หรือแสวงหาประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น โดยไม่ได้รับอนุญาตเป็นลายลักษณ์อักษร ผู้ใช้งานสามารถนำไปใช้เพื่อการศึกษาเท่านั้น ต้องอ้างอิงถึงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.8 การคัดเลือกแบบวงล้อรูเลท

2.8 ตัวดำเนินการเจเนติก

2.8.1 การครอสโอเวอร์

เพื่อให้เกิดการพัฒนาในแต่ระวงรอบของ GA จะมีการใช้ตัวดำเนินการเจเนติก คือ การครอสโอเวอร์และการมิวเทชัน ซึ่งการคัดเลือกอาจพิจารณาว่าเป็นตัวดำเนินการเจเนติกตัวหนึ่งก็ได้ รูปที่ 2.9 แสดงให้เห็นกระบวนการครอสโอเวอร์แบบ 1 จุด โดยตำแหน่งครอสโอเวอร์จะถูกเลือกโดยสุ่ม ส่วนของโครโมโซมที่อยู่ถัดไปทางขวาจากจุดครอสโอเวอร์จะมีการแลกเปลี่ยนกันเพื่อสร้างโครโมโซมลูกหลาน อัตราการทำงาน (Operation rate) หรือความน่าจะเป็นของการครอสโอเวอร์ (P_c) ปกติจะมีค่าระหว่าง 0.6 ถึง 1.0

รูปที่ 2.9 การครอสโอเวอร์แบบ 1 จุด

แม้ว่าการครอสโอเวอร์แบบ 1 จุด เป็นวิธีการที่ได้แนวคิดมาจากกระบวนการทางชีววิทยา แต่ก็มีข้อเสียในบางสถานการณ์ที่สคีมาบางตัวไม่สามารถที่จะรวมกันได้ ตัวอย่างเช่น มีสคีมา 2 ตัว คือ S1 และ S2

S1 = 101****1

S2 = ****11**

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

มีโครโมโซมอยู่ 2 ตัว คือ I1 และ I2 ที่ตรงกับ S1 และ S2 ตามลำดับ

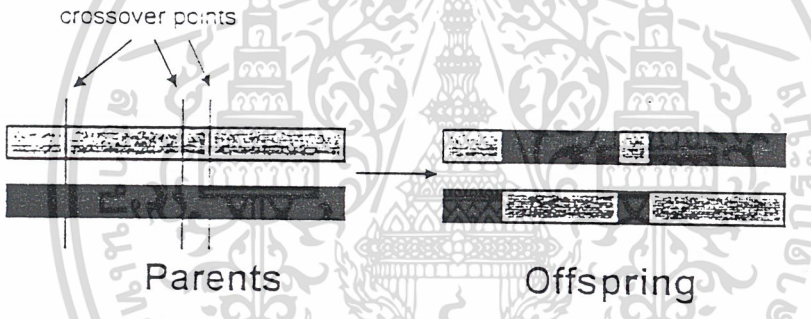
I1 = 10110001

I2 = 01101100

ถ้าใช้การครอสโอเวอร์แบบ 1 จุด จะไม่มีทางเป็นไปได้เลยที่จะมีโครโมโซมตรงกับสคีมา S3

S3 = 101*11*1

การครอสโอเวอร์แบบหลายจุดจะช่วยแก้ปัญหานี้ได้ โดยจะช่วยเพิ่มความสามารถในการสร้างโครโมโซมลูกหลาน ตัวอย่างการทำงานของการครอสโอเวอร์แบบหลายจุดนี้แสดงในรูปที่ 2.10 โดยที่ตำแหน่งของการครอสโอเวอร์จะถูกเลือกโดยวิธีการสุ่ม



รูปที่ 2.10 ตัวอย่างการครอสโอเวอร์แบบหลายจุด

เพื่อให้เข้าใจในการเพิ่มความสามารถของการสร้างโครโมโซมลูกหลานของการครอสโอเวอร์แบบหลายจุด จะยกตัวอย่างเช่น สมมุติว่ามีการใช้การครอสโอเวอร์ I1 และ I2 แบบ 2 จุด ทำให้ได้โครโมโซมลูกหลาน คือ I3 และ I4 จะพบว่า I3 ตรงกับสคีมา S3

I1 = 1011|00|01

I2 = 0110|11|00

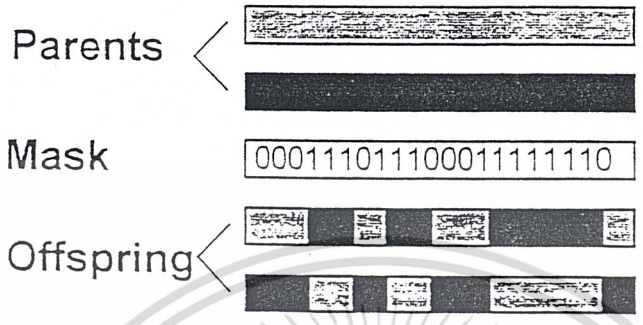
I3 = 1011 11 01

I4 = 0110 00 00

วิธีการครอสโอเวอร์อีกวิธี คือ การครอสโอเวอร์แบบยูนิฟอร์ม ซึ่งมีวิธีนี้จะสร้างโครโม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โซมลูกหลานโดยขึ้นอยู่กับการสร้าง mask โดยการสุ่ม การทำงานของการครอสโอเวอร์วิธีนี้แสดงในรูปที่ 2.11 จากรูปที่ 2.11 จะสังเกตได้ว่าการครอสโอเวอร์แบบยูนิฟอร์มจะมีการแลกเปลี่ยนกันระหว่างบิท มากกว่าทีจะแลกเปลี่ยนกันเป็นส่วน (segment) ทำให้การครอสโอเวอร์แบบยูนิฟอร์มมีประสิทธิภาพมากกว่าสำหรับบางปัญหา เช่น ช่วยลดการทำลายบิวล์ดิ้งบล็อก



รูปที่ 2.11 ตัวอย่างของการครอสโอเวอร์แบบยูนิฟอร์ม

จากตัวอย่างการครอสโอเวอร์แบบต่างๆที่ยกมา จะพบว่าการครอสโอเวอร์จะยังอยู่บนแนวคิดพื้นฐาน คือ ต้องการให้มีการแลกเปลี่ยนข้อมูลกันระหว่างโครโมโซม และมีข้อสังเกตการออกแบบวิธีการครอสโอเวอร์ที่มีประสิทธิภาพ จะช่วยเพิ่มการคอนเวอร์เจนซ์ให้เร็วขึ้น

2.8.2 การมิวเทชัน

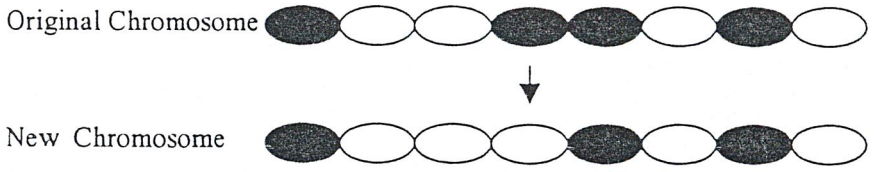
การมิวเทชันเป็นการสร้างความหลากหลายให้แก่โครโมโซม แต่โดยทั่วไปแล้วการมิวเทชันจะออกแบบไว้สำหรับโครโมโซมที่เข้ารหัสไว้แบบไบนารีเท่านั้น การมิวเทชันเป็นวิธีการหนึ่งที่ใช้โครโมโซมแบบจำนวนจริงโดยเป็นไปตามสมการ (2.7)

$$g = g + \psi(\mu, \sigma) \qquad (2.7)$$

เมื่อ g คือ ยีนส์ที่เป็นค่าจริง ψ คือฟังก์ชันสุ่ม ซึ่งอาจจะเป็น Gaussian หรือเป็นการสุ่มแบบกระจายปกติ(Normal distribution) μ,σ คือค่าเฉลี่ยและส่วนเบี่ยงเบนที่สัมพันธ์กับฟังก์ชันสุ่ม ตามลำดับ

สำหรับการมิวเทชัน จะถูกนำมาใช้หลังจากการทำครอสโอเวอร์ โดยจะทำการเปลี่ยนแปลงค่าบิทโดยสุ่ม ด้วยความน่าจะเป็นในการมิวเทชัน (P_m) ที่มีค่าน้อยๆ ซึ่งปกติจะมีค่าไม่เกิน 0.1 รูปที่ 2.12 แสดงการมิวเทชันของโครโมโซมในตำแหน่งบิทที่ 4

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.12 การมิวเทชันในตำแหน่งบิทที่ 4

2.8.3 การตั้งอัตราการทำงาน

การตั้งอัตราการทำงานหรือความน่าจะเป็นในการครอสโอเวอร์และมิวเทชัน ยังเป็นสิ่งที่มีการโต้แย้งกันอยู่ การเพิ่มความน่าจะเป็นในการครอสโอเวอร์ เป็นสาเหตุให้เกิดการรีคอมบิเนชันของบิวล์ดิ้งบล็อกเพิ่มขึ้น แต่ขณะเดียวกันก็จะเพิ่มการทำลายโครโมโซมที่ดีด้วย ในขณะที่การเพิ่มความน่าจะเป็นการมิวเทชัน จะทำให้การค้นหาคำตอบเป็นไปในลักษณะสุ่มมากขึ้นการเลือกค่า Pc และ P_m สำหรับการควบคุมกระบวนการ GA ไม่มีการกำหนดไว้แน่นอน ส่วนใหญ่แล้วจะขึ้นอยู่กับธรรมชาติของฟังก์ชันวัตถุประสงค์ แต่มีการแนะนำไว้เป็นแนวทางดังนี้

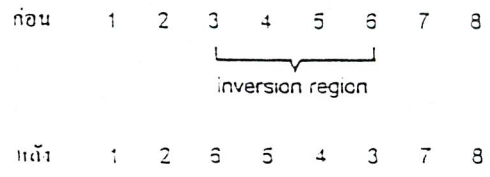
1. สำหรับประชากรขนาดใหญ่ (100)
   - อัตราการครอสโอเวอร์หรือความน่าจะเป็นในการครอสโอเวอร์ (P_c) ควรเป็น 0.6
   - อัตราการมิวเทชันหรือความน่าจะเป็นในการมิวเทชัน (P_m) ควรเป็น 0.001
2. สำหรับประชากรขนาดเล็ก (30)
   - อัตราการครอสโอเวอร์หรือความน่าจะเป็นในการครอสโอเวอร์ (P_c) ควรเป็น 0.9
   - อัตราการมิวเทชันหรือความน่าจะเป็นในการมิวเทชัน (P_m) ควรเป็น 0.01

การตั้งค่า P_c และ P_m ตามข้อแนะนำข้างบนนี้ มีข้อสังเกต คือ ความหลากหลายของประชากรเกิดขึ้นได้จาก จำนวนประชากรมากๆ หรือการใช้อัตราการครอสโอเวอร์และมิวเทชันสูงๆ แต่ในทางปฏิบัติจะต้องคำนึงถึงเวลาในการประมวลผลด้วย ดังนั้นเมื่อขนาดประชากรมากอัตราการครอสโอเวอร์และมิวเทชัน ก็ควรมีค่าต่ำกว่าอัตราการครอสโอเวอร์และมิวเทชันกรณีขนาดประชากรไม่มากนัก

2.8.4 การจัดลำดับใหม่

วัตถุประสงค์ของการจัดลำดับใหม่ เพื่อหาการพัฒนาลำดับของยีนส์ที่ทำให้การพัฒนาคำตอบที่ดีที่สุด กระบวนการนี้รู้จักกันดีในชื่อ อินเวอร์ชัน(Inversion) ขั้นตอนนี้เริ่มจากตำแหน่งที่จะมีการอินเวอร์ชัน ดังในรูปที่ 2.13 แสดงลำดับก่อนและหลังการทำอินเวอร์ชัน โดยตำแหน่งการทำอินเวอร์ชันถูกเลือกที่ 3 และ 6

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.13 ตัวอย่างการจัดลำดับใหม่

2.8.5 การแทนที่

ในกรณีการแทนที่แบบเจนเนอเรชัน(Generation replacement) หลังจากที่สร้างประชากรลูกหลานแล้วจะมีการแทนที่โครโมโซมในประชากรเจนเนอเรชันปัจจุบันด้วยโครโมโซมลูกหลานทุกตัว เช่น ถ้าหากขนาดประชากรมีค่าเท่ากับ N วิธีนี้จะต้องสร้างประชากรลูกหลานเท่ากับ N ด้วย วิธีการแทนที่ดังกล่าวนี้อาจทำให้โครโมโซมเสียหายในเจนเนอเรชันถัดไป ดังนั้นจึงมักมีการทำอิลิทิสซึม จะช่วยเพิ่มความเร็วในการหาคำตอบที่ดีที่สุดถ้าหากกำหนดจำนวนสำเนาโครโมโซมที่ดีที่สุดได้อย่างสมดุล

เนื่องจากว่าการสร้างประชากรลูกหลานที่มีขนาดใหญ่จะทำให้การคำนวณในแต่ละรอบของ GA นานขึ้น จึงมีวิธีการแทนที่อื่นๆ ที่สร้างประชาการลูกหลานจำนวนน้อยลงและช่วยเพิ่มความเร็วในการคำนวณ เช่น การแทนที่โครโมโซมพ่อแม่ด้วยโครโมโซม ลูกหลานโดยตรง โดยเลือกแทนที่โครโมโซมที่เลวที่สุดเมื่อมีการแทรกโครโมโซมใหม่เข้ามาในประชากร หรืออาจแทนที่โครโมโซมที่อาศัยอยู่ประชากรเป็นเวลานาน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
อุปกรณ์รักษาเสถียรภาพระบบไฟฟ้ากำลัง

3.1 อุปกรณ์แบบไฮบริด

อุปกรณ์แบบไฮบริด ประกอบด้วยอุปกรณ์ที่สำคัญดังนี้

3.1.1 อุปกรณ์ควบคุมแบบฟัซซี่ที่มีการเลื่อนแกนจากข้อมูลของ PD (Fuzzy Controller)

ในโครงงานนี้เราใช้ทฤษฎีฟัซซี่นำมาช่วยในการปรับปรุงเสถียรภาพของระบบไฟฟ้ากำลัง จากสัญญาณ ΔP_e ซึ่งจะผ่านเข้าไปในส่วนของ SPD (PD information of generation speed) จะได้เอาท์พุทเป็น Z_s และ Z_a ซึ่งค่า Z_sเป็นค่าของการเบี่ยงเบนความเร่ง ($Z_s(k)$) และ Z_a เป็นค่าใกล้เคียงความเร่ง $Z_a(k)$

การใช้เฟสเพลนในการหาค่า ให้นำเอา $Z_s(k)$ ให้อยู่ในแกนนอน (Horizontal axis) ค่า $\alpha_1 Z_a(k)$ อยู่ในแกนตั้ง (Vertical axis) ดังในรูปที่ 3.1

รูปที่ 3.1 เฟสเพลน (Plase plane) ที่มีการเลื่อนแกน

ในการปรับปรุงตัวรักษาเสถียรภาพของฟัซซี่นั้น ได้มีการเพิ่มในภาคของการควบคุมการลดความเร่งของระบบ โดยใช้ ZAA(k) และ ZSS(k) ซึ่งจะเป็นการเลื่อนแกนเพื่อให้ระบบเข้าสู่เสถียรภาพได้เร็วขึ้น

การเลื่อนขนาดของ ZSS(k) จะถูกปรับเพื่อเร่งเข้าสู่แกนในแนวตั้ง ซึ่งผลรวมของข้อมูล $Z_a(k)$ ความเร็วของเครื่องกำเนิดไฟฟ้าเป็นค่าบวก (Postive) กำหนดให้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$ZSS = -\alpha_4 Z_a \quad \text{เมื่อ} \quad Z_a \leq Z_{a\,min} \tag{3.1}$$

$$ZSS = 0.0 \quad \text{เมื่อ} \quad Z_a > Z_{a\,min} \tag{3.2}$$

การเลื่อนขนาดของ ZAA(k) เป็นการปรับเพื่อเร่งเข้าสู่แกนนอน ซึ่งผลการเบี่ยงเบนความเร่ง $Z_s(k)$ ของเครื่องกำเนิดไฟฟ้าเป็นลบ (Negative)

กำหนดให้

$$ZAA(k) = -\alpha_4 Z_a \quad \text{เมื่อ} \quad Z_s \leq Z_{s\,min} \tag{3.3}$$

$$ZAA = 0.0 \quad \text{เมื่อ} \quad Z_s > Z_{s\,min} \tag{3.4}$$

ซึ่ง $Z_{a\,min}$ และ $Z_{s\,min}$ เป็นช่วงกว้างตายตัว (Dead Bands) ที่พิจารณา ในรูปที่ 3.1 เราสามารถหาค่า $P_r(k)$ ของระบบจาก

$$P_r(k) = D_r(k) \angle \theta_r(k) \tag{3.5}$$

$$D_r(k) = \left[(Z_s(k)+ZSS(k)^2 + (\alpha_1 Z_a(k)+ZSS(k))^2 \right]^{1/2} \tag{3.6}$$

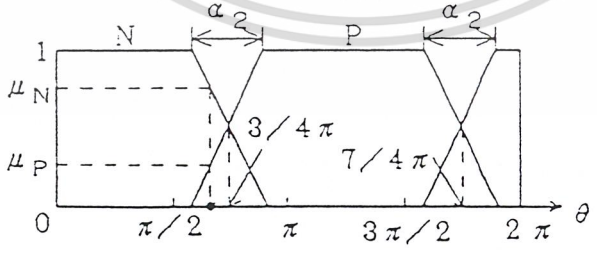
$$\theta_r(k) = \cos^{-1}\left[(Z_s(k)+ZSS(k))/D_r(k) \right] \tag{3.7}$$

เมื่อ P_r คือกำลังไฟฟ้าที่เบี่ยงเบนเมื่อนำมาลงบนเฟสเพลน

D_r คือขนาดของ P_r (magnetude)

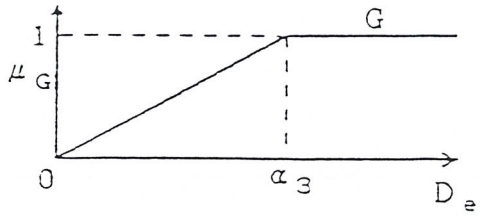
θ_r คือมุมของที่กระทำกับแกนการเบี่ยงเบนความเร็ว

สัญญาณควบคุม $U_1(k)$ ของ Fuzzy Controller หาได้จากวิธีการค่าถ่วงน้ำหนักเฉลี่ยของดิฟัซซี่ฟิเคชัน (Defuzziification) โดยใช้ฟังก์ชันการเป็นสมาชิกตามรูปที่ 3.2 และ 3.3



รูปที่ 3.2ฟังก์ชันการเป็นสมาชิกของ θ_r

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.3 ฟังก์ชันการเป็นสมาชิกของ D_r

สัญญาณควบคุม
$$U_1(k) = \frac{(U_N(k) - U_P(k))}{(U_N(k) + U_P(k))}(U_G(k)U_m) \quad (3.8)$$

และ
$$U_P(k) = 1 - U_N(k) \quad (3.9)$$

เขียนสมการโดยเพิ่มและ
$$U_P(k) = (2U_N(k) - 1)U_G(k)U_m \quad (3.10)$$

เมื่อ
k เป็นเวลาสุ่ม (Sampling Time)

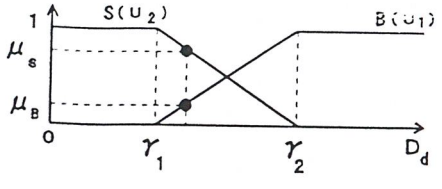
$U_N(k), U_P(k), U_G(k)$ เป็นฟังก์ชันระดับการเป็นสมาชิก

U_m เป็นค่าจำกัดสูงสุดของสัญญาณควบคุม

3.1.2 อุปกรณ์กลไกการตัดสินแบบฟัซซี่ (Fuzzy Judgement Mechanism : FJ)

กลไกการตัดสินของฟัซซี่ FJ เป็นตัวชี้บ่งขนาดของสัญญาณรบกวน และส่วนของระบบลักษณะเฉพาะของการแกว่ง โดยการใช้ขนาดของสัญญาณรบกวน ซึ่งกำหนดให้

$$D_d = \sqrt{Z_a^2 + Z_s^2} \quad (3.11)$$



รูปที่ 3.4 แสดงฟังก์ชันการเป็นสมาชิกของ FJ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ฟังก์ชันการเป็นสมาชิกของ D_d แสดงในรูปที่ 3.4 ได้ประยุกต์ใช้ข้อวินิจฉัยของฟัซซี่บนพื้นฐานหลักการซึ่ง

ถ้า $D_d(k)$ มีขนาดใหญ่ $U_1(k)$ จะถูกนำมาพิจารณา

ถ้า $D_d(k)$ มีขนาดเล็ก $U_2(k)$ จะถูกนำมาพิจารณา

ถ้า $D_d(k)$ อยู่ในส่วนกลางระหว่างทั้งสอง การรวมกันของทั้ง $U_1(k)$ และ $U_2(k)$ ถูกนำมาพิจารณา โดยที่

$$U_C(k) = U_2(k) + \mu_B(k)\left[U_1(k) - U_2(t)\right] \qquad (3.12)$$

ซึ่ง $U_C(k)$ เป็นสัญญาณของการเสถียรภาพแบบผลรวม (Hybrid Type) ที่นำเสนอ

และ $$-U_m \leq U_C \leq U_m \qquad (3.13)$$

$S(U_2)$,$B(U_1)$ เป็นชื่อของฟังก์ชันการเป็นสมาชิกในภาคการควบคุมที่ U_2,U_1 ตามลำดับ

μ_s,μ_B เป็นระดับของฟังก์ชันการเป็นสมาชิกของ $S(U_2)$,$B(U_1)$ ตามลำดับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
ระบบจำลอง

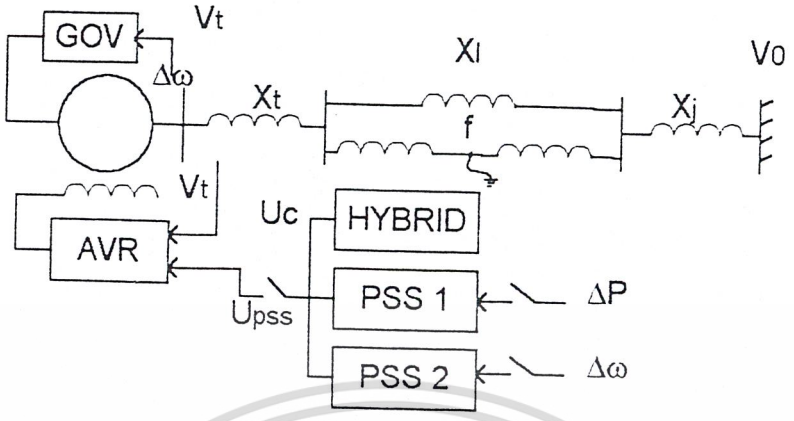
เสถียรภาพของระบบไฟฟ้ากำลังจะพิจารณาจากการตอบสนองการเปลี่ยนแปลงความเร็วเครื่องกำเนิดไฟฟ้าซิงโครนัสเป็นหลัก ถ้าเครื่องกำเนิดไฟฟ้าสามารถกลับมาหมุนที่ความเร็วซิงโครนัสได้อีกครั้ง หลังจากถูกสิ่งรบกวน ไม่ว่าจะเป็นการรบกวนแบบรุนแรงเหนือการรบกวนแบบเล็กน้อย จะถือว่าระบบมีเสถียรภาพ การศึกษาผลตอบสนองของเครื่องกำเนิดไฟฟ้าต่อสิ่งรบกวนจะต้องมีการสร้างแบบจำลอง โดยแบบจำลองที่ใช้จะต้องให้ผลที่ใกล้เคียงกับระบบจริง ด้วยเหตุนี้แบบจำลองที่ใช้จึงต้องรวมอุปกรณ์ควบคุมเครื่องกำเนิดไฟฟ้าต่างๆ คือ GOV (Governor), AVR (Automatic voltage regulator) และ PSS (Power system stabilizer) การติดตั้ง GOV มีวัตถุประสงค์เพื่อควบคุมความเร็วของเครื่องกำเนิดไฟฟ้า โดยควบคุมการเปิดวาล์วของสตีมหรือน้ำที่เทอร์ไบน์ ซึ่งจะมีผลต่อความถี่ไฟของระบบ AVR มีวัตถุประสงค์เพื่อควบคุมแรงดันที่ขั้วเครื่องกำเนิดไฟฟ้าให้คงที่ และยังเพิ่มขีดจำกัดเสถียรภาพสภาวะอยู่ตัวด้วย เพราะ AVR จะช่วยลดผลจากปฏิกิริยาอาร์เมเจอร์ (Armature reaction) เป็นผลให้ค่ารีแอกแตนซ์ของเครื่องกำเนิดไฟฟ้าได้เต็มอัตราพิกัด

เนื้อหาในบทนี้จะเป็นการอธิบายการสร้างสมการแบบจำลองเครื่องกำเนิดไฟฟ้าซิงโครนัสและสมการของบล็อกไดอะแกรมต่างๆ เพื่อที่จะนำมาสร้างแบบจำลองทั้งหมดของระบบ และนำไปใช้ในการสร้างเป็นโปรแกรมเพื่อทดสอบวิธีการควบคุมแบบต่างๆ ต่อไป

4.1 แบบจำลอง

แบบจำลองระบบที่ใช้ในการวิเคราะห์ ประกอบด้วยเครื่องกำเนิดไฟฟ้าซิงโครนัส 1 ตัวต่อกับบัสอนันต์(Infinite bus)ผ่านสายวงจรคู่ ที่เครื่องกำเนิดไฟฟ้าซิงโครนัสจะมีอุปกรณ์ควบคุมความเร็วของเครื่องกำเนิดไฟฟ้า คือ GOV และมี AVR เป็นเครื่องควบคุมแรงดันอัตโนมัติโดยมีอุปกรณ์โดยมีอุปกรณ์ควบคุมแบบไฮบริด อยู่ที่ขั้วของเครื่องกำเนิดกำลังไฟฟ้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.1 ระบบจำลองที่ใช้อุปกรณ์ควบคุมแบบไฮบริด

4.2 สมการเครื่องกำเนิดไฟฟ้าซิงโครนัส

ในหัวข้อนี้จะกล่าวถึงสมการคณิตศาสตร์ต่างๆ ที่สร้างเป็นแบบจำลองของเครื่องกำเนิดไฟฟ้าซิงโครนัส โดยจะพิจารณาการกระทบจากปัจจัยต่างๆ ทั้งภายในและภายนอก เช่น ฟลักซ์ของขดลวดฟิลด์ , ค่าความเสียดทานต่างๆ เป็นต้น แต่เพื่อลดความซับซ้อนของแบบจำลองคณิตศาสตร์เครื่องกำเนิดไฟฟ้าซิงโครนัส จึงจำเป็นที่จะต้องตัดผลจากปัจจัยบางตัวที่มีผลกระทบน้อยมากต่อความละเอียดแม่นยำของแบบจำลอง เช่น ความต้านทานในขดลวดอาร์เมเจอร์ เป็นต้น ดังนั้นจึงเชื่อได้ว่าแบบจำลองที่สร้างขึ้นนี้มีความน่าเชื่อถือสูงและสามารถนำมาอธิบายปรากฏการณ์ต่างๆ ของเครื่องกำเนิดไฟฟ้าซิงโครนัสได้เป็นอย่างดี

สมการต่างๆ ของแบบจำลองคณิตศาสตร์เครื่องกำเนิดไฟฟ้าซิงโครนัที่จะทำการวิเคราะห์ได้แก่ สมการกระแสและแรงดันในแกน d-q, สมการฟลักซ์ขดลวดฟิลด์เครื่องกำเนิดไฟฟ้าซิงโครนัส , สมการสวิง, สมการกำลังไฟฟ้าเอาท์พุทเครื่องกำเนิดไฟฟ้าซิงโครนัส และสมการแรงดันที่ขั้วเครื่องกำเนิดไฟฟ้าซิงโครนัส

4.2.1 สมการกระแสและแรงดันในแกน d-q

สมการแรงดัน Vt ที่ขั้วเครื่องกำเนิดไฟฟ้าซิงโครนัสสามารถจัดให้อยู่ในรูปของสมการแรงดันในแกน d-q โดยการใช้การแปลงของ Park ถ้าไม่คิดผลของความต้านทาน R ในอาร์เมเจอร์ ดิริเวทีฟของฟลักซ์เกี่ยวคล้อง $p\phi_d$, $p\phi_q$ และค่าคงตัวเวลาซับทรานเซียนต์ สมการแรงดันในแกน d-q สามารถลดเหลือรูปอย่างง่ายดังต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$V_d = -\omega\phi_q \tag{4.1}$$

$$V_q = \omega\phi_d \tag{4.2}$$

$$V_f = \frac{d\phi_f}{d_t} + V_{fr} \tag{4.3}$$

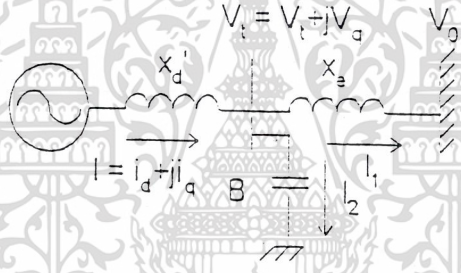
เมื่อ

$$\omega\phi_q = -x_q i_q \tag{4.4}$$

$$\omega\phi_d = V_{fr} - x_d i_d \tag{4.5}$$

$$\phi_f = T'_{d0}\left[V_{fr} - \left(x_q - x'_d\right)i_d\right] \tag{4.6}$$

พิจารณารูปที่ 4.3 เป็นวงจรที่ลดเหลือรูปอย่างง่ายของระบบบัสอนันต์ โดยตัดตัวควบคุมออกจากระบบทั้งหมด



รูปที่ 4.2 ระบบบัสอนันต์ที่ลดเหลือรูปอย่างง่าย

จากทิศทางกระแสที่กำกับในรูปที่ 4.2 สามารถเขียนสมการกระแสได้ดังต่อไปนี้

$$I = i_d + ji_q = I_1 + I_2 \tag{4.7}$$

$$i_d + ji_q = V_t(jB) + (V_t - V_0)jx_e \tag{4.8}$$

โดยที่

$$V_t = V_d + jV_q \tag{4.9}$$

$$V_0 = V_0 \sin\delta + jV_0 \cos\delta \tag{4.10}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แทนค่า V_t และ V_0 จากสมการที่ (4.9) และ (4.10) ลงในสมการที่ (4.8)

$$i_d + ji_q = (V_d + jV_q)(jB) + \frac{V_d + jV_q - V_0 \sin\delta - jV_0 \cos\delta}{jx_e} \tag{4.11}$$

$$i_d + ji_q = jV_dB - V_qB + \frac{-jV_d + V_q + jV_0 \sin\delta - V_0 \cos\delta}{x_e} \tag{4.12}$$

จากสมการที่ (4.12) เทียบสัมประสิทธิ์ของส่วนจริงจะได้

$$i_d = -V_qB + \frac{V_q - V_0 \cos\delta}{x_e} \tag{4.13}$$

$$i_d = \frac{-V_qB + V_q - V_0 \cos\delta}{x_e} \tag{4.14}$$

$$i_d x_e = (1 - x_eB)V_q - V_0 \cos\delta \tag{4.15}$$

$$i_d = \frac{(1 - x_eB)V_q - V_0 \cos\delta}{x_e} \tag{4.16}$$

จากสมการที่ (4.2) และ (4.5)

$$V_q = V_{fr} - x_d i_d \tag{4.17}$$

จากสมการที่ (4.6) ทำการย้ายข้างใหม่เพื่อหา V_{fr} จะได้

$$V_{fr} = \frac{\phi_f}{T'_{d0}} + (x_d - x'_d)i_d \tag{4.18}$$

แทนค่า V_{fr} จากสมการที่ (4.18) ลงในสมการที่ (4.17)

$$V_q = \frac{\phi_f}{T'_{d0}} - x'_d i_d \tag{4.19}$$

แทนค่า V_q จากสมการที่ (4.19) ลงในสมการที่ (4.16)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$i_d = \frac{(1-x_eB)\phi_f}{x_eT_{d0}} - \frac{(1-x_eB)x_di_d}{x_e} - \frac{V_0\cos}{x_e} \quad (4.20)$$

$$i_dx_e = \frac{(1-x_eB)\phi_f}{T'_{d0}}(1-x_eB)x'_di_d - V_0\cos\delta \quad (4.21)$$

$$i_d = \frac{(1-x_eB)\phi_f}{T'_{d0}\left[(1-x_eB)x'_d + x_e\right]} - \frac{V_0\cos\delta}{\left[(1-x_eB)x'_d + x_e\right]} \quad (4.22)$$

เมื่อกำหนดให้

$$X'_d = (1-x_eB)x'_d + x_e \quad (4.23)$$

สมการที่ (4.22) สามารถเขียนใหม่ได้ดังนี้

$$i_d = \frac{(1-x_eB)\phi_f}{X'_dT'_{d0}} - \frac{V_0\cos\delta}{X'_d} \quad (4.24)$$

จากสมการที่ (4.19)

$$V_q = \frac{\phi_f}{T'_{d0}} - x'_di_d$$

แทนค่า I_d จากสมการที่ (4.24) ลงในสมการที่ (4.19)

$$V_q = \frac{\phi_f}{T'_{d0}} - \frac{(1-x_eB)x'_d\phi_f}{X'_dT'_{d0}} + \frac{x'_dV_0\cos\delta}{X'_d} \quad (4.25)$$

$$V_q = \frac{\left[X'_d - (1-x_eB)x'_d\right]\phi_f}{X'_dT'_{d0}} + \frac{x'_dV_0\cos\delta}{X'_d} \quad (4.26)$$

แทนค่า X'_d จากสมการที่ (4.23) ลงในสมการ (4.26)

$$V_q = \frac{\left[(1-x_eB)x'_d + x_e - (1-x_eB)x'd\right]\phi_f}{X'_dT'_{d0}} + \frac{x'_dV_0\cos\delta}{X'_d} \quad (4.27)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$V_q = \frac{x_e \phi_f}{X'_d T'_{d0}} + \frac{x'_d V_0 \cos\delta}{X'_d} \tag{4.28}$$

เทียบสัมประสิทธิ์ส่วนจินตภาพจากสมการที่ (4.12) จะได้

$$i_q = V_d B + \frac{-V_d + V_0 \sin\delta}{x_e} \tag{4.29}$$

$$i_q = i_q x_q B + \frac{(-i_q x_q + V_0 \sin\delta)}{x_e} \tag{4.30}$$

$$i_q x_q = i_q x_q x_e B - i_q x_q + V_0 \sin\delta \tag{4.31}$$

$$\left[(1 - x_e B)x_q + x_e\right] i_q = V_0 \sin\delta \tag{4.32}$$

$$i_q = \frac{V_0 \sin\delta}{(1 - x_e B)x_q + x_e} \tag{4.33}$$

เมื่อกำหนดให้

$$X_q = (1 - x_e B)x_q - x_e \tag{4.34}$$

แทนค่า X_q จากสมการที่ (4.34) ลงในสมการที่ (4.33) จะได้

$$i_q = \frac{V_0 \sin\delta}{X_q} \tag{4.35}$$

จากสมการที่ (4.1) และ (4.4)

$$V_d = x_q i_q \tag{4.36}$$

$$V_d = \frac{x_q V_0 \sin\delta}{X_q} \tag{4.37}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.2สมการฟลักซ์ขดลวดฟิล์ดเครื่องกำเนิดไฟฟ้าซิงโครนัส

จากสมการที่ (4.18)

$$V_{fr} = \frac{\phi f}{T'_{d0}} + (x_d - x'_d) i_d$$

แทนค่า I_d จากสมการที่ (4.24) ลงในสมการที่ (4.18)

$$V_{fr} = \frac{\phi f}{T'_{d0}} + \frac{(x_d - x_d)(1 - x_e B)\phi_f}{X_d T_{d0}} - \frac{(x_d - x_d)V_0 \sin\delta}{X_d} \tag{4.38}$$

แทนค่า X'_d เฉพาะที่เศษของสมการที่ (4.38)

$$V_{fr} = \frac{[(1 - xeB)x'd + x_e + (1 - x_e B)x_d - (1 - x_e B)x'_d]\phi_f}{X'_d T'_{d0}} - \frac{(x_d - x'_d)V_0 \cos\delta}{X'_d} \tag{4.39}$$

$$V_{fr} = \frac{[(1 - x_e B)x_d + x_e)]\phi_f}{X'_d T'_{d0}} - \frac{(x_d - x'_d)V_0 \cos\delta}{X'_d} \tag{4.40}$$

$$V_{fr} = \frac{x_d \phi_f}{X'_d T'_{d0}} - \frac{(x_d - x'_d)V_0 \cos\delta}{X'_d} \tag{4.41}$$

จากสมการที่ (4.3) ทำการย้ายข้างเพื่อจัดรูปแบบใหม่จะได้

$$\frac{d\phi_f}{dt} = V_f - V_{fr} \tag{4.42}$$

แทนค่า V_{fr} จากสมการที่ (4.41) ในสมการที่ (4.42)

$$\frac{d\phi_f}{dt} = V_f - \frac{Xd\phi_f}{X'_d T'_{d0}} - \frac{(x_d - x'_d)V_0 \cos\delta}{X'_d} \tag{4.43}$$

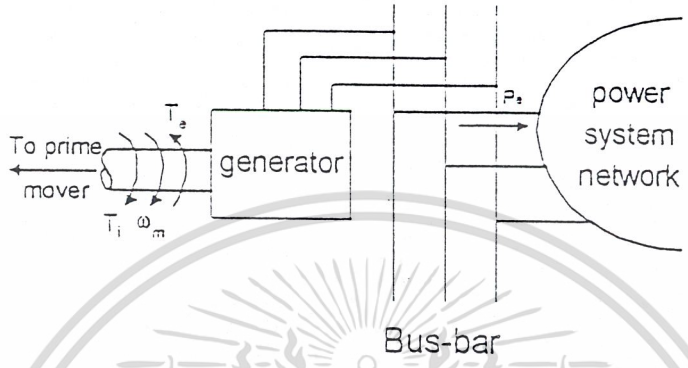
เมื่อ

$$X_d = (1 - X_e B)x_d + x_e \tag{4.44}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.3 สมการสวิง

พิจารณารูปที่ 4.3 เครื่องกำเนิดไฟฟ้าซิงโครนัสได้รับพลังงานกลอินพุท P_i จากเทอร์ไบน์ด้วยทอร์ก T_i และจ่ายพลังงานไฟฟ้าเอาท์พุท P_e โดยทำให้เกิดทอร์กทางแม่เหล็กไฟฟ้า T_e ซึ่งมีทิศทางสวนกับทอร์ก T_i



รูปที่ 4.3 เครื่องกำเนิดไฟฟ้าซิงโครนัสจ่ายกำลังไฟฟ้าสู่โครงข่ายไฟฟ้ากำลัง

เมื่อเกิดสิ่งรบกวน จะเกิดการเร่งหรือหน่วงของโรเตอร์โดยขึ้นกับทอร์กเร่ง T_a ซึ่งกำหนดโดยสมการที่ (4.45)

$$T_a = T_i - T_e \tag{4.45}$$

$$T_a = I\frac{d^2\theta_m}{dt^2} + D\frac{d\theta_m}{dt} \tag{4.46}$$

$$T_a = I\frac{d\omega_m}{dt^2} + D\omega_m \tag{4.47}$$

$D\omega_m$ คือ ทอร์กหน่วง ซึ่งมีสาเหตุจาก ความเสียดทานที่แบริ่ง (Bearing) ของโรเตอร์ , ความเสียดทานจากแรงลม , ความสูญเสียทางแม่เหล็ก และทอร์กอื่นๆ ที่มีทิศทางสวนทางกับทิศทางการหมุนของโรเตอร์

สมการ (4.45) และ (4.47) เมื่อคูณด้วย ω_m ทั้งสองข้างจะได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$P_a = P_i - P_e \tag{4.48}$$

$$P_a = \omega_m I \frac{d\omega_m}{dt} + D\omega_m \tag{4.49}$$

$$P_a = M' \frac{d\omega_m}{dt} + D\omega_m \tag{4.50}$$

เมื่อ M มีหน่วยเป็น pu-sec^2/mech.radian มุม θ_m จะมีการเปลี่ยนแปลงตามเวลาอย่างต่อเนื่อง ทำให้สามารถเขียนสมการการขจัดเชิงมุม δ_m ซึ่งวัดจากแกนอ้างอิงที่หมุนไปด้วยความเร็วซิงโครนัส ω_{0m} (mech.rad/sec) ได้ดังสมการที่ (4.51)

$$\delta_m = \theta_m - \omega_{0m} t \tag{4.51}$$

$$\frac{d\delta_m}{dt} = \frac{d\theta_m}{dt} - \omega_{0m} = \omega_m - \omega_{0m} \tag{4.52}$$

$$\frac{d\delta_m}{dt} = \omega_m - \omega_{0m} = \Delta\omega_m \tag{4.53}$$

สมการที่ (4.52) จัดรูปใหม่จะได้

$$\frac{d\theta_m}{dt} = \frac{d\delta_m}{dt} + \omega_{0m} \tag{4.54}$$

จะเป็นการสะดวกกว่าถ้าการวัดมุมต่างๆ เป็นมุมทางไฟฟ้า ดังนั้นการแปลงมุมทางกลเป็นมุมทางไฟฟ้าสามารถทำได้โดยใช้สมการ

$$\delta = \frac{P}{2}\delta_m \qquad \text{(electrical radian)} \tag{4.55}$$

$$\omega = \frac{P}{2}\omega_m \qquad \text{(electrical radian/s)} \tag{4.56}$$

เมื่อคูณ P/2 ทั้งสองข้างในสมการที่ (4.33) และ (4.54) จะได้

$$\frac{d\theta}{dt} = \omega_0 + \frac{d\delta}{dt} \tag{4.57}$$

$$\frac{d\delta}{dt} = \Delta\omega \tag{4.58}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$\frac{d\theta}{dt} = \omega_0 + \Delta\omega \tag{4.59}$$

สมการที่ (4.50) เมื่อคูณทั้งสองข้างด้วย P/2 แล้วจัดรูปสมการใหม่ จะได้

$$M\frac{d^2\delta}{dt} + \frac{Dd\delta}{dt} = P_i - P_e \tag{4.60}$$

เมื่อ

$$M = \frac{M'}{P/2} \qquad (\text{pu.sec}^2/\text{elect.rad}) \tag{4.61}$$

สมการ (4.60) เขียนใหม่ได้เป็น

$$M\frac{d^2\delta}{dt} D\Delta\omega = P_i - P_e \tag{4.62}$$

จากความสัมพันธ์

$$P_l - P_e = (P_0 + \Delta P_l) - (P_0 + \Delta P_e) \tag{4.63}$$

$$P_l - P_e = \Delta P_l - \Delta P_e \tag{4.64}$$

แทนค่า P_l-P_e จากสมการที่ (4.64) ลงในสมการที่ (4.62) แล้วจัดรูปใหม่ จะได้

$$\frac{d\omega}{dt} = \frac{1}{M}(\Delta P_i - D\Delta\omega - \Delta P_e) \tag{4.65}$$

4.2.4 สมการกำลังไฟฟ้าเอาท์พุทของเครื่องกำเนิดไฟฟ้าซิงโครนัส

กำลังไฟฟ้าเอาท์พุทของเครื่องกำเนิดไฟฟ้าซิงโครนัสสามารถเขียนได้ดังสมการที่ (4.66)

$$P_e = i_d V_d + i_q V_q \tag{4.66}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แทนค่า V_d , i_d , V_q , i_q จากสมการที่ (4.37) , (4.16) , (4.28) , (4.35) ลงในสมการที่ (4.66)

$$P_e = \left(\frac{(1 - x_e B)\phi_f}{X'_d T'_{d0}} - \frac{V_0 \cos\delta}{X'_d} \right) \frac{x_q V_0 \sin\delta}{X_q} + \frac{V_0 \sin\delta}{X_q} \left(\frac{x_e \phi_f}{X'_d T'_{d0}} + \frac{x'_d V_0 \cos\delta}{X_q X'_d} \right) \tag{4.67}$$

$$P_e = \frac{(1 - x_e B) x_q \phi_f V_0 \sin\delta}{X_q X'_d T'_{d0}} - \frac{x_q V_0^{\,2} \sin\delta\cos\delta}{X_q X'_d} + \frac{x_e \phi_f V_0 \sin\delta}{X_q X'_d T'_{d0}} + \frac{x'_d V_0^{\,2} \sin\delta\cos\delta}{X_q X'_d} \tag{4.68}$$

ทำการจัดพจน์ในสมการ (4.68) ใหม่จะได้

$$P_e = \frac{V_0 \sin\delta\phi_f \left[(1 - x_e B) x_q + x_e \right]}{X_q X'_d T'_{d0}} - \frac{V_0^{\,2} \sin\delta\cos\delta (x'_d - x_q)}{X_q X'_d} \tag{4.69}$$

$$P_e = \frac{V_0 \sin\delta\phi_f}{X'_d T'_{d0}} - \frac{2V_0^{\,2} \sin\delta\cos\delta (x'_d - x_q)}{2X_q X'_d} \tag{4.70}$$

สมการที่ (4.70) จัดให้เหลือในรูปอย่างง่ายจะได้

$$P_e = \frac{V_0 \sin\delta\phi_f}{X'_d T'_{d0}} - \frac{(x'_d - x_q) V_0^{\,2} \sin 2\delta}{2X_q X'_d} \tag{4.71}$$

4.2.5 สมการแรงดันที่ขั้วเครื่องกำเนิดไฟฟ้าซิงโครนัส

ความสัมพันธ์ของแรงดันไฟฟ้าที่ขั้วเครื่องกำเนิดไฟฟ้าซิงโครนัสกับแรงดันในแกน d-q ของเครื่องกำเนิดไฟฟ้าสามารถเขียนได้ดังนี้

$$V_t = \sqrt{V_d^{\,2} + V_q^{\,2}} \tag{4.72}$$

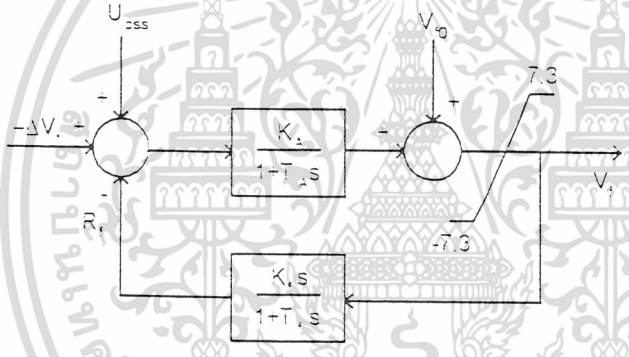
แทนค่า V_d , V_q จากสมการที่ (4.37) , (4.28) ลงในสมการที่ (4.72)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$V_t = \sqrt{\left(\frac{x_q V_0 \sin\delta}{X_q}\right)^2 + \left(\frac{x_e \phi_f}{X_d T_{d0}} + \frac{x_d V_0 \sin\delta}{X_d}\right)^2} \qquad (4.73)$$

4.3 แบบจำลอง AVR

บล็อกไดอะแกรม AVR ในรูปที่ 4.1 สามารถแทนด้วยแบบจำลอง AVR ในรูปที่ 4.4 หน้าที่ของ AVR คือ ควบคุมขนาดของแรงดันที่ขั้วเครื่องกำเนิดไฟฟ้าซิงโครนัส จากแบบจำลองในรูปที่ 4.4 จะสังเกตได้ว่า U_{PSS} เป็นสัญญาณควบคุมเสริมที่ออกจาก PSS โดยจะรวมกับสัญญาณการเปลี่ยนแปลงแรงดันที่ขั้วเครื่องกำเนิดไฟฟ้า ΔV_t วัตถุประสงค์ของสัญญาณควบคุมเสริม U_{PSS} คือ สร้างทอร์กหน่วงให้กับระบบโดยอาศัยการขยายสัญญาณผ่าน AVR เนื่องจากว่ามีค่าเกนของเอกไซเตอร์สูง



รูปที่ 4.4 แบบจำลอง AVR

การวิเคราะห์สมการจากรูปที่ 4.4 สามารถทำได้โดยเขียนสมการในแต่ละบล็อกดังนี้

$$V_f = \frac{K_A}{1+T_A s}(U_{PSS} - \Delta V_t - R_f) + V_{f0} \qquad (4.74)$$

$$V_f + T_A s Vf = K_A(U_{PSS} - \Delta V_t - R_f) + V_{f0} \qquad (4.75)$$

$$T_A s V_f = K_A(U_{PSS} - \Delta V_t - R_f) - (V_f - V_{f0}) \qquad (4.76)$$

$$sV_f = \frac{K_A}{T_A}(U_{PSS} - \Delta V_t - R_f) - \frac{(V_f - V_{f0})}{T_A} \qquad (4.77)$$

ทำการแปลงอินเวอร์สลาปาสสมการที่ (4.77) จะได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$\frac{dV_f}{dt} = \frac{K_A}{T_A}(U_{PSS} - \Delta V_t - R_f) - \frac{(V_f - V_{f0})}{T_A} \tag{4.78}$$

เขียนสมการจากบล็อกไดอะแกรมส่วนป้อนกลับรูปที่ 4.5 ได้ดังนี้

$$R_f = \frac{K_f s}{1 + T_f s} V_f \tag{4.79}$$

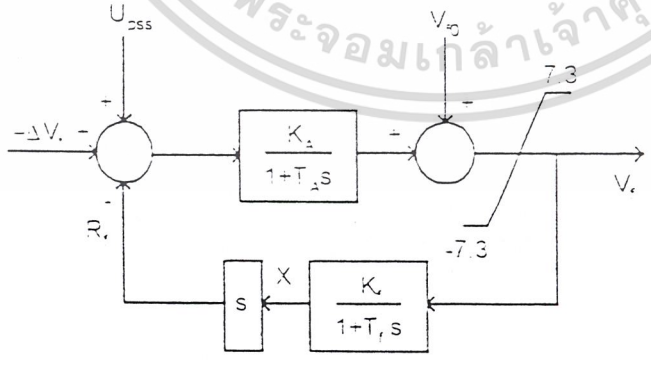
$$R_f + T_f s R_f = K_f s V_f \tag{4.80}$$

$$sR_f \frac{K_f}{T_f} sV_f - \frac{R_f}{T_f} \tag{4.81}$$

ทำการแปลงอินเวอร์สลาปาสสมการที่ (4.81) จะได้

$$\frac{R_f}{dt} = \frac{K_f}{dt}\frac{dV_f}{dt} - \frac{R_f}{T_f} \tag{4.82}$$

หรือทำการจัดรูปส่วนป้อนกลับใหม่ ดังรูปที่ 4.5 แล้วทำการหาสมการบล็อกไดอะแกรมป้อนกลับใหม่อีกรอบ



รูปที่ 4.5 แบบจำลอง AVR ส่วนบล็อกป้อนกลับที่มีการปรับปรุง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 4.5 ส่วนป้อนกลับสามารถเขียนเป็นสมการได้ดังนี้

$$X = \frac{K_f}{1+T_f s} V_f \tag{4.83}$$

$$X + T_f sX = K_f V_f \tag{4.84}$$

$$sX = \frac{K_f V_f - X}{T_f} \tag{4.85}$$

ทำการแปลงอินเวอร์สลาปาสสมการที่ (4.85) จะได้

$$\frac{dX}{dt} = \frac{K_f Vf - X}{T_f} \tag{4.86}$$

จากส่วนป้อนกลับในรูปที่ 4.5 เขียนสมการต่อได้เป็น

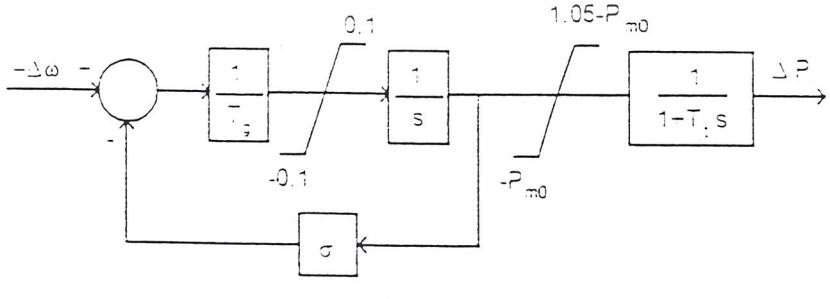
$$R_f = sX \tag{4.87}$$

$$R_f = \frac{K_f V_f - X}{T_f} \tag{4.88}$$

$$R_f = \frac{K_f ZZ - X}{T_f} \tag{4.89}$$

4.4 แบบจำลอง GOV

บล็อกไดอะแกรม GOV ในรูปที่ 4.1 สามารถแทนด้วยแบบจำลอง GOV ในรูปที่ 4.6 วัตถุประสงค์ของการใช้ GOV คือ ใช้ปรับความเร็วของเครื่องกำเนิดไฟฟ้าซิงโครนัสให้มีความเร็วคงที่เท่ากับความเร็วซิงโครนัส



รูปที่ 4.6 แบบจำลอง GOV

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 4.6 สามารถเขียนสมการจากบล็อกไดอะแกรมต่างๆได้ดังนี้

$$g = \frac{1}{T_g s}(-\Delta\omega - \sigma g) \tag{4.90}$$

$$sg = \frac{1}{T_g}(-\Delta\omega - \sigma g) \tag{4.91}$$

ทำการแปลงอินเวอร์สลาปาสสมการที่ (4.91) จะได้

$$\frac{dg}{dt} = \frac{1}{T_g}(-\Delta\omega - \sigma g) \tag{4.92}$$

ส่วนที่เหลือของรูปที่ 4.7 เขียนเป็นสมการได้ดังนี้

$$\Delta P_i = \frac{g}{1 + T_t s} \tag{4.93}$$

$$\Delta P_i + T_t s \Delta P_i = g \tag{4.94}$$

$$s\Delta P_i = \frac{1}{T_t}(g - \Delta P_i) \tag{4.95}$$

ทำการแปลงอินเวอร์สลาปาสสมการที่ (4.95) จะได้

$$\frac{d\Delta P_i}{dt} = \frac{1}{T_t}(g - \Delta P_i) \tag{4.96}$$

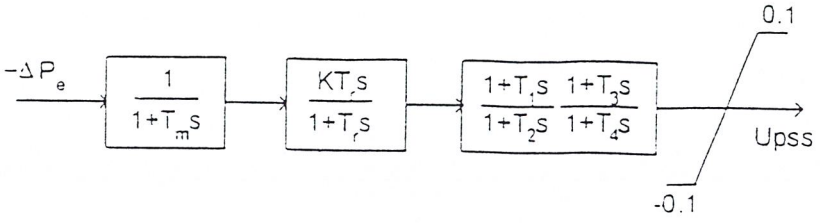
4.5 แบบจำลอง PSS

บล็อกไดอะแกรม PSS ในรูปที่ 4.1 และ รูปที่ 4.2 ซึ่งมีอยู่ 2 บล็อก คือ PSS1 และ PSS2 สามารถแทนด้วยแบบจำลอง PSS 2 ชนิด คือ ΔP-PSS และ $\Delta\omega$-PSS

4.5.1 แบบจำลอง ΔP-PSS

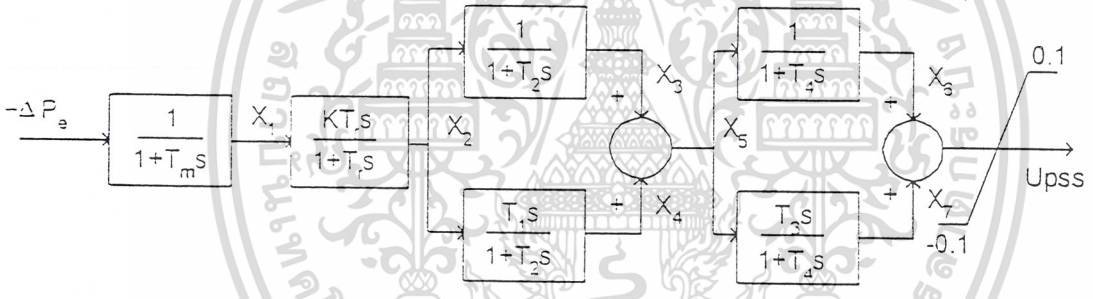
PSS1 สามารถจำลองด้วยแบบจำลอง ΔP-PSS ในรูปที่ 4.7

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.7 แบบจำลอง ΔP-PSS

เพื่อให้เกิดความง่ายและสะดวกในการนำสมการที่เขียนขึ้นไปใช้งาน แบบจำลองในรูปที่ 4.7 จึงปรับปรุงใหม่ได้ดังรูปที่ 4.8



รูปที่ 4.8 แบบจำลอง ΔP-PSS ที่ทำการปรับปรุงใหม่

จากรูปที่ 4.8 สามารถเขียนเป็นสมการได้ดังนี้

$$X_1 = \frac{-\Delta P_e}{1+T_m s} \tag{4.97}$$

$$X_1 + T_m s X_1 = -\Delta P_e \tag{4.98}$$

$$sX_1 = \frac{-\Delta P_e - X_1}{T_m} \tag{4.99}$$

ทำการแปลงอินเวอร์สลาปาสสมการที่ (4.99)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$\frac{dX_1}{dt} = \frac{-\Delta P_e - X_1}{T_m} \tag{4.100}$$

เขียนสมการตามรูปที่ 4.8 ในบล็อกถัดไปจะได้

$$X_2 = \frac{KT_r s}{1+T_r s} X_1 \tag{4.101}$$

$$X_2 + T_r s X_2 = KT_r s X_1 \tag{4.102}$$

$$sX_2 = \frac{KT_r s X_1 - X_2}{T_r} \tag{4.103}$$

แทนค่า sX1 จากสมการที่ (4.99) ในสมการที่ (4.103) จะได้

$$sX_2 = K\frac{(-\Delta P_e - X_1)}{T_m} - \frac{X_2}{T_r} \tag{4.104}$$

ทำการแปลงอินเวอร์สลาปาสสมการที่ (4.104) จะได้

$$\frac{dX_2}{dt} = K\frac{(-\Delta P_e - X_1)}{T_m} - \frac{X_2}{T_r} \tag{4.105}$$

สมการในบล็อกถัดไปในรูปที่ 4.8 เขียนได้ดังนี้

$$X_3 = \frac{X_2}{1+T_2 s} \tag{4.106}$$

$$X_3 + T_2 s X_3 = X_2 \tag{4.107}$$

$$sX_3 = \frac{X_2 - X_3}{T_2} \tag{4.108}$$

ทำการแปลงอินเวอร์สลาปาสสมการที่ (4.108) จะได้

$$\frac{dX_3}{dt} = \frac{X_2 - X_3}{T_2} \tag{4.109}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สมการในบล็อกถัดไปในรูปที่ 4.8 เขียนได้ดังนี้

$$X_4 = \frac{T_1 s}{1+T_2 s} X_3 \tag{4.110}$$

$$X_4 + T_2 s X_4 = T_1 s X_3 \tag{4.111}$$

$$sX_4 = \frac{T_1 s X_3}{T_2} - \frac{X_4}{T_2} \tag{4.112}$$

แทนค่า sX3 จากสมการที่ (4.108) ในสมการที่ (4.112)

$$sX_4 = \frac{T_1 (X_2 - X_3)}{T_2 T_2} - \frac{X_4}{T_2} \tag{4.113}$$

ทำการแปลงอินเวอร์สลาปาสสมการที่(4.113)

$$\frac{dX_4}{dt} = \frac{T_1 (X_2 - X_3)}{T_2 T_2} - \frac{X_4}{T_2} \tag{4.114}$$

สมการในบล็อกถัดไปในรูปที่ 4.8 เขียนสมการได้ดังนี้

$$X_5 = X_3 + X_4 \tag{4.115}$$

$$X_6 = \frac{X_5}{1+T_4 s} \tag{4.116}$$

$$X_6 + T_4 s X_6 = X_5 \tag{4.117}$$

$$sX_6 = \frac{X_5 - X_6}{T_4} \tag{4.118}$$

ทำการแปลงอินเวอร์สลาปาสสมการที่ (4.118) จะได้

$$\frac{dX_6}{dt} = \frac{X_5 - X_6}{T_4} \tag{4.119}$$

สมการในบล็อกถัดไปในรูปที่ 4.8 เขียนได้ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$X_7 = \frac{T_3 s}{1+T_4 s} X_5 \tag{4.120}$$

$$X_7 + T_4 s X_7 = T_3 s X_5 \tag{4.121}$$

$$sX_7 = \frac{T_3 s X_5}{T_4} - \frac{X_7}{T_4} \tag{4.122}$$

สมการที่(4.115) คูณด้วย s ทั้งสองข้าง จะได้ sX5 จากนั้นแทนลงในสมการที่(4.122)

$$sX_7 = \frac{T_3}{T_4}\left[\frac{X_2 - X_3}{T_2} + \left[\frac{T_1(X_2 - X_3)}{T_2 T_2} - \frac{X_4}{T_2}\right]\right] - \frac{X_7}{T_4} \tag{4.123}$$

ทำการแปลงอินเวอร์สลาปาสสมการที่(5.123) จะได้

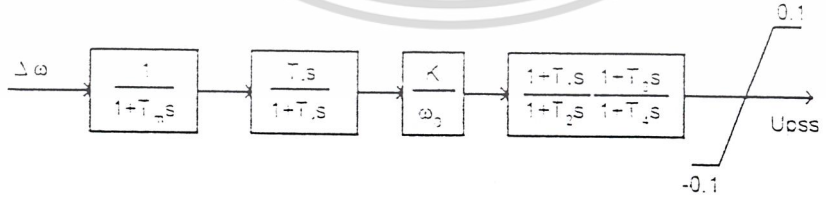
$$\frac{dX_7}{dt} = \frac{T_3}{T_4}\left[\frac{X_2 - X_3}{T_2} + \left[\frac{T_1(X_2 - X_3)}{T_2 T_2} - \frac{X_4}{T_2}\right]\right] - \frac{X_7}{T_4} \tag{4.124}$$

จากรูปที่ 4.6 ผลรวมสุดท้ายของสัญญาณควบคุมเสริม คือ

$$U_{PSS} = X_6 + X_1 \tag{4.125}$$

4.5.2 แบบจำลอง $\Delta\omega$-PSS

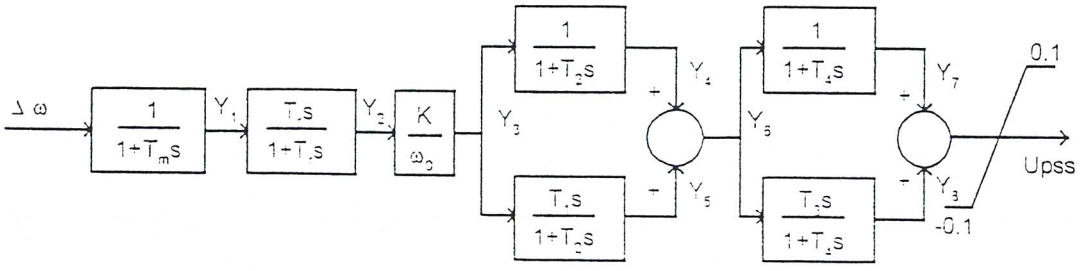
บล็อกไดอะแกรม PSS2 ในรูปที่ 4.1 แทนด้วยแบบจำลอง $\Delta\omega$-PSS ดังแสดงในรูปที่ 4.9



รูปที่ 4.9 แบบจำลอง $\Delta\omega$-PSS

เพื่อความสะดวกและง่ายต่อการนำไปใช้แบบจำลองในรูปที่ 4.9 จึงปรับปรุงใหม่ดังรูปที่ 4.10

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.10 แบบจำลอง $\Delta\omega$-PSS ที่ทำการปรับปรุงใหม่

จากรูปที่ 4.10 เขียนเป็นสมการได้ดังนี้

$$Y_1 = \frac{\Delta\omega}{1+T_m s} \tag{4.126}$$

$$Y_1 + T_m s Y_1 = \Delta\omega \tag{4.127}$$

$$sY_1 = \frac{\Delta\omega - Y_1}{T_m} \tag{4.128}$$

ทำการแปลงอินเวอร์สลาปาสสมการที่(4.128) จะได้

$$\frac{dY_1}{dt} = \frac{\Delta\omega - Y_1}{T_m} \tag{4.129}$$

สมการในบล็อกถัดไปของรูปที่ 4.10 เขียนได้ดังนี้

$$Y_2 = \frac{T_r s}{1+T_r s} Y_1 \tag{4.130}$$

$$Y_2 + T_r s Y_2 = T_r s Y_1 \tag{4.131}$$

$$sY_2 = \frac{T_r s Y_1 - Y_2}{T_r} \tag{4.132}$$

ทำการแปลงอินเวอร์สลาปาสสมการที่ (4.132) จะได้

$$\frac{dY_2}{dt} = \frac{T_r s Y_1 - Y2}{T_r} \tag{4.133}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สมการในบล็อกถัดไปของรูปที่ 4.10 เขียนได้ดังนี้

$$Y_3 = \frac{KY_2}{\omega_0} \tag{4.134}$$

$$Y_4 = \frac{Y_3}{1+T_2 s} \tag{4.135}$$

$$Y_4 + T_2 s Y_4 = Y_3 \tag{4.136}$$

$$sY_4 = \frac{Y_3 - Y_4}{T_4} \tag{4.137}$$

ทำการแปลงอินเวอร์สลาปาสสมการที่ (4.137) จะได้

$$\frac{dY_4}{dt} = \frac{Y_3 - Y_4}{T_4} \tag{4.138}$$

สมการในบล็อกถัดไปของรูปที่ 4.10 เขียนได้ดังนี้

$$Y_5 = \frac{T_1 s}{1+T_2 s} Y_3 \tag{4.139}$$

$$Y_5 + T_2 s Y_5 = T_1 s Y_3 \tag{4.140}$$

$$sY_5 = \frac{T_1 s Y_3}{T_2} - \frac{Y_5}{T_2} \tag{4.141}$$

คูณสมการที่ (4.134) ด้วย s ทั้งสองข้าง จากนั้นแทนค่า sY3 ลงในสมการ (4.141)

$$sY_2 = \frac{T_1 s Y_2}{T_2 \omega_0} - \frac{Y_5}{T_2} \tag{4.142}$$

แทนค่า sY2 จากสมการที่ (4.132) ลงในสมการที่ (4.142)

$$sY_5 = \frac{T_1 K}{T_2 \omega_0}\left[\frac{T_1 s Y_1 - Y_2}{T_r}\right] - \frac{Y_5}{T_2} \tag{4.143}$$

แทนค่า sY1 จากสมการที่ (4.128) ลงในสมการที่ (4.143)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$sY_5 = \frac{T_1 K}{T_2 \omega_0}\left[\frac{T_1}{T_r}\frac{T_1(\Delta\omega - Y_1)}{T_m} - \frac{Y_2}{T_r}\right] - \frac{Y_5}{T_2} \tag{4.144}$$

ทำการแปลงอินเวอร์สลาปาสสมการที่(4.144) จะได้

$$\frac{dY_5}{dt} = \frac{T_1 K}{T_2 \omega_0}\left[\frac{T_1}{T_r}\frac{(\Delta\omega - Y_1)}{T_m} - \frac{Y_2}{T_r}\right] - \frac{Y_5}{T_2} \tag{4.145}$$

บล็อกไดอะแกรมถัดไปในรูปที่ 4.10 เขียนได้ดังนี้

$$Y_6 = Y_4 + Y_5 \tag{4.146}$$

$$Y_7 = \frac{Y_6}{1 + T_4 s} \tag{4.147}$$

$$Y_7 + T_4 s Y_1 = Y_6 \tag{4.148}$$

$$sY_7 = \frac{Y_6 - Y_7}{T_4} \tag{4.149}$$

ทำการแปลงอินเวอร์สลาปาสสมการที่ (4.149) จะได้

$$\frac{dY_7}{dt} = \frac{Y_6 - Y_7}{T_4} \tag{4.150}$$

สมการในบล็อกถัดไปของรูปที่ 4.10 เขียนได้ดังนี้

$$Y_8 = \frac{T_3 s}{1 + T_4 s} Y_6 \tag{4.151}$$

$$Y_8 + T_4 s Y_8 = T_3 s Y_6 \tag{4.152}$$

$$sY_8 = \frac{T_3 s Y_6}{T_4} - \frac{Y_8}{T_4} \tag{4.153}$$

สมการที่ (4.146) คูณด้วย s ทั้งสองข้าง จากนั้นแทนค่า sY6 ลงในสมการที่ (4.153)

$$sY_8 = \frac{T_3}{T_4}\left[\frac{Y_3 - Y_4}{T_4} + \frac{T_1 K}{T_2 \omega_0}\left[\frac{T_1}{T_r}\frac{(\Delta\omega - Y_1)}{T_m} - \frac{Y_2}{T_r}\right] - \frac{Y_5}{T_2}\right] - \frac{Y_8}{T_4} \tag{4.154}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

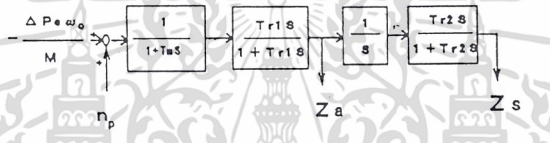
ทำการแปลงอินเวอร์สลาปาสสมการที่ (4.154) จะได้

$$\frac{dY_8}{dt} = \frac{T_3}{T_4}\left[\frac{Y_3 - Y_4}{T_4} + \frac{T_1 K}{T_2 \omega 0}\left[\frac{T_1}{T_r}\frac{(\Delta\omega - Y_1)}{T_m} - \frac{Y_2}{T_r}\right] - \frac{Y_5}{T_2}\right] - \frac{Y_8}{T_4} \qquad (4.155)$$

จากรูปที่ 4.10 ผลรวมสุดท้ายของสัญญาณควบคุมเสริม U_{PSS} คือ

$$U_{PSS} = Y_7 + Y_8 \qquad (4.156)$$

4.6 แบบจำลอง SPD



รูปที่ 4.11 แบบจำลอง SPD

บล็อกไดอะแกรมของ SPD เขียนเป็นสมการได้ดังนี้

$$X_{14} + T_m s X_{14} = X_{13} \qquad (4.157)$$

$$sX_{14} = \frac{X_{13} - X_{14}}{T_m} \qquad (4.158)$$

ทำการแปลงอินเวอร์สลาปาสสมการที่ (4.158)

$$\frac{dX_{14}}{dt} = \frac{X_{13} - X_{14}}{T_m} \qquad (4.159)$$

เขียนสมการในบล็อกถัดไปได้คือ

$$X_{15} = \frac{T_{r1} s}{1 + T_{r1} s} X_{14} \qquad (4.160)$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$$X_{15} + T_{r1} sX_{15} = T_{r1} sX_{14} \tag{4.161}$$

$$sX_{15} = sX_{14} - \frac{X_{15}}{T_{r1}} \tag{4.162}$$

แทนค่า sX14 จากสมการ (4.158) ในสมการ (4.162)

$$sX_{15} = \frac{X_{13} - X_{14}}{T_m} - \frac{X_{15}}{T_{r1}} \tag{4.163}$$

แปลงอินเวอร์สลาปาสสมการ (4.163)

$$\frac{dX_{15}}{dt} = \frac{X_{13} - X_{14}}{T_m} - \frac{X_{15}}{T_{r1}} \tag{4.164}$$

เขียนสมการในส่วนถัดไป จะได้

$$X_{16} = \frac{T_{r2}}{1 + T_{r2} s} X_{15} \tag{4.165}$$

$$X_{16} + T_{r2} sX_{16} = T_{r2} X_{15} \tag{4.166}$$

$$sX_{16} = \frac{T_{r2} X_{15} - X_{16}}{T_{r2}} \tag{4.167}$$

$$sX_{16} = X_{15} - \frac{X_{16}}{T_{r2}} \tag{4.168}$$

$$Z_s = X_{15} - \frac{X_{16}}{T_{r2}} \tag{4.169}$$

แปลงอินเวอร์สลาปาสสมการ (4.168) จะได้

$$\frac{dX_{16}}{dt} = X_{15} - \frac{X_{16}}{T_{r2}} \tag{4.170}$$

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
การออกแบบโปรแกรมเจเนติกอัลกอริทึม

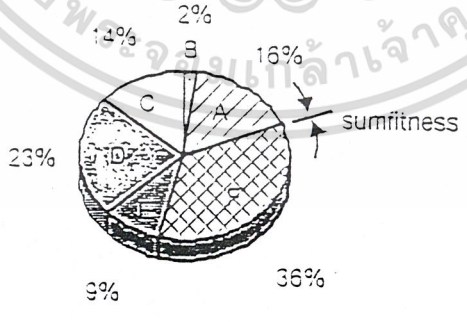
เนื่องจากการใช้งานเจเนติกอัลกอริทึม บ่อยครั้งต้องอาศัยเวลาในการประมวลผลเป็นเวลานาน ดังนั้นถ้าปัญหานั้นต้องอาศัยโครโมโซมที่มีขนาดยาว การรันอาจผิดพลาดได้เนื่องจากหน่วยความจำไม่พอ การออกแบบโปรแกรมเจเนติก อัลกอริทึมนี้ได้มีการคำนึงถึงปัญหานี้ด้วยและเลือกใช้วิธีการจัดการและเก็บข้อมูลแบบบิทคงที่การจัดการและเก็บข้อมูลแบบอาเรย์ โปรแกรมที่เขียนขึ้นประกอบด้วย การครอสโอเวอร์แบบยูนิฟอร์ม การมิวเทชัน การทำสเกลลิ่งแบบเชิงเส้น การทำสเกลลิ่งแบบเอกโปเนนเชียล การคัดเลือกแบบรูเลท

5.1 การคัดเลือก

การคัดเลือกที่เขียนขึ้นในโปรแกรมคือ การคัดเลือกแบบวงล้อรูเลท(Roulette wheel Selection) รายละเอียดของวิธีการคัดเลือกสามารถอธิบายได้ดังต่อไปนี้

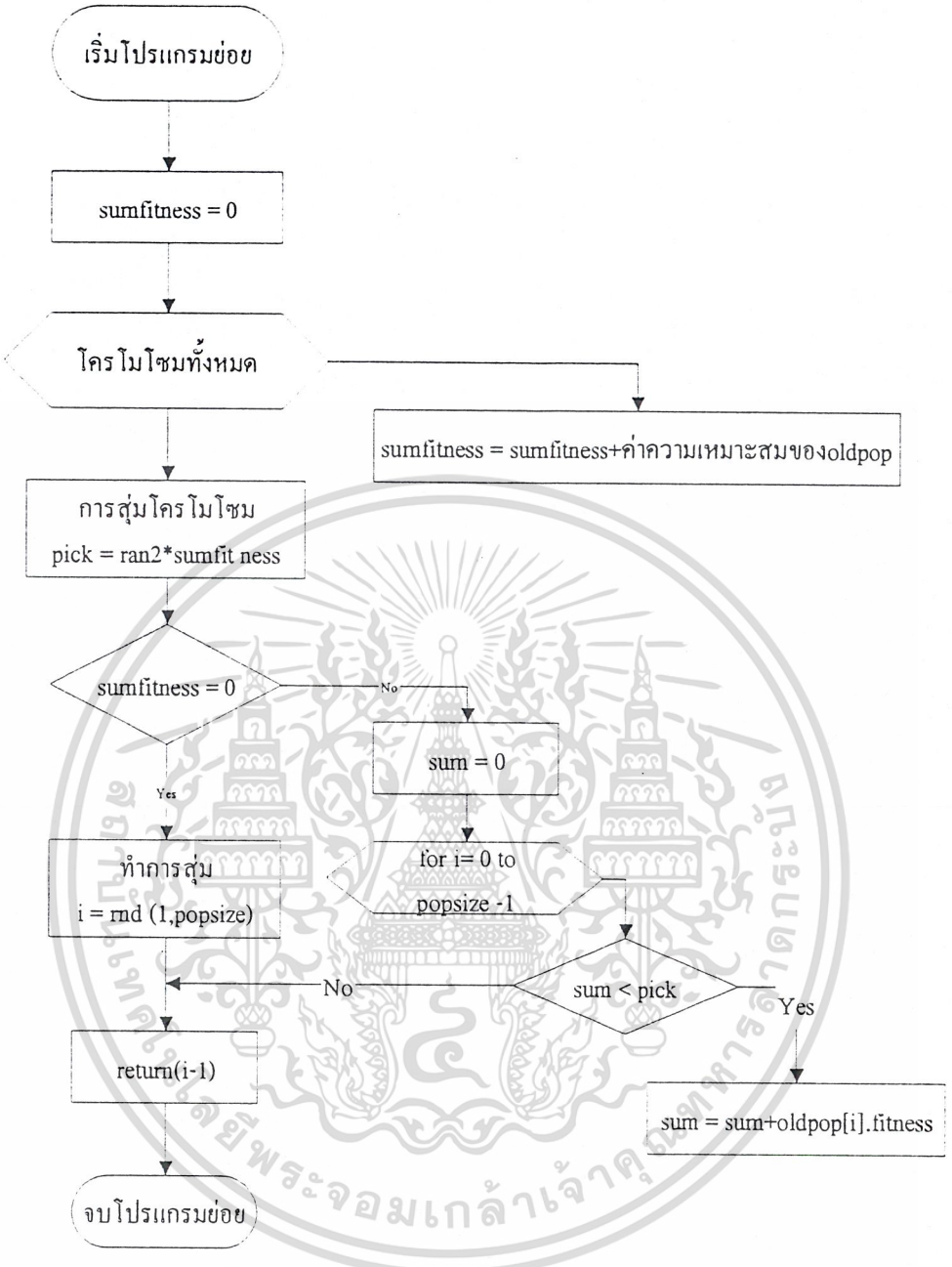
การคัดเลือกแบบวงล้อรูเลท(Roulette wheel Selection)

วิธีการนี้จะเป็นการคัดเลือกตามส่วน โดยพิจารณาจากค่าฟิทเนสของประชากรแต่ละตัว ดังเช่น ในรูปที่ 5.1 ซึ่งมีประชากรทั้งหมด 6 ตัว อันดับแรกจะแบ่งส่วนของพื้นที่รูเลทตามสัดส่วนของค่าฟิทเนส โดยให้เส้นรอบวงของวงล้อรูเลท คือผลรวมของค่าฟิทเนสของประชากรแต่ละตัว จากนั้น จะสุ่มค่าตั้งแต่ 0-sumfitness ถ้าตัวเลขที่สุ่มนั้นไปหยุดส่วนของพื้นที่ใด ประชากรตัวนั้นก็จะถูกเลือก ดังนั้นประชากรที่มีค่าฟิทเนสสูงก็จะมีโอกาสถูกเลือกมากกว่าประชากรที่มีค่าฟิทเนสต่ำวิธีการดังกล่าวเขียนเป็นโฟลชาทได้ดังรูปที่ 5.2



รูปที่ 5.1 พื้นที่ในวงล้อรูเลทที่แบ่งตามสัดส่วนของค่าฟิทเนส

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.2 โฟลชาทการคัดเลือกแบบวงล้อรูเลท

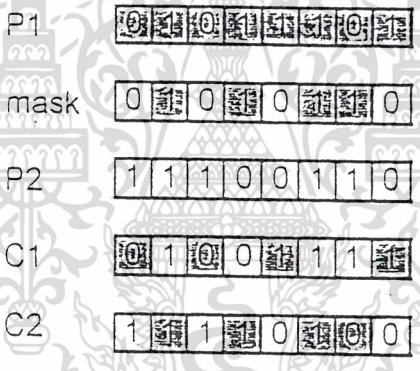
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.2 การครอสโอเวอร์

โปรแกรมย่อยสำหรับการครอสโอเวอร์ใช้วิธีการครอสโอเวอร์แบบยูนิฟอร์ม โดยมีรายละเอียดดังต่อไปนี้

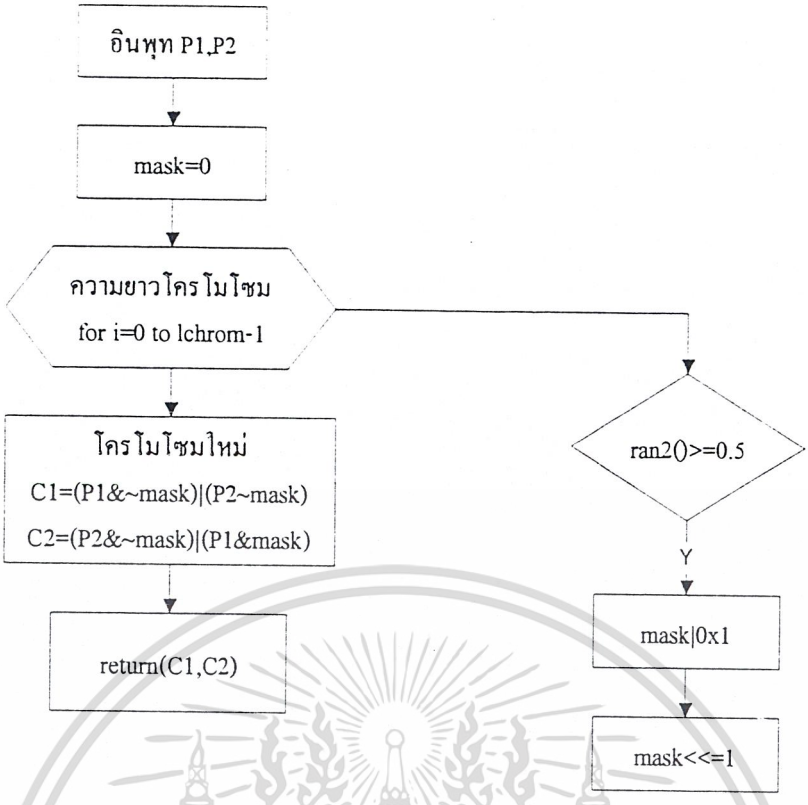
การครอสโอเวอร์แบบยูนิฟอร์ม

การทำงานของการครอสโอเวอร์วิธีนี้แสดงในรูปที่ 5.4 โดยจะเริ่มจากการสร้าง mask ด้วยวิธีการสุ่มลักษณะคล้ายรูปที่ 5.3 สำหรับ C1 บิทที่ 0 ของ mask จะสำเนาโครโมโซม P1 บางส่วน บิทที่เป็น 1 ของ mask จะสำเนาโครโมโซม P2 บางส่วน สำหรับ C2 บิทที่เป็น 0 ของ mask จะสำเนาโครโมโซม P2 บางส่วน บิทที่เป็น 1 จะสำเนา P1 บางส่วน โฟลชาทของการการครอสโอเวอร์แบบยูนิฟอร์มแสดงในรูปที่ 5.4



รูปที่ 5.3 การครอสโอเวอร์แบบยูนิฟอร์ม

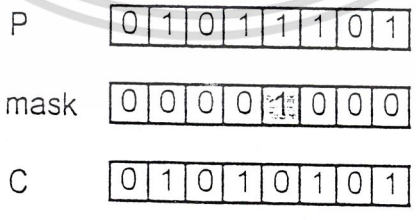
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.4 โฟลชาทการครอสโอเวอร์แบบยูนิฟอร์ม

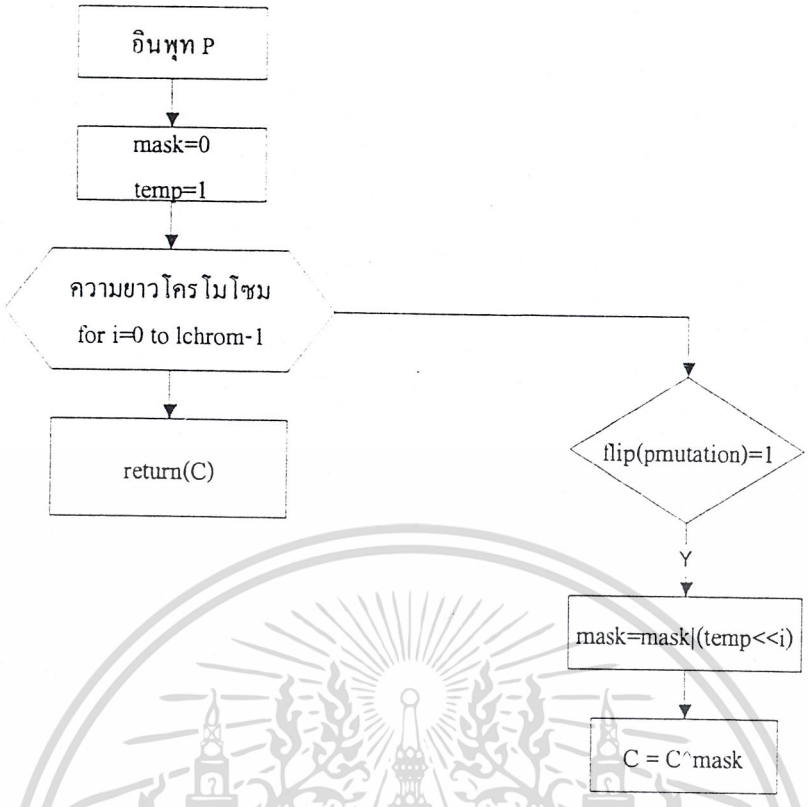
5.3การมิวเทชัน

ขั้นตอนการมิวเทชันแสดงในรูปที่ 5.5 โดยจะเริ่มจากการสร้าง mask ด้วยวิธีการสุ่ม ลักษณะคล้ายรูปที่ 5.5 ตำแหน่งบิทที่เป็น 1 ของ mask จะทำการกลับค่าของ P1 จาก 1 เป็น 0 จาก 0 เป็น 1 โฟลชาทของการมิวเทชันแสดงในรูปที่ 5.6



รูปที่ 5.5 การมิวเทชัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.6 โฟลชาทการมิวเทชัน

5.4 การทำสเกลลิ่ง

วัตถุประสงค์ของการทำสเกลลิ่งก็เพื่อป้องกันการเกิดการลู่เข้าก่อนกำหนด (Premature)อันเนื่องจากการครอบครองพื้นที่วงล้อรูเลทของประชากรบางตัวที่มีค่าฟิทเนสสูงมากๆ ทำให้ประชากรที่มีค่าฟิทเนสต่ำมากๆไม่มีโอกาสถูกเลือก ซึ่งบางครั้งประชากรที่มีค่าฟิทเนสต่ำๆ อาจมีบางส่วนเป็นข้อมูลดีก็ได้ การทำสเกลลิ่งจะเป็นการปรับค่าฟิทเนสใหม่ทั้งหมด ก่อนเข้าสู่การคัดเลือกด้วยวิธีวงล้อรูเลท ซึ่งจะช่วยเพิ่มโอกาสสำหรับประชากรที่มีค่าฟิทเนสต่ำ ได้มีโอกาสถูกเลือกมากขึ้น

สำหรับโปรแกรมที่เขียนจะมีวิธีการทำสเกลลิ่งสองวิธี คือการทำสเกลลิ่งแบบเชิงเส้นและการทำสเกลลิ่งแบบเอกซ์โปเนนเชียล ซึ่งมีรายละเอียดดังต่อไปนี้

1. <u>การทำสเกลลิ่งแบบเชิงเส้น</u>

วิธีการทำสเกลลิ่งวิธีนี้ จะแปลงค่าฟิทเนสใหม่โดยใช้สมการเส้นตรง $y = ax+b$ โดย x คือค่าฟิทเนสเก่า และ y คือค่าฟิทเนสใหม่ ข้อมูลที่ใช้ในสมการเส้นตรงนี้ คือ fmin , favg และ fmax ในเจนเนอเรชันรุ่นก่อน โดยค่าฟิทเนสที่ได้จะต้องมีค่าเป็นบวกเสมอ แต่ก็มีความเป็นไปได้ที่จะได้สมการเส้นตรงที่แปลงค่าฟิทเนสใหม่แล้วได้ค่าลบ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

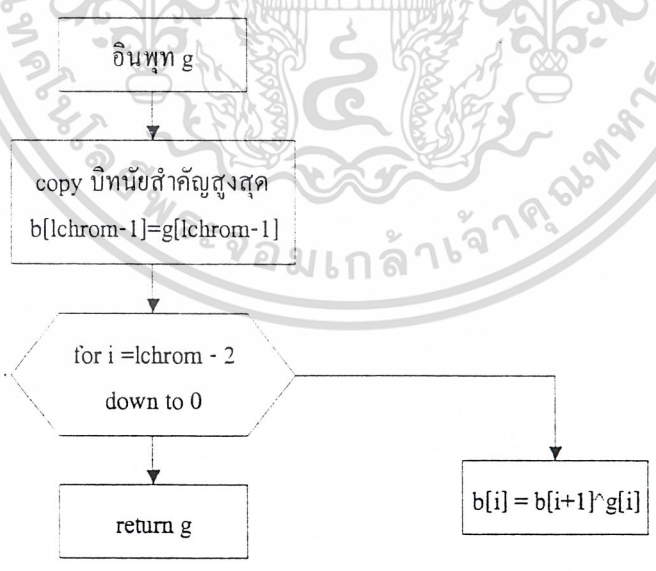
2. การทำสเกลลิ่งแบบเอกซ์โปเนนเชียล

วิธีนี้สามารถแก้ไขปัญหาการลู่เข้าจุดที่เหมาะสมเฉพาะที่ (Local hill) ซึ่งไม่ใช่จุดที่ให้ค่าฟิทเนสสูงสุดได้ดี ทำให้มีโอกาสหลุดออกจากจุดความเหมาะสมเฉพาะที่มากขึ้น เนื่องประชากรที่มีค่าฟิทเนสต่ำๆ มีโอกาสถูกเลือกมากขึ้น

5.5 การเข้ารหัสแบบเกรย์

รหัสแบบเกรย์ คือ รหัสไบนารีที่มีการจัดลำดับใหม่ ให้มีตำแหน่งของบิทข้างเคียงต่างกันเพียง 1 บิท ตัวอย่างเช่น รหัสไบนารีของ [0...7] คือ { 000, 001, 010, 011, 100, 101, 110, 111} ในขณะที่รหัสแบบเกรย์ คือ { 000, 001, 011, 010, 110, 111, 101, 100} จากตัวอย่างนี้จะสังเกตได้ว่า การที่จะเปลี่ยนค่าจาก 0 เป็น 7 สำหรับรหัสแบบไบนารีจะต้องเปลี่ยนบิทถึง 3 ครั้ง แต่สำหรับรหัสแบบเกรย์เปลี่ยนบิทเพียง 1 ครั้งเท่านั้น ทำให้รหัสแบบเกรย์มีส่วนช่วยปรับปรุงประสิทธิภาพในการมิวเทชั่นค่าที่อยู่ข้างเคียง

ในโปรแกรมที่เขียนขึ้น โครโมโซมแต่ละตัวจะเป็นรหัสแบบเกรย์ ดังนั้นในขั้นตอนของการถอดรหัสจะต้องมีการถอดเป็นค่าเลขฐานสิบ โดยจะมีขั้นตอนจาก แปลงรหัสแบบเกรย์ไปเป็นรหัสแบบไบนารีก่อน แล้วจึงแปลงรหัสแบบไบนารีเป็นเลขฐานสิบ การแปลงรหัสแบบเกรย์ไปเป็นรหัสแบบไบนารีมีขั้นตอนตามโฟลชาทรูปที่ 5.7



รูปที่ 5.7 โฟลชาทการแปลงรหัสแบบเกรย์ไปเป็นแบบไบนารี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.6 การเก็บรักษาโครโมโซมที่ดีที่สุด (Elitism)

อิลิทิซึม เป็นการเก็บรักษาโครโมโซมที่ดีที่สุดในแต่ละเจนเนอเรชันเพื่อเป็นหลักประกันว่าโครโมโซมที่ดีที่สุดในเจอเนอเรชันถัดไปจะดีกว่าเจนเนอเรชันก่อนหน้านี้เสมอ

ในโปรแกรมที่เขียนขึ้น จะมีการเก็บรักษาโครโมโซมที่ดีที่สุดไว้ 1 ตัว โดยจะเก็บไว้ที่ตำแหน่ง newpop[0].chrom เสมอ ตัวแปร elitist จะเป็นตัวกำหนดว่า ในโปรแกรมจะมีการเก็บโครโมโซมที่ดีที่สุดไว้หรือไม่ ซึ่งสามารถกำหนดไว้ในตอนเริ่มต้นของโปรแกรม

5.7 การปรับค่าความน่าจะเป็นในการครอสโอเวอร์และมิวเทชัน

โดยปกติแล้วความน่าจะเป็นในการครอสโอเวอร์จะอยู่ในช่วง 0.6-0.9 และความน่าจะเป็นในการมิวเทชันจะไม่เกิน 0.1 ในตอนเริ่มต้นของเจอเนอเรชันแรกๆ นั้น ต้องการความหลากหลายของประชากร ดังนั้นความน่าจะเป็นในการครอสโอเวอร์ควรจะมีค่าสูงและความน่าจะเป็นในการมิวเทชันควรจะมีค่าต่ำ แต่เมื่อเจนเนอเรชันสูงๆ ผลการครอสโอเวอร์จะน้อยลงและอาจนำไปสู่การลู่เข้าก่อนกำหนดได้ ดังนั้นความหลากหลายในช่วงนี้สามารถปรับได้โดยเพิ่มความน่าจะเป็นในการมิวเทชันให้สูงขึ้นและลดความน่าจะเป็นในการครอสโอเวอร์ให้น้อยลง

ในโปรแกรมที่เขียนขึ้น จะปรับค่าความน่าจะเป็นในการครอสโอเวอร์ลดลงอย่างเป็นเชิงเส้นจนถึง 0.6 และความน่าจะเป็นในการมิวเทชันเพิ่มขึ้นอย่างเป็นเชิงเส้นจนถึง 0.1 โดยเป็นไปตามสมการ (5.10) และ (5.11) Pc(0) คือ ความน่าจะเป็นในการครอสโอเวอร์เริ่มต้น Pm(0) คือความน่าจะเป็นในการมิวเทชันเริ่มต้น ซึ่งสามารถกำหนดได้จากค่าเริ่มในโปรแกรม

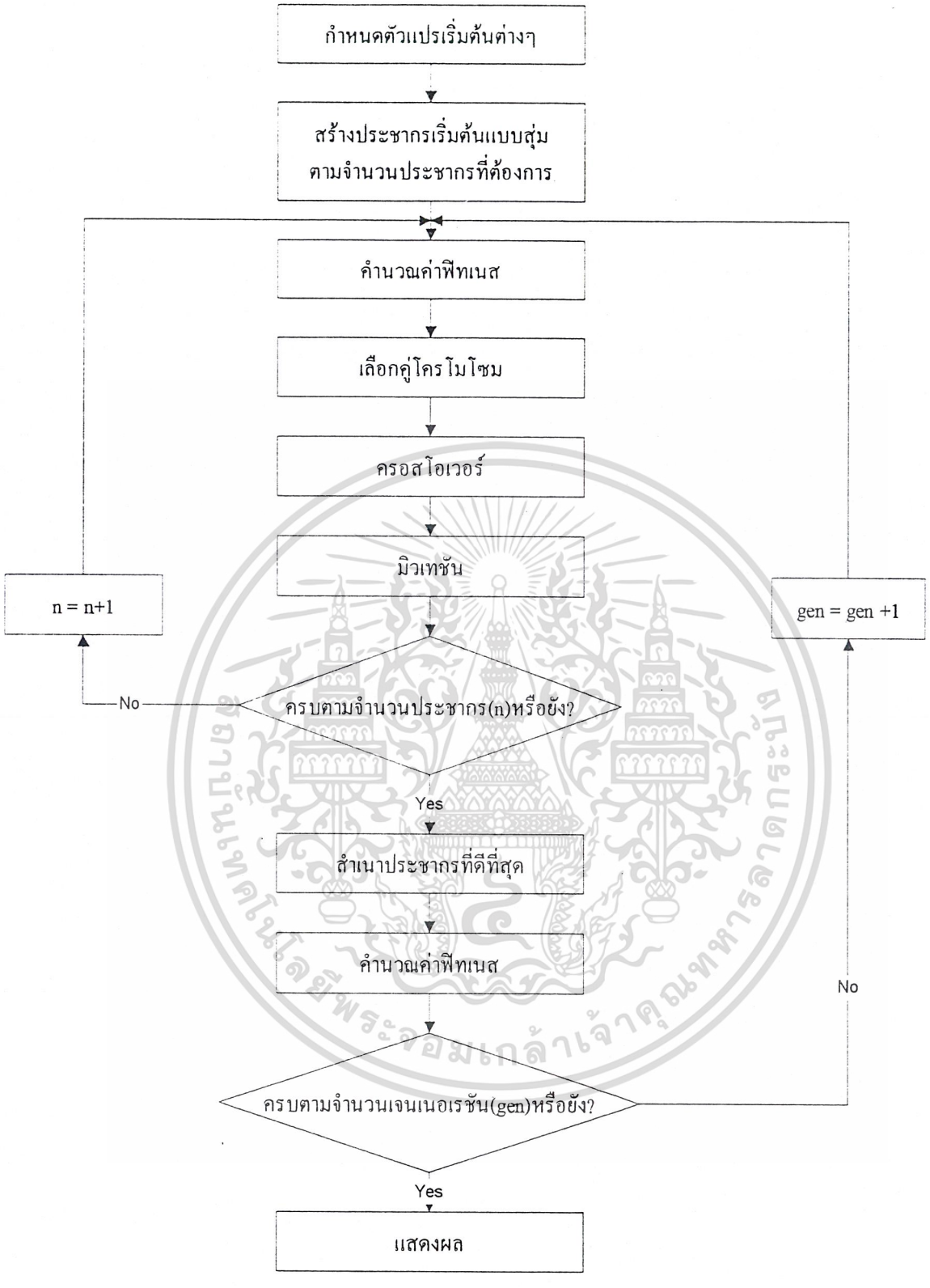
$$Pc(t) = Pc(t-1) - [Pc(0) - 0.6] / \text{Max gen} \qquad (5.1)$$

$$Pm(t) = Pm(t-1) + [0.1 - Pm(0)] / \text{Max gen} \qquad (5.2)$$

5.8 โฟลชาทโปรแกรมเจเนติกอัลกอริทึม

โปรแกรมหลักของโปรแกรมเจเนติกอัลกอริทึมที่เขียนขึ้นแสดงในโฟลชาทรูปที่ 5.8

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 5.8 โฟลชาทโปรแกรมเจเนติกอัลกอริทึม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

5.9 ส่วนเพิ่มเติมในการออกแบบ

โปรแกรมที่เขียนขึ้นนี้ เป็นโปรแกรมออกแบบตัวควบคุมแบบไฮบริดโดยวิธีเจเนติอัลกอริทึม ซึ่งมีโปรแกรมหลักเหมือนกับโฟลชาทรูปที่ 5.11 แต่มีบางส่วนของโปรแกรมที่ต้องมีการดัดแปลง คือ การคำนวณหาความยาวโครโมโซม และการถอดรหัสจากโครโมโซม การคำนวณค่าฟิทเนส

-การคำนวณความยาวโครโมโซม[--]

การคำนวณความยาวโครโมโซมจะใช้สมการ (5.12)

$$L \geq \frac{\ln\left[\frac{\alpha_{max} - \alpha_{min}}{stepsize}\right]}{\ln 2} \qquad (5.3)$$

เมื่อ L คือ ความยาวสตริง α_{max} คือ ค่าขอบเขตบน

α_{mun} คือ ค่าขอบเขตล่าง step size คือ ความละเอียดต่อช่วง เช่น 0.01

โปรแกรมที่เขียนขึ้นมี step size ของ α= 0.001 , α= 1.0 , α= 0.01 ความยาวสตริงของแต่ละจีนจึงเป็น 10,7,7 ตามลำดับ ดังนั้นโครโมโซมจึงมีความยาวเท่ากับ 24 บิท เป็นต้น

-การถอดรหัสจากโครโมโซม

การถอดรหัสโครโมโซมจะแปลงจากรหัสเกรย์เป็นรหัสไบนารีก่อนแล้วใช้สมการ (5.4) และ(5.5) แปลงรหัสไบนารีเป็นเลขฐานสิบ โดยแบ่งเป็น 2 กรณี

<u>กรณี 1</u> การเพิ่มแต่ละขั้นเป็นทศนิยม เช่น 0.001

$$\alpha_i = (\text{double})[(\text{int})(X_i(10^P)/2^L)]/10^P \qquad (5.4)$$

<u>กรณี 2</u> การเพิ่มแต่ละขั้นเป็นจำนวนเต็ม เช่น 1.0

$$\alpha_i = (\text{double})[(\text{int})(X_i(\alpha_{i\,max}-\alpha_{i\,max})/2^L)] \qquad (5.5)$$

สมการที่ (5.4) และ(5.5)จะต้องอยู่ภายใต้เงื่อนไข $2^L \Rightarrow 10^P$

โดยที่ i = 1,2,3

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

X_i คือ ค่าฐานสิบที่ถอดรหัสจากเลขไบนารี P คือ ตำแหน่งความละเอียดทศนิยม

-การคำนวณค่าฟิทเนส

การคำนวณค่าฟิทเนสมีขั้นตอนดังนี้ คือ ถอดรหัสโครโมโซมเลขฐานสิบ และคำนวณค่าดัชนีทำงานจากฟังก์ชัน performance index หลังจากนั้นค่าฟิทเนสที่หาได้คือ

$$J_{\omega 1} = \left(\sum_{k=1}^{N} (|\Delta\omega(k)| tk + U^{\circ}(k)) \right) \quad (5.6)$$

$$J_{\omega 2} = \left(\sum_{k=1}^{N} (|\Delta\omega(k)| tk + U^{\circ}(k)) \right) \quad (5.7)$$

$$J_{\omega} = J_{\omega 1} + \beta J_{\omega 2} \quad (5.8)$$

$$F = \rho / J_{\omega} \quad (5.9)$$

โดยที่ i = 1 คือการรบกวนแบบรุนแรง i = 2 คือการรบกวนแบบเล็กน้อย

ρ คือ แฟกเตอร์ปรับสเกลเท่ากับ 2500 , β คือค่าคงที่ช่วง 0.1

รายละเอียดในส่วนของขั้นตอนการตรวจรหัสจากโครโมโซม การคำนวณค่าฟิทเนส และสำหรับโปรแกรมหลัก ดูได้จากโปรแกรมชื่อ ANGEL.C ในภาคผนวก ก.

5.10 วิธีการใช้โปรแกรม

ในการเรียกใช้โปรแกรมเจเนติกอัลกอริทึม จะอยู่ในโปรแกรมชื่อ ANGEL.C ในภาคผนวก ก. ซึ่งผู้ใช้งานจะต้องมีโปรแกรมปฏิบัติการคือ Turbo C++ เวอร์ชันใดก็ได้ (ผู้จัดทำใช้เวอร์ชัน3.0) เมื่อเข้าสู่โปรแกรม Turbo C++ แล้วปฏิบัติตามขั้นตอนดังต่อไปนี้

1. เรียกใช้ไฟล์ปฏิบัติการ Tc.exe
2. เปิดโปรแกรม Angel.c
3. คลิกที่ปุ่ม Run หรือกด Ctrl+F9

หน้าจอจะแสดงผลดังตารางที่ 5.1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางที่ 5.1 แสดงหน้าจอในการใช้งานโปรแกรม ANGEL.C

```
****************************************************************
        Power System Stabilizers Using Genetic Algorithm
****************************************************************
Parameters Setting :
Enter population size---------->30
Enter chromosome length----->50
Enter maxgen  ---------->50
Enter Pcrossover ---------->0.9
Enter Pmutation  ---------->0.01
```

เมื่อใส่ค่าตามที่แสดงในตารางแล้วโปรแกรมก็จะทำงานตามปกติ

4. ถ้าต้องการดูโปรแกรมของระบบจำลองที่นำเสนอสามารถดูได้จากโปรแกรม Model.c ซึ่งอยู่ในภาคผนวก ข.
5. โปรแกรมแสดงผลทางกราฟสามารถดูได้จากโปรแกรม Gplot.h ซึ่งอยู่ในภาคผนวก ค.
6. โปรแกรมที่ใช้ในการสุ่มสามารถดูได้จากโปรแกรม Random.c ซึ่งอยู่ในภาคผนวก ง.
7. การตรวจสอบการแสดงผลของค่าฟิทเนส สามารถดูได้จากแฟ้มข้อมูลชื่อ Angel.dat

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 6

การออกแบบการทดลองและผลการทดลอง

6.1 ระบบจำลองที่ออกแบบในโครงงาน

ระบบจำลองที่ออกแบบการทดลองในโครงงาน ประกอบด้วย

1. กรณีที่ควบคุมด้วยตัวควบคุมไฮบริด(hybrid)
2. กรณีที่ควบคุมด้วย ΔP-PSS เป็นกรณีที่ศึกษาเพิ่มเติม
3. กรณีที่ควบคุมด้วย $\Delta\omega$-PSS เป็นกรณีที่ศึกษาเพิ่มเติม
4. กรณีที่ควบคุมด้วย SVC+$\Delta\omega$-PSS* เป็นกรณีที่ต้องการศึกษาเพิ่มเติม เพื่อดูผลการตอบสนองต่อการรบกวนแบบรุนแรง

6.2 ค่าคงที่ที่ใช้กับระบบจำลอง

ระบบจำลองที่แสดงในรูปที่ 4.1 มีค่าคงที่ดังตารางที่ 6.1

ตารางที่ 6.1 ค่าคงที่ที่ใช้ในระบบจำลอง

ชนิดอุปกรณ์	ค่าตัวแปร
เครื่องกำเนิดไฟฟ้า(GEN)	T'_{d0}= 5.6 (s), M = 0.0191(pu),D = 0.00531(pu), X'_d = 0.25(pu), X_d= 1.62(pu), X_q= 1.6(pu), V_t= 1.0 (pu), V_b= 1.0 (pu), P_0= 1.0 (pu), ω_0= 1.0 (pu),
สายส่ง	สำหรับแบบจำลองที่ใช้ไฮบริด: X_t= 0.14 (pu), X_j= 0.05 (pu), X_l= 0.28 (pu), สำหรับแบบจำลองที่ใช้ SVC+$\Delta\omega$-PSS : X_t= 0.14 (pu), X_j= 0.05 (pu), X_{l1}= 0.21 (pu), X_{l2}= 0.30 (pu) Stability Limit(A_t) = 1.0 เท่า
AVR	K_A=200 , T_A= 0.05 (s), K_f= 0.07 , T_f= 1.25 (s)
GOV	σ = 0.05, T_g= 0.02 (s), T_t= 2.0 (s)

*อ้างอิงข้อมูลเพื่อใช้ในการเปรียบเทียบจากเอกสารอ้างอิง [10]

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

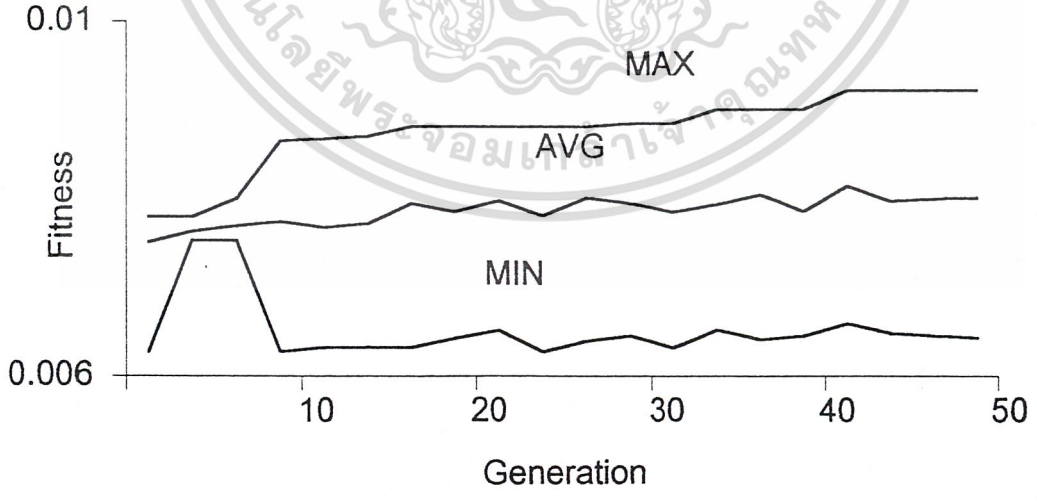
6.3 ผลการตอบสนองของตัวควบคุมแบบไฮบริด โดยใช้โปรแกรมเจเนติกอัลกอริทึมออกแบบ

ตารางที่ 6.2 ค่าของตัวแปรของตัวควบคุมแบบไฮบริด ที่ออกแบบโดยโปรแกรม GA

ตัวควบคุมของแบบไฮบริด	ค่าของตัวแปร	ดัชนีทำงาน(Jw)
ΔP-PSS	Kp=0.92, T1=0.737,T2=0.208, T3=0.863, T4=0.969	270871.40
SPD	T_{r1}= 7.49, T_{r2}=3.0	
FL	α_1=0.162, α_2=82.0, α_3=2.56, α_4=0.601, α_5=0.61,Z_{armin}= 0.001, Z_{smin}= 0.334	
FJ	γ_1= 2.074, γ_2=2.203	

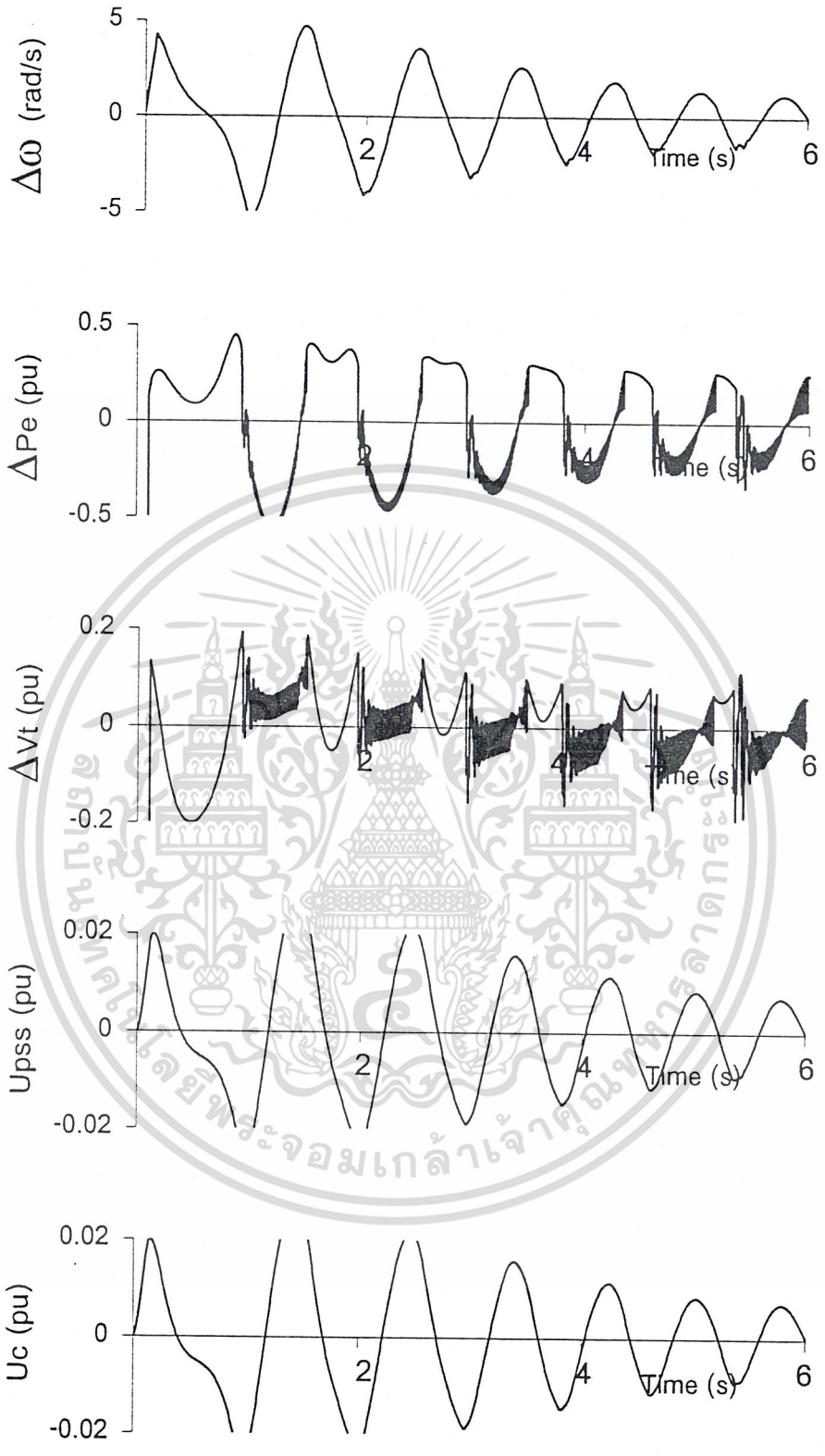
จากค่าดัชนีทำงาน(performance index : Jw) สามารถหาค่าฟิทเนสได้ และนำมาแสดงด้วยรูปที่ 6.1 เป็นค่าฟิทเนสของแบบไฮบริด ซึ่งเป็นผลของค่าฟิทเนสที่ได้จากการรัน 50 เจนเนอเรชัน

สัญญาณควบคุมที่ได้จากตัวควบคุมแบบไฮบริด จะได้ผลตอบสนองต่อการรบกวนแบบรุนแรงแสดงได้ดังรูปที่ 6.2 และผลตอบสนองต่อการรบกวนแบบเล็กน้อยแสดงได้ดังรูปที่ 6.3 ซึ่งจะเป็นสัญญาณป้อน Upss ให้แก่ AVR



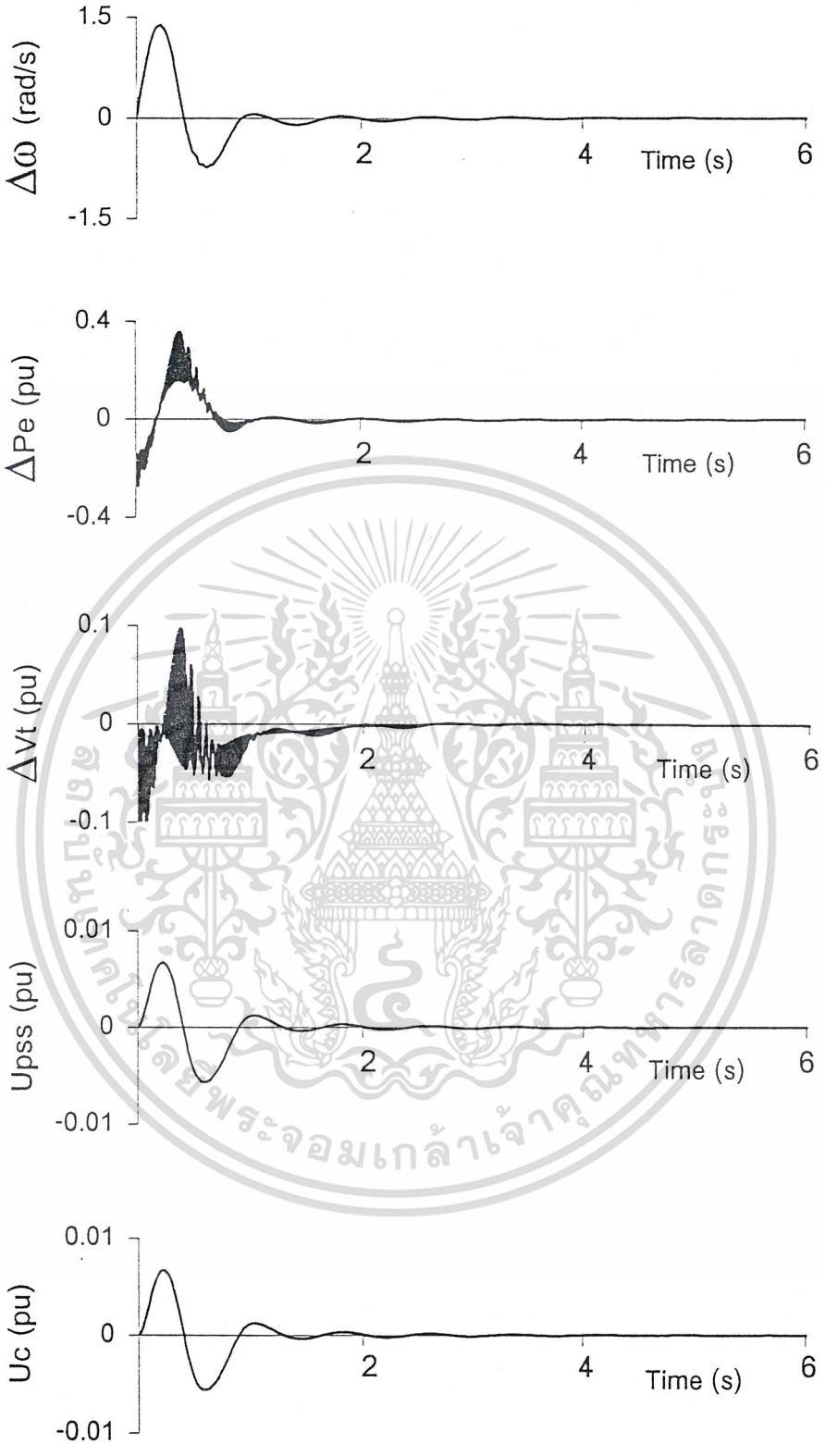
รูปที่ 6.1 ค่าฟิทเนสของตัวควบคุมแบบไฮบริดที่ได้จากการรัน 50 เจนเนอเรชัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6.2 ผลการตอบสนองของตัวควบคุมไฮบริด โดยใช้โปรแกรม ANGEL.C
เมื่อมีการรบกวนแบบรุนแรง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6.3 ผลการตอบสนองของตัวควบคุมไฮบริด โดยใช้โปรแกรม ANGEL.C เมื่อมีการรบกวนแบบเล็กน้อย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

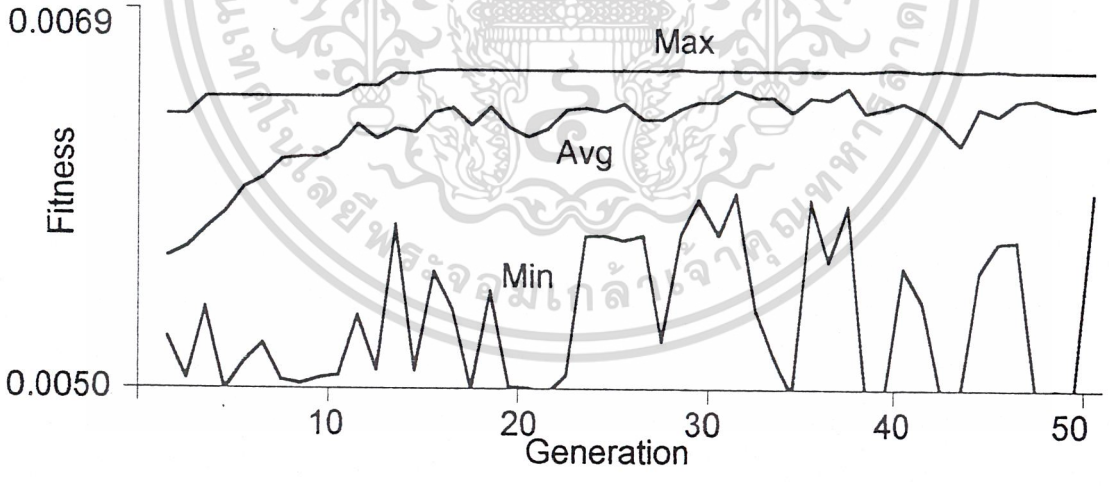
6.4 ผลการตอบสนองของตัวควบคุมแบบ ΔP-PSS โดยใช้โปรแกรมเจเนติกอัลกอริทึมออกแบบ

ตารางที่ 6.3 ค่าตัวแปรของตัวควบคุมแบบ ΔP-PSS ที่ได้จากโปรแกรม GA

ชนิดตัวควบคุม	ค่าของตัวแปร	ดัชนีทำงาน(Jw)
ΔP-PSS	Kp=50, T1=0.2, T2=0.15, T3=0.3, T4=0.13	357840.9

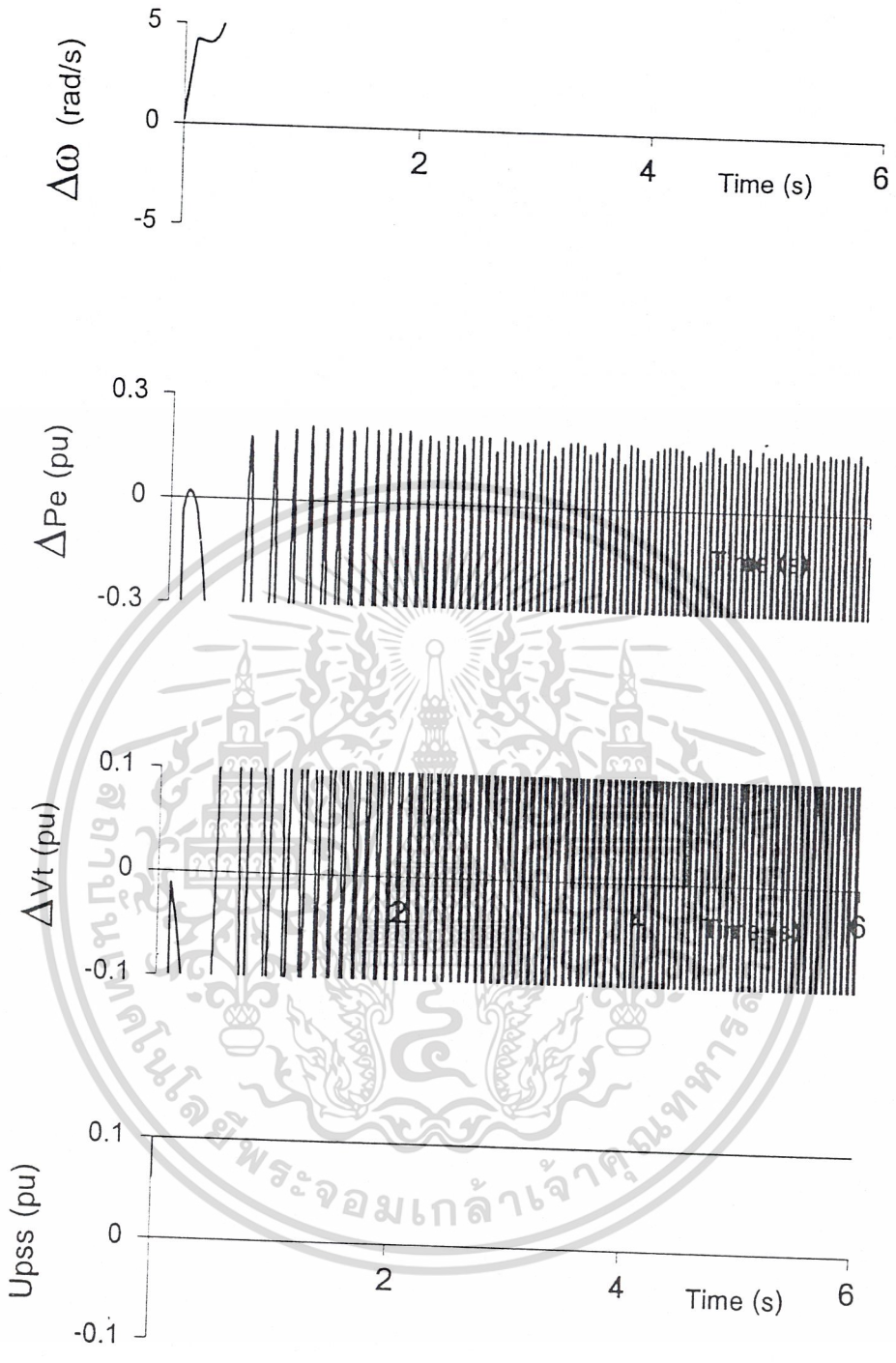
จากค่าดัชนีทำงาน(performance index : Jw) สามารถหาค่าฟิทเนสได้ และนำมาแสดงด้วยรูปที่ 6.4 เป็นค่าฟิทเนสของ ΔP-PSS ซึ่งเป็นผลของค่าฟิทเนสที่ได้จากการรัน 50 เจนเนอเรชัน

สัญญาณควบคุมที่ได้จาก ΔP-PSS จะได้ผลตอบสนองต่อการรบกวนแบบรุนแรงแสดงได้ดังรูปที่ 6.5 และผลตอบสนองต่อการรบกวนแบบเล็กน้อยแสดงได้ดังรูปที่ 6.6 ซึ่งจะเป็นสัญญาณป้อน Upss ให้แก่ AVR



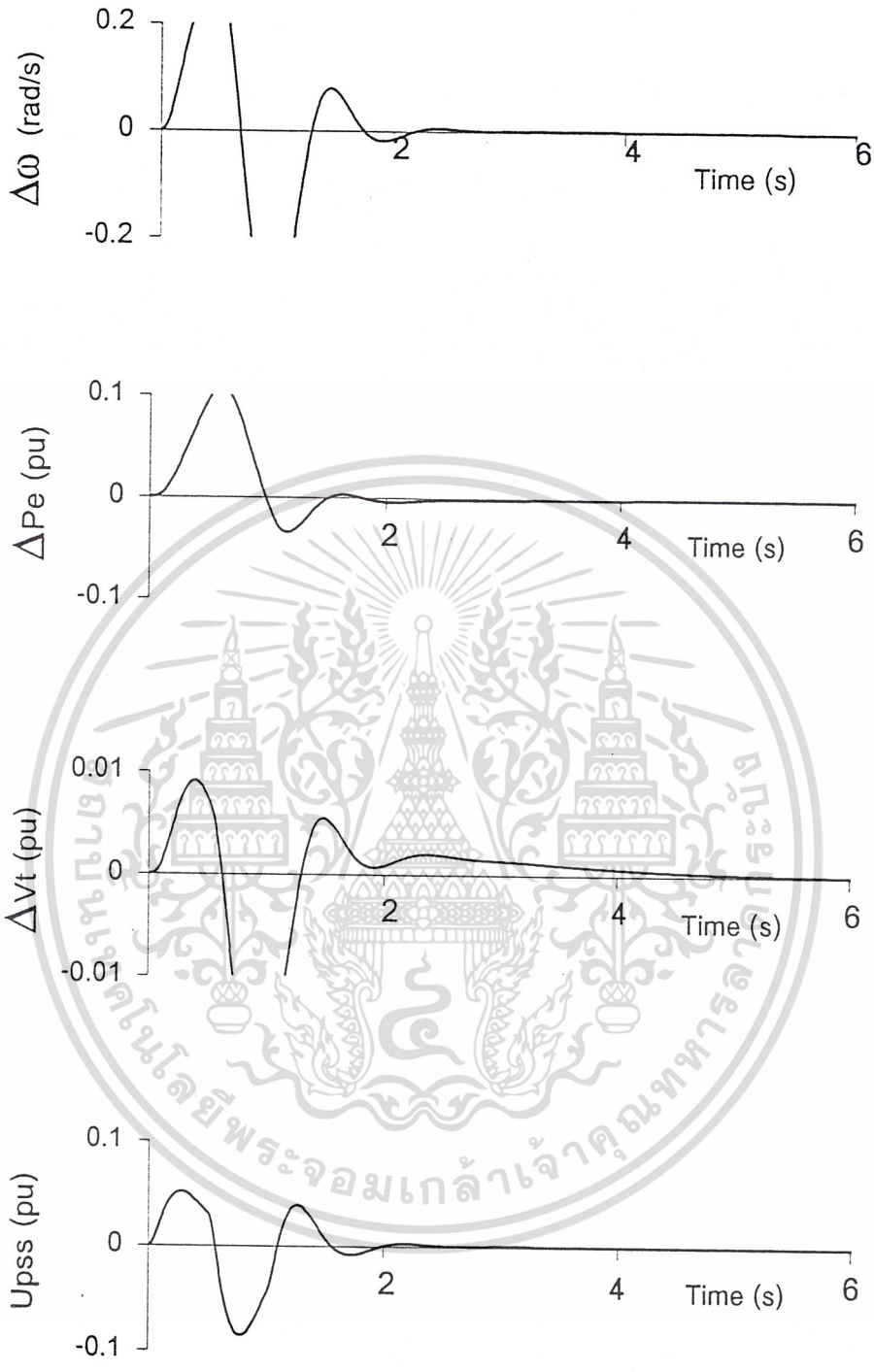
รูปที่ 6.4 ค่าฟิทเนสของ ΔP-PSS ที่ได้จากการรัน 50 เจนเนอเรชัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6.5 ผลการตอบสนองของตัวควบคุม ΔP-PSS โดยใช้โปรแกรม ANGEL.C
เมื่อมีการรบกวนแบบรุนแรง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6.2 ผลการตอบสนองของตัวควบคุม ΔP-PSS โดยใช้โปรแกรม ANGEL.C เมื่อมีการรบกวนแบบเล็กน้อย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

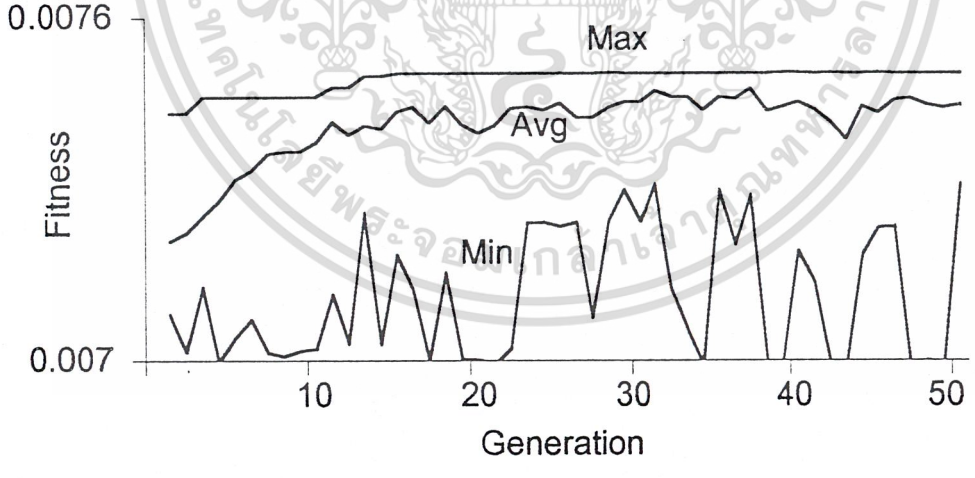
6.5 ผลการตอบสนองของตัวควบคุมแบบ $\Delta\omega$-PSS โดยใช้โปรแกรมเจเนติกอัลกอริทึมออกแบบ

ตารางที่ 6.4 ค่าตัวแปรของตัวควบคุมแบบ $\Delta\omega$-PSS ที่ได้จากโปรแกรม GA

ชนิดของตัวควบคุม	ค่าตัวแปรที่ได้	ดัชนีทำงาน(Jw)
$\Delta\omega$-PSS	Kp=14.7, T1=0.3, T2=0.08, T3=0.2, T4=0.15	333178.05

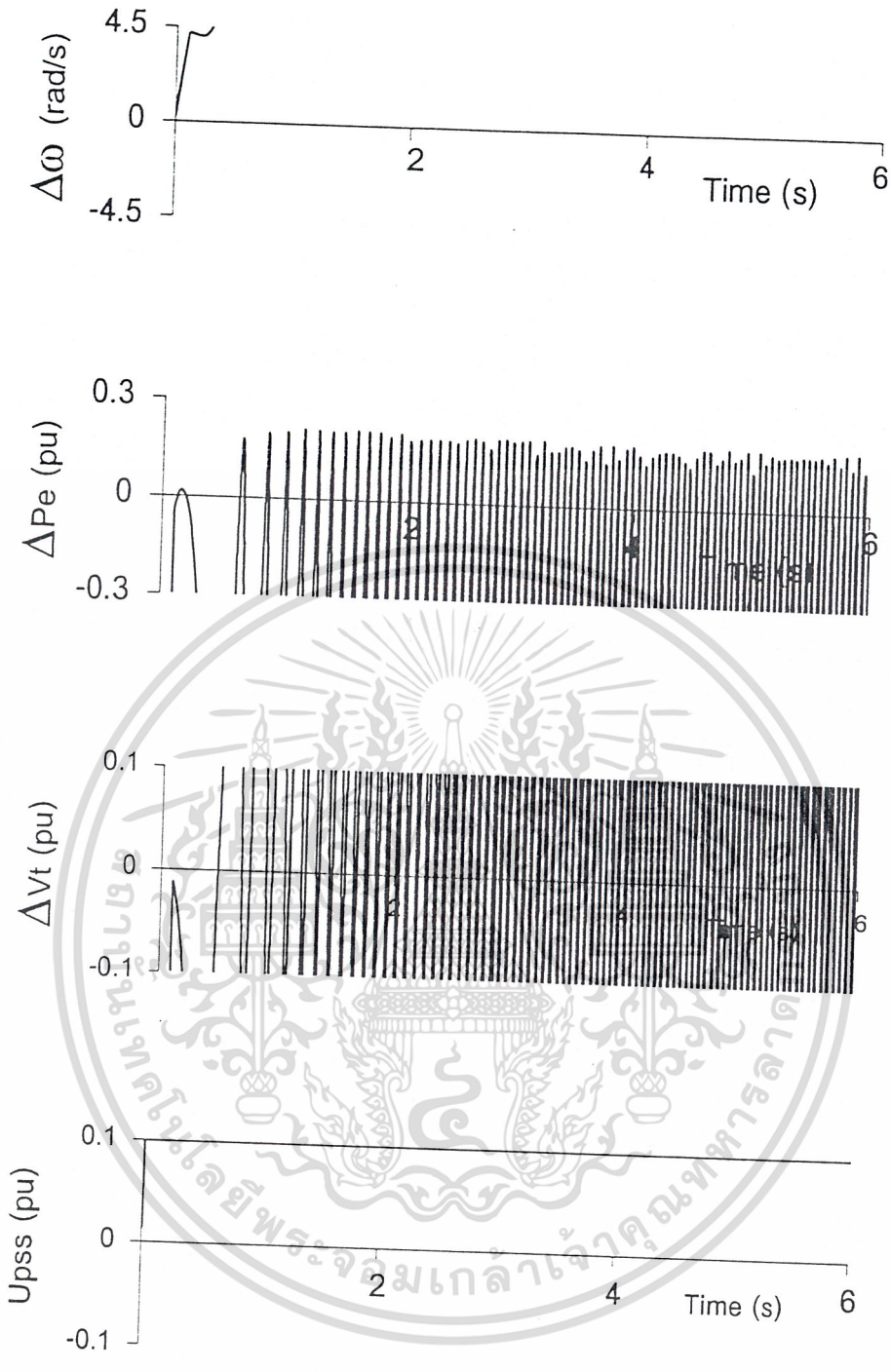
จากค่าดัชนีทำงาน(performance index : Jw) สามารถหาค่าฟิทเนสได้ และนำมาแสดงด้วยรูปที่ 6.7 เป็นค่าฟิทเนสของ $\Delta\omega$-PSS ซึ่งเป็นผลของค่าฟิทเนสที่ได้จากการรัน 50 เจนเนอเรชัน

สัญญาณควบคุมที่ได้จาก $\Delta\omega$-PSS จะได้ผลตอบสนองต่อการรบกวนแบบรุนแรงแสดงได้ดังรูปที่ 6.8 และผลตอบสนองต่อการรบกวนแบบเล็กน้อยแสดงได้ดังรูปที่ 6.9 ซึ่งจะเป็นสัญญาณป้อน Upss ให้แก่ AVR



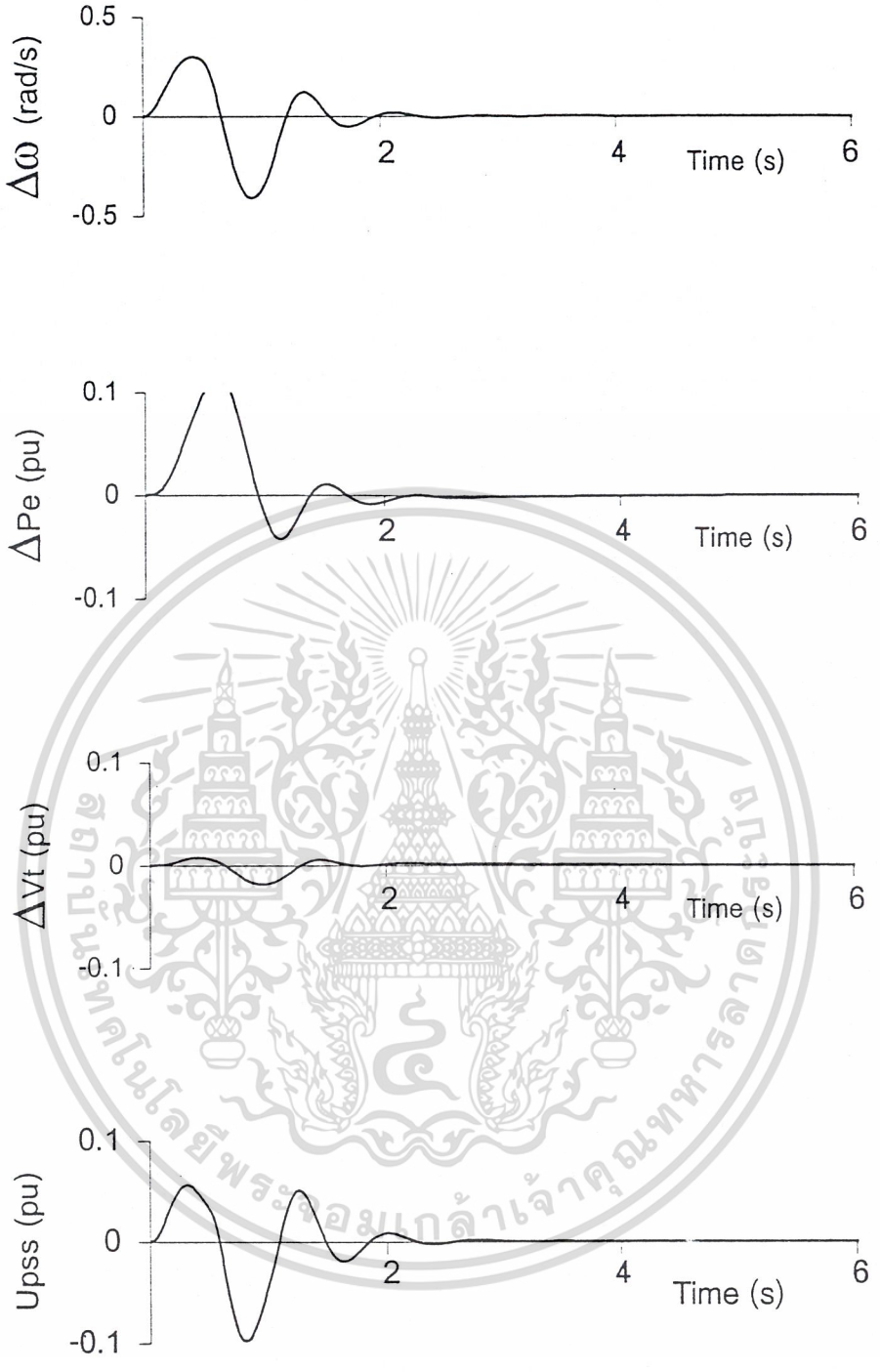
รูปที่ 6.7 ค่าฟิทเนสของ $\Delta\omega$-PSS ที่ได้จากการรัน 50 เจนเนอเรชัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6.8 ผลการตอบสนองของตัวควบคุม $\Delta\omega$-PSS โดยใช้โปรแกรม ANGEL.C เมื่อมีการรบกวนแบบรุนแรง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6.9 ผลการตอบสนองของตัวควบคุม $\Delta\omega$-PSS โดยใช้โปรแกรม ANGEL.C
เมื่อมีการรบกวนแบบเล็กน้อย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.6 ผลการตอบสนองของตัวควบคุมแบบ SVC ชนิดฟัซซี่ร่วมกับ $\Delta\omega$-PSS โดยใช้โปรแกรมเจเนติกอัลกอริทึมออกแบบ

ตารางที่ 6.5 การเปรียบเทียบค่าของตัวแปรจากเอกสารอ้างอิง*[10] และค่าของตัวแปรของตัวควบคุมแบบ SVC ชนิดฟัซซี่ ร่วมกับ $\Delta\omega$-PSS ที่ออกแบบโดยโปรแกรม (ANGEL.C)

ชนิดของตัวควบคุม	ค่าของตัวแปร	ดัชนีทำงาน(Jw)
$\Delta\omega$-PSS*	Kp=13.1*, T1=0.3*,T2=0.1*, T3=0.5*, T4=0.06*	841.64*
SVC ชนิดฟัซซี่*	α_1=0.006*, α_2=93.0*, α_3=0.51*	
$\Delta\omega$-PSS	Kp=8.8, T1=0.6,T2=0.08, T3=0.3, T4=0.08	839.98
SVC ชนิดฟัซซี่	α_1=0.006, α_2=94.0, α_3=0.53	

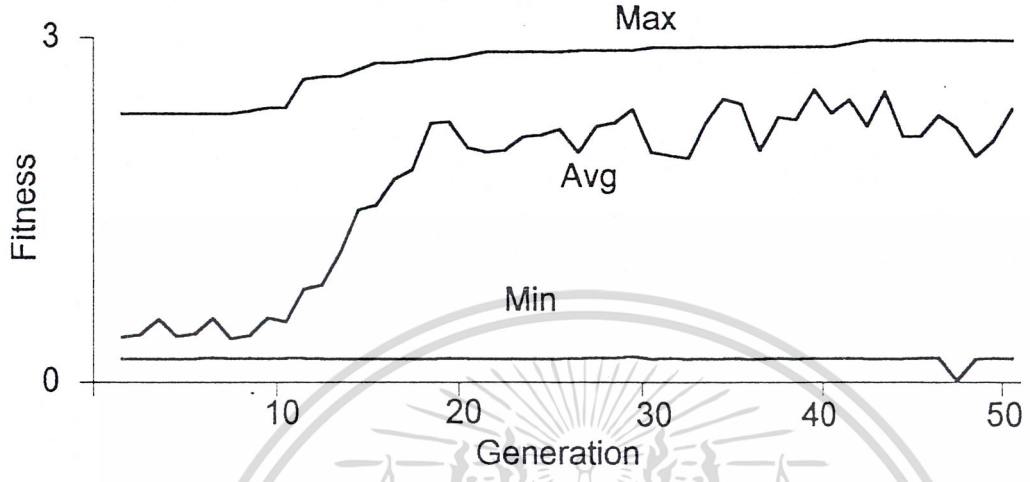
(* ค่าของตัวแปรที่ได้จากบทความวิชาการเรื่อง " การปรับค่าพารามิเตอร์ควบคุมให้เหมาะสมสูงสุดของตัวควบคุม$\Delta\omega$-PSS และตัวควบคุมฟัซซี่ของ SVC สำหรับการควบคุมเสถียรภาพระบบไฟฟ้ากำลัง โดย คมสันต์ หงษ์สมบัติ, มณฑล ลีลาจินดาไกรฤกษ์")

จากตารางจะพบว่าค่าของตัวแปรที่ใกล้เคียงกันมากและเกือบจะทำให้ค่าดัชนีทำงานเท่ากันจึงพอจะเห็นแนวโน้มของการเข้าหาคำตอบของการออกแบบด้วยโปรแกรม ANGEL.C ว่าระบบมีเสถียรภาพอยู่

จากค่าดัชนีทำงาน(performance index : Jw) สามารถหาค่าฟิทเนสได้ และนำมาแสดงด้วยรูปที่ 6.10 เป็นค่าฟิทเนสของแบบ SVC ชนิดฟัซซี่ ร่วมกับ $\Delta\omega$-PSS ซึ่งเป็นผลของค่าฟิทเนสที่ได้จากการรัน 50 เจนเนอเรชัน

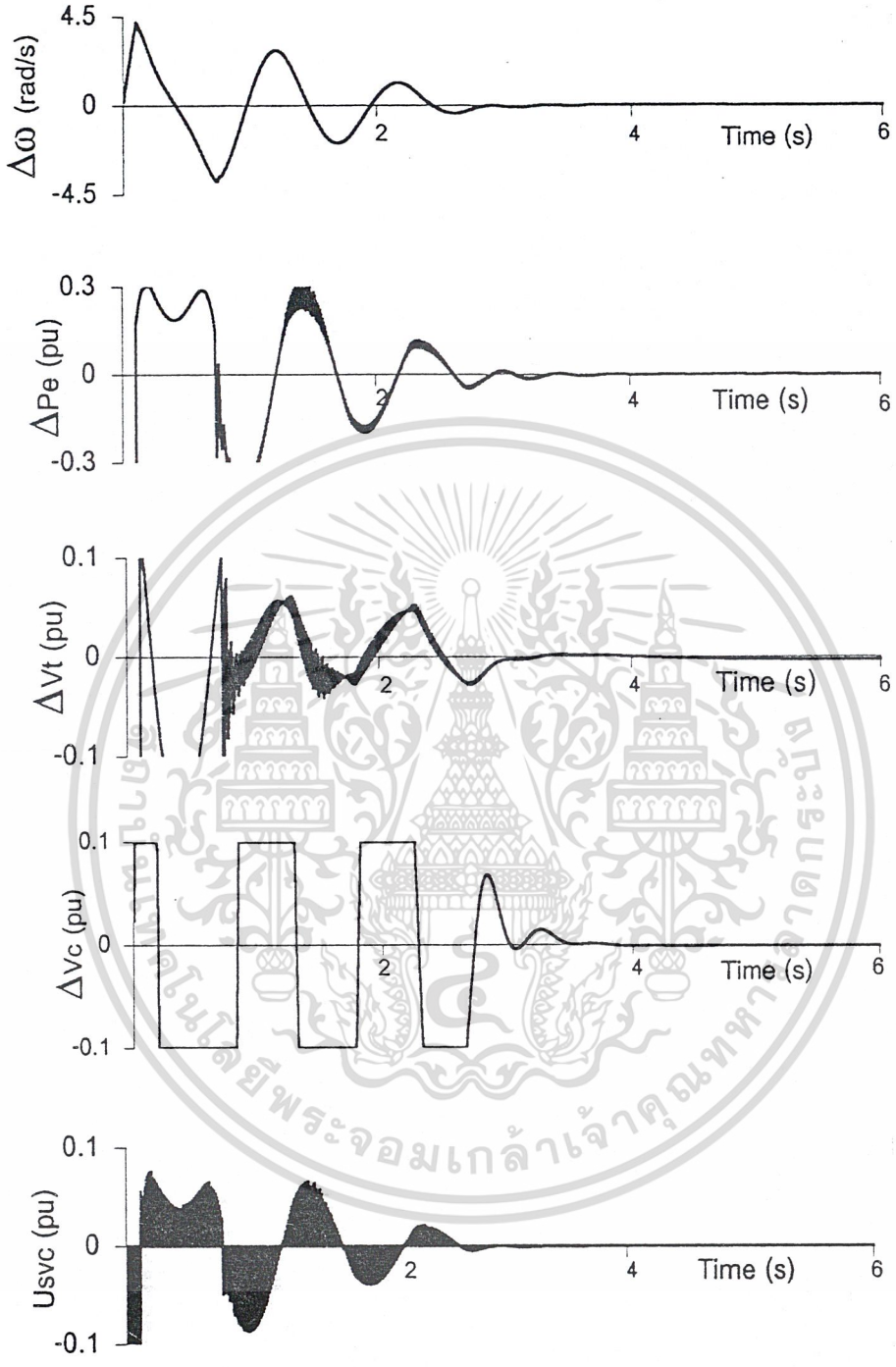
สัญญาณควบคุมที่ได้จากตัวควบคุมแบบ SVC ชนิดฟัซซี่ ร่วมกับ $\Delta\omega$-PSS จะได้ผลตอบสนองต่อการรบกวนแบบรุนแรงแสดงได้ดังรูปที่ 6.11 และผลตอบสนองต่อการรบกวนแบบเล็กน้อยแสดงได้ดังรูปที่ 6.12 ซึ่งจะเป็นสัญญาณป้อน Upss ให้แก่ AVR

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



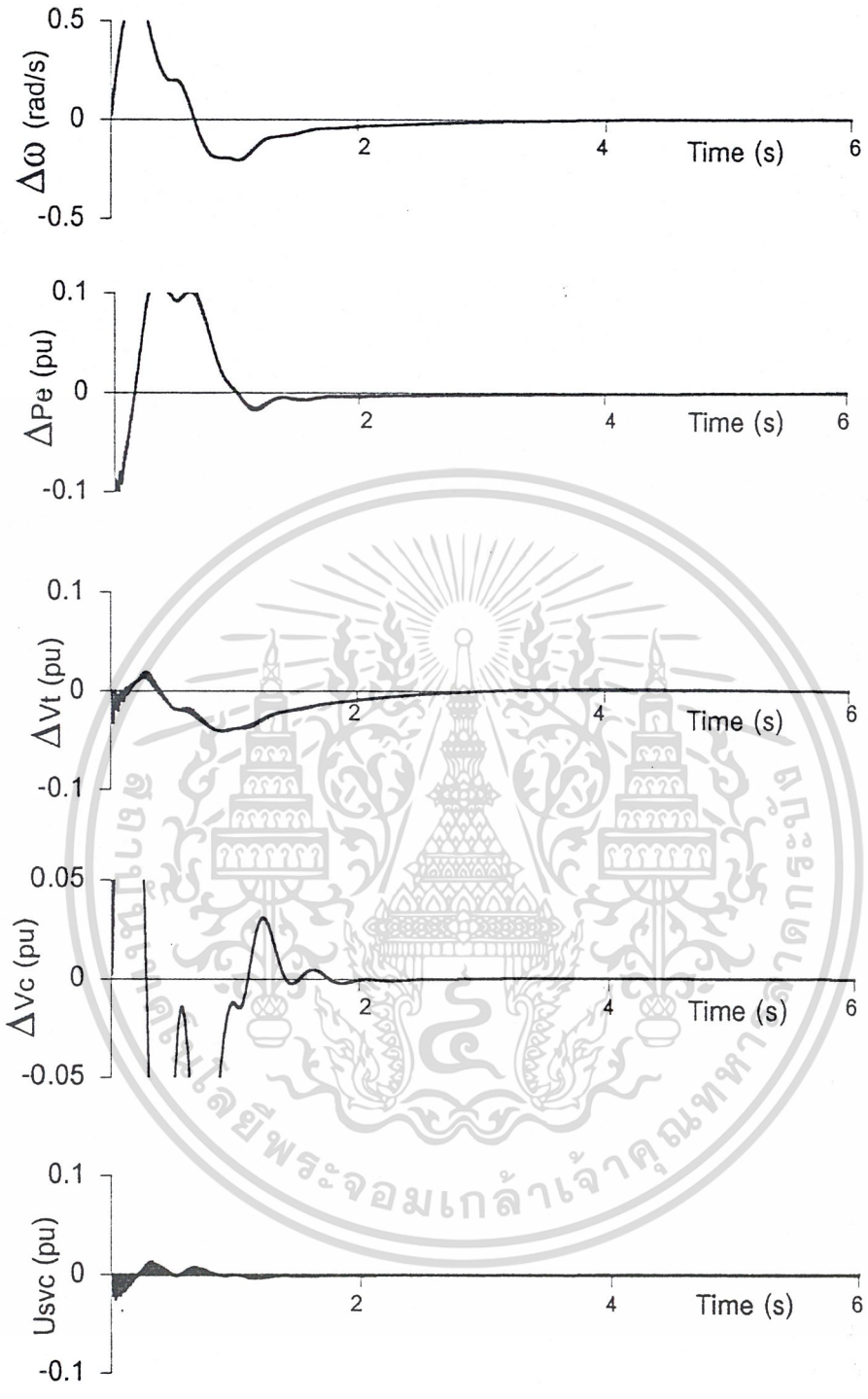
รูปที่ 6.10 ค่าฟิทเนสของตัวควบคุมแบบ SVC ชนิดฟัซซี่ ร่วมกับ $\Delta\omega$-PSS ที่ได้จากการรัน 50 เจนเนอเรชัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6.11 ผลการตอบสนองของตัวควบคุม $\Delta\omega$-PSS ร่วมกับ SVC ชนิดฟัซซี่โดยใช้โปรแกรม ANGEL.C เมื่อมีการรบกวนแบบรุนแรง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 6.12 ผลการตอบสนองของตัวควบคุม $\Delta\omega$-PSS ร่วมกับ SVC ชนิดฟัซซี่โดยใช้โปรแกรม ANGEL.Cเมื่อมีการรบกวนแบบเล็กน้อย

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 7
สรุปผลและข้อเสนอแนะ

จากการนำเอา GA มาใช้ในการออกแบบหาค่าของตัวแปรสำหรับอุปกรณ์ควบคุมแบบต่างๆ โดยการหาค่าที่เหมาะสมที่สุดสำหรับแต่ละระบบจำลอง เพื่อใช้ในการรักษาเสถียรภาพ ปรับปรุงและพัฒนาระบบไฟฟ้ากำลังต่อไป จึงสามารถสรุปผลการทดลองได้ดังนี้

1.GA จะเป็นวิธีการหาคำตอบ หรือค่าที่เหมาะสมที่สุดให้กับตัวแปรของอุปกรณ์ควบคุมต่างๆ ได้เป็นอย่างดีโดยมีความเร็วในการทำงานสูงและค่าที่เหมาะสมที่สุดนั้นทำให้มีเสถียรภาพมาก

2.การใช้อุปกรณ์ควบคุมแบบฟัซซี่และอุปกรณ์ควบคุมแบบไฮบริดโดยใช้ข้อมูลจากเฟสเพลนที่มีการเลื่อนแกน สามารถใช้รักษาเสถียรภาพได้ในกรณีที่มีการรบกวนแบบรุนแรงไม่ได้ ทั้งนี้เพราะการควบคุมแบบ Excitation Control ไม่สามารถควบคุมเสถียรภาพเมื่อเกิดการรบกวนแบบรุนแรงได้ หากต้องการที่ใช้ตัวควบคุมแบบไฮบริดควบคุมให้ได้ต้องมีการปรับเทคนิคให้ถูกต้องเหมาะสมกว่านี้

3.การใช้อุปกรณ์ควบคุม PSS ชนิด ΔP-PSS และ $\Delta\omega$-PSS เมื่อระบบมีการรบกวนแบบเล็กน้อยจะสามารถรักษาเสถียรภาพได้ดี แต่เมื่อมีการรบกวนแบบรุนแรงจะไม่สามารถรักษาเสถียรภาพได้ ทั้งนี้วิธีการทาง GA ถือว่าสามารถออกแบบได้ดี

4.วิธีการทาง GA ที่ใช้ออกแบบค่าของตัวแปรให้กับอุปกรณ์ควบคุม SCV ชนิดฟัซซี่ ร่วมกับ $\Delta\omega$-PSS ได้ผลดีทั้งในกรณีการรบกวนแบบเล็กน้อยและการรบกวนแบบรุนแรง (อุปกรณ์ควบคุมแบบ SVC ชนิดฟัซซี่สามารถใช้รักษาเสถียรภาพได้ทั้งในกรณีการรบกวนแบบเล็กน้อยและการรบกวนแบบรุนแรง)

5.การออกแบบหาค่าตัวแปรที่เหมาะสมด้วยวิธีการทาง GA จะใช้เวลาในการทำงานรวดเร็วมาก โดยใช้เวลาดังนี้(ในกรณีการรันที่ 50 เจนเนอเรชัน)

6.1 สำหรับการควบคุมด้วยอุปกรณ์ไฮบริดใช้เวลาประมาณ 30 นาที

6.2 สำหรับการควบคุม ΔP-PSS ใช้เวาประมาณ 28 นาที

6.3 สำหรับการควบคุมด้วย $\Delta\omega$-PSS ใช้เวลาประมาณ 28 นาที

6.4 สำหรับการควบคุมด้วย SVC ชนิดฟัซซี่ ร่วมกับ $\Delta\omega$-PSS ใช้เวลาประมาณ 13 นาที

ซึ่งสามารถประหยัดเวลามากกว่าการหาค่าที่เหมาะสมด้วยวิธีการทดสอบทุกค่าของตัวแปรแต่ละตัวเป็นอย่างมาก

เอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

วิจารณ์ผลการทดลอง

ในกรณีของการหาค่าตัวแปรที่เหมาะสมสำหรับอุปกรณ์ควบคุมแบบไฮบริด ช่วงค่าของตัวแปรยังไม่เหมาะสมพอ จึงทำให้การรักษาเสถียรภาพโดยวิธีฟัซซี่ที่ใช้ข้อมูลจากเฟสเพลนและมีการเลื่อนแกนอาจยังไม่เหมาะสมจึงทำให้ยังไม่สามารถรักษาเสถียรภาพในกรณีการรบกวนแบบรุนแรงได้

การควบคุมแบบ Excitation เป็นเรื่องยากที่จะทำให้ระบบเสถียรภาพอยู่ได้ จะต้องใช้เทคนิคการควบคุมขั้นสูงที่มีประสิทธิภาพ

ข้อดีและข้อเสีย

ข้อดี 1.วิธีการของ GA ทำให้แก้ปัญหาที่ยังไม่ทราบค่าและต้องการค่าตัวแปรที่เหมาะสมที่สุดได้เป็นอย่างดี

2.เวลาที่ใช้ในการหาค่าตัวแปรใช้เวลาน้อยมากเมื่อเทียบกับการหาค่าแบบที่ต้องทดสอบทุกค่าของตัวแปร

3.สามารถใช้แก้ปัญหากับรูปแบบของระบบจำลองได้หลากหลายมาก

ข้อเสีย 1.หากช่วงตัวแปรไม่เหมาะสม จะทำให้การหาคำตอบของ GA ทำได้ยากหรืออาจไม่ได้คำตอบที่เหมาะสมเลยก็ได้

2.ผลตอบสนองที่ได้จากระบบจำลองต่างๆ อาจจะยังไม่เรียบทีเดียว แต่จะมีการเพียงเล็กน้อยในสภาวะคงที่

ข้อเสนอแนะ

1. การเปลี่ยนสัญญาณอินพุทของวิธีฟัซซี่ในอุปกรณ์ควบคุมแบบไฮบริดจาก ΔPe เป็น $\Delta\omega$ น่าจะทำให้ผลตอบสนองในการเข้าสู่สมดุลได้ดี
2. อุปกรณ์ควบคุมแต่ละชนิดมีจำกัดในการควบคุมหรือรักษาเสถียรภาพ ดังนั้นในการเลือกอุปกรณ์ควบคุมต้องเลือกให้เหมาะสมกับความต้องการรักษาเสถียรภาพ
3. GA สามารถนำมาประยุกต์ใช้ในทางวิศวกรรมหรือปัญหาที่ต้องการทราบค่าที่ดีที่สุด แต่ก็ต้องพิจารณาความเหมาะสมการใช้ GA กับปัญหานั้นๆ ด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ก.

โปรแกรมเจเนติกอัลกอริทึม

ANGEL.C

เป็นโปรแกรมที่ใช้ในการออกแบบหาค่าตัวแปรต่างๆ ของอุปกรณ์ควบคุมแบบไฮบริด และอุปกรณ์ควบคุมแบบ PSS รวมทั้งการควบคุมที่ใช้ $\Delta\omega$ - PSS ร่วมกับ SVC ในภาคผนวก ก. นี้ จะนำเสนอในส่วนของการออกแบบตัวควบคุมแบบไฮบริดเท่านั้น ส่วนอีก 3 ประเภทที่เหลือ จะนำเสนอในส่วนที่มีการเปลี่ยนแปลงเฉพาะแต่ละฟังก์ชันของโปรแกรม

```
/*****************************
---ANGEL.C---
*****************************/
/**** Main program of Power System Stabilizers
Using Genetic Algorithm ****/
/**** Songsak  Muang-ngarn  [Dang]
/**** Kongkiat Gulglangdon [Noom]
/**** Tawin   Phoomthan  [Win ]
/**** King Mongkut's Institute of Technology
Ladkrabang(KMITL),Thailand.***/
/*************************************
*************************************/
#include<stdio.h>
#include<conio.h>
#include<float.h>
#include<math.h>
#include<time.h>
#include<dos.h>
#include<stdlib.h>

#include"c:\tc\model3.c"     /* performance index
calculation */
#include"c:\ga\rand3-2.c"          /* random
algorithm */
#define LINELENGTH 80          /* width of
printout */
#define BITS_PER_BYTE 8          /* number of
bits per byte on this machine */
#define INTSIZE (BITS_PER_BYTE*sizeof
(unsigned))   /* # of bits in unsigned */

/* function prototype */
void fitness(struct individual *critter);
void generation();
void initialize();
void initialdat();
void initialpop();
void initialreport();
void initialmemo();
void freeall();
void nomemory(char *string);
void advdata();
void advfree();
void advmalloc();

/*Advance_random();*/
int   flip(float prob);
void  initrandomnormaldeviate();
double noise(double mu ,double sigma);
void  randomized();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
double randomnormaldeviate();
float  randomperc();
int    rnd(int low,int high);
float  rndreal(float lo ,float hi);

void preselect();
int  roulette();
void select_memory();
void select_free();
void reset();
void statistics(struct individual *pop);
void head();
void repchar (FILE *outfp,char *ch,int repcount);
void skip(FILE *outfp,int skipcount);
int  understand(int i,int j,unsigned *from);
void mutation(unsigned *child);
void crossover(unsigned *parent1,unsigned
*parent2
                    ,unsigned *child1 ,unsigned
*child2);
void report();
void writepop();
void writechrom(unsigned *chrom);
void decode_fitness(struct individual *critter);
void writechrom_bestfitness(unsigned *chrom);
void print_fitness(struct individual *critter1);
FILE *outfp, *infp;
FILE *fpga;

/* Structures and variables */
struct individual
{
  unsigned *chrom;   /* chromosome string for
the individual */
  double  fitness;  /* fitness of the individual */
  int     xsite;    /* crossover site at mating(use
for 1-point crossover) */
  int     parent[2]; /* who the parents of offspring
were */
  int     *utility;  /* utility field can be used as
pointer to a */
                     /* dynamically
allocated, application-specific data structure */
};
struct bestever
{
  unsigned *chrom;   /* chromosome string for
the best-ever individual */
  double  fitness;   /* fitness of the best-ever
individual */
  int     generation; /* generation which produced
it */
};

struct individual *oldpop;  /* last generation of
individuals */
struct individual *newpop;  /* next generation of
individuals */
struct individual *masku;   /* mask for uniform &
2-point crossover */
static struct bestever bestfit;   /* fittest individual
so far */
static double sumfitness;        /* summed fitness
for entire population */
static double max;               /* maximum fitness
of population */
static double avg;               /* average fitness of
population */
static double min;               /* minumum fitness
of population */
```

เอกสารนี้สงวนลิขสิทธิ์สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
float pcross,pcross0;       /* probability of
crossover */
float pmutation,pmutation0; /* probability of
mutation  */
int   num;          /* number of open files */
int   popsize;          /* population size */
int   lchrom;           /* length of the chromosome
per individual */
int   chromsize;        /* number of bytes needed
to store lchrom string */
int   gen;              /* current generation number
*/
int   maxgen;           /* maximum generation
number */
int   run;              /* current run number */
int   maxruns;          /* maximum number of
runs to make */
int   printstrings;     /* flag to print chromosome
strings (default on) */
double nmutation;       /* number of mutations
*/
double ncross;          /* number of crossovers  */
static int field, vector; /* field and vector size */

static int  *roulettelist, roulettepos, roulettesize; /*
for tselection */

/* variables are declared static so that they cannot
conflict with names of   */
/* other global variables in other files.  See K&R,
p 80, for scope of static */
/*double oldrand[55];*/   /* Array of 55 random
numbers */
static int jrand;           /* current random number
*/
static double rndx2;        /* used with random
normal deviate */
static int rndcalcflag;     /* used with random
normal deviate */
double start,finish;        /* used for clock */
struct  time t;             /* used for clock */
int hour1,hour2,
   min1,min2,
   sec1,sec2,
   sechund1,sechund2;
time_t tt;                  /* used for initial seed */
static int elitism ;              /* used for elitist
selection */
/*unsigned G[24];*/

void main()
{
   struct individual *temp;
   FILE   *fopen();

   fpga=fopen("case3.dat","w"); /* data output file
*/
   printstrings=0;field=16;
   elitism=1;roulettesize=2;
   num=0;
   infp=stdin;
   outfp=stdout;

                              /* starting */
   head();
   initialize();
   gettime(&t);    /* get start time */
   hour1 = t.ti_hour;
   min1  = t.ti_min;
   sec1  = t.ti_sec;
   sechund1 = t.ti_hund;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        for(gen=0; gen<maxgen; gen++)
        {
          fprintf(outfp,"\nGENERATION %d->%d\n",gen,maxgen);
          generation();     /* create a new generation */
          statistics(newpop); /* compute fitness statistics on new populations */
          report();         /* report results for new generation */

          temp = oldpop;    /* advance the generation */
          oldpop = newpop;
          newpop = temp;

        }
        gettime(&t);        /* get finish time */
        hour2 = t.ti_hour;
        min2  = t.ti_min;
        sec2  = t.ti_sec;
        sechund2 = t.ti_hund;
        printf("\nThe start process time is: %2d:%02d:%02d.%02d\n",
        hour1,min1,sec1,sechund1);
        fprintf(fpga,"\n");
        fprintf(fpga,"Global Best Individual so far, Generation %d:\n",
                bestfit.generation);
        print_fitness(&(bestfit));
        fprintf(fpga,"Jw1 = %f  Jw2 = %f\n",Jw1,Jw2);
        fprintf(fpga,"performance index = %f: ", (1/bestfit.fitness)*2500);
        fprintf(fpga,"\nThe start process time is: %2d:%02d:%02d.%02d\n",
        hour1,min1,sec1,sechund1);
        printf("The stop process time is: %2d:%02d:%02d.%02d\n",
        t.ti_hour, t.ti_min, t.ti_sec, t.ti_hund);
        fprintf(fpga,"The stop process time is: %2d:%02d:%02d.%02d\n",
        t.ti_hour, t.ti_min, t.ti_sec, t.ti_hund);
        freeall(); fclose(fpga);

}
void head()
{
  int iskip;
  int ll = 59;

  iskip = (LINELENGTH - ll)/2;
  skip(outfp,1);
  repchar(outfp," ",iskip); repchar(outfp,"*",ll);
skip(outfp,1);
  repchar(outfp," ",iskip-1);
  fprintf(outfp,"  POWER  SYSTEM STABILIZERS  USING  GENETIC ALGORITHM  \n");
  fprintf(outfp,"                    Songsak Muang-ngam          \n");
  fprintf(outfp,"                    Tawin Phoomthan             \n");
  fprintf(outfp,"                    Kongkiat Gulglangdon        \n");
  repchar(outfp," ",iskip);
  repchar(outfp,"*",ll); skip(outfp,2);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void advdata()
/* application dependent data input, called by
init_data() */
/* ask your input questions here, and put output in
global variables */
/* In this example application, the utility pointer of
each individual is  */
/* assigned a vector of integers that will be used to
store the interpreted */
/* chromosome. */
{
  int size = INTSIZE;

  if(lchrom < INTSIZE) size = lchrom;
  /* user must specify length of concatenated
integers in the chromosome. */
  /*fprintf(outfp," Enter field size (must be less
than %d) ->", size);
  fscanf(infp,"%d",&field);*/
  vector = lchrom/field;
  if((((float)lchrom/(float)field)-vector) > 0.0)
vector++;
}

void advfree()
/* This routine should free any memory allocated
*/
/* in the application-dependent routines, called by
freeall() */
{
  int i;

  for(i = 0; i < popsize; i++)
  {
      free(newpop[i].utility);
      free(oldpop[i].utility);
  }
}

void advmalloc()
/* application dependent malloc() calls, called by
initialmemo() */
{
  unsigned nbytes;
  int i;

  nbytes = vector * sizeof(int);
  for(i = 0; i < popsize; i++)
  {
      if((newpop[i].utility = (int *) malloc
(nbytes)) == NULL)
          nomemory("newpop utility");
      if((oldpop[i].utility = (int *) malloc
(nbytes)) == NULL)
          nomemory("oldpop utility");
  }
}

void advreport()
/* Application-dependent report, called by report()
*/
{
  int i, j;

  /* Print vector interpretation of ech
chromosome. */
  for(i = 0; i < popsize; i++)
  {
      fprintf(outfp,"oldpop %d = ", i);
      for(j = 0; j < vector; j++)
          fprintf(outfp, "%d ", oldpop[i].utility
[j]);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        fprintf(outfp,"\n");
    }
}

void generation()
{
  int mate1, mate2, jcross, j = 0;
  int k;
  /* perform any preselection actions necessary
before generation */
  preselect();
  pcross = pcross0;
  pmutation = pmutation0;
  /* select, crossover, and mutation */
  do
   {
    /* pick a pair of mates */
    mate1=roulette();
    mate2=roulette();
    /* Crossover and mutation */
    crossover(oldpop[mate1].chrom, oldpop
[mate2].chrom,
                          newpop[j].chrom,
newpop[j+1].chrom);
    mutation(newpop[j].chrom);
    mutation(newpop[j+1].chrom);
    pcross -= (float)((pcross0 - 0.6)/maxgen);
/* decrease crossver probability*/
    pmutation += (float)((0.1 -
pmutation0)/maxgen); /* increase mutation
probability*/
    /* Decode string, evaluate fitness, & record */
    /* parentage date on both children */
    fitness(&(newpop[j]));
    newpop[j].parent[0] = mate1+1;
    newpop[j].xsite = jcross;
    newpop[j].parent[1] = mate2+1;
    fitness(&(newpop[j+1]));
    newpop[j+1].parent[0] = mate1+1;
    newpop[j+1].xsite = jcross;
    newpop[j+1].parent[1] = mate2+1;
    /* Increment population index */
    j = j + 2;
   }
  while(j < (popsize-1));
  statistics(newpop);
  /* perform elitist selection */
  if(elitism)
  {
    /*writechrom((&newpop[0])->chrom);printf
("\n");*/ /* test before change */
     for(k = 0; k < chromsize; k++)
          {
              newpop[0].chrom[k] = bestfit.chrom
[k];
          }
    fitness(&(newpop[0]));
    /*  writechrom((&newpop[0])->chrom);printf
("\n");
    writechrom((&bestfit)->chrom);printf
("\n");*//* test after change */
  }
}

void initialize()  /* Initialization Coordinator */
{

  initialdat();  /* get basic problem values from
input file */
  /* define chromosome size in terms of machine
bytes, ie */
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่อนุญาตให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    /* length of chromosome in bits (lchrom)/(bits-
per-byte) */
    /* chromsize must be known for malloc() of
chrom pointer */
    chromsize = (lchrom/INTSIZE);
    if(lchrom%INTSIZE) chromsize++;
    initialmemo(); /* malloc space for global data
structures */
    randomized(); /* initialize random number
generator */
    /* initialize global counters/values */
    nmutation = 0;
    ncross = 0;
    bestfit.fitness = 0.0;
    bestfit.generation = 0;
    /* initialize the populations and report statistics
*/
    initialpop();
    statistics(oldpop);
    initialreport();
}

void initialdat()   /* data inquiry and setup */
{
   char  answer[2];

   if(num == 0)
   {
          fprintf(outfp,"\n::::::::::GA Factors
Limit::::::::::\n");
          fprintf(outfp,"Population size
==========> ");
          fprintf(fpga,"\n::::::::::GA Factors
Limit:::::::::: \n");
          fscanf(infp,"%d", &popsize);
          fprintf(fpga,"Population size
==========> %d \n",popsize);

   if((popsize%2) != 0)
    {
          fprintf(outfp, "Sorry! only even
population sizes are allowed. \n Incrementing
popsize by one.\n");
          popsize++;
    };

    if(num == 0)
          fprintf(outfp,"Chromosome length(161)
==========> ");
    fscanf(infp,"%d", &lchrom);
          fprintf(fpga,"Chromosome length(161)
==========> %d \n",lchrom);
    if(num == 0)
          fprintf(outfp,"Maximum generations
==========> ");
    fscanf(infp,"%d", &maxgen);
    fprintf(fpga,"Maximum generations
==========> %d \n",maxgen);
    if(num == 0)
          fprintf(outfp,"Pcrossover
==========> ");
    fscanf(infp,"%f", &pcross0);
    fprintf(fpga,"Pcrossover          ==========>
%f \n",pcross0);
    if(num == 0)
          fprintf(outfp,"Pmutation
==========> ");
    fscanf(infp,"%f", &pmutation0);
    fprintf(fpga,"Pmutation ==========>
%f \n",pmutation0);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    advdata(); /* any application-dependent global
input */
}

void initialpop() /* Initialize a population at
random */
{
    int j, j1, k, stop;
    unsigned mask = 1;

    for(j = 0; j < popsize; j++)
    {
        for(k = 0; k < chromsize; k++)
        {
            oldpop[j].chrom[k] = 0;
            if(k == (chromsize-1))
                stop = lchrom - (k*INTSIZE);
            else
                stop = INTSIZE;
            /* A fair coin toss */
            for(j1 = 1; j1 <= stop; j1++)
            {
              oldpop[j].chrom[k] = oldpop
[j].chrom[k]<<1;
              if(flip(0.5))
                  oldpop[j].chrom[k] = oldpop
[j].chrom[k]|mask;
            }
        }
        oldpop[j].parent[0] = 0; /* Initialize
parent info. */
        oldpop[j].parent[1] = 0;
        oldpop[j].xsite = 0;
        fitness(&(oldpop[j])); /* Evaluate initial
fitness */
    }
}

void initialreport() /* Initial report */
{
    void skip();

    fprintf(fpga," GA Parameters\n");
    fprintf(fpga," ----------------------------------------
--------\n");
    fprintf(fpga," Population size          =
%d\n",popsize);
    fprintf(fpga," Chromosome length (lchrom)
=  %d\n",lchrom);
    fprintf(fpga," Maximum generations (maxgen)
=  %d\n",maxgen);
    fprintf(fpga," Pcrossover (pcross)    = %f\n",
pcross0);
    fprintf(fpga," Pmutation  (pmutation)  = %f\n",
pmutation0);
    fprintf(fpga," ----------------------------------------
--------\n\n");
    fprintf(fpga," min    max    avg    fitness
best.fitness    chromosome \n\n");
    fflush(outfp);
}

void initialmemo()
/* memory allocation of space for global data
structures */
{
  unsigned nbytes;
  int j;

  /* memory for old and new populations of
individuals */
  nbytes = popsize*sizeof(struct individual);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  if((oldpop = (struct individual *) malloc(nbytes))
 == NULL)
    nomemory("oldpop");

  if((newpop = (struct individual *) malloc(nbytes))
 == NULL)
    nomemory("newpop");

  if((masku = (struct individual *) malloc(nbytes))
 == NULL)
    nomemory("masku");

  /* memory for chromosome strings in
 populations */
  nbytes = chromsize*sizeof(unsigned);
  for(j = 0; j < popsize; j++)
    {
     if((oldpop[j].chrom = (unsigned *) malloc
(nbytes)) == NULL)
          nomemory("oldpop chromosomes");

     if((newpop[j].chrom = (unsigned *) malloc
(nbytes)) == NULL)
          nomemory("newpop chromosomes");

     if((masku[j].chrom = (unsigned *) malloc
(nbytes)) == NULL)
          nomemory("masku chromosomes");
    }

  if((bestfit.chrom = (unsigned *) malloc(nbytes))
== NULL)
    nomemory("bestfit chromosome");

  select_memory(); /* allocate any auxiliary
memory for selection */
  /* call to application-specific malloc() routines
 */
  /* can be used to malloc memory for utility
 pointer */
  advmalloc();
 }

 void freeall()
 /* A routine to free all the space dynamically
 allocated in initspace() */
 {
  int i;

  for(i = 0; i < popsize; i++)
    {
     free(oldpop[i].chrom);
     free(newpop[i].chrom);
     free(masku[i].chrom);
    }
  free(oldpop);
  free(newpop);
  free(masku);
  free(bestfit.chrom);
  select_free(); /* free any auxiliary memory
 needed for selection */
  /* call to application-specific free() routines  */
  /* can be used to free memory for utility pointer
 */
  advfree();
 }

 void nomemory(char *string)
 {
  fprintf(stderr,"malloc: out of memory making
 %s!!\n",string);
  exit(-1);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
}

int flip(float prob)
/* Flip a biased coin - true if heads */
{
   float randomperc();

   if(randomperc() <= prob)
         return(1);
   else
         return(0);
}

void initrandomnormaldeviate()  /* initialization
routine for randomnormaldeviate */
{
   rndcalcflag = 1;
}

double noise(double mu ,double sigma) /* normal
noise with specified mean & std dev: mu & sigma
*/
{
   double randomnormaldeviate();

   return((randomnormaldeviate()*sigma) + mu);
}


void randomized()  /* Get seed number for random
and start it up */
{
   sran2((unsigned int) time(&tt));
}

double randomnormaldeviate()
/* random normal deviate after ACM algorithm
267 / Box-Muller Method */
{
   double t, rndx1;

   if(rndcalcflag)
   {
         rndx1 = sqrt(- 2.0*log((double)
randomperc()));
         t = 6.2831853072 * (double) randomperc
();
         rndx2 = rndx1 * sin(t);
         rndcalcflag = 0;
         return(rndx1 * cos(t));
   }
   else
   {
         rndcalcflag = 1;
         return(rndx2);
   }
}


float randomperc()
/* Fetch a single random number between 0.0 and
1.0 - Subtractive Method */
/* See Knuth, D. (1969), v. 2 for details */
/* name changed from random() to avoid library
conflicts on some machines*/
{
   return((float)ran2());
}

int rnd(int low,int high)
/* Pick a random integer between low and high */
{
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    int i;

    if(low >= high)
        i = low;
    else
    {
        i = (randomperc() * (high - low + 1)) +
low;
        if(i > high) i = high;
    }
    return(i);
}


float rndreal(float lo ,float hi)
/* real random number between specified limits */
{
    return((randomperc() * (hi - lo)) + lo);
}

void statistics(struct individual *pop)
/* Calculate population statistics */
{
    int i, j;

    sumfitness = 0.0;
    min = pop[0].fitness;
    max = pop[0].fitness;
    /* Loop for max, min, sumfitness */
    for(j = 0; j < popsize; j++)
    {
        sumfitness = sumfitness + pop[j].fitness;
/* Accumulate */
        if(pop[j].fitness >= max) max = pop
[j].fitness;   /* New maximum */
        if(pop[j].fitness <= min) min = pop
[j].fitness;   /* New minimum */
        /* new global best-fit individual */
        if(pop[j].fitness > bestfit.fitness)
        {
          for(i = 0; i < chromsize; i++)
            bestfit.chrom[i]     = pop[j].chrom[i];

          bestfit.fitness    = pop[j].fitness;
          bestfit.generation = gen;
        }
    }
    /* Calculate average */
    avg = sumfitness/popsize;
}

void repchar (FILE *outfp,char *ch,int repcount)
/* Repeatedly write a character to stdout */
{
    int j;

    for (j = 1; j <= repcount; j++) fprintf
(outfp,"%s", ch);
}

void skip(FILE *outfp,int skipcount)  /* Skip
skipcount lines */
{
    int j;

    for (j = 1; j <= skipcount; j++) fprintf
(outfp,"\n");
}

int understand(int i,int j,unsigned *from)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/* interpret bits i thru j of a individual as an integer
*/
/* j MUST BE greater than or equal to i AND j-i <
INTSIZE-1  */
/* from is a chromosome, represented as an array
of unsigneds */
{
   unsigned mask, temp;
   int bound_flag;
   int iisin, jisin;
   int i1, j1, out;

   if(j < i)
   {
         fprintf(stderr,"Error in understand: j <
i\n");
         exit(-1);
   }
   if(j-i+1 > INTSIZE)
   {
         fprintf(stderr,"Error in understand: j-i+1
> INTSIZE\n");
         exit(-1);
   }

   iisin = i/INTSIZE;
   jisin = j/INTSIZE;

   i1 = i - (iisin*INTSIZE);
   j1 = j - (jisin*INTSIZE);

         if(i1 == 0)
         {
               iisin = iisin-1;
               i1 = i - (iisin*INTSIZE);
         };
         if(j1 == 0)
         {
               jisin = jisin-1;
               j1 = j - (jisin*INTSIZE);
         };
   /* check if bits fall across a word boundary */
   if(iisin == jisin)
         bound_flag = 0;
   else
         bound_flag = 1;

   if(bound_flag == 0)
   {
         mask = 1;
         mask = (mask<<(j1-i1+1))-1;
         mask = mask<<(i1-1);
         out = (from[iisin]&mask)>>(i1-1);
         return(out);
   }
   else
   {
         mask = 1;
         mask = (mask<<j1)-1;
         temp = from[jisin]&mask;
         mask = 1;
         mask = (mask<<(INTSIZE-i1+1))-1;
         mask = mask<<(i1-1);
         out = ((from[iisin]&mask)>>(i1-1)) |
(temp<<(INTSIZE-i1+1));
         return(out);
   }
}

void mutation(unsigned *child) /* Mutate an allele
w/ pmutation, count # of mutations */
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
{
   int j, k, stop;
   unsigned mask, temp = 1;

   for(k = 0; k < chromsize; k++)
   {
         mask = 0;
         if(k == (chromsize-1))
            stop = lchrom - (k*INTSIZE);
         else
            stop = INTSIZE;
         for(j = 0; j < stop; j++)
         {
            if(flip(pmutation))
            {
                  mask = mask|(temp<<j);
                  nmutation++;
            }
         }
         child[k] = child[k]^mask;
   }
}

void crossover(unsigned *parent1,unsigned
*parent2
                  ,unsigned *child1 ,unsigned
*child2)
{
 int i, j, b,l, which_way, data_width;
 int  j1, stop;
 unsigned  mask1;
 int cp1,cp2;
 int m;
 int k;

struct individual *idp;
int fraction;

if (flip(pcross))
{
ncross++;
for(k = 0; k < chromsize; k++)
      {
        masku[1].chrom[k] = 0;
        if(k == (chromsize-1))
              stop = lchrom - (k*INTSIZE);
        else
              stop = INTSIZE;
        /* A fair coin toss */
        for(j1 = 1; j1 <= stop; j1++)
        {
         masku[1].chrom[k] = masku
[1].chrom[k]<<1;
         if(flip(0.5))
              masku[1].chrom[k] = masku
[1].chrom[k]|0x1;
        }
      }
 k=(chromsize-1);
 fraction = lchrom-((chromsize-1)*16);
 /* start fraction cross over loop */
m=0x1 << (fraction-1);
mask1=m;
child1[k]=0x0;
child2[k]=0x0;
for(i=1;i<=fraction;i++)
{ cp1=(mask1&(parent1[k]))>>(fraction-i);
 cp2=(mask1&(parent2[k]))>>(fraction-i);
 if ((masku[1].chrom[k]&mask1)>>(fraction-i))
 { child1[k]<<=1; child1[k]=(child1[k]|cp2);
  child2[k]<<=1; child2[k]=(child2[k]|cp1);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  }
  else  { child1[k]<<=1; child1[k]=(child1
[k]|cp1);
          child2[k]<<=1; child2[k]=(child2
[k]|cp2);
          }
 mask1>>=1;
 }
 /* end of fraction cross over loop */
 /* start uniform cross over loop */
 l=(k-1);
 for(k=l;k>=0;k--)
 {
 m=0x1 << 15;
 mask1=m;
 child1[k]=0x0;
 child2[k]=0x0;
 for(i=1;i<=16;i++)
 { cp1=(mask1&(parent1[k]))>>(16-i);
  cp2=(mask1&(parent2[k]))>>(16-i);
  if ((masku[1].chrom[k]&mask1)>>(16-i))
  { child1[k]<<=1; child1[k]=(child1[k]|cp2);
    child2[k]<<=1; child2[k]=(child2[k]|cp1);
  }
  else  { child1[k]<<=1; child1[k]=(child1
[k]|cp1);
          child2[k]<<=1; child2[k]=(child2
[k]|cp2);
          }
 mask1>>=1;
 }
 }
 }
 else
 {
 for(k=0;k<chromsize; k++)
  {
   child1[k] = parent1[k];
   child2[k] = parent2[k];
  }
 }
}

void report()  /* Write the population report */
{
  repchar(outfp,"-",LINELENGTH);
  skip(outfp,1);
  if(printstrings == 1)
  {
        repchar(outfp," ",((LINELENGTH-
17)/2));
        fprintf(outfp,"Population Report\n");
        fprintf(outfp, "Generation %3d", gen);
        repchar(outfp," ",(LINELENGTH-28));
        fprintf(outfp, "Generation %3d\n",
(gen+1));
        fprintf(outfp,"num  string ");
        repchar(outfp," ",lchrom-5);
        fprintf(outfp,"fitness   parents xsite  ");
        fprintf(outfp,"string ");
        repchar(outfp," ",lchrom-5);
        fprintf(outfp,"fitness\n");
        repchar(outfp,"-",LINELENGTH);
        skip(outfp,1);

        writepop();
        repchar(outfp,"-",LINELENGTH);
        skip(outfp,1);

  }
  /* write the summary statistics in global mode
*/
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
   fprintf(outfp,"Generation %d Result of GA:
\n",
                gen);

   fprintf(outfp,"Total Crossovers = %f, Total
Mutations = %f\n",
                        ncross,nmutation);
   fprintf(outfp,"min = %f  max = %f  avg = %f
sum = %f\n",
                    min,max,avg,sumfitness);
   fprintf(fpga," %f  %f  %f  %f  %f
",min,max,avg,sumfitness,bestfit.fitness);
   fprintf(outfp,"Best Fitness = %f: ",
bestfit.fitness);
   fprintf(outfp,"Performance index = %f:\n",
2500*(1/bestfit.fitness));
   fprintf(outfp,"Best Individual from Generation
%d:\n",
                    bestfit.generation);
   writechrom_bestfitness((&bestfit)->chrom);
   writechrom((&bestfit)->chrom);fprintf
(fpga,"\n");
   skip(outfp,1);

   decode_fitness((&bestfit));
   fitness((&bestfit));
   skip(outfp,1);
   repchar(outfp,"-",LINELENGTH);
   skip(outfp,1);
}

void writepop()
{
   struct individual *pind;
   int j;

   for(j=0; j<=popsize-1; j++)
   {
      fprintf(outfp,"%3d)  ",j+1);
      /* Old string */
      pind = &(oldpop[j]);
      writechrom(pind->chrom);
      fprintf(outfp,"  %8f| ", pind->fitness);
      /* New string */
      pind = &(newpop[j]);
      fprintf(outfp,"(%2d,%2d)  %2d  ",
      pind->parent[0], pind->parent[1], pind->
xsite);
      writechrom(pind->chrom);
      fprintf(outfp,"  %8f\n", pind->fitness);
   }
}

void writechrom(unsigned *chrom)
/* Write a chromosome as a string of ones and
zeroes      */
/* note that the most significant bit of the
chromosome is the  */
/* RIGHTMOST bit, not the leftmost bit, as would
be expected...*/
{
   int j, k, stop;
   unsigned mask=1,tmp;
   static int bit_count=1;
   unsigned mask1;
   int m;
   printf(" ");
   bit_count=1;
   for(k = chromsize-1; k>=0; k--)
   {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        tmp = chrom[k];
        if(k == (chromsize-1))
          { stop = lchrom - (k*INTSIZE);m=0x1
<<(stop-1);}
        else
          { stop = INTSIZE;m=0x1<<(stop-1);}
        mask1=m;
        for(j = 0; j < stop; j++)
        {
          if(tmp&mask1)
                fprintf(outfp,"1");
          else
                fprintf(outfp,"0");
          mask1>>=1;
        }
    }
}

void writechrom_bestfitness(unsigned *chrom)
/* Write a chromosome as a string of ones and
zeroes        */
/* note that the most significant bit of the
chromosome is the  */
/* RIGHTMOST bit, not the leftmost bit, as would
be expected... */
{
    int j, k, stop;
    unsigned mask=1,tmp;
    static int bit_count=1;
    unsigned mask1;
    int m;

    bit_count=1;
    for(k = chromsize-1; k>=0; k--)
    {
        tmp = chrom[k];
        if(k == (chromsize-1))
          { stop = lchrom - (k*INTSIZE);m=0x1
<<(stop-1);}
        else
          { stop = INTSIZE;m=0x1<<(stop-1);}
        mask1=m;
        for(j = 0; j < stop; j++)
        {
          if(tmp&mask1)
                fprintf(fpga,"1");
          else
                fprintf(fpga,"0");
          mask1>>=1;
        }
    }
}

void select_memory()
{
  unsigned nbytes;
  int j;

  nbytes = popsize*sizeof(int);
  if((roulettelist = (int *) malloc(nbytes)) ==
NULL)
    nomemory("roulettelist");

 if(roulettesize > popsize)
   {
    fprintf(outfp,"FATAL: Tournament size (%d)
> popsize (%d)\n",
          roulettesize,popsize);
    exit(-1);
   }
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
void select_free()
{
  free(roulettelist);
};

void preselect()
{
   reset();
   roulettepos = 0;
}

int roulette()
{
   int pick, winner, i;
   /* If remaining members not enough for a
tournament, then reset list */
   if((popsize - roulettepos) < roulettesize)
   {
         reset();
         roulettepos = 0;
   }
   /* Select roulettesize structures at random and
conduct a tournament */
   winner=roulettelist[roulettepos];
   for(i=1; i<roulettesize; i++)
   {
         pick=roulettelist[i+roulettepos];
         if(oldpop[pick].fitness > oldpop
[winner].fitness) winner=pick;
   }
   /* Update roulettepos */
   roulettepos += roulettesize;
   return(winner);
}

void reset()  /* Shuffles the roulettelist at random
*/
{
   int i, rand1, rand2, temp;

   for(i=0; i<popsize; i++) roulettelist[i] = i;

   for(i=0; i < popsize; i++)
   {
         rand1=rnd(i,popsize-1);
         rand2=rnd(i,popsize-1);
         temp = roulettelist[rand1];
         roulettelist[rand1]=roulettelist[rand2];
         roulettelist[rand2]=temp;
   }
}

void fitness(struct individual *critter)
{
   unsigned mask=0x1;  /* mask for current bit */
   int bitpos;   /* current bit position */
   unsigned bitpos2=0,bitpos3=0;
   unsigned tp;
   double pow(), bitpow1=0,
coef,bitpow2=0,bitpow3=0;
   int i,j, k, stop;
   int n = 10;
   double
alpha1,alpha2,alpha3,alpha4,Zamin,alpha5,Zsmin,
kp,T1,T2,T3,T4,gamma1,gamma2,tq1,tq2,At;
   static int gray[161],bcd[161];
   alpha1=0.0; alpha2=0.0;
alpha3=0.0;alpha4=0.0;Zamin=0.0;alpha5=0.0;Zs
min=0.0;kp=0.0;T1=0.0;T2=0.0;T3=0.0;T4=0.0;ga
mma1=0.0;gamma2=0.0;tq1=0.0;tq2=0.0;At=0.0;
   bitpow2=0; bitpow3=0;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
critter->fitness = 0.0;
/* loop thru number of bytes holding
chromosome */
for(k = 0; k <= chromsize-1; k++)
{
        if(k == (chromsize-1))
          {stop = lchrom-(k*INTSIZE);}
        else
          {stop = INTSIZE;}
        /* loop thru bits in current byte */
        tp = critter->chrom[k];
        for(j = 0; j < stop; j++)
        {
          bitpos = j + INTSIZE*k;
          /* test for current bit 0 or 1 */
          if((bitpos>=0)&&(bitpos<=9))
          {
           if((tp&mask) == 1)
           {
                  gray[(int)bitpos]=1;
           }
           else
           gray[(int)bitpos]=0;
          }
          if((bitpos>=10)&&(bitpos<=16))
          {
           if((tp&mask) == 1)
            { gray[(int)bitpos]=1;
            }
            else
           gray[(int)bitpos]=0;
          }
          if((bitpos>=17)&&(bitpos<=26))
          {
           if((tp&mask)==1)
            { gray[(int)bitpos]=1;
            }
            else
            gray[(int)bitpos]=0;
          }
         if((bitpos>=27)&&(bitpos<=36))
          {
          if((tp&mask) == 1)
          { gray[(int)bitpos]=1;
          }
          else
          gray[(int)bitpos]=0;
         }
        if((bitpos>=37)&&(bitpos<=46))
          {
          if((tp&mask) == 1)
          { gray[(int)bitpos]=1;
          }
          else
          gray[(int)bitpos]=0;
         }
        if((bitpos>=47)&&(bitpos<=54))
          {
          if((tp&mask) == 1)
          { gray[(int)bitpos]=1;
          }
          else
          gray[(int)bitpos]=0;
         }
        if((bitpos>=55)&&(bitpos<=64))
          {
          if((tp&mask) == 1)
          { gray[(int)bitpos]=1;
          }
          else
          gray[(int)bitpos]=0;
         }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
if((bitpos>=65)&&(bitpos<=71))
  {
   if((tp&mask) == 1)
   { gray[(int)bitpos]=1;
   }
   else
   gray[(int)bitpos]=0;
  }
if((bitpos>=72)&&(bitpos<=81))
  {
   if((tp&mask) == 1)
   { gray[(int)bitpos]=1;
   }
   else
   gray[(int)bitpos]=0;
  }
if((bitpos>=82)&&(bitpos<=91))
  {
   if((tp&mask) == 1)
   { gray[(int)bitpos]=1;
   }
   else
   gray[(int)bitpos]=0;
  }
if((bitpos>=92)&&(bitpos<=101))
  {
   if((tp&mask) == 1)
   { gray[(int)bitpos]=1;
   }
   else
   gray[(int)bitpos]=0;
  }
if((bitpos>=102)&&(bitpos<=111))
  {
   if((tp&mask) == 1)
   { gray[(int)bitpos]=1;
   }
   else
   gray[(int)bitpos]=0;
  }
if((bitpos>=112)&&(bitpos<=121))
  {
   if((tp&mask) == 1)
   { gray[(int)bitpos]=1;
   }
   else
   gray[(int)bitpos]=0;
  }
if((bitpos>=122)&&(bitpos<=131))
  {
   if((tp&mask) == 1)
   { gray[(int)bitpos]=1;
   }
   else
   gray[(int)bitpos]=0;
  }
if((bitpos>=132)&&(bitpos<=141))
  {
   if((tp&mask) == 1)
   { gray[(int)bitpos]=1;
   }
   else
   gray[(int)bitpos]=0;
  }
if((bitpos>=142)&&(bitpos<=151))
  {
   if((tp&mask) == 1)
   { gray[(int)bitpos]=1;
   }
   else
   gray[(int)bitpos]=0;
  }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        if((bitpos>=152)&&(bitpos<=160))
          {
           if((tp&mask) == 1)
           { gray[(int)bitpos]=1;
           }
           else
           gray[(int)bitpos]=0;
          }
          tp = tp>>=1;
        }

}
bcd[9]=gray[9];
for(i=8;i>=0;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[16]=gray[16];
for(i=15;i>=10;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[26]=gray[26];
for(i=25;i>=17;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[36]=gray[36];
for(i=35;i>=27;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[46]=gray[46];
for(i=45;i>=37;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[54]=gray[54];
for(i=53;i>=47;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[64]=gray[64];
for(i=63;i>=55;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[71]=gray[71];
for(i=70;i>=65;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[81]=gray[81];
for(i=80;i>=72;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[91]=gray[91];
for(i=90;i>=82;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[101]=gray[101];
for(i=100;i>=92;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[111]=gray[111];
for(i=110;i>=102;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[121]=gray[121];
for(i=120;i>=112;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[131]=gray[131];
for(i=130;i>=122;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[141]=gray[141];
for(i=140;i>=132;i--)
bcd[i]=bcd[i+1]^gray[i];
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
bcd[151]=gray[151];
for(i=150;i>=142;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[160]=gray[160];
for(i=159;i>=152;i--)
bcd[i]=bcd[i+1]^gray[i];


for(i=0;i<=9;i++)
{
alpha1 += bcd[i]*pow(2.0,(double)i);
}
for(i=10;i<=16;i++)
{
alpha2 += bcd[i]*pow(2.0,(double)(i-10));
}
for(i=17;i<=26;i++)
{
alpha3 += bcd[i]*pow(2.0,(double)(i-17));
}
for(i=27;i<=36;i++)
{
alpha4 += bcd[i]*pow(2.0,(double)(i-27));
}
for(i=37;i<=46;i++)
{
Zamin += bcd[i]*pow(2.0,(double)(i-37));
}
for(i=47;i<=54;i++)
{
alpha5 += bcd[i]*pow(2.0,(double)(i-47));
}
for(i=55;i<=64;i++)
{
Zsmin += bcd[i]*pow(2.0,(double)(i-55));
}
for(i=65;i<=71;i++)
{
kp += bcd[i]*pow(2.0,(double)(i-65));
}
for(i=72;i<=81;i++)
{
T1 += bcd[i]*pow(2.0,(double)(i-72));
}
for(i=82;i<=91;i++)
{
T2 += bcd[i]*pow(2.0,(double)(i-82));
}
for(i=92;i<=101;i++)
{
T3 += bcd[i]*pow(2.0,(double)(i-92));
}
for(i=102;i<=111;i++)
{
T4 += bcd[i]*pow(2.0,(double)(i-102));
}
for(i=112;i<=121;i++)
{
gamma1 += bcd[i]*pow(2.0,(double)(i-112));
}
for(i=122;i<=131;i++)
{
gamma2 += bcd[i]*pow(2.0,(double)(i-122));
}
for(i=132;i<=141;i++)
{
tq1 += bcd[i]*pow(2.0,(double)(i-132));
}
for(i=142;i<=151;i++)
{
tq2 += bcd[i]*pow(2.0,(double)(i-142));
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    }
    for(i=152;i<=160;i++)
    {
    At += bcd[i]*pow(2.0,(double)(i-152));
    }

    alpha1 =
(double)((int)(alpha1*1000/1023))/1000;
    alpha2 = (double)((int)(alpha2*90/127));
    alpha3 = (double)((int)(alpha3*100/127))/100;
    alpha4 =
(double)((int)(alpha4*1000/1023))/1000;
    Zamin =
(double)((int)(Zamin*1000/1023))/1000;
    alpha5 = (double)((int)(alpha5*200/255))/100;
    Zsmin =
(double)((int)(Zsmin*1000/1023))/1000;
    kp   = (double)((int)(kp*100/127))/100;
    T1   = (double)((int)(T1*1000/1023))/1000;
    T2   = (double)((int)(T2*1000/1023))/1000;
    T3   = (double)((int)(T3*1000/1023))/1000;
    T4   = (double)((int)(T4*1000/1023))/1000;
    gamma1 = 2.0+(double)((int)
(gamma1*1000/1023))/1000;
    gamma2 = 2.0+(double)((int)
(gamma2*1000/1023))/1000;
    tq1  = (double)((int)(tq1*1000/1023))/100;
    tq2  = (double)((int)(tq2*1000/1023))/100;
    At   = 1.0+(double)((int)(At*500/511))/1000;
    critter->fitness = (2500/(performance_index
(alpha1,alpha2,alpha3,alpha4,Zamin,alpha5,Zsmin
,kp,T1,T2,T3,T4,gamma1,gamma2,tq1,tq2,At)));
}
void decode_fitness(struct individual *critter1)
{
    unsigned mask=0x1;  /* mask for current bit */
    int bitpos;    /* current bit position */
    unsigned bitpos2=0,bitpos3=0;
    unsigned tp;
    double pow(), bitpow1=0,
coef,bitpow2=0,bitpow3=0;
    int i,j, k, stop;
    int n = 10;
    double
alpha1,alpha2,alpha3,alpha4,Zamin,alpha5,Zsmin,
kp,T1,T2,T3,T4,gamma1,gamma2,tq1,tq2,At;
    static int gray[161],bcd[161];
    alpha1=0.0; alpha2=0.0;
alpha3=0.0;alpha4=0.0;Zamin=0.0;alpha5=0.0;Zs
min=0.0;kp=0.0;T1=0.0;T2=0.0;T3=0.0;T4=0.0;ga
mma1=0.0;gamma2=0.0;tq1=0.0;tq2=0.0;At=0.0;
    bitpow2=0; bitpow3=0;
    critter1->fitness = 0.0;
    /* loop thru number of bytes holding
    chromosome */
    for(k = 0; k <= chromsize-1; k++)
    {
      if(k == (chromsize-1))
        {stop = lchrom-(k*INTSIZE);}
      else
        {stop = INTSIZE;}
      /* loop thru bits in current byte */
      tp = critter1->chrom[k];
      for(j = 0; j < stop; j++)
      {
        bitpos = j + INTSIZE*k;
        /* test for current bit 0 or 1 */
        if((bitpos>=0)&&(bitpos<=9))
        {
          if((tp&mask) == 1)
          {
            gray[(int)bitpos]=1;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    }
    else
    gray[(int)bitpos]=0;
   }
   if((bitpos>=10)&&(bitpos<=16))
   {
    if((tp&mask) == 1)
     { gray[(int)bitpos]=1;
     }
     else
    gray[(int)bitpos]=0;
   }
   if((bitpos>=17)&&(bitpos<=26))
   {
    if((tp&mask)==1)
     { gray[(int)bitpos]=1;
     }
    else
    gray[(int)bitpos]=0;
   }
  if((bitpos>=27)&&(bitpos<=36))
   {
    if((tp&mask) == 1)
    { gray[(int)bitpos]=1;
    }
    else
    gray[(int)bitpos]=0;
   }
if((bitpos>=37)&&(bitpos<=46))
   {
    if((tp&mask) == 1)
    { gray[(int)bitpos]=1;
    }
    else
    gray[(int)bitpos]=0;
   }
if((bitpos>=47)&&(bitpos<=54))
   {
    if((tp&mask) == 1)
    { gray[(int)bitpos]=1;
    }
    else
    gray[(int)bitpos]=0;
   }
if((bitpos>=55)&&(bitpos<=64))
   {
    if((tp&mask) == 1)
    { gray[(int)bitpos]=1;
    }
    else
    gray[(int)bitpos]=0;
   }
   if((bitpos>=65)&&(bitpos<=71))
   {
    if((tp&mask) == 1)
    { gray[(int)bitpos]=1;
    }
    else
    gray[(int)bitpos]=0;
   }
if((bitpos>=72)&&(bitpos<=81))
   {
    if((tp&mask) == 1)
    { gray[(int)bitpos]=1;
    }
    else
    gray[(int)bitpos]=0;
   }
if((bitpos>=82)&&(bitpos<=91))
   {
    if((tp&mask) == 1)
    { gray[(int)bitpos]=1;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    }
    else
    gray[(int)bitpos]=0;
   }
if((bitpos>=92)&&(bitpos<=101))
   {
    if((tp&mask) == 1)
    { gray[(int)bitpos]=1;
    }
    else
    gray[(int)bitpos]=0;
   }
if((bitpos>=102)&&(bitpos<=111))
   {
    if((tp&mask) == 1)
    { gray[(int)bitpos]=1;
    }
    else
    gray[(int)bitpos]=0;
   }
if((bitpos>=112)&&(bitpos<=121))
   {
    if((tp&mask) == 1)
    { gray[(int)bitpos]=1;
    }
    else
    gray[(int)bitpos]=0;
   }
if((bitpos>=122)&&(bitpos<=131))
   {
    if((tp&mask) == 1)
    { gray[(int)bitpos]=1;
    }
    else
    gray[(int)bitpos]=0;
   }
if((bitpos>=132)&&(bitpos<=141))
   {
    if((tp&mask) == 1)
    { gray[(int)bitpos]=1;
    }
    else
    gray[(int)bitpos]=0;
   }
if((bitpos>=142)&&(bitpos<=151))
   {
    if((tp&mask) == 1)
    { gray[(int)bitpos]=1;
    }
    else
    gray[(int)bitpos]=0;
   }
if((bitpos>=152)&&(bitpos<=160))
   {
    if((tp&mask) == 1)
    { gray[(int)bitpos]=1;
    }
    else
    gray[(int)bitpos]=0;
   }
   tp = tp>>=1;
  }
 }
 bcd[9]=gray[9];
 for(i=8;i>=0;i--)
 bcd[i]=bcd[i+1]^gray[i];

 bcd[16]=gray[16];
 for(i=15;i>=10;i--)
 bcd[i]=bcd[i+1]^gray[i];
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
bcd[26]=gray[26];
for(i=25;i>=17;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[36]=gray[36];
for(i=35;i>=27;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[46]=gray[46];
for(i=45;i>=37;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[54]=gray[54];
for(i=53;i>=47;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[64]=gray[64];
for(i=63;i>=55;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[71]=gray[71];
for(i=70;i>=65;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[81]=gray[81];
for(i=80;i>=72;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[91]=gray[91];
for(i=90;i>=82;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[101]=gray[101];
for(i=100;i>=92;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[111]=gray[111];
for(i=110;i>=102;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[121]=gray[121];
for(i=120;i>=112;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[131]=gray[131];
for(i=130;i>=122;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[141]=gray[141];
for(i=140;i>=132;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[151]=gray[151];
for(i=150;i>=142;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[160]=gray[160];
for(i=159;i>=152;i--)
bcd[i]=bcd[i+1]^gray[i];

for(i=0;i<=9;i++)
{
alpha1 += bcd[i]*pow(2.0,(double)i);
}
for(i=10;i<=16;i++)
{
alpha2 += bcd[i]*pow(2.0,(double)(i-10));
}
for(i=17;i<=26;i++)
{
alpha3 += bcd[i]*pow(2.0,(double)(i-17));
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
}
for(i=27;i<=36;i++)
{
alpha4 += bcd[i]*pow(2.0,(double)(i-27));
}
for(i=37;i<=46;i++)
{
Zamin += bcd[i]*pow(2.0,(double)(i-37));
}
for(i=47;i<=54;i++)
{
alpha5 += bcd[i]*pow(2.0,(double)(i-47));
}
for(i=55;i<=64;i++)
{
Zsmin += bcd[i]*pow(2.0,(double)(i-55));
}
for(i=65;i<=71;i++)
{
kp += bcd[i]*pow(2.0,(double)(i-65));
}
for(i=72;i<=81;i++)
{
T1 += bcd[i]*pow(2.0,(double)(i-72));
}
for(i=82;i<=91;i++)
{
T2 += bcd[i]*pow(2.0,(double)(i-82));
}
 for(i=92;i<=101;i++)
{
T3 += bcd[i]*pow(2.0,(double)(i-92));
}
for(i=102;i<=111;i++)
{
T4 += bcd[i]*pow(2.0,(double)(i-102));
}
 for(i=112;i<=121;i++)
{
gamma1 += bcd[i]*pow(2.0,(double)(i-112));
}
for(i=122;i<=131;i++)
{
gamma2 += bcd[i]*pow(2.0,(double)(i-122));
}
for(i=132;i<=141;i++)
{
tq1 += bcd[i]*pow(2.0,(double)(i-132));
}
 for(i=142;i<=151;i++)
 {
 tq2 += bcd[i]*pow(2.0,(double)(i-142));
 }
 for(i=152;i<=160;i++)
 {
 At += bcd[i]*pow(2.0,(double)(i-152));
 }
 alpha1 =
(double)((int)(alpha1*1000/1023))/1000;
 alpha2 = (double)((int)(alpha2*90/127));
 alpha3 = (double)((int)(alpha3*100/127))/100;
 alpha4 =
(double)((int)(alpha4*1000/1023))/1000;
 Zamin =
(double)((int)(Zamin*1000/1023))/1000;
 alpha5 = (double)((int)(alpha5*200/255))/100;
 Zsmin =
(double)((int)(Zsmin*1000/1023))/1000;
 kp = (double)((int)(kp*100/127))/100;
 T1 = (double)((int)(T1*1000/1023))/1000;
 T2 = (double)((int)(T2*1000/1023))/1000;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    T3    = (double)((int)(T3*1000/1023))/1000;
    T4    = (double)((int)(T4*1000/1023))/1000;
    gamma1 =  2.0+(double)((int)
(gamma1*1000/1023))/1000;
    gamma2 =  2.0+(double)((int)
(gamma2*1000/1023))/1000;
    tq1   = (double)((int)(tq1*1000/1023))/100;
    tq2   = (double)((int)(tq2*1000/1023))/100;
    At    = 1.0+(double)((int)(At*500/511))/1000;
    critter1->fitness = (2500/(performance_index
(alpha1,alpha2,alpha3,alpha4,Zamin,alpha5,Zsmin
,kp,T1,T2,T3,T4,gamma1,gamma2,tq1,tq2,At)));
    fprintf(outfp,"alpha1 = %f  alpha2 = %f
alpha3 = %f  alpha4 = %f  \nZamin = %f  alpha5 =
%f Zsmin= %f  kp = %f  T1 = %f  T2 = %f  T3 =
%f  T4 = %f  \ngamma1 = %f  gamma2 = %f  Tr1
= %f  Tr2 = %f\n",alpha1,alpha2,alpha3,alpha4

,Zamin,alpha5,Zsmin,kp,T1,T2,T3,T4,gamma1,ga
mma2,tq1,tq2);
}
  /* alpha1 = alpha1/255;
    alpha2 = (alpha2*90)/127;
    alpha3 = alpha3/127;
    */
    /*for(i=23;i>=0;i--)
    printf("%d",gray[i]); */

void print_fitness(struct individual *critter1)
{
    unsigned mask=0x1;  /* mask for current bit */
    int bitpos;    /* current bit position */
    unsigned bitpos2=0,bitpos3=0;
    unsigned tp;
    double pow(), bitpow1=0,
coef,bitpow2=0,bitpow3=0;
    int i,j, k, stop;
    int n = 10;
    double
alpha1,alpha2,alpha3,alpha4,Zamin,alpha5,Zsmin,
kp,T1,T2,T3,T4,gamma1,gamma2,tq1,tq2,At;
    static int gray[161],bcd[161];
    alpha1=0.0; alpha2=0.0;
alpha3=0.0;alpha4=0.0;Zamin=0.0;alpha5=0.0;Zs
min=0.0;kp=0.0;T1=0.0;T2=0.0;T3=0.0;T4=0.0;ga
mma1=0.0;gamma2=0.0;tq1=0.0;tq2=0.0;At=0.0;
    bitpow2=0; bitpow3=0;
    critter1->fitness = 0.0;
    /* loop thru number of bytes holding
chromosome */
    for(k = 0; k <= chromsize-1; k++)
    {
      if(k == (chromsize-1))
        {stop = lchrom-(k*INTSIZE);}
      else
        {stop = INTSIZE;}
      /* loop thru bits in current byte */
      tp = critter1->chrom[k];
      for(j = 0; j < stop; j++)
      {
       bitpos = j + INTSIZE*k;
       /* test for current bit 0 or 1 */
       if((bitpos>=0)&&(bitpos<=9))
       {
        if((tp&mask) == 1)
        {
              gray[(int)bitpos]=1;
        }
        else
        gray[(int)bitpos]=0;
       }
       if((bitpos>=10)&&(bitpos<=16))
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    {
     if((tp&mask) == 1)
      { gray[(int)bitpos]=1;
      }
      else
     gray[(int)bitpos]=0;
    }
   if((bitpos>=17)&&(bitpos<=26))
    {
     if((tp&mask)==1)
      { gray[(int)bitpos]=1;
      }
      else
     gray[(int)bitpos]=0;
    }
  if((bitpos>=27)&&(bitpos<=36))
    {
     if((tp&mask) == 1)
      { gray[(int)bitpos]=1;
      }
      else
     gray[(int)bitpos]=0;
    }
if((bitpos>=37)&&(bitpos<=46))
    {
     if((tp&mask) == 1)
      { gray[(int)bitpos]=1;
      }
      else
     gray[(int)bitpos]=0;
    }
if((bitpos>=47)&&(bitpos<=54))
    {
     if((tp&mask) == 1)
      { gray[(int)bitpos]=1;
      }
      else
     gray[(int)bitpos]=0;
    }
if((bitpos>=55)&&(bitpos<=64))
    {
     if((tp&mask) == 1)
      { gray[(int)bitpos]=1;
      }
      else
     gray[(int)bitpos]=0;
    }
if((bitpos>=65)&&(bitpos<=71))
    {
     if((tp&mask) == 1)
     { gray[(int)bitpos]=1;
     }
     else
     gray[(int)bitpos]=0;
    }
if((bitpos>=72)&&(bitpos<=81))
    {
     if((tp&mask) == 1)
     { gray[(int)bitpos]=1;
     }
     else
     gray[(int)bitpos]=0;
    }
if((bitpos>=82)&&(bitpos<=91))
    {
     if((tp&mask) == 1)
      { gray[(int)bitpos]=1;
      }
      else
     gray[(int)bitpos]=0;
    }
if((bitpos>=92)&&(bitpos<=101))
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    {
     if((tp&mask) == 1)
     { gray[(int)bitpos]=1;
     }
     else
     gray[(int)bitpos]=0;
    }
if((bitpos>=102)&&(bitpos<=111))
    {
     if((tp&mask) == 1)
     { gray[(int)bitpos]=1;
     }
     else
     gray[(int)bitpos]=0;
    }
if((bitpos>=112)&&(bitpos<=121))
    {
     if((tp&mask) == 1)
     { gray[(int)bitpos]=1;
     }
     else
     gray[(int)bitpos]=0;
    }
if((bitpos>=122)&&(bitpos<=131))
    {
     if((tp&mask) == 1)
     { gray[(int)bitpos]=1;
     }
     else
     gray[(int)bitpos]=0;
    }
if((bitpos>=132)&&(bitpos<=141))
    {
     if((tp&mask) == 1)
     { gray[(int)bitpos]=1;
     }
     else
     gray[(int)bitpos]=0;
    }
if((bitpos>=142)&&(bitpos<=151))
    {
     if((tp&mask) == 1)
     { gray[(int)bitpos]=1;
     }
     else
     gray[(int)bitpos]=0;
    }
if((bitpos>=152)&&(bitpos<=160))
    {
     if((tp&mask) == 1)
     { gray[(int)bitpos]=1;
     }
     else
     gray[(int)bitpos]=0;
     tp = tp>>=1;
    }
   }
   bcd[9]=gray[9];
   for(i=8;i>=0;i--)
   bcd[i]=bcd[i+1]^gray[i];

   bcd[16]=gray[16];
   for(i=15;i>=10;i--)
   bcd[i]=bcd[i+1]^gray[i];

   bcd[26]=gray[26];
   for(i=25;i>=17;i--)
   bcd[i]=bcd[i+1]^gray[i];

   bcd[36]=gray[36];
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
for(i=35;i>=27;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[46]=gray[46];
for(i=45;i>=37;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[54]=gray[54];
for(i=53;i>=47;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[64]=gray[64];
for(i=63;i>=55;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[71]=gray[71];
for(i=70;i>=65;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[81]=gray[81];
for(i=80;i>=72;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[91]=gray[91];
for(i=90;i>=82;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[101]=gray[101];
for(i=100;i>=92;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[111]=gray[111];
for(i=110;i>=102;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[121]=gray[121];
for(i=120;i>=112;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[131]=gray[131];
for(i=130;i>=122;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[141]=gray[141];
for(i=140;i>=132;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[151]=gray[151];
for(i=150;i>=142;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[160]=gray[160];
for(i=159;i>=152;i--)
bcd[i]=bcd[i+1]^gray[i];

for(i=0;i<=9;i++)
{
alpha1 += bcd[i]*pow(2.0,(double)i);
}
for(i=10;i<=16;i++)
{
alpha2 += bcd[i]*pow(2.0,(double)(i-10));
}
for(i=17;i<=26;i++)
{
alpha3 += bcd[i]*pow(2.0,(double)(i-17));
}
for(i=27;i<=36;i++)
{
alpha4 += bcd[i]*pow(2.0,(double)(i-27));
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
for(i=37;i<=46;i++)
{
Zamin += bcd[i]*pow(2.0,(double)(i-37));
}
for(i=47;i<=54;i++)
{
alpha5 += bcd[i]*pow(2.0,(double)(i-47));
}
for(i=55;i<=64;i++)
{
Zsmin += bcd[i]*pow(2.0,(double)(i-55));
}
for(i=65;i<=71;i++)
{
kp += bcd[i]*pow(2.0,(double)(i-65));
}
for(i=72;i<=81;i++)
{
T1 += bcd[i]*pow(2.0,(double)(i-72));
}
for(i=82;i<=91;i++)
{
T2 += bcd[i]*pow(2.0,(double)(i-82));
}
for(i=92;i<=101;i++)
{
T3 += bcd[i]*pow(2.0,(double)(i-92));
}
for(i=102;i<=111;i++)
{
T4 += bcd[i]*pow(2.0,(double)(i-102));
}
for(i=112;i<=121;i++)
{
gamma1 += bcd[i]*pow(2.0,(double)(i-112));
}
for(i=122;i<=131;i++)
{
gamma2 += bcd[i]*pow(2.0,(double)(i-122));
}
for(i=132;i<=141;i++)
{
tq1 += bcd[i]*pow(2.0,(double)(i-132));
}
for(i=142;i<=151;i++)
{
tq2 += bcd[i]*pow(2.0,(double)(i-142));
}
for(i=152;i<=160;i++)
{
At += bcd[i]*pow(2.0,(double)(i-152));
}
alpha1 =
(double)((int)(alpha1*1000/1023))/1000;
alpha2 = (double)((int)(alpha2*90/127));
alpha3 = (double)((int)(alpha3*100/127))/100;
alpha4 =
(double)((int)(alpha4*1000/1023))/1000;
Zamin =
(double)((int)(Zamin*1000/1023))/1000;
alpha5 = (double)((int)(alpha5*200/255))/100;
Zsmin =
(double)((int)(Zsmin*1000/1023))/1000;
kp   = (double)((int)(kp*100/127))/100;
T1   = (double)((int)(T1*1000/1023))/1000;
T2   = (double)((int)(T2*1000/1023))/1000;
T3   = (double)((int)(T3*1000/1023))/1000;
T4   = (double)((int)(T4*1000/1023))/1000;
gamma1 = 2.0+(double)((int)
(gamma1*1000/1023))/1000;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    gamma2 =  2.0+(double)((int)
(gamma2*1000/1023))/1000;
    tq1    =  (double)((int)(tq1*1000/1023))/100;
    tq2    =  (double)((int)(tq2*1000/1023))/100;
    At     =  1.0+(double)((int)(At*500/511))/1000;
    critter1->fitness = (2500/(performance_index
(alpha1,alpha2,alpha3,alpha4,Zamin,alpha5,Zsmin
,kp,T1,T2,T3,T4,gamma1,gamma2,tq1,tq2,At)));
    fprintf(outfp,"alpha1 = %f  alpha2 = %f
alpha3 = %f  alpha4 = %f \nZamin = %f  alpha5 =
%f Zsmin= %f  kp = %f  \nT1 = %f  T2 = %f  T3
= %f  T4 = %f  \ngamma1 = %f  gamma2 = %f
Tr1 = %f  Tr2 =
%f\n",alpha1,alpha2,alpha3,alpha4

,Zamin,alpha5,Zsmin,kp,T1,T2,T3,T4,gamma1,ga
mma2,tq1,tq2);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

****

****

ถ้าต้องการที่จะใช้โปรแกรมในการออกแบบตัวควบคุม ΔP – PSS และ $\Delta\omega$ - PSS สามารถแทนฟังก์ชัน fitness,print_fitness และ decode_fitness ได้โดยฟังก์ชันดังต่อไปนี้คือ fitness1, print_fitness1 และ decode_fitness1 ตามลำดับ โดยให้ประกาศและเรียกใช้งานแทนฟังก์ชันดังกล่าว ฟังก์ชัน fitness1, print_fitness1 และ decode_fitness1 มีดังนี้

```
void fitness1(struct individual *critter)
{
   unsigned mask=0x1;  /* mask for current bit */
   int bitpos;    /* current bit position */

   unsigned tp;

   int i,j, k, stop;
   int n = 10;
   double kpss,t1,t2,t3,t4;
   static int gray[25],bcd[25];

   kpss =0.0;
   t1=0.0;
   t2=0.0;
   t3=0.0;
   t4=0.0;
   critter->fitness = 0.0;
   /* loop thru number of bytes holding
chromosome */
   for(k = 0; k <= chromsize-1; k++)
   {
      if(k == (chromsize-1))
         {stop = lchrom-(k*INTSIZE);}
      else
         {stop = INTSIZE;}
      /* loop thru bits in current byte */
      tp = critter->chrom[k];
      for(j = 0; j < stop; j++)
      {
         bitpos = j + INTSIZE*k;
         /* test for current bit 0 or 1 */
         if((bitpos>=0)&&(bitpos<=3))
         {
          if((tp&mask) == 1)
          {
                gray[(int)bitpos]=1;
                /*alpha3 += pow(2.0,(double)
bitpos);
                /*else G[bitpos]=0; */
          }
          else
          gray[(int)bitpos]=0;
          }
         if((bitpos>=4)&&(bitpos<=7))
          {
          if((tp&mask) == 1)
          { gray[(int)bitpos]=1;
                /*alpha2 +=pow(2.0,(double)
(bitpos-7));
          /*else G[bitpos]=0; */
          }
          else
          gray[(int)bitpos]=0;
         }
         if((bitpos>=8)&&(bitpos<=11))
         {
          if((tp&mask)==1)
          { gray[(int)bitpos]=1;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                /*alpha1 +=pow(2.0,(double)
(bitpos-14));
                /* else G[bitpos]=0;*/
                }
                else
                gray[(int)bitpos]=0;
              }
              /* for PSS2 */
              if((bitpos>=12)&&(bitpos<=15))  /*
Kpss*/
              {
               if((tp&mask) == 1)
               {
                     gray[(int)bitpos]=1;
                     /*alpha3 += pow(2.0,(double)
bitpos);

                     /*else G[bitpos]=0; */
               }
               else
               gray[(int)bitpos]=0;
              }
              if((bitpos>=16)&&(bitpos<=24))
              {
               if((tp&mask) == 1)
                { gray[(int)bitpos]=1;
                     /*alpha2 +=pow(2.0,(double)
(bitpos-7));
                /*else G[bitpos]=0; */
                }
                else
                gray[(int)bitpos]=0;
              }
              tp = tp>>=1;
            }
            }
            bcd[3]=gray[3];
            for(i=2;i>=0;i--)
            bcd[i]=bcd[i+1]^gray[i];

            bcd[7]=gray[7];
            for(i=6;i>=4;i--)
            bcd[i]=bcd[i+1]^gray[i];

            bcd[11]=gray[11];
            for(i=10;i>=8;i--)
            bcd[i]=bcd[i+1]^gray[i];

            bcd[15]=gray[15];
            for(i=14;i>=12;i--)
            bcd[i]=bcd[i+1]^gray[i];

            bcd[24]=gray[24];
            for(i=23;i>=16;i--)
            bcd[i]=bcd[i+1]^gray[i];

            for(i=0;i<=3;i++)
            {
            t4 += bcd[i]*pow(2.0,(double)i);
            }
            for(i=4;i<=7;i++)
            {
            t3 += bcd[i]*pow(2.0,(double)(i-4));
            }
            for(i=8;i<=11;i++)
            {
            t2 += bcd[i]*pow(2.0,(double)(i-8));
            }
            for(i=12;i<=15;i++)
            {
            t1 += bcd[i]*pow(2.0,(double)(i-12));
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    }
    for(i=16;i<=24;i++)
    {
    kpss += bcd[i]*pow(2.0,(double)(i-16));
    }

    kpss  = 0.1 + (double)((int)(kpss*499/511))/10;
    t1    = 0.2 + (double)((int)(t1*13/15))/10;
    t2    = 0.02 + (double)((int)(t2*13/15))/100;
    t3    = 0.2 +  (double)((int)(t3*13/15))/10;
    t4    = 0.02 + (double)((int)(t4*13/15))/100;
    critter->fitness = (2500/(performance_index
(kpss,t1,t2,t3,t4)));
}


void decode_fitness1(struct individual *critter1)
{
    unsigned mask=0x1;  /* mask for current bit */
    int bitpos;    /* current bit position */
    unsigned bitpos2=0,bitpos3=0;
    unsigned tp;
    double pow(), bitpow1=0,
coef,bitpow2=0,bitpow3=0;
    int i,j, k, stop;
    int n = 10;
    double kpss,t1,t2,t3,t4;
    static int gray[25],bcd[25];

    kpss =0.0;
    t1=0.0;
    t2=0.0;
    t3=0.0;
    t4=0.0;
    critter1->fitness = 0.0;
    /* loop thru number of bytes holding
chromosome */
    for(k = 0; k <= chromsize-1; k++)
    {
        if(k == (chromsize-1))
            {stop = lchrom-(k*INTSIZE);}
        else
            {stop = INTSIZE;}
        /* loop thru bits in current byte */
        tp = critter1->chrom[k];
        for(j = 0; j < stop; j++)
        {
            bitpos = j + INTSIZE*k;
            /* test for current bit 0 or 1 */
            if((bitpos>=0)&&(bitpos<=3))
            {
             if((tp&mask) == 1)
             {
                 gray[(int)bitpos]=1;
                 /*alpha3 += pow(2.0,(double)
bitpos);

                 /*else G[bitpos]=0; */
             }
              else
              gray[(int)bitpos]=0;
             }
             if((bitpos>=4)&&(bitpos<=7))
             {
              if((tp&mask) == 1)
               { gray[(int)bitpos]=1;
                    /*alpha2 +=pow(2.0,(double)
(bitpos-7));

                 /*else G[bitpos]=0; */
                 }
                 else
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 gray[(int)bitpos]=0;
}
if((bitpos>=8)&&(bitpos<=11))
{
 if((tp&mask)==1)
  { gray[(int)bitpos]=1;
        /*alpha1 +=pow(2.0,(double)
(bitpos-14));
 /* else G[bitpos]=0;*/
 }
 else
 gray[(int)bitpos]=0;
}
/* for PSS2 */
if((bitpos>=12)&&(bitpos<=15)) /*
Kpss*/
{
 if((tp&mask) == 1)
 {
        gray[(int)bitpos]=1;
        /*alpha3 += pow(2.0,(double)
bitpos);

        /*else G[bitpos]=0; */
 }
 else
 gray[(int)bitpos]=0;
}
if((bitpos>=16)&&(bitpos<=24))
{
 if((tp&mask) == 1)
  { gray[(int)bitpos]=1;
        /*alpha2 +=pow(2.0,(double)
(bitpos-7));
    /*else G[bitpos]=0; */
    }
 else
 gray[(int)bitpos]=0;
 }
 tp = tp>>=1;
 }
}
bcd[3]=gray[3];
for(i=2;i>=0;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[7]=gray[7];
for(i=6;i>=4;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[11]=gray[11];
for(i=10;i>=8;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[15]=gray[15];
for(i=14;i>=12;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[24]=gray[24];
for(i=23;i>=16;i--)
bcd[i]=bcd[i+1]^gray[i];

for(i=0;i<=3;i++)
{
t4 += bcd[i]*pow(2.0,(double)i);
}
for(i=4;i<=7;i++)
{
t3 += bcd[i]*pow(2.0,(double)(i-4));
}
for(i=8;i<=11;i++)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    {
    t2 += bcd[i]*pow(2.0,(double)(i-8));
    }
    for(i=12;i<=15;i++)
    {
    t1 += bcd[i]*pow(2.0,(double)(i-12));
    }
    for(i=16;i<=24;i++)
    {
    kpss += bcd[i]*pow(2.0,(double)(i-16));
    }

    kpss  = 0.1 + (double)((int)(kpss*499/511))/10;
    t1    = 0.2 + (double)((int)(t1*13/15))/10;
    t2    = 0.02 + (double)((int)(t2*13/15))/100;
    t3    = 0.2 +  (double)((int)(t3*13/15))/10;
    t4    = 0.02 + (double)((int)(t4*13/15))/100;

    critter1->fitness = (2500/(performance_index
(kpss,t1,t2,t3,t4)));
    fprintf(outfp,"Kwpss = %f  T1 = %f  T2 = %f
T3 = %f  T4 = %f \n",kpss,t1,t2,t3,t4);
}

void print_fitness1(struct individual *critter1)
{
    unsigned mask=0x1;  /* mask for current bit */
    int bitpos;    /* current bit position */
    unsigned bitpos2=0,bitpos3=0;
    unsigned tp;
    double pow(), bitpow1=0,
coef,bitpow2=0,bitpow3=0;
    int i,j, k, stop;
    int n = 10;
    double kpss,t1,t2,t3,t4;
    static int gray[25],bcd[25];
    kpss =0.0;
    t1=0.0;
    t2=0.0;
    t3=0.0;
    t4=0.0;
    critter1->fitness = 0.0;
    /* loop thru number of bytes holding
chromosome */
    for(k = 0; k <= chromsize-1; k++)
    {
      if(k == (chromsize-1))
        {stop = lchrom-(k*INTSIZE);}
      else
        {stop = INTSIZE;}
      /* loop thru bits in current byte */
      tp = critter1->chrom[k];
      for(j = 0; j < stop; j++)
      {
        bitpos = j + INTSIZE*k;
        /* test for current bit 0 or 1 */
        if((bitpos>=0)&&(bitpos<=3))
        {
          if((tp&mask) == 1)
          {
            gray[(int)bitpos]=1;
            /*alpha3 += pow(2.0,(double)
bitpos);

            /*else G[bitpos]=0; */
          }
          else
          gray[(int)bitpos]=0;
        }
        if((bitpos>=4)&&(bitpos<=7))
        {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
            if((tp&mask) == 1)
             { gray[(int)bitpos]=1;
                  /*alpha2 +=pow(2.0,(double)
(bitpos-7));
             /*else G[bitpos]=0; */
             }
             else
             gray[(int)bitpos]=0;
            }
            if((bitpos>=8)&&(bitpos<=11))
            {
             if((tp&mask)==1)
              { gray[(int)bitpos]=1;
                  /*alpha1 +=pow(2.0,(double)
(bitpos-14));
             /* else G[bitpos]=0;*/
             }
             else
             gray[(int)bitpos]=0;
            }
            /* for PSS2 */
            if((bitpos>=12)&&(bitpos<=15)) /*
Kpss*/
            {
             if((tp&mask) == 1)
             {
                  gray[(int)bitpos]=1;
                  /*alpha3 += pow(2.0,(double)
bitpos);

                  /*else G[bitpos]=0; */
             }
             else
             gray[(int)bitpos]=0;
            }
            if((bitpos>=16)&&(bitpos<=24))
            {
             if((tp&mask) == 1)
              { gray[(int)bitpos]=1;
                  /*alpha2 +=pow(2.0,(double)
(bitpos-7));
             /*else G[bitpos]=0; */
             }
             else
             gray[(int)bitpos]=0;
            }
            tp = tp>>=1;
           }

           }
           bcd[3]=gray[3];
           for(i=2;i>=0;i--)
           bcd[i]=bcd[i+1]^gray[i];

           bcd[7]=gray[7];
           for(i=6;i>=4;i--)
           bcd[i]=bcd[i+1]^gray[i];

           bcd[11]=gray[11];
           for(i=10;i>=8;i--)
           bcd[i]=bcd[i+1]^gray[i];

           bcd[15]=gray[15];
           for(i=14;i>=12;i--)
           bcd[i]=bcd[i+1]^gray[i];

           bcd[24]=gray[24];
           for(i=23;i>=16;i--)
           bcd[i]=bcd[i+1]^gray[i];

           for(i=0;i<=3;i++)
           {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    t4 += bcd[i]*pow(2.0,(double)i);
    }
    for(i=4;i<=7;i++)
    {
    t3 += bcd[i]*pow(2.0,(double)(i-4));
    }
    for(i=8;i<=11;i++)
    {
    t2 += bcd[i]*pow(2.0,(double)(i-8));
    }
    for(i=12;i<=15;i++)
    {
    t1 += bcd[i]*pow(2.0,(double)(i-12));
    }
    for(i=16;i<=24;i++)
    {
    kpss += bcd[i]*pow(2.0,(double)(i-16));
    }

    kpss  = 0.1 + (double)((int)(kpss*499/511))/10;
    t1    = 0.2 + (double)((int)(t1*13/15))/10;
    t2    = 0.02 + (double)((int)(t2*13/15))/100;
    t3    = 0.2 +  (double)((int)(t3*13/15))/10;
    t4    = 0.02 + (double)((int)(t4*13/15))/100;

    critter1->fitness = (2500/(performance_index
(kpss,t1,t2,t3,t4)));
    fprintf(outfp,"Kwpss = %f  T1 = %f  T2 = %f
T3 = %f  T4 = %f \n",kpss,t1,t2,t3,t4);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

****

****

ถ้าต้องการที่จะใช้โปรแกรมในการออกแบบตัวควบคุม $\Delta\omega$ - PSS และ SVC สามารถแทนฟังก์ชัน fitness,print_fitness และ decode_fitness ได้โดยฟังก์ชันดังต่อไปนี้คือ fitness2, print_fitness2 และ decode_fitness2 ตามลำดับ โดยให้ประกาศและเรียกใช้งานแทนฟังก์ชันดังกล่าว ฟังก์ชัน fitness2, print_fitness2 และ decode_fitness2 มีดังนี้

```
void fitness2(struct individual *critter)
{
   unsigned mask=0x1;  /* mask for current bit */
   int bitpos;    /* current bit position */

   unsigned tp;
   int i,j, k, stop;
   int n = 10;
   double alpha1,alpha2,alpha3;
   double kpss=0.0,t1=0.0,t2=0.0,t3=0.0,t4=0.0;
   static int gray[50],bcd[50];

   alpha1=0.0;alpha2=0.0;alpha3=0.0;kpss
=0.0;t1=0.0;t2=0.0;t3=0.0;t4=0.0;
   critter->fitness = 0.0;

   for(k = 0; k <= chromsize-1; k++)
   {
        if(k == (chromsize-1))
           {stop = lchrom-(k*INTSIZE);}
        else
           {stop = INTSIZE;}
        /* loop bits in current byte */
tp = critter->chrom[k];
for(j = 0; j < stop; j++)
{
   bitpos = j + INTSIZE*k;
   /* test for current bit 0 or 1 */
   if((bitpos>=0)&&(bitpos<=6))
   {
    if((tp&mask) == 1)
    {
          gray[(int)bitpos]=1;
    }
    else
    gray[(int)bitpos]=0;
   }
   if((bitpos>=7)&&(bitpos<=14))
   {
    if((tp&mask) == 1)
    {
          gray[(int)bitpos]=1;
    }
    else
    gray[(int)bitpos]=0;
   }
   if((bitpos>=15)&&(bitpos<=24))
   {
    if((tp&mask)==1)
    {
          gray[(int)bitpos]=1;
    }
    else
    gray[(int)bitpos]=0;
   }
   /* for W-PSS */
   if((bitpos>=25)&&(bitpos<=33))  /*
                                        Kpss*/
   {
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
if((tp&mask) == 1)
{
        gray[(int)bitpos]=1;
}
else
gray[(int)bitpos]=0;
}
if((bitpos>=34)&&(bitpos<=37))
{
if((tp&mask) == 1)
{
        gray[(int)bitpos]=1;
}
 else
gray[(int)bitpos]=0;
}
if((bitpos>=38)&&(bitpos<=41))
{
if((tp&mask)==1)
{
        gray[(int)bitpos]=1;
}
else
gray[(int)bitpos]=0;
}
if((bitpos>=42)&&(bitpos<=45))
{
if((tp&mask) == 1)
{
        gray[(int)bitpos]=1;
}
 else
gray[(int)bitpos]=0;
}
if((bitpos>=46)&&(bitpos<=49))
{
if((tp&mask)==1)
{
        gray[(int)bitpos]=1;
}
else
gray[(int)bitpos]=0;
}
tp = tp>>=1;
}
}
bcd[6]=gray[6];
for(i=5;i>=0;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[14]=gray[14];
for(i=13;i>=7;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[24]=gray[24];
for(i=23;i>=15;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[33]=gray[33];
for(i=32;i>=25;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[37]=gray[37];
for(i=36;i>=34;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[41]=gray[41];
for(i=40;i>=38;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[45]=gray[45];
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
for(i=44;i>=42;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[49]=gray[49];
for(i=48;i>=46;i--)
bcd[i]=bcd[i+1]^gray[i];



for(i=0;i<=6;i++)
{
alpha3 += bcd[i]*pow(2.0,(double)i);
}
for(i=7;i<=14;i++)
{
alpha2 += bcd[i]*pow(2.0,(double)(i-7));
}
for(i=15;i<=24;i++)
{
alpha1 += bcd[i]*pow(2.0,(double)(i-15));
}
for(i=25;i<=33;i++)
{
kpss += bcd[i]*pow(2.0,(double)(i-25));
}
for(i=34;i<=37;i++)
{
t1 += bcd[i]*pow(2.0,(double)(i-34));
}
for(i=38;i<=41;i++)
{
t2 += bcd[i]*pow(2.0,(double)(i-38));
}
for(i=42;i<=45;i++)
{
t3 += bcd[i]*pow(2.0,(double)(i-42));
}
for(i=46;i<=49;i++)
{
t4 += bcd[i]*pow(2.0,(double)(i-46));
}

alpha1 =
(double)((int)(alpha1*1000/1023))/1000;
alpha2 = (double)((int)(alpha2*180/255));
alpha3 = (double)((int)(alpha3*100/127))/100;
kpss  = 0.1 + (double)((int)(kpss*499/511))/10;
t1    = 0.2 + (double)((int)(t1*13/15))/10;
t2    = 0.02 + (double)((int)(t2*13/15))/100;
t3    = 0.2 + (double)((int)(t3*13/15))/10;
t4    = 0.02 + (double)((int)(t4*13/15))/100;

critter->fitness = (2500/(performance_index
(alpha1,alpha2,alpha3,
                    kpss,t1,t2,t3,t4)));
}


void decode_fitness2(struct individual *critter1)
{
unsigned mask=0x1;  /* mask for current bit */
int bitpos;         /* current bit position */
unsigned bitpos2=0,bitpos3=0;
unsigned tp;
double pow(), bitpow1=0,
coef,bitpow2=0,bitpow3=0;
int i,j, k, stop;
int n = 10;
double alpha1,alpha2,alpha3;
double kpss,t1,t2,t3,t4;
static int gray[50],bcd[50];
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
alpha1=0.0;alpha2=0.0;alpha3=0.0;kpss
=0.0;t1=0.0;t2=0.0;t3=0.0;t4=0.0;
bitpow2=0; bitpow3=0;
critter1->fitness = 0.0;
/* loop number of bytes holding chromosome */
for(k = 0; k <= chromsize-1; k++)
{
    if(k == (chromsize-1))
      {stop = lchrom-(k*INTSIZE);}
    else
      {stop = INTSIZE;}
    /* loop bits in current byte */
    tp = critter1->chrom[k];
    for(j = 0; j < stop; j++)
    {
      bitpos = j + INTSIZE*k;
      /* test for current bit 0 or 1 */
      if((bitpos>=0)&&(bitpos<=6))
      {
       if((tp&mask) == 1)
       {
            gray[(int)bitpos]=1;
       }
       else
       gray[(int)bitpos]=0;
      }
      if((bitpos>=7)&&(bitpos<=14))
      {
       if((tp&mask) == 1)
       {
            gray[(int)bitpos]=1;
       }
       else
       gray[(int)bitpos]=0;
      }
      if((bitpos>=15)&&(bitpos<=24))
      {
       if((tp&mask)==1)
       {
            gray[(int)bitpos]=1;
       }
       else
       gray[(int)bitpos]=0;
      }
      /* for W- PSS */
      if((bitpos>=25)&&(bitpos<=33)) /*
                                    Kpss*/
      {
       if((tp&mask) == 1)
       {
            gray[(int)bitpos]=1;
       }
       else
       gray[(int)bitpos]=0;
      }
      if((bitpos>=34)&&(bitpos<=37))
      {
       if((tp&mask) == 1)
       {
            gray[(int)bitpos]=1;
       }
       else
       gray[(int)bitpos]=0;
      }
      if((bitpos>=38)&&(bitpos<=41))
      {
       if((tp&mask)==1)
       {
            gray[(int)bitpos]=1;
       }
       else
       gray[(int)bitpos]=0;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        }
        if((bitpos>=42)&&(bitpos<=45))
        {
         if((tp&mask) == 1)
         {
               gray[(int)bitpos]=1;
         }
          else
         gray[(int)bitpos]=0;
        }
        if((bitpos>=46)&&(bitpos<=49))
        {
         if((tp&mask)==1)
         {
               gray[(int)bitpos]=1;
         }
         else
         gray[(int)bitpos]=0;
        }
        tp = tp>>=1;
      }

}
bcd[6]=gray[6];
for(i=5;i>=0;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[14]=gray[14];
for(i=13;i>=7;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[24]=gray[24];
for(i=23;i>=15;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[33]=gray[33];
for(i=32;i>=25;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[37]=gray[37];
for(i=36;i>=34;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[41]=gray[41];
for(i=40;i>=38;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[45]=gray[45];
for(i=44;i>=42;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[49]=gray[49];
for(i=48;i>=46;i--)
bcd[i]=bcd[i+1]^gray[i];

for(i=0;i<=6;i++)
{
alpha3 += bcd[i]*pow(2.0,(double)i);
}
for(i=7;i<=14;i++)
{
alpha2 += bcd[i]*pow(2.0,(double)(i-7));
}
for(i=15;i<=24;i++)
{
alpha1 += bcd[i]*pow(2.0,(double)(i-15));
}
for(i=25;i<=33;i++)
{
kpss += bcd[i]*pow(2.0,(double)(i-25));
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    for(i=34;i<=37;i++)
    {
    t1 += bcd[i]*pow(2.0,(double)(i-34));
    }
    for(i=38;i<=41;i++)
    {
    t2 += bcd[i]*pow(2.0,(double)(i-38));
    }
    for(i=42;i<=45;i++)
    {
    t3 += bcd[i]*pow(2.0,(double)(i-42));
    }
    for(i=46;i<=49;i++)
    {
    t4 += bcd[i]*pow(2.0,(double)(i-46));
    }

    alpha1 =
(double)((int)(alpha1*1000/1023))/1000;
    alpha2 =  (double)((int)(alpha2*180/255));
    alpha3 =  (double)((int)(alpha3*100/127))/100;
    kpss   = 0.1 + (double)((int)(kpss*499/511))/10;
    t1     = 0.2 + (double)((int)(t1*13/15))/10;
    t2     = 0.02 + (double)((int)(t2*13/15))/100;
    t3     = 0.2 +  (double)((int)(t3*13/15))/10;
    t4     = 0.02 + (double)((int)(t4*13/15))/100;

    critter1->fitness = (2500/(performance_index
(alpha1,alpha2,alpha3,
                              kpss,t1,t2,t3,t4)));
    fprintf(outfp,"t1  = %f  t2  = %f  t3  = %f
\n",alpha1,alpha2,alpha3);
    fprintf(outfp,"Kpss = %f  T1 = %f  T2 = %f  T3
= %f  T4 = %f\n",kpss,t1,t2,t3,t4);
}

void print_fitness2(struct individual *critter1)
{
    unsigned mask=0x1;   /* mask for current bit */
    int bitpos;    /* current bit position */
    unsigned bitpos2=0,bitpos3=0;
    unsigned tp;
    double pow(), bitpow1=0,
coef,bitpow2=0,bitpow3=0;
    int i,j, k, stop;
    int n = 10;
    double alpha1,alpha2,alpha3;
    double kpss,t1,t2,t3,t4;
    static int gray[50],bcd[50];

    alpha1=0.0; alpha2=0.0; alpha3=0.0;kpss = 0.0;
    t1 = 0.0;t2 = 0.0;t3 = 0.0;t4 = 0.0;
    bitpow2=0; bitpow3=0;
    critter1->fitness = 0.0;
    /* loop number of bytes holding chromosome */
    for(k = 0; k <= chromsize-1; k++)
    {
      if(k == (chromsize-1))
        {stop = lchrom-(k*INTSIZE);}
      else
        {stop = INTSIZE;}
      /* loop bits in current byte */
      tp = critter1->chrom[k];
      for(j = 0; j < stop; j++)
      {
        bitpos = j + INTSIZE*k;
        /* test for current bit 0 or 1 */
        if((bitpos>=0)&&(bitpos<=6))
        {
         if((tp&mask) == 1)
         {
                gray[(int)bitpos]=1;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 }
 else
 gray[(int)bitpos]=0;
}
if((bitpos>=7)&&(bitpos<=14))
{
 if((tp&mask) == 1)
 {
        gray[(int)bitpos]=1;
 }
  else
 gray[(int)bitpos]=0;
}
if((bitpos>=15)&&(bitpos<=24))
{
 if((tp&mask)==1)
 {
        gray[(int)bitpos]=1;
 }
 else
 gray[(int)bitpos]=0;
}
/* for W-PSS */
if((bitpos>=25)&&(bitpos<=33)) /*
Kpss*/
{
 if((tp&mask) == 1)
 {
        gray[(int)bitpos]=1;
 }
 else
 gray[(int)bitpos]=0;
}
if((bitpos>=34)&&(bitpos<=37))
{
 if((tp&mask) == 1)
 {
        gray[(int)bitpos]=1;
 }
  else
 gray[(int)bitpos]=0;
}
if((bitpos>=38)&&(bitpos<=41))
{
 if((tp&mask)==1)
 {
        gray[(int)bitpos]=1;
 }
 else
 gray[(int)bitpos]=0;
}
if((bitpos>=42)&&(bitpos<=45))
{
 if((tp&mask) == 1)
 {
        gray[(int)bitpos]=1;
 }
  else
 gray[(int)bitpos]=0;
}
if((bitpos>=46)&&(bitpos<=49))
{
 if((tp&mask)==1)
 {
        gray[(int)bitpos]=1;
 }
 else
 gray[(int)bitpos]=0;
}
tp = tp>>=1;
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
}
bcd[6]=gray[6];
for(i=5;i>=0;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[14]=gray[14];
for(i=13;i>=7;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[24]=gray[24];
for(i=23;i>=15;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[33]=gray[33];
for(i=32;i>=25;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[37]=gray[37];
for(i=36;i>=34;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[41]=gray[41];
for(i=40;i>=38;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[45]=gray[45];
for(i=44;i>=42;i--)
bcd[i]=bcd[i+1]^gray[i];

bcd[49]=gray[49];
for(i=48;i>=46;i--)
bcd[i]=bcd[i+1]^gray[i];


for(i=0;i<=6;i++)
{
alpha3 += bcd[i]*pow(2.0,(double)i);
}
for(i=7;i<=14;i++)
{
alpha2 += bcd[i]*pow(2.0,(double)(i-7));
}
for(i=15;i<=24;i++)
{
alpha1 += bcd[i]*pow(2.0,(double)(i-15));
}
for(i=25;i<=33;i++)
{
kpss += bcd[i]*pow(2.0,(double)(i-25));
}
for(i=34;i<=37;i++)
{
t1 += bcd[i]*pow(2.0,(double)(i-34));
}
for(i=38;i<=41;i++)
{
t2 += bcd[i]*pow(2.0,(double)(i-38));
}
for(i=42;i<=45;i++)
{
t3 += bcd[i]*pow(2.0,(double)(i-42));
}
for(i=46;i<=49;i++)
{
t4 += bcd[i]*pow(2.0,(double)(i-46));
}

alpha1 =
(double)((int)(alpha1*1000/1023))/1000;
alpha2 = (double)((int)(alpha2*180/255));
alpha3 = (double)((int)(alpha3*100/127))/100;
kpss = 0.1 + (double)((int)(kpss*499/511))/10;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    t1    = 0.2 + (double)((int)(t1*13/15))/10;
    t2    = 0.02 + (double)((int)(t2*13/15))/100;
    t3    = 0.2 +  (double)((int)(t3*13/15))/10;
    t4    = 0.02 + (double)((int)(t4*13/15))/100;

    critter1->fitness = (2500/(performance_index
(alpha1,alpha2,alpha3,
                                kpss,t1,t2,t3,t4)));
    fprintf(fpga,"t1  = %f  t2  = %f  t3  = %f
\n",alpha1,alpha2,alpha3);
    fprintf(fpga,"kpss = %f  T1 = %f  T2 = %f T3 =
%f T4 = %f  \n",kpss,t1,t2);
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ข.

โปรแกรมของระบบจำลอง

MODEL.C

โปรแกรมนี้ใช้เป็นแบบจำลองในการทดลองใช้ตัวควบคุมแบบไฮบริดเป็นตัวรักษาเสถียรภาพของระบบไฟฟ้ากำลัง โดยจะนำเสนอเพียงส่วนของระบบจำลองที่ใช้ควบคุมแบบไฮบริดเท่านั้น หากผู้ใดต้องการศึกษาเพิ่มเติมเกี่ยวกับแบบจำลอง PSS หรือ SVC ต้องดัดแปลงการใช้งานเอง

```
#include <stdio.h>
#include <stdlib.h>
#include     <conio.h>
#include <float.h>
#include <math.h>
#include "c:\ga\gtool.h"
#define          Umfp    0.2  /*Max of control
signal */
#define          SL3     135.0 /* switching
line */
#define          Pmax    1.05 /*    Limit of
GOV */
#define          Efmax   7.30 /*--- Limit of
field voltage (max)---*/
#define          Efmin   -7.30 /*--- Limit of
field voltage (min)---*/
#define          Bsmax   1.0  /*--- Limit of
SVC susceptance(max)---*/
#define          Bsmin   -1.0  /*--- Limit of
SVC susceptance(min)---*/
#define          Ue      200.0     /*
          Exciter gain              */
#define          Te      0.05      /*
Exciter time constant          */
#define          Kf    /*0.0562*1.25*/0.07
/*  Hunting parameter                */
#define          Tf    1.250          /*  Time setting
                                      */
#define          SIGM    0.05
/*          Permanent drop          */
#define          Tg      0.20
/*          Gate time conatant */
#define          Tw      2.000
/*          Water time constant          */
#define          P0      1.0
/*          Real power                    */
#define          V0      1.0
/*          Reactive power                */
#define          Vt0     1.0               /*
                    Terminal voltage   */
#define          W0      377.0             /*
                    Synchronous speed*/
#define          Xdd     (( 1.0 - xe*xa ) * xdd
+ xe )
#define          Xq      (( 1.0 - xe*xa ) * xq +
xe )
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
#define Xd (( 1.0 - xe*xa ) * xd +
xe )
#define T 0.005 /*
Sampling time */
#define N 1201
/* Number of data */
#define LIMIT (T*(double)(N-1))/*
Max. number of data */
#define NA 20 /*
Number of differential equations */
#define Vc0 1.0
#define Tc 0.05
#define Ksv 50.0
#define Tv 0.01
#define TQ1 4.0
/*fuzzy control*/
#define TP1 0.01 /*fuzzy
control*/
#define Umpss 0.1
#define Kpss 1
#define Tm 0.056
#define Tr 4.0
#define PI 3.14159265358979312
        /*----fuzzy parameters ----*/
static double
alpha1,alpha2,alpha3,alpha4,Zamin,alpha5,
    Zsmin,gamma1,gamma2,Tr1,Tr2,At;
        /* alpha1,2,3 */
        static double Kppss,T1,T2,T3,T4;
/*--- initial condition variables ---*/
static double
vd0,vq0,id0,iq0,del0,del1,Q0,Q0l,
Q0t,fif0,vf0,vc0,v0,Bs0;
static double xc,xe,xa,xb,xl,
              xt = 0.14, //Xt
              xl1 = 0.21, //1/2 per cct
              xl2 = 0.30, //1/2 per cct
              xee; //Xj

static double xd = 1.620, /*
d-axis reactance */
              xdd = 0.250, /*
d-axis transient reactance */
              xq = 1.600,
              /* q-axis reactance */
              Td0d = 5.600, /*
d-axis open circuit time constant */

              M = 0.0191, /*
Inertia constant */

              D = 0.00531; /*
Damping coeffient */

static double u1[2],u11[2], /* Control
signal for SVC */
dy1[N], /* Real power Pe */
dy2[N], /* Terminal voltage Vt */
dy3[N], /* SVC Vc*/
ti, /* simulation time */
Ups,Upss,Ufl,
P1,out,out1,out22;
static double Pd; //small disturbance
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
static double        ZZ;  //use in AVR limit
static double  Za,Zs,Za0,Zs0,MYU,DE3,
DE1,DE,EE3,EE1,Dd,Dr;

static double        SDEP,SEEP,SDkP,STkP;
//fuzzy control

static double        Jw1;
//performance index

static double        Jw2;
//performance index

static int DD1,DD2,DXD1,DXD2;
//limit in GOV

static    int        AVRL, AVRL1;      //limit
in AVR

static    int        SVCL, SVCL1;      //limit in
SVC

static int    PSS,PSS1;

static int kpt,CH,KH,select;  //count number

   //select disturbance

   //trip time(large disturbance)
   //select control
/* Prototype */
void      shoki(void);
void      x_stand(double x[]);
void   d_out(double          y1[]);
void      runge4(double x[],double u11[],double
t,int n,double dy2[N],double dy3[N],int flag1);
void      sys1(double x[],double u11[],double f
[],double y[],double dy2[N],double dy3[N],int
flag1);
void      svcfuz(double u11[],double Vt,double
Pe,double x1);
void      svcPD(double u11[],double f[],double x1
[]);
void      svcsita(double DE,double EE,double
Dk,double *Tk);
void      svcsimem(double Tk,double a,double
*Nk,double SL);
void      prepa(double Zs,double Zs0,double
Za,double Za0,double u11[],double Pe,double x2);
double  performance_index(double alpha1,double
alpha2,double alpha3,double alpha4,double
Zamin,double alpha5,double Zsmin,double
k,double tt1,double tt2,double tt3,double
tt4,double gamma1,double gamma2,double
Tr1,double Tr2,double At);
unsigned _stack = 50000;
FILE *fp,*fpw;

double  performance_index(double alpha1,double
alpha2,double alpha3,double alpha4,double
Zamin,double alpha5,double Zsmin,double
k,double tt1,double tt2,double tt3,double
tt4,double gamma1,double gamma2,double
Tr1,double Tr2,double At)
{

static    double    GD;
static    double    x1[NA], y1[4], f[NA];
static  double      dt1[2],dt2[2],dt3[2],dt4[2];
static  int flag,fri;
int       stop;
float     Time;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Kppss = k;
T1 = tt1;
T2 = tt2;
T3 = tt3;
T4 = tt4;
Ups=0;
Ufl=0;
Upss=0;
Pd=0.0;
ZZ=0.0;

Za=0.0;Zs=0.0;Za0=0.0;Zs0=0.0;DE=0.0;DE1=0.0
;DE3=0.0;
EE1=0.0;EE3=0.0;Dd=0.0;Dr=0.0;

SDEP=0.0;
SEEP=0.0;
SDkP=0.0;
STkP=0.0;
AVRL=0.0;
AVRL1=0.0;
DD1=0.0;
DD2=0.0;
DXD1=0.0;
DXD2=0.0;
SVCL=0.0;
SVCL1=0.0;
PSS=PSS1 = 0.0;
vd0=0.0;
vq0=0.0;
id0=0.0;
iq0=0.0;
del0=0.0;
del1=0.0;
Q0=0.0;
Q0l=0.0;
Q0t=0.0;
fif0=0.0;
vf0=0.0;
vc0=0.0;
v0=0.0;
Bs0=0.0;
xc=0.0;
xe=0.0;
xa=0.0;
xb=0.0;
xl=0.0;
for (kpt = 0; kpt < N; kpt++)
{
 dy1[kpt] = 0.0;
 dy2[kpt] = 0.0;
 dy3[kpt] = 0.0;
 u1[0]  = 0.0;
 u1[1]  = 0.0;
 u11[0]  = 0.0;
 u11[1]  = 0.0;
}
Jw1=Jw2=0.0;
xt  = 0.14;
xl1 = 0.21;
xl2 = 0.30;
xee = 0.05*At;
xd   =1.620; /*  d-axis reactance      */
xdd  =0.250; /*  d-axis transient reactance  */
xq   =1.600; /*  q-axis reactance      */
Td0d  =5.600;      /*  d-axis open circuit time
constant  */
M    =0.0191;      /*  Inertia constant*/
D    =0.00531;
GD=0.0;
for (kpt = 0; kpt < NA; kpt++)
{
```

นี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  x1[kpt] = 0.0;
  f[kpt] = 0.0;
}
for (kpt = 0; kpt < 4; kpt++) y1[kpt] = 0.0;
for (kpt = 0; kpt < 2; kpt++)
{
  dt1[kpt] = 0.0;
  dt2[kpt] = 0.0;
  dt3[kpt] = 0.0;
  dt4[kpt] = 0.0;
}
flag=0;
fri=0;
kpt=0.0;
ti=0.0;
select=3;
Time=0.1;
(int)KH=Time/0.005;
//sbp=alpha1;sap=alpha2;sDrp=alpha3;

for(CH=0;CH<=1;CH++){

Ups=0;
Ufl=0;
Upss=0;
Pd=0.0;
ZZ=0.0;

Za=0.0;Zs=0.0;Za0=0.0;Zs0=0.0;DE=0.0;DE1=0.0
;DE3=0.0;
EE1=0.0;EE3=0.0;Dd=0.0;Dr=0.0;

SDEP=0.0;
SEEP=0.0;
SDkP=0.0;
STkP=0.0;
AVRL=0.0;
AVRL1=0.0;
DD1=0.0;
DD2=0.0;
DXD1=0.0;
DXD2=0.0;
SVCL=0.0;
SVCL1=0.0;
PSS=PSS1 = 0.0;
vd0=0.0;
vq0=0.0;
id0=0.0;
iq0=0.0;
del0=0.0;
del1=0.0;
Q0=0.0;
Q0l=0.0;
Q0t=0.0;
fif0=0.0;
vf0=0.0;
vc0=0.0;
v0=0.0;
Bs0=0.0;
xc=0.0;
xe=0.0;
xa=0.0;
xb=0.0;
xl=0.0;
for (kpt = 0; kpt < N; kpt++)
{
 dy1[kpt] = 0.0;
 dy2[kpt] = 0.0;
 dy3[kpt] = 0.0;
 u1[0]  = 0.0;
 u1[1]  = 0.0;
 u11[0]  = 0.0;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 u11[1]  = 0.0;
}

xt  = 0.14;
xl1 = 0.21;
xl2 = 0.30;
xee = 0.05*At;
xd    =1.620; /*  d-axis reactance       */
xdd   =0.250; /*  d-axis transient reactance  */
xq    =1.600; /*  q-axis reactance       */
Td0d  =5.600;       /*  d-axis open circuit time
constant  */
M     =0.0191;      /*  Inertia constant*/
D     =0.00531;
GD=0.0;
for (kpt = 0; kpt < NA; kpt++)
{
 x1[kpt] = 0.0;
 f[kpt] = 0.0;
}
for (kpt = 0; kpt < 4; kpt++) y1[kpt] = 0.0;
for (kpt = 0; kpt < 2; kpt++)
{
 dt1[kpt] = 0.0;
 dt2[kpt] = 0.0;
 dt3[kpt] = 0.0;
 dt4[kpt] = 0.0;
}
flag=0;
fri=0;
kpt=0.0;
ti=0.0;
flag = 0;
xl = xt + xl1 + xl2 + xee;

/*--------     Initial Condition of System     ---------
--*/
shoki();
if(CH==1) Pd = 0.0;      //small disturbance
else        Pd = 0.0;

/*--------     Initial Condition of State Equations --
----*/
            x_stand(x1);
/*---------    On line 1 ( Loop for simulation)  -----
----*/

for (kpt = 0;kpt <=N-1;kpt++)
/* start loop */

{           ti = (double)kpt * T;
if(CH==0)
//large disturbance
if(kpt==0) flag = 1;   //3-phase fault
if(CH==0)
if(kpt==0+KH)              //trip 1 line
{           flag = 0;
            xl1 = 2.0*xl1;
}

if(CH==1)
//small disturbance

if(kpt>=0)

            if (kpt < 100)

            Pd = -0.1/100*kpt;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
if(CH==1)

        if(kpt>=100)
        if (kpt < 200)
        Pd = -0.2 + 0.1 /100*kpt;
if(CH==1)
        if(kpt>=200)
        Pd = 0.0;

/*---------  starting point of control action  -------
--*/

if(kpt%2==0)
{
Zs0=Zs;
if(Zs0>(Zsmin)){Zs0=0.0;}
Za0=Za;
if(Za0>(Zamin)){Za0=0.0;}
prepa(Zs,Zs0,Za,Za0,u11,x1[14],x1[1]);
}
else      u11[1]=u11[0];
if(select==3)if(kpt%2==0)   //sampling time for
control=0.01s
Ufl=MYU*(u11[1]-Ups);
/*---------  ending point of control action  -------
--*/
/*---System state equations calculation using
Rungekutta---*/
        AVRL = AVRL1;
        DD1  = DXD1;
        DD2  = DXD2;
        SVCL = SVCL1;
        PSS  = PSS1;
runge4(x1, u11, T, NA,dy2,dy3,flag);
/*---------Limiter of A V R          ---------*/
if(x1[3] > Efmax - vf0)
{       AVRL1 = 1;
        ZZ = Efmax - vf0;
}
else    if(x1[3] < Efmin - vf0)
{       AVRL1 = -1;
ZZ = Efmin - vf0;
}
else  {
AVRL1 = 0;
ZZ = x1[3];
        }

Ups = out;
if(Ups>Umpss) Ups = Umpss;
if(Ups<-Umpss) Ups = -Umpss;
if(Ups+Ufl>Umpss)  {
        PSS1 = 1;
        Upss = Umpss;
        }
      else
      if (Ups+Ufl<-Umpss) {
        PSS1 = -1;
        Upss = - Umpss;
      }
      else    {
        PSS1 = 0.0;
        Upss = Ups+Ufl;
      }
/*---------Limiter of G O V          ---------*/
GD=(-x1[1]/W0 - SIGM*x1[5]) /Tg;
// dg/dt

if(GD > 0.1)

        DXD1=1;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        else    if(GD < -0.1)

        DXD1=-1;
        else    DXD1=0;
        if(x1[5] > (Pmax-(P0-Pd)))
        DXD2=1;
        else    if(x1[5] < -(P0-Pd))
        DXD2=-1;
/*---------calulation system          ---------*/
sys1(x1, u11, f, y1,dy2,dy3,flag);
/*---------  Calculation of Deviation of Output
---------*/
d_out(y1);

/*---------calulation Moment of inertia      ---------
*/
if(CH==0){
        Jw1 += fabs(x1[1])* ti+fabs((u11[1]-u11
[0])/T);
        }
if(CH==1){
        Jw2 += fabs(x1[1])* ti+fabs((u11[1]-u11
[0])/T);
        }
                u11[0] = u11[1];

} /* End 1200 loop */

   /*printf("%10.5f \n",Jw1); */
}/* end CH loop */
   return(Jw1+0.1*Jw2);

}
/*---------          End main
---------*/
```

```
/*-------------------------------------------------
*/
/*      Initial Condition of state variables
*/
/*-------------------------------------------------
*/

void    x_stand(double    x[])
{
        x[0] = del0;
        x[1] = 0.0;
        x[2] = fif0;
        x[3] = 0.0;
        x[4] = 0.0;
        x[5] = 0.0;
        x[6] = 0.0;
        x[7] = 0.0;
        x[8] = 0.0;
        x[9] = 0.0;
        x[10] = 0.0;
        x[11]= 0.0;
        x[12]= 0.0;
        x[13]= 0.0;
        x[14]= 0.0;
        x[15]= 0.0;
        x[16]= 0.0;
        x[17]= 0.0;
        x[18]= 0.0;
        x[19]= 0.0;
}

/*-------------------------------------------------
*/
/*              Innitial system equation
*/
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*-------------------------------------------------
*/
void    shoki()
{
static double          A, C;
/*--------- 1.delta1 ---------*/
del1 = asin(P0*xl/2.0);
/*--------- 2. xc ---------*/
          /*xc = xl/(4.0*(1.0 - cos(del1)));
          Bs0 = 1.0/xc;*/
/*--------- 3.xe,xa,xb ---------*/
/*xe = (xc*(xt+xl1+xl2+xee)-(xt+xl1)*
(xl2+xee))/xc;*/
/*xa = (xc*(xt+xl1+xl2+xee)-(xt+xl1)*(xl2+xee))/
(xl2+xee);*/
/*xb = (xc*(xt+xl1+xl2+xee)-(xt+xl1)*(xl2+xee))/
(xt+xl1);xa = 1.0/xa;*/
xe = xt+xl1+xl2+xee;
xa = 0;
/*--------- 4.delta1 ---------*/
del1 = asin(P0*xe);
/*--------- 5. Q0 ---------*/
          Q0l = (1.0-cos(del1))/xe;
          Q0t = xa;
          Q0  = Q0l - Q0t;
/*--------- 6.vd0 Calculation ---------*/
          A   = ( Vt0 * Vt0 / xq + Q0);
          A   = A * A;
          A   = A + P0 * P0;
          vd0 = P0 * Vt0 / sqrt(A);
/*--------- 7.iq0 Calculation ---------*/
          C   = Vt0 * Vt0 + xq * Q0;
          C   = C * C;
          C   = C + xq * xq * P0 * P0;
          iq0 = P0 * Vt0 / sqrt(C);
/*--------- 8.vq0 Calculation ---------*/
          A   = Vt0 * Vt0 - vd0 * vd0;
          vq0 = sqrt(A);
/*--------- 9.id0 Calculation ---------*/
          id0 = (P0 - vq0 * iq0) / vd0;
/*--------- 10.del0 Calculation ---------*/
          del0 = asin((iq0*xe+vd0-
vd0*xa*xe)/V0);

/*--------- 11.fif0 Calculation ---------*/
          fif0 = ( xdd * V0 * cos(del0)/Xdd )
          fif0 = ( vq0 - fif0 ) * Xdd * Td0d / xe ;
/*--------- 12.vf0 Calculation ---------*/
                    vf0 = (Xd*fif0) /
(Xdd*Td0d) - (xd-xdd)*V0*cos(del0) / Xdd;
          /*--------- 13.vc0 calculation ---
------*/
                    vc0 = sqrt((vd0+
(xt+xl1)*iq0)*(vd0+(xt+xl1)*iq0)
                              +(vq0-
(xt+xl1)*id0)*(vq0-(xt+xl1)*id0));
          /*---------14.v0 Calculation ----
-----*/
                    A = iq0 * xe + vd0 -
vd0 * xa * xe;
                    C = sin (del0);
                    v0 = A/C;
          }

/*--------------------------------*/
/*      Deviation of Output
*/
/*--------------------------------*/
void      d_out(double      y1
[])
{
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                dy1[kpt+1] = y1[0] -
P0;
                dy2[kpt+1] = y1[1] -
Vt0;
                dy3[kpt+1] = y1[3] -
vc0;
        }

        /*----------------------------*/
/*      System Equation     */
/*----------------------------*/
void    sys1(double x[],double u11[],double f
[],double y[],double dy2[N],double dy3[N],int
flag1)
{       static double
        A1,A2,B1,B2,B3,C1,C2;
        double
vd,     /*      d axis voltage      */
vq,     /*      q axis voltage      */
id,     /*      d axis current  */
iq;     /*      q axis current  */
/*--- Operating point calculation ---*/
if(flag1==1)
{       /*A1 = xl1*xc/(xc-xl1);*/xl1;
        A2 = xl2+xee;
        B1 = 2.0*xl1*xl1 / (3.0*xl1+A1);
        B2 = xl1*A1/(3.0*xl1+A1);
        B3 = 2.0*xl1*A1/(3.0*xl1+A1);
        C1 = xt+B1;
        C2 = A2+B3;
        xe = (C1*C2+B2*(C1+C2))/B2;
        xa = (C1*C2+B2*(C1+C2))/C2;
        xa = -1.0/xa;
        xb = (C1*C2+B2*(C1+C2))/C1;
}
if(flag1==0)
{       /*xe = (xc*(xt+xl1+xl2+xee)-(xt+xl1)*
(xl2+xee))/xc;
xa = (xc*(xt+xl1+xl2+xee)-(xt+xl1)*(xl2+xee))/
(xl2+xee);
xb = (xc*(xt+xl1+xl2+xee)-(xt+xl1)*(xl2+xee))/
(xt+xl1);
xa = 1.0/xa;*/
 xe = xt+xl1+xl2+xee;
xa = 0;
}
/*---------d axis voltage calculation      ---------*/
vd = xq * V0 * sin(x[0]) / Xq;
/*---------q axis voltage calculation      -------*/
vq = (xe*x[2])/(Xdd*Td0d)+xdd*V0*cos(x
[0])/Xdd;
/*---------d axis current calculation      --------*/
id = x[2]*(1.0-xe*xa)/(Xdd*Td0d)-V0*cos(x
[0])/Xdd;
/*---------          q axis current calculation  -----
----*/
iq = V0 * sin(x[0]) / Xq;
/*---------  Real Power calculation:Pe        --------
*/
y[0] = vd * id + vq * iq;
/*---------  Terminal Voltage calculation:Vt    -----
----*/
y[1] = sqrt(vd * vd + vq * vq);
/*---------  Reactive Power calculation:Qe     ------
--*/
y[2] = vq * id - vd * iq;
/*---------  S V C Terminal Voltage calculation:Vc
---------*/
y[3] = sqrt((vd+(xt+xl1)*iq)*(vd+(xt+xl1)*iq)
+(vq-(xt+xl1)*id)*(vq-(xt+xl1)*id));
/*---------  Nonlinear differential equation  ---------
*/
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
/*--------- delta: torque angle          ---------*/
        f[0] = x[1];
/*--------- 1: w        ---------*/
f[1] = 1.0 / M * (x[6] - D * x[1] - (y[0]-(P0-Pd)));
/*--------- 2: f        ---------*/
switch(AVRL)
        {       case 1 :f[2] = Efmax -  x[2] /
Td0d - (xd - xdd)*id;
        break;
                case -1:f[2] = Efmin -  x[2] /
Td0d - (xd - xdd)*id;
        break;
                case 0 :f[2] = (vf0 + x[3])  -  x
[2] / Td0d - (xd - xdd)*id;
        break;
        }
/*--------- 3: vf                    ---------*/
switch(PSS)
        {
                case 1 :
            f[3] = (- x[3] + Ue * (- dy2[kpt] -
(Kf*ZZ-x[4])/Tf+Umpss))/Te;
            break;
                case -1:
            f[3] = (- x[3] + Ue * (- dy2[kpt] -
(Kf*ZZ-x[4])/Tf-Umpss))/Te;
            break;
                case 0 :
            f[3] = (- x[3] + Ue * (- dy2[kpt] -
(Kf*ZZ-x[4])/Tf+Upss))/Te;
            break;
        }
         /*      f[3] = (- x[3] + Ue * (- dy2[kpt]
-(Kf*ZZ-x[4])/Tf))/Te;  */
/*--------- 4: Rf                    ---------*/
f[4] = (Kf * ZZ -x[4] ) / Tf;
/*--------- 5: g        ---------*/
switch(DD1)
        {   case 1 :f[5] =  0.1;
                break;
            case -1:f[5] = -0.1;
                break;
            case 0 :f[5] = -SIGM * x[5] / Tg + 1.0 /
Tg * - x[1] / W0;
                break;
        }
/*--------- 6: h                     ---------*/
switch(DD2)
{   case  1 :f[6] = (-x[6]+(Pmax-(P0-Pd)))/Tw;
                break;
case -1 :f[6] = (-x[6]-(P0-Pd))/Tw;
                break;
case  0 :f[6] = (-x[6]+x[5])/Tw;
                break;
}
/*--------- 7: SVC -> Vcm                ----*/
            f[7] = (-dy3[kpt] - x[7]) /Tv;
/*---------          8: SVC -> Pin        ---------*/
f[14] = ((y[0] - (P0 -Pd )) - x[14]) / TP1;
/*--------- 9: SVC -> Pout       ---------*/
f[15] = ((y[0] - (P0 -Pd )) - x[14]) / TP1 -x[15]/
TQ1;
f[16] = ((y[0] - (P0 -Pd ))*W0 / M - x[14]) / TP1 ;
f[17] = x[16]-x[17]/Tr1;
Za = x[16]-x[17]/Tr1;
f[18] = Za-x[18]/Tr2;
Zs = Za - x[18]/Tr2;
   /*      f[10] =  (x[9]-x[10])/0.05;
f[11] = (0.04*((-(y[0] - (P0 -Pd )) - x[8]) / TP1 -x
[9]/ TQ1)-x[11])/0.05;
x[12] =x[10]+x[11];
out = x[12]; for □ PSS   */
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
f[8] = (x[1]-x[8])/Tm;
f[9] = (Tr*(Kppss/W0)*(x[1]-x[8])/Tm-x[9])/Tr;
f[10] = (x[9]-x[10])/T2;
f[11] = (T1*(Tr*(Kppss/W0)*(x[1]-x[8])/Tm-x
[9])/Tr-x[11])/T2;
out1 = x[10]+x[11];
f[12] = (out1-x[12])/T4;
f[13]=(T3*((x[9]-x[10])/T2+(T1*(Tr*
(Kppss/W0)*(x[1]-x[8])/Tm-x[9])/Tr-x[11])/T2)-x
[13])/T4;
out = (x[12]+x[13]);
/*f[14]=((-y[0]+(P0-Pd))/M-x[14])/Tpi;
f[9] =x[14]-x[9]/TQ1;
Za=x[14]-x[9]/TQ1;
/*** 1/s ***/
/*f[16]=x[14]-x[9]/TQ1;
f[17]=x[16]-x[17]/TQ2;
Zs=x[16]-x[17]/TQ2;*/

}
/*-----------------------------*/
/*     Rungekutta Method     */
/*-----------------------------*/
void       runge4( double x[],double u11[],double
t,int n,double dy2[N],double dy3[N],int flag1)
{
double k1[NA], k2[NA], k3[NA], k4[NA], y
[3], z[NA];
int    i;

sys1(x, u11, k1, y,dy2,dy3,flag1);
    for (i = 0; i < n; i++)
  z[i] = x[i] + t / 2 * k1[i];

sys1(z, u11, k2, y,dy2,dy3 ,flag1);
    for (i = 0; i < n; i++)
  z[i] = x[i] + t / 2 * k2[i];

sys1(z, u11, k3, y,dy2,dy3,flag1);
    for (i = 0; i < n; i++)
  z[i] = x[i] + t * k3[i];

sys1(z, u11, k4, y,dy2,dy3,flag1);
    for (i = 0; i < n; i++)
  x[i] = x[i] + t * (k1[i] + 2 * k2[i] + 2 * k3[i] + k4
[i]) / 6;
}

/*-----------------------------*/
/*        SVC fuzzy control */
/*-----------------------------*/
void svcfuz(double u1[],double Vt,double
Pe,double x1)
{
static double SNkP,SGkP;
SNkP = 0.0;
SDEP = 0.012*(x1-SEEP)/(2.0*T);
SEEP = x1;
SDkP = sqrt(SDEP*SDEP + x1*x1);
/*---------          Phase plan select          ------
---*/
svcsita(SDEP,x1,SDkP,&STkP);
svcsimem(STkP,90,&SNkP,SL3);
/*--------- determine Gain factor          ---------*/
SGkP = SDkP/0.24;
if(SGkP>1.0)
SGkP=1.0;
/*----------  calculate the SVC control signal  -----
----*/
u1[1] = (2.0*SNkP-1.0)*Umfp*SGkP;
        if(u1[1] < -Umfp) u1[1] =-Umfp;
        if(u1[1] >  Umfp) u1[1] = Umfp;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
}

void    prepa(double Zs,double Zs0,double
Za,double Za0,double u11[],double Pe,double x2)
{
static    double    Nk3,Gk3,x5,x6;
Nk3=0.0;
x6=alpha1*Za-alpha5*Zs0;
DE3=x6;
/******  DE3=b3*(Zs-EE3)/ST;  ******/
DE=Zs-EE3;
EE3=Zs;
x5=(Zs-alpha4*Za0);
/*****   x5=(Zs+sos*x7); *********/
Dr=sqrt(DE3*DE3+x5*x5);
Dd=sqrt(Za*Za+Zs*Zs);
DE1=(Pe-EE1)/0.01;
EE1=Pe;
svcsita(DE3,x5,Dr,&STkP);
svcsimem(STkP,alpha2,&Nk3,SL3);
Gk3=Dr/alpha3;
if(Gk3>1.0) Gk3=1.0;
if(gamma1==gamma2)
{
if(Dd<gamma1) MYU=1.0;
else          MYU=0.0;
}
else
{
MYU=(Dd-gamma2)/(gamma1-gamma2);
if(MYU>1.0) MYU=1.0;
if(MYU<0.0) MYU=0.0;
}
MYU=1.0-MYU;
/***      MYU=1.0;   ***/
/***    MYU=0.0;   ***/
/****   MYU=1.0-MYU;    ****/
/****      MYU=1.0;  F1    ****/
/****      MYU=0.0;  PPSS   ****/
          u11[1]=(2.0*Nk3-1.0)*Umfp*Gk3
/**+MYU*uu1**/;
if(u11[1]<-Umfp)u11[1]=(-Umfp);
if(u11[1]>Umfp)u11[1]=Umfp;
/****      U1=(u[0][kpt]-U2);***/
}

/*--------------------------*/
/*          Find angle              */
/*--------------------------*/
void      svcsita(double DE,double EE,double
Dk,double *Tk)
{
          if(DE>0.0 && EE>0.0)
 *Tk= asin(DE/Dk)*180.0/PI;
          if(DE>0.0 && EE<0.0)
 *Tk= -asin(DE/Dk)*180.0/PI+180.0;
          if(DE<0.0 && EE<0.0)
 *Tk= -asin(DE/Dk)*180.0/PI+180.0;
          if(DE<0.0 && EE>0.0)
 *Tk= asin(DE/Dk)*180.0/PI+360.0;
}

/*---------------------------------------*/
/*Find N-membership gain for U(f)     */
/*---------------------------------------*/
void      svcsimem(double Tk,double a,double
*Nk,double SL)
{
if(Tk >= 0.0 && Tk < SL - a / 2.0)
          *Nk = 1.0;
if(Tk >= SL - a/2.0 && Tk <  SL + a / 2.0)
          *Nk = ( - Tk + SL ) / a + 0.5;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
if(Tk >= SL + 180.0 - a / 2.0 && Tk <  SL + 180.0
+ a / 2.0)
        *Nk = ( Tk - ( SL + 180.0 ) ) / a + 0.5;
if(Tk >= SL + 180.0 + a / 2.0  && Tk <= 360.0)
        *Nk = 1.0;
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ค.

โปรแกรมที่ใช้สำหรับแสดงผลทางกราฟ

GPLOT.H

(GTOOL.H)

เป็นโปรแกรมที่ช่วยแสดงผลทางกราฟ จะทำให้การศึกษาในเรื่องผลตอบสนองชัดเจนมากขึ้นและเป็นการตรวจคำตอบของค่าที่ได้จากเจเนติกอัลกอริทึมด้วย

```
#include <dos.h>
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

#define PI 22./7.
#define  coright   0
#define  cotop    0
#define  coleft     639
#define  cobottom  399

typedef unsigned  word;
int       w_vx1,w_vy1,w_vx2,w_vy2;
double  w_sx = 0.0;
double  w_sy = 0.0;
double  w_sy2= 0.0;
double  w_dx = 1.0;
double  w_dy = 1.0;
int    w_col = 7;
word      w_pat = 0xffff;
int       maxcolor;

#ifndef __BCPLUSPLUS__
/*   prototype */
void   w_view(double, double, double, double);
void   w_axis(double, double, double, double);
void   w_axiss(double, double, double, double);/*
Draw Graph */
void   w_attr(int, word);  /*  set attribute */
void   w_line(double, double, int);
void   w_line2(double, double, int, int);
/* Function Prototypes */
void g_init (void);          /* initial graphics
mode */
void g_print(int x,int y,char *st);
void g_screen(int x1,int x2,int x3,int x4); /* initial
attribute */
void g_cls(void);/* clear viewport */
void t_cls(void);/* clear text mode screen */
void g_line(int x1,int y1,int x2,int y2,int col,int
linestyle,int thickness,word pat);
void g_view(int x1,int x2,int x3,int x4,int x5,int
x6);/* set viewport area */
void w_attr22(int ,word );
void w_line222(double,double,int ,int );
#endif

/*      Function      */

void g_view(int x1,int x2,int x3,int x4,int x5,int
x6)
{
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        if (x6 == -1) setviewport
(x1,x2,x3+100,x4+10,x5);
        else setviewport(x1,x2,x3,x4,x5);
}

void g_init (void)          /* initial grahics mode */
{
        int driver = DETECT, mode, ErrorCode;

/* Set up screen in graphics mode, virtual coords
*/
        initgraph (&driver, &mode, "");
        if ((ErrorCode = graphresult ()) != grOk)
        {
                clrscr ();
                gotoxy (5,12);
                printf ("%s",grapherrormsg
(ErrorCode));
                delay (3000);
                exit (0);
        }
        maxcolor=getmaxcolor()+1;
}

void g_print (int x,int y,char *st)
{
        int oldc;

        oldc = getcolor();
        setcolor(maxcolor-5);
        settextjustify
(CENTER_TEXT,CENTER_TEXT);
        settextstyle (2,HORIZ_DIR,6);
        outtextxy (x,y,st);
        setcolor (oldc);
}

void g_cls(void)
{
        cleardevice();
}

void g_line(int x1,int y1,int x2,int y2,int col,int
linestyle,int thickness,word pat)
{
        int   oldc,a,b,c;
        struct linesettingstype *oldlineinfo=0;

        oldc=getcolor();
        setcolor(col);
        getlinesettings(oldlineinfo);
        setlinestyle(linestyle,pat,thickness);
        line(x1,y1,x2,y2);
        setlinestyle((*oldlineinfo).linestyle,
(*oldlineinfo).upattern,(*oldlineinfo).thickness);
        setcolor(oldc);
}

void   w_view(x1, y1, x2, y2)
double  x1, y1, x2, y2;
{
  w_vx1 = (int)x1;
  w_vy1 = (int)y1;
  w_vx2 = (int)x2;
  w_vy2 = (int)y2;

}

void   w_axiss(x1, y1, x2, y2)
double  x1, y1, x2, y2;
{
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    int i1, i2, i3, i4;

    i2 = 5;

    g_line(w_vx1, w_vy2, w_vx1, w_vy1, 9, 0,
0,w_pat);

    i1 = (int)((w_vy2+w_vy1)/2);
    g_line(w_vx1, i1, w_vx2, i1, 9, 0, 0,w_pat);

    i3 = w_vx1+i2;
    g_line(w_vx1, w_vy1, i3, w_vy1, 9, 0,
0,w_pat);
    g_line(w_vx1, w_vy2, i3, w_vy2, 9, 0,
0,w_pat);

    i4 = (int)((i1+w_vy1)/2);
    g_line(w_vx1, i4, i3, i4, 9, 0, 0,w_pat);

    i4 = (int)((i1+w_vy2)/2);
    g_line(w_vx1, i4, i3, i4, 9, 0, 0,w_pat);

    i4 = (int)((w_vx2+w_vx1)/2);
    g_line(i4, i1+3, i4, i1-3, 9, 0, 0,w_pat);

    g_line(w_vx2, i1+3, w_vx2, i1-3, 9, 0, 0,w_pat);

    g_print(i4-10, i1+10, "  3.0");

    g_view(0, 0, 639, 399, -1, -1);
    g_print(w_vx2-14, i1+10, "   6.0");
    g_print((int)((i4+w_vx2)/2)-45, i1+25, "     t [s]
");

    g_view(w_vx1, w_vy1, w_vx2, w_vy2, -1, -1);
    w_sx = x1;
    w_sy = y1;

    w_dx = (x2 - x1) / (w_vx2 - w_vx1);
    w_dy = (y2 - y1) / (w_vy2 - w_vy1);
}

void   w_axis(x1, y1, x2, y2)
double  x1, y1, x2, y2;
{
    w_sx = x1;
    w_sy = y1;

    w_dx = (x2 - x1) / (w_vx2 - w_vx1);
    w_dy = (y2 - y1) / (w_vy2 - w_vy1);
}

void   w_attr(col, pat)
int    col;
word     pat;
{
    w_col = col;
    w_pat = pat;
}

void   w_attr22(col, pat)
int    col;
word     pat;
{
    w_col = col;
    w_pat = pat;
}

void   w_line(x, y, flag)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
double  x, y;
int     flag;
{
   int    ix, iy;
   static int  prex = 0;
   static int  prey = 0;


   ix = (int)(w_vx1 + (x - w_sx) / w_dx);
   iy = (int)(w_vy1 + (y - w_sy) / w_dy);

   if (flag) g_line(prex, prey, ix, iy, w_col, 0, 1,
w_pat);

   prex = ix;
   prey = iy;
}

void    w_line2(xxx, yyy, flag ,no)
double  xxx, yyy;
int     flag ,no;
{
   int    ix, iy;
   static int  prex[15];
   static int  prey[15];


   ix = (int)(/*w_vx1 +*/ (xxx - w_sx) / w_dx);
   iy = (int)(/*w_vy1 +*/ (yyy - w_sy) / w_dy);

  /*  if (fabs(yyy)< fabs(w_sy))   */


   if (flag) {g_line(prex[no], prey[no], ix, iy,
 w_col, 0, 1,w_pat);}


   prex[no] = ix;
   prey[no] = iy;
}

void    w_line222(xxx, yyy, flag ,no)
double  xxx, yyy;
int     flag ,no;
{
   int    ix, iy;
   static int  prex[15];
   static int  prey[15];


   ix = (int)(/*w_vx1 +*/ (xxx - w_sx) / w_dx);
   iy = (int)(/*w_vy1 +*/ (yyy - w_sy) / w_dy);

  /*  if (fabs(yyy)< fabs(w_sy))   */


   if (flag) {g_line(prex[no], prey[no], ix, iy,
 w_col, 0, 1,w_pat);}


   prex[no] = ix;
   prey[no] = iy;
}


void t_cls(void)
{
        clrscr();
}

void g_screen(int x1,int x2,int x3,int x4)
{
        int screen_x,screen_y;

        screen_x= getmaxx();
        screen_y= getmaxy();
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        if (!((screen_x>=600) &&
(screen_y>=400)))
                        {
                                outtextxy
(10,10,"Can not use with this monitor");
                                exit(0);
                        }/*========== test
monitor is 640 X 380 More */
        setcolor(x1);
        setbkcolor(x2);
        settextstyle(x3,x3,x4);
}
/*
main()
{
        g_init();
        g_line(10,10,100,100,7,0,0,w_pat);
        getch();
        w_view(100,10,500,110);
        w_axiss(0,4.5,.005*1201,-4.5);
        getch();
        g_screen(3,0,0,1);
        getch();
        g_cls();g_print(100,10," test");getch();
        closegraph();
}
*/
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก ง.

โปรแกรมการสุ่ม

RANDOM.C

เป็นโปรแกรมที่ใช้ในการสุ่มของโปรแกรมเจเนติกอัลกอริทึม

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

// The ran3 pseudo-random number generator.  It is
*not* linear congruential.

#define MBIG 1000000000
#define MSEED 161803398
#define MZ 0
#define FAC (1.0/MBIG)
void sran2(unsigned int seed);
float ran2();

static int inext,inextp;
static long ma[56];

void sran2(unsigned int seed)
{
 long idum = seed;
 long mj,mk;
 int i,ii,k;

 mj=MSEED-idum;
 mj %= MBIG;
 ma[55]=mj;
 mk=1;
 for (i=1;i<=54;i++) {
  ii=(21*i) % 55;
  ma[ii]=mk;
  mk=mj-mk;
  if (mk < MZ) mk += MBIG;
  mj=ma[ii];
 }
 for (k=1;k<=4;k++)
  for (i=1;i<=55;i++) {
   ma[i] -= ma[1+(i+30) % 55];
   if (ma[i] < MZ) ma[i] += MBIG;
  }
 inext=0;
 inextp=31;
}

float ran2()
{
 long mj;
 int i,ii,k;

 if (++inext == 56) inext=1;
 if (++inextp == 56) inextp=1;
 mj=ma[inext]-ma[inextp];
 if (mj < MZ) mj += MBIG;
 ma[inext]=mj;
 return mj*FAC;
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

โครงงานนี้ประสบความสำเร็จได้เพราะมีบุคคลต่างๆ ให้ความอนุเคราะห์ในการจัดทำโครงงาน ซึ่งคณะผู้จัดทำขอกราบขอบพระคุณเป็นอย่างมากไว้ ณ ที่นี้

คุณพ่อ คุณแม่ของพวกเราทุกคนที่ให้ความรักแก่เราทุกคน

อาจารย์มณฑล ลีลาจินดาไกรฤกษ์ อาจารย์ที่ปรึกษา ผู้ให้โอกาส ความรู้และคำแนะนำต่างๆ ที่เป็นประโยชน์อย่างยิ่งแก่คณะผู้จัดทำ

Prof.Yashibumi Mizutani ผู้จุดประกายการศึกษาความรู้ทางด้านเจเนติกอัลกอริทึม(GA)

คุณคมสันต์ หงษ์สมบัติ ผู้ช่วยเหลือในด้านความรู้ ให้คำปรึกษาและทุกๆ อย่างเกี่ยวกับเจเนติกอัลกอริทึม

อาจารย์กิตติ ไพฑูรย์วัฒนกิจ ผู้ช่วยเหลือในด้านแหล่งข้อมูลและความรู้

อาจารย์เชาว์ ชมพูอินไหว ผู้ช่วยเหลือในด้านความรู้ ให้คำปรึกษาและทุกๆ อย่าง

อาจารย์สมศักดิ์ มิตะถา ผู้ให้ความอนุเคราะห์ทางคอมพิวเตอร์ และคำแนะนำที่เป็นประโยชน์

คุณคงสิน กุลกลางดอน ช่วยเหลือเครื่องคอมพิวเตอร์ตลอดเวลาที่ทำโครงงาน

คุณวิทยากรณ์ สินธุ์ไสย ช่วยเหลือเครื่องคอมพิวเตอร์

คุณสันติ ศิริพันธ์ ช่วยเหลือเครื่องคอมพิวเตอร์ตลอดภาคเรียนที่ 1

คุณทรงชัย ศรีพล ช่วยเหลือเครื่องคอมพิวเตอร์

คุณณัฐวุฒิ อุตสาหภูมิ ช่วยเหลือเครื่องคอมพิวเตอร์

คุณไตรภูมิ มาสสะอาด ช่วยเหลือเครื่องพิมพ์ตลอดเวลาทำโครงงาน

และขอบคุณทุกคนที่อยู่เบื้องหลัง ที่คอยเป็นกำลังใจ สร้างความสนุกสนานและเสียงรบกวน คอยอุปการะอาหารการกิน ขอบคุณทุกๆ คน ด้วยความคารวะและจริงใจ

ขอบคุณเพื่อนรักผู้ทำงานร่วมกันอย่างขยันขันแข็ง

ขอบคุณโลกที่ยังสวยงามและสดใสเสมอ.....

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารอ้างอิง

[1] Y.Mizutani,M.leelajindakrairerk,T.Okabe,Y.Kinoshita," A Study on High Speed Genetic Algorithm for Power system Stabilizer Considering Stability Limit", Japan,1998.

[2] Y.Mizutani,M.Leelajindakrairerk,K.Yamasaki,S.Naniyo,T.Okabe, Y.Kinoshita,S.Matoba, M,Ishiseki,"Hybrid Type Power System Stability Control Method Based on Combination of Conventional PSS and Fuzzy comtrol Usimg PID Information",ICEE Trans,August 12-15,1996,Beijing,China.

[3] M.A.Iskandar , M.Satoh ,Y.Ohmori,S.Matoba,T.Okabe,Y.Mizutani",On Fuzzy Control Based Static Var Power System Stability Control" ,IEEE Trans ,PP.201-205 ,19-22 Aprol 1993.

[4] กาญจนี วงศ์วิภาพร ,การจัดตารางสอนของโรงเรียนอัตโนมัติโดยจีเนติกอัลกอริทึม",บัณฑิตวิทยาลัย,สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง,ISBN 974-622-126-4 ,พ.ศ. 2541.

[5] โตศักดิ์ ทัศนานุตริยะ,"การวิเคราะห์ระบบไฟฟ้ากำลัง",กรุงเทพฯ:ซีเอ็ดยูเคชัน,ISBN 974-512-869-4 ,2540.

[6] สุเธียร เกียรติสุนทร ,"พื้นฐานวิศวกรรมระบบควบคุมในกระบวนการอุตสาหกรรม เล่ม 1 ",สมาคมส่งเสริมเทคโนโลยี(ไทย-ญี่ปุ่น),ISBN 974-7970-36-8,2537.

[7] Katsuhiko Okata,"Modern Control Engineering : ,2nd ed.,Prentice Hall,International Edition,New Jersy,1990.

[8] คมสันต์ หงษ์สมบัติ ,มณฑล ลีลาจินดาไกรฤกษ์,"การปรับปรุงเสถียรภาพระบบไฟฟ้ากำลัง ด้วยตัวชดเชยกำลังไฟฟ้าแบบสถิตย์โดยใช้วิธีฟัซซี่", ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

[9] คมสันต์ หงษ์สมบัติ ,มณฑล ลีลาจินดาไกรฤกษ์,"เจเนติกอัลกอริทึมสำหรับช่วยออกแบบตัวควบคุมฟัซซี่ของตัวชดเชยกำลังไฟฟ้าแบบสถิตย์เพื่อปรับปรุงเสถียรภาพระบบไฟฟ้ากำลัง", คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

[10] คมสันต์ หงษ์สมบัติ ,มณฑล ลีลาจินดาไกรฤกษ์,"การปรับค่าพารามิเตอร์ควบคุมให้เหมาะสมสูงสุดของตัวควบคุม $\Delta\omega$-PSS และตัวควบคุมฟิซซี่ของ SVC สำหรับการควบคุมเสถียรภาพระบบไฟฟ้ากำลัง", ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

[11] DE.Goldberg,"Genetic Algorithm in Search,Optimization and Machine Learning",New York,Addison-Wesley,1989.

[12] L.Davis,"Handbook of Genetic Algorithms",Van Nostran Rienhold,1991.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้